

Vivado Design Suite 用户指南

设计分析与收敛技巧

UG906 (v2019.2) 2019 年 10 月 30 日

条款中英文版本如有歧义，概以英文版本为准。



修订历史

下表列出了本文档的修订历史。

章节	修订综述
2019 年 10 月 30 日 2019.2 版	
QoR 建议报告 (Report QoR Suggestions)	<ul style="list-style-type: none">重写章节新增“增量流程中的 RQS”
QoR 评估报告 (Report QoR Assessment)	重写章节以作澄清
创建设计相关报告	新增“RAM 使用情况报告”
2019 年 5 月 22 日 2019.1 版	
QoR 建议报告 (Report QoR Suggestions)	重写章节以作澄清。
读取和解释时序路径特性报告	移除有关更新“保持修复绕行 (Hold Fix Detour)”特性的内容。
验证设计方法 DRC	新增有关聚焦严重警告 (Critical Warning) 的注释。
从命令行创建豁免	在“通配符关键字”表中新增一行。
报告豁免	新增有关多比特规则的信息。
TIMING-10：同步器上缺失属性	<ul style="list-style-type: none">新增有关 TIMING-10 违例触发的信息。替换“同步器上缺失属性”图。

目录

修订历史.....	2
第 1 章：IDE 中的逻辑分析.....	6
IDE 中的设计分析.....	6
逻辑分析特性.....	6
使用“网表 (Netlist)”窗口.....	6
使用“层级 (Hierarchy)”窗口.....	8
使用“利用率 (Utilization)”报告.....	9
使用“原理图 (Schematic)”窗口.....	10
使用“查找 (Find)”对话框搜索对象.....	14
分析器件使用情况统计数据.....	17
使用 DRC 报告.....	17
验证设计方法 DRC.....	18
第 2 章：时序分析功能.....	20
时序汇总报告 (Report Timing Summary).....	20
时钟网络报告 (Report Clock Networks).....	34
时钟交互报告 (Report Clock Interaction).....	36
脉冲宽度报告 (Report Pulse Width).....	42
时序报告 (Report Timing).....	42
数据手册报告 (Report Datasheet).....	47
例外报告 (Report Exceptions).....	52
Vivado IDE 中的例外报告.....	59
时钟域交汇报告 (Report Clock Domain Crossings).....	65
总线偏差报告 (Report Bus Skew).....	84
第 3 章：实现结果分析功能.....	92
使用“设计运行 (Design Runs)”窗口.....	92
布局分析.....	93
布线分析.....	99
设计分析报告 (Report Design Analysis).....	104
QoR 评估报告 (Report QoR Assessment).....	123
QoR 建议报告 (Report QoR Suggestions).....	126
第 4 章：查看报告和消息.....	136
报告和消息简介.....	136
在 IDE 中查看和管理消息.....	136
Vivado 生成的消息.....	139
设计检查的生成与豁免.....	140

可配置的报告策略.....	153
创建设计相关报告.....	158
第 5 章：执行时序分析.....	186
时序分析简介.....	186
了解时序分析的基础知识.....	189
查看时序路径报告.....	198
时序验收的验证.....	205
第 6 章：综合分析与收敛技巧.....	206
使用细化视图对 RTL 进行最优化.....	206
分解深层内存配置，实现功耗与性能平衡.....	208
当内存深度不为 2 的幂时，优化 RAMB 利用率.....	211
最优化 RAMB 输入逻辑以允许输出寄存器推断.....	214
改进 RAMB 输出上的关键逻辑.....	217
第 7 章：实现分析与收敛技巧.....	222
使用 report_design_analysis 命令.....	222
识别设计中最长的逻辑延迟路径.....	225
识别高扇出信号线驱动.....	225
判断保持修复对设计是否存在负面影响.....	227
快速分析所有失败路径.....	228
布局规划.....	229
附录 A：时序方法检查.....	243
TIMING-1：时钟修改块上的时钟波形无效.....	243
TIMING-2：基准时钟源管脚无效.....	244
TIMING-3：时钟修改块上的基准时钟无效.....	245
TIMING-4：时钟树上的基准时钟重新定义无效.....	246
TIMING-5：时钟树上的波形重定义无效.....	247
TIMING-6：相关时钟间无公共基准时钟.....	248
TIMING-7：相关时钟间无公共节点.....	249
TIMING-8：相关时钟间无公共周期.....	250
TIMING-9：未知 CDC 逻辑.....	250
TIMING-10：同步器上缺失属性.....	251
TIMING-11：含“Datapath Only”选项的最大延迟不适用.....	252
TIMING-12：已禁用“时钟再收敛消极因素移除”.....	253
TIMING-13：因路径分段而忽略的时序路径.....	253
TIMING-14：时钟树上的 LUT.....	253
TIMING-15：时钟间路径上的保持时间严重违例.....	254
TIMING-16：建立时间严重违例.....	255
TIMING-17：未设置时钟的时序单元.....	255
TIMING-18：输入或输出延迟缺失.....	255
TIMING-19：ODDR 上的生成时钟波形反向.....	256
TIMING-20：未设置时钟的锁存器.....	256
TIMING-21：MMCM 上的 COMPENSATION 属性无效.....	256

TIMING-22: MMCM 上缺少外部延迟.....	257
TIMING-23: 发现组合循环.....	257
TIMING-24: 仅最大延迟数据路径已被覆盖.....	258
TIMING-25: 千兆位收发器 (GT) 上的时钟波形无效.....	258
TIMING-26: 千兆位收发器 (GT) 上时钟缺失.....	259
TIMING-27: 层级管脚上的基准时钟无效.....	259
TIMING-28: 时序约束引用的自动衍生时钟.....	259
TIMING-29: 多周期路径对不一致.....	260
TIMING-30: 生成时钟的欠优化主时钟源管脚选择.....	260
附录 B: QoR 建议报告 RTL 代码更改示例.....	261
TIMING-201: 为 RAM 添加输出寄存器.....	261
TIMING-202: 添加额外流水线以拓宽倍频器.....	264
UTIL-203: 使用分布式 RAM 推断的大型 ROM.....	267
UTIL-204: RAM 阵列未能有效使用.....	271
参考设计文件.....	275
附录 C: 附加资源与法律提示.....	276
赛灵思资源.....	276
解决方案中心.....	276
Documentation Navigator 与设计中心.....	276
参考资料.....	276
培训资料.....	277
请阅读: 重要法律提示.....	277

第 1 章

IDE 中的逻辑分析

IDE 中的设计分析

以下章节提供了赛灵思 Vivado® Design Suite 集成设计环境 (IDE) 中的设计分析简介：

- [第 1 章：IDE 中的逻辑分析](#) (本章节)
- [第 2 章：时序分析功能](#)
- [第 3 章：实现结果分析功能](#)

逻辑分析特性

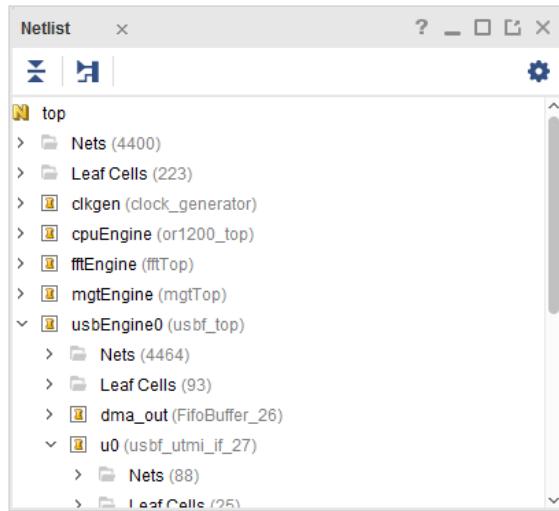
本章讨论逻辑分析特性，包括：

- 使用“网表 (Netlist)”窗口
- 使用“层级 (Hierarchy)”窗口
- 使用“原理图 (Schematic)”窗口
- 使用“查找 (Find)”对话框搜索对象
- 分析器件使用情况统计数据
- 使用 DRC 报告
- 验证设计方法 DRC

使用“网表 (Netlist)”窗口

“Netlist”窗口显示了网表中由综合工具所处理的设计层级。它对于浏览设计的逻辑层级很有用。

图 1：“Netlist”窗口



根据综合设置，网表层级与原始 RTL 可能 100% 匹配，也可能不存在层级。通常，默认情况下综合在对逻辑进行最优化时会保留大部分用户层级。由此将产生更小更快的网表。

通过使用综合工具默认设置，即可识别网表层级，但层级接口也可供修改。层级中可能缺少部分管脚和层次。

网表层级以文件夹树的形式展现。在每一层中，工具将显示如下内容：

- 该层次存在的任意信号线的“Nets”文件夹
- “Leaf Cells”文件夹，前提是该层次存在硬件原语实例
- 在该层次例化的任意层级的“hierarchy”文件夹

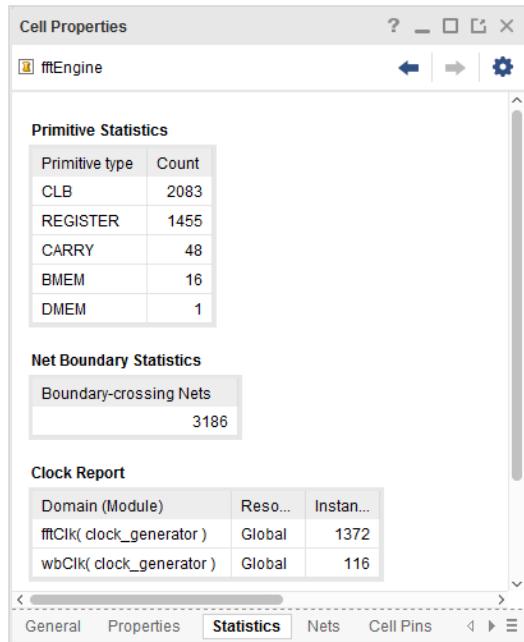
展开“hierarchy”文件夹即可显示该层次的“Nets”、“Leaf Cells”和“hierarchies”。单元旁的图标可显示有关设计状态的信息。

欲知详情，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的相应内容。

选定“hierarchy”的“单元属性 (Cell Properties)”窗口可提供实用信息，可通过窗口底部的分类按钮来筛选所显示的信息。选择“统计数据 (Statistics)”按钮可显示使用情况统计数据，包括：

- 整个层级分支的原语使用情况，按更高层次存储桶加以分组。
- 跨层级边界的信号线数量
- 层级中所用的时钟

图 2：“Cell Properties” 窗口



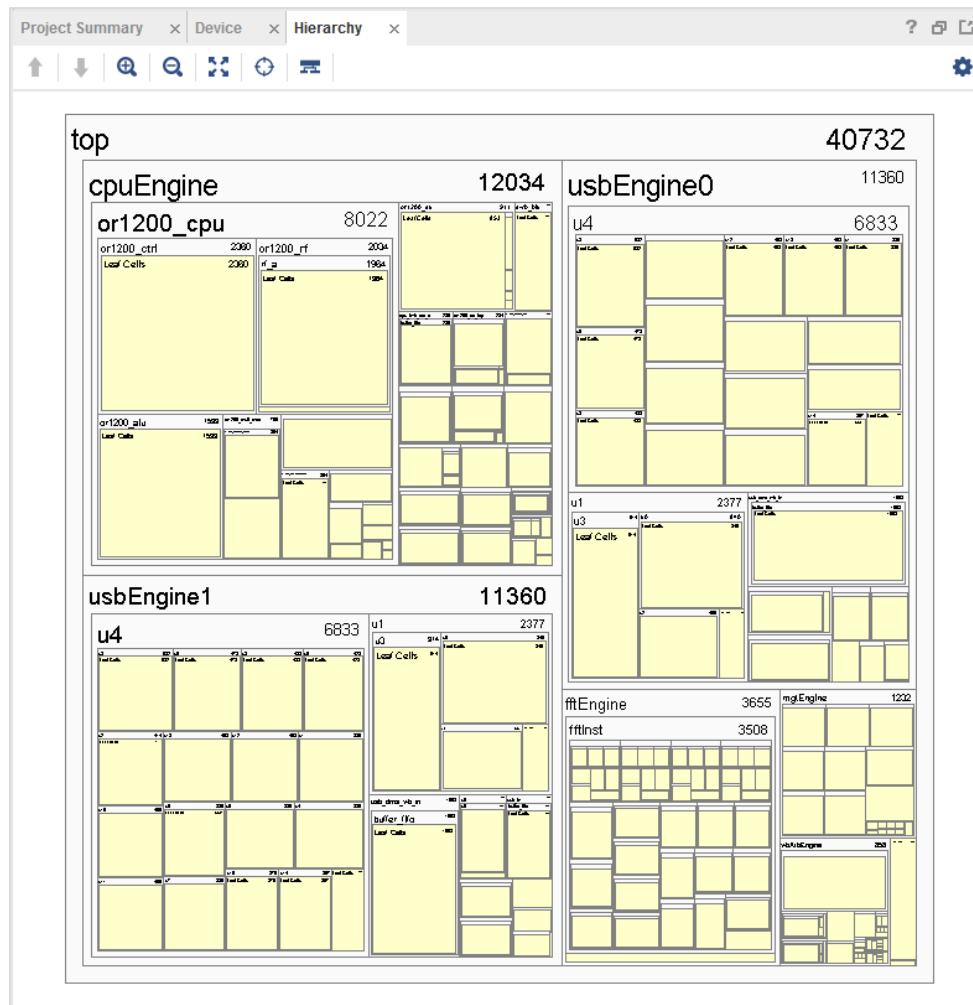
如对设计进行布局规划，那么针对 Pblock 将显示类似的属性。

使用“层级 (Hierarchy)”窗口

此窗口用于以物理方式浏览层级以了解资源使用情况。要打开“Hierarchy”窗口，请选择“Tools > Show Hierarchy”，或者从“网表 (Netlist)”窗口中，单击“F6”。

如下图所示，“Hierarchy”窗口中显示了选定层级的层级映射。层级映射将叶节点单元显示为黄色块，并嵌套在对应其父层级的矩形中。层级中每个层次的大小均根据该层次上的实例数量与设计中实例总数的占比关系来设置。

图 3：“Hierarchy”窗口



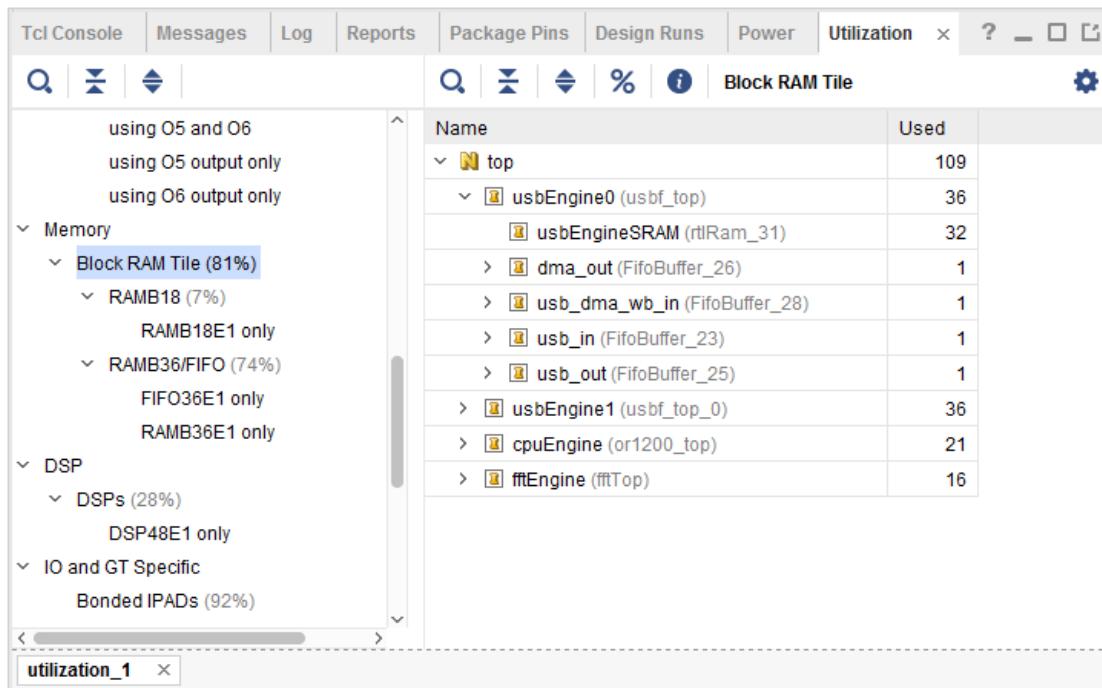
上图显示 `cpuEngine`、`usbEngine0` 和 `usbEngine1` 包含设计中的大部分逻辑，并且使用的资源数量都几乎相同。

使用“利用率 (Utilization)”报告

“Utilization Report”可根据资源类型细分设计利用率。左侧面板提供按资源类型汇总的利用率，右侧面板按层级显示利用率。

要查看“Utilization Report”，请选择“Reports”→“Report Utilization”。下图显示了“Utilization Report”。

图 4：“利用率 (Utilization)” 报告



在此设计中，RAMB36 和 FIFO36 块主要供 2 个 `usbEngine` 块使用。单击“加号 (+)”图标以查看子层级的使用情况。

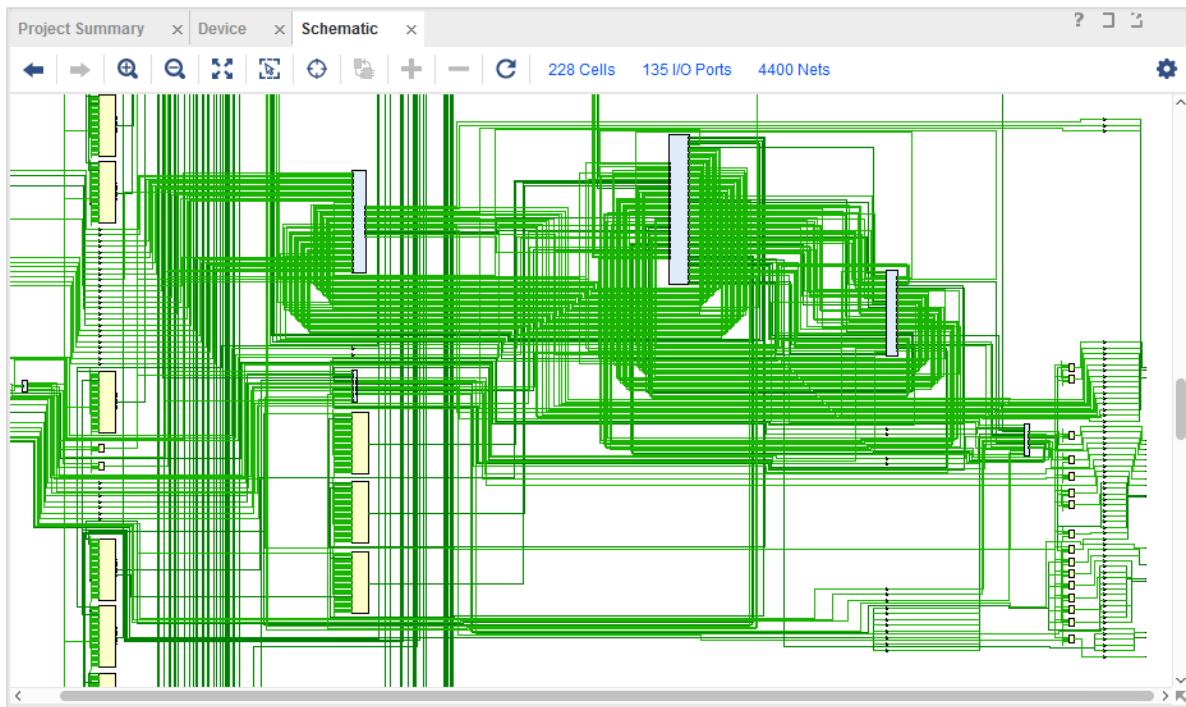
使用“原理图 (Schematic)”窗口

原理图 (Schematic) 是网表的一种图形化表示法。通过查看原理图即可：

- 查看网表的图形化表示。
- 检查门电路、层级以及连接。
- 追踪并扩展逻辑推。
- 分析设计。
- 更准确了解设计内部发生的状况。

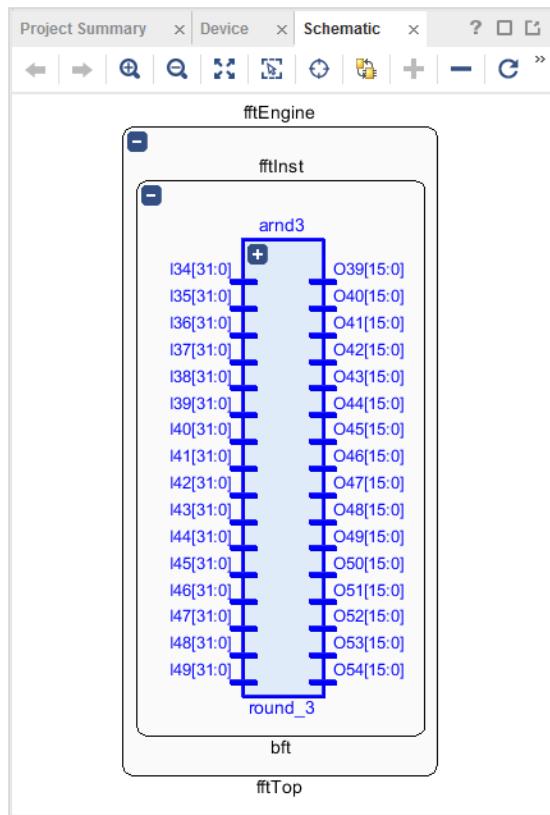
在“Elaborated Design”中的 RTL 层级上可查看工具对代码的解释。在“Synthesize Design”和“Implemented Design”中，可查看由综合工具生成的门电路。要打开原理图，请选择“Tools > Schematic”。如果不选择任何项，门电路、层级和连接会显示在设计顶层，如下图所示。

图 5：顶层原理图



提示：如果仅使用单层级，则原理图更为简单。在原理图中以蓝色高亮显示选定元素。其中将显示对应单层级的端口。

图 6：含选定单层级的原理图



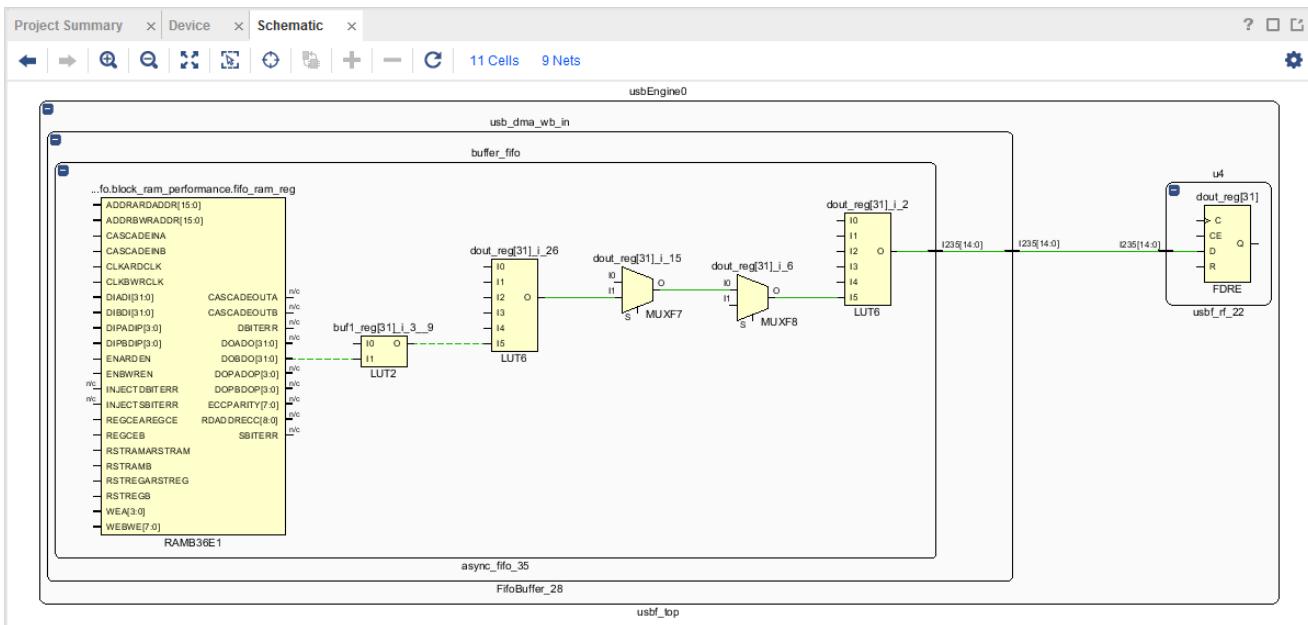
您可通过多种方式来追踪原理图。

- 单击左上角的“+”（加号）图标，以显示层级中的门电路。
- 双击端口或元素以将其展开。
- 右键单击并从弹出菜单中选择“Schematic”。
- 单击“<- ->”导航箭头以在前后原理图视图之间进行切换。

如需了解有关原理图的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的相应内容。

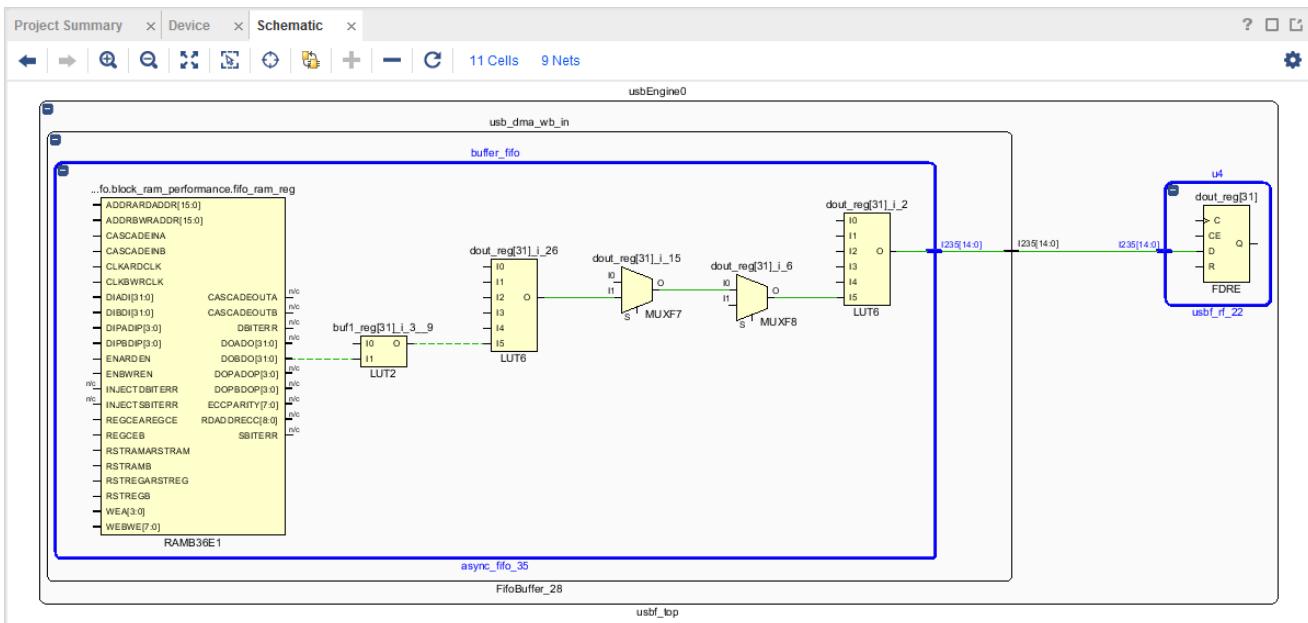
完成实现后，原理图是将时序路径中的门电路可视化的最简单方法。选择路径，然后即可打开原理图，其中包含来自该路径的门电路和信号线。

图7：含时序路径的原理图



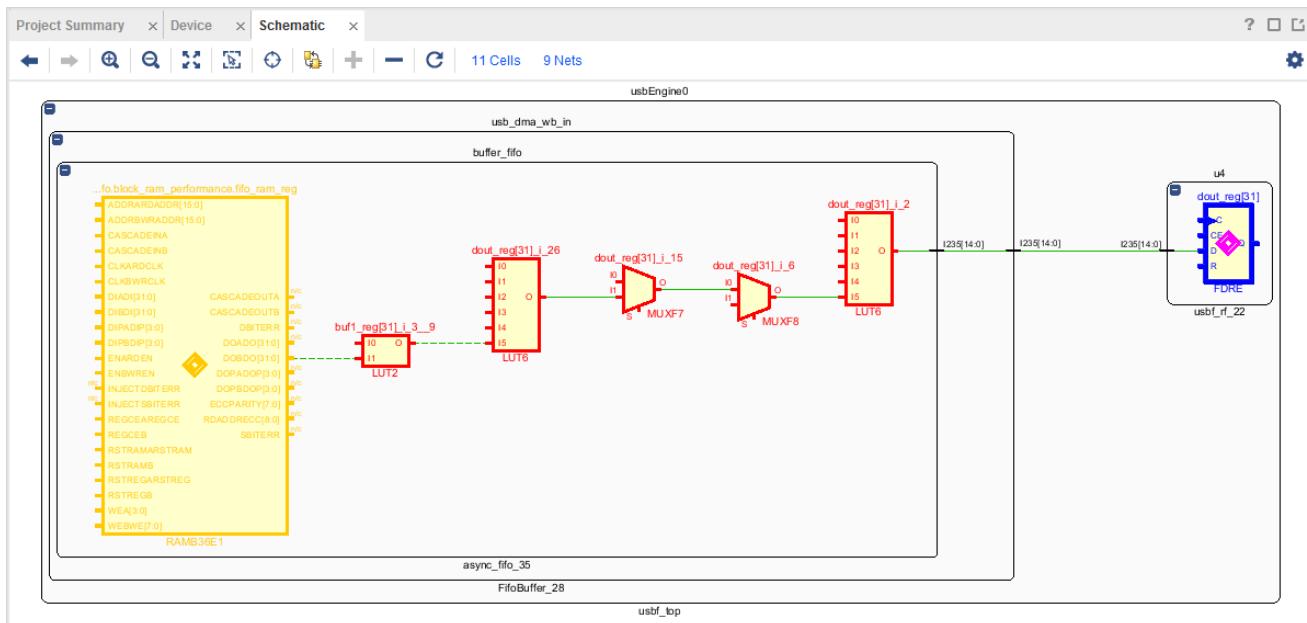
请从弹出菜单中选择“Select Leaf Cell Parents”以识别原理图中选定单元的层级的相关层次。

图8：含选定父级原语的时序路径



查看原理图时，请选择“Highlight”和“Mark”命令以跟踪感兴趣的叶节点单元。为单元添加颜色编码（使用标记或高亮）以便于追踪来自原始路径的逻辑和添加的逻辑。

图 9：含时序路径标记的原理图

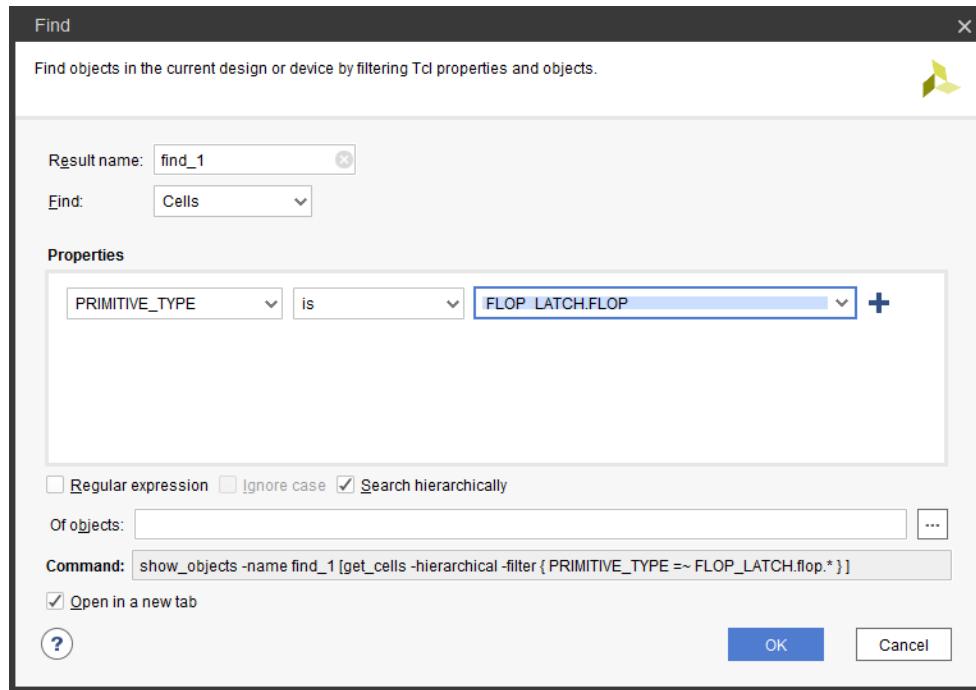


使用“查找 (Find)”对话框搜索对象

Vivado® IDE 包含强大的查找和搜索功能。要打开“Find”对话框，请选择“Edit” → “Find”。（请参阅下图。）

注释：您还可按以下键组合来打开“Find”窗口 “Ctrl+F。”

图 10：“Find”对话框



查找 (Find) 条件

“Find”对话框允许您根据各种条件和属性来搜索网表，如下图所示。

图 11：显示搜索条件的“Find”对话框

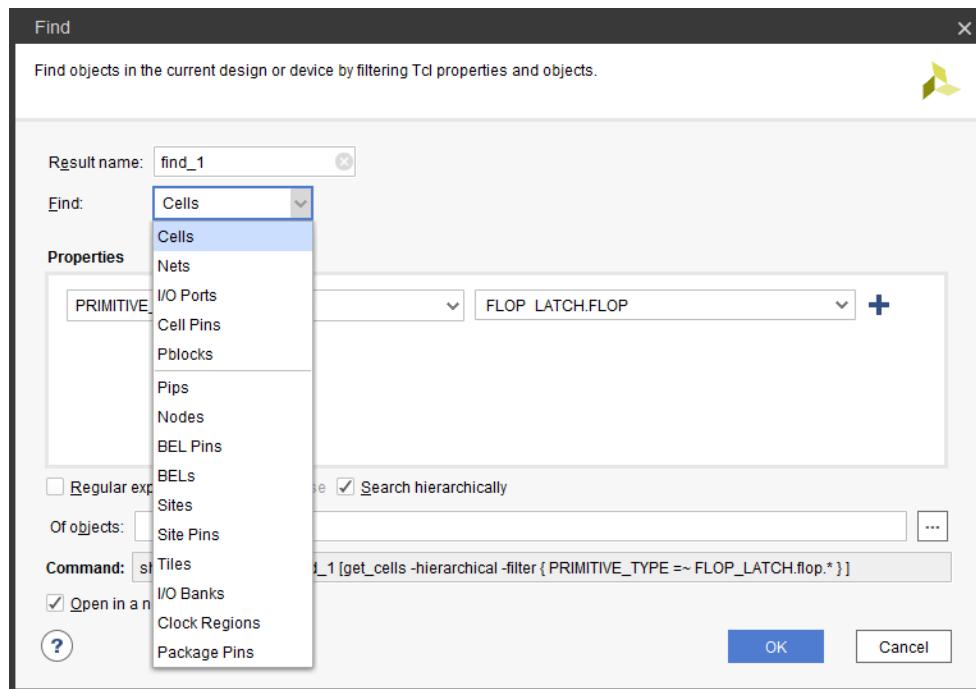
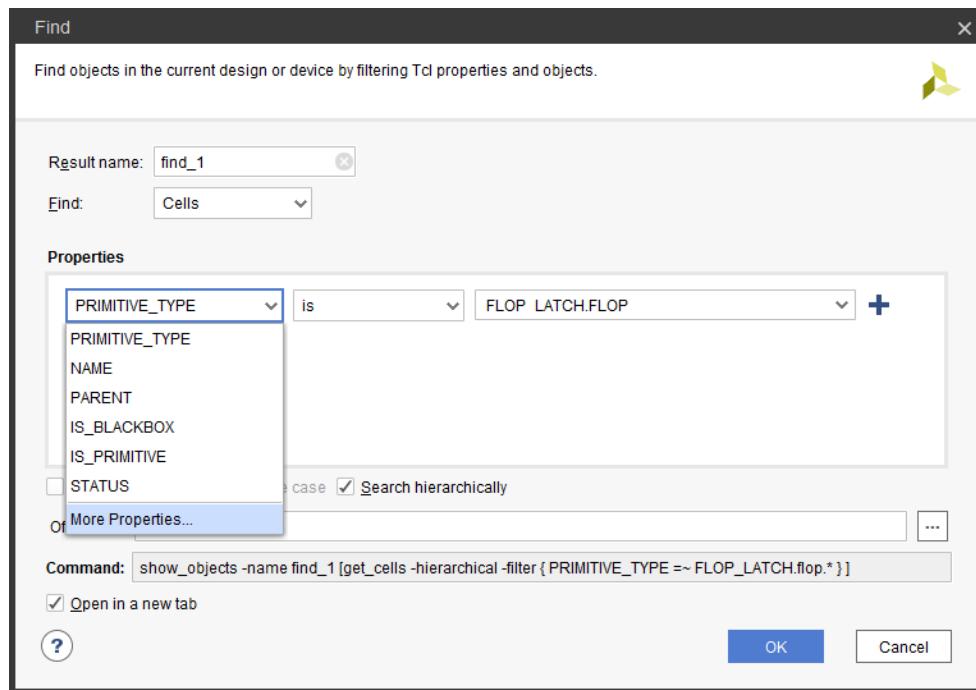


图 12：显示“属性 (Properties)”选项的“Find”对话框



复杂查找

要运行复杂查找，请执行以下操作：

1. 设置首要搜索条件。
2. 单击“属性 (Properties)”下拉选项旁的“+”（加号）。
3. 添加其它搜索条件。
4. 将其它条件与逻辑运算符 (AND、OR) 配合一起使用

Find 示例

选择“Edit” → “Find”以执行查找，例如：

- 所有未布局 I/O：
“Find:” <Cells>, “Properties:” <Primitive> <is> <IO> + <AND> <STATUS> <is> <UNPLACED>
- 删除超出 10,000 的信号线：
“Find:” <Nets>, “Properties:” <FLAT_PIN_COUNT> <is greater than> <10000>
- 使用 PREG 嵌入式寄存器的所有 DSP：
“Find:” <Cells>, “Properties:” <PRIMITIVE_TYPE> <is> <ARITHMETIC.DSP> + <AND> <PREG> <is greater than> <0>

Tcl 查找

在脚本或 Tcl 控制台中，使用等效的 Tcl `get_*` 命令（例如，`get_cells`）来查询 Vivado 对象。



提示：位于 Vivado® IDE 底部的 Tcl 控制台 (Tcl console) 可显示 GUI 中执行的每项操作的 Vivado Design Suite Tcl 命令运行过程。在 Tcl 控制台中，还可输入 Vivado Design Suite Tcl 命令。

如需了解有关 Tcl 脚本编制的更多信息，请参阅《Vivado Design Suite 用户指南：使用 Tcl 脚本》([UG894](#))。

如需了解有关 Tcl 命令的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#))，或输入 `<command> -help`。

分析器件使用情况统计数据

实现问题常源于未考量显式或隐式物理约束。例如，管脚输出 (pinout) 在逻辑布局上变为显式物理约束。slice 逻辑在大部分器件中都是一致的。但是，如下专用资源表示隐式物理约束，因为这些资源仅在某些位置可用，并且会影响逻辑布局：

- I/O
- 高性能 bank
- 高量程 bank
- MGT
- DSP slice
- 块 RAM
- MMCM
- BUFG
- BUFR

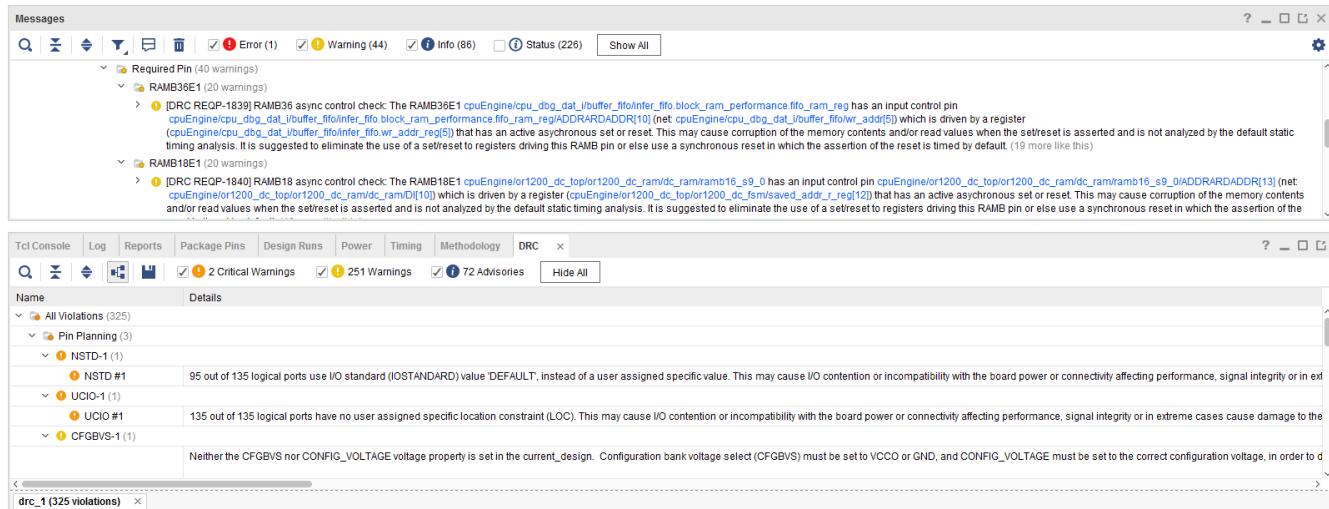
大量耗用此类专用资源的块必须围绕器件分散排列。在规划与设计其余部分的连接时，请考量这对于布局布线所产生的物理性约束。此外，Pblock 为显式物理约束，用于定义可支持的指定逻辑布局区域。通过搭配使用以下方法来分析器件上的块资源使用情况：

- `report_utilization`
- 网表属性
- Pblock 属性

使用 DRC 报告

设计规则检查 (DRC) 将检查设计并报告常见问题。从 2016.1 版起，DRC 分为两条不同命令。Methodology DRC 已迁移至 `report_methodology` 命令，所有其它 DRC 均包含在 `report_drc` 命令中。`report_drc` 命令用于运行非 Methodology DRC。执行实现期间，工具也会运行 DRC。DRC 随布局布线不断完善。

图 13：显示严重警告和错误



在流程初通过检查 DRC 消息、“严重警告 (Critical Warnings)” 和 “警告 (Warnings)” 可防止后续出现问题。

设计早期阶段中的“严重警告”在后续实现流程中会转变为“错误”并阻碍比特流的创建。在上述“综合后设计”生成的示例中，可选“DRC 报告”步骤会在报告中将未约束的 I/O 列为“严重警告”。布线后设计 DRC 报告也会报告此类“严重警告”。您必须复查此报告，因为在 `write_bitstream` 阶段此 DRC 将升级为“错误 (Error)”。请尽早审查 DRC 报告以判断设计中哪些领域需要修改。

验证设计方法 DRC

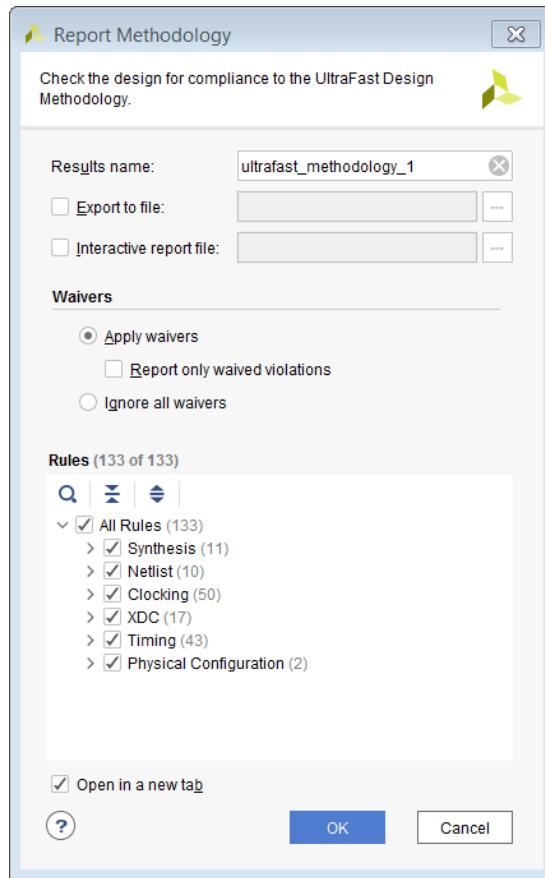
由于方法的重要性，Vivado® 工具提供了 `report_methodology` 命令，专用于检查方法 DRC 的合规性。DRC 类型因设计进程阶段而异。RTL lint 样式的检查在细化的 RTL 设计上运行；基于网表的逻辑和约束检查在综合后设计上运行；而实现和时序检查则在实现后设计上运行。

要在 Tcl 命令提示符中运行这些检查，请打开待验证的设计，并输入以下 Tcl 命令：

```
report_methodology
```

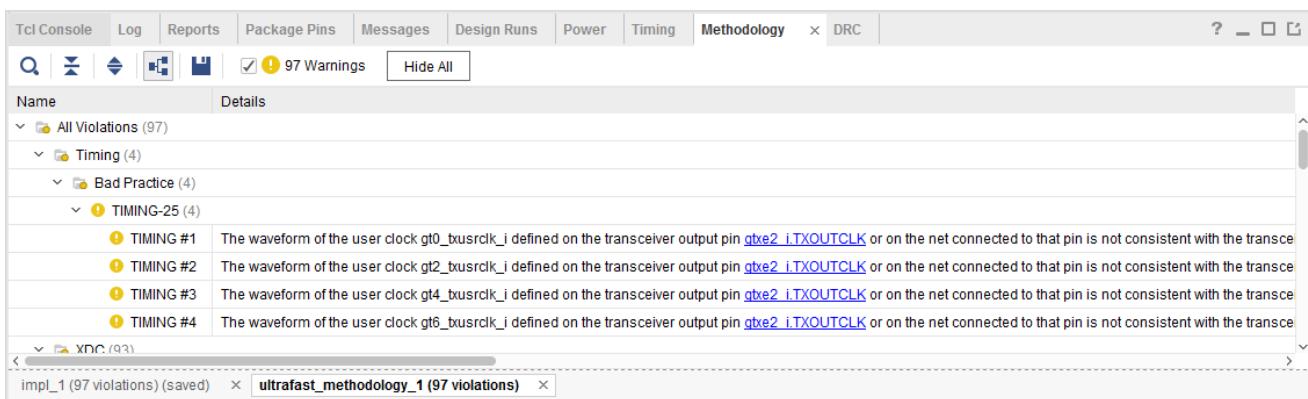
要从 IDE 运行这些检查，请打开待验证的设计，从 Flow Navigator 工程模式下运行“Report Methodology”命令或者从“Reports”→“Report Methodology”运行该命令。这样将显示如下图所示对话框。

图 14：“Report Methodology”对话框



在“Methodology”窗口中将列出违例（如有），如下图所示。

图 15：DRC 违例



如需了解有关运行设计方法 DRC 的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：系统级设计输入》(UG895) 中的相应内容。

注释：建议解决所有方法违例，特别是影响时序收敛和验收质量的严重警告。

第 2 章

时序分析功能

时序汇总报告 (Report Timing Summary)

综合后即可在流程中随时执行时序分析。您可复查由综合和实现运行自动创建的“时序汇总 (Timing Summary)”报告文件。

如果在内存中已加载综合后设计或实现后设计，那么还可通过以下方式生成交互式“Timing Summary”报告：

- “Flow Navigator” → “Synthesis”
- “Flow Navigator” → “Implementation”
- “Reports” → “Timing” → “Report Timing Summary”

等效的 Tcl 命令：`report_timing_summary`

如需了解有关 `report_timing_summary` 选项的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的相应内容。

在综合后设计中，Vivado® IDE 时序引擎会基于连接和扇出来估算信号线延迟。对于已由用户布局的单元之间的信号线，延迟准确性更高。在包含部分预布局单元（例如，I/O 和 GT）的路径上，时钟偏差可能更大。

在实现后设计中，基于实际布线信息来估算信号线延迟。对于已完全布线的设计，必须使用“Timing Summary”报告来实现时序验收。要验证设计是否已完全布线，请复查“布线状态 (Route Status)”报告。

从 Tcl 控制台或从 GUI 运行此报告时，可使用 `-cells` 选项将其限定于 1 个或多个层级单元。限定报告范围后，将仅报告含如下数据路径部分的路径：数据路径开始或结束于此类单元、与此类单元交汇或者完全包含于此类单元内。

“Report Timing Summary” 对话框

在 Vivado IDE 中，“Report Timing Summary” 对话框包含以下选项卡：

- “选项 (Options)” 选项卡
- “高级 (Advanced)” 选项卡
- “定时器设置 (Timer Settings)” 选项卡

位于“Report Timing Summary”对话框顶部的“Results name”字段用于指定在“Results”窗口中打开的图形化报告的名称。图形化版本的报告包含超链接，支持您将来自报告的信号线和单元交叉引用至“Device”和“Schematic”窗口以及设计源文件。

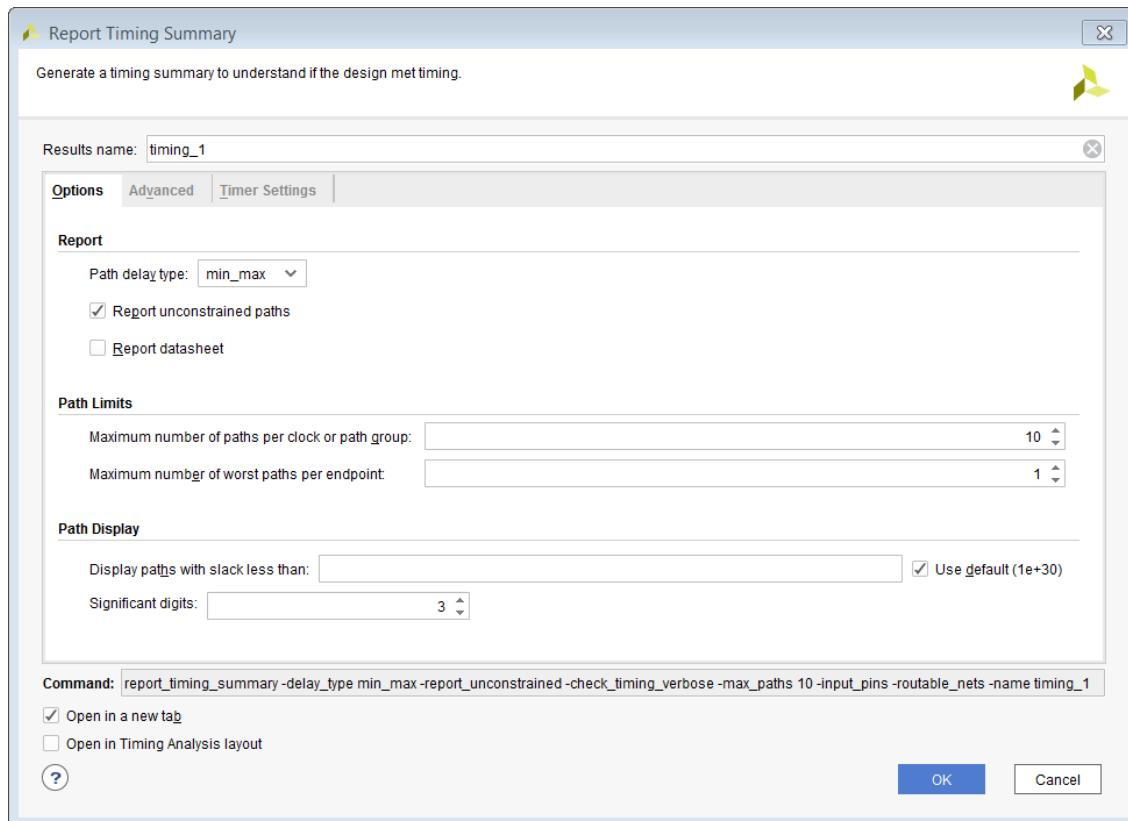
如果该字段留空，那么报告将返回 Tcl 控制台，且在“Results”窗口中不会打开图形化版本的报告。

等效的 Tcl 选项：`-name`

“选项 (Options)” 选项卡

“Report Timing Summary” 对话框中的“Options”选项卡如下图所示。

图 16：“Report Timing Summary” 对话框：“Options” 选项卡



“报告 (Report)” 部分

“Report Timing Summary” 对话框的“Options”选项卡的“Report”部分包含以下内容：

- 路径延迟类型 (Path delay type)

用于设置要运行的分析类型。对于已综合的设计，默认情况下仅执行最大延迟分析（建立/恢复）。对于已实现的设计，默认情况下，将执行最小和最大延迟分析（建立/保持和恢复/移除）。要仅运行最小延迟分析（保持和移除），请选择延迟类型 min。

等效的 Tcl 选项：-delay_type

- 报告未约束路径 (Report unconstrained paths)

生成不含时序要求的路径的相关信息。默认情况下，在 Vivado IDE 中已选中该选项，但在等效的 Tcl 命令 report_timing_summary 中默认不开启该选项。

等效的 Tcl 选项：-report_unconstrained

- 报告数据手册 (Report datasheet)

生成本章中的[数据手册报告 \(Report Datasheet\)](#)中所定义的设计数据手册。

等效的 Tcl 选项：-datasheet

“路径限制 (Path Limits)” 部分

“Report Timing Summary” 对话框中 “Options” 选项卡的 “Path Limits” 部分包括：

- 每个时钟或路径组的最大路径数 (Maximum number of paths per clock or path group): 控制每个时钟对或每个路径组所报告的最大路径数。

等效的 Tcl 选项: `-max_paths`

- 每个端点的最差路径的最大数量 (Maximum number of worst paths per endpoint): 控制每个路径端点可能报告的最大路径数。此限制受到每个时钟对或路径组的最大数量的限制。因此，报告的路径总数仍受到 `-max_paths` 数量的限制。

等效的 Tcl 选项: `-nworst`

“路径显示 (Path Display)” 部分

“Report Timing Summary” 对话框的 “Options” 选项卡的 “Path Display” 部分包括：

- 显示裕量小于该值的路径 (Display paths with slack less than): 基于路径的裕量值筛选报告的路径。此选项不影响汇总表的内容。

等效的 Tcl 选项: `-slack_lesser_than`

- 有效位数 (Significant digits): 控制报告中显示的数值的精确度。

等效的 Tcl 选项: `-significant_digits`

通用部分

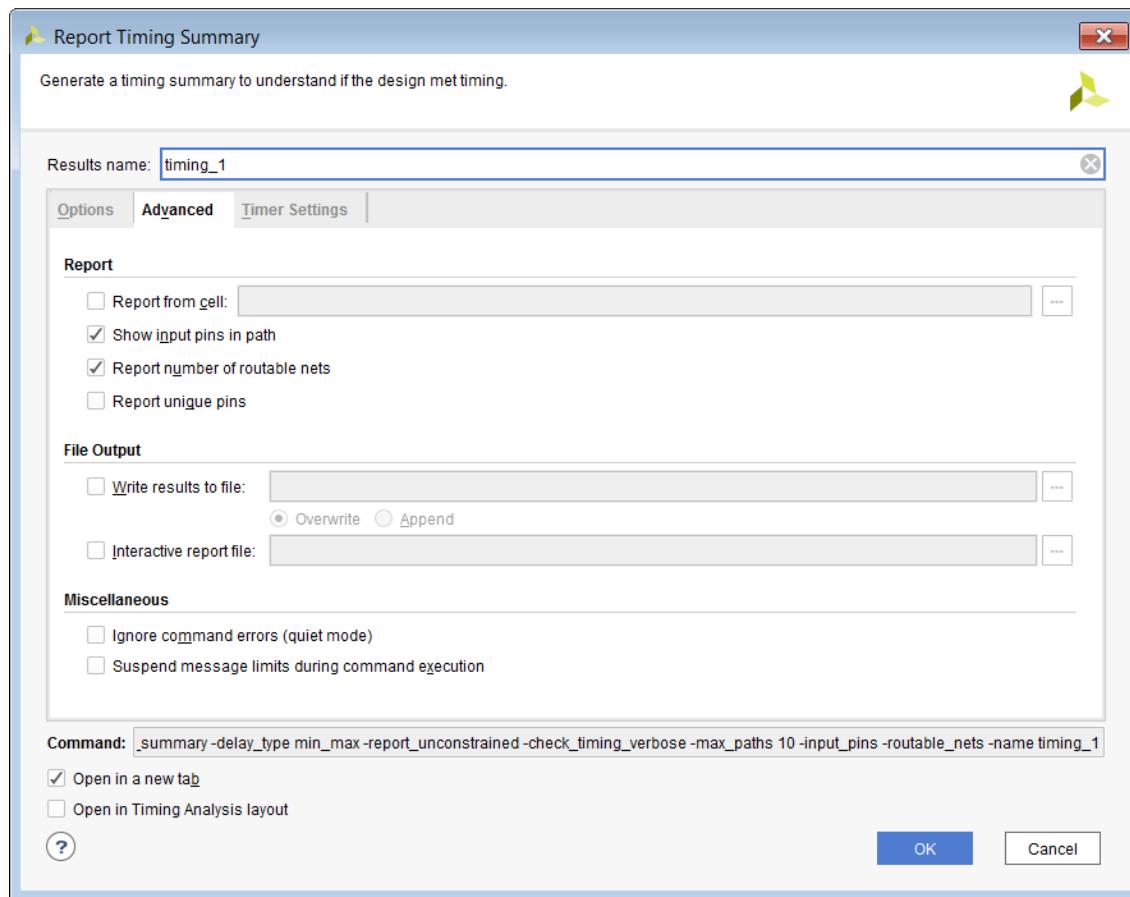
以下控件为位于 “Report Timing Summary” 对话框底部的全部 3 个选项卡通用的控件。

- 命令 (Command): 显示等效于 “Report Timing Summary” 对话框中指定的各种选项的 Tcl 命令行。
- 在新选项卡中打开 (Open in a New Tab): 在新选项卡中打开结果，或替换 “Results” 窗口中打开的最后一个选项卡。
- 在时序分析布局中打开 (Open in Timing Analysis layout): 将当前视图布局复位为 “Timing Analysis” 视图布局。
如需了解有关视图布局的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的相应内容。

“高级 (Advanced)” 选项卡

“Report Timing Summary” 对话框中的 “Advanced” 选项卡如下图所示。

图 17：“Report Timing Summary”对话框：“Advanced”选项卡



“报告 (Report)” 部分

- 从单元报告 (Report from cell): 启用该选项即可将时序报告限制在设计的特定单元上。报告将仅包含数据路径部分始于指定单元、止于该单元、跨该单元或完全包含于该单元内的路径。
等效的 Tcl 选项: `-cells`
- 显示路径中的输入管脚 (Show input pins in path): 显示单元中用于路径的输入管脚。
等效的 Tcl 选项: `-input_pins`

 建议: 保持该选项处于选中状态可提供有关路径中使用的所有管脚的更多信息。

- 报告唯一管脚 (Report unique Pins): 针对每一组唯一的管脚仅显示 1 条时序路径。
等效的 Tcl 选项: `-unique_pins`

“文件输出 (File Output)” 部分

- 将结果写入文件 (Write results to file): 将结果写入指定文件名。默认情况下，报告将写入 Vivado IDE 的“时序 (Timing)”窗口。
等效的 Tcl 选项: `-file`

- “覆盖 (Overwrite)” 或 “追加 (Append)”: 当报告写入文件时，这 2 个选项可用于确定 (1) 覆盖指定文件，还是 (2) 向现有报告追加新信息。

等效的 Tcl 选项: `-append`

- 交互式报告文件 (Interactive report file): 将结果以赛灵思 RPX 格式写入指定的文件名。RPX 文件属交互式报告，其中包含所有报告信息，可在 Vivado Design Suite 中使用 `open_report` 命令将其重新加载到内存中。

“杂项 (Miscellaneous)” 部分

- 忽略命令错误 (Ignore command errors): 以静默方式执行命令，忽略所有命令行错误，不返回任何消息。此命令还会返回 `TCL_OK`，忽略执行期间遇到的所有错误。

等效的 Tcl 选项: `-quiet`

- 命令执行期间暂挂消息限制 (Suspend message limits during command execution): 临时覆盖所有消息限制并返回所有消息。

等效的 Tcl 选项: `-verbose`

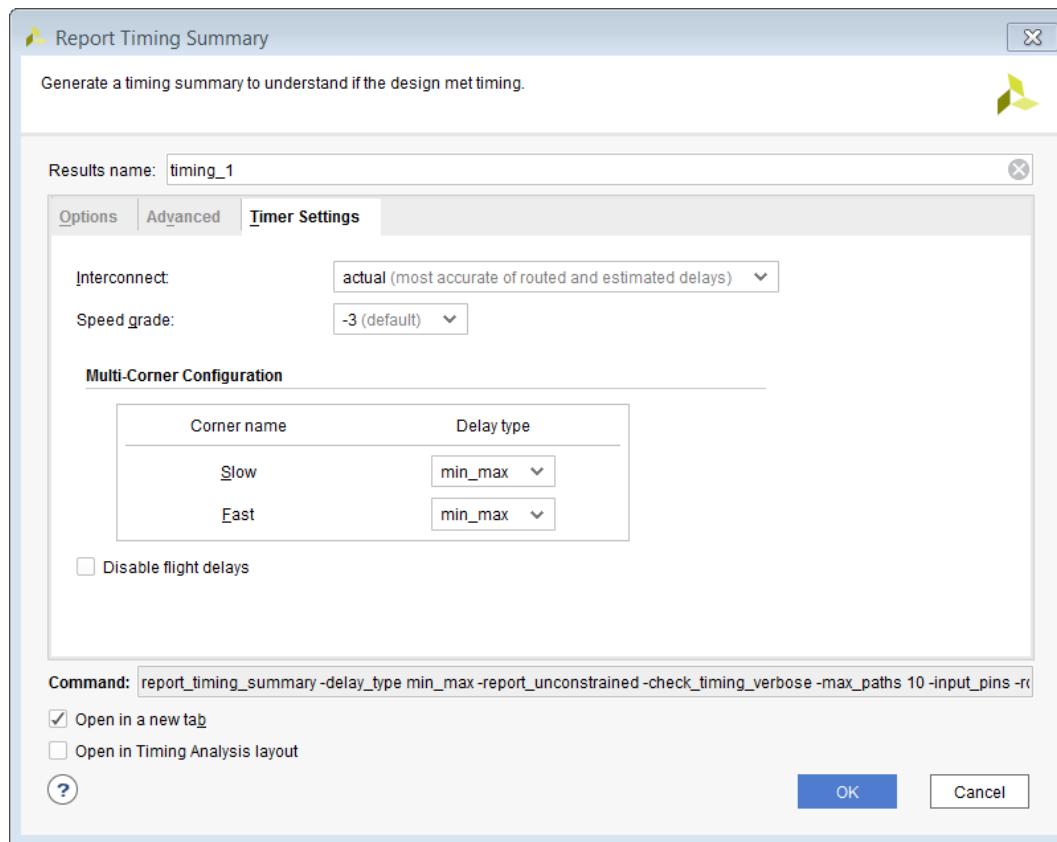
“定时器设置 (Timer Settings)” 选项卡

如需执行定时器设置，请使用如下任一方法：(1) 任一 Vivado IDE 时序分析对话框；或者，(2) 本节中列出的任一 Tcl 命令。这些设置会影响同一 Vivado IDE 会话内运行的时序相关命令，但综合和实现命令除外。

定时器设置不保存为工具首选项。每个新会话都会恢复默认值。请勿更改默认值。保留默认值将以最准确的延迟值来提供最大的时序分析覆盖范围。

“时序汇总报告 (Report Timing Summary)” 对话框中的“定时器设置 (Timer Settings)” 选项卡如下图所示。

图 18：“Report Timing Summary”对话框：“Timer Settings”选项卡



“Interconnect Setting”

控制信号线延迟计算方式：根据估算的叶节点单元管脚间布线距离来计算，还是根据实际布线的信号线来计算，或者从时序分析中排除信号线延迟。对于综合后设计，该选项自动设置为“Estimated”，对于实现后设计，该选项自动设置为“Actual”。

- 估算 (Estimated): 对于未布局的单元，信号线延迟值对应于可能实现的最佳布局的延迟，基于驱动和负载的性质以及扇出来计算。在时序路径报告中，未布局的叶节点单元管脚之间的信号线标记为未布局 (unplaced)。
- 对于已布局的单元，信号线延迟取决于驱动和负载之间的距离以及扇出。此信号线在时序路径报告中标记为 estimated。
- 实际 (Actual): 对于已布线的信号线，信号线延迟对应于已布线的互连的实际硬件延迟。此信号线在时序路径报告中标记为 routed。
- 无 (None): 在时序报告中不考虑互连延迟，信号线延迟强制为 0。

等效的 Tcl 命令： set_delay_model

“速度等级 (Speed Grade)” 设置

设置器件速度等级。默认情况下，根据创建工程时或打开设计检查点时所选的部分来设置该选项。通过更改该选项即可在同一个设计数据库上按不同速度等级报告时序，而无需重新运行整个实现流程。

等效的 Tcl 命令： set_speed_grade

“多角配置 (Multi-Corner Configuration)” 设置

指定要针对指定时序角点分析的路径延迟类型。有效值包括 `none`、`max`、`min` 和 `min_max`。选择 `none` 以禁用针对指定角点的时序分析。

 **建议：**针对两个角点均保留选中的建立时间（最大值）和保持时间（最小值）分析。

等效的 Tcl 命令：`config_timing_corners`

禁用飞行延迟

请勿将封装延迟添加到 I/O 延迟计算中。

等效的 Tcl 命令：`config_timing_analysis`

“时序汇总 (Timing Summary)” 报告详情

“Timing Summary” 报告包含下列部分：

- “常规信息 (General Information)” 部分
- “定时器设置 (Timer Settings)” 部分
- “设计时序汇总 (Design Timing Summary)” 部分
- “时钟汇总 (Clock Summary)” 部分
- “检查时序 (Check Timing)” 部分
- “时钟内部路径 (Intra-Clock Paths)” 部分
- “时钟间路径 (Inter-Clock Paths)” 部分
- “其它路径组 (Other Path Groups)” 部分
- “用户忽略的路径 (User-Ignored Paths)” 部分
- “未约束的路径 (Unconstrained Paths)” 部分

“Timing Summary” 报告中包含的综合信息类似于 Vivado IDE 中的各项报告（Report Clock Interaction、Report Pulse Width、Report Timing 和 Check Timing）所提供的信息，以及仅限 Tcl 中可用的部分报告 (`report_clocks`) 所提供的信息。但 “Report Timing Summary” 还包含此报告特有的信息，例如，“未约束路径 (Unconstrained Paths)”。

“常规信息 (General Information)” 部分

“Timing Summary” 报告的 “General Information” 部分可提供如下内容的相关信息：

- 设计名称
- 所选器件、封装和速度等级（带有速度文件版本）
- Vivado Design Suite 版本
- 当前日期
- 为生成报告所执行的等效 Tcl 命令

“定时器设置 (Timer Settings)” 部分

“Timing Summary” 报告的 “Timer Settings” 部分包含有关 Vivado IDE 时序分析引擎设置的详细信息，这些设置用于在报告中生成时序信息。下图通过示例显示了 “Timer Settings” 部分的默认选项，包括：

- 启用多角分析 (Enable Multi-Corner Analysis): 针对每个角点启用此分析，即多角配置 (Multi-Corner Configuration)。
 - 启用消极因素移除 (Enable Pessimism Removal) 和消极因素移除解决办法 (Pessimism Removal Resolution): 确保每条路径的源时钟和目标时钟的报告中均显示其公共节点无偏差。
- 注释:** 此设置必须始终启用。
- 启用输入延迟默认时钟 (Enable Input Delay Default Clock): 在无用户约束的输入端口上创建默认空输入延迟约束。默认禁用该选项。
 - 启用预置/清除 Arc (Enable Preset / Clear Arcs): 启用通过异步管脚进行时序路径传输。默认禁用该选项，它不影响恢复/移除检查。
 - 禁用飞行延迟 (Disable Flight Delays): 为 I/O 延迟计算禁用封装延迟。

图 19: “Timing Summary” 报告: “Timer Settings” 部分

Timer Settings				
Settings		Multi-Corner Configuration		
		Corner Name	Analyze Max Paths	Analyze Min Paths
Enable Multi Corner Analysis:	Yes	Slow	Yes	Yes
Enable Pessimism Removal:	Yes	Fast	Yes	Yes
Pessimism Removal Resolution:	Nearest Common Node			
Enable Input Delay Default Clock:	No			
Enable Preset / Clear Arcs:	No			
Disable Flight Delays:	No			
Ignore I/O Paths:	No			
Timing Early Launch at Borrowing Latches:	false			

要获取有关默认定时器设置以及如何对其进行更改的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的 config_timing_analysis。

“设计时序汇总 (Design Timing Summary)” 部分

“Timing Summary” 报告的 “Design Timing Summary” 部分（如下图所示）可提供设计时序汇总信息，并将所有其它部分的结果组合到单一视图内。

 **建议:** 请复查 “Design Timing Summary” 部分以验证布线后是否已满足所有时序约束，或者了解流程中任意时间点的设计状态。

图20：设计时序汇总 (Design Timing Summary)

Design Timing Summary			
Setup	Hold	Pulse Width	
Worst Negative Slack (WNS): 0.066 ns	Worst Hold Slack (WHS): 0.028 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints:	0
Total Number of Endpoints: 46285	Total Number of Endpoints: 46285	Total Number of Endpoints:	15989

All user specified timing constraints are met.

“Design Timing Summary”部分包含以下内容：

- “Setup”区域（最大延迟分析）
- “Hold”区域（最小延迟分析）
- “Pulse Width”区域（管脚开关限制）

“Setup”区域（最大延迟分析）

“Design Timing Summary”部分的“Setup”区域用于显示最大延迟分析相关的所有检查：建立、恢复和数据检查。

- Worst Negative Slack (WNS)：该值对应于最大延迟分析的所有时序路径的最差裕量。该值可为正值或负值。
- Total Negative Slack (TNS)：所有 WNS 违例的总和，前提是仅考量每个时序路径端点的最差违例。其值为：
 - 0 ns，前提是针对最大延迟分析满足所有时序约束。
 - 如果存在违例，则为负值。
- Number of Failing Endpoints：含违例 ($WNS < 0$ ns) 的端点总数。
- Total Number of Endpoints：已分析的端点总数。

“Hold”区域（最小延迟分析）

“Design Timing Summary”部分的“Hold”区域用于显示最小延迟分析相关的所有检查：保持、移除和数据检查。

- Worst Hold Slack (WHS)：对应于最小延迟分析的所有时序路径的最差裕量。该值可为正值或负值。
- Total Hold Slack (THS)：所有 WHS 违例的总和，前提是仅考量每个时序路径端点的最差违例。其值为：
 - 0 ns，前提是针对最小延迟分析满足所有时序约束。
 - 如果存在违例，则为负值。
- Number of Failing Endpoints：含违例 ($WHS < 0$ ns) 的端点总数。
- Total Number of Endpoints：已分析的端点总数。

“Pulse Width”区域（管脚开关限制）

“Design Timing Summary”部分的“Pulse Width”区域可显示与管脚开关限制相关的所有检查：

- 最小低脉冲宽度
- 最小高脉冲宽度
- 最小周期

- 最大周期
- 例如，PCIE 或 GT [仅限 UltraScale 器件] 的相同叶节点单元的 2 个时钟管脚之间最大偏差。

报告值为：

- Worst Pulse Width Slack (WPWS)：对应于同时使用最小延迟和最大延迟时以上列出的所有时序检查的最差裕量。
- Total Pulse Width Slack (TPWS)：仅考量设计中每个管脚的最差违例时，所有 WPWS 违例的总和。其值为：
 - 0 ns，前提是满足所有相关约束。
 - 负值，前提是存在违例。
- Number of Failing Endpoints：含违例 ($WPWS < 0$ ns) 的管脚总数。
- Total Number of Endpoints：已分析的端点总数。

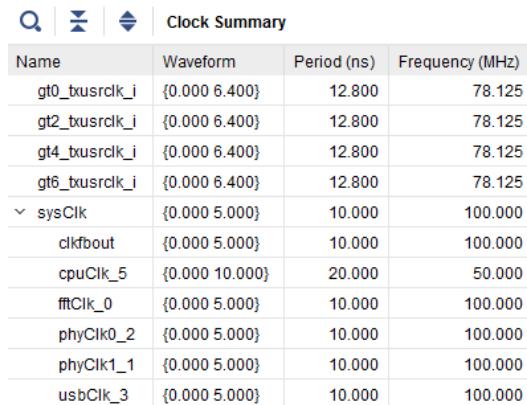
“时钟汇总 (Clock Summary)” 部分

“Timing Summary” 报告的 “Clock Summary” 部分所含信息与 `report_clocks` 所生成的信息相似：

- 设计中的所有时钟（包括 `create_clock` 和 `create_generated_clock` 创建的时钟以及工具自动创建的时钟）。
- 每个时钟的属性，如名称、周期、波形和目标频率。

 提示：名称的缩进反映了主时钟与生成时钟之间的关系。

图 21：“Timing Summary” 报告：“Clock Summary” 部分



Name	Waveform	Period (ns)	Frequency (MHz)
gt0_txusrclk_i	{0.000 6.400}	12.800	78.125
gt2_txusrclk_i	{0.000 6.400}	12.800	78.125
gt4_txusrclk_i	{0.000 6.400}	12.800	78.125
gt6_txusrclk_i	{0.000 6.400}	12.800	78.125
sysClk	{0.000 5.000}	10.000	100.000
clkfbout	{0.000 5.000}	10.000	100.000
cpuClk_5	{0.000 10.000}	20.000	50.000
fmClk_0	{0.000 5.000}	10.000	100.000
phyClk0_2	{0.000 5.000}	10.000	100.000
phyClk1_1	{0.000 5.000}	10.000	100.000
usbClk_3	{0.000 5.000}	10.000	100.000

“检查时序 (Check Timing)” 部分

“Timing Summary” 报告的 “Check Timing” 部分包含有关缺失时序约束或存在需审查的约束问题的路径的信息。要实现完整时序验收，所有路径端点都必须达成约束。

如需了解有关约束定义的更多信息，请参阅《Vivado Design Suite 用户指南：使用约束》([UG903](#))。

图22：“Timing Summary”报告：“Check Timing”部分

The screenshot shows a table titled "Check Timing" with the following data:

Timing Check	Count	Worst Severity
pulse_width_clock	8	Low
no_input_delay	1	Medium
no_clock	0	
constant_clock	0	
unconstrained_internal_endpoints	0	
no_output_delay	0	
multiple_clock	0	
generated_clocks	0	
loops	0	
partial_input_delay	0	
partial_output_delay	0	
latch_loops	0	

要以独立报告形式生成“Check Timing”，请执行以下任一操作：

- 运行“Reports”→“Timing”→“Check Timing”菜单命令。
- 运行Tcl `check_timing`命令。

从Tcl控制台运行时，可使用`-cells`选项将`check_timing`报告限定于一个或多个层级单元。该选项在“Check Timing”GUI中不可用。请注意，在Vivado Design Suite 2018.1中不限定`loops`和`latch_loops`类别。

如上图所示，默认情况下报告的检查列表包括：

- `pulse_width_clock`: 报告如下类型的时钟管脚：仅含与管脚关联的脉冲宽度检查、不含建立或保持时间检查、不含恢复、移除或`clk > Q`检查。
- `no_input_delay`: 不含任何输入延迟约束的非时钟输入端口数量。
- `no_clock`: 已定义的时序时钟无法到达的时钟管脚数量。此外还报告恒定时钟管脚。
- `constant_clock`: 检查连接到恒定信号(gnd/vss/data)的时钟信号。
- `unconstrained_internal_endpoints`: 无时序要求的路径端点(不包括输出端口)数量。此数值与缺失时钟定义数量存在直接关联，此定义数量同样可通过`no_clock`检查来报告。
- `no_output_delay`: 不含至少一个输出延迟约束的非时钟输出端口数量。
- `multiple_clock`: 多个时序时钟可到达的时钟管脚数量。如果在某一个时钟树中存在时钟多路复用器，则可能出现此情况。默认情况下，共享相同时钟树的时钟将结合在一起进行定时，这无法反映真实的时序情况。在任意给定时间，任一时钟树上仅限存在1个时钟。

如果您认为时钟树不应包含MUX，请复查时钟树，了解有多个时钟到达特定时钟管脚的方式和原因。

- `generated_clocks`: 引用不属于相同时钟树的主时钟源的生成时钟的数量。当主时钟与生成时钟源点之间的逻辑路径上禁用时序arc时，可能出现此情况。使用`-edges`选项可指定将此项检查应用于生成时钟的各个时钟沿：逻辑路径单边性(反向/非反向)必须与主时钟和生成时钟之间的时钟沿关联相匹配。
- `loops`: 设计中发现的组合循环数量。Vivado IDE时序引擎会自动断开循环以报告时序。
- `partial_input_delay`: 仅含最小输入延迟或最大输入延迟约束的非时钟输入端口数量。建立与保持时间分析均不报告这些端口。
- `partial_output_delay`: 仅含最小输出延迟或最大输出延迟约束的非时钟输出端口数量。建立与保持时间分析均不报告这些端口。

- `latch_loops`: 对穿越设计中的锁存器的循环进行检查并发出警告。报告的组合循环中将不包含这些循环，并且这些循环将影响相同路径上的锁存时间借用计算能力。

“时钟内部路径 (Intra-Clock Paths)” 部分

“Timing Summary Report”的“Intra-Clock Paths”部分（如下图所示）汇总了具有相同源时钟和目标时钟的时序路径的最差裕量和全部违例。

图 23: “Timing Summary” 报告: “Intra-Clock Paths” 部分

Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	WPWS (ns)	TPWS (ns)	Failing Endpoints (TPWS)	Total Endpoints (TPWS)
gt0_busrclk_i	rise - rise	9.887	0.000	0	218	rise - rise	0.135	0.000	0	218	6.000	0.000	0	154
gt2_busrclk_i	rise - rise	9.668	0.000	0	218	rise - rise	0.095	0.000	0	218	6.000	0.000	0	154
gt4_busrclk_i	rise - rise	8.853	0.000	0	218	rise - rise	0.123	0.000	0	218	6.000	0.000	0	154
gt6_busrclk_i	rise - rise	9.339	0.000	0	218	rise - rise	0.120	0.000	0	218	6.000	0.000	0	154
sysClk											3.000	0.000	0	10
clkfbout											8.592	0.000	0	3
cpuClk_5	rise - rise	9.754	0.000	0	5761	rise - rise	0.065	0.000	0	5761	9.600	0.000	0	3369
fftClk_0	rise - rise	3.658	0.000	0	8320	rise - rise	0.046	0.000	0	8320	4.358	0.000	0	1438
phyClk0_2	rise - rise	1.759	0.000	0	5958	rise - rise	0.076	0.000	0	5958	4.358	0.000	0	3787
phyClk1_1	rise - rise	1.136	0.000	0	5958	rise - rise	0.028	0.000	0	5958	4.358	0.000	0	3787
usbClk_3	rise - rise	1.114	0.000	0	2554	rise - rise	0.049	0.000	0	2554	4.600	0.000	0	1482
wbClk_4	rise - rise	9.921	0.000	0	2855	rise - rise	0.082	0.000	0	2855	9.600	0.000	0	1497

要查看详细信息，请单击左侧索引窗格内的“Intra-Clock Paths”下的名称。例如，您可查看有关每个时钟的裕量和违例汇总信息以及有关 SETUP、HOLD 或脉冲宽度检查的 N 条最差路径的详细信息。N 值定义方法如下：在命令行上使用 `-max_paths`，或者按时钟或路径组的最大路径数 (GUI) 来义。

针对每种分析类型，在其标签旁显示最差裕量值和报告的路径数。在下图中，已选中左侧索引窗格内的“Intra-Clock Paths”部分下的“Setup”汇总信息，并且在右侧窗口中显示的表格中列出了与该时钟相关的所有路径。

图 24: “Timing Summary” 报告: Intra-Clock Paths 详细信息

Tcl Console	Log	Reports	Package Pins	Messages	Design Runs	Power	Timing	x	Methodology	DRC	?	—	□	□
Intra-Clock Paths - phyClk1_1 - Setup														
> gt0_busrclk_i														
> gt2_busrclk_i														
> gt4_busrclk_i														
> gt6_busrclk_i														
> sysClk														
> clkfbout														
> cpuClk_5														
> fftClk_0														
> phyClk0_2														
phyClk1_1														
Setup 1.136 ns (10)														
Hold 0.028 ns (10)														
Timing Summary - impl_1 (saved) ×														

“时钟间路径 (Inter-Clock Paths)” 部分

与“Intra-Clock Paths”部分相似，“Timing Summary”报告的“Inter-Clock Paths”部分（如下图所示）汇总了不同源时钟和目标时钟之间的时序路径的最差裕量和全部违例。

图 25：“Timing Summary” 报告：Inter-Clock Paths 详细信息

The screenshot shows the Vivado IDE interface with the 'Timing' tab selected. A tree view on the left shows 'Inter-Clock Paths' expanded, with 'cpuClk_5 to sysClk' selected. This selection highlights four specific paths in the table below:

Name	Slack	^	Levels	High Fan...	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Except..
Path 201	7.703		1	1	cpuEngine/pm...d_o_reg[1]/C	or1200..._out[1]	4.051	2.630	1.421	10.000	cpuClk_5	sysClk	
Path 202	7.724		1	1	cpuEngine/pm...d_o_reg[2]/C	or1200..._out[2]	4.030	2.611	1.419	10.000	cpuClk_5	sysClk	
Path 203	7.776		1	1	cpuEngine/pm...d_o_reg[3]/C	or1200..._out[3]	3.978	2.666	1.312	10.000	cpuClk_5	sysClk	
Path 204	8.025		1	1	cpuEngine/pm...d_o_reg[0]/C	or1200..._out[0]	3.673	2.626	1.047	10.000	cpuClk_5	sysClk	

要查看详细信息，请单击左侧索引面板中“Inter-Clock Paths”下的名称。例如，您可查看有关每个时钟的裕量和违例汇总信息以及有关 SETUP、HOLD 或脉冲宽度检查的 N 条最差路径的详细信息。N 值定义方法如下：在命令行上使用 `-max_paths`，或者按时钟或路径组的最大路径数 (GUI) 来义。

“其它路径组 (Other Path Groups)” 部分

“Timing Summary” 报告中的“Other Path Groups”部分用于显示默认路径组和用户定义的路径组。下图显示了“Other Path Groups”汇总表的示例。选择左侧窗格中的“Other Path Groups”即可访问该表。

图 26：“Timing Summary” 报告：“Other Path Groups” 部分

The screenshot shows the Vivado IDE interface with the 'Timing' tab selected. The left sidebar has 'Other Path Groups' selected. The main area displays a table for 'Other Path Groups' with the following columns:

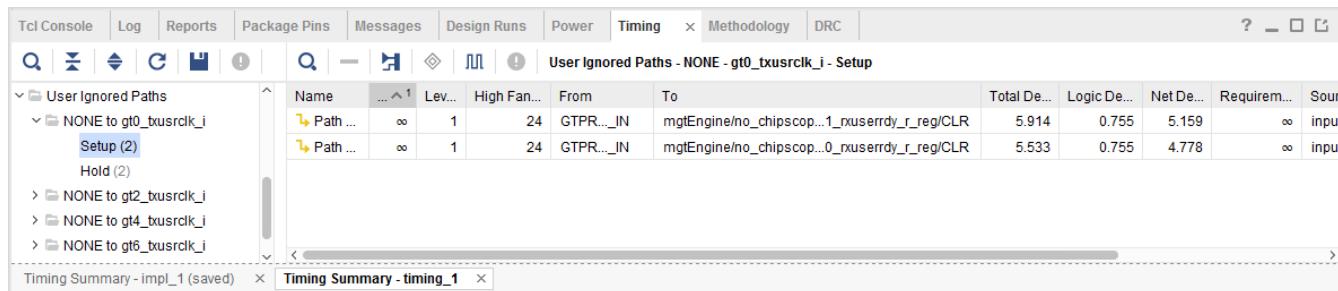
Path Group	From Clock	To Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)
async_default	wbClk_4	cpuClk_5	rise - rise	9.713	0.000	0	2934	rise - rise	0.471	0.000	0	2934
async_default	wbClk_4	fftClk_0	rise - rise	0.066	0.000	0	1280	rise - rise	0.802	0.000	0	1280
async_default	gt0_busrclk_i	gt0_busrclk_i	rise - rise	10.307	0.000	0	4	rise - rise	1.137	0.000	0	4
async_default	gt2_busrclk_i	gt2_busrclk_i	rise - rise	10.249	0.000	0	4	rise - rise	1.262	0.000	0	4
async_default	gt4_busrclk_i	gt4_busrclk_i	rise - rise	10.476	0.000	0	4	rise - rise	1.201	0.000	0	4
async_default	gt6_busrclk_i	gt6_busrclk_i	rise - rise	10.278	0.000	0	4	rise - rise	1.277	0.000	0	4
async_default	wbClk_4	usbClk_3	rise - rise	0.092	0.000	0	328	rise - rise	0.515	0.000	0	328
async_default	sysClk	wbClk_4	rise - rise	2.791	0.000	0	143	rise - rise	1.192	0.000	0	143

提示：**async_default** 是由 Vivado IDE 时序引擎自动创建的路径组。其中包括以异步时序检查结束的所有路径，例如，恢复和移除。这 2 项检查分别在“SETUP”类别和“HOLD”类别下报告，这 2 个类别分别对应于最大延迟分析和最小延迟分析。使用 `group_path` 创建的所有组也同样显示在此部分下。每个路径组中都可包含源时钟与目标时钟的任意组合。

“用户忽略的路径 (User-Ignored Paths)” 部分

“Timing Summary” 报告中的“User-Ignored Paths”部分（如下图所示）显示了时序分析期间由于 `set_clock_groups` 和 `set_false_path` 约束而忽略的路径。报告的裕量为无限。

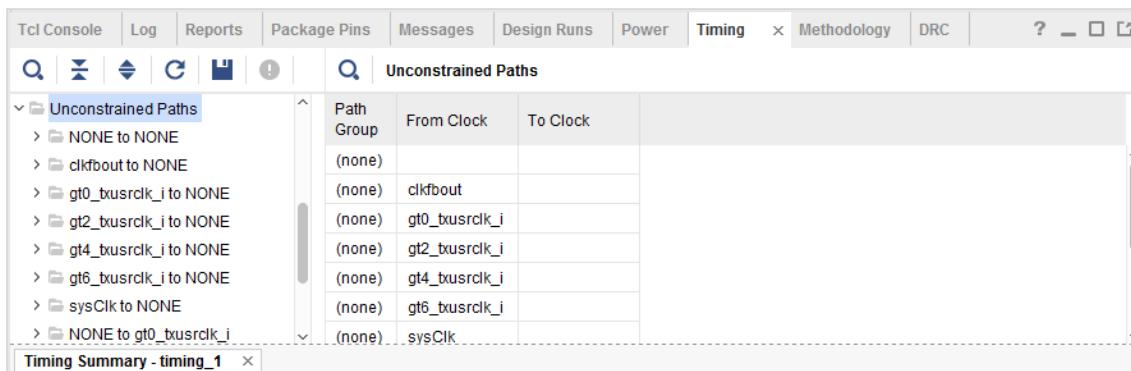
图 27：“Timing Summary”报告：“User-Ignored Paths”部分



“未约束的路径 (Unconstrained Paths)”部分

“Timing Summary”报告中的“Unconstrained Paths”部分可显示由于缺少时序约束而未定时的逻辑路径。这些路径按源和目标时钟对来分组。如果不存在与路径起点或端点关联的时钟，则时钟名称信息显示为空（或 NONE）。

图 28：“Timing Summary”报告：“Unconstrained Paths”部分



复查时序路径详情

大部分内容均可展开以显示按时钟对组织的路径。对于每个 SETUP、HOLD 和 Pulse Width 子部分，您可查看已报告的 N 条最差路径。选中其中任意路径即可在“路径属性 (Path Properties)”窗口的“报告 (Report)”选项卡下查看其详情。

要在新窗口中查看这些详情，请双击路径。

如需了解有关时序路径详情的更多信息，请参阅[第 5 章：执行时序分析](#)。

要访问每条路径的更多分析视图，请执行以下操作：

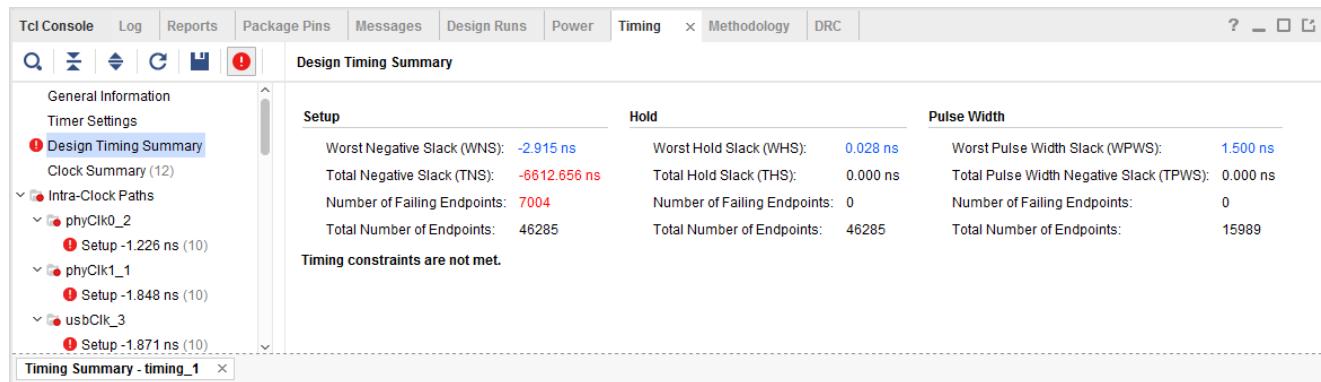
1. 在正确的面板内右键单击路径。
2. 从弹出菜单中选择下列选项之一：
 - “Schematic:” 打开路径的“原理图 (Schematic)”。
 - “Report Timing on Source to Destination:” 对此路径重新运行时序分析。
 - “Highlight:” 在“Device”和“Schematic”窗口中高亮此路径。

筛选含违例的路径

此报告可显示失败路径（红色）的裕量值。要聚焦这些违例，请单击“Show only failing paths”按钮❶。

下图显示了“Timing Summary”窗口，其中仅显示失败路径。

图 29：“Timing Summary”报告：违例路径筛选



时钟网络报告 (Report Clock Networks)

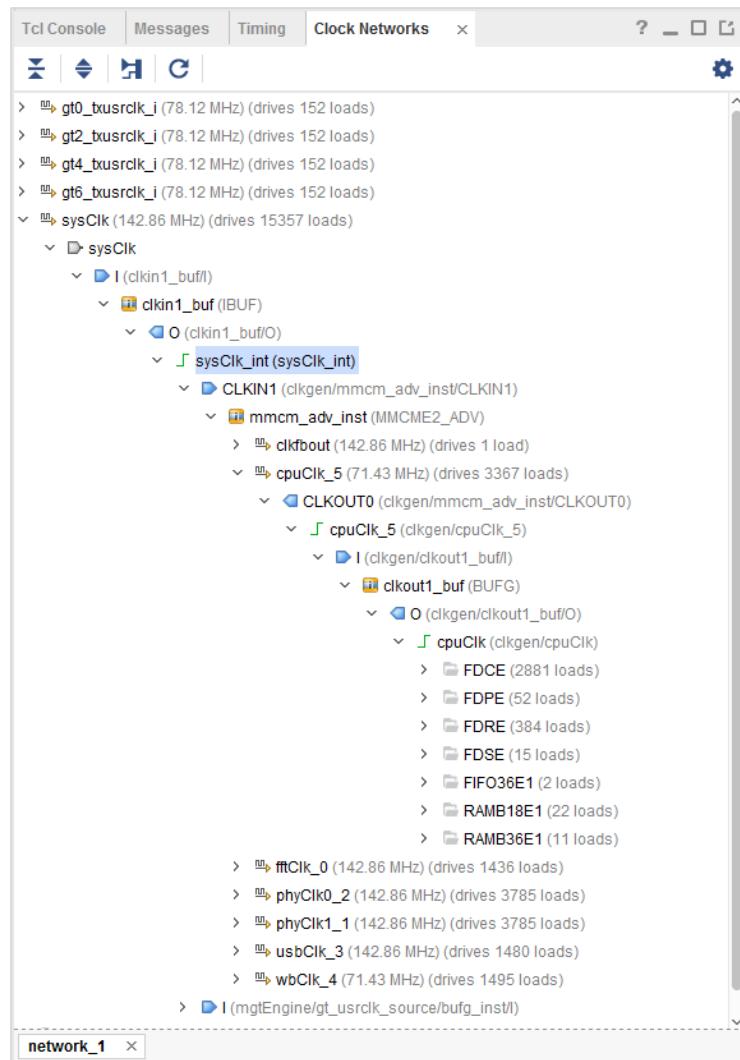
“Report Clock Network”命令可从以下位置运行：

- Vivado® IDE 中的 Flow Navigator，或者使用
- Tcl 命令：

```
report_clock_networks -name {network_1}
```

“Report Clock Networks”可提供设计中时钟树的树形视图。请参阅下图。每个时钟树都显示从源到端点的时钟网络，其中端点按类型排序。

图 30：时钟网络



时钟树：

- 显示用户定义的时钟或工具自动生成的时钟。
- 报告从 I/O 端口到负载的时钟。

注释：完整的时钟树仅在报告的 GUI 表单中详细说明。此报告的文本版本仅显示时钟根的名称。

- 可用于查找驱动其它 BUFG 的 BUFG。
- 显示驱动非时钟负载的时钟。

其中有 1 个文件夹包含设计中定义的每个基准时钟和所有生成时钟。有 1 个单独文件夹可显示每个未约束的时钟根。

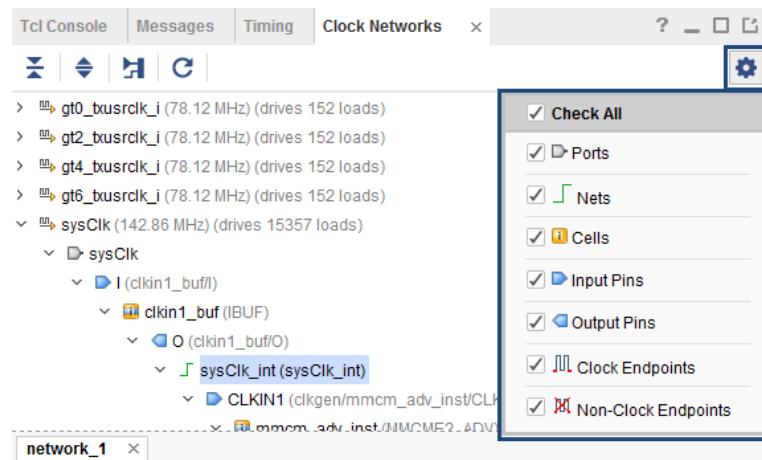
使用“端口 (Ports)”、“信号线 (Nets)”、“实例 (Instances)”筛选工具和相关按钮可减少时钟树上显示的数据量。筛

选选项可通过单击



图标来查看。

图31：时钟网络筛选工具



要查看时钟路径原理图，请执行以下操作：

1. 选择时钟树中的对象。
2. 运行“Trace to Source”弹出命令。

时钟交互报告 (Report Clock Interaction)

要查看“Clock Interaction Report”，请选择以下任一项：

- “Reports” → “Timing” → “Report Clock Interaction”
- “Flow Navigator” → “Synthesis” → “Report Clock Interaction”
- “Flow Navigator” → “Implementation” → “Report Clock Interaction”

等效的 Tcl 命令：`report_clock_interaction -name clocks_1`

当从 Tcl 控制台运行时，可使用 `-cells` 选项将交互报告限定于 1 个或多个层级单元。限定报告范围后，将仅报告含如下数据路径部分的路径：数据路径开始或结束于此类单元、与此类单元交汇或者完全包含于此类单元内。

“Report Clock Interaction”对话框

在 Vivado® IDE 中，“Report Clock Interaction”对话框包含以下选项卡：

- “结果名称 (Results Name)”字段
- “命令 (Command)”字段
- “在新选项卡中打开 (Open in a New Tab)”复选框
- “选项 (Options)”选项卡
- “定时器设置 (Timer Settings)”选项卡

“结果名称 (Results Name)” 字段

“Report Clock Interaction” 对话框顶部的 “Results name” 字段用于指定打开的图形报告的名称。

等效的 Tcl 选项： -name

“命令 (Command)” 字段

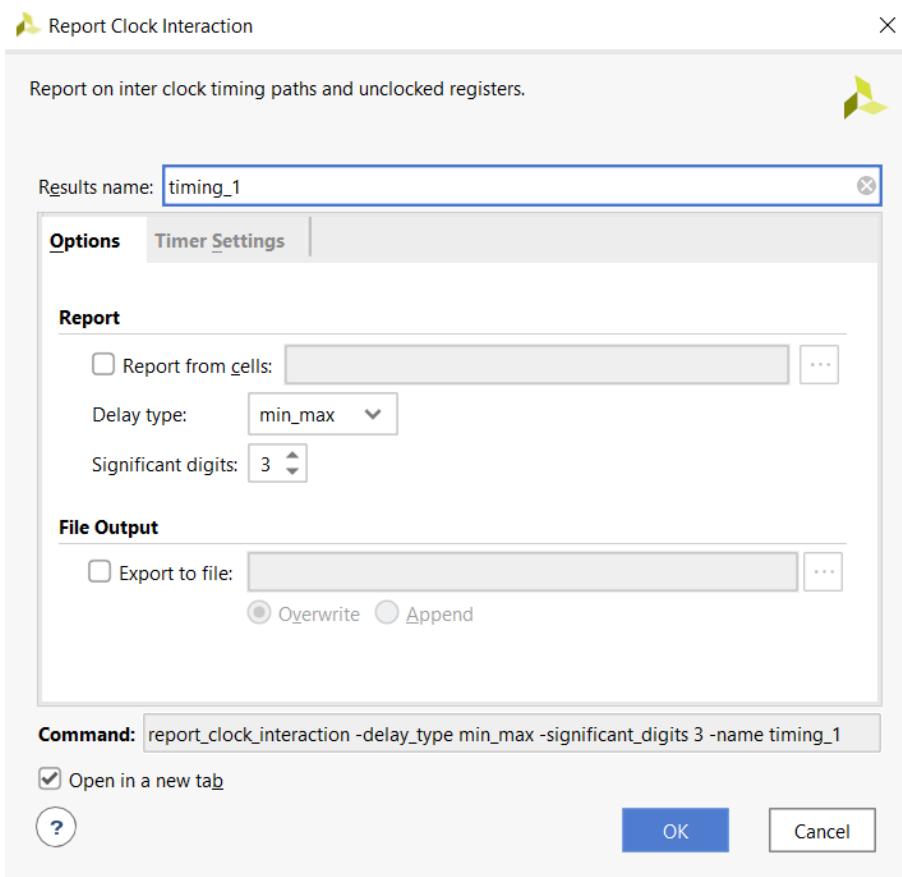
“Command” 字段用于显示等效于 “Report Clock Interaction” 对话框中指定的各种选项的 Tcl 命令行。

“在新选项卡中打开 (Open in a New Tab)” 复选框

使用 “Open in a New Tab” 复选框可执行以下任一操作：在新选项卡中打开结果，或者替换 “Results” 窗口中打开的最后一个选项卡。

“选项 (Options)” 选项卡

图 32：“Report Clock Interaction”：“Options” 选项卡



“Report Clock Interaction” 对话框的 “Options” 选项卡包含以下部分：

- “从单元报告 (Report from Cells)” 字段
- “延迟类型 (Delay Type)” 字段

- “有效位数 (Significant Digits)” 字段
- “文件输出 (File Output)” 部分

“从单元报告 (Report from Cells)” 字段

支持将时序报告范围限制于设计的一个或多个特定单元。报告将仅包含数据路径部分始于指定单元、止于该单元、跨该单元或完全包含于该单元内的路径。

等效的 Tcl 选项：-cells

“延迟类型 (Delay Type)” 字段

“Delay Type” 字段可用于设置要运行的分析类型。

- 对于综合后设计，默认情况下仅执行最大延迟分析（建立/恢复）。
- 对于已实现的设计，默认情况下将执行最小延迟和最大延迟分析（建立/保持，恢复/移除）。

要仅运行最小延迟分析（保持和移除），请选择延迟类型 min。

等效的 Tcl 选项：-delay_type

“有效位数 (Significant Digits)” 字段

“Significant Digits” 字段用于指定报告的值的有效位数。默认为 3 位数字。

等效的 Tcl 选项：-significant_digits

“文件输出 (File Output)” 部分

“File Output” 部分包括：

- “Write Results to File”：使用 “Write Results to File” 字段将结果写入指定文件。在 Vivado IDE 中，此报告显示在 “Clock Interaction” 窗口中。

等效的 Tcl 选项：-file

- “Overwrite/Append”：选择 “覆盖 (Overwrite)/追加 (Append)” 选项按钮以判定将此报告写入文件时：(1) 覆盖指定文件；还是 (2) 向现有报告追加新信息。

等效的 Tcl 选项：-append

“定时器设置 (Timer Settings)” 选项卡

如需了解有关该选项卡的详情，请参阅 [“定时器设置 \(Timer Settings\)” 选项卡](#)。

“时钟交互报告 (Clock Interaction Report)” 详情

“Clock Interaction” 报告用于分析从 1 个时钟域（源时钟）穿越到另 1 个时钟域（目标时钟）的时序路径。“Clock Interaction” 报告有助于识别可能存在数据丢失或亚稳态问题的情况。

运行 “Report Clock Interaction” 命令后，将在 “Clock Interaction” 窗口中打开结果。如下图所示，“Clock Interaction Report” 以时钟域矩阵的形式展示，其中源时钟位于纵轴，目标时钟位于横轴。

图33：时钟交互报告 (Report Clock Interaction)



矩阵颜色编码

矩阵拼块按颜色编码。矩阵颜色可通过以下方式确定：根据“Tools”→“Settings”→“Colors”→“Clock Interaction Chart”下定义的“图形编辑器(Graphical Editors)”的背景色，或者通过选择“时钟交互(Clock Interactions)”选项卡上的齿轮图标。欲知详情，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的相应内容。如需隐藏图例，请单击矩阵左侧工具栏上的“?”按钮。

- 无路径 - 黑色：不存在从源时钟跨越到目标时钟的时序路径。在此情况下，不存在时钟交互，无需报告。
- 已定时 - 绿色：源时钟与目标时钟存在同步关系，并且组合在一起安全定时。当2个时钟具有公共基准时钟和简单的周期比时，时序引擎会判定达成此状态。
- 用户忽略的路径 - 深蓝色：用户定义的伪路径或时钟组约束覆盖从源时钟跨越到目标时钟的所有路径。
- 部分伪路径 - 浅蓝色：用户定义的伪路径约束覆盖从源时钟跨越到目标时钟的部分时序路径，其中源时钟与目标时钟存在同步关系。
- 已定时（不安全） - 红色：源时钟与目标时钟存在异步关系。在此情况下，不存在公共基准时钟，也不存在可扩展周期。如需了解有关异步时钟和不可扩展时钟的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的相应内容。
- 部分伪路径（不安全） - 橙色：此类别与“已定时（不安全）”相同，区别在于从源时钟到目标时钟的至少1条路径因伪路径例外而被忽略。
- 仅最大延迟数据路径 - 灰色：`set_max_delay -datapath_only` 约束覆盖从源时钟跨越到目标时钟的所有路径。



重要提示！矩阵中单元颜色反映了时钟域之间约束的状态，而不是时钟域之间时序路径最差裕量的状态。绿色单元并不表示满足时序，仅表示跨时钟域的所有时序路径已正确定时，并且其时钟具有已知的相位关系。

时钟对分类 (Clock Pair Classification)

“Clock Pair Classification” 列提供了有关两个时钟之间缺失的公共基准时钟 (primary clock)、缺失的公共节点、缺失的公共路径和缺失的公共周期以及存在的虚拟时钟的信息。

以下按优先级从高到低顺序列出了可能的值。一旦检测到满足任一条件，报告命令就不会执行剩余的检查。

- 已忽略 (Ignored): 当 “时钟组 (Clock Group)” 、“伪路径 (False Path)” 或 “仅最大延迟数据路径 (Max Delay Datapath Only)” 完全覆盖时钟对时，将忽略分析。
- 虚拟时钟 (Virtual Clock): 至少 1 个时钟为虚拟时钟，并且不适用公共基准时钟检查或公共节点检查。
- 无公共时钟 (No Common Clock): 2 个时钟无公共基准时钟。
- 无公共周期 (No Common Period): 2 个时钟的周期不可扩展。
- 部分公共节点 (Partial Common Node): 2 个时钟显示为同步，但一小部分交汇路径不具有公共节点，并且无法安全定时。
- 无公共节点 (No Common Node): 2 个时钟显示为同步，但交汇路径无公共节点。
- 无公共相位 (No Common Phase): 2 个时钟不存在已知的相位关系。
- 无错 (Clean): 以上条件均不适用。

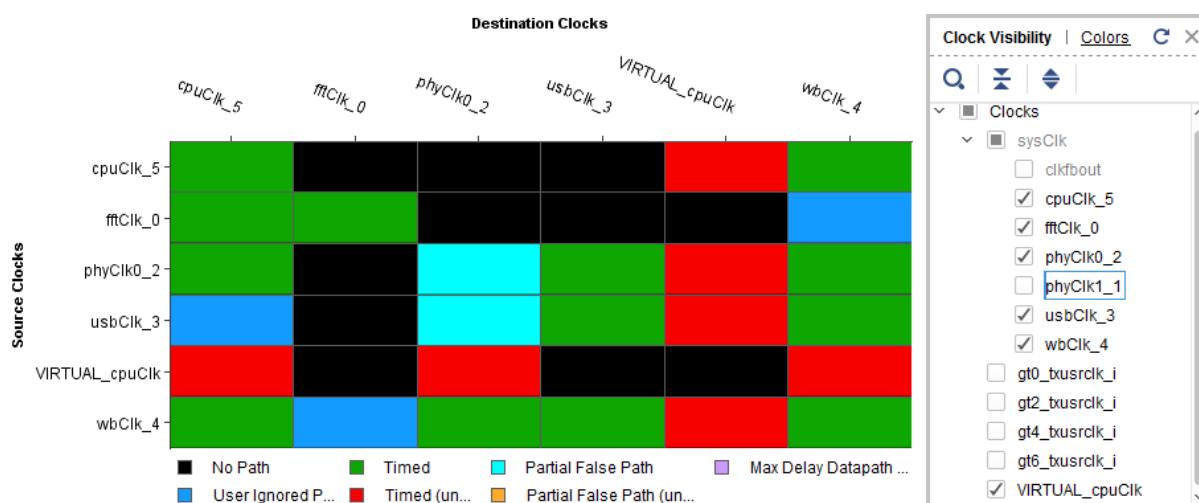
筛选时钟

如需筛选 “时钟交互 (Clock Interaction)” 报告中显示的源时钟，请执行以下操作：

1. 单击设置按钮以显示 “时钟可视性 (Clock Visibility)” 。
2. 选择要显示的源时钟。表中显示的目标时钟列表是从选定源时钟自动衍生的。

“Clock Visibility” 筛选工具通过限制时钟的数量来降低矩阵的复杂性，且不会减少矩阵下方的表格中报告的时钟交互的数量。通过单击工具栏中的 “隐藏不使用的时钟 (Hide Unused Clocks)” 按钮，还可显示和隐藏不直接对设计中的逻辑路径进行定时的时钟。由于这些时钟不参与 WNS/TNS/WHS/THS 计算，因此默认处于隐藏状态。

图 34: 时钟交互视图层次



时钟对裕量表

矩阵下的该表提供了源/目标时钟对的建立/恢复和/或保持/移除的时序裕量的完整综述。它还显示了有关最差路径、公共基准时钟和约束状态的路径要求的实用信息。请参阅“[时钟交互报告 \(Clock Interaction Report\)](#)”详情。其中提供了以上矩阵中未显示的详细信息。

数据排序

单击列标题即可对表中数据按值递增或递减顺序进行排序。

选择单元格与行

选择矩阵中的单元格会交叉选择下表的特定行。

选中表格中的某一行将高亮显示以上矩阵中的某个单元格。

表格中的列

表格中包含以下列：

- ID：所显示的源/目标时钟对的数字 ID。
- 源时钟 (Source Clock)：作为路径起点的时钟域。
- 目标时钟 (Destination Clock)：作为路径终点的时钟域。
- 时钟沿 (Edges (WNS))：用于计算最大延迟分析（建立/恢复）的最差负时序裕量的时钟沿。
- 最差负时序裕量 (WNS)：针对跨越指定时钟域的各条路径计算的最差时序裕量。负裕量表示存在路径违例问题，不满足所要求的建立（或恢复）时间。
- 总体时序负裕量 (TNS)：跨越指定时钟域的路径所包含的所有端点的最差时序裕量违例总和。
- 失败端点 (Failing Endpoints) (TNS)：未能满足时序的交汇路径中的端点数。违例总和对应于 TNS。
- 端点总数 (Total Endpoints) (TNS)：交汇路径中的端点总数。
- 路径要求 (Path Req) (WNS)：对应于 WNS 列中报告的路径的时序路径要求。如果 2 个时钟中至少 1 个时钟的上升沿和下降沿处于活动状态，或者如果已对 2 个时钟间的路径应用部分时序例外，那么在任意时钟对之间都可能存在多项路径要求。此列中报告的值并非总是对应难度最大的要求。

如需了解更多信息，请参阅[路径要求](#)。

- 时钟对分类 (Clock Pair Classification)：提供有关时钟对之间的公共节点和公共周期的信息。优先级按从高到低：“已忽略 (Ignored)”、“虚拟时钟 (Virtual Clock)”、“无公共时钟 (No Common Clock)”、“无公共周期 (No Common Period)”、“部分公共节点 (Partial Common Node)”、“无公共节点 (No Common Node)”、“无公共相位 (No Common Phase)” 和 “无错 (Clean)”。请参阅[时钟对分类 \(Clock Pair Classification\)](#)。
- 时钟间约束 (Inter-Clock Constraints)：显示源时钟和目标时钟之间所有路径的约束汇总信息。在[矩阵颜色编码](#)中列出了可能的值。以下是这些约束的定义示例：

```
set_clock_groups -async -group wbClk -group usbClk  
set_false_path -from [get_clocks wbClk] -to [get_clocks cpuClk]
```

同时选中最小延迟分析（保持/移除）时，在表中还会显示以下列：

- 时钟沿 (Edges (WHS))：用于计算最差保持时序裕量的时钟沿。
- 最差保持时序裕量 (WHS)：针对跨越指定时钟域的各条路径计算的最差时序裕量。负裕量表示存在路径违例问题，不满足所要求的保持（或移除）时间。

- 总体保持负时序裕量 (THS): 跨越最小延迟分析 (保持/移除) 的时钟域的路径所包含的所有端点的最差时序裕量违例总和。
- 失败端点 (Failing Endpoints) (THS): 未能满足时序的交汇路径中的端点数。违例总和对应于 THS。
- 端点总数 (Total Endpoints) (THS): 最小延迟分析 (保持/移除) 的交汇路径中的端点总数。
- 路径要求 (Path Req) (WHS): 对应于 WHS 列中报告的路径的时序路径要求。与 WNS 一样，2 个时钟间的最小延迟分析存在多个可能的路径要求，并且此列中报告的值并非总是对应于难度最大的要求。

如需了解更多信息，请参阅[第 5 章：执行时序分析](#)。

可从该表中选择 1 个或多个时钟对。从弹出菜单中可运行选定源/目标时钟对之间的“Report Timing”。

导出表格

运行“导出到电子数据表 (Export to Spreadsheet)”命令可将表格导出到 XLS 文件，以便在电子数据表中使用。

脉冲宽度报告 (Report Pulse Width)

下图所示“Pulse Width Report”用于检查设计是否满足每个实例时钟管脚的最小周期、最大周期、高脉冲时间和低脉冲时间要求。它还会检查已实现的设计中的相同实例（例如，PCIe® 时钟）的 2 个时钟管脚之间的最大偏差要求是否已得到满足。脉冲宽度裕量公式不包含抖动或时钟不确定性。

等效的 Tcl 命令：`report_pulse_width`

从 Tcl 控制台运行时，可通过使用 `-cells` 选项将脉冲宽度报告限定于一个或多个层级单元。限定报告范围时，报告中仅包含单元内部的管脚。该选项在“Report Pulse Width”GUI 中不可用。

注释：赛灵思 Integrated Software Environment (ISE) Design Suite 实现将此检查称为“组件开关限制 (Component Switching Limits)”。

图 35：脉冲宽度报告 (Report Pulse Width)

Timing Checks (4)	Check Type	Corner	Lib Pin	Reference Pin	Required	Actual	Slack	Location	Pin
gf6_husrclk_i (3)	Low Pulse Width	Slow	FDCE/C	n/a	0.400	3.500	3.100	SLICE_X4Y43	usbEngine0/dma_out/buffer...fo.next_wr_addr_reg[...
usbClk_3 (4)	High Pulse Width	Slow	FDRE/C	n/a	0.350	3.500	3.150	SLICE_X2Y46	OpMode_pad_0_o_reg[0]/C
usbClk_3 (4)	Min Period	n/a	RAMB36E1/CLKARDCLK	n/a	2.095	7.000	4.905	RAMB36_X2Y0	usbEngine0/usbEngineS...pyRam_reg_0/CLKARD...
sysClk (4)	Max Period	n/a	MMCM2_ADV/CLKOUT2	n/a	213.360	7.000	206.360	MMCM2_ADV_X0Y0	clkgen/mmcm_adv_inst/CLKOUT2

时序报告 (Report Timing)

综合后，在流程中可随时查阅“时序报告 (Report Timing)”以查看特定时序路径，对“Report Timing Summary”报告的时序问题进行进一步调查，或者报告特定时序约束的有效性和覆盖范围。“Report Timing”并不涵盖“脉冲宽度 (Pulse Width)”报告。

从 Tcl 控制台或从 GUI 运行此时序报告时，可使用 `-cells` 选项将其限定于 1 个或多个层级单元。限定报告范围后，将仅报告含如下数据路径部分的路径：数据路径开始或结束于此类单元、与此类单元交汇或者完全包含于此类单元内。

运行 “Report Timing”

如果设计已加载到内存中，则可以从 Clock Interaction Report 或 Report Timing Summary 路径列表运行 Report Timing，也可从菜单中运行。

从菜单运行 “时序报告 (Report Timing)”

要从菜单运行 “Report Timing”，请依次单击 “Reports” → “Timing” → “Report Timing”。

从 “时钟交互报告 (Clock Interaction Report)” 运行 “时序报告 (Report Timing)”

要从 “Clock Interaction Report” 运行 “Report Timing”，请执行以下操作：

1. 选中 “from/to” 时钟对。
2. 右键单击并选择 “Report Timing” 以将选定时钟作为源时钟或目标时钟来运行报告。

从 “路径列表 (Paths List)” 运行 “时序报告 (Report Timing)”

要从 “Paths List” 运行 “Report Timing”，请执行以下操作：

1. 选择路径。
2. 右键单击并选择 “Report Timing” 以在选定的路径起点和端点之间运行报告。

等效的 Tcl 命令：`report_timing`

设置特定 “Report Timing” 选项时，可在以下位置查看等效的 `report_timing` 命令：

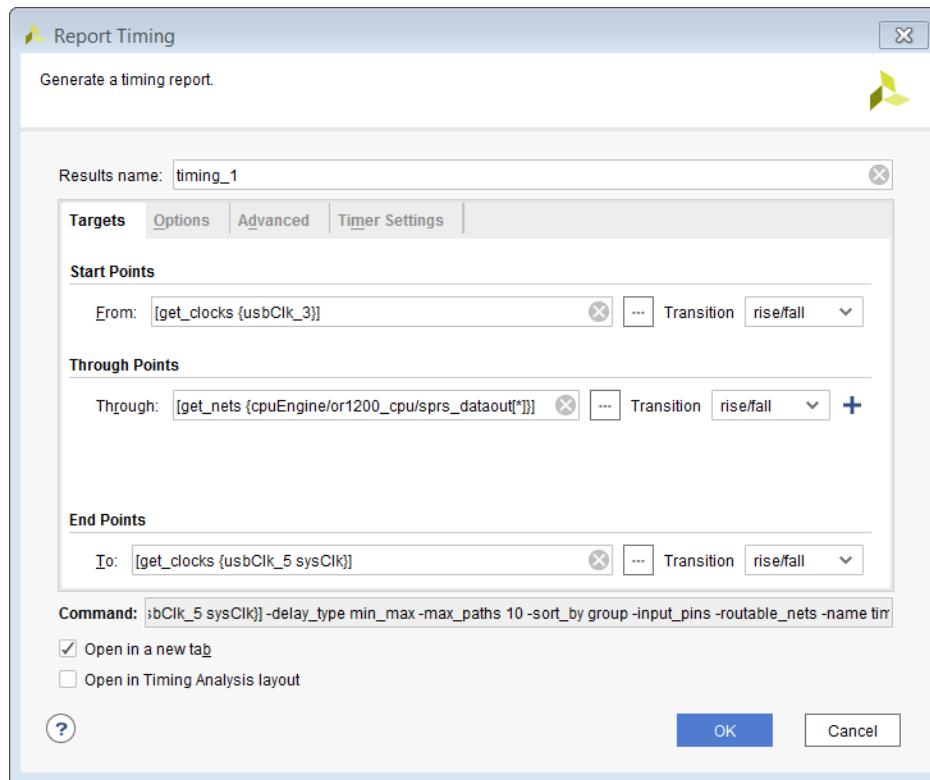
- 位于对话框底部的 “Command” 字段中，以及
- 执行之后在 Tcl 控制台中

在下一小节中对话框描述中列出了 `report_timing` 选项。总体上，“Report Timing” 选项与 “Report Timing Summary” 选项相同，但增加了部分筛选选项。

“Report Timing” 对话框

“目标 (Targets)” 选项卡

图 36：“Report Timing” 对话框：“Targets” 选项卡



“Report Timing” 提供了多个筛选选项，您必须使用这些选项才能报告特定路径或路径组。筛选依据为时序路径的结构。

- Startpoints (From): 起点列表，例如，时序单元时钟管脚、时序单元、输入端口、双向端口或源时钟。

如果将几个起点组合到 1 个列表中，报告的路径将始于其中任意网表对象。

“Rise/Fall” 筛选工具用于选择特定源时钟沿。

等效的 Tcl 选项：-from、-rise_from 或 -fall_from

- Through Points (Through): 管脚、端口、组合单元或信号线列表。

您可将多个网表对象组合到 1 个列表中，以便根据遍历其中任一对象的路径进行筛选。

您还可指定多个 “Through” 选项以优化筛选并报告路径，这些路径按命令选项中所列顺序遍历所有 “Through” 点分组。

“Rise/Fall” 筛选工具适用于数据沿。

 建议：使用默认值 (Rise/Fall)。

等效的 Tcl 选项：-through、-rise_through 或 -fall_through

- Endpoints (To): 端点列表，例如，时序单元的输入数据管脚、时序单元、输出端口、双向端口或目标时钟。

如果将几个端点组合到 1 个列表中，报告的路径将止于其中任意网表对象。

通常，“Rise/Fall”选项会选择一个特定数据沿。但如果您已指定目标时钟，那么它会选中特定时钟沿。

等效的 Tcl 选项：`-to`、`-rise_to` 或 `-fall_to`

“Report Timing”对话框中的“Targets”选项卡（请参阅上图）可定义从 `usbClk_3` 的上升时钟沿穿越任意 `cpuEngine/or1200_cpu/sprs_dataout[*]` 信号线到达 `cpuClk_5` 或 `sysClk` 时钟沿的路径。

“选项 (Options)” 选项卡

“Options”选项卡包含下列选项：

- [报告 \(Reports\)](#)
- [报告 \(Reports\)](#)
- [路径显示 \(Path Display\)](#)

报告 (Reports)

- 路径延迟类型 (Path delay type): 请参阅[“报告 \(Report\)” 部分](#)。
- 不报告未约束的路径 (Do not report unconstrained paths): 默认情况下，如果与筛选工具匹配的所有路径 (from/through/to) 都未约束，那么“Report Timing”会报告未约束的路径。如果您不希望报告中显示未约束的路径，请勾选此框。

等效的 Tcl 选项：`-no_report_unconstrained`

路径限制 (Path Limits)

- 每组路径数 (Number of paths per group): 请参阅[时序汇总报告 \(Report Timing Summary\)](#)。
- 单端点路径数 (Number of paths per endpoint): 请参阅[时序汇总报告 \(Report Timing Summary\)](#)。
- 将路径限制到组 (Limit paths to group): 筛选 1 个或多个时序路径组。每个时钟都与 1 个组关联。Vivado IDE 时序引擎还会创建默认组（例如，`**async_default**`），用于对以恢复或移除时序检查结尾的所有路径进行分组。

等效的 Tcl 选项：`-group`

路径显示 (Path Display)

- 显示裕量大于该值的路径 (Display paths with slack greater than): 基于路径的裕量值显示报告的路径。
等效的 Tcl 选项：`-slack_greater_than`
- 显示裕量小于该值的路径 (Display paths with slack less than): 请参阅[时序汇总报告 \(Report Timing Summary\)](#)。
- 有效位数 (Number of significant digits): 请参阅[时序汇总报告 \(Report Timing Summary\)](#)。
- 路径排序依据 (Sort paths by): 按组（默认）或者按裕量显示报告的路径。按组排序时，按每个组和每种分析类型 (`-delay_type min/max/min_max`) 报告 N 条最差的路径。

各组根据各自的最差路径排序。在此列表中，按组中所含违例的严重性从高到低排列。

按裕量排序时，将按分析类型报告 N 条最差的路径（所有组已组合在一起），并按裕量值升序排序。

等效的 Tcl 选项：`-sort_by`

“高级 (Advanced)” 选项卡

“Advanced” 选项卡与[时序汇总报告 \(Report Timing Summary\)](#) 包含相同的选项。

“定时器设置 (Timer Settings)” 选项卡

“Timer Settings” 选项卡与[时序汇总报告 \(Report Timing Summary\)](#) 包含相同的选项。

复查时序路径详情

单击 “OK” 以运行报告命令后，会打开 1 个新窗口。这样您即可复查其中内容。在其中可查看针对选定的每一种类型 (min/max/min_max) 的分析所报告的 N 条最差路径。

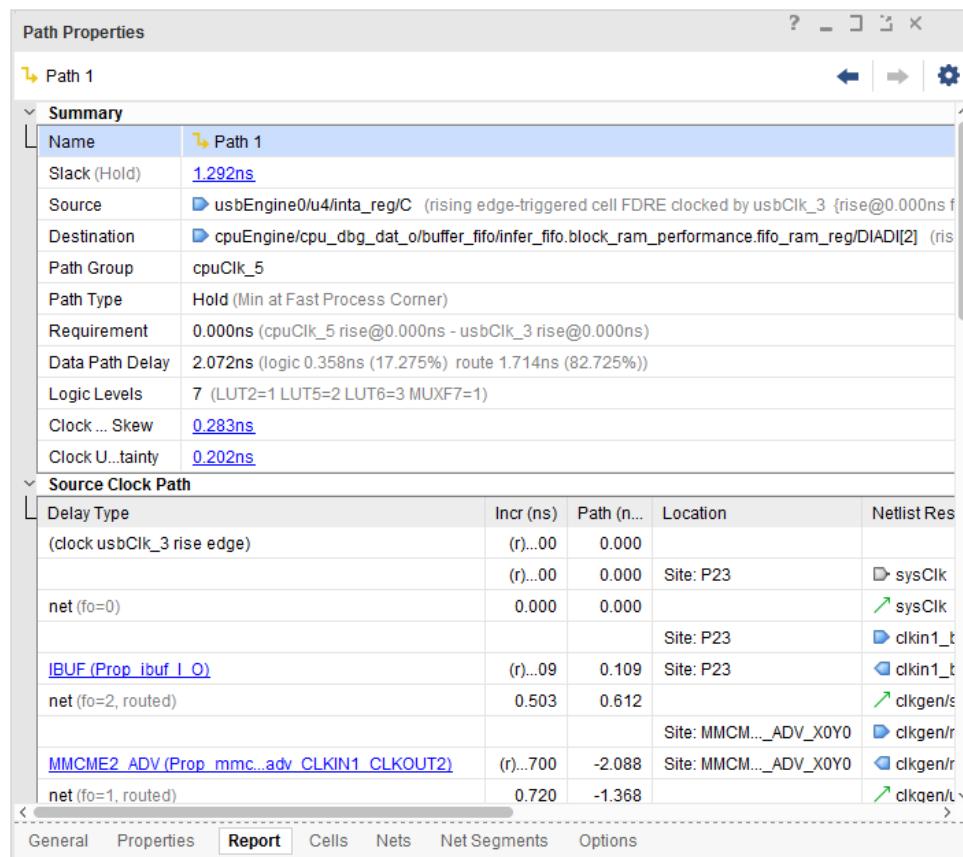
下图显示的 “Report Timing” 窗口中已选中最小和最大分析 (SETUP 和 HOLD) ，且 N=4。

图 37：时序报告路径列表

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay
Path 1	1.292	7	165	usbEngine0/u4/inta_reg/C	cpuEngine/cpu...m_reg/DIADI[2]	2.072	0.358
Path 2	1.373	7	165	usbEngine0/u4/inta_reg/C	cpuEngine/or120...36_s36/DIADI[4]	2.159	0.358
Path 3	1.416	7	165	usbEngine0/u4/inta_reg/C	cpuEngine/or120...36_s36/DIADI[2]	2.201	0.358
Path 4	1.445	8	165	usbEngine0/u4/inta_reg/C	cpuEngine/or120...and_b_reg[4]/D	2.062	0.386
Path 5	1.456	7	165	usbEngine0/u4/inta_reg/C	cpuEngine/or12...6_s36/DIBDI[3]	2.241	0.358
Path 6	1.463	7	165	usbEngine0/u4/inta_reg/C	cpuEngine/or120...36_s36/DIADI[6]	2.248	0.358
Path 7	1.491	7	165	usbEngine0/u4/inta_reg/C	cpuEngine/or12...6_s36/DIBDI[5]	2.276	0.358
Path 8	-	-	-	-	-	-	-

选中其中任意路径即可在 “路径属性 (Path Properties)” 窗口的 “报告 (Report)” 选项卡下查看其详情。

图 38：时序路径属性窗口



要在新窗口中查看这些详情，请双击路径。

如需了解有关时序路径详情的更多信息，请参阅[第 5 章：执行时序分析](#)。

要访问每条路径的更多分析视图，请右键单击右侧窗格中的路径，选择以下操作之一：

- 查看时序路径原理图。
- 在选定路径的相同起点和端点上重新运行时序分析。
- 在“Device”和“Schematic”窗口中高亮此路径。

筛选含违例的路径

在此报告中，失败路径的裕量值以红色显示。要集中显示这些违例，请单击“Show only failing checks mode”。

数据手册报告 (Report Datasheet)

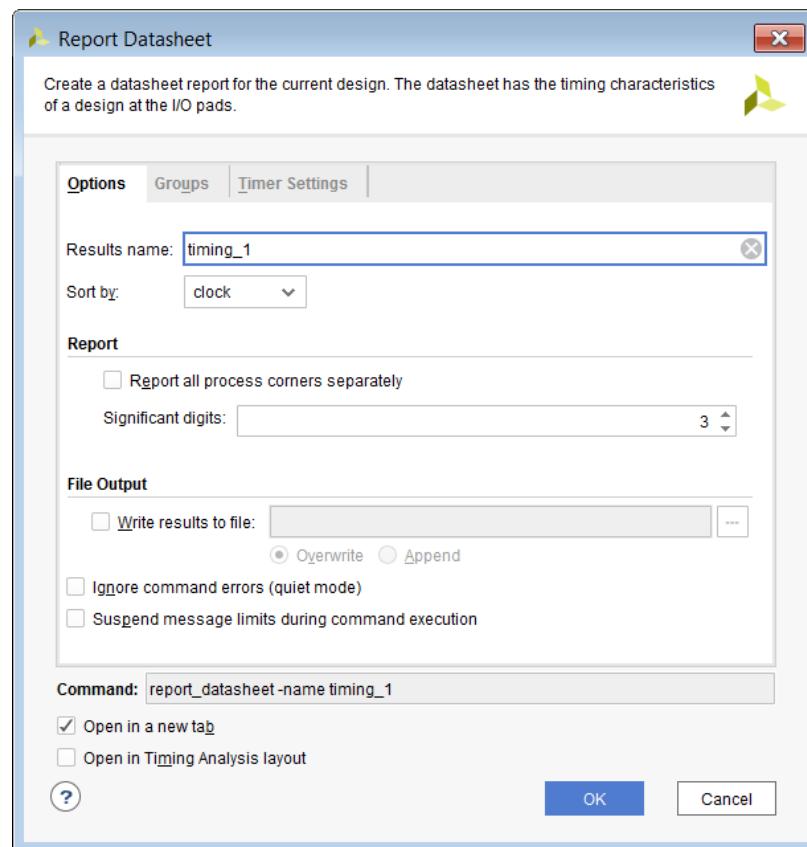
Report Datasheet 命令用于报告系统级集成中使用的 FPGA 操作参数。

“数据手册报告 (Report Datasheet)” 对话框

在 Vivado® IDE 中，选择 “Reports” → “Timing” → “Report Datasheet” 以打开 “Report Datasheet” 对话框。请参阅下图：

“数据手册报告 (Report Datasheet)” 对话框：“选项 (Options)” 选项卡

图 39： “数据手册报告 (Report Datasheet)” 对话框：“选项 (Options)” 选项卡



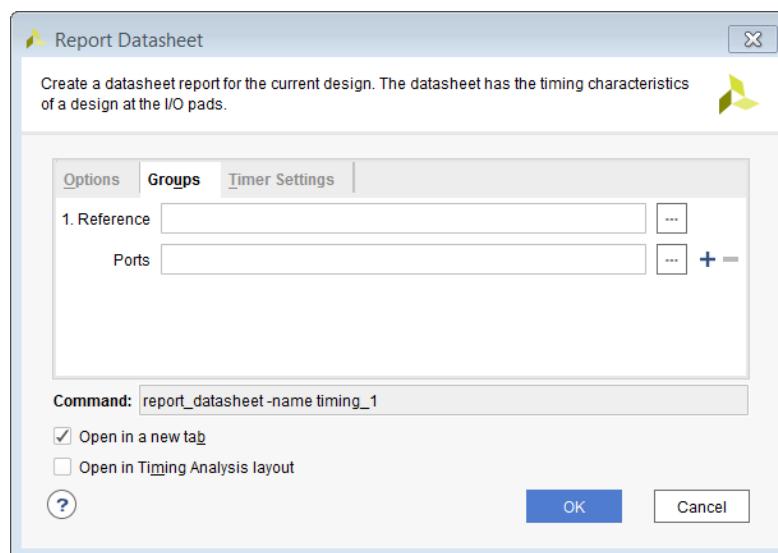
“Report Datasheet” 对话框的 “Options” 选项卡包含以下内容：

- 结果名称 (Results name): 指定 Report Datasheet 命令返回的结果的名称。报告将在 Vivado IDE 的 “Timing” 窗口中以指定名称打开。
等效的 Tcl 选项：-name
- 排序依据 (Sort by): 按端口名称或时钟名称对结果进行排序。
等效的 Tcl 选项：-sort_by
- 单独报告所有工艺角点 (Report all process corners separately): 报告当前设计中所有已定义的工艺角点的数据。
等效的 Tcl 选项：-show_all_corners
- 有效位数 (Significant digits): 指定报告的值的有效位数。默认为 3 位数字。
等效的 Tcl 选项：-significant_digits

- 将结果写入文件 (Write results to file): 将结果写入指定文件名。默认情况下，报告将写入 Vivado IDE 的“时序 (Timing)”窗口。
等效的 Tcl 选项: `-file`
- “覆盖 (Overwrite)” 或 “追加 (Append)”: 当报告写入文件时，这 2 个选项可用于确定是覆盖指定文件还是向现有报告追加新信息。
等效的 Tcl 选项: `-append`
- 忽略命令错误 (Ignore command errors): 以静默方式执行命令，忽略所有命令行错误，不返回任何消息。返回 `TCL_OK`，忽略执行期间遇到的所有错误。
等效的 Tcl 选项: `-quiet`
- 暂挂消息限制 (Suspend message limits): 临时覆盖所有消息限制。返回来自此命令的所有消息。
等效的 Tcl 选项: `-verbose`
- 命令 (Command): 显示等效于“Report Datasheet”对话框中指定的各种选项的 Tcl 命令行。
- 在新选项卡中打开 (Open in a new tab): 在新选项卡中打开结果，或替换“Results”窗口中打开的最后一个选项卡。
- 在时序分析布局中打开 (Open in Timing Analysis layout): 将当前视图布局复位为“Timing Analysis”视图布局。

“数据手册报告 (Report Datasheet)” 对话框：“组 (Groups)” 选项卡

图 40: “数据手册报告 (Report Datasheet)” 对话框：“组 (Groups)” 选项卡



“Report Datasheet”对话框的“Groups”选项卡支持您通过指定参考端口和要报告的其它端口来定义自己的定制端口组以供分析。如果不指定“Groups”，定时器会基于发送时钟自动查找输出端口组，并基于该时钟报告偏差。

“Report Datasheet”对话框的“Groups”选项卡包括：

- 参考 (Reference): 指定用于偏差计算的参考端口。大部分情况下，此端口为源同步输出接口的时钟端口。
等效的 Tcl 选项: `-group`

- 端口 (Ports): 定义要报告的其它端口。
 - 请注意 “Ports” 字段右侧的 “+” 和 “-” (加号和减号) 按钮。
 - “+” (加号) 按钮用于指定多个组，每个组都带有其自己的参考时钟端口，使您能够定义新的端口组，包括新的参考时钟。
 - “-” (减号) 按钮可根据需要移除其它端口组。

“数据手册报告 (Report Datasheet)” 对话框：“定时器设置 (Timer Settings)” 选项卡

如需了解有关该选项卡的详情，请参阅 [“定时器设置 \(Timer Settings\)” 选项卡](#)。

数据手册报告详情

常规信息

本章节提供了有关设计和赛灵思器件的详细信息以及报告时的工具环境信息。

- 设计名称 (Design Name): 设计的名称
- 器件 (Part): 目标赛灵思器件和速度文件信息
- 版本 (Version): 生成报告时所使用的 Vivado 工具版本
- 日期 (Date): 报告的日期和时间戳
- 命令 (Command): 用于生成报告的命令行

输入端口建立/保持 (Input Ports Setup/Hold)

此报告可显示每个输入端口有关参考时钟的最差情况建立和保持要求。此外还可报告用于捕获输入数据的内部时钟。

输出端口的最大/最小延迟 (Max/Min Delays for Output Ports)

显示每个输出端口有关参考时钟的最差情况最大和最小延迟。此外还可报告用于发送输出数据的内部时钟。

时钟间的建立时间 (Setup Between Clocks)

针对每一对时钟，将报告所有时钟沿组合的最差情况建立时间要求。

输入总线的建立/保持 (Setup/Hold for Input Buses)

自动推断输入总线，并显示其最差情况建立和保持时间要求。整个总线的最差情况数据窗口是最大建立和保持时间值的总和。如果输入端口受到约束，则将同时报告裕量。

针对已定义 IDELAY 的输入时钟，报告将显示最优化的分接点。最优化的分接点可用于配置 IDELAY 以实现平衡的建立和保持裕量。

源偏移是 2 个窗口之间的增量。第 1 个窗口由输入端口的时钟相关建立和保持时间定义。第 2 个窗口衍生自输入延迟和时钟周期。如果输入时钟采用该值偏移，则它将位于窗口的中心。

下图报告的设计中，DDR 输入总线 `vsf_data[0:9]` 的最差情况数据窗口为 1.663 ns。理想的时钟偏移为 1.063 ns。

图 41：输入总线的建立和保持延迟

注释: 可使用以下 Tcl 命令指定最优化分接点:

```
set_property IDELAY_VALUE 13 [get_cells idelay_clk]
```

输出总线的最大/最小延迟 (Max/Min Delays for Output Buses)

输出总线采用自动推断，并显示其最差情况下的最大和最小延迟。总线偏差也将一并报告。针对总线偏差计算，将1个位视为参考位，其它每个位的偏移都基于此参考位来计算。最差偏移即整个总线的偏差。

组的最大/最小延迟 (Max/Min Delays for Groups)

对于“源同步输出接口 (Source Synchronous Output Interfaces)”，需要存在前向时钟相关的输出偏差。通过指定参考端口作为前向时钟端口，可生成定制组报告。该表类似于“Max/Min Delays for Output Buses”，但参考端口用作为计算源偏移和总线偏差的参考位。

注释: 如内容为空, 则隐藏此部分。

例如，对于 DDR 输出偏差计算，如果应将前向时钟端口 (`rldii_i_ck_n[0]`) 相关的多个位（如，`rldii_i_a[0-19]`、`rldii_i_ba[0-3]`、`rldii_i_ref_n`、`rldii_i_we_n`）分组在一起，可使用以下命令：

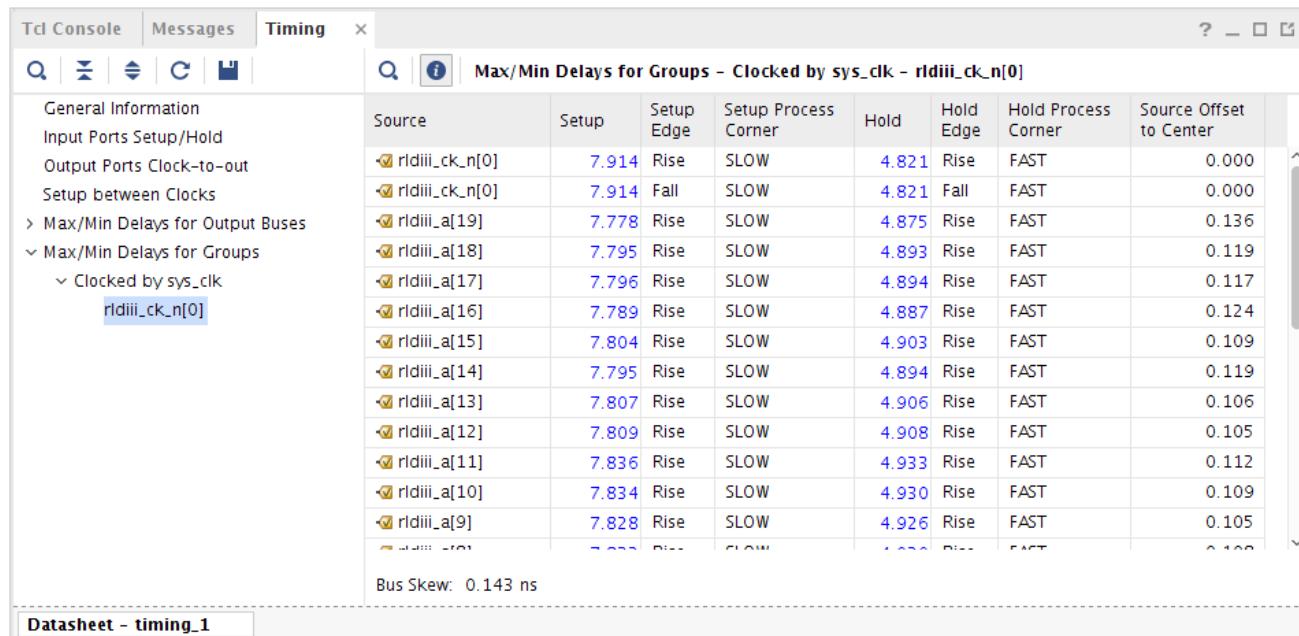
```
report_datasheet -group [get_ports {rldii_cke_n[0] rldii_a[*] rldii_ba[*] rldii_ref_n rldii_we_n}] -name timing_1
```

该组列表中的首个端口被视为参考管脚。

对于所有这些部分，通过多角分析 (multi-corner analysis) 来计算最差情况的数据。如果使用 `-show_all_corners`，那么将单独报告每个角点的最差情况数据。

下图显示了对应此示例的报告数据手册。

图 42：报告数据手册的最大/最小延迟示例



例外报告 (Report Exceptions)

在综合后的流程中可随时使用 Report Exceptions 命令。Report Exception 命令用于报告以下信息：

- 在设计中已设置并且影响时序分析的所有时序例外。
- 在设计中已设置但由于被其它时序例外覆盖而被忽略的所有时序例外

Report Exception 命令分析的时序例外包括（按优先级顺序）：

- 时钟组
- 伪路径
- 最大/最小延迟
- 多周期路径

Report Exception 是 1 条强大的命令，有助于对时序例外相关问题进行调试。某些设计的时序约束包含复杂的时序例外。由于时序例外的优先级不尽相同，因此很快就会难以理解哪些时序例外将被其它例外部分或全部忽略。Report Exception 可报告被部分覆盖和完全覆盖的时序例外。它还可提供覆盖约束的提示。

如需了解有关 report_exceptions 命令行选项的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的相应内容。如需了解有关时序例外优先级顺序的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的相应内容。

report_exceptions 命令具有多种操作模式：

- 报告影响时序分析的时序例外
- 报告已忽略的时序例外
- 报告时序例外覆盖范围
- 报告针对 -from/-through/-to 命令行选项指定的无效对象
- 写出仅含有效对象的时序例外
- 写出由时序引擎合并的时序例外

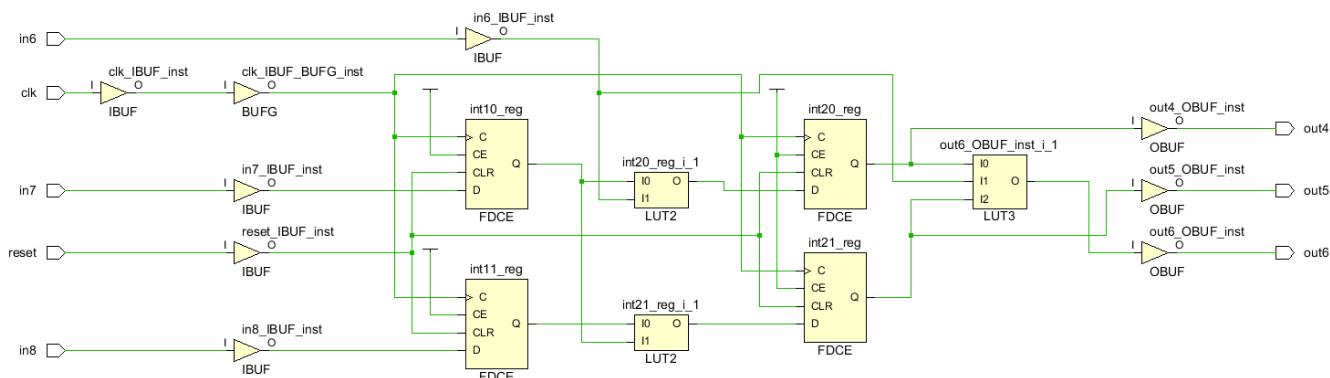
注释：即使“时钟组 (Clock Groups)”严格意义上不属于时序例外，但仍在 report_exceptions 命令的覆盖范围内，因为时钟组可能覆盖其它时序例外。

注释：将 report_exceptions 命令与 -from/-through/-to 选项搭配使用，即可仅报告使用相同 -from/-through/-to 命令行选项定义的时序例外。指定的模式可能不同，但在每个 -from/-through/-to 中必须至少存在 1 个匹配对象（单元、信号线、管脚或端口）方可将其报告为例外。

示例：完成时序分析后报告时序例外

本示例描述了如何对设计中的部分时序例外进行处理，如下图所示。设计已完全约束（clk 以及 clk 相关的输入/输出延迟已定义）。

图 43：对应时序例外的完全约束设计示例



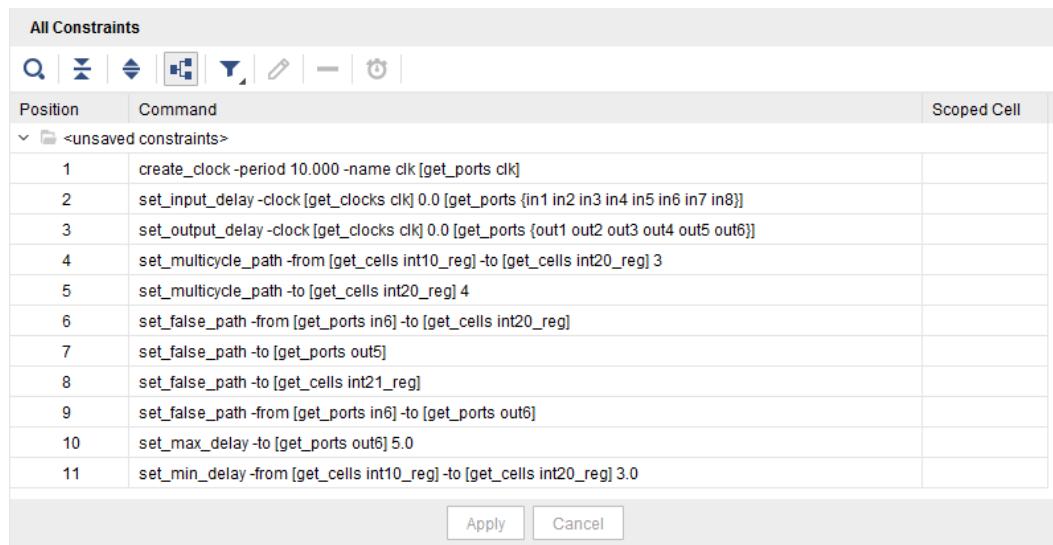
Report Exception 命令的第一种操作模式为 report_exceptions。

1. 请选择“Window” → “Timing Constraints”。
2. 在“Timing Constraints”窗口中，为设计添加以下时序例外。

```
set_multicycle_path 3 -from [get_cell int10_reg] -to [get_cell int20_reg]
id="ab439753">>set_multicycle_path 4 -to [get_cell int20_reg]
set_false_path -from [get_ports in6] -to [get_cell int20_reg]
set_false_path -to [get_ports out5]
set_false_path -to [get_cell int21_reg]
set_false_path -from [get_ports in6] -to [get_ports out6]
set_max_delay 5 -to [get_ports out6]
set_min_delay 3 -from [get_cells int10_reg] -to [get_cell int20_reg]
```

“Timing Constraints”窗口会显示应用于设计的时序约束，如下图所示。

图 44：显示时序约束变更的约束窗口



下图显示了实际例外报告 (report_exceptions)。

图 45：例外报告

Position	From	Through	To	Setup	Hold	Status
4	[get_cells int10_reg]	*	[get_cells int20_reg]	cycles=3	-	
5	*	*	[get_cells int20_reg]	cycles=4	-	Partially overridden path by MCP 4 - FP 6
6	[get_ports in6]	*	[get_cells int20_reg]	false	false	
7	*	*	[get_ports out5]	false	false	
8	*	*	[get_cells int21_reg]	false	false	
9	[get_ports in6]	*	[get_ports out6]	false	false	
10	*	*	[get_ports out6]	max=5	-	Partially overridden path by FP 9
11	[get_cells int10_reg]	*	[get_cells int20_reg]	-	min=3	

“例外报告”包含以下信息：

- “Position”列指示约束位置编号。此位置编号与“Timing Constraint”窗口报告的位置相同（如前文中所示）。
- “From” / “Through” / “To”列指示使用 `-*from/-*through/-*to` 命令行选项（包括这些选项的所有 rise/fall 版本）所指定的模式或对象。未指定关联选项时，将显示星号。
- “Setup” / “Hold”列指示约束适用于建立时间检查和/或保持时间检查。下表显示了“Setup” / “Hold”列的命名约定：

表 1：“Setup” / “Hold”列的命名约定

短名称	时序例外
<code>cycles=</code>	<code>set_multicycle_path</code>
<code>false</code>	<code>set_false_path</code>
<code>max=</code>	<code>set_max_delay</code>
<code>max_dpo=</code>	<code>set_max_delay -datapath_only</code>
<code>min=</code>	<code>set_min_delay</code>
<code>clock_group=</code>	<code>set_clock_group</code>

- 当约束被另一个时序例外部分覆盖时，“Status”列将报告 1 条消息。下表显示了“Status”列的命名约定：

表2：“Status”列命名约定

短名称	时序例外
MCP	多周期路径
FP	伪路径
MXD	最大延迟
MND	最小延迟
CG	时钟组

注释：仅当时钟组约束覆盖另一个时序例外时，才会在 report_timing -ignored 命令的“Status”列中报告时钟组。

本例中有2条有关被部分覆盖的约束的消息：

- 时序约束位置5 (set_multicycle_path 4 -to [get_cell int20_reg])，基于“Timing Constraints”窗口) 被多周期约束位置4 (set_multicycle_path 3 -from [get_cell int10_reg] -to [get_cell int20_reg]) 和伪路径约束位置6 (set_false_path -from [get_ports in6] -to [get_cell int20_reg]) 部分覆盖。
- 时序约束位置10 (set_max_delay 5 -to [get_ports out6]) 被伪路径位置9 (set_false_path -from [get_ports in6] -to [get_ports out6]) 部分覆盖。

报告已忽略的时序例外

“Report Exception”命令的第2种操作模式为 report_exceptions -ignored.

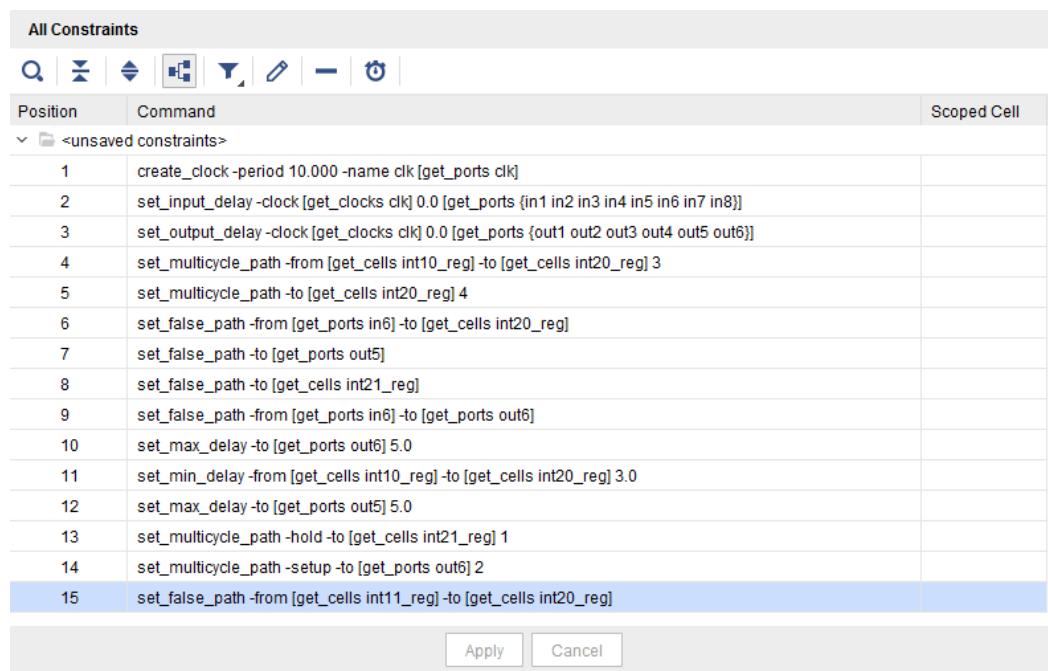
作为演示，请在前述时序例外的基础上添加以下时序例外：

```
set_max_delay 5 -to [get_ports out5]
set_multicycle_path 1 -hold -to [get_cell int21_reg]
set_multicycle_path 2 -setup -to [get_ports out6]
set_false_path -from [get_cell int11_reg] -to [get_cell int20_reg]
```

先前章节中的时序例外已涵盖所有这些例外（报告影响时序分析的时序例外），或者这些例外均以不存在的路径为目标（在寄存器 int11_reg 和 int20_reg 之间不存在物理连接）。

添加上述4项约束后，“Timing Constraints”窗口如下图所示。

图 46：“Timing Constraints”窗口



“例外报告 (Exceptions Report)” (report_exceptions -ignored) 如下图所示：

图 47：例外报告

Exceptions Report						
Position	From	Through	To	Setup	Hold	Status
12	*	*	[get_ports out5]	max=5	-	Totally overridden path by FP 7
13	*	*	[get_cells int21_reg]	-	cycles=1	Totally overridden path by FP 8
14	*	*	[get_ports out6]	cycles=2	-	Totally overridden path by FP 9 - MXD 10
15	[get_cells int11_reg]	*	[get_cells int20_reg]	false	false	Non-existent path

注释：“状态 (Status)” 列提供了有关忽略时序例外的原因的一些解释。

报告时序例外覆盖范围

Vivado 工具可生成应用于设计的每个有效时序例外的详细覆盖范围。报告中包含所有时序例外，包括已完全覆盖的时序例外或起点和端点间不含路径的时序例外。

例外覆盖范围报告是使用 -coverage 命令行选项生成的：

```
report_exceptions -coverage
```

对于每个有效的时序例外，此报告都包括以下信息：

- 约束位置编号。
- 通过 -from/-through/-to 命令行选项所选的对象数。
- 将时序例外应用到的管脚数量与通过 -from/-through/-to 命令行选项指定的管脚数量进行比较所得的覆盖范围（以百分比来表示）。

注释：指定单元对象时，Vivado 工具会将单元扩展到有效的管脚对象中。此单元到管脚转换可能导致覆盖率下降，因为通常时序例外仅到达小部分管脚。

下图显示了例外覆盖范围报告。

图 48：例外覆盖范围报告

Exceptions Report										
Position	Type	Setup	Hold	From	Through	To	Endpoints	From (%)	Through (%)	To (%)
4	Multicycle Path	cycles=3	-	1 cells		1 cells 1		100.00		33.33
5	Multicycle Path	cycles=4	-			1 cells 2				66.67
6	False Path	false	false	1 ports		1 cells 1		100.00		33.33
7	False Path	false	false			1 ports 1				100.00
8	False Path	false	false			1 cells 2				66.67
9	False Path	false	false	1 ports		1 ports 1		100.00		100.00
10	Max Delay	max=5	-			1 ports 1				100.00
11	Min Delay	-	min=3	1 cells		1 cells 1		20.00		25.00
12	Max Delay	max=5	-			1 ports 1				100.00
13	Multicycle Path	-	cycles=1			1 cells 2				66.67
14	Multicycle Path	cycles=2	-			1 ports 1				100.00
15	False Path	false	false	1 cells		1 cells 0		0.00		0.00

Warning: the percentages reported indicate the number of pins covered by the exception relative to the number of pins specified implicitly or explicitly.

当时序例外的起点和端点之间不存在路径时，覆盖范围报告显示 0.0。在上述示例中，时序例外位置 15 不存在时序路径。这与来自 `report_exceptions -ignored` 结果（约束位置 15 报告为不存在的路径）不匹配。

覆盖率报告有助于编写有效的时序例外。下图显示了以下 `set_multicycle_path` 约束的覆盖范围报告的另一个示例：

```
set_multicycle_path -setup 2 -from [all_registers] -to [get_cells
cpuEngine/or1200_cpu/or1200_ctrl/ex_insn_reg[*]]
```

图 49：多周期路径覆盖范围

Exceptions Report										
Position	Type	Setup	Hold	From	Through	To	Endpoints	From (%)	Through (%)	To (%)
54	Multicycle Path	cycles=2	-	15901 cells		21 cells	63	0.95		100.00

在上图所示示例中，`-from` 选项的覆盖范围针对 `all_registers` 返回的 15901 个单元对象仅为 0.95%。可通过优化 `-from` 选项指定的对象列表，使其中对象仅包含指向单元 `cpuEngine/or1200_cpu/or1200_ctrl/ex_insn_reg[*]` 的路径，即可提高约束效率。

报告已忽略的对象

`Report Exception` 命令可为每个时序例外约束生成无效的起点和端点列表。Vivado 工具会忽略无效的起点和端点，因为时序路径不可能始于这些起点，也不可能止于这些端点。`report_exceptions -ignored_objects` 可报告已忽略的管脚。

注释：含“最大延迟 (Max Delay)” 约束或“最小延迟 (Min Delay)” 约束的无效起点和端点不会被忽略，但会导致路径分段。

注释：在已忽略的对象列表中会报告绑定到 POWER 或 GROUND 的起点或端点管脚。

作为演示，请在小型设计示例上设置如下时序约束：

```
create_clock -period 10.000 -name clk [get_ports clk]
set_false_path -from [get_cells int10_reg] -to [get_cells int20_reg]
set_false_path -from [get_pins int11_reg/*] -to [get_pins int21_reg/*]
```

注释：输入第 2 个“伪路径 (False Path)”约束时，Vivado 工具会生成“Warning Constraints 18-402”警告，因为部分起点和端点无效。

WARNING: [Constraints 18-402] set_false_path: 'int11_reg/CE' is not a valid startpoint.

解决办法：有效的起点包括：主时钟或生成时钟管脚或端口、时序单元的时钟管脚，或者主输入或输入输出 (inout) 端口。请确认您的查询返回的所有对象都包含在此列表中。

- 第 1 个 set_false_path 约束使用 get_cells 命令。Vivado 工具仅使用有效的起点管脚或端点管脚来将单元从 get_cells 转换为管脚。这将确保约束仅引用有效对象。
- 第 2 个 set_false_path 约束使用 get_pins 命令，并对 -from 和 -to 强制使用所有寄存器管脚。这导致 -from 和 -to 包含多个无效管脚。

下图显示了 report_exceptions -ignored_objects 生成的报告。

图 50：已忽略的对象

Exceptions Report				
Position	Exception	Ignored Startpoints	Ignored Endpoints	
3	False Path	int11_reg/Q	int21_reg/Q	
3	False Path	int11_reg/CE	int21_reg/C	
3	False Path	int11_reg/CLR	int21_reg/CE	
3	False Path	int11_reg/D		

导出有效例外

Report Exception 命令可导出时序例外列表。仅支持导出覆盖至少 1 条路径的约束。在 Vivado Design Suite 定时器 (Timer) 内存中展开用于指定时序例外的模式时，仅导出有效起点和端点管脚。此报告可配合覆盖范围报告一起使用，以帮助优化用于定义时序例外的对象的模式以及集合。

注释：不导出时序约束 set_clock_group 和 set_bus_skew。

下图显示了[报告已忽略的对象](#)章节中所述的 2 个伪路径 (False Path) 约束上的 report_exceptions -write_valid_exceptions。

图 51：有效例外

```
# position: 2
set_false_path \
-from [get_pins {int10_reg/C}] \
-to [get_pins {int20_reg/D}]

# position: 3
set_false_path \
-from [get_pins {int11_reg/C}] \
-to [get_pins {int21_reg/D}]
```

导出合并例外

“Report Exception” 命令可导出 STA 引擎发现的时序例外列表。Vivado 时序引擎在内部合并时序例外，以减少内存和运行时间。如果合并的时序例外数量与针对设计指定的时序例外数量不同，则表示时序例外定义可能未最优化。合并的时序例外使用 `report_exceptions -write_merged_exceptions` 来报告。

注释：不导出时序约束 `set_clock_group` 和 `set_bus_skew`。

注释：导出合并的时序例外时，不会筛选掉无效的起点和端点。

下图显示了 2 条伪路径上的 `report_exceptions -write_merged_exceptions`，如[报告已忽略的对象](#)部分中所述。第 2 条伪路径包含所有寄存器管脚，因为 `get_pins` 命令的 `-from/-to` 模式为 `int21_reg/*`。

图 52：合并例外

```
set_false_path \
    -from [get_pins {int10_reg/C}] \
    -to [list [get_pins {int20_reg/CE}] \
            [get_pins {int20_reg/CLR}] \
            [get_pins {int20_reg/D}]]]

set_false_path \
    -from [list [get_pins {int11_reg/C}] \
            [get_pins {int11_reg/CE}] \
            [get_pins {int11_reg/CLR}] \
            [get_pins {int11_reg/D}] \
            [get_pins {int11_reg/Q}]] \
    -to [list [get_pins {int21_reg/C}] \
            [get_pins {int21_reg/CE}] \
            [get_pins {int21_reg/CLR}] \
            [get_pins {int21_reg/D}] \
            [get_pins {int21_reg/Q}]]]
```

Vivado IDE 中的例外报告

“例外报告 (Report Exceptions)” 对话框

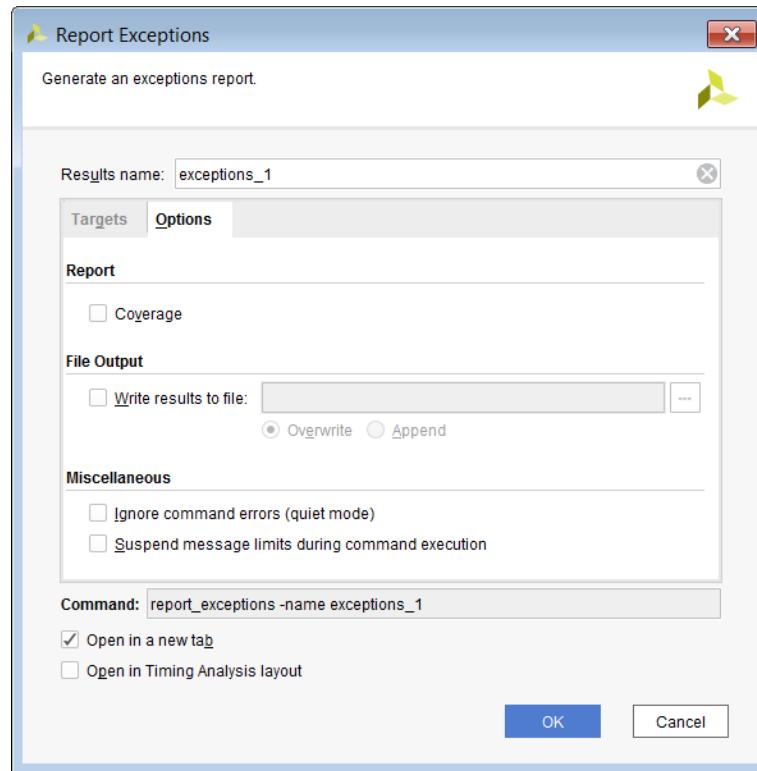
在 Vivado® IDE 中，选择 “Reports” → “Timing” → “Report Exceptions” 以打开 “Report Exceptions” 对话框。

从 Report Exceptions GUI 生成的报告在单次运行中可合并多个表。通常需使用不同命令行选项来多次运行 `report_exceptions` 才能生成此类报告。因此，通过 GUI 运行 Report Exceptions 的运行时间可能比通过 Tcl 控制台运行 `report_exceptions` 的时间更长。

“例外报告 (Report Exceptions)” 对话框：“选项 (Options)” 选项卡

“Report Exceptions” 对话框中的 “Options” 选项卡如下图所示。

图 53：“例外报告 (Report Exceptions)”对话框：“选项 (Options)”选项卡



“Report Exceptions”对话框中的“Options”选项卡包含以下几个部分：

“报告 (Report)”部分

- 覆盖范围 (Coverage): 通过在详情表内附加的列来报告时序例外覆盖范围。

注释：此选项可能会导致运行时间显著增加。

等效的 Tcl 选项：-coverage

“文件输出 (File Output)”部分

- 将结果写入文件 (Write results to file): 将结果写入指定文件。默认情况下，报告将写入 Vivado IDE 的“时序 (Timing)”窗口。

等效的 Tcl 选项：-file

- “覆盖 (Overwrite)”或“追加 (Append)”: 当报告写入文件时，这 2 个选项可用于确定是覆盖指定文件还是向现有报告追加新信息。

等效的 Tcl 选项：-append

“杂项 (Miscellaneous)”部分

- 忽略命令错误 (Ignore command errors): 以静默方式执行命令，忽略所有命令行错误，不返回任何消息。此命令还会返回 TCL_OK，忽略执行期间遇到的所有错误。

等效的 Tcl 选项：-quiet

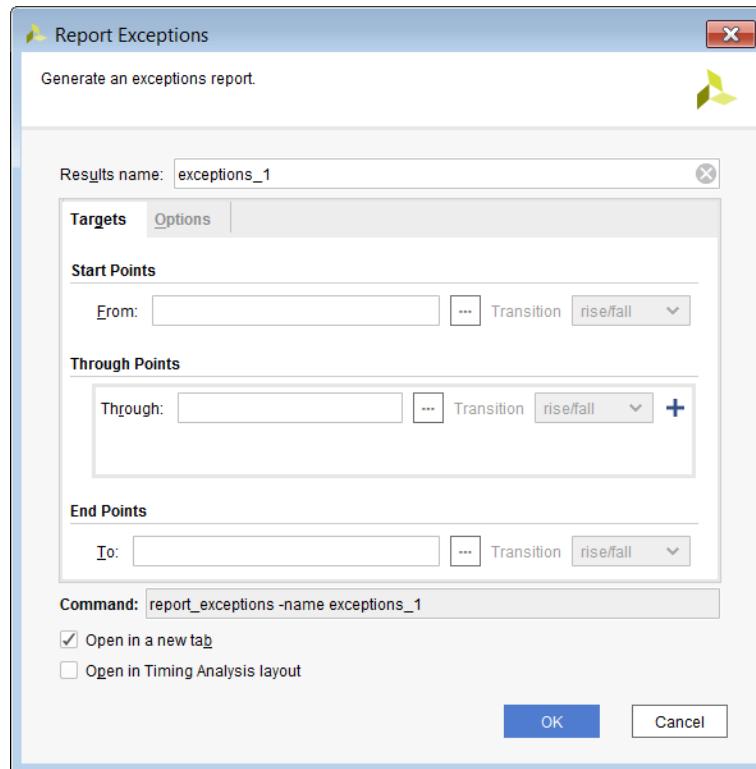
- 命令执行期间暂挂消息限制 (Suspend message limits during command execution)：临时覆盖所有消息限制并返回所有消息。

等效的 Tcl 选项：`-verbose`

“例外报告 (Report Exceptions)”：“目标 (Targets)” 选项卡

“Report Exceptions”对话框中的“Targets”选项卡如下图所示。

图 54：“例外报告 (Report Exceptions)”：“目标 (Targets)” 选项卡



“Report Exceptions”提供了多个筛选选项，可用于报告特定的时序例外或时序例外组：

- “起点 (Start Points)”：请参阅[“目标 \(Targets\)” 选项卡](#)。
- “穿越点 (Through Points)”：请参阅[“目标 \(Targets\)” 选项卡](#)。
- “端点 (End Points)”：请参阅[“目标 \(Targets\)” 选项卡](#)。

使用筛选选项时，仅报告根据这些选项严格定义的时序例外。

“例外报告 (Exceptions Report)” 详细信息

“Exceptions Report”包含以下几个部分：

“常规信息 (General Information)” 部分

“Exceptions Report”的“General Information”部分提供了有关以下内容的信息：

- 设计名称
- 选定的器件、封装和速度等级（带有速度文件版本）
- Vivado Design Suite 版本
- 当前日期
- 为生成报告所执行的等效 Tcl 命令

“汇总 (Summary)” 部分

本节提供了所有时序例外和时钟组约束的汇总信息。对于每一种约束类型，报告将包含有效约束数量、已忽略的约束数量、已忽略的对象数量以及已覆盖的建立和保持端点数量。该表提供的信息比从命令行 (`report_exceptions -summary`) 运行 `report_exceptions` 时提供的汇总表更丰富。

如需获取每一种例外类型的详细信息，可参阅该汇总表中提供的指向“例外”部分或“已忽略的对象”部分的超链接。“有效约束 (Valid Constraints)” 和“已忽略的约束 (Ignored Constraints)” 链接至同一个“例外”详情表。

注释：如果不存在连接 `-from`、`-through` 或 `-to` 的物理路径或者如果约束完全被其它约束所覆盖，则忽略例外。

图 55：Report Exceptions：“Summary” 部分

Exception	Valid Constraints	Ignored Constraints	Ignored Objects	Number of Setup Endpoints	Number of Hold Endpoints
False Path	22	11	157	6,604	6,530
Clock Groups	7	0	0	15,420	15,420
Multicycle Path	4	0	0	28	28
Max Delay	0	0	0	0	0
Max Delay Data Path Only	3	14	0	734	0
Min Delay	0	0	0	0	0

“例外 (Exceptions)” 部分

此部分可支持访问每个时序例外的详情表。针对每种类型的时序例外都有 1 个对应类别，并且这些类别具有源自“汇总 (Summary)” 表的超链接。详情表的格式取决于在 GUI 中是否已选中“覆盖率 (Coverage)” 选项。

以下是未选中“Coverage”选项的详情表示例。

图56：Report Exceptions：未选中“Coverage”的详情表

The screenshot shows the 'Timing' tab of the Report Exceptions window. On the left, a tree view lists categories like 'Exceptions' (with 'False Path (33)' selected), 'Clock Groups (7)', 'Multicycle Path (4)', etc. The main area displays a table titled 'False Path' with columns: Position, Setup, Hold, From, Through, To, and Status. The table contains 38 rows of data, mostly showing invalid endpoints or non-existent paths.

	Position	Setup	Hold	From	Through	To	Status
	19	false	false		-through [get_pins -hierarchical ...	[get_pins -hierarchical ...	Invalid endpoint
	20	false	false	[get_cells -hierarchical...			
	23	false	false		-through [get_pins -hierarchical ...	[get_pins -hierarchical ...	Invalid endpoint
	24	false	false	[get_cells -hierarchical...			
	35	false	false			[get_pins -hierarchical ...	Invalid endpoint
	36	false	-	[get_clocks CLK_pc...		[get_clocks CLK_dclk...	
	37	false	-	[get_clocks CLK_pc...		[get_clocks CLK_dclk...	
	38	false	-	[get_clocks CLK_pc...		[get_clocks CLK_dclk...	Non-existent path

表格中的“位置 (Position)”列表示与 Timing Constraints Editor (TCE) 所报告的位置编号相匹配的时序约束位置编号。您可双击任一行以重定向至 TCE 内选中的约束。或者也可以右键单击该行并从弹出菜单中选择“查看约束 (View Constraint)”。

图57：Report Exceptions 上下文菜单

This screenshot is similar to Figure 56, but a context menu has been opened over the second row of the 'False Path' table. The menu items shown are 'View Constraint' (highlighted with a blue box) and 'Export to Spreadsheet...'.

	Position	Setup	Hold	From	Through	To	Status
	19	false	false		-through [get_pins design/...	[get_pins -hierarchical-filt...	Invalid endpoint
	20	false	false	[get_cells -hierarchical -fil...			
	23	false	false		-through [get_pins design/...	[get_pins -hierarchical ...	Invalid endpoint
	24	false	false	[get_cells -hierarchical -fil...			
	35	false	false			[get_pins -hierarchical zvi...	Invalid endpoint
	36	false	-	[get_clocks CLK_pcclk]		[get_clocks CLK_dclk_del...	
	37	false	-	[get_clocks CLK_pcclk]		[get_clocks CLK_dclk_del...	
	38	false	-	[get_clocks CLK_pcclk_free]		[get_clocks CLK_dclk_del...	Non-existent path

下图显示了 Timing Constraint Editor (TCE) 内选中的约束。

图58：Timing Constraint Editor 内的时序例外

The screenshot shows the 'Timing Constraints' tab of the TCE. On the left, a tree view shows 'Exceptions (54)' expanded, with 'Set False Path (33)' selected. The main area displays a table titled 'Set False Path' with columns: Position, Setup/Hold, Rise/Fall, Reset Path, and Start Points. The second row of the table is highlighted with a blue selection bar, indicating it is the currently selected constraint.

	Position	Setup/Hold	Rise/Fall	Reset Path	Start Points
	19			<input type="checkbox"/>	
	20			<input type="checkbox"/>	[get_cells -hierarchical-filter {NAME =~ design/zkprctrl(wrapper/srb_bridge/u_xst_wrapper_0}
	23			<input type="checkbox"/>	
	24			<input type="checkbox"/>	[get_cells -hierarchical-filter {NAME =~ design/zkprctrl(wrapper/srb_bridge/u_xst_wrapper_0}
	35			<input type="checkbox"/>	
	36	setup		<input type="checkbox"/>	[get_clocks CLK_pcclk]
	37	setup		<input type="checkbox"/>	[get_clocks CLK_pcclk]
	38	setup		<input type="checkbox"/>	[get_clocks CLK_pcclk_free]

“From”、“Through”和“To”列可报告用于定义时序例外的原始模式。您还可参考 TCE 中的约束位置编号来查看这些模式。

下图显示了 Report Exception GUI 内选中“Coverage”选项情况下的详情表示例。

图59：Report Exceptions：选中“Coverage”的详情表

Position	Setup	Hold	From	Through	To	Endpoints	From (%)	Through (%)	To (%)	Status
19	false	false		1 pins	14 pins	0		0.000	0.000	Invalid endpoint
20	false	false	5 cells			134	100.000			
23	false	false		1 pins	14 pins	0		0.000	0.000	Invalid endpoint
24	false	false	5 cells			134	100.000			
35	false	false			448 pins	382			85.268	Invalid endpoint
36	false	-	1 clocks		1 clocks	69	100.000		100.000	
37	false	-	1 clocks		1 clocks	2	100.000		100.000	
38	false	-	1 clocks		1 clocks	0	0.000		0.000	Non-existent path

表格中的“Position”列表示上述时序约束位置编号。

选中“Coverage”选项后，表格中的“From”、“Through”和“To”列将包含指向作为时序约束目标的设计对象的超链接。对象可以是单元、信号线、管脚或时钟。可单击蓝色超链接来选择对象。选中对象后，使用“F4”键可打开原理图。此外，覆盖率信息还可在表格中添加以下列以指示覆盖率百分比：“From (%)”、“To (%)”和“Through (%)”。

表格中的“状态 (Status)”列可报告约束状态，例如，“无效端点 (Invalid endpoint)”、“路径被部分覆盖 (Partially overridden path)”、“路径不存在 (Non-existent path)”或者“完全覆盖 (Totally overridden)”。在命令行上运行 report_exception 时，同样可报告这些状态：

- Non-existent path：例外被视为无效（不影响时序分析）。
- Totally overridden：例外被视为无效（不影响时序分析）。

注释：覆盖率按如下顺序计算：“From”、“Through”和“To”。针对某一层次计算的覆盖率取决于上一层次。当给定层次计算所得覆盖率为 0% 时，所有后续层次都继承此覆盖率 0%。

注释：覆盖率 0% 的约束可视作为无效，因为它不影响时序分析。

注释：绑定到 VCC/GND 的管脚将报告为无效管脚。

时钟组并非是由 -from、-through 和 -to 定义的，因此详情表与此不同。

图60：时钟组的详情表

Position	Clock Group1	Clock Group2	Status
442	[get_clocks { SFS40CLK_0 }]	[get_clocks { SFCLK_0 }]	
443	[get_clocks { RFSCLK }]	[get_clocks { LB_MD1SFCLK }]	
443	[get_clocks { LB_MD1SFCLK }]	[get_clocks { RFSCLK }]	
444	[get_clocks { RFS40CLK_0 }]	[get_clocks { A2YCKP_DIV_64 IA...]	Non-existent path
444	[get_clocks { MODCLK_0 MODC...]	[get_clocks { A2YCKP_DIV_64 IA...]	Non-existent path
444	[get_clocks { _ETH_REF125MC...]	[get_clocks { A2YCKP_DIV_64 IA...]	Non-existent path

当时钟组约束涉及多个组并且每个组都具有多个时钟时，该表包含所有可能的时钟对组合，每个时钟对独立显示一行。在此情况下，约束将跨多个行，并且其中每一行都引用同一个约束位置编号。

以上设计中的约束位置编号 443 定义为：

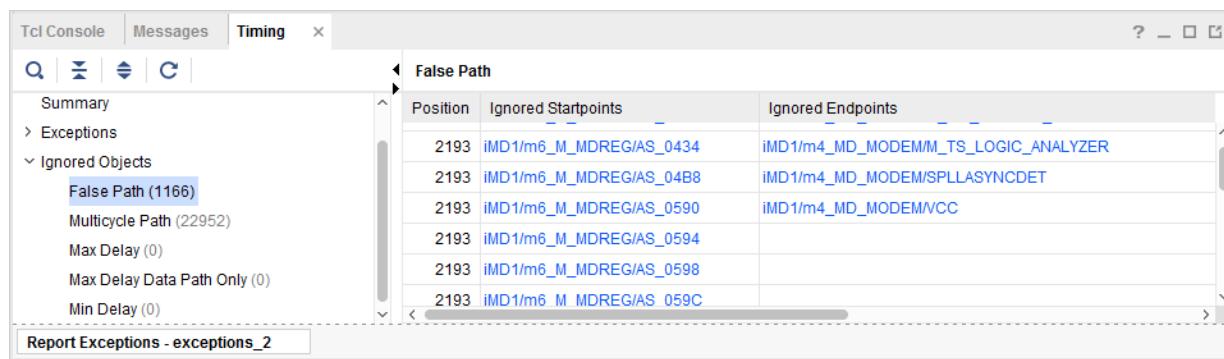
```
set_clock_groups -physically_exclusive -group RFCLK -group LB_M1SFCLK
```

约束跨 2 行，因为从时钟 RFCLK 到时钟 LB_M1SFCLK 之间存在部分时序路径，而在时钟 LB_M1SFCLK 到时钟 RFCLK 之间同样存在部分时序路径。

“已忽略的对象 (Ignored Objects)” 部分

这部分用于报告已忽略的起点和端点，按约束类型组织。这等同于从 Tcl 控制台运行 `report_exceptions -ignored_objects`。

图 61：“Report Exceptions” 对话框：“Ignored Objects” 部分



表中的“Position”列表示与 TCE 内部报告的时序约束位置编号相匹配的位置编号。您可双击任一行以重定向到 TCE 内选定的约束。或者也可以右键单击该行并从弹出菜单中选择“查看约束 (View Constraint)”。

表中的“已忽略的起点 (Ignored Startpoints)”列和“已忽略的端点 (Ignored Endpoints)”列用于报告已忽略的管脚。根据指定的管脚模式为 `-from` 或 `-to`，如果管脚不是有效起点或端点，则忽略该管脚。任一约束均可跨多个行，具体取决于报告的管脚数量。请使用超链接选择设计对象。完成选择后，可在“Property”页面中复查属性，并可按“F4”键打开原理图。

时钟域交汇报告 (Report Clock Domain Crossings)

“时钟域交汇 (Clock Domain Crossings, CDC)” 报告可对设计中的时钟域交汇执行结构分析。此信息可用于识别潜在不安全的 CDC，此类 CDC 可能导致亚稳态或数据一致性问题。虽然 CDC 报告与“时钟交互 (Clock Interaction)” 报告类似，但 CDC 报告侧重于结构及其时序约束，不提供有关时序裕量的信息。

从 Tcl 控制台运行 CDC 报告时，可使用 `-cells` 选项将此报告限定于 1 个或多个层级单元。如果 CDC 报告已限定范围，那么当层级单元列表中包含源管脚或目标管脚时，将报告发现结果。限定范围选项在“CDC 报告 (Report CDC)” GUI 中不可用。

简介

生成 CDC 报告前，必须确保设计已正确约束，并且未缺失时钟定义。Report CDC 仅分析并报告已定义源时钟和目标时钟的路径。Report CDC 可对如下对象执行结构分析：

- 针对异步时钟间的所有路径。

- 仅针对具有以下时序例外的同步时钟间的路径：
 - 时钟组
 - 伪路径
 - 仅最大延迟数据路径

对于不含此类时序例外的同步时钟路径，CDC 引擎假定此类路径已安全定时且不对其进行分析。Report CDC 运行中不考虑任何信号线延迟或单元延迟。

术语

在时钟域交汇 (CDC) 和时钟间时序分析的上下文中，术语“安全 (safe)”、“不安全 (unsafe)”和“端点 (endpoints)”的含义不尽相同。

在 CDC 上下文中，使用同步电路来防止亚稳态时，异步交汇即为安全。例如，安全的单比特 CDC 可通过同步器实现，即具有相同时钟和控制信号的寄存器链。安全的多比特 CDC 可通过 MUX 保持电路或时钟使能控制的电路来实现。

相反，当 CDC 分析引擎无法识别异步 CDC 路径上已知安全的同步电路时，此 CDC 即为不安全。

针对两个时钟域之间的 CDC 报告的端点数量可能与时序分析命令所报告的端点数量不同。例如，异步复位同步器涉及多个时序路径端点。但是，同步电路作为单一元素来报告，因此计为单一 CDC 端点。同样，多比特 CDC 可包含多个单比特交汇，但报告为单一 CDC 端点。然而，其它时序报告会将该总线报告未多个时序端点。



重要提示！ 由于 `report_clock_interaction` 与 `report_cdc` 用途不同，因此每项命令所报告的端点数量不可比较。在 `report_clock_interaction` 上下文中，安全/不安全表示时序分析引擎提供与硬件中最差情况相匹配的裕量的能力。对于 `report_cdc`，安全/不安全表示设计中实现的 CDC 电路的类型。

运行“时钟域交汇报告 (Report Clock Domain Crossings)”

从 Vivado IDE 运行“Report CDC”时，默认情况下它可提供有关指定时钟之间的 CDC 路径的所有详细信息。当从 Tcl 控制台运行“Report CDC”时，它仅打印“Summary by Clock Pairs”表。您必须指定 `-details` 选项才能像 GUI 模式下一样报告所有详细信息。报告详细信息可能会生成非常长的文件或日志文件。

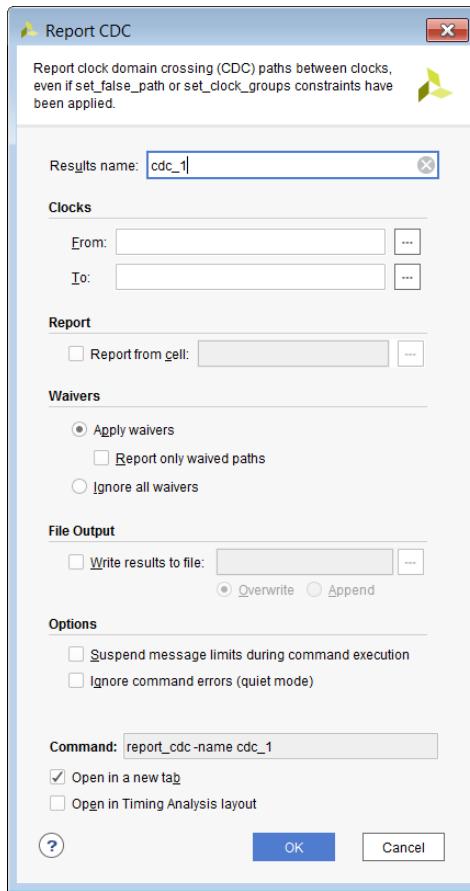
要在 Vivado IDE 中运行“Report Clock Domain Crossings”，请选择“Reports”→“Timing”→“Report CDC”。

等效的 Tcl 命令：`report_cdc -name cdc_1`

在 Vivado IDE 中，“Report CDC”对话框包含以下字段，如下图所示：

- “结果名称 (Results Name)” 字段
- “时钟 (Clocks)” 字段 (From/To)
- “文件输出 (File Output)” 字段
- “选项 (Options)” 字段

图 62：“Report CDC” 对话框



“结果名称 (Results Name)” 字段

在“时钟域交汇报告 (Report Clock Domain Crossings)”对话框顶部的“结果名称 (Results Name)”字段中指定报告的图形窗口名称。

等效的 Tcl 选项：`-name <windowName>`

“时钟 (Clocks)” 字段 (From/To)

“Clocks”下的“目标 (To)”和“源 (From)”字段支持您选择源时钟和/或目标时钟以便在其中运行 CDC 分析。您可使用“From/To”选项来控制“Report CDC”，将其限定于特定时钟并生成更通俗易懂的报告。单击右侧“浏览 (Browse)”按钮 可打开搜索对话框，以便查找时钟对象。

等效的 Tcl 选项：`-from <clockNames> -to <clockNames>`

“文件输出 (File Output)” 字段

“File Output”字段支持您选择指定用于写入结果的文件。您可以覆盖该文件或追加到该文件。

等效的 Tcl 选项：`-file <fileName> -append`

“选项 (Options)” 字段

“Options” 字段支持您执行以下操作：

- 命令执行期间暂挂消息限制
等效的 Tcl 选项：`-verbose`
- 忽略命令错误
等效的 Tcl 选项：`-quiet`

理解时钟域交汇报告规则

“Report CDC” 尝试将每条 CDC 路径与 1 个已知 CDC 拓扑结构相匹配。每个 CDC 拓扑结构都与 1 项或多项 CDC 规则相关联，如表 3：CDC 规则与描述中所示。请注意，您无法修改规则的严重性，就像 DRC 和消息一样。在 [CDC 拓扑的简化原理图](#) 中包含检测到的 CDC 拓扑结构的简化原理图和描述。

CDC 拓扑结构根据某些优先级规则来进行分析。[表 4：CDC 规则与优先级（从高到低）](#) 按优先级从高到低的顺序显示 CDC 规则。默认情况下，每个端点最多仅报告 1 项 CDC 违例，并且如果在特定端点上存在多项违例，则报告优先级最高的 CDC 规则，并屏蔽所有优先级更低 CDC 违例。

注释：使用命令行选项 `-all_checks_per_endpoint` 可覆盖此默认行为。此选项会强制工具报告应用于端点的所有“参考 (Info)”、“警告 (Warning)” 和“严重警告 (Critical)” 检查，而不考虑规则优先级。

表 3：CDC 规则与描述

CDC 拓扑结构	CDC 规则	严重性	描述
单比特 CDC	CDC-1	Critical	单比特 CDC 路径未同步或者具有未知的 CDC 电路。
	CDC-2	Warning	单比特 CDC 路径已与 2 个以上的阶段同步器同步，但全部或部分同步器触发器上缺失 <code>ASYNC_REG</code> 属性。
	CDC-3	Info	单比特 CDC 路径已与 2 个以上阶段同步器同步，并且存在 <code>ASYNC_REG</code> 属性。
多比特 CDC	CDC-4	Critical	多比特总线 CDC 路径未同步，或者具有未知的 CDC 电路。
	CDC-5	Warning	多比特总线 CDC 路径已与 2 个以上阶段同步器同步，但全部或部分同步器触发器上缺失 <code>ASYNC_REG</code> 属性。
	CDC-6	Warning	多比特总线 CDC 路径已与 2 个以上阶段同步器同步，并且存在 <code>ASYNC_REG</code> 属性。
异步复位	CDC-7	Critical	1 个异步信号（清除或预置）不同步或具有未知的 CDC 电路。
	CDC-8	Warning	1 个异步信号（清除或预置）已同步，但在全部或部分同步器触发器上缺失 <code>ASYNC_REG</code> 属性。
	CDC-9	Info	1 个异步复位已同步，并且存在 <code>ASYNC_REG</code> 属性。
组合逻辑	CDC-10	Critical	在同步电路扇入中已检测到组合逻辑。
扇出	CDC-11	Critical	在同步电路之前已检测到删除。
多时钟扇入	CDC-12	Critical	在同步电路扇入中已找到来自多个时钟的数据。
非 FD 原语	CDC-13	Critical	在非 FD 原语上已监测到 CDC。
CE 控制的 CDC	CDC-15	Warning	时钟使能控制的 CDC。
Mux 控制的 CDC	CDC-16	Warning	多路复用器控制的 CDC。
Mux 数据保持 CDC	CDC-17	Warning	多路复用器数据保持 CDC。
HARD_SYNC 原语	CDC-18	Info	1 个信号已与 <code>HARD_SYNC</code> 原语同步。

表 3：CDC 规则与描述（续）

CDC 拓扑结构	CDC 规则	严重性	描述
LUTRAM-to-FD CDC	CDC-26	Warning	LUTRAM 读取/写入存在潜在冲突。

表 4：CDC 规则与优先级（从高到低）

CDC 拓扑结构	CDC 规则
HARD_SYNC 原语	CDC-18
非 FD 原语	CDC-13
Mux 数据保持 CDC	CDC-17
Mux 控制的 CDC	CDC-16
CE 控制的 CDC	CDC-15
LUTRAM-to-FD CDC	CDC-26
异步复位	CDC-7
单比特 CDC 未同步	CDC-1
多比特 CDC 未同步	CDC-4
多时钟扇入	CDC-12
组合逻辑	CDC-10
扇出	CDC-11
异步复位已同步并包含属性	CDC-9
单比特 CDC 已同步并包含属性	CDC-3
多比特 CDC 已同步并包含属性	CDC-6
异步复位已同步，不含属性	CDC-8
单比特 CDC 已同步，不含属性	CDC-2
多比特 CDC 已同步，不含属性	CDC-5

复查时钟域交汇报告的各部分内容

在 GUI 模式下， 默认将生成三个部分：

- [按时钟对汇总 \(Summary by Clock Pair\)](#)
- [按类型汇总 \(Summary by Type\)](#)
- [详细报告 \(Detailed Report\)](#)

各汇总部分提供了需复查并且可能需更改设计的问题综述。这些部分可用于浏览严重性最高的违例，在“详细报告 (Detailed Report)”部分中包含了相关附加信息。

注释：默认情况下，以文本模式运行报告时，仅生成“按时钟对汇总 (Summary by clock pair)”部分。

按时钟对汇总 (Summary by Clock Pair)

在“Summary (by clock pair)”部分中，提供了有关 2 个时钟之间的 CDC 路径数量以及这些路径中找到的最严重问题的严重性的实用信息。该表包含以下几个列：

- 严重性 (Severity): 报告往来列示的时钟的所有 CDC 路径的最高严重性问题。值包括：参考 (Info)、警告 (Warning) 或严重 (Critical)。
- 源时钟 (Source Clock): 显示 CDC 源时钟的名称。
- 目标时钟 (Destination Clock): 显示 CDC 目标时钟的名称。
- CDC 类型 (CDC Type): 反映 2 个时钟之间的关系及其主要的时序例外（如果有）。类型包括：
 - 已安全定时 (Safely Timed): 所有 CDC 路径都已安全定时，因为时钟已同步，且时序例外并未妨碍准确定时。
 - 用户已忽略 (User Ignored): `set_false_path` 或 `set_clock_groups` 已涵盖所有 CDC 路径。
 - 无公共基准时钟 (No Common Primary Clock): CDC 时钟处于异步状态，在 2 个不含公共基准时钟的时钟间至少 1 条 CDC 路径已正常定时。
 - 无公共周期 (No Common Period): CDC 时钟处于异步状态，在 2 个不含公共周期的时钟之间至少 1 条 CDC 路径已正常定时。如需了解无公共周期的时钟的定义，请参阅[了解时序分析的基础知识](#)。
 - 无公共相位 (No Common Phase): CDC 时钟处于异步状态，因为 2 个时钟之间不存在已知的相位关系。
- 例外 (Exceptions): 应用于 CDC 的时序例外（如果有）包括：
 - 无 (None): CDC 路径上不存在任何时钟组 (Clock Group)、伪路径 (False Path) 或仅最大延迟数据路径 (Max Delay Datapath Only) 时序例外。`report_cdc` 不报告其它时序路径，如多周期路径 (Multicycle Paths)、最小延迟 (Min Delay) 和最大延迟 (Max Delay) 等。
 - 异步时钟组 (Asynch Clock Groups): `set_clock_groups -asynchronous` 例外已应用于 CDC 时钟。
 - 专属时钟组 (Exclusive Clock Groups): `set_clock_groups -exclusive` 例外已应用于 CDC 时钟。
 - 伪路径 (False Path): `set_false_path` 例外已应用于 CDC 源时钟或目标时钟上，或者已应用于所有 CDC 路径上。
 - 仅最大延迟数据路径 (Max Delay Datapath Only): `set_max_delay -datapath_only` 例外已应用于所有 CDC 路径。请注意，当 `set_max_delay -datapath_only` 仅涵盖至少 1 条 CDC 路径，而所有其它 CDC 路径均已因 `set_false_path` 约束而被忽略时，才报告“Max Delay Datapath Only”。
 - 部分例外 (Partial Exceptions): `set_false_path` 约束和 `set_max_delay -datapath_only` 约束已混合应用于部分 CDC 路径，并且至少 1 条 CDC 路径已正常定时。
- 端点 (Endpoints): CDC 路径端点总数。这是处于“安全 (Safe)”、“不安全 (Unsafe)”和“未知 (Unknown)”状态的端点总和。在此情况下，端点属于时序单元输入数据管脚。根据 D、CE 和 SET/RESET/CLEAR/PRESET 连接，FD 单元可多次反复计数。对于某些 CDC 拓扑结构，虽然有多条路径能够有效跨越时钟域边界以到达 CDC 结构，但端点数量仅计为 1。例如，在异步复位同步器中，有多个 CLEAR 管脚连接到交汇信号线，但仅对同步器链的首个管脚进行计数。
- 安全 (Safe): 处于安全状态的 CDC 路径端点数量。安全的端点即 CDC 路径上具有如下标识的端点：
 - 具有已知安全的 CDC 结构的异步时钟
 - 具有例外和已知安全的 CDC 结构的同步时钟
 - 不含任何已安全定时的例外的同步时钟，与 CDC 结构无关
 - CDC 已与 HARD_SYNC 宏同步
- 不安全 (Unsafe): 已识别为具有不安全结构的 CDC 路径端点数量。不安全的端点包括 CDC-10、CDC-11、CDC-12 和 CDC-13。
 - 组合逻辑拓扑结构
 - 扇出拓扑结构
 - 多时钟扇入拓扑结构

- 非 FD 原语拓扑结构
- 未知 (Unknown): 处于未知状态的 CDC 路径端点的数量。在这些端点上没有任何 CDC 结构可供匹配，或者已检测到未知 CDC 电路 (CDC-1、CDC-4 和 CDC-7)。
- 无 ASYNC_REG (No ASYNC_REG): 已识别具有如下特征的同步器的数量：在单元链上的前 2 个 FD 单元中至少 1 个 FD 单元上缺失 ASYNC_REG 属性。

下图显示了“Summary (by clock pair)”部分的示例。

图 63: “Summary (by clock pair)” 部分

Severity	Source Clock	Destination Clock	CDC Type	Exceptions	Endpoints	Safe	Unsafe	Unknown	No ASYNC_REG
! Critical	input port clock	gt0_txusrclk_i	No Common Primary Clock	False Path	2	0	0	2	0
! Critical	input port clock	gt2_txusrclk_i	No Common Primary Clock	False Path	2	0	0	2	0
! Critical	input port clock	gt4_txusrclk_i	No Common Primary Clock	False Path	2	0	0	2	0
! Critical	input port clock	gt6_txusrclk_i	No Common Primary Clock	False Path	2	0	0	2	0
! Info	fttClk_0	cpuClk_5	Safely Timed	None	61	61	0	0	0
! Info	phyClk0_2	cpuClk_5	Safely Timed	None	61	61	0	0	0
! Info	phyClk1_1	cpuClk_5	Safely Timed	None	61	61	0	0	0
! Info	usbClk_3	cpuClk_5	Safely Timed	None	3882	3882	0	0	0
! Info	wbClk_4	cpuClk_5	Safely Timed	None	6772	6772	0	0	0
! Info	wbClk_4	fttClk_0	Safely Timed	None	1027	1027	0	0	0
! Info	usbClk_3	phyClk0_2	Safely Timed	None	4451	4451	0	0	0

按类型汇总 (Summary by Type)

“Summary by Type” 表适用于快速查看当前报告中找到的 CDC 结构的性质。示例如下图所示。

图 64: “Summary by Type” 表

Severity	ID	Count	Description
! Critical	CDC-7	8	Asynchronous reset unknown CDC circuitry

“Summary by Type” 表包含以下列：

- 严重性 (Severity): 报告 CDC 规则的严重性，值为：Info、Warning 或 Critical。
- ID: CDC 规则的唯一标识号，详见表 3: CDC 规则与描述。
- 计数 (Count): 整个报告中出现的 CDC 规则数量。
- 描述 (Description): CDC 规则的简短描述。

分析该表时，重要的是从严重性级别最高的项开始。严重性级别分为：

- 严重 (Critical): 此严重性对应于含未知或不安全的 CDC 结构的 CDC 路径。您必须复查每条路径，并通过修改 RTL 来修复结果或者将问题豁免。默认情况下，使用 Vivado IDE 时会生成路径详情，前提是在命令行上将 -details 选项与 report_cdc 命令配合使用。
 - 在交汇信号线上存在一些组合逻辑，或者在交汇信号线的扇入中发现若干源时钟。这可能导致“平均故障间隔时间 (MTBF)”特性劣化。
 - 在交汇信号线上存在通向相同目标时钟域的扇出。这可能导致数据一致性问题。
- 警告 (Warning): 此严重性对应于含已知安全的 CDC 结构的 CDC 路径，但由于以下任一原因，该结构并非理想结构。
 - 前 2 个同步器触发器中至少其一的 ASYNC_REG 属性未设置为 1 (或 true)
 - 所识别的 CDC 结构需要执行 CDC 引擎无法验证的功能性纠正措施。这些结构包括时钟使能控制的 CDC 拓扑、MUX 控制的 CDC 拓扑和 MUX 数据保持时间控制的 CDC 拓扑结构。
- 参考 (Info): 此严重性表明 CDC 结构全部安全且已完成正确约束。

详细报告 (Detailed Report)

通过查看报告中的“CDC 详情 (CDC Details)”部分可查看“CDC 报告 (Report CDC)”详情。您可以使用详细报告来查看所选路径的原理图（按 F4 键）、查看时序报告，或通过右键单击各条目以生成新时序报告。

您可使用时序报告和原理图来复查设计中不符合期望的 CDC 路径、识别错误或缺失的时序例外，以及查找缺失的 ASYNC_REG 属性。“CDC 详情 (CDC Details)”报告示例如下图所示。

图 65: CDC 详情报告

Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Critical	CDC-7	Asynchronous reset unknown CDC circuitry	0	False Path	GTPRESET_IN	mgtEngine/_r_reg/CLR	Unknown
Critical	CDC-7	Asynchronous reset unknown CDC circuitry	0	False Path	GTPRESET_IN	mgtEngine/_r_reg/CLR	Unknown

“CDC 详情报告”表包含以下几个列：

- 严重性 (Severity): 报告 CDC 规则的严重性，值为：Info、Warning 或 Critical。
- ID: CDC 规则的唯一标识号，详见表 3: CDC 规则与描述。
- 描述 (Description): CDC 规则的简短描述。
- 深度 (Depth): 已找到的同步器阶段数量（仅适用于同步器拓扑结构）。
- 例外 (Exception): 应用于 CDC 路径的时序例外。
- 源 (Source (From)): CDC 时序路径起点。
- 目标 (Destination (To)): CDC 时序路径端点。
- 类别 (Category): 显示为安全 (Safe)、不安全 (Unsafe)、未知 (Unknown)。
- 源时钟 (Source Clock (From)): 源时钟名称。
- 目标时钟 (Destination Clock (To)): 目标时钟的名称。

注释：仅当您单击“CDC Details”（“Timing-Report CDC”窗口的左侧列）时，才会显示该列。

注释：仅当您单击“Timing-Report CDC”窗口的左侧列的“CDC Details”时，才会显示该列。



重要提示！ CDC 报告可标记某些赛灵思 IP 中的问题，因为 CDC 引擎无法识别所有可能的 CDC 拓扑结构并且不提供内置豁免机制。如需了解更多信息，请参阅各赛灵思 IP 产品指南。

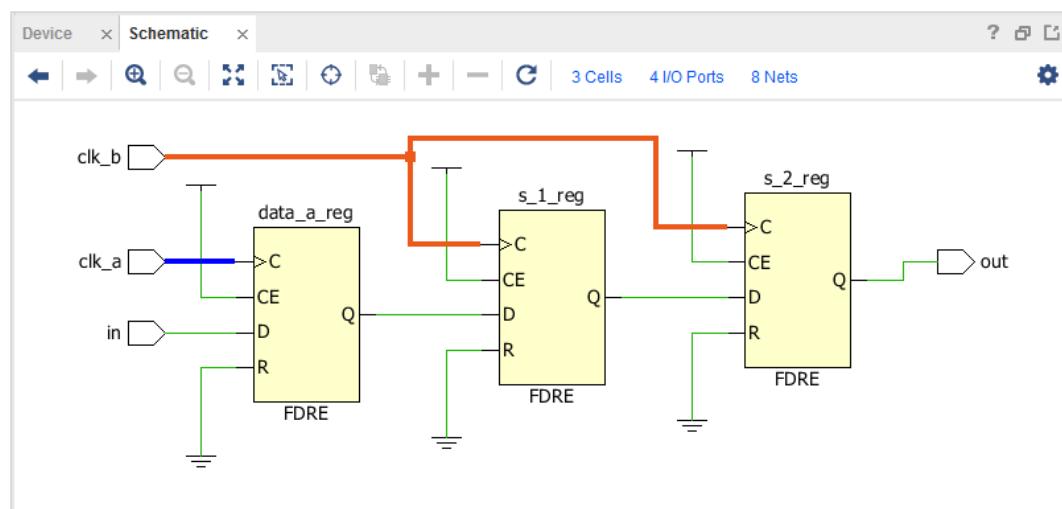
CDC 拓扑的简化原理图

以下部分展示了 CDC 拓扑的简化原理图以及简要说明。在所有原理图中，源时钟信号线（通常为 `clk_a`）以蓝色高亮，目标时钟信号线（通常为 `clk_b`）以橙色高亮。

单比特同步器

下图显示了单比特同步器的简化拓扑结构。ASYNC_REG 属性必须至少设置在同步链的前 2 个触发器上。同步器深度由共享相同控制信号的已链接的触发器数量来定义。

图 66：单比特同步器的简化拓扑结构

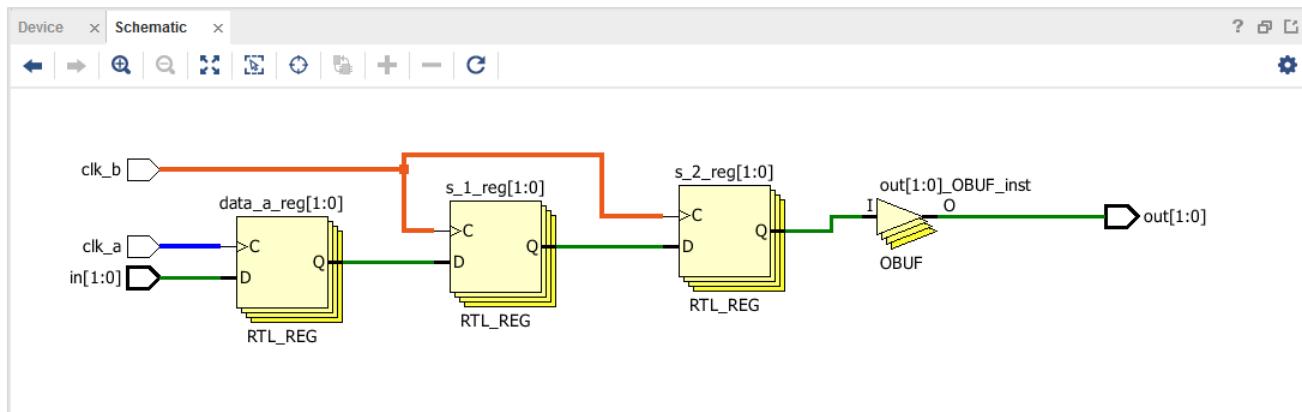


如果触发器的 CLEAR 或 PRESET 管脚同时连接到异步源，那么此同步器在报告中仅显示为单比特同步器而非异步复位同步器。

多比特同步器

检测到的多比特同步器等同于根据起点/端点名称和匹配的 CDC 规则将多个单比特同步器分组在一起。在此情况下，总线由起点和端点单元名称（而不是信号线名称）来定义。标准总线名称格式为 `baseName[index]`。此外，起点与端点索引必须匹配。下图显示了位宽为 2 的多比特同步器示例。

图 67：位宽为 2 的多比特同步器



如果 CDC 总线的某些位与相同 CDC 规则不匹配，那么在报告中该总线将显示为独立的位或总线分段，并具有与相同 CDC 规则匹配的连续索引。

必须明确的是，在总线上采用基于寄存器的同步器并不能确保总线跨域的安全性。因此，CDC 规则 CDC-6 属于“警告 (Warning)”，因为工具无法判定拓扑结构对于设计是否足够。CDC 的安全性由设计人员判定。

如果总线采用格雷编码，那么只要在总线上设置充足的时序约束以确保接收域每次最多只能捕获一项数据，在总线的所有位上使用基于寄存器的同步器的安全性即可得到保证。

如果总线并未采用格雷编码，则应改为使用其它同步器拓扑结构，例如，CD 控制的 CDC 或 MUX 控制的 CDC。

异步复位同步器

在下图中显示了基于 CLEAR 同步的异步复位同步过程，后一张图中显示了基于 PRESET 同步的异步复位同步过程。FF1 单元分别连接到已同步的清除 (CLEAR) 信号或预置 (PRESET) 信号，可根据 `clk_a` 对这两个信号的解除有效进行安全定时。请注意，在异步复位同步器内不得混用含 CLEAR 和 PRESET 的触发器。

图 68：基于 CLEAR 的异步复位同步器

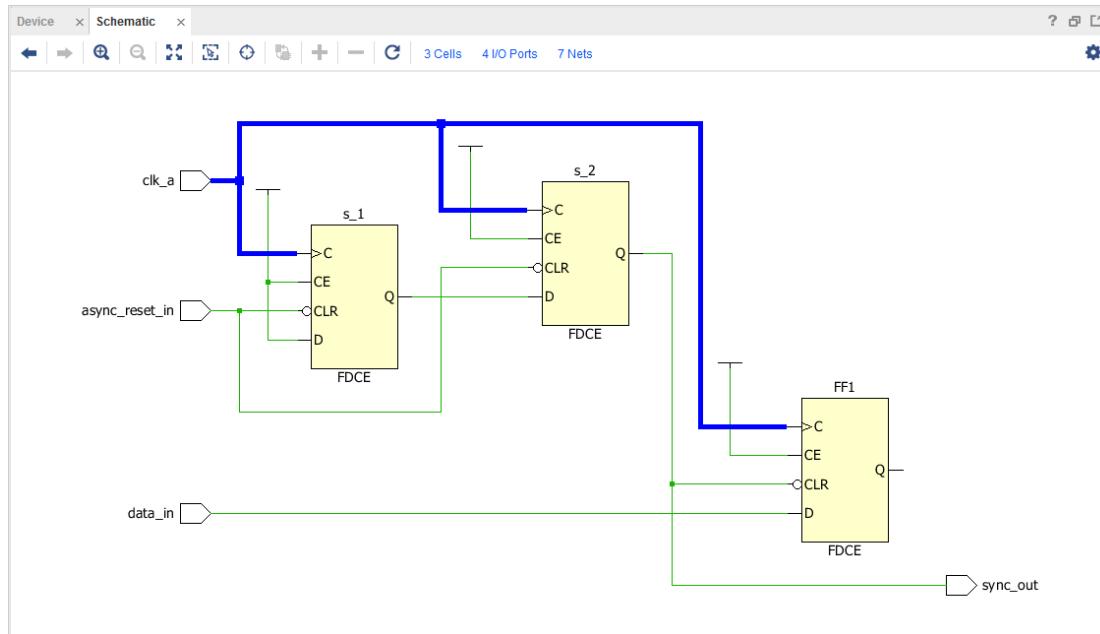
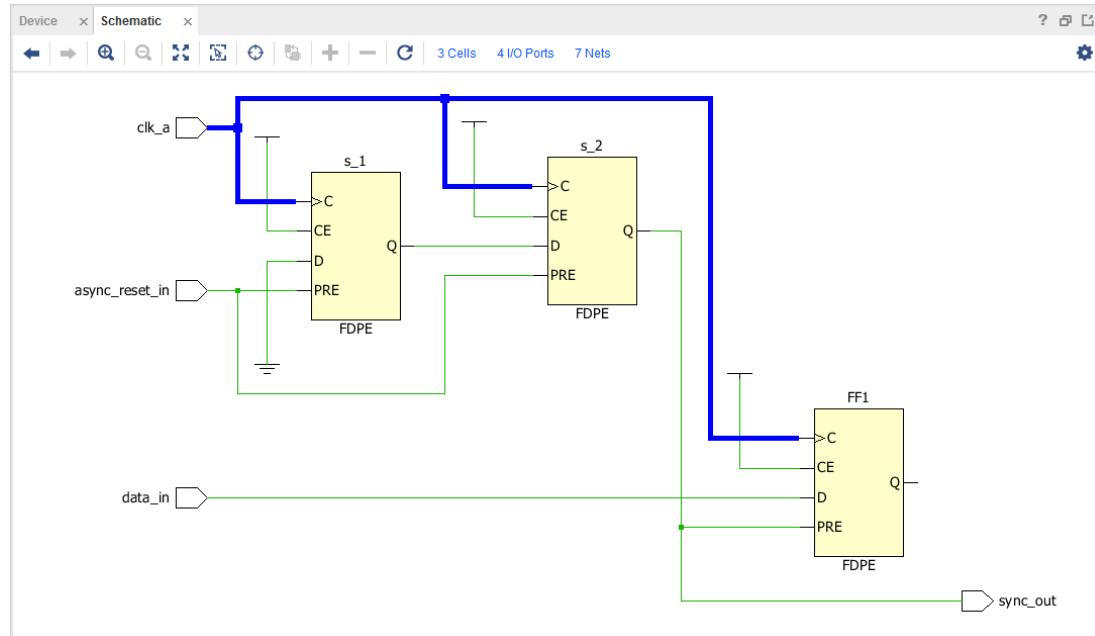


图 69：基于 PRESET 的异步复位同步器



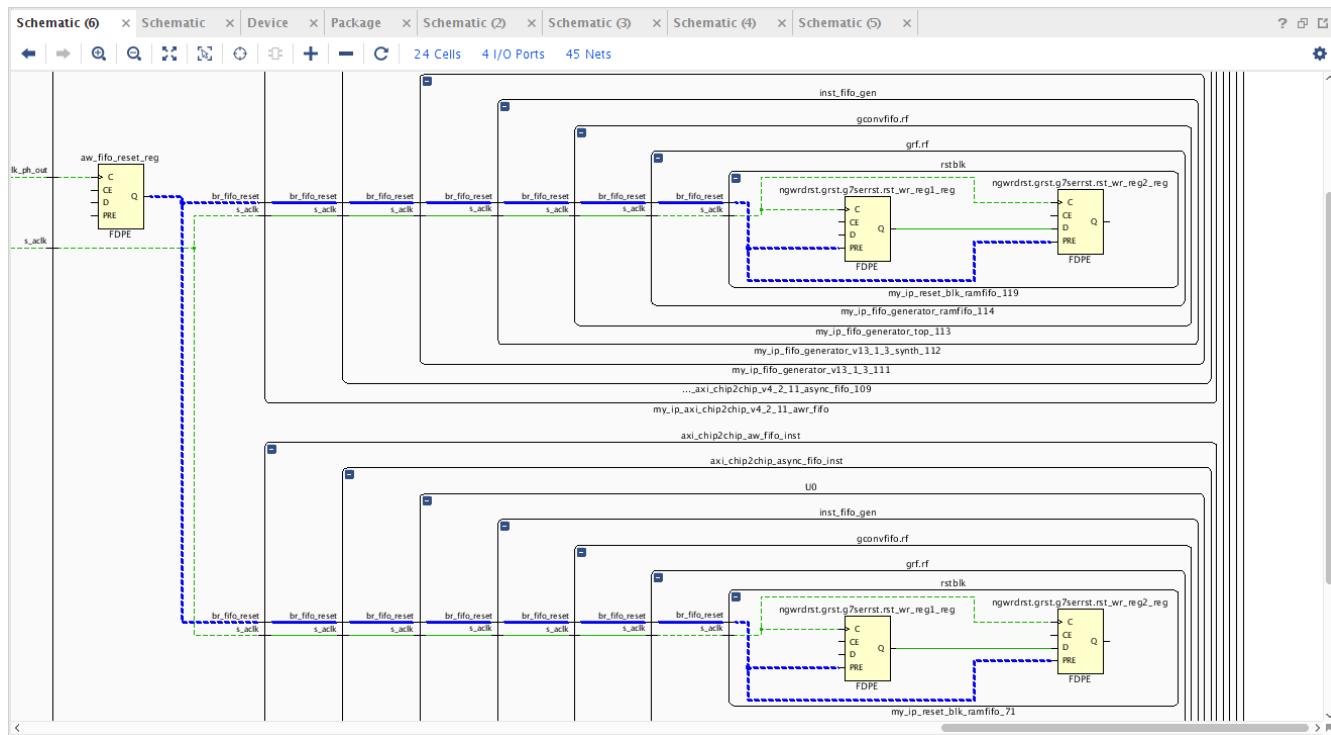
通常建议避免在目标时钟域内包含多个复位信号的同步。这意味着从源时钟域到目标时钟域的复位不应存在任何扇出。此建议可防止目标时钟域在不同时间脱离复位从而导致设计处于未知状态。不遵循此建议会导致从发送触发器到目标时钟出现严重的 CDC-11 扇出违例。

但在某些涉及 FIFO Generator IP 的场景中，可在目标时钟域内安全进行多次复位信号同步。FIFO Generator 将异步进入复位状态，并脱离同步。它会对块 RAM 应用真正的同步复位，但 FIFO 会收到异步复位。只要设计使用逻辑的 wr_rst_busy 信号来保持数据流，就不会出现部分逻辑脱离复位而部分逻辑仍处于复位状态的状况。

AXI 接口使用 5 个 FIFO Generator IP 来同步每个目标时钟域中的复位，这也是构造安全的复位电路的另一个示例。在可放心对复位信号进行多次同步的场景中，可忽略 CDC-11 违例。

下图显示的是安全复位同步的示例，其中同一个目标时钟域中涉及 2 个 FIFO Generator。

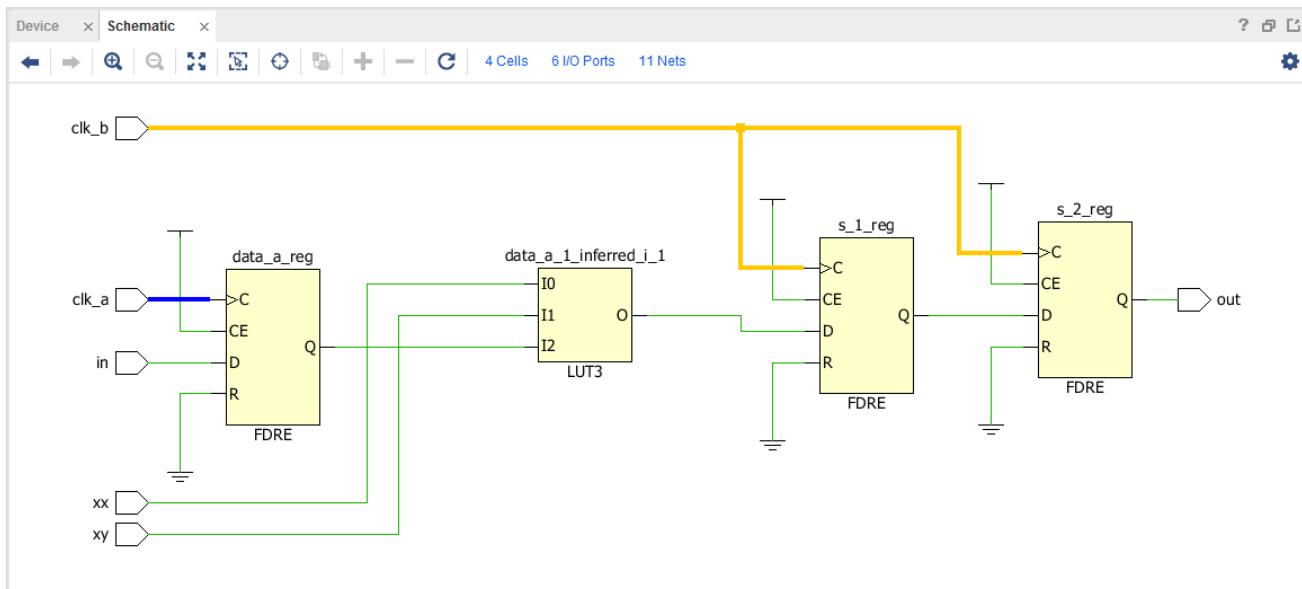
图 70：安全复位同步示例



组合逻辑

在下图所示的组合逻辑简化示例中，在从 `clk_a` 到 `clk_b` 同步器的 CDC 之间布局有 1 个 LUT3 表示的逻辑函数。

图 71：组合逻辑简化示例

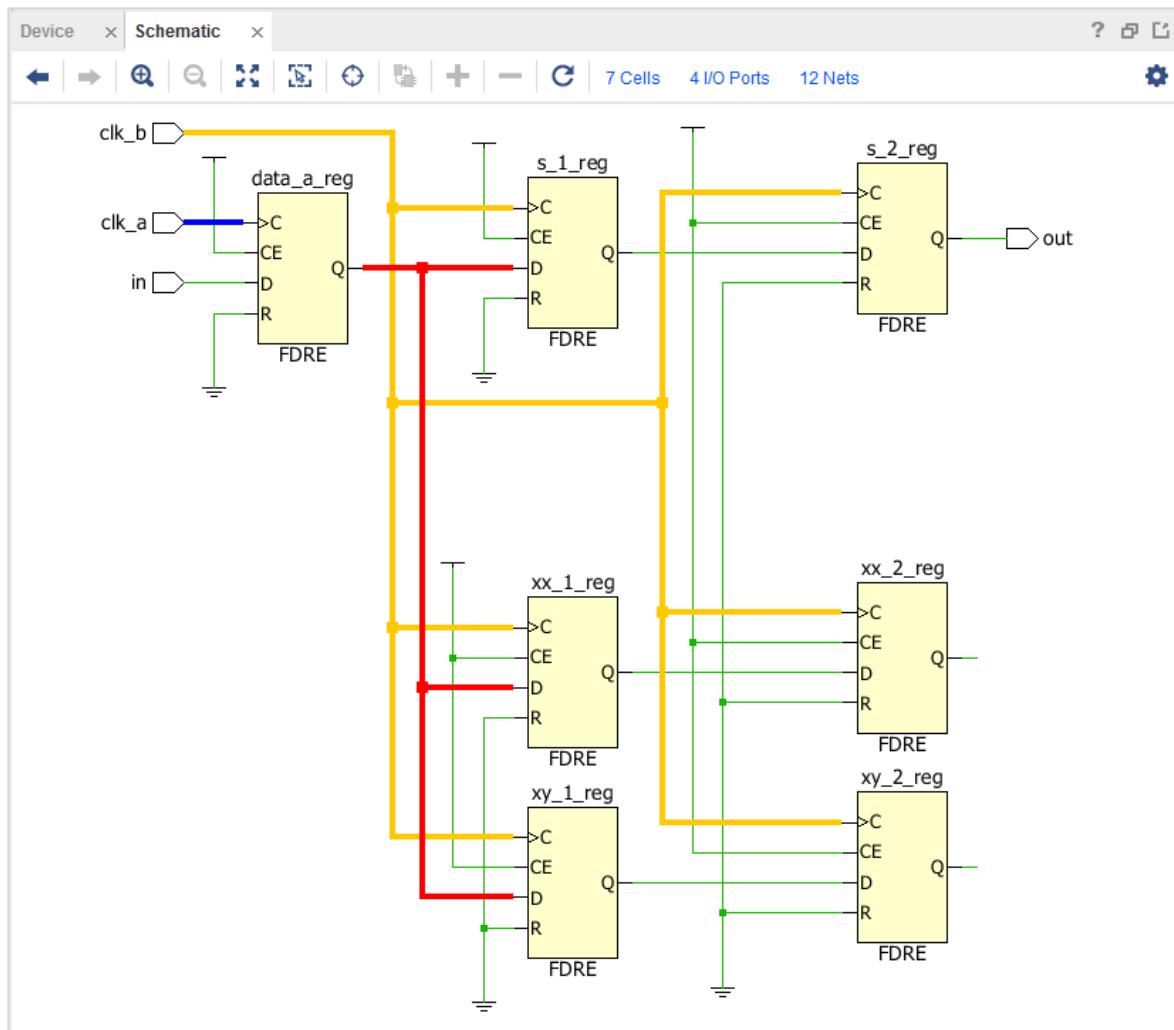


原先不推荐采用此结构，因为在组合逻辑的输出上可能出现毛刺，同步器会捕获此类毛刺并传输至设计下游其余部分。

扇出

在下图所示简化的扇出示例中，源触发器用于驱动在 `clk_b` 域（红色高亮）中同步 3 次的信号线。不建议采用这种结构，因为它可能导致在目标时钟域中出现数据一致性问题，原因在于穿过同步器的时延受到约束，但并不具备周期精确性。

图 72：经简化的扇出示例

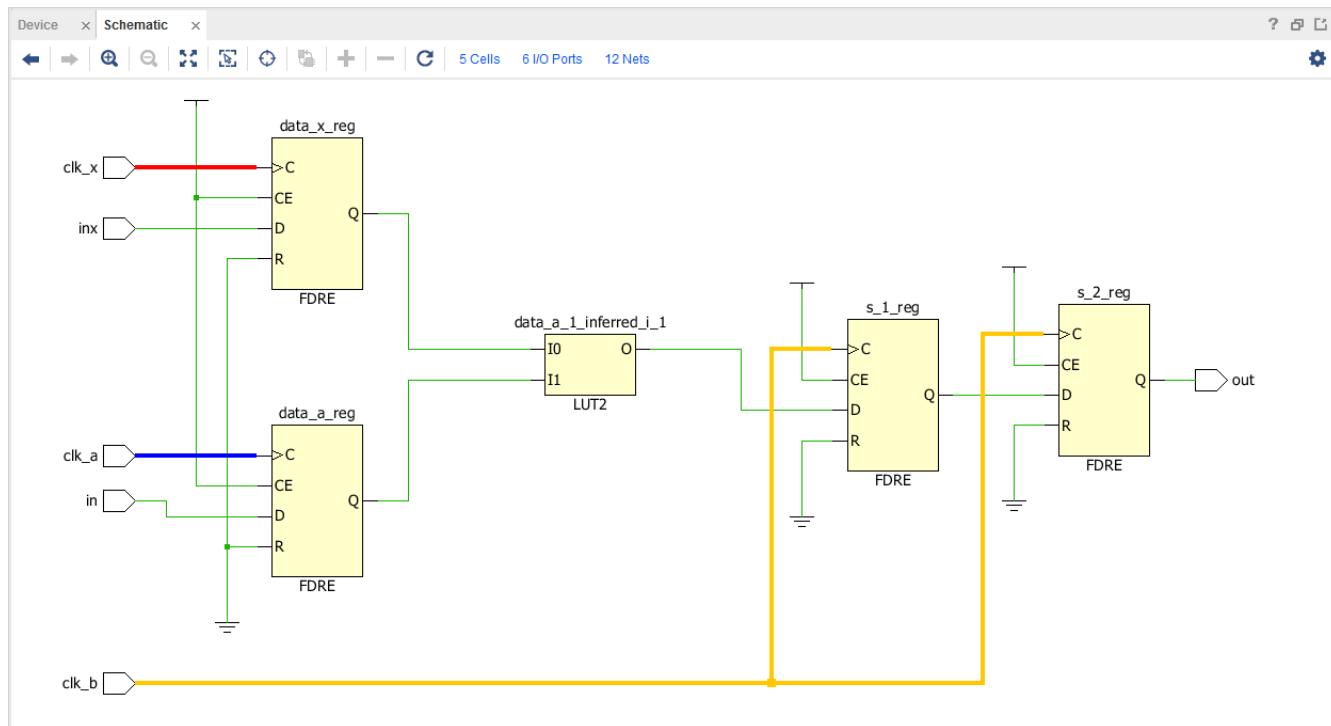


注释：N 到 N 个不同时钟域的扇出并不构成 CDC 问题，不会触发 CDC-11 违例。请参阅[异步复位同步器部分](#)，以获取复位信号上的安全扇出示例。

多时钟扇入 (Multi-Clock Fanin)

在下图所示“多时钟扇入 (Multi-Clock Fanin)”示例中，`clk_a` 与 `clk_x` 正在同时通过组合逻辑 (LUT2) 将数据传输至 `clk_b` 域中的同步器电路。建议首先单独同步来自 `clk_a` 和 `clk_x` 的源数据，然后再通过某些互连逻辑 | FPGA 逻辑将其组合在一起。这样可改善总体 CDC 结构的 MTBF 特性，并且可防止毛刺传输至目标时钟域。

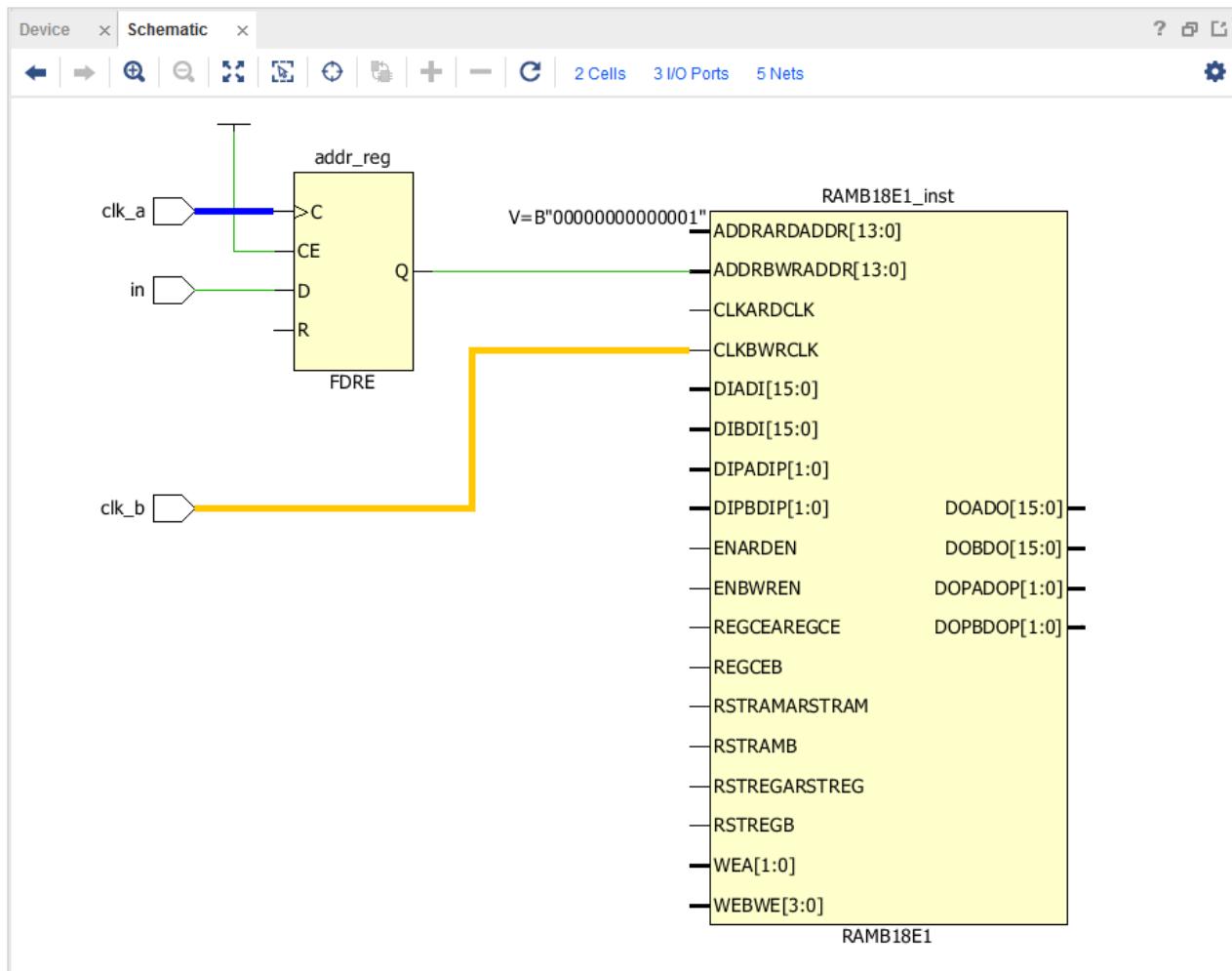
图 73：多时钟扇入示例



非 FD 原语

在下图所示“非 FD 原语”示例中，当 RAMB 原语内部不存在同步逻辑时，在 FDRE 与 RAMB 之间发生 CDC。即使在 RAMB 前插入连接到 `clk_b` 的单阶触发器，受 FDRE 与 RAMB 单元之间的布线距离所限，此同步器仍被视为不足以满足要求。

图 74：非 FD 原语示例

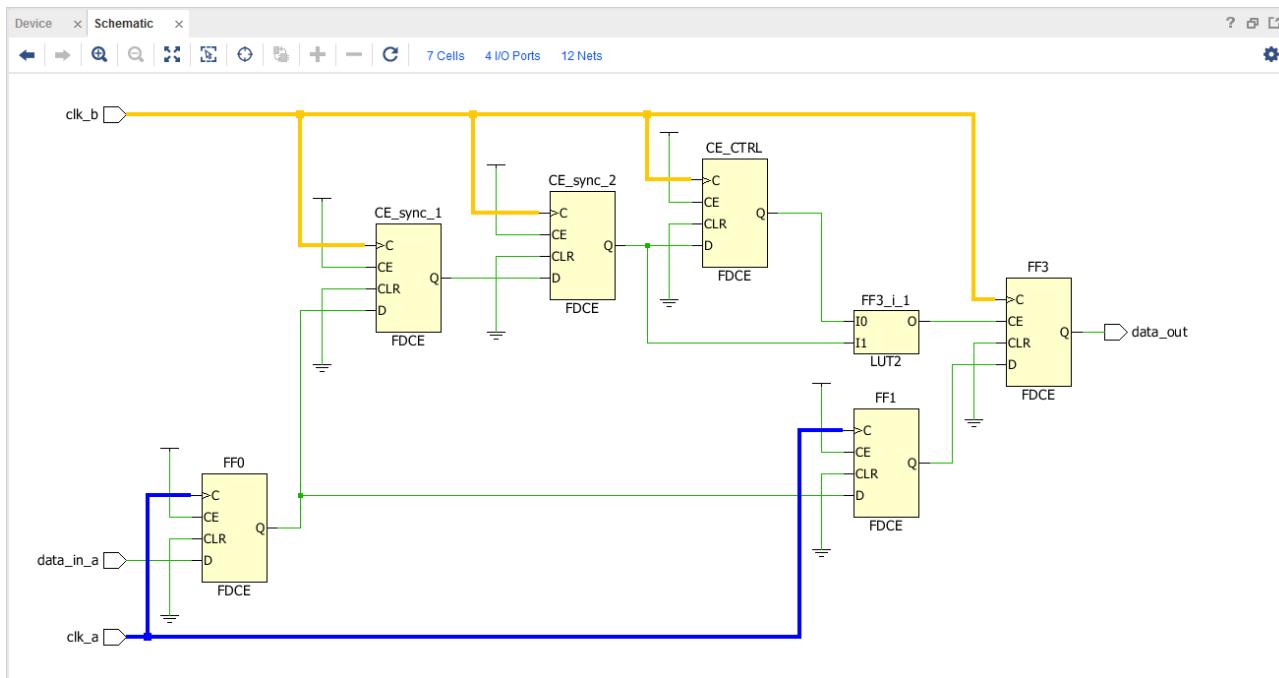


注释：此规则不包含由 CDC-18 检测和覆盖的 HARD_SYNC 宏。

CE 控制的 CDC

在下图所示 CE 控制的 CDC 示例中，时钟使能信号在用于控制交汇触发器之前在目标 **clk_b** 域中已同步。

图 75：CE 控制的 CDC 示例

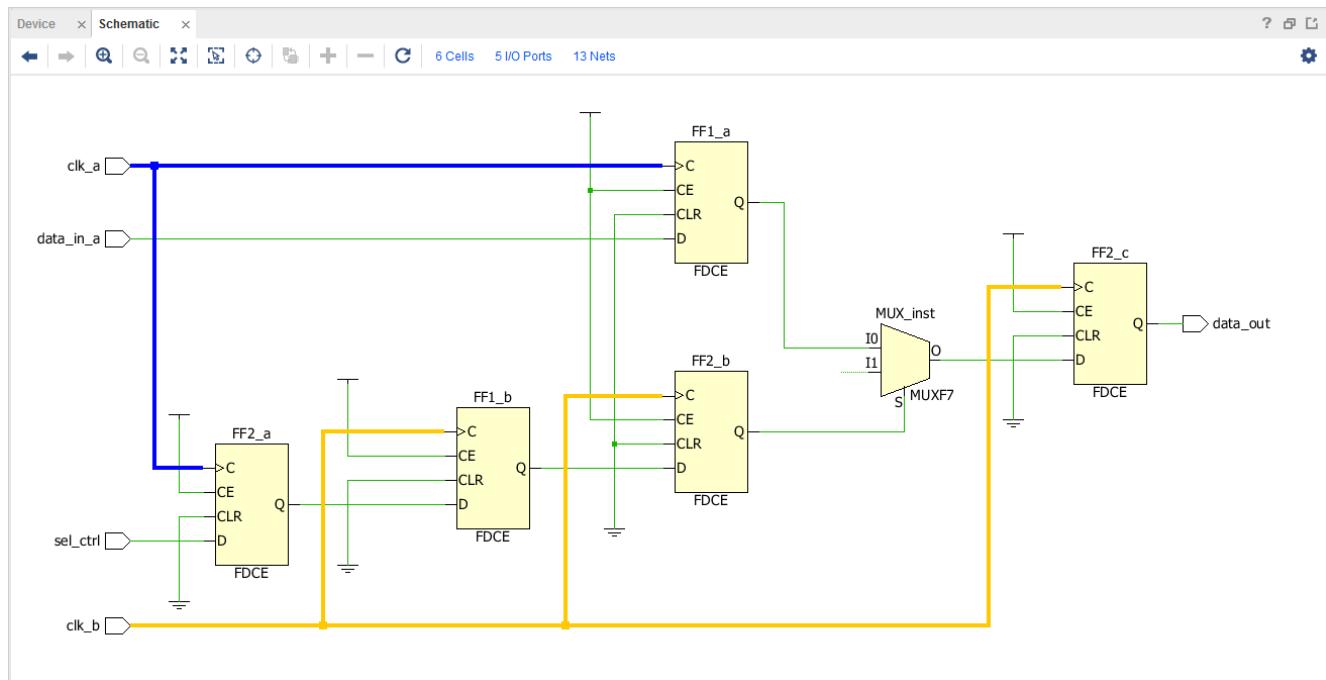


CDC 引擎仅检查连接到 FF3/CE 的信号是否同样由 `clk_b` 发送。对于时钟使能信号的同步方式不存在限制，前提是将其作为安全的 CDC 路径单独报告即可。此外，您还负责约束从 `clk_a` 域到 FF3 的时延，这通常是通过 `set_max_delay -datapath_only` 约束来完成的。

Mux 控制的 CDC

在“Mux 控制的 CDC”示例中，如下图所示，多路复用器选择信号与目标时钟域 `clk_b` 同步。

图 76：Mux 控制的 CDC 示例

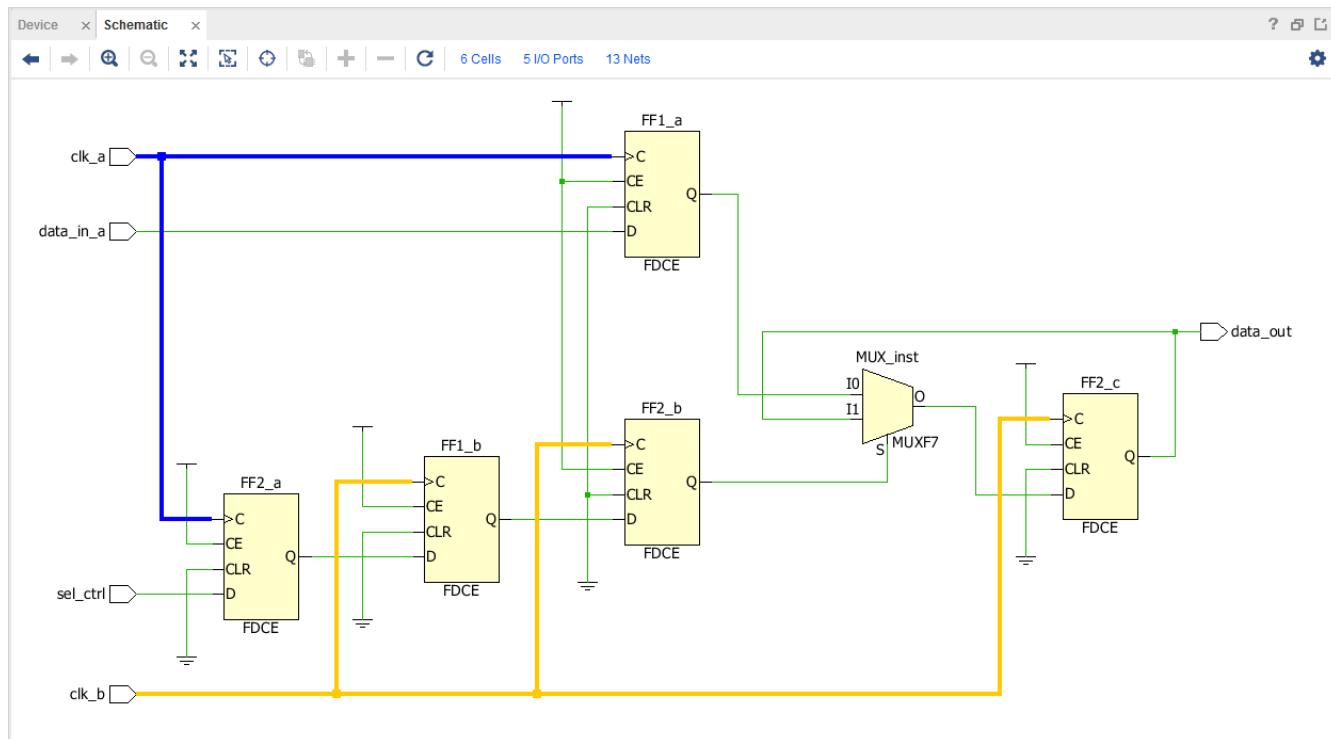


与 CE 控制的 CDC 类似，对于选择信号的同步方式并无限制，前提是此信号单独报告为安全，并且用户负责约束 FF2_c 上的交汇延迟。

Mux 数据保持 CDC

在下图所示的 Mux 数据保持 CDC 示例中，多路复用器的选择信号已同步到目标时钟域 `clk_b`，而 `data_out` 则馈送回多路复用器。

图 77：Mux 数据保持 CDC 示例



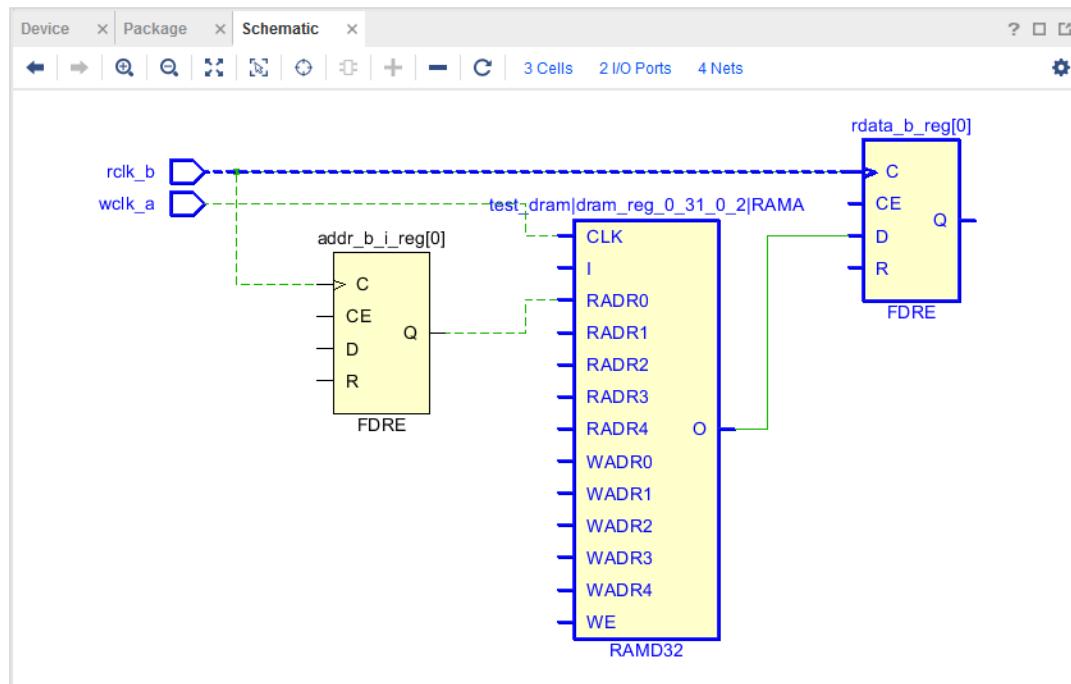
与 CE 控制的 CDC 类似，对于多路复用器选择信号的同步方式并无限制，前提是此信号单独报告为安全，并且用户负责约束 FF2_c 上的交汇延迟。

LUTRAM 读写潜在冲突

在以下 LUTRAM 读写潜在冲突示例中，数据写入含写时钟的 LUTRAM 内，而 LUTRAM 的输出则由读时钟捕获。当读写地址不同时，写时钟与读时钟之间不存在 CDC 路径。但当读写地址相同时，即在写时钟与读时钟之间存在 CDC 路径。

为避免读写时钟之间出现 CDC 路径，需确保 LUTRAM 周围的逻辑在执行活动的读写操作期间，永远无法生成相同的读写地址。确保满足此条件时，与此拓扑结构相关联的 CDC 违例即可获得豁免。例如，赛灵思的 FIFO Generator IP 具有防止任意读写冲突的内置逻辑。

图 78：LUTRAM 读写潜在冲突



总线偏差报告 (Report Bus Skew)

“总线偏差 (Bus Skew)” 报告涵盖了设计中使用 `set_bus_skew` 设置的所有总线偏差约束。总线偏差报告当前不包含在时序汇总报告中，您必须手动生成总线偏差报告和时序汇总报告以完成时序验收。

从 Tcl 控制台运行时，可通过使用 `-cells` 选项将总线偏差报告限定于 1 个或多个层级单元。限定报告范围后，将仅报告含如下数据路径部分的路径：数据路径开始或结束于此类单元或者完全包含于此类单元内。该选项在 “Report Bus Skew” GUI 中不可用。

运行 “总线偏差报告 (Report Bus Skew)”

总线偏差报告可通过在命令行中使用 `report_bus_skew` Tcl 命令或者从 GUI 中的 “Reports” → “Timing” → “Report Bus Skew” 下获取。此命令与 `report_timing` 命令共享许多选项，用于筛选和格式化输出报告。总线偏差约束按其定义顺序进行报告。使用 `-sort_by_slack` 按升序（按裕量从小到大）对约束进行排序。

复查总线偏差路径详情

总线偏差报告包含 2 个部分：

1. 总线偏差报告汇总。
2. 总线偏差报告（按约束）。

“汇报报告 (Report Summary)” 部分

“Report Summary” 部分用于报告设计中定义的所有 `set_bus_skew` 约束。针对每项约束，将报告如下信息：

- **Id**: 报告后续引用的约束 ID (便于在报告内搜索该约束)。
- **From**: 针对 `set_bus_skew -from` 选项提供的模式。
- **To**: 针对 `set_bus_skew -to` 选项提供的模式。
- **Corner**: 计算得到最差总线偏差的角点 (corner) (Slow 或 Fast)。
- **Requirement**: 总线偏差目标值。
- **Actual**: 约束所覆盖的所有路径中计算所得最差总线偏差。
- **Slack**: 最差总线偏差与约束要求之差。

在以下示例中，设计仅含 1 项总线偏差约束，其要求为 1 ns。约束所覆盖的所有路径中的最差偏差为 1.107 ns。

1. Bus Skew Report Summary							
Id	Position	From	To	Corner	Requirement(ns)	Actual(ns)	Slack(ns)
1	4	[get_pins {data1_reg[*]/C}]	[get_pins {data2_reg[*]/D}]	Slow	1.000	1.107	-0.107

“按约束报告 (Report Per Constraint)” 部分

“Report Per Constraint” 部分提供了有关每项 `set_bus_skew` 约束的更多详细信息。每项报告的约束都包含 2 个部分：

1. 约束覆盖的路径的详细汇总。
2. “按约束 (Per Constraint)” 汇总中报告的路径的详细时序路径。

详细汇总表提供了以下信息：

- 源时钟 (From Clock): 起点时钟域。
- 目标时钟 (To Clock): 端点时钟域。
- 端点管脚 (Endpoint Pin): 报告的路径中所含的端点管脚。
- 参考管脚 (Reference Pin): 用于计算偏差的参考管脚。该表每一行都引用导致该端点路径产生最大偏差的参考管脚。
- 角点 (Corner): 用于计算此端点的最差偏差的快/慢 (Fast/Slow) 角点。
- 实际值 (Actual): 计算所得偏差。偏差是端点管脚 (Endpoint Pin) 的相对延迟减去参考管脚 (Reference Pin) 的相对延迟减去相对 CRPR 的差值。
- 裕量 (Slack): 实际路径偏差与要求之差。

注释： 定义总线偏差约束时，必须同时指定 `-from` 和 `-to` 选项。

默认情况下，仅报告含最差总线偏差的端点。要报告多个端点，可使用命令行选项 `-max_paths` 和 `-nworst`。其工作方式与 `report_timing` 命令类似。例如，针对每项约束，`-nworst 1 -max_path 16` 的组合可报告最多 16 个端点，每个端点一条路径。

2. Bus Skew Report Per Constraint							

Id: 1 set_bus_skew -from [get_pins {data1_reg[*]/C}] -to [get_pins {data1c_reg[*]/D}] 1.000 Requirement: 1.000ns							

From Clock	To Clock	Endpoint Pin	Reference Pin	Corner	Actual(ns)	Slack(ns)	-----
clk1	clk2	data1c_reg[1]/D	data1c_reg[2]/D	Slow	1.107	-0.107	
clk1	clk2	data1c_reg[2]/D	data1c_reg[1]/D	Slow	1.107	-0.107	
clk1	clk2	data1c_reg[3]/D	data1c_reg[2]/D	Slow	0.986	0.014	
clk1	clk2	data1c_reg[0]/D	data1c_reg[2]/D	Slow	0.920	0.080	

详细时序路径部分可为“Per Constraint”汇总表中报告的每个管脚对提供详细时序路径。报告的详细路径数量与汇总表中报告的端点数量相同，可使用 `-max_paths/-nworst` 命令行选项来加以控制。

详细总线偏差时序路径的格式与传统时序路径相似，但区别在于，不在端点或参考路径层次报告时钟不确定性。而是改在总线偏差头文件中报告来自端点或参考路径的最差时钟不确定性。此外还请注意，目标时钟的发送时间为 0。对于每个裕量，将打印到端点的时序路径和到参考管脚的时序路径。当时钟和/或数据路径穿越多个 SLR 时，裕量计算期间会应用 SLR 间补偿以防止不必要的消极因素。随后，在总线偏差头文件中报告此类补偿。

以下详细路径是使用命令行选项 `-path_type short` 报告的，用于折叠时钟网络详情。指向端点管脚的路径位于指向参考管脚的路径之前。路径头文件汇总了来自 2 条详细路径的信息以及要求和相对 CRPR：

Slack (MET) :	0.563ns (requirement - actual skew)		
Endpoint Source:	SRC_FF1_CLK0/C (rising edge-triggered cell FDRE clocked by gclk0_1)		
Endpoint Destination:	DST_FF1_CLK1/D (rising edge-triggered cell FDRE clocked by gclk1)		
Reference Source:	SRC_FFO_CLK0/C (rising edge-triggered cell FDRE clocked by gclk0_1)		
Reference Destination:	DST_FFO_CLK1/D (rising edge-triggered cell FDRE clocked by gclk1)		
Path Type:	Bus Skew (Max at Slow Process Corner)		
Requirement:	1.000ns		
Endpoint Relative Delay:	2.844ns		
Reference Relative Delay:	2.403ns		
Relative CRPR:	0.157ns		
Uncertainty:	0.154ns		
Actual Bus Skew:	0.437ns (Endpoint Relative Delay - Reference Relative Delay - Relative CRPR + Uncertainty)		
Endpoint path:			
Location	Delay type	Incr(ns)	Path(ns)
	(clock gclk0_1 rise edge)	0.000	0.000 r
	clock source latency	0.948	0.948
	propagated clock network latency	4.418	5.366
SLICE_X112Y869	FDRE	0.000	5.366 r SRC_FF1_CLK0/C
SLICE_X112Y869	FDRE (Prop_EFF2_SLICEL_C_Q)	0.116	5.482 r SRC_FF1_CLK0/Q
SLICE_X112Y868	net (fo=1, routed)	0.288	5.770 r DST_FF1_CLK1/D
	(clock gclk1 rise edge)	0.000	0.000 r
	propagated clock network latency	2.882	2.882
	clock pessimism	0.000	2.882
SLICE_X112Y868	FDRE (Setup_HFF_SLICEL_C_D)	0.044	2.926 DST_FF1_CLK1
	data arrival	5.770	
	clock arrival	2.926	
	relative delay	2.844	
Reference path:			
Location	Delay type	Incr(ns)	Path(ns)
	(clock gclk0_1 rise edge)	0.000	0.000 r
	clock source latency	1.604	1.604
	propagated clock network latency	3.978	5.582
SLICE_X112Y869	FDRE	0.000	5.582 r SRC_FFO_CLK0/C
SLICE_X112Y869	FDRE (Prop_EFF_SLICEL_C_Q)	0.112	5.694 r SRC_FFO_CLK0/Q
SLICE_X112Y868	net (fo=1, routed)	0.094	5.788 r DST_FFO_CLK1/D
	(clock gclk1 rise edge)	0.000	0.000 r
	propagated clock network latency	3.282	3.282
	clock pessimism	0.000	3.282
SLICE_X112Y868	FDRE (Hold_HFF2_SLICEL_C_D)	0.103	3.385 DST_FFO_CLK1
	data arrival	5.788	
	clock arrival	3.385	
	relative delay	2.403	

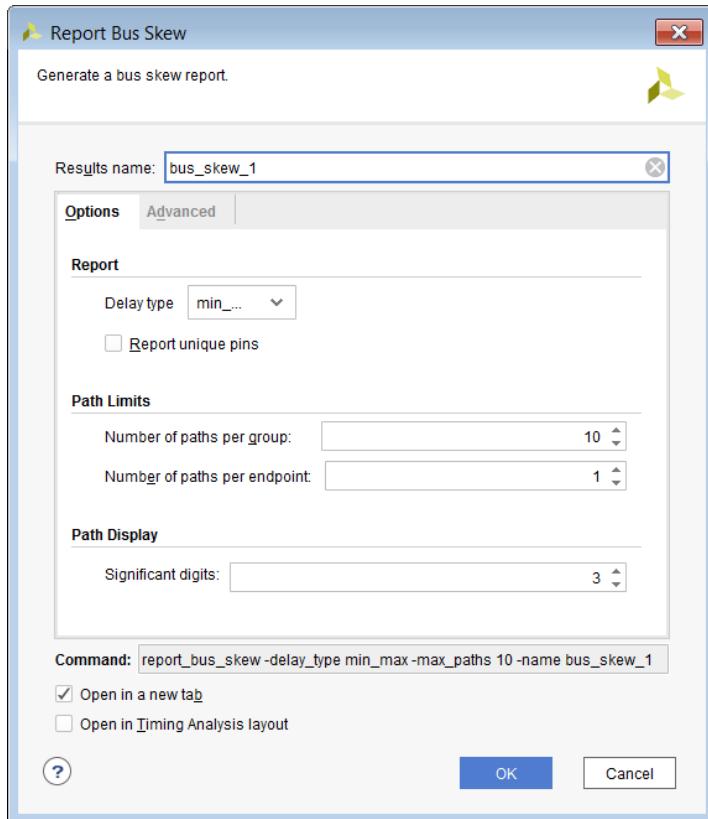
“总线偏差报告 (Report Bus Skew)” 对话框

在 Vivado® IDE 中，要打开 “Report Bus Skew” 对话框，请选择 “Reports” → “Timing” → “Report Bus Skew”。

“Report Bus Skew”对话框：“Options”选项卡

“Report Bus Skew”对话框中的“Options”选项卡如下图所示。

图79：“Report Bus Skew”对话框：“Options”选项卡



“Report Bus Skew”对话框的“Options”选项卡包含以下内容：

- 报告 (Report)
- 路径限制 (Path Limits)
- 路径显示 (Path Display)

报告 (Report)

- 延迟类型 (Delay type): 请参阅“[报告 \(Report\)](#)”部分。
- 唯一管脚报告 (Report unique Pins): 针对每一组唯一的管脚仅显示 1 条时序路径。
等效的 Tcl 选项: `-unique_pins`。

路径限制 (Path Limits)

- 每个组的路径数量: 请参阅[时序汇总报告 \(Report Timing Summary\)](#)。
- 每个端点的路径数量: 请参阅[时序汇总报告 \(Report Timing Summary\)](#)。

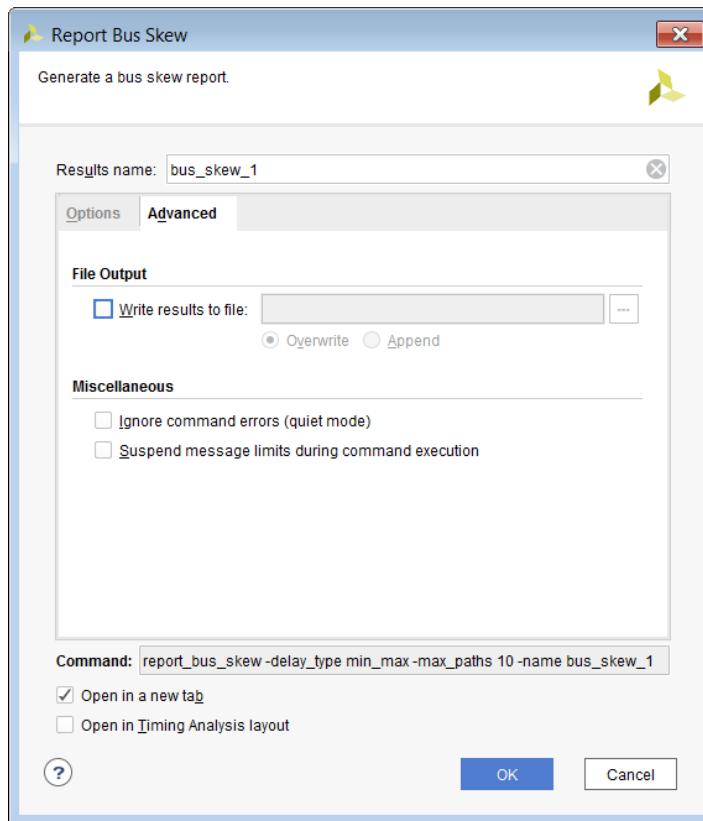
路径显示 (Path Display)

- 有效位数：请参阅[时序汇总报告 \(Report Timing Summary\)](#)。

“Report Bus Skew” 对话框：“Advanced” 选项卡

下图显示了“Report Bus Skew”对话框中的“Advanced”选项卡。

图 80：“Report Bus Skew”对话框：“Advanced”选项卡



“Report Bus Skew”对话框中的“Advanced”选项卡包含以下几个部分：

- [文件输出 \(File Output\)](#)
- [杂项 \(Miscellaneous\)](#)

文件输出 (File Output)

- 将结果写入文件 (Write results to file)：将结果写入指定文件名。默认情况下，报告将写入 Vivado IDE 的“时序 (Timing)”窗口。
等效的 Tcl 选项：`-file`。
- “覆盖 (Overwrite)” 或 “追加 (Append)”：当报告写入文件时，这 2 个选项可用于确定 (1) 覆盖指定文件，还是 (2) 向现有报告追加新信息。
等效的 Tcl 选项：`-append`。

杂项 (Miscellaneous)

- Ignore command errors: 以静默方式执行命令，忽略所有命令行错误，不返回任何消息。此命令还会返回 TCL_OK，忽略执行期间遇到的所有错误。
等效的 Tcl 选项: -quiet。
- 命令执行期间暂挂消息限制 (Suspend message limits during command execution): 临时覆盖所有消息限制并返回所有消息。
等效的 Tcl 选项: -verbose。

“时序汇总 (Timing Summary)” 报告详情

“总线偏差报告 (Bus Skew Report)” 包含下列部分：

- “常规信息 (General Information)” 部分
- “汇总 (Summary)” 部分
- “设置总线偏差 (Set Bus Skew)” 部分

“常规信息 (General Information)” 部分

“Timing Summary” 报告的 “General Information” 部分可提供如下内容的相关信息：

- 设计名称
- 所选器件、封装和速度等级（带有速度文件版本）
- Vivado Design Suite 版本
- 当前日期
- 为生成报告所执行的等效 Tcl 命令

“汇总 (Summary)” 部分

该部分提供了所有总线偏差约束、其要求、实际最差情况总线偏差和每项约束的裕量的汇总信息。汇总表可用于快速查看任意总线偏差约束是否存在违例。

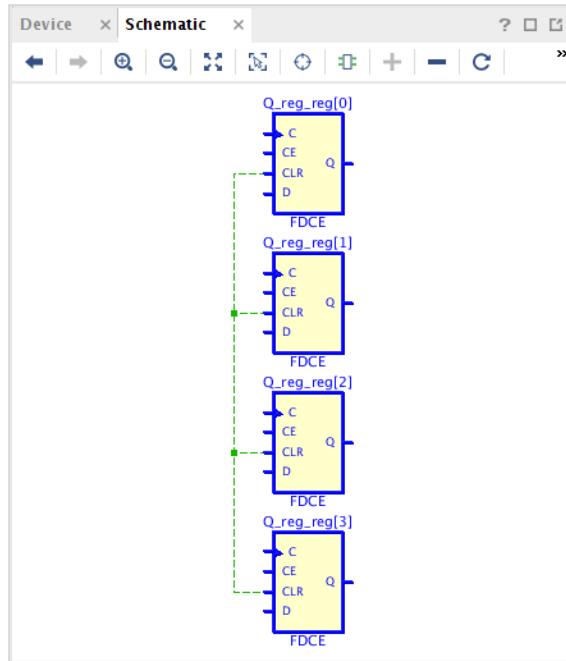
图 81: Report Bus Skew: “Summary” 部分

Constraint	From	To	Corner	Requirement (ns)	Actual (ns)	Slack (ns)
956	cells	cells	Slow	4.000	0.875	3.125
958	cells	cells	Slow	4.000	1.017	2.983
961	cells	cells	Slow	4.000	0.826	3.174
963	cells	cells	Slow	4.000	0.638	3.362
1004	cells	cells	Slow	4.000	2.559	1.441
1044	cells	cells	Slow	4.000	5.561	-1.561
1084	cells	cells	Slow	16.000	0.669	15.331

表格中的“约束 (Constraint)” 列表示时序约束位置编号，与 Timing Constraints Editor (TCE) 中报告的位置编号相匹配。

表格中的“From”和“To”列包含指向 `set_bus_skew` 命令中指定的对象的超链接。对象可以是单元或引脚。单击超链接即可选择特定总线偏差约束的所有起点或端点。选择对象后，可使用 F4 键打开原理图。

图 82：总线偏差端点的原理图示例



“设置总线偏差 (Set Bus Skew)”部分

此部分提供了有关每项“总线偏差 (Bus Skew)”约束的详细时序路径。针对每个时序路径端点都存在 1 个关联的可扩展参考路径

Path 1
Ref Path

总线偏差计算方法为：

实际总线偏差 (Actual Bus Skew) = 端点相对延迟 (Endpoint Relative Delay) - 参考相对延迟 (Reference Relative Delay) - 相对 CRPR (Relative CRPR)

图 83：首个端点及其参考路径的详细路径示例

Set Bus Skew - Constraint 3667									
Name	Slack	Requirement	Levels	Source Clock	Destination Clock	Relative Delay	Relative CRPR	Actual Bus Skew	From
Path 1	-0.318	7.000	0	wire_IF_CLK_0	CLK	4.542	1.306	7.318	wrapper/RED_DAT_r^
Ref Path	-0.318	7.000	0	wire_IF_CLK_0	CLK	-4.082	1.306	7.318	wrapper/RED_DAT_r^
Path 2	-0.248	7.000	0	wire_IF_CLK_0	CLK	4.472	1.306	7.248	wrapper/RED_DAT_r^
Path 3	-0.101	7.000	0	wire_IF_CLK_0	CLK	4.407	1.388	7.101	wrapper/RED_DAT_r^
Path 4	-0.053	7.000	0	wire_IF_CLK_0	CLK	4.359	1.388	7.053	wrapper/RED_DAT_r^
Path 5	0.014	7.000	0	wire_IF_CLK_0	CLK	4.300	1.388	6.986	wrapper/RED_DAT_r^
Path 6	0.092	7.000	0	wire_IF_CLK_0	CLK	4.214	1.388	6.908	wrapper/RED_DAT_r^
Path 7	0.096	7.000	0	wire_IF_CLK_0	CLK	4.210	1.388	6.904	wrapper/RED_DAT_r^
Path 8	0.500	7.000	0	wire_IF_CLK_0	CLK	3.724	1.306	6.500	wrapper/RED_DAT_r^

可选中任一路径并在“属性 (Property)”窗格中查看详细的时序路径报告。可通过单击原理图图标并按 F4 键来生成路径和/或参考路径的原理图（可一并选择端点路径与参考路径）。

第3章

实现结果分析功能

使用“设计运行 (Design Runs)”窗口

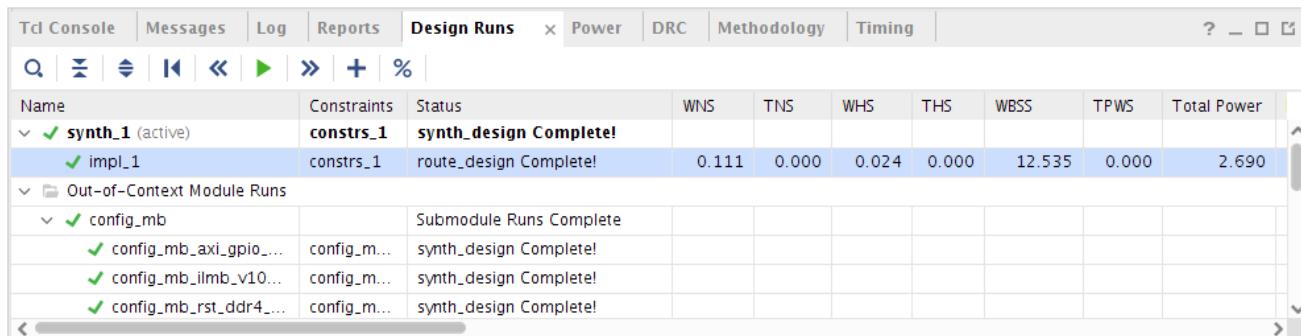
“Design Runs”窗口显示当前运行状态。

欲知详情，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的相应内容。

无论运行处于正在运行、已完成（无错误）还是已完成（有错误）状态，运行完成后都会显示“Design Runs”窗口。

 提示：如果运行并非处于最新状态，可选择弹出菜单中的“Force Up-to-Date”。

图 84：“Design Runs”窗口



Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power
✓ synth_1 (active)	constrs_1	synth_design Complete!							
✓ impl_1	constrs_1	route_design Complete!	0.111	0.000	0.024	0.000	12.535	0.000	2.690

“Design Runs”窗口列显示：

- 运行名称
- 目标器件
- 运行相关约束集
- 运行策略
- 运行最后完成步骤的状态
- 运行进度
- 运行开始时间
- 执行过程中运行的耗用时间或已完成的运行的最终运行时间
- 运行的时序评分：WNS、TNS、WHS、THS、WBSS 和 TPWS（如需了解有关这些数值的更多信息，请参阅[时序汇总报告 \(Report Timing Summary\)](#)）。您可在此快速验证运行是否满足时序要求。如果未能满足时序要求，您必须使用“时序汇总 (Timing Summary)”报告启动分析。

注释：WBSS 表示 report_bus_skew 报告的“最差总线偏差时序裕量 (Worst Bus Skew Slack)”。

- 未能成功布线的信号线数量
- 设计 LUT、FF、块 RAM、DSP 以及（如果适用）UltraRAM 的利用率。
- 总功耗估算
- 运行策略简介
- 设计运行的增量模式

如果您适用的是 Vivado® IDE 工程流程，请复查处于活动状态的综合和实现运行的“消息 (Messages)”选项卡。消息按流程中的运行步骤来分组。在此经过整合和筛选的视图中会显示运行日志文件中保存的所有信息以及 Vivado 主会话日志文件。

图 85：按步骤分组的消息



某些消息会对源文件进行回溯性交叉探测（单击文件名即可打开源文件），或者在某些情况下，还会对消息相关的设计对象进行回溯性交叉探测。根据当前分析的流程步骤，您必须打开综合后设计或实现后设计才能使用从消息交叉探测的对象。

布局分析

本节探讨布局分析，具体包括：

- 高亮显示布局
- 显示连接 (Showing Connectivity)
- 查看指标

高亮显示布局

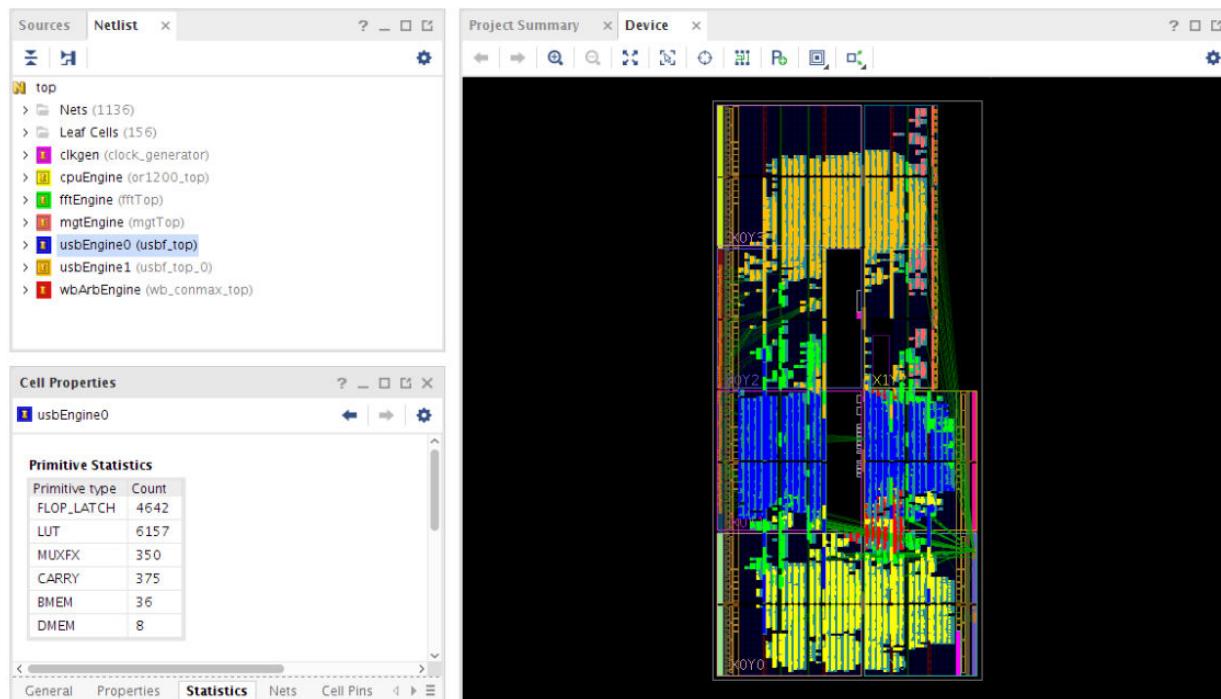
审查设计布局的另一途径是分析单元布局。“高亮叶节点单元 (Highlight Leaf Cells)”命令可帮助完成此分析。

- 在“网表 (Netlist)”窗口中，选择要分析的层级。

2. 从弹出菜单中选择“Highlight Leaf Cells”→“Select a color”。
3. 如果要选择多个层级，请选择“Cycle Colors”。

在“Device”窗口中，构成层级单元的叶节点单元以颜色编码标示。

图 86：高亮显示层级



颜色编码用于显示器件内主要层级块的布局。`usbEngine0`（蓝色）：

- 使用一定数量的块 RAM 和 DSP48 单元。
- 位于芯片的中央时钟区域。
- 与设计内的其它逻辑(`fftEngine`)混合。

显而易见，`fftEngine`（绿色）和`cpuEngine`（黄色）已混合。这两个块主要使用不同资源（DSP48 和 slice）。混合能使器件得到充分利用。

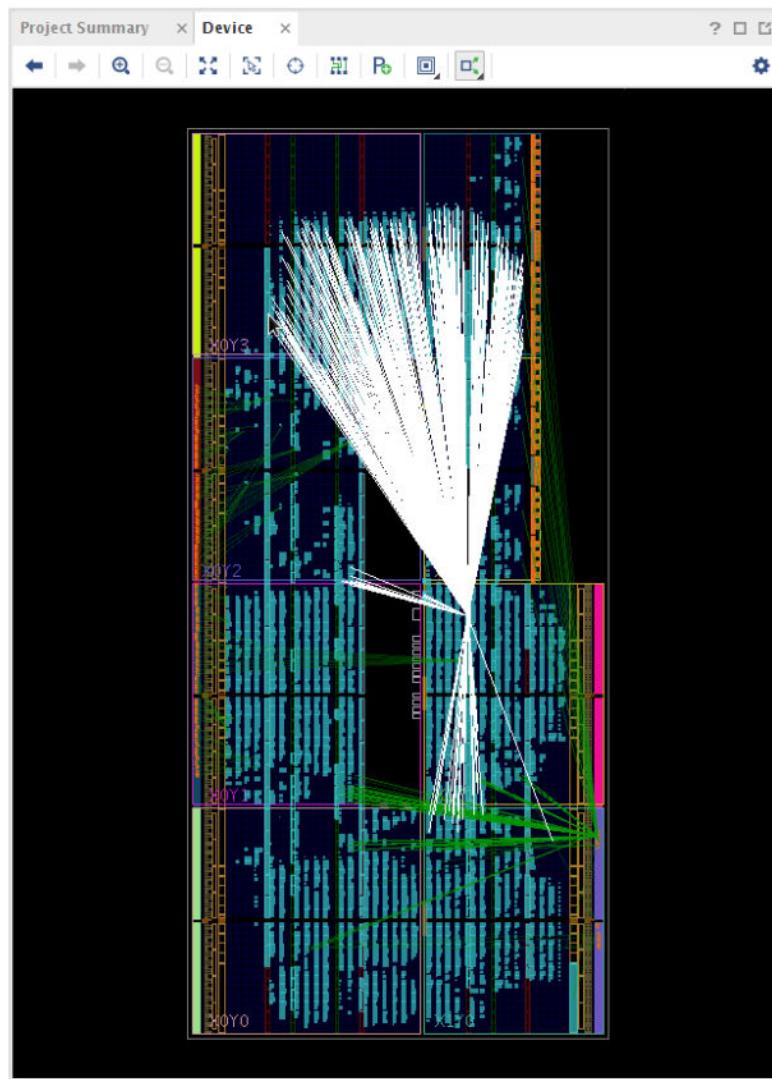
显示连接 (Showing Connectivity)

根据连接来分析设计是很实用的方法。运行“Show Connectivity”以复查由输入、块 RAM 或 DSP bank 驱动的所有逻辑的布局。“显示连接(Show Connectivity)”将一组单元或信号线作为种子，并选择另一种类型的对象。

提示：通过使用此方法来构建设计并查看其中的逻辑椎。

下图显示的是包含 OBUF 的器件内驱动逻辑的块 RAM。综合编译指示会在内存推断期间阻止综合在块 RAM 内对输出触发器进行布局。

图 87：显示连接 (Showing Connectivity)



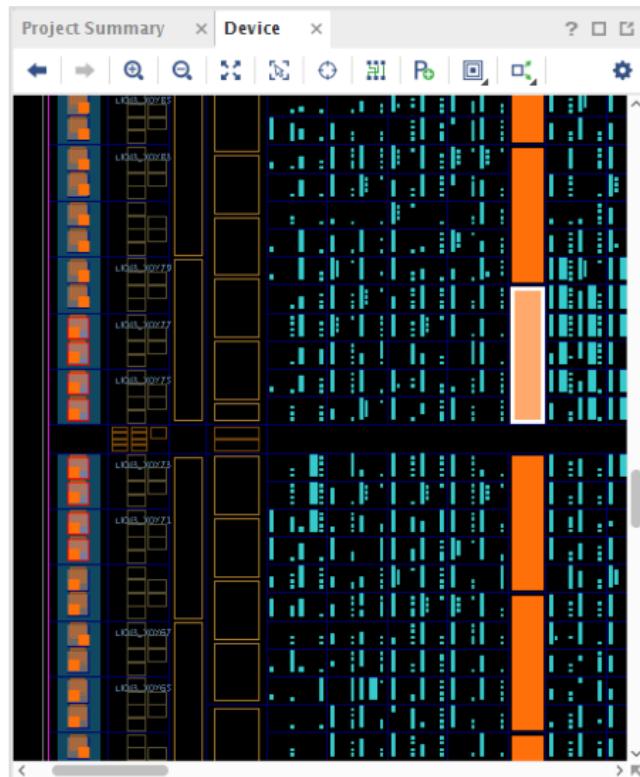
固定逻辑和非固定逻辑

Vivado 工具可追踪 2 种不同类型的布局：

- 用户布局的元素（显示为橙色）为固定逻辑。
 - 固定逻辑存储在 XDC 内。
 - 正常情况下，固定逻辑包含 LOC 约束，也有可能包含 BEL 约束。
- 工具布局的元素（显示为蓝色）为非固定逻辑。

在下图中，I/O 和块 RAM 布局属于固定逻辑。slice 逻辑属于非固定逻辑。

图88：固定布局和非固定布局



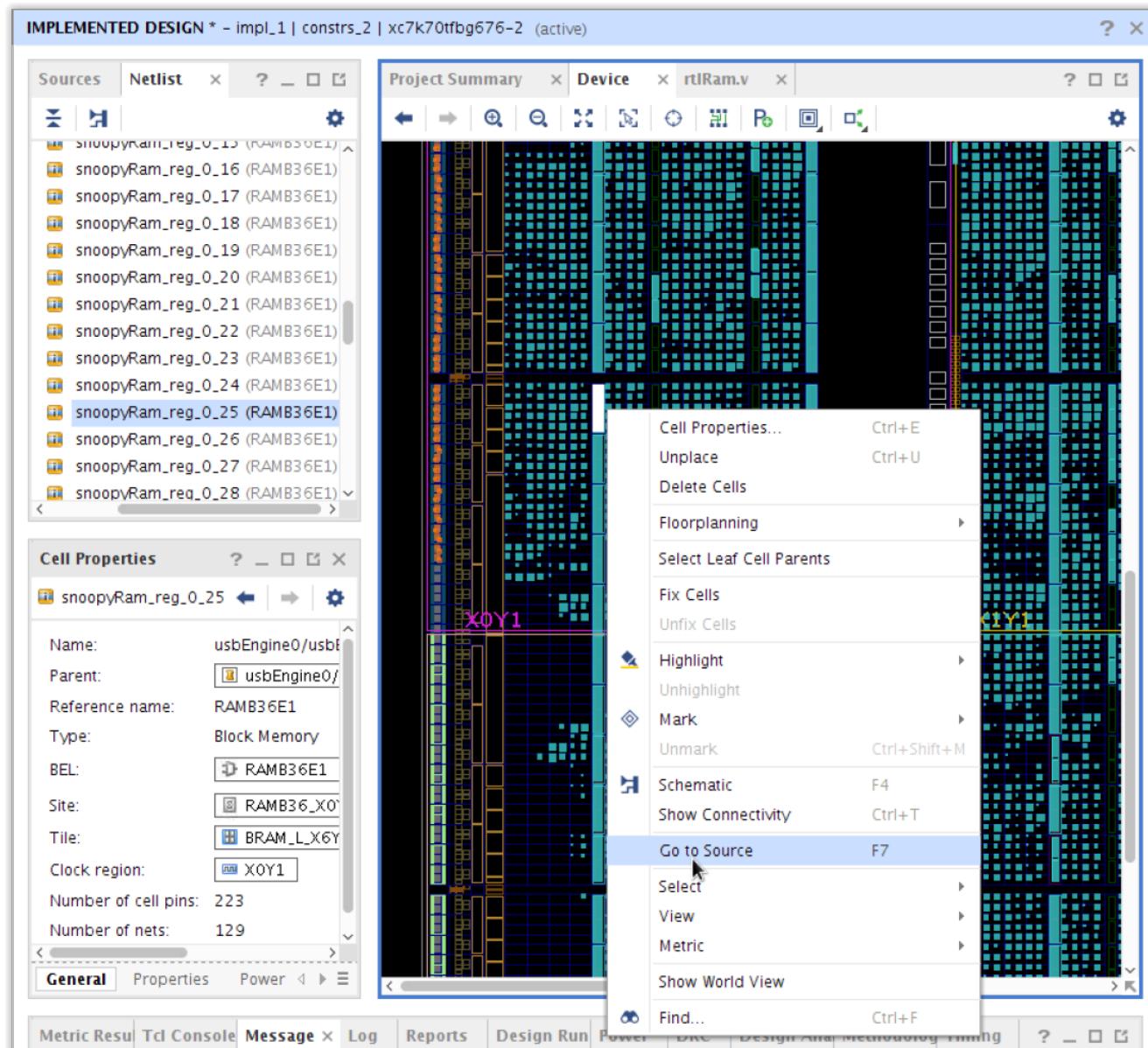
交叉探测

对于使用 Vivado Synthesis 进行综合的设计，当网表设计进入内存后，即可对源文件进行反向交叉探测。

要执行交叉探测：

1. 选择门电路。
2. 从弹出菜单中选择“Go to Source”，如下图所示。

图89：向源时钟进行反向交叉探测

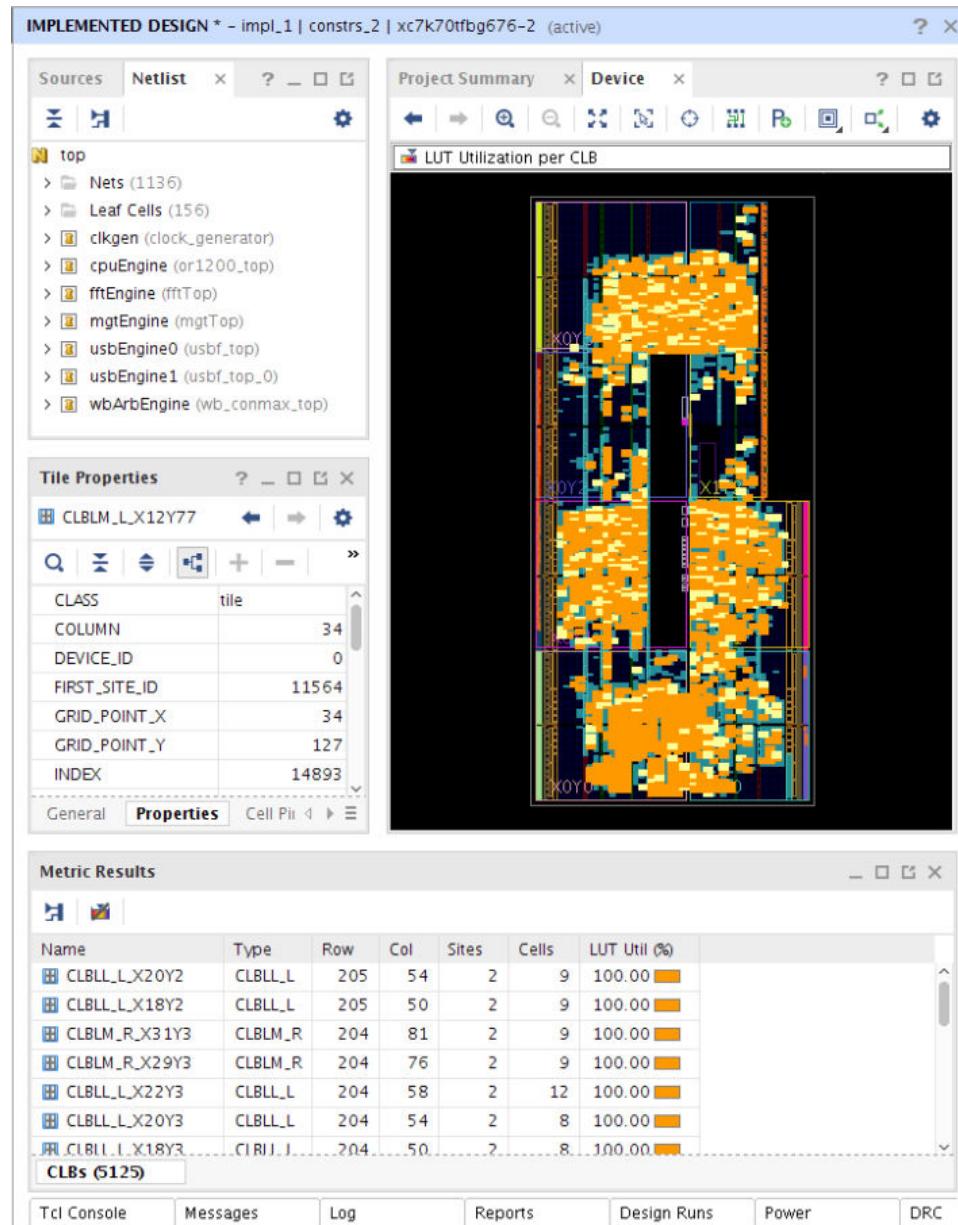


使用交叉探测判定网表门电路中是否涉及源时钟。由于综合变换的性质，无法对设计中每个门电路的源时钟进行反向交叉探测。

查看指标

完成实现后，可分析设计以查看设计与器件的交互方式。Vivado IDE 具有多个指标用于帮助您判定器件中的逻辑和布线使用情况。这些指标根据指定规则对器件窗口进行颜色编码。要查看指标，请右键单击“Device”窗口、选择“Metric”，然后选择要查看的指标。请参阅下图。

图90：指标



需设计完成布局才有效的指标

有4项指标要求设计完成布局后才能保证准确性。但这些指标不要求设计完全完成布线。

- LUT利用率（按 CLB）(LUT Utilization per CLB): 基于已布局的 LUT 利用率对各 slice 进行颜色编码。
- FF利用率（按 CLB）(FF Utilization per CLB): 基于已布局的 FF 利用率对各 slice 进行颜色编码。
- 垂直布线拥塞（按 CLB）(Vertical Routing Congestion per CLB): 基于垂直布线利用率的最佳情况估算对结构进行颜色编码。
- 水平布线拥塞（按 CLB）(Horizontal Routing Congestion per CLB): 基于水平布线利用率的最佳情况估算对结构进行颜色编码。

对于 UltraScale+ 和更新的架构：

- 互连拥塞等级 (Interconnect Congestion Level)：基于连续区域上的布线利用率的最差情况估算对“互连拥塞等级”进行颜色编码。

不含布局的网表设计中的指标

如果存在 Pblock，则适用 2 项指标。这些指标与布局无关联：

- 每个 Pblock 的 LUT 利用率 (LUT Utilization per Pblock)：根据在 Pblock 所含 slice 中对 LUT 进行布局的方式预测，对 Pblock 进行颜色编码。
- 每个 Pblock 的 FF 利用率 (FF Utilization per Pblock)：根据在 Pblock 所含 slice 中对 FF 进行封装的方式预测，对 Pblock 进行颜色编码。

每次均可使用多条规则，如下图所示。“LUT 利用率（按 CLB）”和“FF 利用率（按 CLB）”已同时启用。

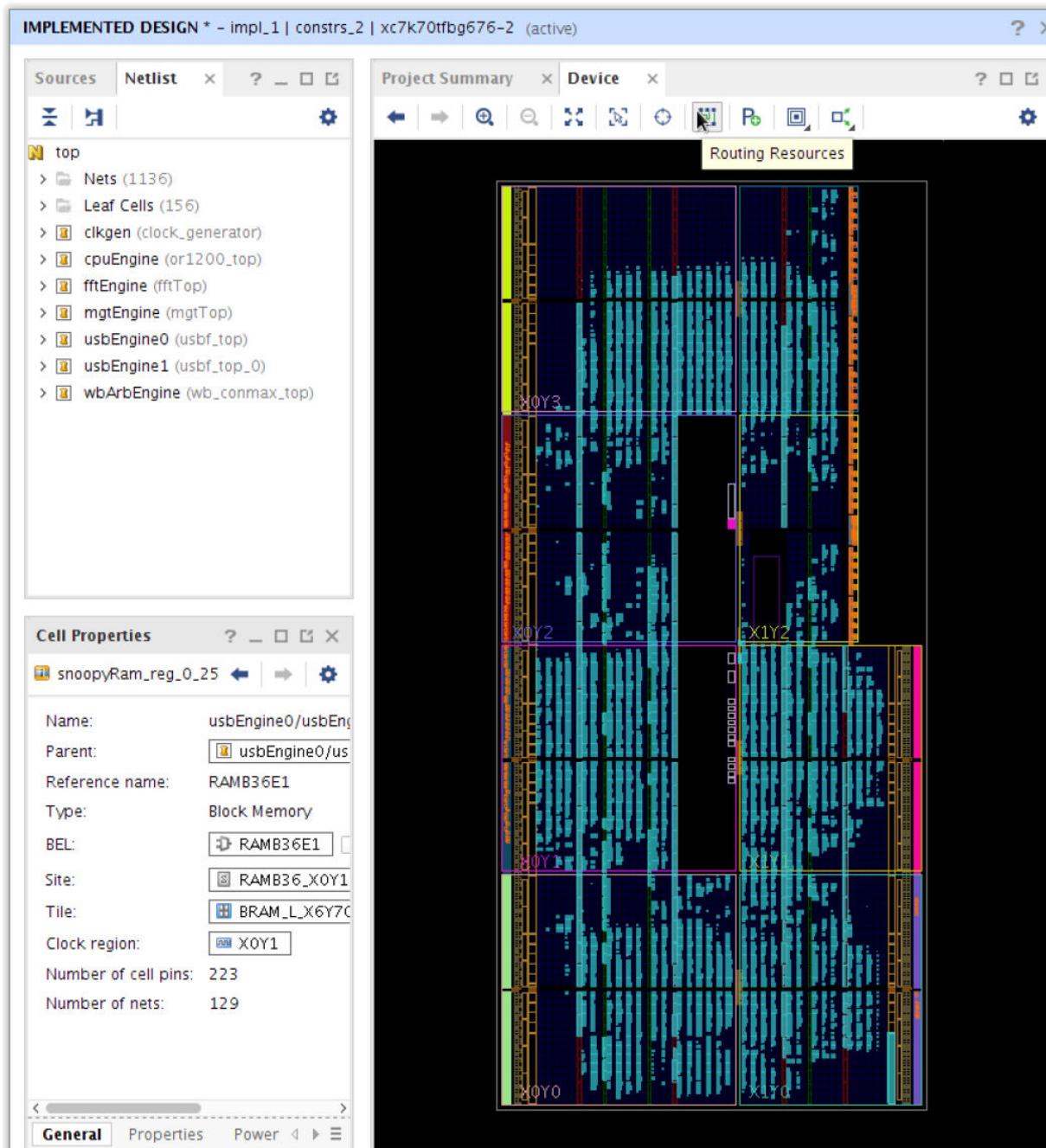


提示：如果设计中某些部分的布线拥塞利用率较高或估算值较高，请考虑对 RTL 或布局约束进行微调以降低该区域内的逻辑和布线利用率。

布线分析

在“Device”窗口中开启“布线资源 (Routing Resources)”即可查看具体的布线资源。

图91：启用布线



显示布线与布局

布线与布局可根据缩放比例以2种不同方式显示：

- 缩小
- 放大



提示：“Device”窗口的2种可视化方式有利于最大限度降低运行时间和内存占用，同时显示各种规模的设计的细节。

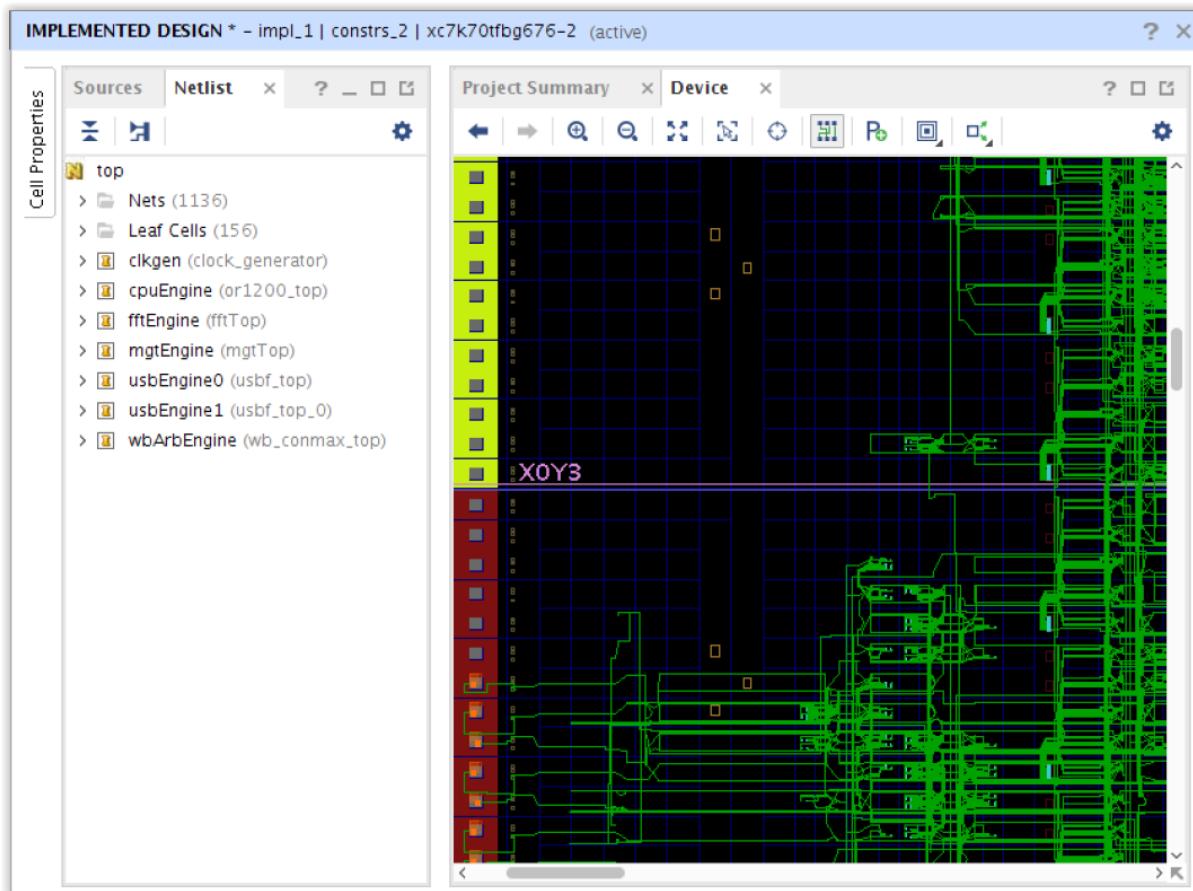
显示缩小的布局布线

缩小时显示抽象视图。抽象视图：

- 精简穿过器件的布线。
- 根据穿过特定区域的布线数量显示不同粗细的线条。

类似地，布局以块来表示包含已布局的逻辑的每个拼块(tile)。拼块中的逻辑越多，表示此拼块的块尺寸越大。

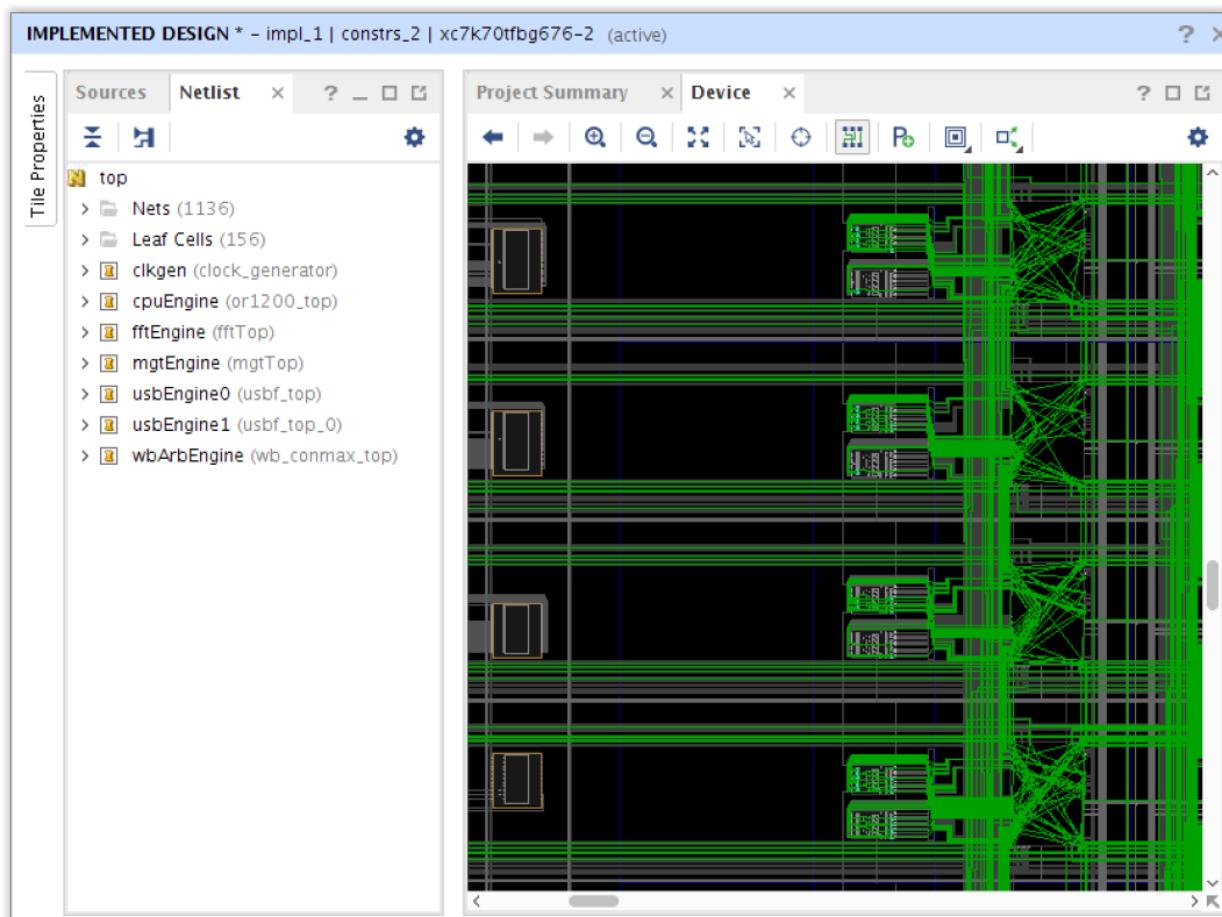
图92：抽象视图



放大情况下显示布线与布局

在放大情况下，会显示实际逻辑单元与布线。

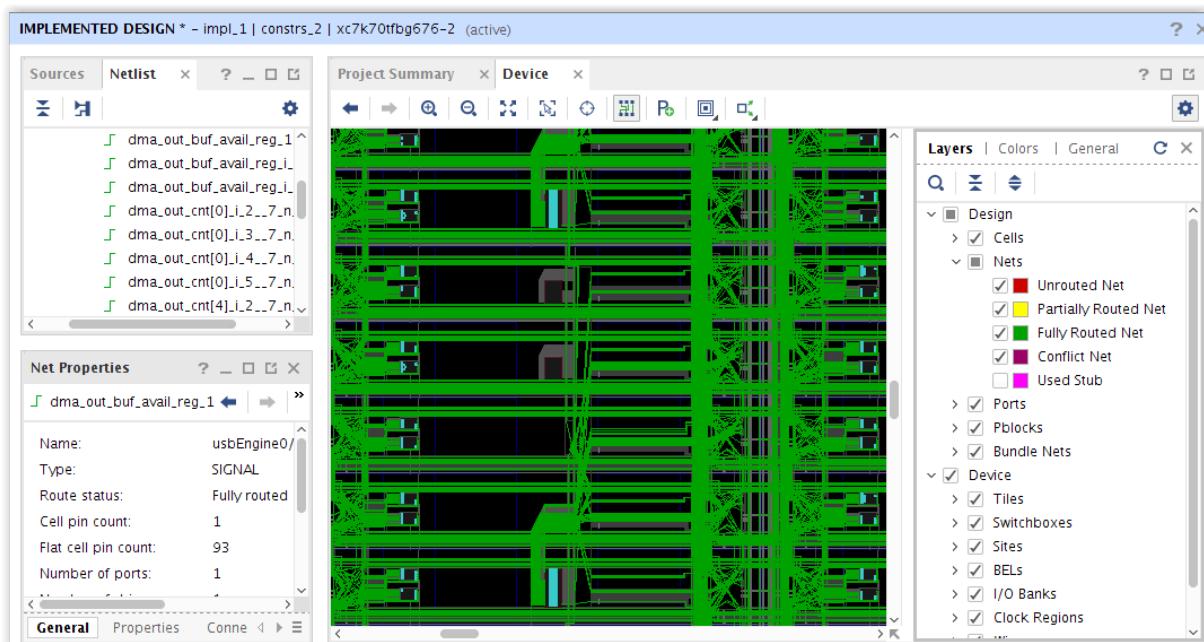
图93：详细视图



查看选项

“Device”窗口可自定义，以便通过多种方式来显示器件和设计。大部分选项均可通过“器件选项 (Device Options)”滑出菜单来控制。

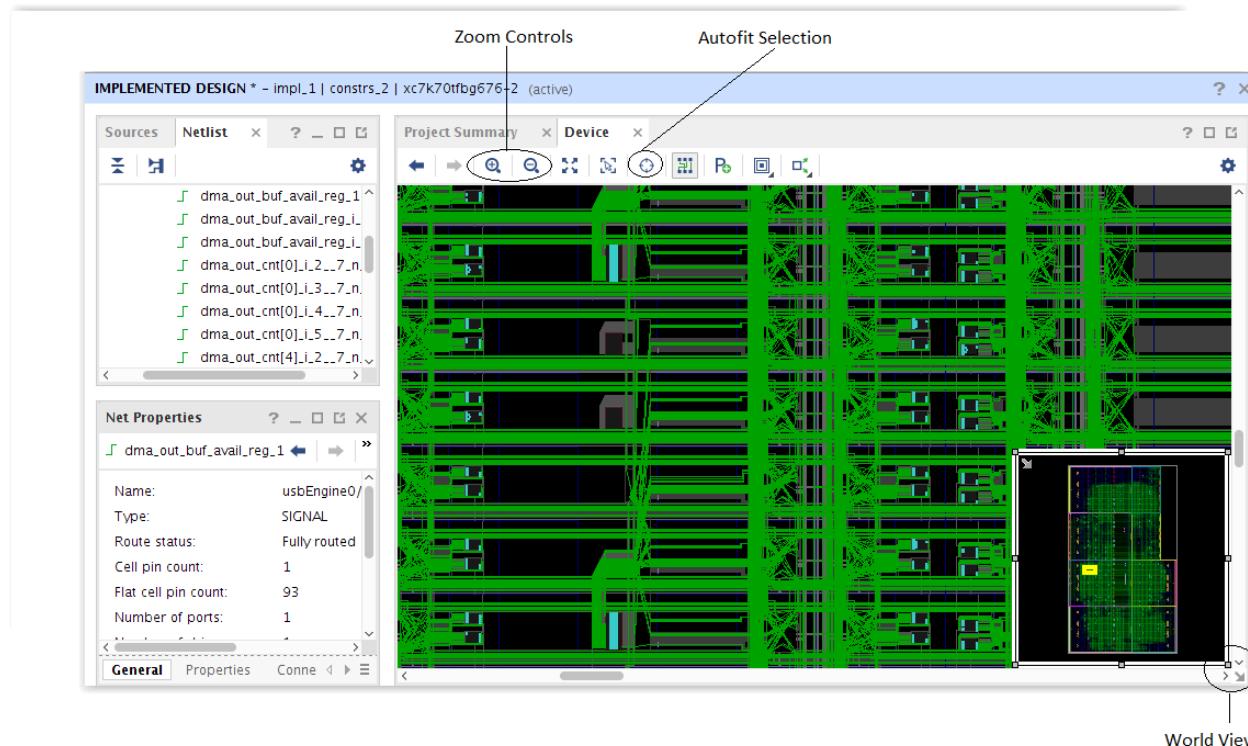
图94：“Device”窗口层次



您可为不同设计和器件资源启用或禁用图形显示功能，也可以修改显示颜色。

浏览“Device”窗口

图95：浏览“Device”窗口



使用以下工具浏览“Device”窗口：

- 缩放控件：标准“放大 (Zoom In)”、“缩小 (Zoom Out)”和“缩放至全屏 (Zoom Full)”工具。
- 自适应选择：自动缩放和平移至器件外部任意视图中的选定对象。自适应选择对交叉探测特别有用。
- “World”视图：“World”视图用于显示器件的当前可见部分在整个器件上的位置。您可移动和缩放“World”视图，也可以拖曳和缩放黄色框以进行缩放和平移。
- 控制热键：在单击和拖曳平移视图时按 Ctrl 键。使用“Ctrl”键和鼠标滚轮可缩放光标所在位置。

设计分析报告 (Report Design Analysis)

“设计分析 (Design Analysis)” 报告可提供有关时序路径特性、设计互连复杂性以及拥塞的信息。此信息可供您用于执行设计或约束更改，以改善 QoR 并且有可能缓解布线拥塞。

运行“设计分析报告 (Report Design Analysis)”

您可从 Tcl 控制台或 Vivado® IDE 运行“Report Design Analysis”。 “Report Design Analysis” 可生成的报告分 3 种类别：

- 时序报告 (Timing)：用于报告时序路径的时序和物理特性。

- 复杂性报告 (Complexity): 分析设计的布线复杂性和 LUT 分布情况
- 拥塞报告 (Congestion): 分析设计的布线拥塞

要在 Vivado IDE 中运行“Report Design Analysis”，请选择“Reports”→“Report Design Analysis”。

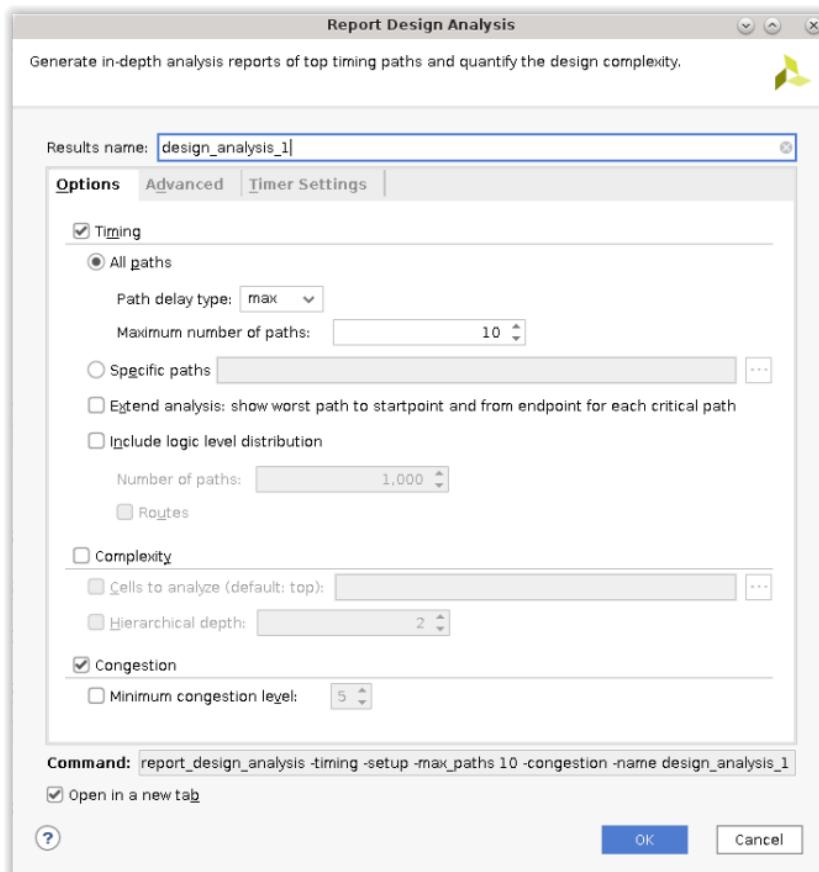
等效的 Tcl 命令：`report_design_analysis -name design_analysis_1`

注释：某些“Report Design Analysis”选项仅限通过运行 `report_design_analysis` Tcl 命令才能使用。您可使用 `-name` 选项在 GUI 中查看此 Tcl 命令的结果。

在 Vivado IDE 中，“Report Design Analysis”对话框（如下图所示）包含以下内容：

- “结果名称 (Results Name)”字段
- “选项 (Options)”选项卡
- “高级 (Advanced)”选项卡
- “定时器设置 (Timer Settings)”选项卡

图 96：“设计分析报告 (Report Design Analysis)”对话框



“结果名称 (Results Name)”字段

在位于“Report Design Analysis”对话框顶部的“Results Name”字段中，指定报告的图形窗口的名称。

等效的 Tcl 选项：-name <windowName>

“Options” 标签

在“选项 (Options)” 选项卡（如下图所示）中，包含下列字段：

- 时序 (Timing)
- 复杂性 (Complexity)
- 拥塞 (Congestion)

“时序 (Timing)” 字段

“Timing” 字段允许您报告时序路径的时序和物理特性。

等效的 Tcl 选项：-timing

您可选择为所有路径或者为特定时序路径生成报告。如果选择“所有路径 (All Paths)” 选项，则可指定路径延迟类型：max（对应建立）、min（对应保持）或 min_max（对应建立和保持）。

等效的 Tcl 选项：-setup/-hold

您还可指定每个时钟组的最大路径数（默认值为 10）。

等效的 Tcl 选项：-max_paths <arg>

如果选择“特定路径 (Specific Paths)” 选项，则将针对指定路径对象执行分析。单击右侧的“Browse” 按钮即可打开搜索对话框，以帮助查找路径对象。如需了解有关 get_timing_paths 的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的相应内容。

等效的 Tcl 选项：-of_timing_paths <args>

选择“Extend Analysis” 选项可针对每条感兴趣的路径执行扩展分析，增加指向起点的最差路径和源自端点的最差路径的相关报告。

等效的 Tcl 选项：-extend

注释：运行“Extend Analysis” 选项 (Tcl 选项 -extend) 以执行保持路径分析时，工具生成的报告会显示含相同起点和端点的路径的建立和保持时间特性，以显示保持时间修复是否会影响建立时序。

选中“Include logic-level distribution” 选项并指定要使用的路径数量即可包含逻辑层次分布信息。如果同时分析所有路径，那么所选路径数量会覆盖每个时钟组的最大路径数。如果分析特定路径，那么只提供指定路径的逻辑层次分布信息。

等效的 Tcl 选项：-logic_level_distribution -logic_level_dist_paths <arg>

“复杂性 (Complexity)” 字段

“Complexity” 字段允许您报告设计网表的复杂性，它可反映整个层级中的连接密度。请参阅[复杂性报告](#)。

等效的 Tcl 选项：-complexity

选择“Cells to Analyze” 选项，以指定要用于复杂性报告的层级单元。单击右侧的“Browse” 按钮以打开搜索对话框，帮助查找单元对象。

等效的 Tcl 选项：-cells <args>

如果选择“层级深度 (Hierarchical Depths)”选项，那么您可在顶层（默认）或者在 -cells 选项指定的单元层次选择要检验的层级数。

等效的 Tcl 选项：`-hierarchical_depth <arg>`

“拥塞 (Congestion)” 字段

“Congestion”字段用于切换 -congestion Tcl 的开启和关闭。

选择“Minimum congestion level”选项可指定在设计中显示布线器拥塞的最低拥塞水平。如果不指定，默认最低拥塞水平为 5。该值的取值范围在 3 到 8 之间。

等效的 Tcl 选项：`-min_congestion_level <args>`

注释：如果所有布线器拥塞区域全都低于阈值等级 (-min_congestion_level)，则不生成/显示拥塞报告。在此情况下，请降低阈值（介于 3 到 8 之间的值）并重新运行此命令。

“高级 (Advanced)” 选项卡

在“Advanced”选项卡（如下图所示）中，包含下列字段：

- 文件输出 (File Output)
- 杂项 (Miscellaneous)

“文件输出 (File Output)” 字段

除生成 GUI 报告外，您还可通过选择“Export to file”并在右侧字段中指定文件名来将结果写入文件。单击“Browse”按钮可选择其它目录。

等效的 Tcl 选项：`-file <arg>`

选择“Overwrite”选项可使用新的分析结果来覆盖现有文件。

选择“Append”可追加新结果。

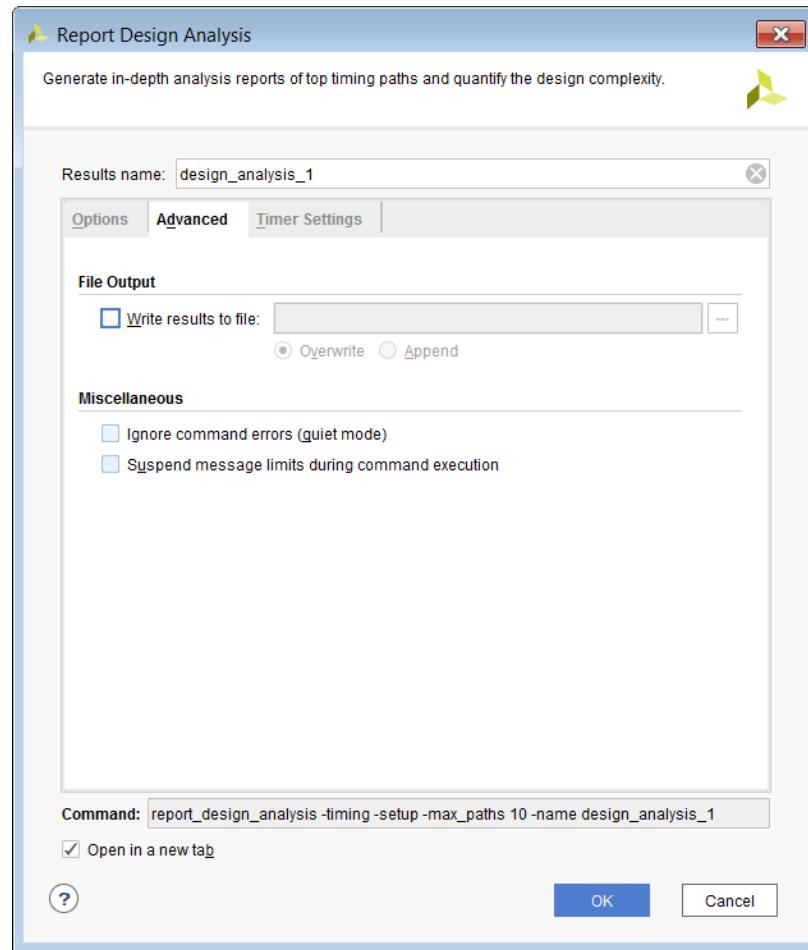
等效的 Tcl 选项：`-append`

“杂项 (Miscellaneous)” 字段

“Miscellaneous”字段提供了在命令执行期间忽略命令错误和暂挂消息限制的选项。

等效的 Tcl 选项：`-quiet/-verbose`

图97：“Report Design Analysis”对话框：“Advanced”选项卡



“定时器设置 (Timer Settings)” 选项卡

在“Timer Settings”选项卡（如下图所示）中，包含下列字段和选项。

- “互连 (Interconnect)” 选项
- “速度等级 (Speed Grade)” 选项
- “多角配置 (Multi-Corner Configuration)” 字段
- “禁用飞行延迟 (Disable Flight Delays)” 选项

“互连 (Interconnect)” 选项

您可选择时序路径分析中使用的互连模型。

- actual：为已布线设计提供最准确的延迟。
- estimated：包含基于设计布局和设计连接到器件（实现前）的方式估算的互连延迟。即使设计已完全布线，仍可指定估算延迟。
- none：在时序分析中不包含任何互连延迟；仅应用逻辑延迟。

等效的 Tcl 命令：`set_delay_model -interconnect <arg>`

如需了解有关 `set_delay_model` 的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#))。

“速度等级 (Speed Grade)” 选项

您可对默认速度等级执行分析，或者也可以选择其它速度等级进行分析。

等效的 Tcl 命令：`set_speed_grade <arg>`

如需了解有关 `set_speed_grade` 的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#))。

“多角配置 (Multi-Corner Configuration)” 字段

您可使用该字段中的可用选项来限制由 Vivado 时序分析引擎执行的默认四角分析（如果适用）。

等效的 Tcl 命令：`config_timing_corners -corner <arg> -delay_type <arg>`

如需了解有关 `config_timing_corners` 的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#))。

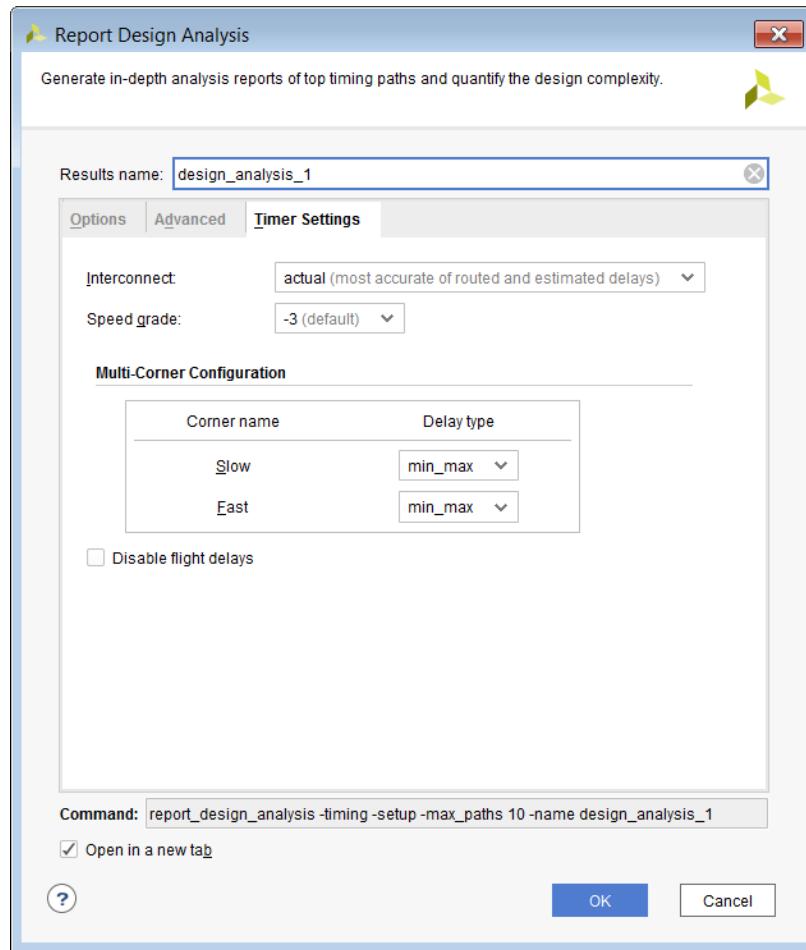
“禁用飞行延迟 (Disable Flight Delays)” 选项

您可选择该选项以禁用向 I/O 时序计算添加封装延迟。

等效的 Tcl 命令：`config_timing_analysis -disable_flight_delays <arg>`

如需了解有关 `config_timing_analysis` 的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#))。

图98：“Report Design Analysis”对话框的“Timer Settings”选项卡



仅限命令行使用的选项

以下“时序 (Timing)”选项仅限通过 Tcl 命令行使用，可配合 `-name` 选项一起使用以生成 GUI 报告。

- `-routed_vs_estimated`

该选项用于报告相同路径的估算延迟与实际布线延迟并列对比结果。报告中的“时序类别 (Timing Category)”中的某些字段带有“Estimated”或“Routed”前缀以便比较。

- `-return_timing_paths`
- `-end_point_clock`
- `-logic_levels`

以下“复杂性 (Complexity)”选项仅限通过命令行使用，可配合 `-name` 选项一起使用以生成 GUI 报告。

- `-bounding_boxes <arg>`

该选项用于执行指定边框的复杂性分析。例如：

```
-bounding_boxes { "CLE_M_X21Y239:CLEL_R_X28Y254" "CLEL_R_X18Y171:CLE_M_X26Y186" }
```

注释：在左括号“{”与边框起始位置之间需添加1个空格，如以上示例中所示。

时序路径特性报告

下图显示了在“时序模式 (Timing Mode)”下运行“设计分析报告 (Report Design Analysis)”的输出示例，其中显示了设计中10条最差建立路径的路径特性。可通过GUI（“Reports > Report Design Analysis”）或使用Tcl命令生成此报告：

```
report_design_analysis -name <arg>
```



提示：要创建保持路径特性，请选择“Report Design Analysis”对话框的“Options”选项卡中的“Path delay type: min”或者在Tcl命令中添加`-hold`。如需了解有关Tcl命令语法的更多信息，请参阅《Vivado Design Suite Tcl命令参考指南》(UG835)。

图 99：建立路径特性示例

Name	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Clock Relationship	Routes	Logical Path	Start Point Clock	End Point Clock	DSP Block	BRAM	High Fanout	Start Point Pin Primitive	End Point Pin Primitive
Path 1	10	8.968	0.223	8.745	-0.377	0.150	Safely Timed	1	FDRE FDSE wbClk	phyClk1_1	None	None	10420	FDRE/C	FDSE/S	
Path 2	10	8.968	0.223	8.745	-0.377	0.150	Safely Timed	1	FDRE FDSE wbClk	phyClk1_1	None	None	10420	FDRE/C	FDSE/S	
Path 3	10	8.968	0.223	8.745	-0.377	0.150	Safely Timed	1	FDRE FDSE wbClk	phyClk1_1	None	None	10420	FDRE/C	FDSE/S	
Path 4	10	8.97	0.223	8.747	-0.374	0.150	Safely Timed	1	FDRE FDSE wbClk	phyClk1_1	None	None	10420	FDRE/C	FDSE/S	
Path 5	10	8.968	0.223	8.745	-0.374	0.152	Safely Timed	1	FDRE FDSE wbClk	phyClk1_1	None	None	10420	FDRE/C	FDSE/S	
Path 6	10	8.972	0.223	8.749	-0.37	0.152	Safely Timed	1	FDRE FDSE wbClk	phyClk1_1	None	None	10420	FDRE/C	FDSE/S	

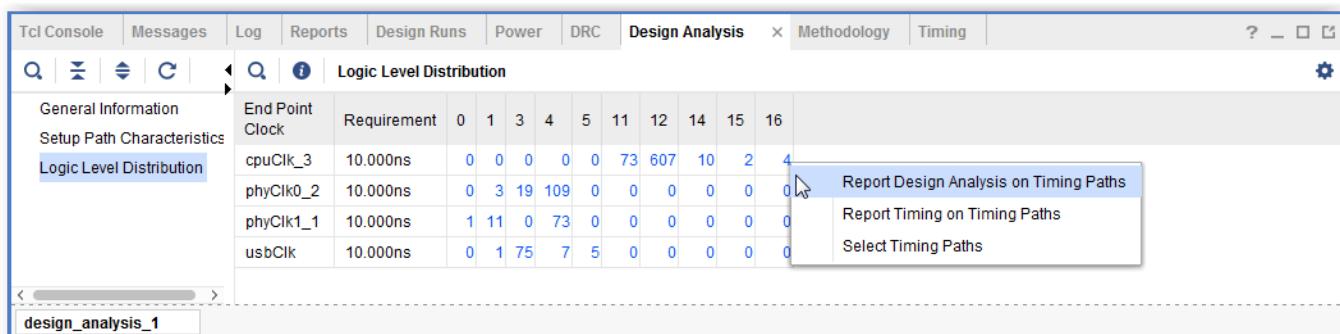
此外，“Report Design Analysis”还提供了最差时序路径的“逻辑层次分布 (Logic Level Distribution)”表。“Logic Level Distribution”表分析的默认路径数量为1,000条，可在“Report Design Analysis”对话框中更改此数量。默认情况下不生成“Logic Level Distribution”表，但通过选择“Report Design Analysis”对话框的“Options”选项卡中的“Include logic level distribution”即可生成该表。“Logic Level Distribution”表的示例如下图所示。

图 100：“Logic Level Distribution”报告示例

End Point Clock	Requirement	0	1	3	4	5	11	12	14	15	16
cpuClk_3	10.000ns	0	0	0	0	0	73	607	10	2	4
phyClk0_2	10.000ns	0	3	19	109	0	0	0	0	0	0
phyClk1_1	10.000ns	1	11	0	73	0	0	0	0	0	0
usbClk	10.000ns	0	1	75	7	5	0	0	0	0	0

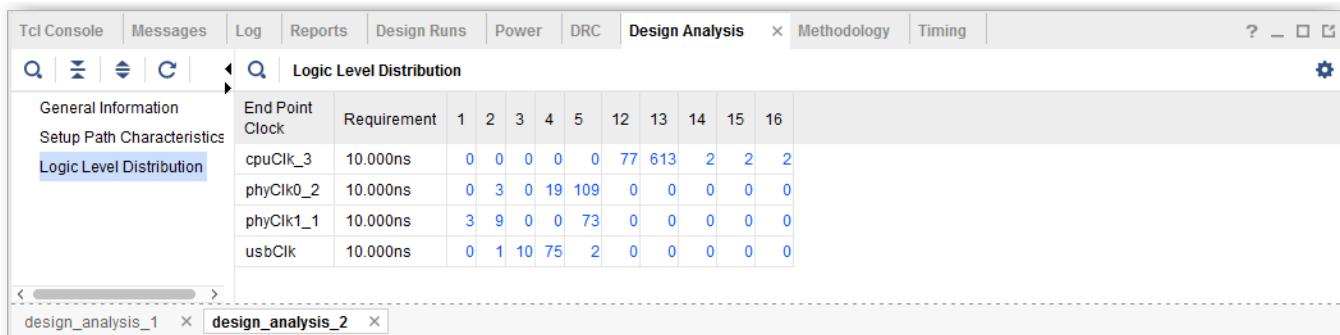
Logic Level Distribution GUI已实现功能增强，包含对应各bin文件的独立超链接。通过单击这些超链接，即可在路径上运行`report_design_analysis`或`report_timing`，或者选择时序路径，如下图所示。

图101：选定路径的“Report Design Analysis”



命令行选项 `-routes` 可与 `-logic_level_distribution` 搭配使用，以便基于布线数量而不是逻辑层次数量来生成报告。

图102：使用 -routes 的“Logic Level Distribution”报告示例

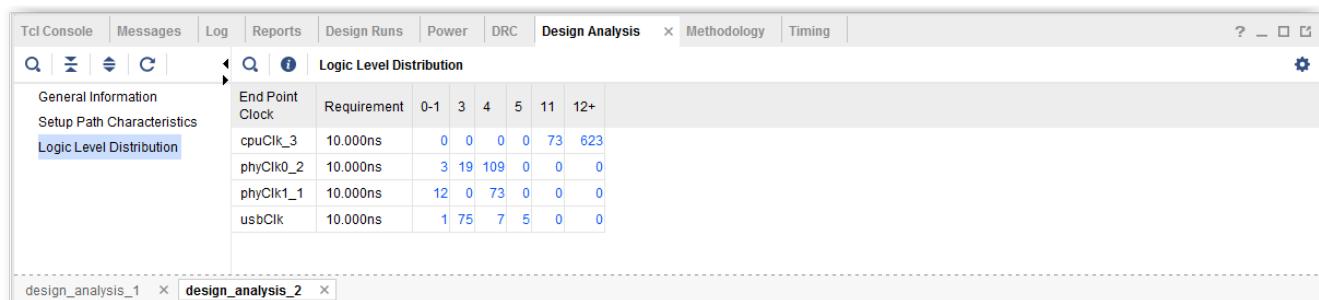


命令行选项 `-min_level` 和 `-max_level` 可与 `-logic_level_distribution` 搭配使用以控制 bin 文件。

逻辑级数小于 `-min_level` 的所有路径都置于单个 bin 文件内，逻辑级数大于 `-max_level` 的所有路径也都置于单个 bin 文件内。

为每个逻辑层次创建一个独立 bin 文件，使逻辑层次间存在至少 1 条路径。例如，如果设计包含的路径的逻辑级数为 0、1、3、4、5、11、12、14、15、16（请参阅[时序路径特性报告](#)）并使用 `-min_level 3` 和 `-max_level 11`，那么 `report_design_analysis` 会使用分别以下 bin 文件生成报告：0-2、3、4、5、11 和 12+。

图103：使用 -min_level 和 -max_level 的“Logic Level Distribution”报告示例



分析特定路径

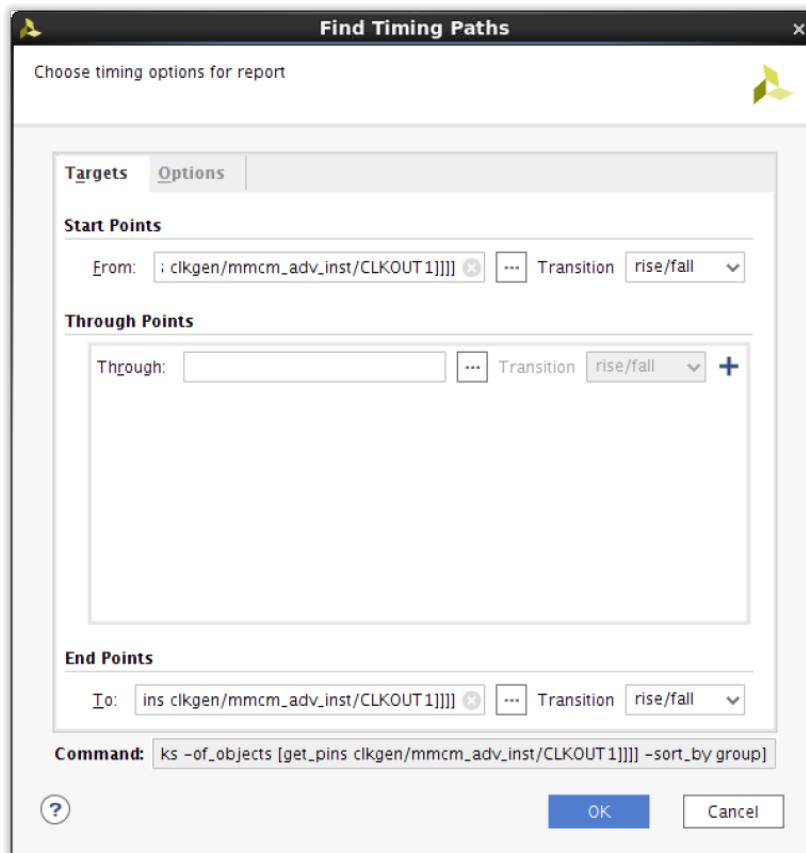
[分析特定路径](#)显示了来自“Report Design Analysis”的报告示例，采用“时序模式(Timing Mode)”，选定特定路径。

图 104：特定时序路径特性示例

Name	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Clock Relationship	Logic Levels	Routes	Logical Path	Start Point Clock	End Point Clock
Path 1	20	9.84	0.266	9.574	-0.335	9.415	Safely Timed	1	1	FDRE LUT6 RAMB36E1	wbClk	wbClk
Path 2	20	9.467	0.266	9.2	-0.343	9.780	Safely Timed	1	1	FDRE LUT6 RAMB36E1	wbClk	wbClk
Path 3	20	7.567	0.266	7.301	-0.324	11.699	Safely Timed	1	1	FDRE LUT6 RAMB36E1	wbClk	wbClk
Path 4	20	7.015	2.2	4.815	-0.077	12.626	Safely Timed	7	8	RAMB36E1 LUT5 LUT6 LUT6 LUT6 LUT5 LUT4 LUT5 FDCE	wbClk	wbClk
Path 5	20	7.015	2.2	4.815	-0.077	12.626	Safely Timed	7	8	RAMB36E1 LUT5 LUT6 LUT6 LUT6 LUT5 LUT4 LUT5 FDCE	wbClk	wbClk
Path 6	20	7.015	2.2	4.815	-0.077	12.626	Safely Timed	7	8	RAMB36E1 LUT5 LUT6 LUT6 LUT6 LUT5 LUT4 LUT5 FDCE	wbClk	wbClk

在本例中，“路径特性(Path Characteristics)”和“逻辑层次分布(Logic Level Distribution)”表（如果已选中）已限制为指定路径。要指定路径，请单击“Report Design Analysis”对话框中的所选特定路径右侧的“Browse”按钮。这样会打开“查找时序路径(Find Timing Paths)”对话框（如下图所示）。

图 105：“Find Timing Paths”对话框



结合前后最差路径执行最差路径分析

下图显示了“时序模式 (Timing Mode)”下已选中“扩展分析 (Extend analysis)”选项情况下的“设计分析报告 (Report Design Analysis)”的报告示例。

注释：“针对所有路径扩展分析 (Extend Analysis for All Paths)”选项当前仅适用于建立时间分析。

“报告特性 (Path Characteristics)”报告中包括最差建立路径、止于起点单元的最差建立路径 (PrePath) 和始于端点单元的最差建立路径 (PostPath)。-extend 选项产生的运行时间更长，因为需执行多项时序分析以收集所有已报告路径的特性。

等效的 Tcl 命令：report_design_analysis -extend

图 106：最差建立路径的扩展路径特性

Name	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Clock Relationship	Logic Levels	Routes	Logical Path	Start Point Clock	End Point Clock	DSP Block	B
PrePath 1	10	1.206	0.559	0.647	-1.521	7.072	Safely Timed	1	1	IBUF FDRE	sysClk	wbClk	None	N
Path 1	10	8.968	0.223	8.745	-0.377	0.15	Safely Timed	0	1	FDRE FDSE	wbClk	phyClk1_1	None	N
PostPath 1	10	2.547	0.585	1.962	-0.267	7.056	Safely Timed	5	3	FDSE LUT6 MUX7 MUXF8 LUT6 LUT6 FDRE	phyClk1_1	usbClk_3	None	N
PrePath 2	10	1.206	0.559	0.647	-1.521	7.072	Safely Timed	1	1	IBUF FDRE	sysClk	wbClk	None	N

读取和解释时序路径特性报告

路径特性可分为 4 大类：时序、逻辑、物理和属性。您可通过命令行帮助功能查看每项特性的定义。

Tcl 命令：report_design_analysis -help

或者，也可通过《Vivado Design Suite Tcl 命令参考指南》(UG835) 找到这些信息。

第1类：时序

- 时序分析 (Timing Analysis)：“路径类型 (Path Type)”和“要求 (Requirement)”详列了时序分析类型 (SETUP 或 HOLD) 和时序路径要求。“裕量 (Slack)”指示根据时序约束规定的时序分析，时序路径要求是否已得到满足。“时序例外 (Timing Exception)”指示是否已将任何时序例外（例如，多周期路径或最大延迟）应用于时序路径。
检查路径要求通常是对时序约束缺失或不正确问题进行调试的第一步。
 - 对于建立时间要求低于 4 ns 的路径，必须加以审查并验证其在设计中的有效性，对于时钟域交汇路径尤其如此。
 - 对于建立时间要求小于 2 ns 的路径，通常此类路径难以满足时序，因而必须避免，对于早期架构尤其如此。
 - 通常如果建立时间要求较小，请检查是否缺失时序例外约束，并检查源和目标时钟沿。时序分析始终假定源与目标时钟沿之间存在最小的正差值，除非被时序例外约束覆盖。
 - 对于保持路径要求为正值的要求，必须加以审查，因为此类要求不常见且难以满足。当保持路径要求为正值时，请检查对于可能仅应用于建立分析路径的保持分析，是否缺失多周期路径约束。此外，还需检查源时钟与目标时钟之间的关系是否正确。
- 数据路径 (Datapath)：“路径延迟 (Path Delay)”、“逻辑延迟 (Logic Delay)”和“信号线延迟 (Net Delay)”详列了数据路径延迟总量及其细分，按逻辑单元和信号线显示其所占延迟比例。
 - 如果 “Logic Delay” 在数据路径延迟总量中所占比例异常高，例如，达 50% 甚至更高，那么建议检查数据路径逻辑深度和逻辑路径上的单元类型，可能需要修改 RTL 或综合选项以减少路径深度或者使用延迟较小的单元。

- 如果“Net Delay”在建立路径的路径延迟总量中占绝对主导地位，且“Requirement”合理，则建议对本部分中所列的路径的部分物理特性和属性特性进行分析。具体需要查看的项包括“高扇出 (High Fanout)”和“累积扇出 (Cumulative Fanout)”特性，以便了解路径中的部分信号线是否存在高扇出，以及是否可能导致出现布局问题。此外，还要检查“Hold Fix Detour”特性，以了解在路径上是否发生了保持修复。



重要提示！ LUT 输入管脚具有不同的延迟特性。索引值较高的物理管脚（或站点管脚）比索引值较低的管脚更快。请注意 7 系列与 UltraScale 器件 LUT 延迟报告之间的差异。在 7 系列器件中，LUT 延迟的可变部分作为 LUT 前的信号线延迟的一部分来进行报告。在 UltraScale 器件中，LUT 延迟的可变部分作为逻辑延迟来报告。因此，7 系列器件的 Net Delay/Logic Delay 比值将大于 UltraScale 器件的比值。

- 时钟 (Clocks)：有关时序路径时钟的“起点时钟 (Start Point Clock)”、“端点时钟 (End Point Clock)”、“时钟关系 (Clock Relationship)”和“时钟偏差 (Clock Skew)”详细信息。“Start Point Clock”和“Endpoint Clock”分别列出了对应的时序路径源时钟和目标时钟。
 - 检查“时钟关系 (Clock Relationship)”是否正确且符合预期。对于时钟内部路径或同步时钟域交汇路径，关系标记为“已安全定时 (Safely Timed)”。您必须验证“Requirement”和“Clock Skew”是否合理。对于异步时钟，此关系标记为“无公共基准时钟 (No Common Primary Clock)”、“无公共周期 (No Common Period)”、“无公共节点 (No Common Node)”或“无公共相位 (No Common Phase)”。时序例外必须覆盖异步时钟域交汇路径（检查“Timing Exception”值）。
 - 检查“Clock Skew”是否合理。在分析时钟偏差时，请检查级联时钟缓存的时钟树结构。在 7 系列器件中，请检查源时钟和目标时钟的不同时钟缓存类型。在 UltraScale 中，可能需要检查时钟信号线的布局布线，因为它取决于逻辑负载布局。跨越时钟区域边界或 I/O 列可能导致时钟偏差增大，这符合预期。

注释：在时序报告中可提供 report_design_analysis 所提供的几乎所有“时序特性”。

第2类：逻辑

- 路径：“起点管脚原语 (Start Point Pin Primitive)”、“端点管脚原语 (End Point Pin Primitive)”、“起点管脚 (Start Point Pin)”、“端点管脚 (End Point Pin)”、“逻辑层次 (Logic Levels)”、“逻辑路径 (Logical Path)”和“布线 (Routes)”可提供有关时序路径的部分基本信息。
 - “Start Point Pin Primitive”和“End Point Pin Primitive”是时序路径起点和端点的参考管脚名称。请检查“Start Point Pin Primitive”和“End Point Pin Primitive”是否是期望的时序路径起点和端点。“Start Point Pin”和“End Point Pin”用于识别实际的时序路径起点和端点，这些起点和端点将显示在典型时序报告的标题中。

请检查可能包含在控制信号的高扇出信号线中的端点管脚（例如，CLR、PRE、RST 和 CE），例如，异步复位和时钟使能信号。另请检查单元类型，因为部分原语（如块 RAM 和 DSP）具有比其它单元更大的时钟输出 (Clock-to-Q) 延迟和建立/保持时间要求。如果出现在路径上，这些单元可能占用大量路径时序预算。

- “Logic Levels”和“Logical Path”详列了逻辑级数以及数据路径中的原语类型。“Routes”表示数据路径中可布线的信号线数量。您可使用此信息快速检查大量逻辑层次源于 LUT 还是源于 LUT/CARRY/MUXF 单元混合。CARRY 和 MUXF 单元通常连接到具有专用布线（延迟较小或为空）的信号线，而 LUT 输入的布线则始终需穿越结构。

虽然路径基本上已经对 LUT 进行了约束，但检查 LUT 大小同样是很重要的。请尝试理解多个较小的 LUT（非 LUT6）链接在一起的原因，以及导致综合无法仅以 LUT6 为目标的因素，这样可能可以减少逻辑级数。路径中可能存在 KEEP/DONT_TOUCH/MARK_DEBUG 之类的属性或中高扇出信号线导致影响映射效率。

根据分析结果，您可修改 RTL 源、添加/修改 RTL 中的属性，或者使用其它综合设置来减少路径上的 LUT 数量。另外，也可以使用 opt_design 命令的 -remap 选项来对 LUT 映射重新进行最优化，这可能可消除部分较小的 LUT。

- 单元：数据路径中存在的 DSP 块和 BRAM。在来自不含输出寄存器且含有数个逻辑层次的 RAMB 或 DSP 的路径上，满足时序的难度较高。如果这些路径难以满足时序要求，那么您应考虑将自己的设计修改为使用 RAMB 或 DSP 输出寄存器。

第3类：物理

- 架构边界交汇：“IO 交汇”和“SLR 交汇”用于识别路径是否跨越架构资源（例如，IO 列或 SLR 边界）。

跨越大量架构列并不总是意味着存在问题。请检查信号线延迟过高或偏差过大是否与跨越大量架构列存在关联。如果跨越大量架构列可能是导致特定模块内多个实现运行间的时序问题的原因，请考虑使用 Pblock 来最大限度减少布局规划，从而减少架构列或 SLR 边界交汇。

- 路径布局限制：Pblock。过度布局规划有时可能会阻碍工具达到最理想结果。跨越多个 Pblock 的路径有时会遇到时序问题。
 - 如果路径跨越多个 Pblock，请检查 Pblock 的位置及其对时序路径布局造成的影响。
 - 如果 Pblock 彼此相邻，应考虑为每个单独 Pblock 创建单个 Pblock 作为其超集。这样可以放宽对布局器的限制，从而改善时序。

如果物理要求规定 Pblock 分散布局，请考虑在 Pblock 间采用流水线化来帮助满足时序要求。

- 布局框：边界框大小、时钟区域距离、合并 LUT 对
 - 如果时序路径的边界框大小或时钟区域距离过大，请尝试使用 place_design 中的指令。在 UltraScale 器件中，请注意“时钟区域距离”及其可能对时序路径“时钟偏差”产生的影响。
- 信号线扇出和绕行：
 - “高扇出 (High Fanout)” 用于显示数据路径中所有信号线的最高扇出，而“累积扇出 (Cumulative Fanout)” 对应于所有数据路径信号线扇出总和。

如果“High Fanout”和“Cumulative Fanout”值都较大，鉴于扇出会影响布线和信号线延迟，因此出现时序违例的概率很大。

如果已运行物理最优化，但未降低扇出，请检查 MARK_DEBUG 和 DONT_TOUCH 约束，防止复制。

如果实现前需要在信号线上执行复制，可在综合中（在 RTL 内部或 XDC 文件中）使用 MAX_FANOUT 约束。由于依靠布局实现高扇出信号线的有效时序，因此通常不建议在综合中执行复制，最好依靠布局后的物理最优化 (phys_opt_design) 来执行复制。使用不同指令（如，Explore、AggressiveExplore 或 AggressiveFanoutOpt）还可增强物理最优化，从而对含少量正裕量的路径也一并执行最优化。

如果实现期间需要在特定信号线上减少扇出，可以使用以下命令强制执行复制：phy_opt_design -force_replication_on_nets <netName>

- 如果断言“保持修复绕行 (Hold Fix Detour)”，则表示数据路径上的布线已延迟，以满足路径保持时间要求。如果路径未满足建立时间要求，请检查源时钟和目标时钟之间偏差是否过大。此外还需检查源时钟和目标时钟之间的时序约束是否正常，以避免保持路径要求为正值（大部分情况下该值应为 0 或负值）。

第 4 类：属性

- LUT 组合：“Combined LUT Pairs” 表示路径中存在组合的 LUT 对。虽然组合 LUT 对可降低逻辑利用率，但也会限制布局解决方案，可能由于管脚密度过高而导致拥塞。如果怀疑设计中存在 LUT 组合问题，建议在综合中使用 -no_lc 选项来禁用 LUT 组合。
- 最优化阻塞：“Mark Debug” 和 “Dont Touch” 可快速识别路径中是否存在不允许工具对其执行最优化的任何信号线或单元。
 - 设置 MARK_DEBUG 属性的默认行为是同时设置 DONT_TOUCH 属性。请考虑将 DONT_TOUCH 设置为 FALSE 以允许执行最优化。
 - DONT_TOUCH 禁用单元或信号线复制等最优化。请评估是否需要 DONT_TOUCH 约束，如可行，请将其移除。当信号线进入含 DONT_TOUCH 的层级单元时，层级单元内的部分信号线无法执行复制。如果使用 DONT_TOUCH 来阻止逻辑裁剪，请检查设计是否正确。由于未连接的输出而移除的逻辑就是一个简单的示例。
- 固定布局布线：“固定逻辑和固定布线” 可快速识别是否存在可能影响时序路径裕量的任何固定布局或布线约束。
 - 使用单元位置约束有助于稳定高难度设计的 QoR。如果修改设计后无法再满足时序，可尝试移除布局约束，以便为布局器提供更大的灵活性。

- 使用固定布线会阻止布线器最优化信号线延迟以满足时序。含锁定布线的时序路径常常与其它路径共享信号线，而这些路径可能受到此约束的负面影响。仅在必要时且不影响交互路径的前提下使用固定布线。请务谨记，对其他 Pblock 等物理约束执行更改可能还需要更新固定的单元位置或固定布线。

第 5 类：部分重配置

对于部分重配置 (PR) 设计，将追加逻辑路径以识别单元是属于可重配置分区 (:RP#) 还是属于设计的静态区域 (:S)。位于报告底部的转换表可将 :RP# 映射到特定的可重配置分区。

- PR 路径类型：指定路径属于完全处于静态区域中、完全处于可重配置分区 (RP) 中还是跨越区域间的边界。时序路径的延迟元素同样也细分到各区域内。
- 静态交汇：报告 RP 路径穿越进入静态区域的次数。
- RP 交汇：报告静态区域路径穿越进入 RP 区域的次数。
- 边界扇出：报告位于 PPLOC 的边界路径扇出到其下游负载的行为。

设计 QoR 汇总

命令行选项 `-qor_summary` 可用于为流程中每个步骤生成 QoR 汇总信息。该选项只能从 Tcl 控制台使用。

```
report_design_analysis -qor_summary
```

图 107：设计分析报告 QoR 汇总

Report Design Analysis						
Table of Contents						
1. Design QoR Summary						
1. Design QoR Summary						

+	Task Name	Options	Directives	Wns(ns)	Tns(ns)	Runtime(mins)
+	synth_design					4
+	opt_design					0
+	place_design			0.562		0
+	route_design			1.305	0.000	0
+	* Data is not available for the empty fields					

复杂性报告

对于顶层设计和/或包含 1000 个以上叶节点单元的层级单元，复杂性报告会显示每个叶节点单元类型的 Rent 指数、“平均扇出 (Average Fanout)” 和分布。Rent 指数是指在使用最小割 (min-cut) 算法以递归形式对设计进行分区时，网表分区的端口数量和单元数量之间的关系。其计算方法与在全局布局期间布局器所使用的算法类似。因此，它可明确指出布局器所遇到的困难，当设计层级与全局布局期间发现的物理分区精确匹配时尤其如此。

Rent 指数根据 Rent 规则定义如下：

$$\text{ports} = \text{constant} \times \text{cells}^{\text{Rent}}$$

$$\log(\text{ports}) = \text{Rent} \times \log(\text{cells}) + \text{constant}$$

Rent 指数较高的设计表示此类设计中包含逻辑紧密相连的分组，并且这些分组与其它分组同样连接紧密。这通常可理解为全局布线资源利用率较高并且布线复杂性也更高。此报告中提供的 Rent 指数是根据未布局和未布线的网表来计算的。

完成布局后，相同设计的 Rent 指数可能改变，因为它基于物理分区而不是逻辑分区。Report Design Analysis 命令不会报告布局后的 Rent 指数，因为建议改为在设计完成布局后再执行拥塞报告分析。

执行以下任一操作时，将以“复杂性模式 (Complexity Mode)”来运行“设计分析报告 (Report Design Analysis)”：

- 在“Report Design Analysis”对话框的“Options”选项卡中选中“Complexity”选项。
- 执行 `report_design_analysis` Tcl 命令，并使用下表中所示任意选项。

表 5：在“Complexity Mode”下运行“Report Design Analysis”的选项

Tcl 选项	说明
<code>-complexity</code>	在“Complexity Mode”下运行“Report Design Analysis”时必须指定此项。
<code>-cells <arg></code>	用于指定分析复杂性时要使用的层级单元。
<code>-hierarchical_depth <arg></code>	默认情况下在顶层检查的层级，或在 <code>-cells</code> 选项所指定的单元层次检查的层级。

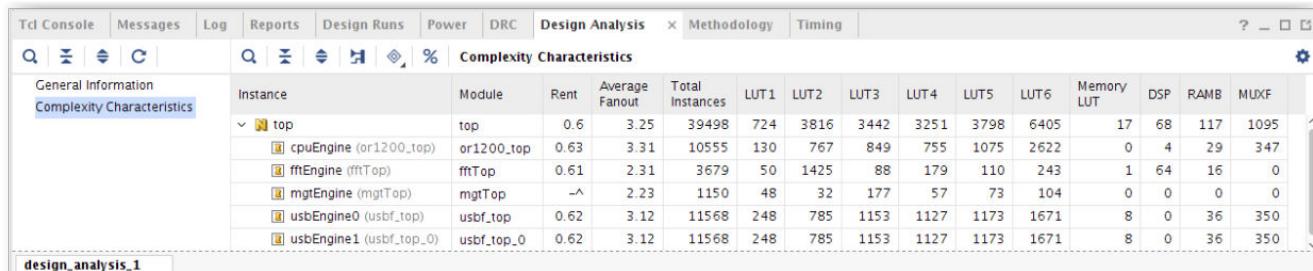
分析顶层设计复杂性

下图显示的示例报告来自“复杂性模式 (Complexity Mode)”下的“设计分析报告 (Report Design Analysis)”，最多可报告至顶层模块下一层级。

Tcl 命令：

```
report_design_analysis -complexity -hierarchical_depth 1
```

图 108：顶层和层级深度为 1 的层级的复杂性分析



读取和解释复杂性报告

前例所示的“复杂性特性 (Complexity Characteristics)”表显示了顶层以下每个层级的 Rent 指数和平均扇出。一般情况下，复查这些指标时需要考量的范围包括：

- Rent 指数：
 - 介于 0.0 到 0.65 之间：判定复杂性处于较低到正常水平，不会导致任何潜在问题。
 - 介于 0.65 到 0.85 之间：判定复杂性处于较高水平，当实例数量超过 25k 时尤其如此。
 - 高于 0.85：复杂性非常高，如果实例数量同样很高，那么实现期间设计可能失败。
- 平均扇出：
 - 低于 4：判定为正常。

- 介于 4 到 5 之间：实现工具可能在布局设计时无法避免拥塞。就 SSI 器件而言，当实例总数大于 100k 时，布局器将难以找到能布局在 1 个 SLR 内或分布到 2 个以上 SLR 的解决方案。
- 高于 5：实现期间设计可能失败。

对于重要性较高的大型模块，必须解决 Rent 指数过高和/或平均扇出过高的问题。对于较小的模块（尤其是实例总数低于 10k 的情况下），Rent 指数和平均扇出可能较高，但仍可轻松成功完成布局布线。因此，必须始终将“实例总数 (Total Instances)”列与 Rent 指数和平均扇出结合在一起进行审查。

复杂性特性有时可能无法预测布线拥塞。诸如目标器件中的 I/O 位置约束、布局规划和宏原语位置等其它因素还会限制布局解决方案的空间，从而引起拥塞。完成布局后才能通过拥塞报告对此类约束的影响进行更准确的分析。

在解读“Complexity Characteristics”表时需要考虑的其它事项：

- 模块中 LUT6 所占比例的增高通常会导致平均扇出增大，从而可能导致 Rent 指数增大。
- 大量 RAMB 和 DSP 的存在可能导致 Rent 指数增大，因为这些原语都具有大量连接。
- Rent 指数较高或者平均扇出较高的层级实例有时不会导致问题，因为布局器在平面网表上工作，可将这些实例细分为更便于布局的逻辑组。如果存在明显不同寻常的模块，此报告可提供有关可能存在网表问题的区域的指示信息。

当某个大型模块的 Rent 指数和/或平均扇出较高并导致拥塞和时序问题时，请考虑执行如下操作：

- 减少模块的连接。保留层级，防止在综合内执行跨边界最优化，这样可减少 LUT6 的使用，从而降低网表密度。
- 尝试在综合中禁用 LUT 组合。
- 在实现期间使用“拥塞策略 (Congestion Strategy)”或者使用 SpreadLogic 布局指令，这可能有助于缓解拥塞。如果设计目标为 SSI 器件，请尝试采用多种不同 SSI 布局指令。
- 在 SLR 层次针对 SSI 器件或者在通用时钟区域层次使用简单的布局规划使拥塞的逻辑组保持独立，或者引导全局布局向类似先前明确的良好布局相似的方向发展。

“拥塞 (Congestion)” 报告

“Congestion” 报告用于显示器件的拥塞区域以及这些区域内存在的设计模块名称。如果在拥塞区域内部或附近布局关键路径，那么拥塞可能引发时序收敛问题。

分析设计拥塞

要在“拥塞模式 (Congestion Mode)”下运行“Report Design Analysis”，必须在“Report Design Analysis”对话框的“选项 (Options)”选项卡中指定“拥塞 (Congestion)”选项，并且必须已完成设计的布局和/或布线。在未布局的设计上以“Congestion Mode”运行“Report Design Analysis”将导致报告不含任何内容。

“Report Design Analysis” 可生成 3 个拥塞表：

- [布局器最终拥塞报告 \(Placer Final Congestion Reporting\)](#)
- [布线器初始拥塞 \(Router Initial Congestion\) 报告](#)
- [SLR 信号线交汇 \(SLR Net Crossing\) 报告](#)

“最大拥塞 (Maximum Congestion)” 报告

这些表格用于报告特定方向上发现的具有相同最大拥塞级别的所有窗口。其中各列具体定义如下：

- 方向 (Direction)：发生拥塞的资源的方向（北 (North)、南 (South)、西 (West) 或东 (East)）。

- 拥塞级别 (Congestion Level): CLB 拼块中的最大拥塞级别。
- 拥塞 (Congestion): 指示定义的窗口中估算的布线资源利用率。该值可大于 100%。
- 拥塞窗口 (Congestion Window): 指示针对指定方向存在拥塞的绑定 CLB 拼块。CLB 坐标对应于窗口的左下角和右上角。



提示: “Congestion Window” 列仅显示在文本报告中。在 GUI 报告中，您可选择拥塞窗口，这将在“Device”窗口中高亮拥塞区域。

- 单元名称 (Cell Names): 指示包含“Congestion Window”中所涉及的层级单元的父实例，包含前 3 大拥塞实例及其各自拥塞占比。



提示: 在 GUI 报告中，可选择含超链接的单元名称以在“Congestion Window”中高亮相应的叶节点单元。

- 平均 LUT 输入 (Avg LUT Input): 这是拥塞窗口内 LUT 的平均 LUT 输入。
- 组合 LUT (COMBINED LUTs) (以 % 为单位) : 指示窗口内组合 LUT 的百分比。
- LUT 利用率 (LUT usage) (以 % 为单位) : 窗口内 LUT 利用率。
- LUTRAM 利用率 (LUTRAM usage) (以 % 为单位) : 窗口内 LUTRAM 利用率。
- 触发器利用率 (Flop usage) (以 % 为单位) : 窗口内 FD (含 LD) 利用率。
- MUX 利用率 (MUX usage) (以 % 为单位) : 窗口内 MUXF 利用率。
- RAMB 利用率 (RAMB usage) (以 % 为单位) : 窗口内 RAMB 利用率。
- URAM 利用率 (URAM usage) (以 % 为单位) : 窗口内 URAM 利用率。
- DSP 利用率 (DSP usage) (以 % 为单位) : 窗口内 DSP 利用率。
- CARRY 利用率 (CARRY usage) (以 % 为单位) : 窗口内 CARRY 利用率。
- SRL 利用率 (SRL usage) (以 % 为单位) : 窗口内 SRL 利用率。

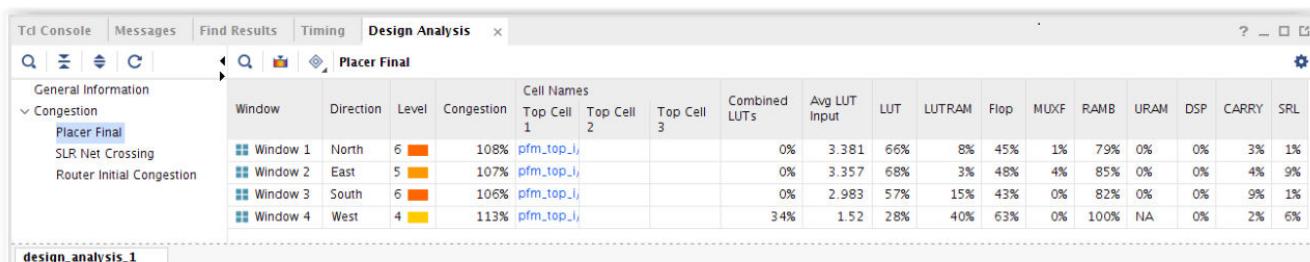
布局器最终拥塞报告 (Placer Final Congestion Reporting)

分析设计的“Placer Final Congestion Reporting”表的“拥塞 (Congestion)”和“时序 QoR (Timing QoR)”情况时，请留意如下信息：

- 如果 LUT 利用率较高，请检验实例的“复杂性 (Complexity)”报告中是否存在较高百分比的 LUT6。
- 如果拥塞区域中 RAMB 或 DSP 利用率较高，请检查 Pblock 约束，此类约束可能限制报告的模块的可用布局区域。使用各种针对性的布局指令可缓解拥塞，例如，BlockPlacement 或 SpreadLogic 指令。在某些情况下，最好复用来自先前运行的拥塞较低且生成良好“Timing QoR”的 RAMB 或 DSP 布局。

下图显示的是“Placer Final Congestion Reporting”表示例。通过使用该报告即可检验“Congestion”窗口定义的器件区域以及该窗口内存在的模块。资源利用率 (%) 表明了位于拥塞区域内的资源类型。

图 109: “Placer Final Congestion Reporting”表示例



“布线器初始拥塞 (Router Initial Congestion)” 报告

仅当已运行布线器时，“Router Initial Congestion”（针对 7 系列 FPGA 名为“Initial Estimated Router Congestion”）才可用。它可显示布线早期阶段布线器最初面临的布线拥塞情况。

图 110：“Router Initial Congestion 报告”表示例

Window	Direction	Type	Level	Combined LUTs	Avg LUT Input	LUT	LUTRAM	Flop	MUXF	RAMB	URAM	DSP	CARRY
INT_X34Y104->INT_X41Y215 (CLEL_R_X41Y215)	South	Global	5	25%	4.09	83%	2%	57%	1%	13%	NA	0%	14%
INT_X32Y419->INT_X63Y482 (BRAM_X32Y415->CLEL_R_X63Y482)	North	Long	6	18%	3.75	80%	12%	43%	0%	23%	0%	5%	1%
INT_X48Y419->INT_X63Y546 (CLEM_X48Y419->CLEL_R_X63Y546)	North	Long	6	24%	3.62	82%	6%	46%	4%	38%	0%	1%	2%
INT_X36Y369->INT_X67Y464 (CMT_L_X36Y360->CLEL_R_X67Y464)	South	Long	6	17%	3.59	80%	6%	43%	1%	10%	0%	0%	5%

当拥塞等级不低于 5 时，`report_design_analysis` 会生成拥塞表以提供有关与特定方向和类型内最严重的拥塞相关联的拥塞性质以及区域的详细信息。

- 全局拥塞的估算方式与布局器拥塞相似，根据所有互连类型来进行估算。
- 长拥塞只考虑给定方向的长互连利用率。
- 短拥塞则考虑给定方向的所有其它互连利用率。

大于 32x32（5 级）的所有拥塞区域都可能会影响 QoR 和可布线性。长互连上的拥塞会导致短互连利用率增加，从而导致布线延迟增加。短互连上的拥塞通常会导致运行时间延长，如果窗口大小过大，则可能导致 QoR 劣化。

分析“Router Initial Congestion”表时，请注意：

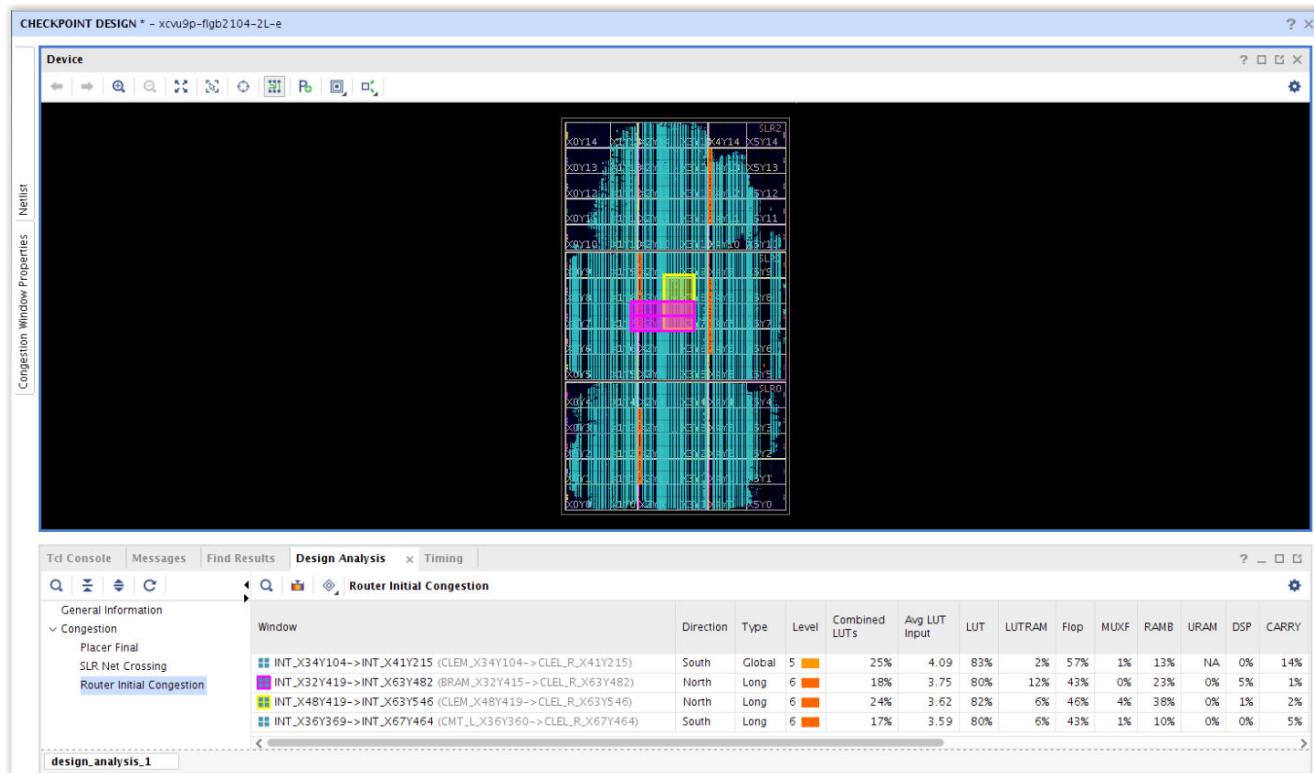
- 当拥塞等级大于 6 时，设计满足时序要求的概率很低，且可能在布线期间失败。
- 当拥塞等级为 4 或 5 时，请识别位于拥塞区域内的模块。您可对这些模块应用拥塞缓解技巧，或者使用其它指令（例如，`*SpreadLogic*`）重新运行布局器。
- 当拥塞等级不超过 3 时，拥塞可能不会导致问题，除非设计的时序预算非常紧凑。

在上图所示的“Router Initial Congestion”示例中，报告了拥塞等级不小于 5 的区域。要以更低的拥塞阈值生成拥塞报告，请使用 `-min_congestion_level` 开关。默认最小拥塞等级为 5。该值必须介于 3 到 8 之间。

拥塞报告包含如下区域：设计中给定方向和类型内拥塞达最高等级的区域，以及该给定方向和类型内拥塞达最大等级的其它区域（如果有）。这些区域可能存在重叠，或者可能存在于器件的不同区域内。

在下图所示的示例中，设计的多个区域内显示在 North（方向）和 Long（类型）中拥塞等级达到 6。

图111：“Router Initial Congestion 报告”表示例

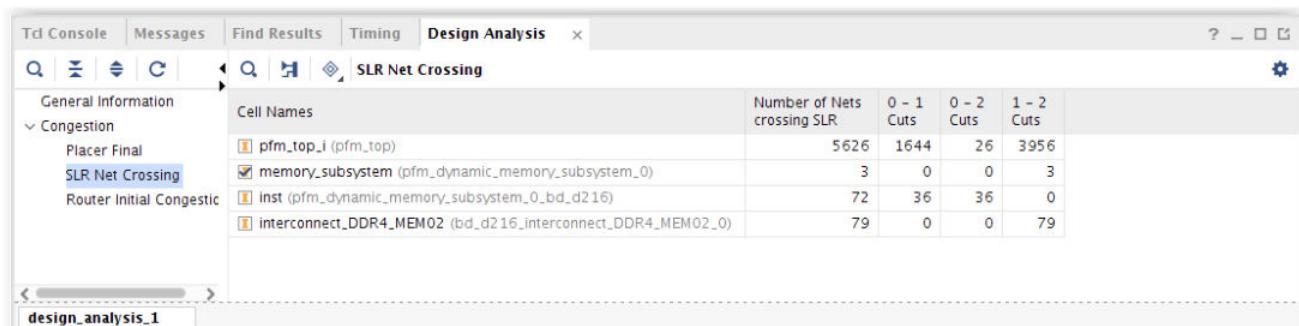


“SLR 信号线交汇 (SLR Net Crossing)” 报告

“SLR Net Crossing” 报告仅适用于 SSI 器件，可报告跨越 SLR 边界的模块中包含的信号线数量。对于每个模块，该表可提供有关信号线跨越的 SLR 的进一步详情。下图显示了“SLR Net Crossing” 报告表示例。

注释：当信号线负载遍布多个 SLR 时，仅按最远的切割计算 1 次跨越。例如，对于从 SLR0 驱动到 SLR1、SLR2 和 SLR3 中的负载的信号线，在 0-3 切割下仅计作 1 次跨越，且 SLR3 为距离 SLR0 最远的扇出。这种计数方法支持对每一列（0-1 切割、1-2 切割、以此类推）下的信号线数量求和，以与信号线跨越总数相匹配，因为每个信号线仅计数 1 次。

图112：“SLR Net Crossing” 报告表示例



分析设计的“SLR Net Crossing”报告表的“拥塞 (Congestion)”和“时序 QoR (Timing QoR)”情况时，请留意如下信息：

1. 使用SSI器件时，SSI布局指令可能有助于解决时序和拥塞。
2. 如果跨越SLR的特定模块在使用各布局指令执行的多次实现运行期间持续遇到时序问题，请尝试减少使用Pblock以将模块约束到单一SLR。

QoR评估报告 (Report QoR Assessment)

`report_qor_assessment`命令可生成文本报告以提供如下信息：

- 评估得分，用于指示其设计满足性能目标的概率。
- 有关建议的后续步骤的流程指南。
- 利用率和性能指标汇总信息。
- QoR的关键方法检查汇总信息。

随着设计收敛流程的执行，需采用多种不同策略以使设计达成收敛。掌握在实现流程中解决问题的顺序以及更改方法的时机并不容易，即使经验丰富的设计人员也可能犯错。错失切换时机则可能导致实现时间出现不必要的增加和实现运行的分析周期额外增加。例如，解决拥塞问题时必须首先解决关键时钟设置问题。一步错则步步错，导致事倍功半。

下图简单展示了RQA与设计中的变更进行交互的方式。

图113：RQA与设计变更的交互



X23299-031720

典型设计的设计收敛流程往往难以顺畅进行。发生更改时，可能会跳过某些阶段或者从头重来。RQA可用于重新评估设计，并随时提供更新的流程指南。

在早期阶段，RQA可与`report_methodology`紧密配合。在流程进行期间，运行`report_qor_suggestions`可生成自动建议，以提升RQA得分。当设计接近实现时序收敛时，焦点将转至如何最大限度发挥实现工具的作用。RQA将建议采用通过检验设计所生成的最优实现策略来运行，并使用机器学习算法来推荐策略。最后，将启动增量建议为最后一步的时序收敛尽一份力，以帮助维持QoR并加速实现。

综合后可随时运行 RQA 命令。在流程中越早运行该命令，RQA 评分的准确性将越低。但此得分较最终布线后得分高应不超过 1。仅当运行完整的布局布线后，才会提供 ML 策略或增量流程指导。

总体评估汇总

汇总包含 QoR 评估得分和流程指南。

评估得分用于判定您应继续改进设计（RTL、约束等），还是继续执行实现运行。此得分是通过分析一组复杂的设计指标来生成的，这些指标涉及 Ultrafast 方法、器件利用率、控制集、时钟设置、建立裕量和保持裕量等领域。此外，其中还考量了特定于器件的特征。评分范围为从 1 至 5。小于 5 时，应使用 `report_qor_suggestions` 提升评分。

在下表中提供了评分详细信息：

表 6：QoR 评估报告评分

得分	含义
1	设计实现流程的概率较低
2	设计能完成实现流程，但无法满足时序
3	设计满足时序概率较低
4	设计可能能够满足时序
5	设计将轻松满足时序要求

流程指南属于“总体评估汇总”的一部分。它可提供如下相关信息：

- 方法状态 (Methodology Status)。
- 设计是否可采用 ML 策略。
- 设计是否可采用增量编译。
- Vivado® 为您提供的后续操作建议，包含多个选项。
 - 包含使用 RQS 提升 RQA 得分。

下图显示了 QoR 评估得分为 2 的设计示例：

图 114：总体评估汇总

```
1. Overall Assessment Summary
-----
+-----+
| QoR Assessment Score | 2 - Implementation may complete. Timing will not meet |
+-----+
| Methodology Status    | Failed with warnings |
+-----+
| ML Strategy Compatible | No |
+-----+
| Incremental Compatible | No |
+-----+
| Next Recommended Flow Stage | Run report_qor_suggestions and follow suggestion guidelines |
+-----+
```

方法检查不合格表示某些路径存在困难，需解决其中问题。RQS 在此情况下将可用于识别此类路径，并且可能通过建议来解决时序失败。此处建议的“后续建议流程阶段 (Next Recommended Flow Stage)”为运行 `report_qor_suggestions`。对于更严重的方法警告，建议解决问题或者将其豁免。

ML策略仅限于完成布局布线后使用。这些策略只能用于所有指令都设置为“Default”或“Explore”的运行，并且运行必须包含`phys_opt_design`。

仅当完成布局布线后并且通常在设计无关键问题而时序收敛已接近达成的情况下才会建议运行增量实现。

即使Vivado不建议，也仍然可以运行ML策略或增量实现。

就像建议后续流程阶段一样，它同样可以提供相关建议，使设计满足ML策略和增量编译的要求。不给定任何具体指南时，用户应继续使用RQS提升RQA评分。

QoR评估详情 (QoR Assessment Details)

“QoR Assessment Details”表（如下图所示）提供了便利的设计概况，其中涵盖如下领域：

- 利用率 (Utilization)
- 时钟设置 (Clocking)
- 约束 (Constraints)
- 拥塞 (Congestion)
- 时序 (Timing)

所示数字针对使用的资源提供了“阈值 (Threshold)”和“实际值 (Actual)”百分比，以及绝对值。这些是近似值，组合在一起构成“RQA得分 (RQA Score)”。下图显示了示例报告；

图 115：QoR评估详情 (QoR Assessment Details)

2. QoR Assessment Details					
Name	Threshold	Actual	Used	Available	Status
Utilization					
Registers	50.00	21.67	70514	325440	OK
LUTs	70.00	10.52	17115	162720	OK
LUT Combined	20.00	103.99	17798	17115	REVIEW
Memory LUTs	25.00	0.00	2	99840	OK
MUXF7	15.00	0.00	0	108480	OK
CARRY8	25.00	0.00	0	27120	OK
RAMBs	80.00	0.00	0	360	OK
URAMs	80.00	0.00	0	48	OK
DSPs	80.00	0.00	0	1368	OK
Control Sets	7.50	0.01	6	40680	OK
Pblocks - pblock_clk300_to_clk600_fs_i	50.00	71.85	20004	27840	REVIEW
Registers					
Pblocks - pblock_lut_combiner_i	50.00	82.79	44508	53760	REVIEW
Registers					
Clocking - (causing high skew)					
Number of unbalanced inter clock pairs	0	0	-	-	OK
Number of inter clock pairs not using BUFGE_DIV	0	0	-	-	OK
Constraints					
DONT_TOUCH (cells/nets)	0	3	-	-	REVIEW
DONT_TOUCH because of MARK_DEBUG	0	0	-	-	OK
Congestion					
Number of Level 5 or above regions					
Global	5	-	-	-	OK
Short	5	-	-	-	OK
Average Fanout for modules > 100k cells	4	1.79	-	-	OK
Non-FD high fanout nets > 10k loads	0	1	-	-	REVIEW
Timing					
WNS	0.0	-1.26	-	-	REVIEW
TNS	0.0	-1.26	-	-	REVIEW
WHS	0.0	-0.22	-	-	REVIEW
THS	0.0	-2783.12	-	-	REVIEW
Number of paths above Max LUT Budgeting	0	1	-	-	REVIEW
Number of paths above Max Net Budgeting	0	2	-	-	REVIEW

方法检查

运行有限数量的方法检查，确保为提供有效的 QoR 建议奠定坚实基础。如果方法检查已运行，则将复用缓存的结果，除非设计中存在变更。如果需运行方法检查，将导致运行时间增加。此检查可使用 `-no_methodology_checks` 开关禁用。

QoR 建议报告 (Report QoR Suggestions)

`report_qor_suggestions` 是处理建议对象时使用的主要命令。建议对象用于提升设计满足时序要求的能力，具体方式是将开关添加到命令（如 `opt_design`）、设计对象的属性（如单元和信号线）以及完整的实现策略中。

`report_qor_suggestions` 在 Vivado® IDE 中可生成报告或者生成文本报告。此报告可用于两种情况：

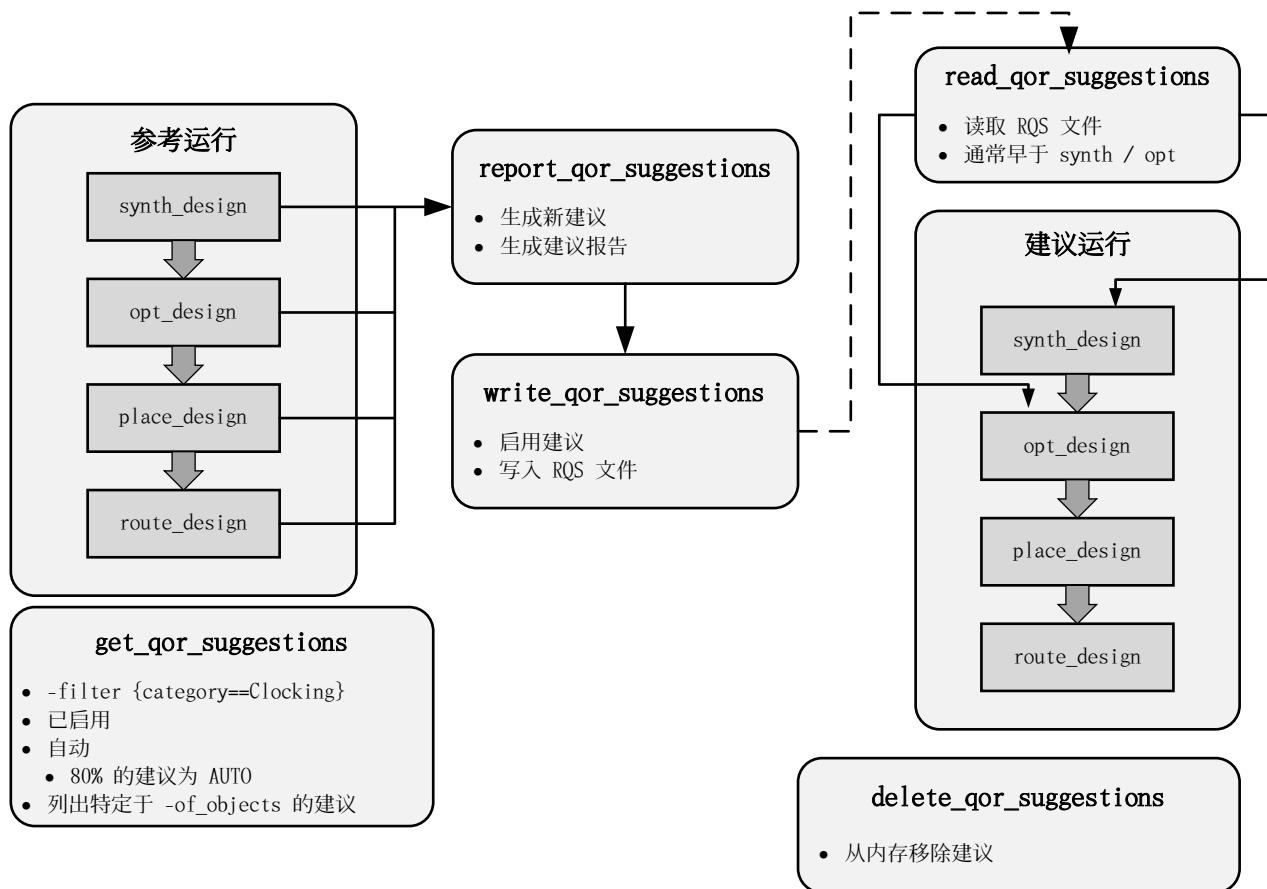
- 生成和查看有关内存中当前设计的新建议。
- 查看使用 `read_qor_suggestions` 命令读入的现有建议

综合后，随时可对内存中加载的设计运行 `report_qor_suggestions` 命令。生成的建议对象会考量诸多设计特性并按如下类别生成建议：

- 时钟设置 (Clocking)
- XDC
- 利用率 (Utilization)
- 拥塞 (Congestion)
- 时序 (Timing)
- 策略 (Strategies)

生成的建议随后必须馈送回流程才能生效。通常必须重新运行各设计阶段，如下图所示：

图116：建议流程



X23300-031720

生成新建议前，必须将设计加载到内存中。`report_qor_suggestions` 将查看时序约束、网表特性、失败的时序路径以及拥塞信息，以确定可增强 QoR 的建议。在综合后的任意阶段均可生成此建议。

新建议按重要性顺序返回，在报告中对应操作按重要性从高到低排序。设计失败概率越高，生成的建议对 `report_qor_assessment` 得分和满足时序要求的能力影响越大。

使用 `report_qor_suggestions` 可帮助设计符合赛灵思 UltraFast 设计方法的建议。返回的许多建议均可自动执行《UltraFast 设计方法指南（适用于 Vivado Design Suite）》(UG949) 中所述的分析和建议。它还将提升 `report_qor_assessment` 报告的 QoR 评分。

最后一个类别“Strategies”是一个特殊类别，其中包含实现策略。这些策略是使用机器学习算法通过分析大量设计特性所生成的。使用这些对象的流程与上述流程略有不同，在本章后文中对此提供了更详细的描述。

执行建议

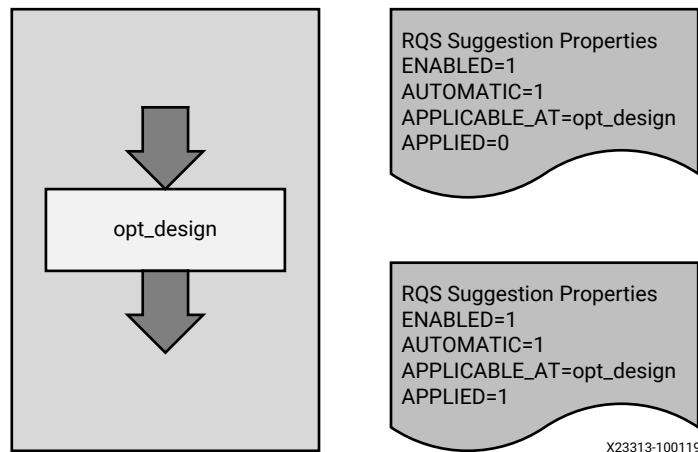
满足以下条件时，在建议运行轮次期间执行建议：

- 建议处于已启用 (ENABLED) 状态。
- 必须运行 APPLICABLE_FOR 阶段。
- 建议必须设置为自动 (AUTOMATIC)。

- 对于新建议，请使用增量实现流程，建议必须设置为 INCR_FRIENDLY。参考检查点中的现有建议不适用此限制。

执行建议时，APPLIED 设置将会更新，如下图所示：

图 117：建议执行



对于未能正确应用的建议，FAILED_TO_APPLY 将设置为 1。

如果 APPLICABLE_FOR 阶段位于生成建议的阶段之后，那么可在生成建议的相同运行轮次中执行这些建议。为此，您必须首先手动启用建议。

```
set_property ENABLED 1 [get_qor_suggestions <SuggID>]
```

使用此方法时，当运行完成后，请务必此建议写入 RQS 文件以便后续继续使用此建议。

其它相关命令

有 5 条相关命令可用于处理 QoR 建议对象：

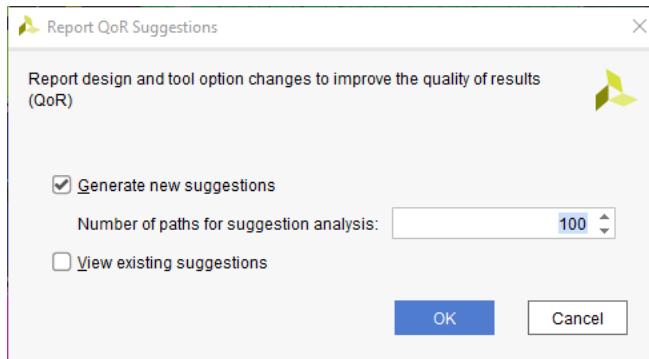
表 7：其它相关命令

命令	功能
report_qorSuggestions	生成新建议 报告现有建议
write_qorSuggestions	将建议对象写入文件。在此进程中，将自动启用建议。
read_qorSuggestions	从文件中读取建议对象
get_qorSuggestions	返回 QoR 建议对象
delete_qorSuggestions	从内存中移除 QoR 建议

生成 QoR 建议报告

可从 Vivado® IDE 使用 “Reports” → “Report QoR Suggestions” 下拉菜单运行 report_qor_suggestions。

图118：“Report QoR Suggestions”对话框



在 Vivado IDE 中运行时，用户可以选择生成新建议（为分析设计，运行时间会延长）和/或报告现有建议。

在 Tcl 控制台上运行时，如果运行的 `report_qor_suggestions` 命令不含 `-of_objects` 开关，那么它将生成并报告新建议，例如：

```
report_qor_suggestions
```

当运行的 `report_qor_suggestions` 带有 `-of_objects`（可简写为 `-of`）时：此命令仅报告现有建议。

```
report_qor_suggestions -of [get_qor_suggestions <objectNames>]
```

因此，要生成新建议并报告现有建议，此命令必须运行 2 次。例如：

```
# Generates new suggestions
report_qor_suggestions -quiet
# Reports on all existing and new suggestions
report_qor_suggestions -of [get_qor_suggestions]
```

对于某些建议（如利用率或拥塞），RQS 将检验整个设计。但对于时序问题，将聚焦顶层路径。默认为 100 条路径。要在 Tcl 中放宽此限制，请使用 `-max_paths N` 开关，其中 N 为整数。

注释：在早期阶段，定时失败的常见主要原因通常与时钟或约束相关。使用此限制即可轻松找到这些原因。接近时序收敛的最终阶段时，随着建议趋向于适用特定路径，最好增大该数值。

QoR 建议报告

此报告分为建议和细节两部分。下表显示了所生成报告的 1 个示例。

图119：`report_qor_suggestions` 报告示例

	ID	GENERATED_AT	APPLICABLE_FOR	SOURCE	AUTOMATIC	DESCRIPTION
General	RQS_XDC-1	✓	opt_design	synth_design	Current Run	No
RQS_Summary	RQS_XDC-1	✓	opt_design	opt_design	Current Run	No
GENERATED	RQS_XDC-1	✓	opt_design	opt_design	Current Run	No
XDC	RQS_UTIL-3	✓	opt_design	opt_design	Current Run	No
RQS_XDC-1	RQS_UTIL-3	✓	opt_design	opt_design	Current Run	No
Utilization	RQS_UTIL-3	✓	opt_design	opt_design	Current Run	No
RQS_UTIL-3	RQS_UTIL-3	✓	opt_design	opt_design	Current Run	No
Timing	RQS_TIMING-33	✓	opt_design	synth_design	Current Run	Yes
RQS_TIMING-33	RQS_TIMING-33	✓	opt_design	opt_design	Current Run	Yes
Timing	RQS_TIMING-54	✓	opt_design	opt_design	Current Run	Yes
RQS_TIMING-54	RQS_TIMING-54	✓	opt_design	opt_design	Current Run	Yes
Clocking	RQS_CLOCK-15	✓	opt_design	opt_design	Current Run	No
RQS_CLOCK-15	RQS_CLOCK-15	✓	opt_design	synth_design	Current Run	No
Clocking	RQS_CLOCK-14	✓	opt_design	synth_design	Current Run	No
RQS_CLOCK-14	RQS_CLOCK-14	✓	opt_design	synth_design	Current Run	No

在“RQS Summary”下的报告中有 1 个列包含所有建议。这些建议分 4 个类别显示，以便于用户建议每一组特定建议。这些建议按如下方式成对显示：

- GENERATED 和 EXISTING
 - “Generated” 建议为流程当前阶段新生成的建议
 - “Existing” 建议可能来自流程先前阶段或者通过读入 RQS 文件获得
- APPLIED 和 FAILED TO APPLY
 - “Applied” 建议为已启用的建议，已通过 APPLICABLE_AT 阶段。这些建议已成功应用。
 - “Failed to apply” 建议已启用并已通过 APPLICABLE_AT 阶段，但尚未成功应用。用户应检查日志文件，以了解未能应用这些建议的原因。

报告的下半部分包含有关生成的建议的详细信息。它分为如下几个类别，以供 report_qorSuggestions 据此分析设计：

- 时钟设置 (Clocking)
- 拥塞 (Congestion)
- 利用率 (Utilization)
- 时序 (Timing)
- XDC
- 策略 (Strategy)

查看“GENERATED”建议时，用户可通过细节部分获取适量的信息，以便用户推断报告建议的原因。可通过“GENERATED”建议的细节部分进行交叉探测。以下交叉探测方法非常实用：

- 选择对象会高亮其它窗口（例如，“Device”窗口）中的对象
- 按 F4 可显示选定对象的原理图
- 右键单击菜单可生成时序报告

查看“EXISTING”建议时，对象可能已发生修改且不存在（例如，opt_design 可能已从网表移除对象）。因此，选择“EXISTING”建议时，交叉探测有时不可用。

对于每项建议，在某些列中都会向用户提供有关如何运用建议的额外实用信息。这些列将显示如下信息：

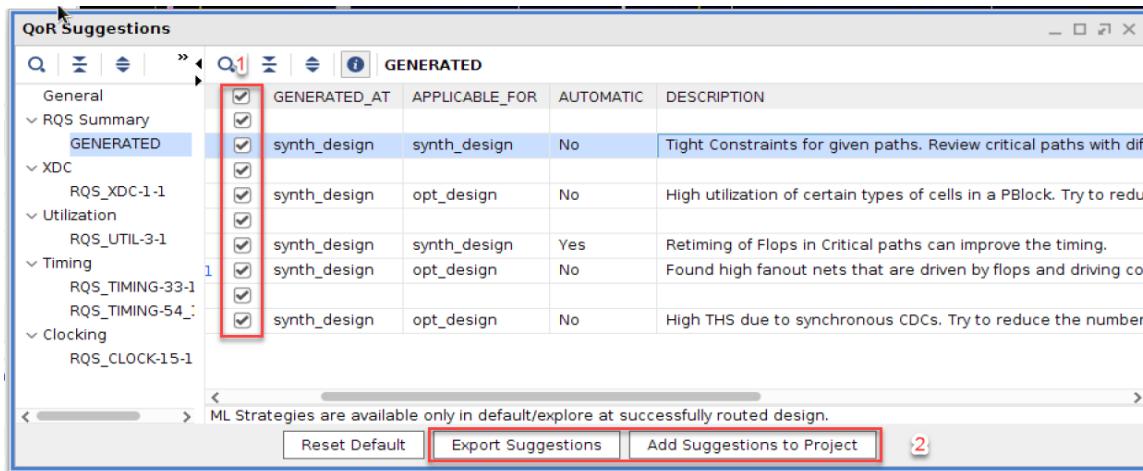
表 8：额外信息

属性	值	说明
GENERATED_AT	设计阶段，例如 opt_design	生成建议的阶段。
APPLICABLE_FOR	设计阶段，例如 opt_design	必须在启用建议的前提下重新运行的阶段。
SOURCE	current_run 或 <filename>.rqs	建议的来源。
AUTOMATIC	Yes 或 No	描述建议由 Vivado 自动执行还是需手动执行。

在 IDE 中处理 QoR 建议对象

生成 QoR 建议报告后，必须生成 RQS 文件以馈送到“建议运行”。为此，请首先选中要包含在运行中的建议，然后写入 QoR 建议文件。如下图所示

图120：选择/写入建议



工程模式

在工程模式中包含允许 Vivado 自动管理 RQS 文件的选项。使用“Add Suggestions to Project”时即可激活该选项。选中该选项时，会将此文件自动添加到工程内的实用工具源文件集中。

在工程内的“Design Runs”窗口中，可右键单击运行并选择“Get QoR Suggestions...”，可能需将此文件同时添加到综合运行和实现运行中。

注释：每个父综合运行都可能包含多个实现。任一给定运行只能使用 1 个 RQS 文件。

此流程的等效 Tcl 命令为：

```
write_qorSuggestions -of_objects [get_qorSuggestions \
{<NAME_1> <NAME_2>}] -file <fn.rqs>
add_files -fileset utils_1 <fn>.rqs
set_property RQS_FILES <fn>.rqs [get_runs <run name>]
```

非工程模式

打开检查点时，只有“导出建议(Export Suggestions)”按钮可用。这样会编写建议文件，随后必须使用 read_qorSuggestions 将此文件添加到运行中。read_qorSuggestions 命令应在 synth_design 或 opt_design 之前运行。

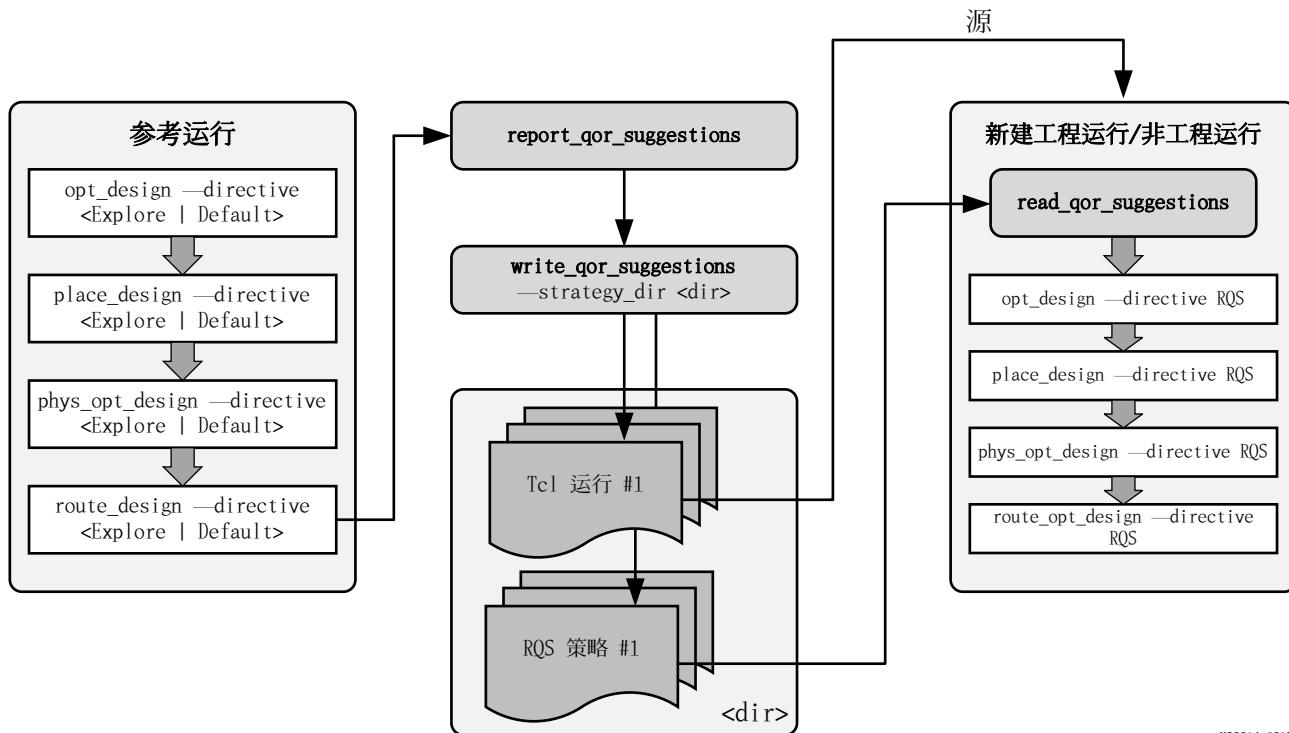
此流程的等效 Tcl 命令为：

```
write_qorSuggestions -of_objects [get_qorSuggestions \
{<NAME_1> <NAME_2>}] -file <fn.rqs>
...
read_vhdl <some_file>.vhd
read_qorSuggestions all_enabledSuggestions.rqs
synth_design -top <top> -part <part>
opt_design
...
write_qorSuggestions -force all_enabledSuggestions.rqs
```

策略建议

策略建议是一种特殊类型的建议。此类建议用于设置最适合设计的实现指令。在 IDE 中不报告此类建议，只能通过 Tcl 生成，并且仅适用于实现运行。预测是基于网表功能来执行的，因此策略运行中的综合设置应与参考运行中的设置相同。此流程如下图所示：

图 121：策略建议流程



X23314-031720

如上图所示，此流程包含 4 个关键点。

首先，应在使用“Default”或“Explore”指令生成并已完全布线的设计上运行 `report_qorSuggestions`。

其次，`write_qorSuggestions -strategy_dir <dir>` 可在指定目录中生成 Tcl 文件和 RQS 文件。默认情况下，可生成 3 项策略。对于生成的每项策略，每个 RQS 文件都包含所有建议对象以及策略建议对象。使用 `write_qorSuggestions -file <fn>.rqs` 指定的 RQS 文件可丢弃，因为在每个策略 RQS 文件中都包含此重复信息。

注释：要生成更多策略，请使用 `report_qorSuggestions -max_strategies <n>` 增加此数量。

第三，必须在新的实现运行中读取生成的 RQS 文件。

最后，必须设置指令 RQS，并且脚本必须包含针对 `opt_design`、`place_design`、`phys_opt_design` 和 `route_design` 的调用。RQS 指令用于指示工具参考此建议。

在工程模式下，基于 Tcl 脚本使用 `source` 命令生成工程。这将基于现有运行自动创建新运行、设置要读取的 RQS 文件并调整指令。

在非工程模式下，提供了 1 个 Tcl 脚本示例。它显示了 RQS 文件的读取方式以及用于设置到 RQS 的实现命令的指令。这些脚本旨在作为设计上的示例，以供在 `pre-opt_design` 阶段中加载到内存中。其中不包含任何检查点报告或写入操作。

增量流程中的 RQS

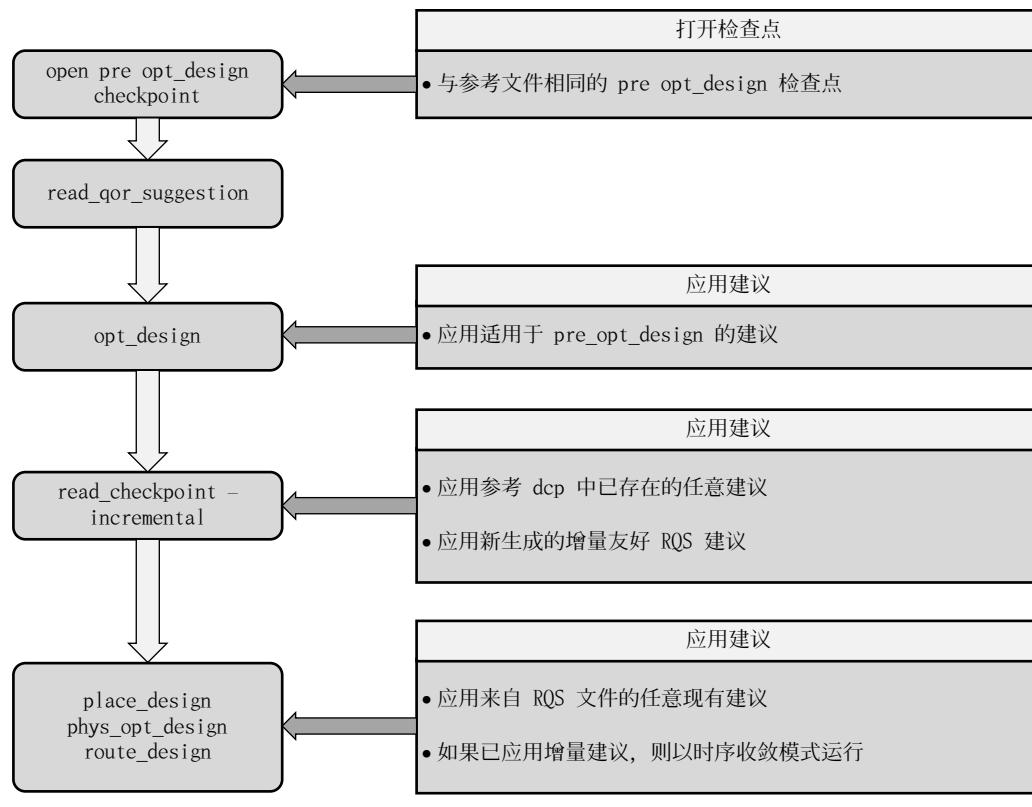
当设计非常接近时序收敛（通常 WNS 小于 -250ps）时，可随 RQS 建议启用增量流程。这样即可利用增量流程和 RQS 建议来实现时序收敛并节省迭代时间。`report_qor_assessment` 可指示应使用此流程的时机。通常当评分大于或等于 4 时，会出现此提示。它显示在“Next Recommended Flow Stages”中，并伴随如下消息：

```
Run report_qor_suggestions and run incremental implementation with  
TimingClosure incremental directive
```

在运行中，运行增量流程命令前，将读入从参考布线 DCP 生成的建议。流程其余部分将为用户自动应用。Vivado 将通过区分新生成 (GENERATED) 的建议以及参考运行中已应用 (APPLIED) 的建议，从而明确应用于流程中各阶段的具体建议。

如下图所示：

图 122：增量流程



X23315-031720

运行增量流程时，在 `pre_opt_design` 阶段的 `opt_design` 中应用的建议将从 RQS 文件执行。还会从参考 DCP 读取在参考中已应用的建议。这些建议无论是否包含在 RQS 文件中都会加以应用，但必须读取 RQS 文件才能激活从参考 DCP 读取建议的操作。后续在 `read_checkpoint` 阶段，将从 RQS 文件执行适合增量流程的新建议。RQS 文件中任何不适合增量流程的建议都将被忽略。

如果增量流程还原为默认流程（通常源于更改产生的负面影响），则将从 RQS 文件执行所有建议。因此，用户应将所有建议导出到 RQS 文件，而不只是导出适合增量流程的建议。

采用此流程前，请注意满足如下前提条件：

- 参考运行与增量运行的器件部分应相匹配。
- 参考检查点应为布线后检查点。
- 针对参考运行和增量运行中的 opt_design 应使用相同的指令。
- 设计不应存在重大设计问题，如高拥塞、时钟设置不平衡或 RQS 评分低于 4。
- 应从参考检查点重新生成建议。
- 仅当新生成的建议属于适合增量流程的建议时，才会加以应用。如果建议不适合增量流程，则仅当流程还原为默认流程时才会执行这些建议。如果不发生还原，则将忽略这些建议。

运行流程所需的命令示例如下所示：

参考运行

```
# Generate RQS suggestions from the reference DCP
open_checkpoint reference_routed.dcp
report_qorSuggestions -file postroute_rqs.rpt
write_qorSuggestions -force ./post_route.rqs
```

增量运行

```
# RQS-Incremental Run:
open_checkpoint <pre_opt.dcp>
read_qorSuggestions ./post_route.rqs
# opt_design directive must be same as the reference run
opt_design -directive {same directive as reference run}
read_checkpoint -auto_incremental -incremental postroute.dcp
# place_design is running in TimingClosure mode
place_design
# phys_opt_design is optimized for incremental
phys_opt_design
# route_design is running in TimingClosure mode
route_design
write_checkpoint postroute_incre_rqs.dcp
```

自动删除建议

为了防止建议过多累积，Vivado 会对建议进行了自动管理。包括：

- 生成新建议时，删除与先前生成的建议相同的建议。
- 使用 write_qorSuggestions 编写建议时，删除并非源自 current_run 且未启用的建议（已指定 -all 的情况下除外）。current_run 中生成的建议无论是否启用都将予以保留。

以 Tcl 或文本格式查看建议

建议对象存储在二进制文件中，因此读取建议的唯一途径是加载设计、读取建议并运行 report_qorSuggestions。对于不希望使用对象流程的用户，支持在 Tcl 中查看和执行建议。

要在 Tcl 中输出建议，用户必须使用以下命令：

```
write_qorSuggestions -tcl_output_dir <outputDir>
```

此命令将把 1 个或多个 Tcl 文件输出到指定目录。该选项在 Vivado IDE 中不可用。

当对象显示在 Tcl 中时，用户必须维护 Tcl，以移除不再需要的对象，并将新生成的 Tcl 脚本追加到 Tcl 中。

`report_qorSuggestions` 将不再报告通过 Tcl 输入的建议。

第 4 章

查看报告和消息

报告和消息简介

赛灵思 Vivado® 集成设计环境 (IDE) 可生成报告和消息，以便告知您各种工具交互期间的设计或设计进程状态。报告由您（或工具）在执行设计流程中的任意关键步骤时生成。报告可汇总有关设计的具体信息。

工具可在设计进程中每个步骤自动生成消息，也可以为各项用户操作生成消息。

消息和报告存储在“结果 (Results)”窗口区域中的“消息 (Messages)”窗口和“报告 (Reports)”窗口中。

运行以下任一命令时，该工具会启动新进程：

- 运行综合 (Run Synthesis)
- 运行实现 (Run Implementation)
- `launch_runs` (Tcl)

如需了解有关 Tcl 命令的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#))，或输入 `<command> -help`。

此进程生成的消息和报告保留在硬盘上，直至您运行复位为止。打开工程时，会显示运行相关的消息。该工具仅在“Messages”窗口中显示处于活动状态的运行的消息。

在 Vivado IDE 中报告各项操作结果：

- 加载设计时，可通过“工具 (Tools)”菜单使用各种不同报告命令。
- 运行综合或实现可在运行过程中创建报告。

在 IDE 中查看和管理消息

消息可提供有关设计特定元素或者有关工具流程中发生的错误的简洁状态声明。

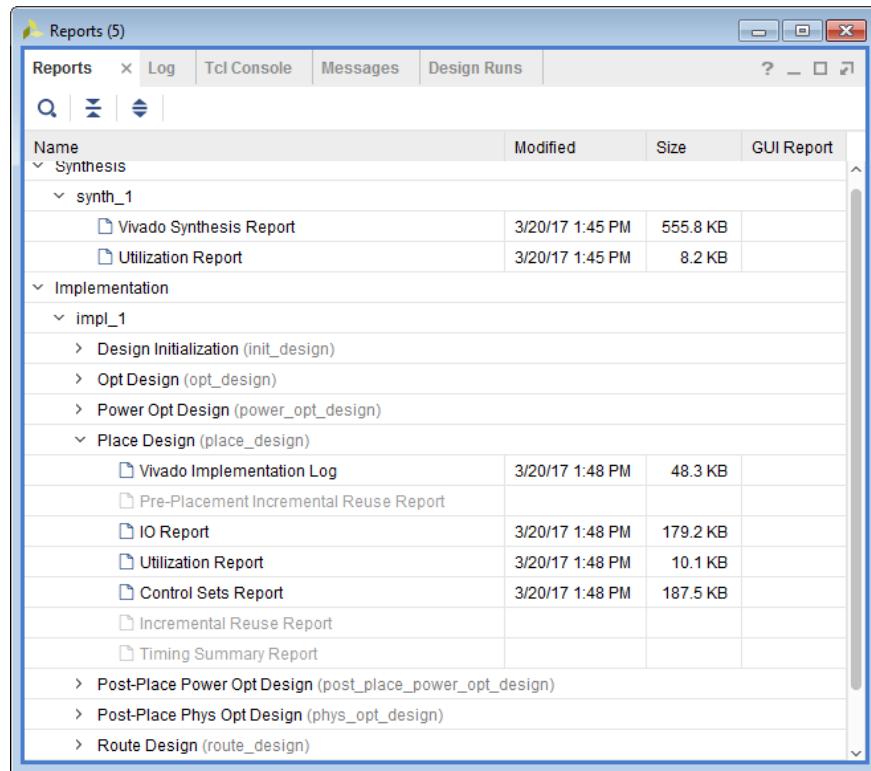


提示：请复查消息以判定 Vivado 工具是否在设计中的任意部分遇到困难或错误。

使用“报告 (Reports)”窗口

正在运行的综合与实现的报告显示在“Reports”窗口中。选择“运行属性 (Run Properties)”窗口的“Reports”选项卡即可查看“设计运行 (Design Runs)”窗口中选中的运行轮次的报告。双击报告即可在文本查看器中查看此报告。

图 123：“Reports”窗口



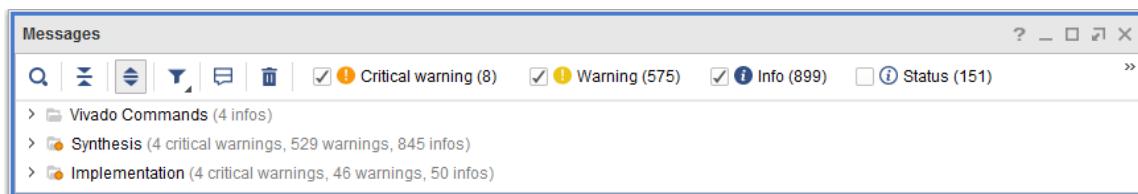
使用“消息 (Messages)”窗口

消息分为两种类型：

- 存储在磁盘上的消息
- 存储在内存中的消息

Vivado 集成设计环境 (IDE) 按创建消息时的操作在“Messages”窗口中对消息进行分组。使用工具栏菜单上的设置按钮可按消息 ID 或文件对消息进行分组。

图 124：“Messages”窗口



某些消息包含指向文件或设计元素的超链接，可帮助调试。单击链接均可查看来源。

提示： 使用弹出菜单可将消息复制粘贴到其它窗口或文档。

每条消息都带有消息 ID 和消息严重性标签。

- 消息 ID (Message ID): 消息 ID 用于识别不同消息，以便对其进行分组和排序。
- 消息严重性 (Message Severity): 消息严重性用于描述所显示的信息的性质。

部分消息要求您关注并解决相关问题后方可细化、综合和实现设计。部分消息仅供参考。参考消息可提供有关设计或流程的详情，但无需用户操作。

表 9：消息严重性

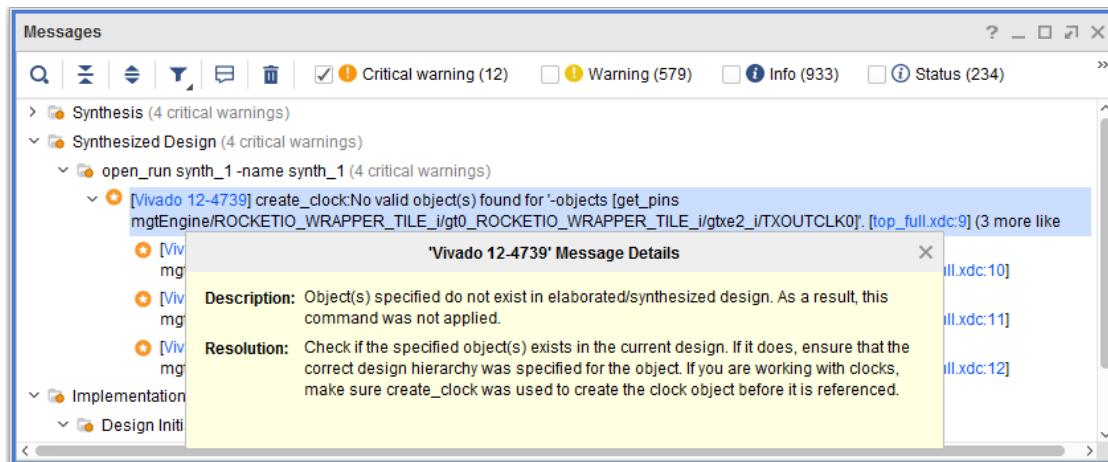
图标	严重性	消息
	Status	用于传达设计处理的一般状态。
	Info	有关设计处理的工艺和反馈的一般状态。
	Warning	可能由于约束或规格未按期望方式应用，导致设计结果处于欠优化状态。
	Critical Warning	某些用户输入或约束将不予应用，或者超出最佳实践范围，这通常导致流程后续出现错误。请检验其来源和约束。强烈建议执行更改。
	Error	出现问题，表明设计结果不可用且必须用户干预才能解决。设计流程停止。



建议：请仔细审查工具在加载内存中的设计时或者从当前运行的综合和实现加载设计时所发出的所有错误 (Error) 和严重警告 (Critical Warning)。这些消息可提供需要您关注的问题的相关信息。许多消息都包含冗长的描述以及解决方案建议，可通过单击消息 ID 来显示这些建议。

请参阅下图所示示例。在此例中，设计中无法找到基准时钟约束所引用的端口（首次警告），因此未创建时钟（首次严重警告），而引用该时钟的所有所有其它时钟也同样失败。

图 125：审查“错误 (Error)”和“严重警告 (Critical Warning)”



筛选消息

您可按严重性对消息进行筛选。

要启用或禁用显示特定类型的消息，请执行以下操作：

1. 转至“消息 (Messages)”窗口。
2. 选中（以启用）或取消选中（以禁用）窗口标题中的消息严重性旁的复选框。

您可更改特定消息 ID 的严重性。例如，您可将自己认为不严重的消息的严重性降低，或者将您认为需要更多关注的消息的严重性提升。

要提高或降低消息严重性，请使用 `set_msg_config` Tcl 命令。例如：

```
set_msg_config -id "[Common 17-81]" -new_severity "CRITICAL WARNING"
```

如需了解有关 `set_msg_config` Tcl 命令的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#)) 中的相应内容。

Vivado 生成的消息

本节探讨 Vivado 生成的消息，具体包括：

- [综合日志](#)
- [实现日志](#)
- [WebTalk 报告](#)

综合日志

Vivado 综合日志是 Vivado 综合工具的主要输出，其中包括：

- 所处理的文件，包括：
 - VHDL
 - Verilog
 - System Verilog
 - XDC
- 每个单元的参数设置
- 含多个驱动的信号线
- 未驱动的层级管脚
- 最优化信息
- 黑盒
- 最终原语计数
- 单元利用率（按层级）
- 运行时间和内存利用率



重要提示！ 请复查此报告或“消息 (Messages)”选项卡中显示的“错误 (Errors)”、“严重警告 (Critical Warnings)”和“警告 (Warnings)”。综合工具可发出“Critical Warnings”和“Warnings”，后者在后续流程中严重性可能进一步提升。

实现日志

Vivado 实现日志包括如下内容：

- 有关位置、网表和所使用的约束的信息。
- 逻辑最优化任务。默认情况下，该工具运行逻辑最优化例程来生成更小更快的网表。
- 布局阶段以及布局后时序估算（仅限 WNS 和 TNS）。
- 布线器阶段以及多项时序估算和估算的布线后时序汇总信息（仅限 WNS、TNS、WHS 和 THS）。
- 每条实现命令和阶段所耗用的时间和内存。

请复查此报告或“消息 (Messages)”选项卡相应部分中显示的“错误 (Errors)”、“严重警告 (Critical Warnings)”和“警告 (Warnings)”。 “布局器 (Placer)”会生成警告，这些警告在流程后续可能升级为“错误 (Errors)”。如果使用单步运行，那么日志将仅包含最后一步的结果。



重要提示！ 请复查“时序汇总 (Timing Summary)”报告以查看：(1)“脉冲宽度 (Pulse Width)”时序汇总信息，以及(2)有关时序违例或约束缺失的其它信息。

WebTalk 报告

“WebTalk 报告”是在生成比特流期间生成的。此报告可帮助赛灵思了解其客户使用赛灵思 FPGA 器件、软件和 IP 的方式。由 WebTalk 收集并传输的信息有助于赛灵思改进对客户最重要的功能。不收集任何专有信息。欲知详情，请访问 <https://china.xilinx.com/webtalk/>。

设计检查的生成与豁免

豁免机制提供了对 CDC、DRC 和 Methodology 违例进行豁免的途径。豁免后，`report_cdc`、`report_drc` 和 `report_methodology` 命令将不再报告这些违例。在作为实现命令（如 `opt_design`、`place_design`、`phys_opt_design`、`route_design` 或 `write_bitstream`）的前提条件运行的强制性 DRC 中同样会过滤掉已豁免的 DRC。

豁免与 XDC 兼容，可通过 `read_xdc` 或 `source` 命令导入。在工程模式或非工程模式下，豁免可包含在任意 XDC 文件或 Tcl 脚本中。可从顶层创建豁免，或者也可将其限定于层级模块。将豁免添加到设计中后，会将其自动保存在检查点内，并在重新加载检查点时将其复原。可使用 `write_xdc` 和 `write_waivers` 命令写出豁免。

豁免可提供跟踪功能。Vivado 工具会记录创建豁免的用户、创建日期和时间以及简短描述。此信息对于跟踪至关重要。建议审查并验证应用于设计的所有豁免以确保其有效性。

豁免是可供创建、查询、报告和删除的第一类对象。豁免可引用由 Vivado `get_*` 命令返回的其它第一类对象，例如，管脚、单元、信号线、Pblock 和站点。在设计中，这些对象必须已存在，才能创建豁免。创建豁免时不存在的设计对象将不涵盖在豁免范围内。



重要提示！ 与其它约束类似，建议在综合后设计上创建豁免。在实现后设计上创建的豁免可引用综合后网表内不存在的设计对象。如果此类豁免应用于综合后设计，则将被丢弃。

豁免机制支持复制和删除网表对象。复制豁免中涉及的对象时，会将复制的对象自动添加到豁免中。同样，删除对象时，还将从豁免移除针对该对象的所有引用。如果删除对象导致豁免引用的对象列表为空，此豁免将从内存内设计中删除，且不保存在后续检查点内。同样的机制也适用于时序约束和时钟对象。通过逻辑最优化或通过移除时序约束 (`reset_timing`) 删除时钟时，引用该时钟对象的所有豁免也将被删除，且不保存在后续检查点内。

注释：定制设计规则检查无法豁免。如需了解有关用户编写的 DRC 的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用 Tcl 脚本》(UG894) 中的相应内容。

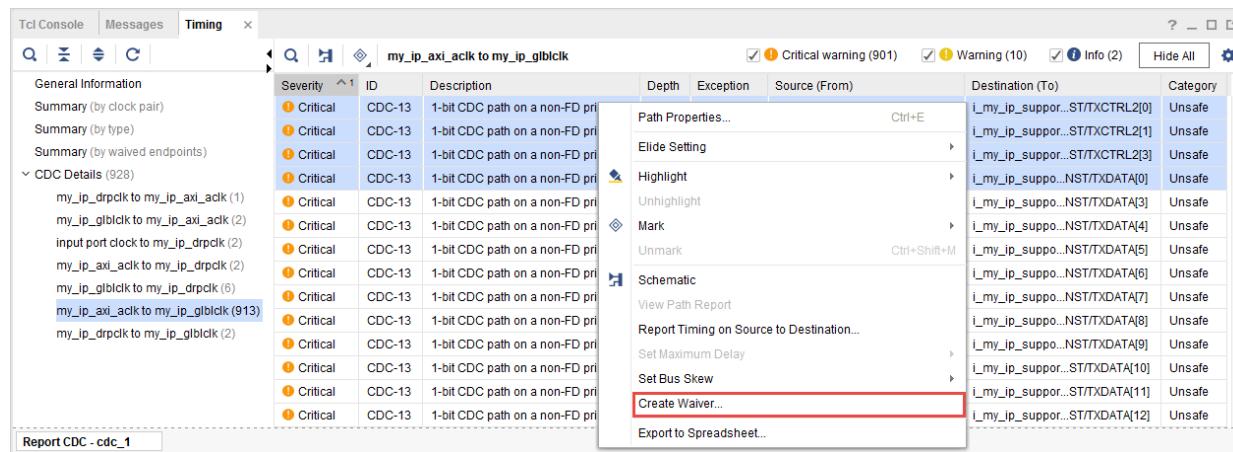
创建豁免

可从 GUI、DRC、Methodology 或 CDC 违例对象创建豁免，也可通过手动指定所有必需实参来指定豁免。

从 GUI 创建豁免

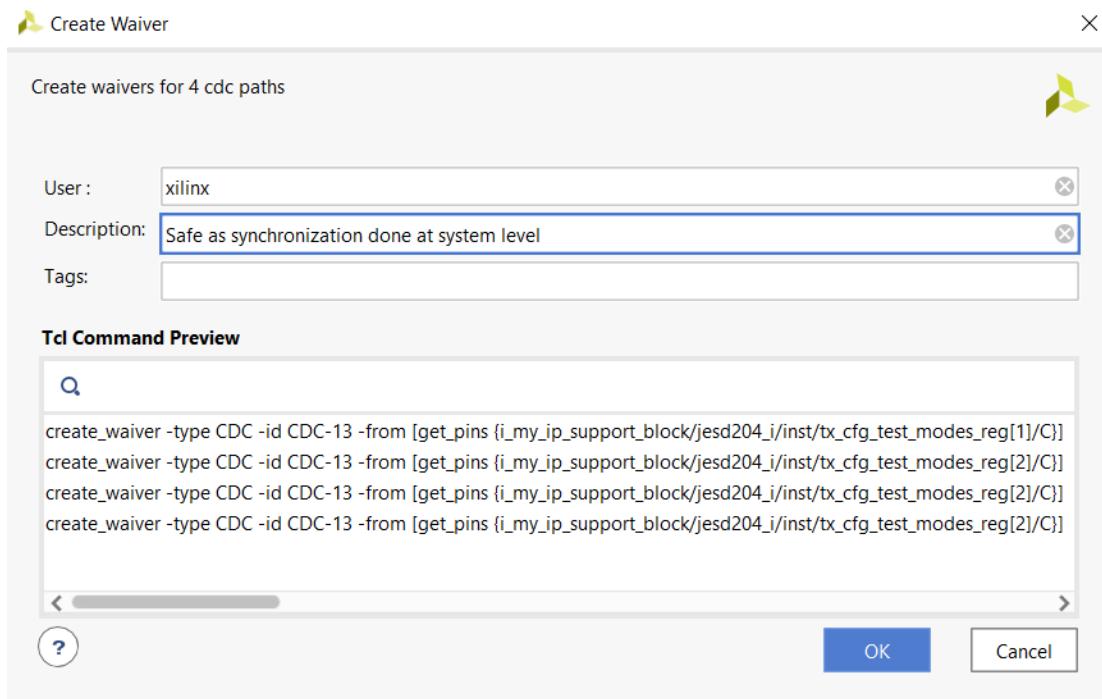
可从 Report DRC、Report Methodology 或 Report CDC GUI 结果窗口直接创建豁免。要从结果窗口中豁免违例，请选中一项或多项违例，右键单击上下文菜单并从中选择“Create Waiver”。在下图中，已选中 4 个 CDC 违例并为其创建 4 项豁免。

图 126：从 CDC 违例创建豁免



单击“Create Waiver”会打开以下小组件。

图 127：Create Waiver GUI



即使 Vivado 工具会填充用户名，但此“User”字段仍可编辑。“描述 (Description)”为必填项，建议提供详细信息以供设计团队复查。“标签 (Tags)”字段为可选字段，可用于通过关键字字符串或列表来提供额外描述。它主要用于记录，可用于搜索 XDC 文件或者使用 `get_waivers` 命令筛选豁免等。该对话框包含 GUI 发送给 Tcl 控制台的 Tcl 命令预览。

提交“Create Waiver”窗口后，针对豁免的每项违例，GUI 将向 Tcl 控制台发送 1 条 `create_waiver` 命令。对于 DRC 违例和 Methodology 违例，GUI 生成的 `create_waiver` 命令会引用违例对象，但这只是种转换形式。豁免引擎将违例对象转换为含完整描述的豁免，并引用违例中所涉及的所有设计对象。创建的豁免从不引用作为构建基础的原违例对象。创建豁免时，引擎会自动添加时间戳。

从 GUI 创建豁免后，选定的行将变为灰显并禁用，而此报告将变为旧报告。这是一种直观的确认，表明已从该结果窗口创建了部分豁免。重新运行 GUI 报告后，将从新结果窗口中过滤掉已豁免的违例。

图 128：创建豁免后已禁用的行

Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppor...ST/TXCTRL[2][0]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppor...ST/TXCTRL[2][1]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppor...ST/TXCTRL[2][3]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[1]/C	I_my_ip_suppo...NST/TXDATA[0]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[1]/C	I_my_ip_suppo...NST/TXDATA[3]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppo...NST/TXDATA[4]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppo...NST/TXDATA[5]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppo...NST/TXDATA[6]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[1]/C	I_my_ip_suppo...NST/TXDATA[7]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[1]/C	I_my_ip_suppo...NST/TXDATA[8]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[1]/C	I_my_ip_suppo...NST/TXDATA[9]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppor...ST/TXDATA[10]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	I_my_ip_suppor...odes_reg[2]/C	I_my_ip_suppor...ST/TXDATA[11]	Unsafe

注释：对于 DRC 违例和 Methodology 违例，从 GUI 创建豁免的过程是相同的。

从违例对象创建豁免。

创建豁免的第 2 种方法是使用 DRC、Methodology 或 CDC 违例对象。GUI 使用此方法向 Tcl 控制台发送 `create_waiver` 命令。

以下语法用于从 1 个或多个违例对象创建豁免：

```
create_waiver -of_objects <ViolationObject(s)> -description <string> [-user <name>]
```

描述 (description) 为必填字段。不指定用户 (user) 时，系统使用当前运行 Vivado 工具的用户 ID。

注释：指定多个违例对象时，系统会为每个违例创建 1 项豁免。创建的豁免不引用原始违例对象。而改为在豁免中包含违例所含的字符串和对象列表。

违例对象通过 `get_cdc_violations`、`get_drc_violations` 和 `get_methodologyViolations` 命令返回。仅当先前已运行 `report_cdc`、`report_drc` 和 `report_methodology` 时，这些命令才会返回对象。请使用命令行选项 `-name` 从任一 GUI 报告获取违例对象的列表。

以下示例代码用于为起点位于 `top/sync_1` 模块内部的所有 CDC-1 违例创建豁免。

```
report_cdc -name cdc_1
set vios [get_cdc_violations -name cdc_1 -filter {CHECK == CDC-1}]
foreach vio $vios {
    if {[regexp {^top/sync_1} [get_property STARTPOINT_PIN $vio]]} {
        create_waiver -of $vio -description {Safe by protocol}
    }
}
```

这样会根据违例内部引用的所有对象和字符串来构建从违例对象创建的豁免。这使该豁免成为该违例的独有豁免。如果希望此豁免涵盖多个违例，则必须将豁免导出到外部文件并编辑 `create_waiver` 命令以将单个对象和字符串替换为模式和通配符等。请参阅[创建 DRC 豁免和 Methodology 豁免](#)，以获取有关使用模式和通配符的更多信息。

注释：对于某些 DRC 和 Methodology 检查，部分字符串将自动转换为通配符，例如，UCIO-1、NSTD-1、TIMING-15 和 TIMING-16。对于 TIMING-15 或 TIMING-16，违例内部的建立和保持裕量并不重要。从 TIMING-15 或 TIMING-16 违例对象创建豁免时，`create_waiver` 命令会自动将表示裕量的字符串替换为通配符。这使豁免可应用于特定对象相关的违例，与报告的裕量无关。类似的行为也适用于 UCIO-1 和 NSTD-1。

从命令行创建豁免

特定 DRC 和 Methodology 违例的豁免是唯一的。违例是字符串和/或各种器件和设计对象（例如，管脚、单元、信号线、Pblock、站点 (site) 和拼块 (tile)）的聚合。部分 DRC、Methodology 和 CDC 违例可能仅包含上述某一元素，而某些违例可能包含多项元素。所有字符串和对象的顺序和内容都至关重要，必须予以保留。当使用按错误顺序指定的实参创建豁免程序时，豁免将无法应用于任何违例，或者可能应用于错误的违例。

为特定违例（例如，TIMING-14#1）或某类违例（例如，TIMING-14）手动创建豁免前，建议首先从 GUI 或者从违例对象创建示例豁免。使用 `write_waiver` 或 `write_xdc` 命令导出豁免。随后，可将系统创建的豁免内容与原始违例关联，并确认需指定的字符串和对象顺序。此信息可扩展至含相同 CDC、DRC 或 Methodology ID（例如，TIMING-14）的其它豁免。

对于任意 CDC、DRC 和 Methodology 豁免，都存在 2 个必需实参：

- **ID：**要豁免的违例或检查 ID。ID 是使用 `-id` 指定的。例如，CDC-1、TIMING-14 或 PDRC-1569。每次仅限指定 1 个 ID。
- **Description：**支持多行字符串。必须提供充分信息以供团队审查。Description 是使用 `-description` 指定的。

您可使用命令行选项 `-type` 来强制设置豁免类型：CDC、DRC 或 Methodology。以错误类型创建的豁免无法与任何违例匹配。例如，要豁免 CDC 违例，豁免类型必须设置为 CDC。不指定类型时，系统会从使用 `-id` 指定的检查 ID 推断类型。用户名可使用 `-user` 选项覆盖。默认情况下，系统使用当前运行 Vivado Design Suite 的用户 ID。

豁免支持 XDC 范围限定机制，创建豁免后可更改当前实例。在此情况下，当前实例信息将与豁免保存在一起，并在豁免导出为 XDC 时还原。创建限定范围的豁免时，建议使用命令行选项 `-scope` 以确保限定通配符范围。

如果豁免所含实参与已存在的其它豁免所含所有实参完全相同，那么系统会将该豁免视为重复。为减少内存占用和运行时间，重复豁免不予保存，并生成如下消息：

```
WARNING: [Vivado_Tcl 4-935] Waiver ID 'CDC-13' is a duplicate and will not  
be added  
again.
```

某些 DRC/Methodology 检查（如 RTSAT-*）为只读，无法豁免。可通过选中 DRC/Methodology 检查对象上的 `IS_READ_ONLY` 属性来筛选可豁免的 DRC/Methodology 检查列表。例如：

```
set allWaivableChecks [get_drc_checks -filter {!IS_READ_ONLY}]  
set allWaivableChecks [get_methodology_checks -filter {!IS_READ_ONLY}]
```

以下消息是豁免只读检查（如 DRC RTSTAT-12）时，`create_waiver` 生成的错误消息示例：

```
ERROR: [Vivado_Tcl 4-934] Waiver ID 'RTSTAT-12' is READONLY and may not be  
waived.
```

创建 DRC 豁免和 Methodology 豁免

`create_waiver` 的附加实参的数量和类型取决于需豁免的 DRC 和 Methodology 违例。极少数 DRC 和 Methodology 违例（如 TIMING-9）不含其它实参，因为消息为通用消息而非专用消息。其它 DRC 和 Methodology 违例可能包含多个字符串和不同类型的对象。

注释：对于不引用任何字符串或对象的违例（例如，TIMING-9 和 TIMING-10），不建议将其豁免。

违例内的字符串以豁免内的 `-string` 来指定。器件或设计对象（管脚、单元、信号线、Pblock 和站点 (site)）以 `-objects` 选项来指定。对于违例包含的上述每个元素，都应单独指定上述每个命令行选项。

从 GUI 或者从违例对象创建豁免时，豁免通过指定所有相应字符串和对象来构成唯一违例，并定义为仅适用于该违例。手动创建豁免时，可拓宽豁免适用范围以便使单一豁免适用于多个违例。要将豁免扩展为覆盖多个违例，请执行以下操作：

- 针对 `get_*` 命令使用模式来代替具体名称
- 使用通配符代替单一字符串或对象。通配符为特殊字符，例如，以 “*” 代替“任意字符串”，或者以 “*PIN” 代替“任意管脚”（请参阅下表）。只要相同类型的对象与相同位置的违例中找到的元素匹配，即表示成功匹配。如果来自豁免的所有元素都与违例匹配，则该违例可享受豁免。
- 指定对象列表代替单一对象。只要违例内的对象与豁免内相同位置的对象列表内的任一对象匹配，即表示成功匹配。如果来自豁免的所有元素都与违例匹配，则该违例可享受豁免。

例如，以下命令将豁免引用 `mux2_inst/mux_out_INST_0` 单元的单一 TIMING-14 违例：

```
create_waiver -id "TIMING-14" -description "Reviewed by the team" \
    -objects [get_cells mux2_inst/mux_out_INST_0 ]
```

假设设计包含多个 `mux2_inst/mux_out_INST_*` 单元，那么可通过修改上述豁免，来豁免与所有这些单元相关的 TIMING-14 违例，方法是针对 `get_cells` 命令使用如下模式：

```
create_waiver -id "TIMING-14" -description "Reviewed by the team" \
    -objects [get_cells mux2_inst/mux_out_INST_* ]
```

下表汇总了基于对象类型用作为通配符的关键字。

表 10：通配符关键字

对象	通配符
单元	*CELL
信号线	*NET
管脚	*PIN
端口	*PORT
站点 (Site)	*SITE
拼块 (Tile)	*TILE
BEL	*BEL
封装 Bank	*PKG_BANK
时钟区域	*CLKREGION
时钟	*CLOCK
Pblock	*PBLOCK
字符串	*

注释：`create_waiver -scope` 会将管脚和单元的通配符强制限定为创建豁免的当前实例。创建限定范围的豁免时，`-scope` 可确保管脚和单元的通配符与位于比该范围更高层次的对象不匹配，否则可能导致对不允许豁免的违例执行豁免。

创建 CDC 豁免

CDC 豁免的定义较为简单，因为每个 CDC 违例都仅引用源和目标元素的 2 个管脚或端口对象。请使用命令行选项 `-from/-to` 来指定源和目标管脚或端口。CDC 豁免无法通过 `-string/-objects` 来定义。



重要提示！ CDC 豁免不受源和目标时钟影响，仅受源和目标管脚影响。因此，从 GUI 或者从某些 CDC 违例对象创建豁免时，如果这些对象引用不同时钟对的相同源和目标管脚，则可能会导致生成如下警告：WARNING: [Vivado_Tcl 4-935] Waiver ID 'CDC-7' is a duplicate and will not be added again。

以下命令用于在源管脚 `U_CORE/U00_TOP/sr_reg[3]/C` 与目标管脚 `U_CORE/U10/ar_reg[3]/CE` 之间创建 CDC-1 豁免。

```
create_waiver -id {CDC-1} -description "CDC violations" \
-from [get_pins {U_CORE/U00_TOP/sr_reg[3]/C}] \
-to [get_pins {U_CORE/U10/ar_reg[3]/CE}]
```

如果省略命令行选项 `-from` 或 `-to`，则豁免引擎会将缺失的选项作为通配符来处理。

以下 2 条命令效果相同，用于豁免端点管脚 `U_CORE/U10/ar_reg[3]/CE` 的所有 CDC-1，与起点无关：

```
create_waiver -id {CDC-1} -description "CDC violations" \
-from {*PIN} \
-to [get_pins {U_CORE/U10/ar_reg[3]/CE}]
create_waiver -id {CDC-1} -description "CDC violations" \
-to [get_pins {U_CORE/U10/ar_reg[3]/CE}]
```

CDC 规则优先级

默认情况下，“Report CDC”针对每个端点和每个时钟对仅报告 1 个违例。如果针对任一特定时钟对存在多个违例，则仅报告优先级最高的 CDC 违例。

CDC 规则按优先级从高到低排序，如下表所示。

表 11：CDC 规则优先级

规则 ID	CDC 拓扑结构	严重性	类别
CDC-18	使用 HARD_SYNC 原语同步	Info	Safe
CDC-13、14	非 FD 原语上的 1 位和多位 CDC 路径	Critical	Unsafe
CDC-17	MUX 保持类型	Warning	Safe
CDC-16	MUX 类型	Warning	Safe
CDC-15	CE 类型	Warning	Safe
CDC-26	LUTRAM 读取/写入存在潜在冲突	Warning	Safe
CDC-7	异步复位未同步	Critical	Unknown
CDC-1、4	1 位和多位 CDC 未同步	Critical	Unknown
CDC-12	多时钟扇入	Critical	Unsafe
CDC-10	同步器间检测到组合逻辑	Critical	Unsafe
CDC-11	从启动触发器扇出到目标域	Critical	Unsafe
CDC-9	异步复位，使用 ASYNC_REG 属性同步	Info	Safe
CDC-3、6	1 位和多位，使用 ASYNC_REG 属性同步	Info	Safe
CDC-8	异步复位，使用缺失的 ASYNC_REG 属性同步	Warning	Safe

表 11：CDC 规则优先级（续）

规则 ID	CDC 拓扑结构	严重性	类别
CDC-2、5	1 位和多位，使用缺失的 ASYNC_REG 属性同步	Warning	Safe

注释：以上列出的严重性为“Warning”的部分规则的优先级比其它严重性为“Critical”的规则的优先级更高，原因是这些规则实际上因 CDC 拓扑结构不同而并未应用于相同端点。

当任一端点具有多个 CDC 违例时，如果优先级最高的违例享有豁免，则报告的违例为按优先级排序次之的违例。

要为设计创建豁免，较为简便的方法是在单次运行中针对单一端点报告所有 CDC 违例，忽略规则优先级。使用 report_cdc 命令行选项 -all_checks_per_endpoint 可生成 1 份包含设计中所有 CDC 违例的详尽报告。

注释：-all_checks_per_endpoint 只能通过 Tcl 控制台运行，在“Report CDC”对话框中不予支持。但可使用 -name 选项在 Vivado IDE 中显示 -all_checks_per_endpoint 的结果。

报告豁免

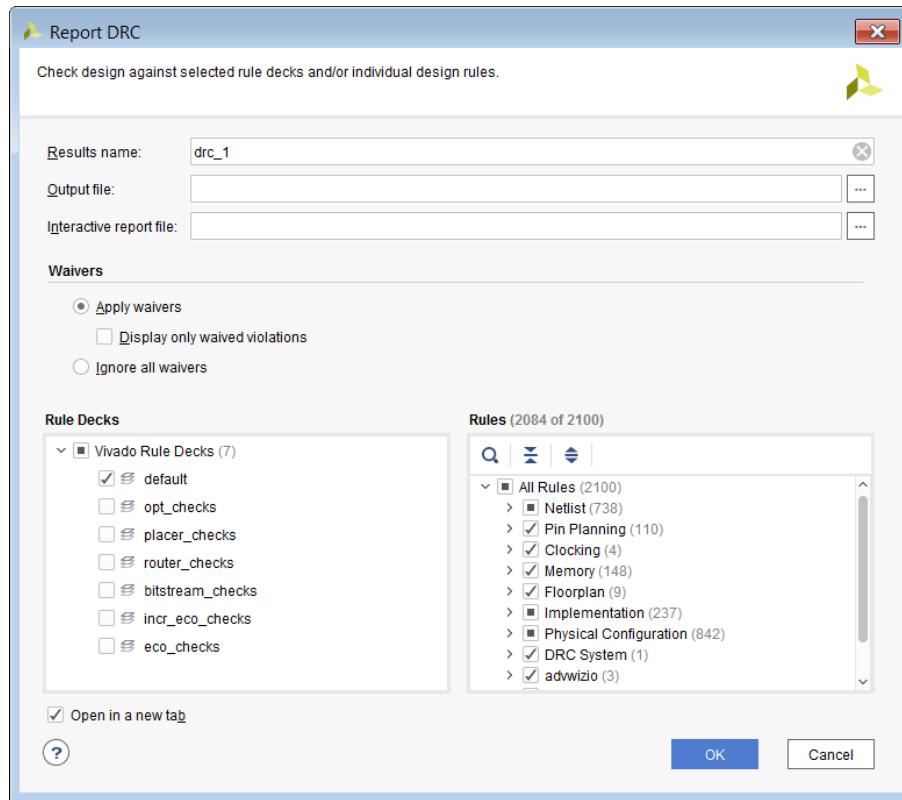
建议验证豁免的违例是否符合预期。必须在定义豁免后且最终比特流之前执行此操作。

Report CDC、Report DRC 和 Report Methodology 命令支持多种报告模式：

- 默认情况下，report_cdc、report_drc 和 report_methodology 命令仅报告未豁免的违例。
- 请使用 -waived 来强制 report_cdc、report_drc 和 report_methodology 命令仅报告已豁免的违例。必须复审报告，确认所有已豁免的违例都符合预期。
- 使用 -no_waiver 可强制 report_cdc、report_drc 和 report_methodology 命令在不应用豁免的情况下运行。在此模式下，所有违例无论是否豁免都会包含在报告中。

在命令行和 GUI 的“Report”对话框窗口中提供了 3 种报告模式。下图来自“Report DRC”，显示了“Waivers”部分下的报告模式选择。在“Report CDC”和“Report Methodology”小组件下同样包含此“Waivers”部分。

图 129：豁免的报告模式



当 CDC、DRC 和 Methodology GUI 的结果窗口中包含已豁免的违例时，这些窗口存在显而易见的差异，每个违例前的图标（ ）都不同，且结果窗口的名称包含已豁免的违例数量。

注释：已豁免的 CDC、DRC 和 Methodology 违例的结果窗口中无法创建豁免。

以下示例显示了已豁免的 DRC 的结果窗口，其中仅含 2 项豁免违例。

图 130：已豁免的 DRC 违例



要获取所有豁免和已豁免的违例的汇总报告，请使用 `report_waivers` 命令。此报告只能从 Tcl 控制台使用。

report_waivers 仅报告由 report_cdc、report_drc 和 report_methodology 命令提取的统计数据。要获取准确的统计数据，需在运行 report_waivers 前以及对豁免进行任意修改（添加或删除）时运行 report_cdc、report_drc 和 report_methodology。无论是从命令行还是从 GUI 运行报告，都会更新统计数据。如果未能获得统计数据更新，report_waivers 会发出以下 1 条或多条消息，具体取决于哪些 CDC、DC 或 Methodology 信息已过期：

```
WARNING: [Vivado_Tcl 4-972] Waiver counts for 'CDC' will be invalid because
report_cdc has not been run since waivers were changed; please run the
report_cdc
command.
WARNING: [Vivado_Tcl 4-972] Waiver counts for 'DRC' will be invalid because
report_drc has not been run since waivers were changed; please run the
report_drc
command.
WARNING: [Vivado_Tcl 4-972] Waiver counts for 'METHODOLOGY' will be invalid
because
report_methodology has not been run since waivers were changed; please run
the
report_methodology command.
```

来自 report_waivers 的报告包含汇总表格和对应每项 CDC、DRC 和 Methodology 豁免的详细表格。表格中包含以下列：

- 违例总数 (Total Vios)：应用豁免前的违例总数。将报告的违例数量（不含违例）。在端点数量中一并计算多比特规则数量。
- 剩余违例数 (Remaining Vios)：应用豁免后的违例数。没有任何违例获得豁免时，此数值与“违例总数”相同。在端点数量中一并计算多比特规则数量。
- 已豁免的违例数 (Waived Vios)：已豁免的违例数。没有任何违例获得豁免时，此数值为 0。在端点数量中一并计算多比特规则数量。
- 已用豁免数 (Used Waivers)：已豁免部分违例的豁免数。如果豁免包含某些模式或通配符，则每项豁免均可应用于多项违例。
- 已落实的豁免数 (Set Waivers)：已应用于设计的豁免数。理想情况下，“Used Waivers”与“Set Waivers”的数值应相同。当某些豁免已定义但不匹配任何违例时，这两者的数值不匹配。

默认情况下，在详细表格中仅报告已定义部分豁免的规则。但是，首个“Summary”表会报告设计中的所有违例。

图 131：report_waivers 默认报告

```
-----
Table Of Contents
-----
1. REPORT SUMMARY
2. REPORT DETAILS (DRC: no waivers)
3. REPORT DETAILS (METHODOLOGY)
4. REPORT DETAILS (CDC)

-----
1. REPORT SUMMARY

Waiver Type Total Vios Remaining Vios Waived Vios Used Waivers Set Waivers
----- ----- ----- ----- -----
DRC 240 240 0 0 0
METHODOLOGY 166 162 4 4 4
CDC 929 923 6 6 6
Note: This report is based on the most recent report_drc/report_methodology/report_cdc runs.

-----
2. REPORT DETAILS (DRC)

Rule Severity Description Total Vios Remaining Vios Waived Vios Used Waivers Set Waivers
----- ----- ----- ----- -----
LUTAR-1 Warning LUT drives async reset alert 49 45 4 4 4

-----
3. REPORT DETAILS (METHODOLOGY)

Rule Severity Description Total Vios Remaining Vios Waived Vios Used Waivers Set Waivers
----- ----- ----- ----- -----
CDC-1 Critical 1-bit unknown CDC circuitry 534 530 4 4 4
CDC-11 Critical Fan-out from launch flop to destination clock 2 0 2 2 2

Note: Any 'Rule' which is flagged by '*' is an aggregating message and its counts are based on the number of objects represented,
rather than the number of messages.
```

通过使用命令行选项 `-show_msgs_with_no_waivers`，详细表格可报告含违例的所有检查，与此项特定规则是否存在豁免无关。

图 132: report_waivers -show_msgs_with_no_waivers

```

-----
Table Of Contents
-----
1. REPORT SUMMARY
2. REPORT DETAILS (DRC: no waivers)
3. REPORT DETAILS (METHODOLOGY)
4. REPORT DETAILS (CDC)

-----
1. REPORT SUMMARY
-----
Waiver Type Total Vios Remaining Vios Waived Vios Used Waivers Set Waivers
-----
DRC 240 240 0 0 0
METHODOLOGY 166 162 4 4 4
CDC 929 923 6 6 6
Note: This report is based on the most recent report_drc/report_methodology/report_cdc runs.

-----
2. REPORT DETAILS (DRC)
-----
Rule Severity Description Total Vios Remaining Vios Waived Vios Used Waivers Set Waivers
-----
LOCE-1 Warning Pblock ranges contradict LOC constraints on logic assigned to the Pblock 1 1 0 0 0
NSTD-1 Critical Warning Unspecified I/O Standard 113 113 0 0 0
RISTAT-13 Critical Warning Insufficient Routing 1 1 0 0 0
UCIO-1 Critical Warning Unconstrained Logical Port 125 125 0 0 0

-----
3. REPORT DETAILS (METHODOLOGY)
-----
Rule Severity Description Total Vios Remaining Vios Waived Vios Used Waivers Set Waivers
-----
LUTAR-1 Warning LUT drives async reset alert 49 45 4 4 4
TIMING-9 Warning Unknown CDC Logic 1 1 0 0 0
TIMING-10 Warning Missing property on synchronizer 1 1 0 0 0
TIMING-18 Warning Missing input or output delay 115 115 0 0 0

-----
4. REPORT DETAILS (CDC)
-----
Rule Severity Description Total Vios Remaining Vios Waived Vios Used Waivers Set Waivers
-----
CDC-1 Critical 1-bit unknown CDC circuitry 534 530 4 4 4
CDC-11 Critical Fan-out from launch flop to destination clock 2 0 2 2 2
CDC-3 Info 1-bit synchronized with ASYNC_REG property 9 9 0 0 0
CDC-4 Critical Multi-bit unknown CDC circuitry 5 5 0 0 0
CDC-9 Info Asynchronous reset synchronized with ASYNC_REG property 7 7 0 0 0
CDC-10 Critical Combinational logic detected before a synchronizer 187 187 0 0 0
CDC-13 Critical 1-bit CDC path on a non-FD primitive 170 170 0 0 0
CDC-14 Critical Multi-bit CDC path on a non-FD primitive 5 5 0 0 0
CDC-15 Warning Clock enable controlled CDC structure detected 10 10 0 0 0

Note: Any 'Rule' which is flagged by '*' is an aggregating message and its counts are based on the number of objects represented,
rather than the number of messages.

```

除上述报告外，report_waivers 还可导出具有匹配的 CDC、DRC 或 Methodology 违例的豁免列表以及不含任何匹配的违例的豁免列表。-write_valid_waivers 选项可用于导出含匹配的违例的豁免，而 -write_ignore_waivers 可用于导出不含任何匹配的违例的豁免。建议复查不匹配任何违例的豁免列表。当不匹配的豁免不符合期望时，请确保这些豁免的定义正确。

-write_valid_waivers 和 -write_ignore_waivers 选项可根据最近执行的 report_cdc、report_drc 和 report_methodology 命令所报告的信息来对豁免进行筛选。当通过规则卡或少量检查运行 report_drc 或 report_methodology 时，对于尚未运行的检查可忽略部分豁免。建议在使用 -write_valid_waivers 和 -write_ignore_waivers 前运行所有 DRC/Methodology 检查。例如：

```

report_cdc -all_checks_per_endpoint
report_drc -checks [get_drc_checks]
report_methodology -checks [get_methodology_checks]
report_waivers -write_valid_waivers -file waivers_valid.xdc
report_waivers -write_ignored_waivers -file waivers_ignored.xdc

```

导出豁免

作为设计约束的一部分，豁免将自动保存在检查点内，并从检查点恢复。豁免保存在明文 XDC 和二进制约束中。

`write_xdc` 和 `write_waivers` 命令可用于将豁免导出为独立 XDC 文件。可通过 `read_xdc` 或 `source` 命令将 XDC 重新加载到 Vivado 工具中。

`write_xdc` 命令可将 XDC 文件内的所有豁免随所有设计约束一并导出。包括用户定义的豁免和赛灵思 IP 豁免。XDC 中的约束排序与其应用于设计的顺序相同。如需仅导出豁免，请使用命令行选项 `-typewaiver`。例如：

```
write_xdc -type waiver -file waivers.xdc
```



重要提示！IP 豁免以 `create_waiver -internal` 选项来识别。用户豁免不得使用 `create_waiver -internal` 选项。该选项是赛灵思 IP 豁免专用的保留选项。

`write_waivers` 命令不同于 `write_xdc`，因为前者仅导出用户豁免，可提供更高的控制权和颗粒度。赛灵思 IP 豁免无法通过 `write_waivers` 导出。默认情况下，将导出所有用户 CDC、DRC 和 Methodology 豁免。`-type` 选项仅用于导出 CDC、DRC 和 Methodology 豁免。

例如，以下命令将把所有 CDC 豁免导出至 `waivers_cdc.xdc` 文件：

```
write_waivers -type CDC -file waivers_cdc.xdc
```

可通过 `-id` 选项导出特定检查 ID 的所有豁免。以下示例将导出方法检查 TIMING-15 的所有豁免：

```
write_waivers -id TIMING-15 -file waivers_timing_15.xdc
```

下表汇总了 `write_xdc` 命令与 `write_waivers` 命令之间有关用户豁免和赛灵思 IP 豁免的差异。

表 12：导出豁免

Vivado 命令	导出用户豁免	导出赛灵思 IP 豪免
<code>write_xdc</code>	支持	支持
<code>write_waivers</code>	支持	不支持

其它豁免命令

`get_waivers` 命令用于返回豁免对象的集合。豁免可按类型、名称或模式返回。

以下命令可返回所有 DRC 豁免：

```
get_waivers -type DRC  
get_waivers -filter {TYPE == DRC}
```

以下命令可返回所有 DRC DPIR-2 豁免：

```
get_waivers DPIR-2#*  
get_waivers -filter {ID == DPIR-2}  
get_waivers -filter {NAME =~ DPIR-2#*}
```

注释：`get_waivers` 命令无法返回赛灵思 IP 豪免。

`delete_waivers` 命令用于删除用户豁免对象。必须从 `get_waivers` 构建豁免对象集合。

以下命令用于删除所有豁免：

```
delete_waivers [get_waivers]
```

以下命令用于删除所有 CDC 豁免：

```
delete_waivers [get_waivers -type CDC]
```

注释：无法删除赛灵思 IP 豁免。

可配置的报告策略

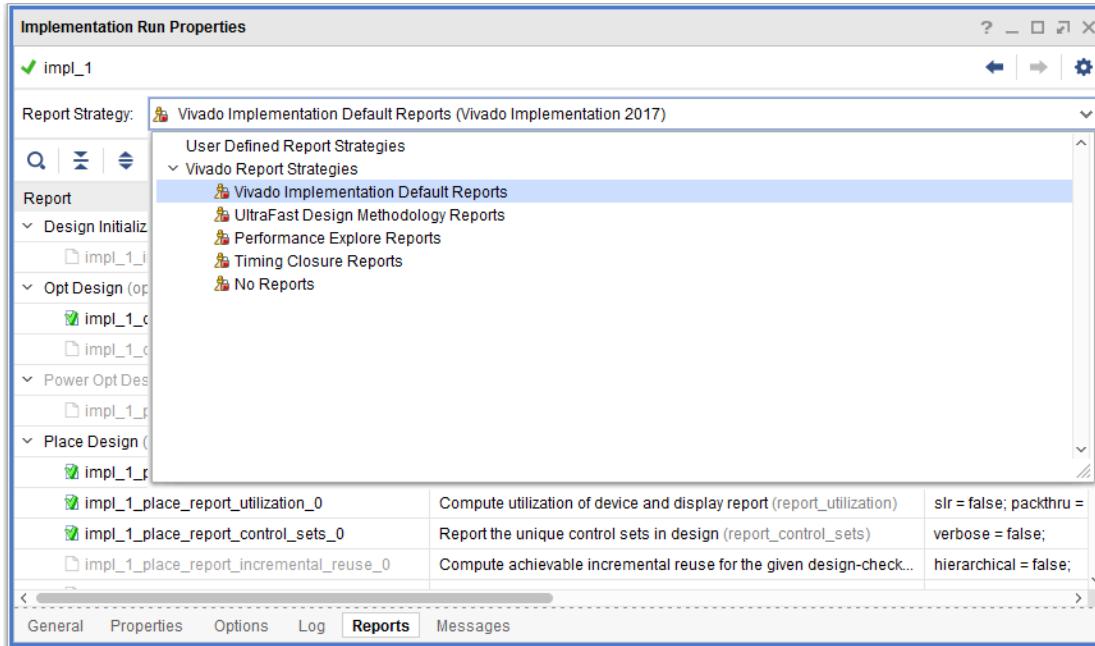
可配置的报告策略 (Configurable Report Strategies) 支持选择在 Vivado 工程模式下运行综合与实现的每个步骤后运行的报告命令。根据设计阶段、设计复杂性和用户首选项，需自动生成一组不同的报告以供频繁查看。默认情况下，有多项预定义的综合和实现报告策略可供使用。此外，Vivado® IDE 支持创建新报告策略，并将这些策略与用户首选项和任何其它 Vivado IDE 设置保存在一起。

设置运行报告策略

默认情况下，所有综合和实现运行都将使用其对应的默认报告策略 (Reports Strategy)。要设置其它报告策略，请执行以下操作：

1. 在“设计运行 (Design Runs)”窗口中，选择运行。
2. 选择“运行属性 (Run Properties)”窗口中的“Reports”选项卡。
3. 从“报告策略 (Report Strategy)”下拉列表中选择策略。

图 133：选择对应“实现运行 (Implementation Run)”的“Report Strategy”。



“流程 (Flow)” 分 2 个类别，每个类别都可提供多项预定义的报告策略以及任何用户定义的策略。

表 13：流程和报告策略

流程	报告策略	注
综合	Vivado 综合默认报告	仅在综合结束后运行利用率报告
	无报告	最大限度缩短运行时间的最佳策略
实现	Vivado 实现默认报告	与低于 2017.3 的 Vivado 版本运行相同的报告
	UltraFast 设计方法报告	运行《UltraFast 设计方法指南（适用于 Vivado Design Suite）》(UG949) 中建议的所有报告
	性能浏览报告	与默认报告相同
	时序收敛报告	与 UltraFast 设计方法报告相同，此外还会运行 report_design_analysis 和 report_qor_suggestions 报告
	无报告	最大限度缩短运行时间的最佳策略

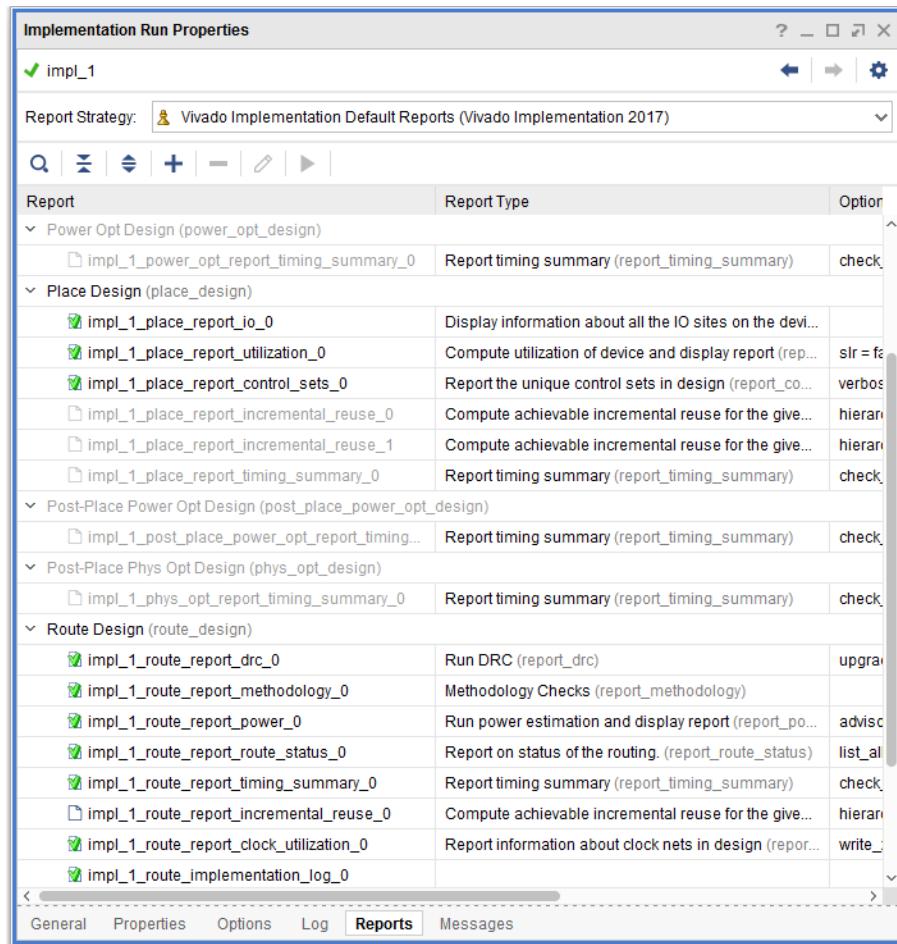
启动运行前，由于以下某一原因，部分报告可能为灰显：

- 报告与已禁用的流程步骤关联
- 报告已被用户禁用

运行完成后，所有可用报告都包含绿色勾选标记，并可通过在“Reports”选项卡上双击打开。由于以下某一原因，部分报告不可用：

- 报告与已禁用的流程步骤关联
- 报告已被用户禁用
- 报告仅在“增量编译 (Incremental Compile)”运行中启用

图134：查看生成的运行报告



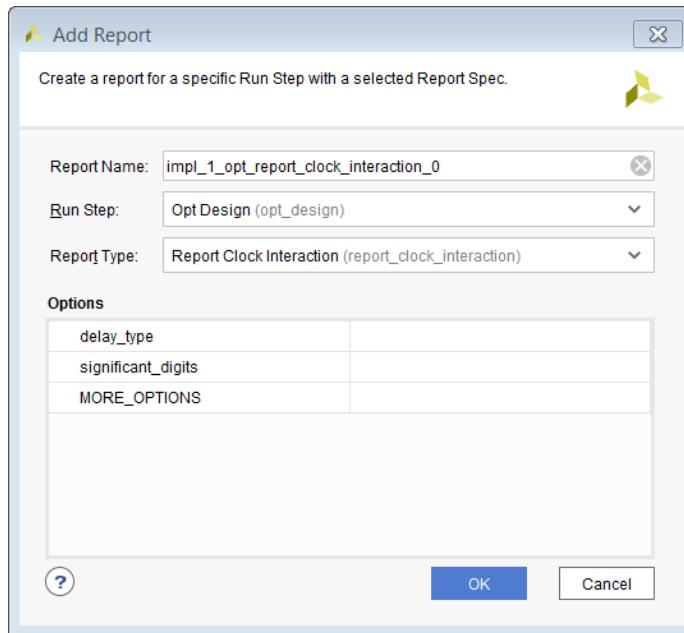
提示：设置旧工程属性 ENABLE_OPTIONAL_RUNS_STA 时，仅生成少量时序报告。赛灵思保留在将来发行版中删除该属性的权力。示例：set_property ENABLE_OPTIONAL_RUNS_STA 1 [current_project]

编辑运行报告策略

选择运行的报告策略 (Report Strategy) 后，即可通过选择报告然后单击“Reports”选项卡中的对应按钮或者右键单击“运行属性 (Run Properties)”窗口中的任意现有报告来决定要添加、删除还是编辑报告。

- 添加报告 (Add Report) (+)：选择“运行步骤 (Run Step)”和“报告类型 (Report Type)”，然后复查并编译报告选项。如果某个选项不可见，您可通过 MORE_OPTIONS 字段来添加该选项。默认唯一报告名称是基于运行名称、流程步骤和报告命名名称生成的。

图 135：向“运行报告策略”添加报告



- 删除报告 (Delete Report) (-): 此按钮用于从“运行报告策略 (Run Report Strategy)”删除选定报告。此操作不可撤销。

- 编辑报告 (Edit Report) (): 编辑报告名称、启用或禁用报告或者编辑报告选项。

要启用 (Enable) 或禁用 (Disable) 报告，请选中报告、右键单击并使用上下文弹出菜单。

完成运行后，即可为特定步骤添加新报告，或者启用先前已禁用的报告。在此情况下，必须单击运行按钮才能生成报告。



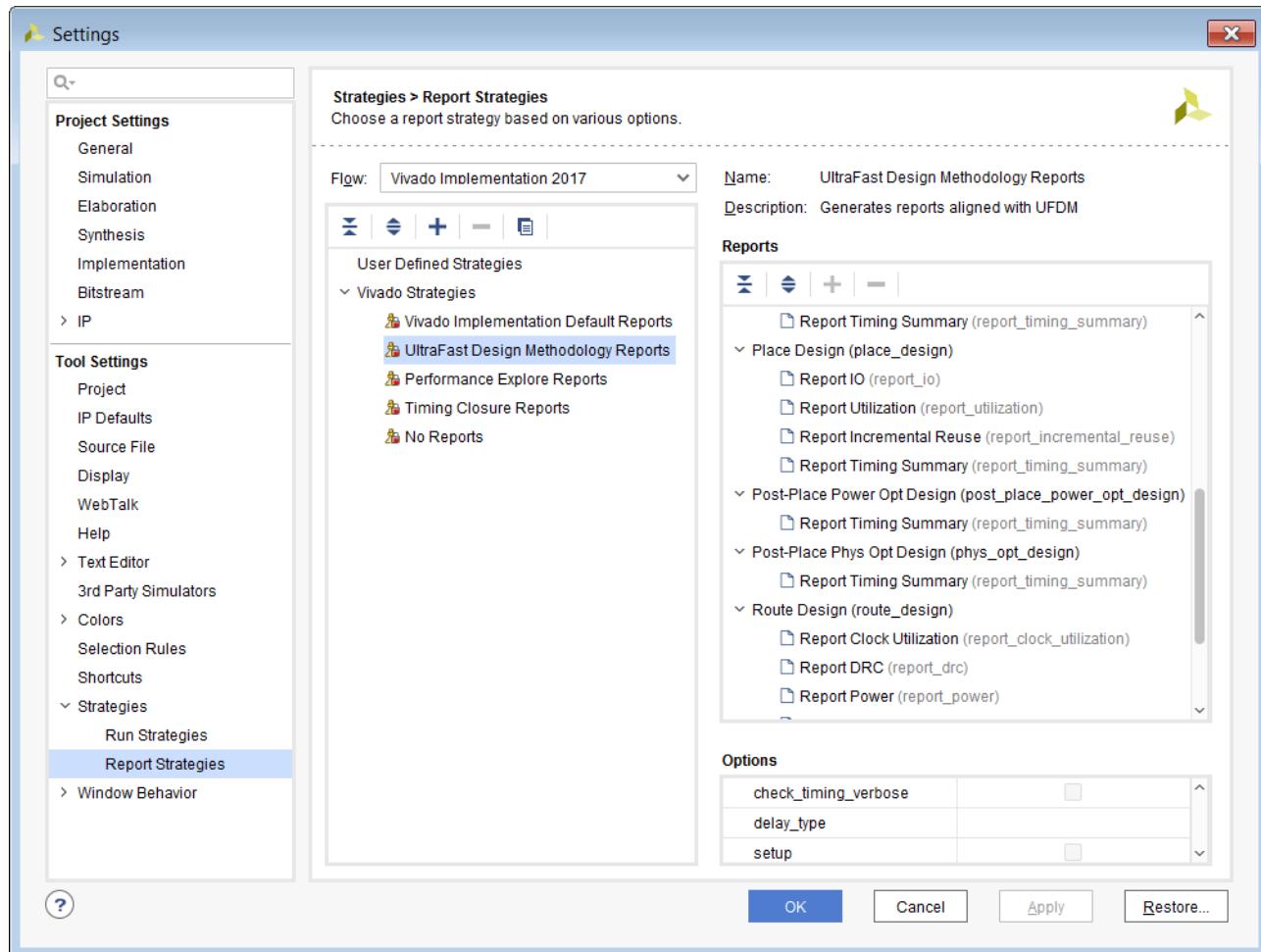
重要提示！ 运行完成后生成新报告时，Vivado 会在后台打开对应流程步骤的检查点，以生成报告文件。此操作会阻止大部分 Vivado IDE 功能（包括 Tcl 控制台）直至成功生成报告为止。生成报告可能需要几分钟到 1 小时时长，具体取决于您的设计大小以及报告的复杂性。

对特定运行的“Report Strategy”执行的任意修改都无法保存为新报告策略。您必须改为在“Project Settings”窗口中创建新报告策略，或者修改现有用户定义的报告策略，如下文所述。

创建新的报告策略 (Report Strategies)

新“Report Strategies”必须在 Vivado IDE “Project Settings”窗口中的“Tool Settings”下的“Strategies”下创建。同样，只能通过“Project Settings”窗口对任意现有用户定义的“Report Strategies”进行永久性修改。预定义的“Report Strategies”以及“Run Strategy”与“Report Strategy”的默认关联无法修改。

图 136：“Settings”窗口的“Report Strategies”配置



要创建新的报告策略：

1. 请单击“Strategy”窗口中的“+”，并执行下列步骤：
 - a. 指定策略的名称。
 - b. 选择目标“流程 (Flow)”、“综合 (Synthesis)”或“实现 (Implementation)”。
 - c. 提供描述（可选）。
 - d. 单击“OK”。
- 或：
2. 选择现有策略、单击“copy”按钮，并编辑策略名称。然后：
 - a. 使用“+”按钮添加报告。
 3. 选择报告并编辑报告选项。对于不可用的选项，可在 MORE_OPTIONS 字段中添加该选项。
 4. 使用“-”按钮移除报告。
 5. 完成所有编辑后，请单击“OK”。

每个流程类别均提供多项预定义的“Report Strategies”。请参阅[表 13：流程和报告策略](#)。

创建设计相关报告

本部分将讨论如何创建设计相关报告，包括：

- [利用率报告 \(Report Utilization\)](#)
- [I/O 报告 \(Report I/O\)](#)
- [“时钟利用率 \(Clock Utilization\)” 报告](#)
- [控制集报告](#)
- [DRC 报告 \(Report DRC\)](#)
- [布线状态报告 \(Report Route Status\)](#)
- [噪声报告 \(Report Noise\)](#)
- [功耗报告 \(Report Power\)](#)
- [RAM 使用情况报告 \(Report RAM Utilization\)](#)

利用率报告 (Report Utilization)

“Report Utilization” 报告有助于您从层级、用户定义的 Pblock 或 SLR 层面来分析含不同资源的设计的利用率。您可在流程中各步骤间使用 `report_utilization` Tcl 命令生成 “Utilization Report”。(如需了解有关 Tcl 命令使用方式的详细信息，请参阅《Vivado Design Suite Tcl 命令参考指南》([UG835](#))。) 以下显示的报告详细信息适用于 UltraScale 系列和 UltraScale+ 系列。其中包含用于运行和使用以下对象的器件（每个类别中可能包含其它项）：

- slice 逻辑
 - LUT
 - MuxFx
 - 寄存器
 - slice
 - LUT (作为内存)
 - LUT 触发器对
 - LUT (作为逻辑)
- 内存
 - 块 RAM
 - FIFO
- DSP
- I/O 资源
- 时钟资源
- 特定的器件资源。示例：
 - STARTUPE2

- XADC
- 原语类型计数（按利用率排序）
- 黑盒
- 例化网表
- SLR 交汇利用率

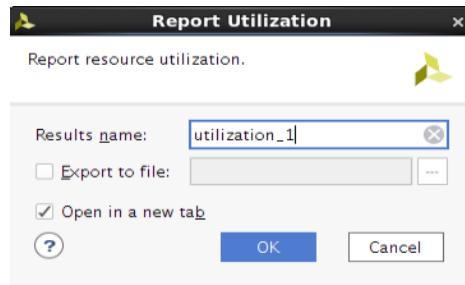
从 Tcl 控制台运行时，此报告可包含使用 `-cells` 选项时的特定层级单元的使用情况。从 Vivado IDE 运行时，此信息会显示在交互表格中。

当逻辑最优化命令导致网表发生更改时，流程中各时间点显示的数值可能不尽相同。

运行“利用率报告 (Report Utilization)”

要从 Vivado IDE 生成“利用率报告 (Utilization Report)”，请选择“Reports” → “Report Utilization”。

图 137：“Report Utilization”对话框



“结果名称 (Results Name)”字段

在“Report Clock Utilization”对话框顶部的“Results Name”字段中指定结果窗口的名称。

等效的 Tcl 命令：

```
report_utilization -name utilization_1
```

下图显示了详细利用率报告。

图 138：利用率报告 (Report Utilization)

Name	^1	Slice LUTs (41000)	Slice Registers (82000)	F7 Muxes (20500)	F8 Muxes (10250)	Slice (10250)	LUT as Logic (41000)	LUT as Memory (13400)	LUT Flip Flop Pairs (41000)	Block RAM Tile (135)	DSPs (240)	Bon (
top		18863	15635	884	174	6123	18854	9	7422	109	68	
clkgen (clock_gene...		0	0	0	0	0	0	0	0	0	0	0
> cpuEngine (or1200...		5335	3773	336	10	1719	5335	0	1261	21	4	
> fftEngine (fftTop)		1542	1487	0	0	754	1541	1	459	16	64	
> mgtEngine (mgtTop)		371	642	0	0	209	371	0	204	0	0	
> usbEngine0 (usbf_t...		5659	4642	258	74	1796	5655	4	2624	36	0	
> usbEngine1 (usbf_t...		5664	4642	258	74	1855	5660	4	2625	36	0	
> wbArbEngine (wb_c...		317	430	32	16	192	317	0	80	0	0	

可使用报告窗口中的按钮切换显示利用率数值或利用率百分比。

图 139：利用率报告百分比切换

The screenshot shows the Xilinx Vivado Hierarchy window. At the top, there is a toolbar with search, filter, and sort icons. Below the toolbar is a header bar labeled "Hierarchy". In the center, there is a table with columns: "Name", "%", and "Slice LUT (41000)". The first row shows "Slice LUT (41000)" with a percentage value of 1. The second row is expanded to show "top" with a value of 1886, and under it is "clkgen (clock_gene...)".

Name	%	Slice LUT (41000)
Slice LUT (41000)	1	
top	1886	
clkgen (clock_gene...)		

显示特定单元的利用率

选择 `-cells {cell_name_list}` 选项时，生成报告会显示指定单元及其子单元的利用率。

```
-cells {cell_name_list}
```

可从目标单元级别执行特定单元：

```
-exclude_cells {cell_name_list}
```

显示特定 Pblock 的利用率

选择以下选项时，利用率报告可反映 Pblock 的特性，例如嵌套的子级 Pblock 和重叠的 Pblock。仅支持在 Tcl 模式下使用这些命令行选项。

```
-pblocks {pblock_list}
-exclude_child_pblocks {child_pblock_list}
-exclude_non_assigned
```

在此模式下，文本报告可额外显示与指定父级 Pblock 相关的 2 个表，还会一并打印子级 Pblock。

图 140：Pblock 汇总报告

The screenshot shows two tables from the Xilinx Vivado Text Report. The first table, titled "1. Pblock Summary", has columns: Index, Parent, Child, EXCLUDE_PLACEMENT, CONTAIN_ROUTING, SLR(s) Covered. The second table, titled "2. Clock Region Statistics", has columns: CLOCKREGION, Pblock Sites in CR. Both tables are displayed as tabular data.

1. Pblock Summary					
Index	Parent	Child	EXCLUDE_PLACEMENT	CONTAIN_ROUTING	SLR(s) Covered
1	pblock_cpuEngine		0	0	SLR0
2	pblock_fftEngine		0	0	SLR0
3			0	0	SLR0
4	pblock_usbEngine1	pblock_usbEngine0	0	0	SLR0

2. Clock Region Statistics	
CLOCKREGION	Pblock Sites in CR
XOY0	18.19%
XOY1	13.24%
XOY2	11.02%
XOY3	14.41%
XIY0	14.21%
XIY1	14.21%
XIY2	6.66%
XIY3	8.05%

布局前和布局后的利用率数值因 LUT 组合以及未分配单元（布局前无法考量）而异。

使用 Pblock 时，利用率表包含以下附加列：

- 父级 (Parent): 仅分配到父级 Pblock

- 子级 (Child): 仅分配到子级 Pblock
- 已用 (Used): 指定 Pblock 定义的区域内已用资源总量
- 固定 (Fixed): 指定 Pblock 已定义区域内 LOC 约束已固定的资源总量
- 未分配 (Non-Assigned): 位于指定 Pblock 定义的区域内且已分配至指定 Pblock 及其子级 Pblock 的资源总量。
- 可用 (Available): 指定 Pblock 定义的区域内可用资源总量。
- 利用率 (Util%): 已用/可用

图 141: 表头文件

Site Type	Parent	Child	Non-Assigned	Used	Fixed	Available	Util%
-----------	--------	-------	--------------	------	-------	-----------	-------

以下示例有助于更全面理解此报告。下图显示了设计层级示例，后图显示了 Pblock 矩形，并高亮显示了每个 Pblock 内来自布线后网表的资源。

图 142: 设计层级示例

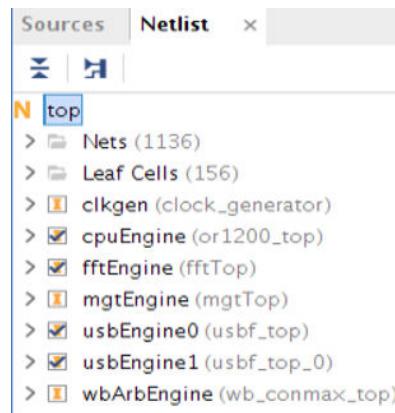
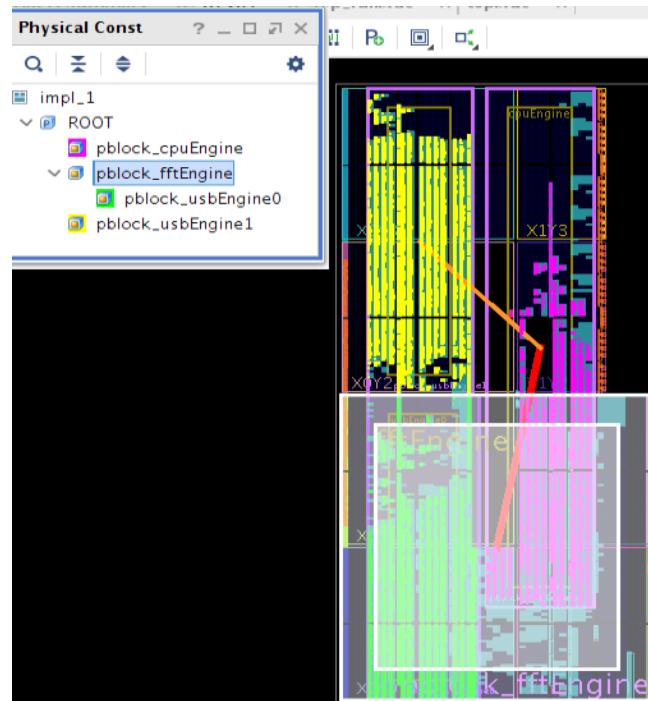


图143：设计 Pblock



在此示例中：

- pblock_usbEngine1 Pblock 不含任何子级 Pblock
- pblock_fftEngine Pblock 含 1 个子级 Pblock：pblock_usbEngine0
- pblock_cpuEngine Pblock 与 pblock_fftEngine 重叠

要生成整个设计的报告，请 report_utilization（运行不含任何选项）。

图144：顶层利用率报告

Site Type	Used	Fixed	Available	Util%
Slice LUTs	18863	0	41000	46.01
LUT as Logic	18854	0	41000	45.99
LUT as Memory	9	0	13400	0.07
LUT as Distributed RAM	0	0		
LUT as Shift Register	9	0		
Slice Registers	15635	0	82000	19.07
Register as Flip Flop	15635	0	82000	19.07
Register as Latch	0	0	82000	0.00
F7 Muxes	884	0	20500	4.31
F8 Muxes	174	0	10250	1.70

要生成 pblock_usbEngine1 Pblock 的报告，请使用以下命令：

```
report_utilization -pblocks pblock_usbEngine1
```

图 145：Pblock pblock_usbEngine1 的利用率报告

Site Type	Parent	Child	Non-Assigned	Used	Fixed	Available	Util%
Slice LUTs	5664	0	49	5713	0	9600	59.51
LUT as Logic	5660	0	49	5709	0	9600	59.47
LUT as Memory	4	0	0	4	0	4000	0.10
LUT as Distributed RAM	0	0	0	0	0		
LUT as Shift Register	4	0	0	4	0		
Slice Registers	4642	0	28	4670	0	19200	24.32
Register as Flip Flop	4642	0	28	4670	0	19200	24.32
Register as Latch	0	0	0	0	0	19200	0.00
F7 Muxes	258	0	0	258	0	4800	5.38
F8 Muxes	74	0	0	74	0	2400	3.08

要生成 pblock_fftEngine Pblock 的报告，请使用以下命令。在此例中，嵌套的子级 Pblock pblock_usbEngine0 资源已计入已用资源总量。

注释：如果将属性 EXCLUDE_PLACEMENT 应用于子级 Pblock，那么子级资源将与父级 Pblock（包括已用和可用）隔离。

重叠的 Pblock pblock_cpuEngine 中部分单元布局在 pblock_fftEngine Pblock 范围内，并作为外部资源报告为“未分配”。

```
report_utilization -pblocks pblock_fftEngine
```

图 146：含嵌套和重叠子级 Pblock 的父级 Pblock

Site Type	Parent	Child	Non-Assigned	Used	Fixed	Available	Util%
Slice	754	1796	1512	3731	0	5600	66.63
SLICEL	513	1081	1055	2458	0		
SLICEM	241	715	457	1273	0		
LUT as Logic	1541	5655	4649	11822	0	22400	52.78
using 05 output only	12	11	0	0			
using 06 output only	974	4651	3986	9588			
using 05 and 06	555	993	663	2234			
LUT as Memory	1	4	0	5	0	7200	0.07
LUT as Distributed RAM	0	0	0	0	0		
LUT as Shift Register	1	4	0	5	0		
using 05 output only	1	0	0	1			
using 06 output only	0	0	0	0			
using 05 and 06	0	4	0	4			
LUT Flip Flop Pairs	459	2624	954	4098	0	22400	18.29
fully used LUT-FF pairs	240	204	145	603			
LUT-FF pairs with one unused LUT output	219	2182	713	3125			
LUT-FF pairs with one unused Flip Flop	147	2315	772	3274			
Unique Control Sets	39	202	118	354			

要排除部分 Pblock 或未分配资源，请使用 -exclude_child_pblocks 或 -exclude_non_assigned 开关。以下示例显示报告中已移除“Non-Assigned”列。

```
report_utilization -pblocks [get_pblocks pblock_fftEngine] -  
exclude_non_assigned
```

图 147：不含未分配资源的利用率报告

Site Type	Parent	Child	Used	Fixed	Available	Util%
Slice	754	1796	2395	0	5600	42.77
SLICEL	513	1081	1521	0		
SLICEM	241	715	874	0		
LUT as Logic	1541	5655	7184	0	22400	32.07
using 05 output only	12	11	11			
using 06 output only	974	4651	5613			
using 05 and 06	555	993	1560			
LUT as Memory	1	4	5	0	7200	0.07
LUT as Distributed RAM	0	0	0	0		
LUT as Shift Register	1	4	5	0		
using 05 output only	1	0	1			
using 06 output only	0	0	0			
using 05 and 06	0	4	4			
LUT Flip Flop Pairs	459	2624	3088	0	22400	13.79
fully used LUT-FF pairs	240	204	444			
LUT-FF pairs with one unused LUT output	219	2182	2403			
LUT-FF pairs with one unused Flip Flop	147	2315	2465			
Unique Control Sets	39	202	239			

下表描述了各种场景的报告内容。

表 14：含 Pblock 分配的报告表

案例	标题	描述	报告
1	整个器件报告（根 Pblock）： report_utilization	EXCLUDE_PLACEMENT 对利用率报告无影响。	利用率 (Util%): 已用/可用
2	父级 Pblock 报告： report_utilization -pblocks <parentPblockName>	子级 Pblock 嵌套在父级 Pblock 内。针对子级 Pblock 未指定 EXCLUDE_PLACEMENT 属性。	未分配 (Non-Assigned): 父级 Pblock 边界内已布局但未分配给父级或子级 Pblock 的单元总数 固定 (Fixed): 父级 Pblock 边界内已固定的单元总数 已用 (Used): 父级 Pblock 边界内已布局的父级 + 子级 + 未分配单元数量 可用 (Available): 父级 Pblock 边界内物理资源总数 利用率 (Util%): 已用/可用
		子级 Pblock 嵌套在父级 Pblock 内。针对子级 Pblock 已指定 EXCLUDE_PLACEMENT 属性。已报告的区域对应于父级 Pblock 范围减去子级 Pblock 范围。	未分配 (Non-Assigned): 报告区域内已布局但未分配给父级 Pblock 和子级 Pblock 的单元总数。 固定 (Fixed): 报告区域内已固定的单元总数 已用 (Used): 父级 Pblock 单元数 (不包括子级 Pblock 单元数) 可用 (Available): 报告区域内物理资源总数 利用率 (Util%): 已用/可用
3	报告父级 Pblock 和子级 Pblock： report_utilization -pblocks {<parentPblockName> <childPblockName>}	指定未设置 EXCLUDE_PLACEMENT 的子级 Pblock 为冗余 Pblock。 如果子级 Pblock 已设置 EXCLUDE_PLACEMENT，则此报告等效于父级 Pblock 和子级 Pblock 的并集。	与 显示特定 Pblock 的利用率 中的首个案例相同。
4	报告重叠 Pblock	类似于默认 Pblock 报告，但“Available”变为已报告的 Pblock 的并集。忽略 EXCLUDE_PLACEMENT 属性。	可用 (Available): Pblock 物理资源的并集 利用率 (Util%): 已用/可用

显示 SLR 使用情况

选择 `-slr` 选项时，生成的报告会显示 SLR 相关使用情况。从 Vivado® Design Suite 2018.3 起，SLR 使用情况表在 GUI 和文本报告中已实现功能增强，包含以下 4 个不同表：

- SLR 连接情况 (SLR Connectivity)
- SLR 连接矩阵 (SLR Connectivity Matrix)
- SLR CLB 逻辑和专用块使用情况 (SLR CLB Logic and Dedicated Block Utilization)
- SLR IO 使用情况 (SLR IO Utilization)

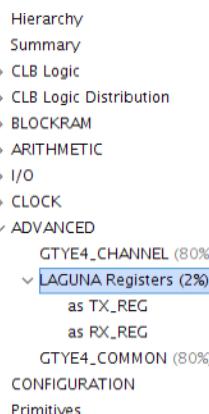
默认情况下，运行 `report_utilization` 时也会显示这些表。“SLR Connectivity” 表可显示用于 SLR 每侧的 TX 和 RX 方向的 LAGUNA 寄存器。

图 148：GUI 中的 SLR Connectivity 报告

Connectivity	^ 1	Used	Fixed	Available
SLR1 <-> SLR0	229		23040	
SLR0 -> SLR1	0			
Using Both TX_REG and RX_REG	0	0		
Using RX_REG only	0	0		
Using TX_REG only	0	0		
SLR1 -> SLR0	229			
Using Both TX_REG and RX_REG	0	0		
Using RX_REG only	0	0		
Using TX_REG only	0	0		
SLR2 <-> SLR1	311		23040	
SLR1 -> SLR2	0			
Using Both TX_REG and RX_REG	0	0		
Using RX_REG only	0	0		
Using TX_REG only	0	0		
SLR2 -> SLR1	311			
Using Both TX_REG and RX_REG	0	0		
Using RX_REG only	0	0		
Using TX_REG only	0	0		

在对应 2 个方向的 ADVANCED 表中还会报告 LAGUNA 的使用总量。

图 149：GUI 中的 LAGUNA 寄存器报告



“SLR CLB Logic and Dedicated Block Utilization” 表可显示每个 SLR 的资源使用情况。

图 150：每个 SLR 中的使用情况

Site Type	SLR0	SLR1	SLR2
Block RAM Tile	0	0	197
RAMB18	0	0	10
RAMB36/FIFO	0	0	192
CARRY8	0	2	1004
CLB	0	0	0
CLBL	3	43	11750
CLBM	3	0	1019
CLB LUTs	43	344	102149
LUT as Logic	19	344	93998
LUT as Memory	24	0	8151
LUT as Distributed RAM	0	0	5275
LUT as Shift Register	24	0	2876
CLB Registers	89	609	104659
DSPs	0	0	0
F7 Muxes	0	0	1302
F8 Muxes	0	0	242
F9 Muxes	0	0	0
MMCM	0	0	0
PLL	0	0	0
Unique Control Sets	7	51	4514
URAM	0	0	0

显示含自定义选项的层级信息

选择以下选项时，可将报告限制为显示部分特定层级的相关信息。根据层级报告使用情况时，可指定要报告的层级深度。默认深度为 0，即默认情况下 `-hierarchical` 仅报告顶层相关信息。

```
-hierarchical  
-hierarchical_depth <args>  
-hierarchical_percentage
```

显示自定义表格报告

选中以下选项时，可自定义报告，仅显示某些类型的资源及层级深度。

```
-spreadsheet_table <args>  
-spreadsheet_depth
```

I/O 报告 (Report I/O)

“I/O 报告 (I/O Report)” 用于取代赛灵思 ISE® Design Suite PAD 文件。“I/O Report” 可列出：

- 管脚编号 (Pin Number)：器件中的所有管脚
- 信号名称 (Signal Name)：分配给管脚的用户 I/O 的名称
- Bank 类型 (Bank Type)：I/O 所在的 bank 类型（高量程 (High Range)、高性能 (High Performance)、专用 (Dedicated) 等）
- 管脚名称 (Pin Name)：管脚名称
- 用途 (Use)：I/O 使用类型（输入 (Input)、输出 (Output)、功耗/接地 (Power/Ground)、未连接 (Unconnected) 等）
- I/O 标准 (I/O Standard)：用户 I/O 的 I/O 标准

任意星号 (*) 均表示默认值。这不同于 Vivado IDE 的“*I/O Ports*”窗口。

- I/O bank 编号 (I/O Bank Number): 管脚所在的 I/O Bank
- 驱动 (Drive) (mA): 驱动强度 (以毫安为单位)
- 压摆率 (Slew Rate): 缓存的压摆率配置: 值为 Fast 或 Slow
- 终端 (Termination): 片上/片外终端设置
- 电压 (Voltage): 各管脚的值, 包括 VCCO、VCCAUX 和相关管脚
- 约束 (Constraint): 如果管脚已由用户约束, 则显示为“Fixed”
- 信号完整性 (Signal Integrity): 管脚的信号完整性。

“时钟利用率 (Clock Utilization)” 报告

“Clock Utilization” 报告可帮助您分析器件内时钟区域级别或时钟信号线级别的时钟原语和布线资源的利用率。它可用于调试时钟布局问题, 并识别布局约束, 从而最大限度提升资源利用率。“Clock Utilization” 报告提供如下相关信息:

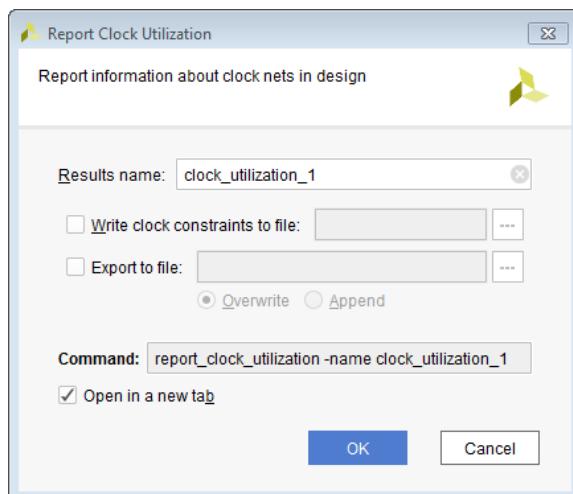
- 可用和已用时钟原语的数量及其物理约束。
- 时序时钟名称以及与每条时钟信号线关联的周期
- 每个时钟区域的时钟设置和结构负载利用率
- 每个时钟区域内的每条时钟信号线的负载

此外, Vivado IDE 中的“Clock Utilization” 报告支持选择网表和器件对象, 以便高亮显示布局信息和创建原理图。

运行“时钟利用率报告 (Report Clock Utilization)”

要在 Vivado IDE 中生成“时钟利用率报告 (Clock Utilization Report)”, 请选择“Reports” → “Report Clock Utilization”。

图 151: “Report Clock Utilization” 对话框



等效的 Tcl 命令：

```
report_clock_utilization -name clock_utilization_1
```

“结果名称 (Results Name)” 字段

在 “Report Clock Utilization” 对话框的 “Results Name” 字段中，指定报告的图形窗口的名称。

等效的 Tcl 选项：

```
-name <windowName>
```

仅显示时钟根 (Show Clock Roots Only)

选中该选项时，“全局时钟资源 (Global Clock Resources)” 表仅显示每个时钟信号线的时钟根位置，而不显示完整的源时钟、负载时钟和时序时钟详情。

等效的 Tcl 选项：

```
-clock_roots_only
```

将时钟约束写入文件 (Write Clock Constraints to File)

选择该选项并指定新约束文件的名称以导出时钟源并加载对应于内存中的设计布局信息的物理约束。

等效的 Tcl 选项：

```
-write_xdc <filename>
```

导出至文件 (Export to File)

除生成 GUI 报告外，您还可通过选择 “Export to file” 并在右侧字段中指定文件名来将结果写入文件。单击 “Browse” 按钮可选择其它目录。

等效的 Tcl 选项：

```
-file <arg>
```

选择 “Overwrite” 选项可使用新的分析结果来覆盖现有文件。选择 “Append” 可追加新结果。

等效的 Tcl 选项：

```
-append
```

“时钟利用率 (Clock Utilization)” 报告表

此报告提供了时钟拓扑和布局信息，按类别组织：

- 时钟原语利用率 (Clock Primitive Utilization)
- 全局时钟资源 (Global Clock Resources)
- 全局时钟源详情 (Global Clock Source Details)
- 局部时钟资源 (Local Clock Resources)

- 时钟区域利用率详情 (Clock Regions utilization details)
- 全局时钟布局详情 (Global Clocks placement details)

由于典型设计中网表对象名称较长且时钟信号线和时钟原语数量庞大，因此对特定时钟资源分配短 ID：

- 针对时钟缓存驱动的每条信号线分配唯一全局 ID “`g<n>`”
- 针对时钟生成器（例如，MMCM 或输入缓存）分配唯一源 ID “`src<n>`”
- 对于不使用全局时钟资源布线的时钟信号分配唯一局部 ID “`<n>`”。

“全局源 ID” 和 “局部 ID” 可简化整个报告中特定时钟信号线的搜索操作。在每个表的最后 2 列（如果适用）中提供了原始网表对象名称。

“Clock Primitive Utilization” 表

“Clock Primitive Utilization” 表可显示每种时钟原语类型及其物理约束的使用情况汇总。

图 152: Report Clock Utilization - “Clock Primitive Utilization” 表

Type	Used	Available	LOC	Clock Region	Pblock
BUFGE	11	240	0	0	0
BUFGCE_DIV	2	40	0	0	0
BUFGCTRL	1	80	0	1	0
BUFG_GT	2	120	0	0	0
MMCM	2	10	1	0	0
PLL	3	20	0	0	0

注释：“时钟区域 (Clock Region)” 约束不适用于 7 系列器件。

“Global Clock Resources” 表

“Global Clock Resources” 表仅显示每个时钟信号线的汇总信息（包括重要的约束和布局信息），如下图所示。

图 153: Report Clock Utilization - “Global Clock Resources” 表

Global Id	Source Id	Driver Type/Pin	Constraint	Site	Clock Region	Root	Clock Delay Group	Load Clock Region	Clock Loads	Non-Clock Loads	Clock Period	Clock
g6	src3	BUFGCE/O	None	BUFGCE_X1Y25	X2Y1	X1Y0		2	234	0	33.333	config_mb_i/mdm_1/U0/Us
g7	src1	BUFGCE_DIV/O	None	BUFGCE_DIV_X1Y4	X2Y1	X2Y0	cdg0	2	167	0	3.200	clk312_out
g8	src1	BUFGCE/O	None	BUFGCE_X1Y36	X2Y1	X2Y0	cdg0	1	5	1	1.600	clk625_i
g9	src4	BUFGCE/O	None	BUFGCE_X0Y25	X0Y1	X0Y1		1	17	0	3.332	default_sysclk_300_clk_p
g10	src5	BUFGCE_DIV/O	None	BUFGCE_DIV_X1Y0	X2Y0	X2Y0	cdg1	1	1	0	16.276	div_clk_buf
g11	src6	BUFG_GT/O	None	BUFG_GT_X0Y76	X3Y3	X2Y0		1	1	0	16.276	sys_clk_122
g12	src7	BUFG_GT/O	None	BUFG_GT_X0Y102	X3Y4	X2Y0		1	1	0	32.552	sys_clk_30_72
g13	src8	BUFGCE/O	None	BUFGCE_X1Y27	X2Y1	X2Y0	cdg1	1	1	0	2.043	sys_clk_491

下表中列出了 “Global Clock Resources” 表中的列。

表 15：“Global Clock Resources” 表详情

列	描述
全局 ID (Global Id)	全局时钟信号线唯一 ID
源 ID (Source Id)	生成原语（连接到时钟缓存）的时钟 ID
驱动类型/管脚 (Driver Type/ Pin)	连接到时钟信号线的原语管脚
约束 (Constraint)	已应用到时钟缓存的最高优先级的用户物理约束。优先级规则如下所示： 1. LOC 2. CLOCK_REGION* 3. PBLOCK * 不适用于 7 系列。
站点 (Site)	由用户或者由 Vivado 实现工具所设置的时钟缓存位置。
时钟区域 (Clock Region)	缓存所在的器件时钟区域。 不适用于 7 系列。
根 (Root)	时钟信号线 CLOCK_ROOT 所在的时钟区域。 不适用于 7 系列。
时钟延迟组 (Clock Delay Group)	由用户指定的时钟信号线组的名称，用于强制由 Vivado® 实现工具进行布线匹配。 不适用于 7 系列。
负载时钟区域 (Load Clock Region)	时钟信号线负载所在的时钟区域数量。
时钟负载 (Clock Loads)	时钟管脚负载的数量。
非时钟负载 (Non-Clock Loads)	非时钟管脚负载（例如，FDCE/CE 管脚）的数量。
时钟周期 (Clock Period)	在时钟信号线上传输的时序时钟周期 (ns)。如果在同一个时钟信号线上有多个时钟在传输，则报告最小的时钟周期。
时钟 (Clock)	在时钟信号线上传输的时序时钟的名称。如果在同一个时钟信号线上有多个时钟在传输，则报告“Multiple”。
驱动管脚 (Driver Pin)	时钟信号线驱动管脚的逻辑名称。
信号线 (Net)	连接到时钟驱动管脚的时钟信号段的逻辑名称。

“全局时钟源详情 (Global Clock Source Details)” 表

“Global Clock Source Details” 表显示了每个时钟生成器输出的全局时钟连接和时序时钟信息。下图显示了 MMCME3 (src0/src1) 的每个输出到时钟缓存的连接。src1 的输出 CLKOUT0 用于驱动 2 个全局时钟：g7 和 g8。

图 154：Report Clock Utilization - “Global Clock Source Details” 表

Global Clock Source Details									
Source Id	Global Id	Driver Type/Pin	Constraint	Site	Clock Region	Clock Loads	Non-Clock Loads	Source Clock Period	Source Clock
src0	g0	MMCME3_ADV/CLKOUT0	MMCME3_ADV_X0Y1	MMCME3_ADV_X0Y1	X0Y1	1	0	3.332	mmcm_clkout0
src0	g1	MMCME3_ADV/CLKOUT1	MMCME3_ADV_X0Y1	MMCME3_ADV_X0Y1	X0Y1	1	0	9.996	mmcm_clkout1
src0	g5	MMCME3_ADV/CLKOUT5	MMCME3_ADV_X0Y1	MMCME3_ADV_X0Y1	X0Y1	1	0	13.328	mmcm_clkout5
src0	g3	MMCME3_ADV/CLKOUT6	MMCME3_ADV_X0Y1	MMCME3_ADV_X0Y1	X0Y1	1	0	6.664	mmcm_clkout6
src1	g2	MMCME3_ADV/CLKOUT0	None	MMCME3_ADV_X1Y1	X2Y1	1	0	8.000	clk125_i
src1	g7, g8	MMCME3_ADV/CLKOUT2	None	MMCME3_ADV_X1Y1	X2Y1	2	0	1.600	clk625_i

下表中列出了“Global Clock Source Details”表中的列。

表 16：“Global Clock Source Details”表中包含以下列

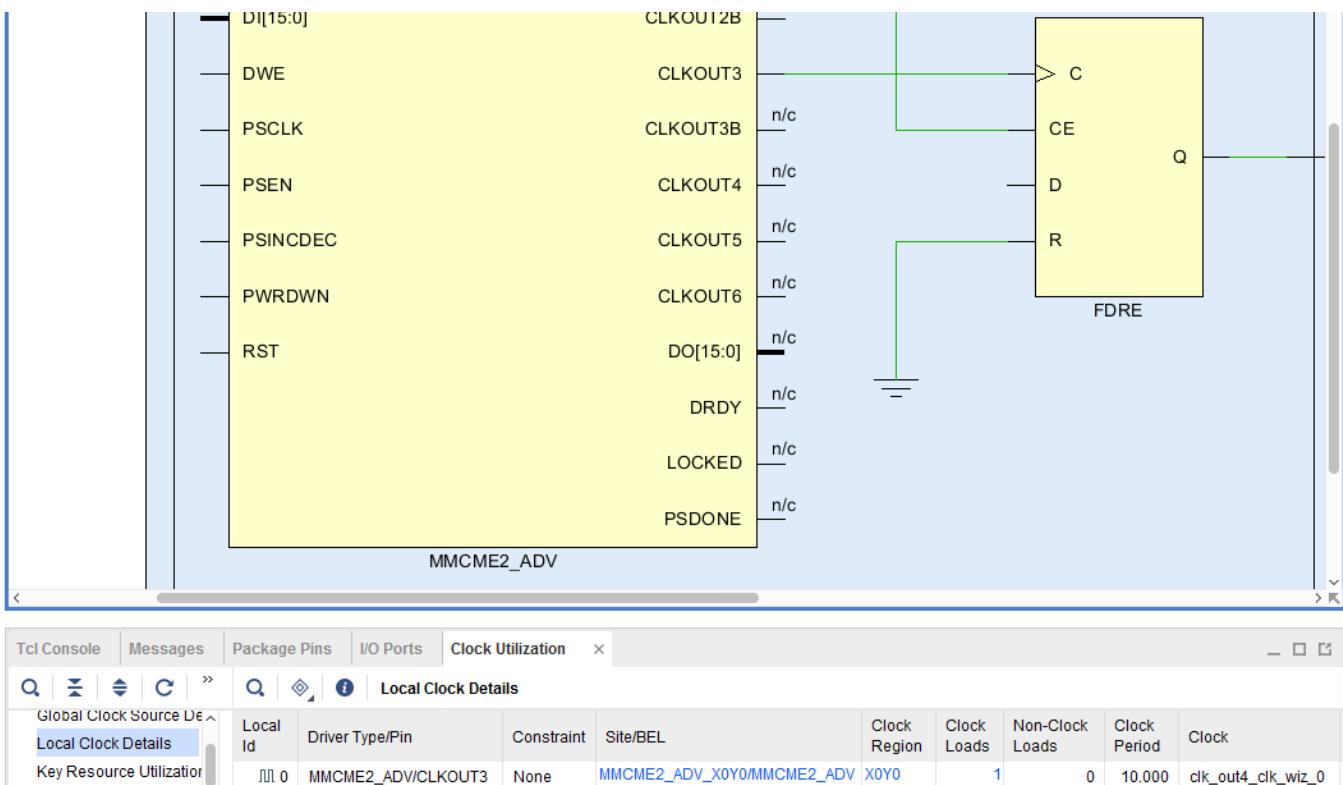
列	描述
源 ID (Source Id)	生成原语的时钟的 ID。
全局 ID (Global Id)	由全局时钟源管脚驱动的全局时钟 ID。
驱动类型/管脚 (Driver Type/Pin)	生成时钟的输出原语管脚。
约束 (Constraint)	含应用于时钟缓存的最高优先级的用户物理约束。优先级规则如下所示： 1. LOC 2. PBLOCK
站点 (Site)	由用户或 Vivado 实现工具设置的全局时钟源位置。
时钟区域 (Clock Region)	时钟源所在的器件时钟区域。
时钟负载 (Clock Loads)	已连接到全局时钟源管脚的时钟管脚负载数量。
非时钟负载 (Non-Clock Loads)	非时钟管脚负载（例如，FDCE/CE 管脚）的数量。
源时钟周期 (Source Clock Period)	由全局时钟源管脚生成的时序时钟的周期 (ns)。如果在同一个时钟信号线上有多个时钟在传输，则报告最小的时钟周期。
时钟 (Clock)	由全局时钟源管脚生成的时序时钟数量。如果在同一个时钟信号线上有多个时钟在传输，则报告“Multiple”。
驱动管脚 (Driver Pin)	全局时钟源管脚的逻辑名称。
信号线 (Net)	连接到全局时钟源管脚的时钟信号线段的逻辑名称。

“局部时钟详情 (Local Clock Details)”表

仅当设计中发现局部时钟时，才会报告“Local Clock Details”表。局部时钟表示以常规结构布线资源而不是全局时钟资源进行布线的时钟信号线。通常当时钟信号线不受时钟缓存驱动时会发生这种情况。该表所提供的信息与“Global Clock Resources”表中所提供的信息类似。

下图显示了由 7 系列 MMCM 输出驱动的局部时钟信号线，此输出直接驱动寄存器时钟管脚 (FDRE/C)。

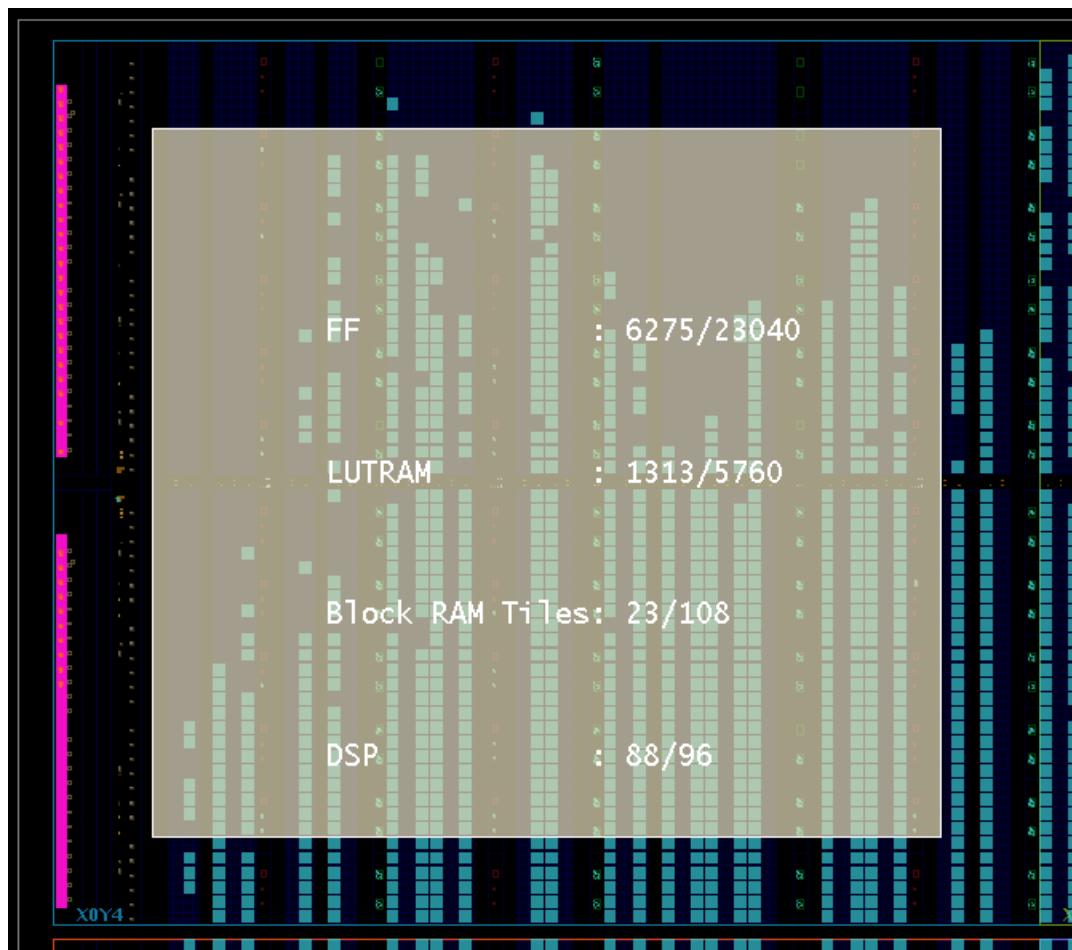
图 155：Report Clock Utilization - 局部时钟示例



“时钟区域 (Clock Regions)” 表

“Clock Regions” 部分仅适用于 UltraScale 器件系列，其中包含多个表以涵盖每个时钟区域的原语和布线资源使用情况。在 “Clock Utilization” 窗口中，“Show Metrics In Device Window” 按钮 可用于选择要在 “Device” 窗口中显示的每个时钟区域上显示的资源类型，如下图所示。

图 156: Report Clock Utilization - “Device” 窗口中的时钟区域资源使用情况指标



“Clock Regions” 表包括：

- 时钟原语 (Clock Primitives): 每个时钟区域内的每种时钟原语类型的使用情况。
- 负载原语 (Load Primitives): 每个时钟区域内非时钟时序原语的使用情况。

对于“Clock Primitives”和“Load Primitives”表，“Global Clock”列可显示水平分布层上布线的全局时钟信号线的数量，包括所报告的时钟区域内含负载和不含负载的信号线。如果时钟信号线在垂直分布层上已布线，但未分叉到所报告的时钟区域内的水平层，则不纳入计数范围。布线层上已布线的时钟信号线不纳入计数范围。

- 全局时钟汇总 (Global Clock Summary): 在对应于器件时钟区域布局规划的表中，显示全局时钟的使用情况（按时钟区域划分），如下图所示。该表仅显示在文本报告中。

图 157：Report Clock Utilization - “Global Clock Summary” 示例

6. Clock Regions : Global Clock Summary						
<hr/>						
	X0	X1	X2	X3		
+	-----+-----+-----+-----+					
	Y4	4	5	5	3	
	Y3	5	5	6	6	
	Y2	5	9	11	7	
	Y1	6	9	14	8	
	Y0	6	6	13	12	
+	-----+-----+-----+-----+					

- 布线资源使用情况 (Routing Resource Utilization): 按类型和时钟区域显示全局时钟布线使用情况。

“关键资源使用情况 (Key Resource Utilization)” 表

“Key Resource Utilization” 表仅适用于 7 系列器件，它等同于对应 UltraScale 器件的所有“时钟区域 (Clock Regions)”表的组合。“全局时钟汇总 (Global Clock Summary)”表同样只能以文本报告方式显示。

“全局时钟 (Global Clocks)” 表

“Global Clocks” 表用于报告每个全局时钟信号线的每个时钟区域中的负载类型，以及用于完成时钟信号线布线的时序时钟信息和时钟轨道 ID。在 Vivado IDE 中按“全局 ID (Global ID)”进行排序时，只需选中表中对应的行即可轻松识别并高亮显示器件中用于对每个全局时钟信号线进行布线的时钟区域。

列描述与“Clock Primitive Utilization”、“Global Clock Resources”和“Global Clock Source Details”表中的列描述相同。

对于 UltraScale 器件，对于在任一时钟区域内无需驱动任何负载即可完成布线的时钟信号线，其“Global ID”以“+”字符来标记，如下图所示。

图 158：Report Clock Utilization - 时钟区域单元布局 (Clock Region Cell Placement) 示例

Global Id	Clock Region	Track	Driver Type/Pin	Constraint	Clock Loads	Non-Clock Loads	FF	LUTRAM	RAMB	URAM	DSP	GT	MMCM	PLL	Hard IP
g11+	X2Y1	4	BUFG_GT/O	None	0	0	0	0	0	n/a	0	0	0	0	0
g12+	X2Y1	6	BUFG_GT/O	None	0	0	0	0	0	n/a	0	0	0	0	0
g13+	X2Y1	3	BUFGCE/O	None	0	0	0	0	0	n/a	0	0	0	0	0
g14+	X2Y1	0	BUFGCTRL/O	X2Y4	0	0	0	0	0	n/a	0	0	0	0	0
g15	X2Y1	20	BUFGCE/O	None	0	1	0	0	0	n/a	0	0	1	0	0
g1	X2Y0	2	BUFGCE/O	None	2433	0	2425	0	8	n/a	0	0	0	0	0
g2	X2Y0	14	BUFGCE/O	None	3669	0	3627	40	2	n/a	0	0	0	0	0

控制集报告

控制集表示时钟信号、时钟使能信号和置位/复位信号的唯一组合。每个 slice 都支持有限数量的控制集，以供位于其中的触发器组合使用。根据所使用的架构，部分控制集允许在 slice 内共享。用户应熟悉目标系列的“可配置逻辑块 (Configurable Logic Block)”架构才能理解兼容性规则。

报告主要包含以下 2 部分：

1. 控制集的绝对数量。任意给定器件中控制集的有限数量。超出建议的控制集数量可能对 QoR 造成不利影响。
2. 控制器的负载剖析。需减少控制集时，减少包含少量负载的控制集数量最有效，因为这给设计增加的逻辑量最少。

以下是“控制集报告汇总 (Control Sets Report Summary)”示例。请遵循《UltraFast 设计方法指南（适用于 Vivado Design Suite）》(UG949) 中有关建议的控制集数量的指示进行操作。

图 159：控制集汇总表

1. Summary	
<hr/>	
Status	Count
Total control sets	7520
Minimum number of control sets	5744
Addition due to synthesis replication	9
Addition due to physical synthesis replication	1767
Unused register locations in slices containing registers	11175
<hr/>	
* Control sets can be merged at opt_design using control_set_merge or merge_equivalent_drivers	
** Run report_qorSuggestions for automated merging and remapping suggestions	

通常在综合时复制的信号线很可能出现重叠，从而给布线资源施加较大的压力。通常由物理综合所复制的信号线重叠较少，计算控制集最大数量时可忽略。

当控制集计数超过建议数量时，用户应使用 BEL 计数最少的负载来最优化控制集以减少其数量。以下提供了统计图汇总报告以供概览：

图 160：控制集统计图表

2. Histogram	
<hr/>	
Fanout	Count
Total control sets	7520
>= 0 to < 4	2016
>= 4 to < 6	1720
>= 6 to < 8	663
>= 8 to < 10	849
>= 10 to < 12	279
>= 12 to < 14	168
>= 14 to < 16	218
>= 16	1607
<hr/>	
* Control sets can be remapped at either synth_design or opt_design	

如果需要更多目标信息，-hierarchical 和 -hierarchical_depth 开关将有助于突出显示特定目标层级。综合 BLOCK_SYNTH.CONTROL_SET_THRESHOLD 属性可用于在特定层级重新设置控制集目标。

控制集报告还可提供设计中使用的“触发器分布 (Flip Flop Distribution)”类型的详细信息。Vivado 无法重新设置异步复位的复位控制目标。

图 161：控制集触发器分布

4. Flip-Flop Distribution					
Clock Enable	Synchronous Set/Reset	Asynchronous Set/Reset	Total Registers	Total Slices	
No	No	No	352377	69230	
No	No	Yes	43736	11204	
No	Yes	No	15857	6332	
Yes	No	No	235485	49465	
Yes	No	Yes	72497	16790	
Yes	Yes	No	29089	12761	

要获取设计中的所有控制集的综合列表，请使用 `-verbose` 开关。它可列出每个控制集的如下信息：

- 时钟信号 (Clock Signal): 逻辑时钟信号名称
- 使能信号 (Enable Signal): 逻辑时钟使能信号名称
- 置位/复位信号 (Set/Reset Signal): 逻辑置位/复位信号名称
- slice 负载计数 (slice Load Count): 包含连接到控制集的单元的唯一 slice 数量
- BEL 负载计数 (BEL Load Count): 连接到控制集的单元数量

DRC 报告 (Report DRC)

“DRC 报告 (DRC Report)” 是由布线器生成的。布线器运行前，工具会检查常见设计问题。此报告可列出运行中使用的检查。



重要提示！ 请复查 “严重警告 (Critical Warnings)”。特定检查的严重性可能在后续流程中提升。

Report DRC 可运行常用 “设计规则 (Design Rule)” 检查以查找常见的设计问题和错误。

Elaborated Design

该工具用于检查 I/O、时钟布局、HDL 潜在编码问题和 XDC 约束相关的 DRC。RTL 网表通常不包含所有 I/O 缓存、时钟缓存和综合后设计包含的其它原语。Elaborated Design DRC 检查的错误数量少于后续 DRC 检查。

综合后的设计和实现后的设计

- 检查综合后网表相关的 DRC。
- 检查 I/O、BUFG 和其它布局。
- 对 MGT、IODELAY 和其它原语上的属性连线执行基本检查。
- 考量所有可用布局布线的前提下运行相同的 DRC。
- DRC 具有 4 种严重性级别：参考 (Info)、警告 (Warning)、严重警告 (Critical Warning) 和错误 (Error)。严重警告和错误当前不会阻止设计流程。

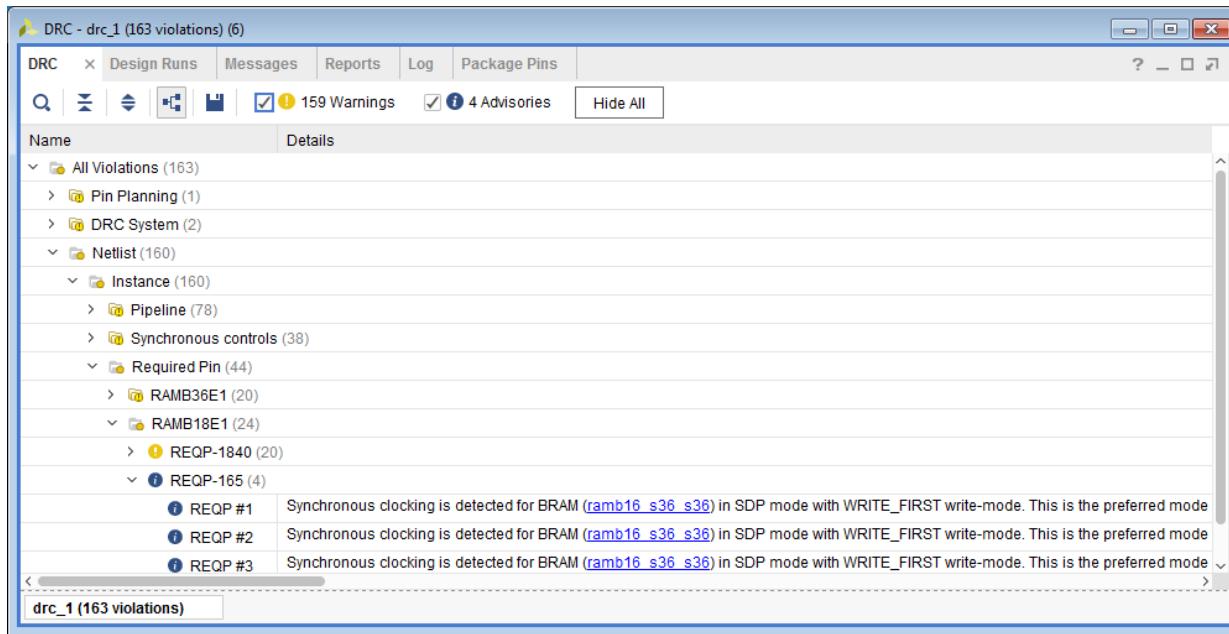
实现流程的步骤同样会运行 DRC，这些步骤可能在关键点停止运行此流程。布局器和布线器会检查导致阻止布局的问题。根据阶段，某些消息的严重性级别可能较低。这些 DRC 标记条件不会阻止 `opt_design`、`place_design` 或 `route_design` 完成，但可能导致板上出现问题。

例如，某些 DRC 会检查用户是否手动约束了所有设计端口的封装管脚位置和 I/O 标准。如果其中部分约束缺失，那么 `place_design` 和 `route_design` 会发出严重警告。但这些 DRC 在 `write_bitstream` 中则显示为 “ERROR”。工具将不会对不含这些约束的器件进行编程。

在流程初期降低严重性可便于您在判定最终管脚输出 (pinout) 前运行设计并完成实现迭代。您必须运行比特流生成以实现完整的 DRC 验收。

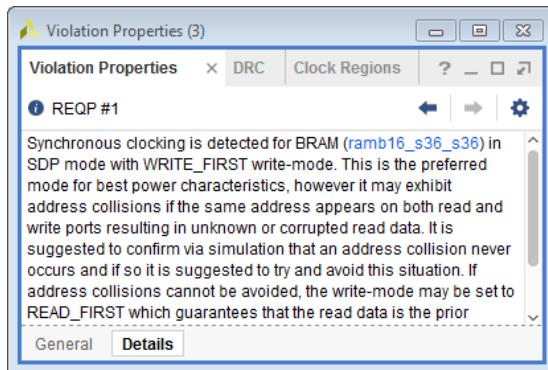
下图显示了 Report DRC 的 Vivado IDE 图形用户界面。

图 162：DRC 报告



单击任一 DRC 可打开属性并显示详细消息。请查看“属性 (Properties)”窗口以查看详细信息。大部分消息都包含对应于 DRC 中应用的信号线、单元和端口的超链接。

图 163：“Violation Properties”窗口



DRC 报告为静态报告。您必须重新运行 Report DRC 以使该报告反映设计变更。工具判定完成某些设计操作（例如，删除对象和移动对象）后，链接已过时，并使这些链接失效。

从超链接中选择对象会选中该对象，但不会刷新“Properties”窗口。要显示该对象的属性，请将其取消选中，然后重新选中。

要在 Tcl 中创建 DRC 报告，请运行 `report_drc` 命令。

要将结果写入文件，请运行 `report_drc -file myDRCs.txt` 命令。



提示：如需了解有关 report_drc 的更多信息，请运行 report_drc -help。

布线状态报告 (Report Route Status)

“布线状态报告 (Route Status Report)” 是在实现流程期间生成的，可通过 report_route_status Tcl 命令来使用。

“Route Status Report” 可显示设计中的信号线细分信息，如下所示：

- 设计中的逻辑信号线总数
 - 无需布线资源的信号线数量
 - 不使用拼块外部的布线资源的信号线数量示例包括 CLB、块 RAM 或 I/O 焊盘内部的信号线。
 - 无负载的信号线（如果存在）数量
 - 需布线资源的可布线信号线数量
 - 未布线的信号线（如果存在）数量
 - 已完全布线的信号线数量
 - 含布线错误的信号线数量
 - 含部分未布线管脚的信号线（如果存在）数量
 - 含天线/电源岛的信号线（如果存在）数量
 - 含资源冲突的信号线（如果存在）数量

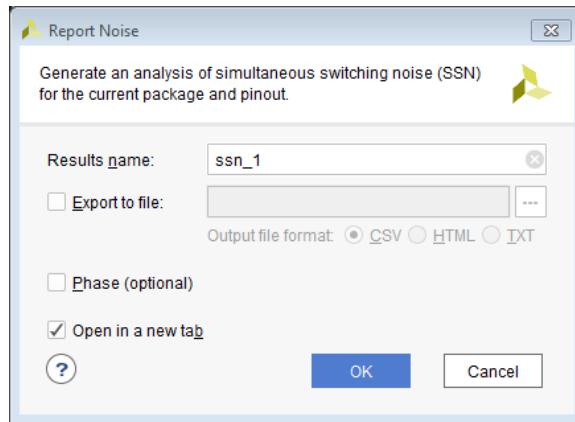
以下是已完全布线的设计的 “Report Route Status” 示例：

```
Design Route Status
                      :      # nets :
-----
# of logical nets.....:      6137 :
    # of nets not needing routing.....:      993 :
        # of internally routed nets.....:      993 :
    # of routable nets.....:      5144 :
        # of fully routed nets.....:      5144 :
    # of nets with routing errors.....:      0 :
-----
```

噪声报告 (Report Noise)

“Report Noise” 命令用于为赛灵思 7 系列 FPGA 器件执行同步开关噪声 (SSN) 计算。默认情况下，在 Vivado IDE 的 “Noise” 窗口区域中的新选项卡中打开 “Noise” 报告。您可将结果导出至 CSV 文件或 HTML 文件。

图 164：运行 SSN 分析



“Noise” 报告包含 4 个部分：

- “噪声报告 (Noise Report)” 的 “汇总 (Summary)” 部分
- “噪声报告 (Noise Report)” 的 “消息 (Messages)” 部分
- “噪声报告 (Noise Report)” 的 “I/O Bank 详情 (I/O Bank Details)” 部分
- “噪声报告 (Noise Report)” 的 “链接 (Links)” 部分

“噪声报告 (Noise Report)” 的 “汇总 (Summary)” 部分

“Noise Report”的“Summary”部分包括：

- 报告运行时间
- 已分析的适用端口数量和百分比
- 状态（包括是否成功）
- “Critical Warnings”、“Warnings” 和 “Info” 消息数量

“噪声报告 (Noise Report)” 的 “消息 (Messages)” 部分

“Noise Report”的“Messages”部分包含报告期间生成的消息的详细列表。

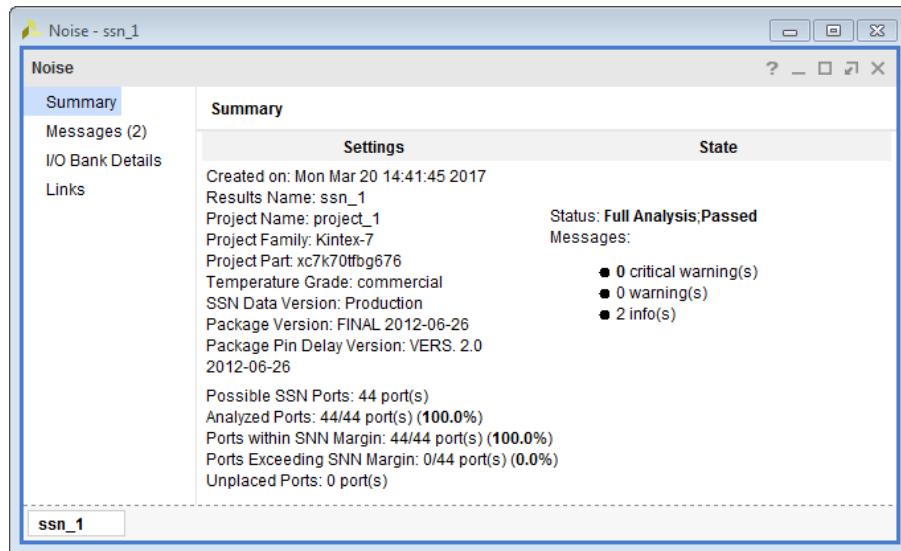
“噪声报告 (Noise Report)” 的 “I/O Bank 详情 (I/O Bank Details)” 部分

“Noise Report”的“I/O Bank Details”部分包含管脚 (Pins)、标准 (Standards) 和剩余裕度 (Remaining Margin) 的列表。

“噪声报告 (Noise Report)” 的 “链接 (Links)” 部分

“Noise Report”的“Links”部分包含指向位于 china.xilinx.com/support 的文档的链接。

图 165：噪声报告



要创建 HTML 版本的报告，请选择相应的选项或者运行以下 Tcl 命令：

```
report_ssn -format html -file myImplementedDesignSSN.html
```

功耗报告 (Report Power)

在布线后会生成“功耗报告 (Power Report)”以基于当前器件工作条件和设计的开关率报告功耗详情。功耗分析要求网表已完成综合或设计已完成布局布线。

- 使用 `set_operating_conditions` 命令来设置工作条件。
- 使用 `set_switching_activity` 命令来定义开关活动。

当“综合后设计”或“实现后设计”打开时，即可使用“Report Power”命令。

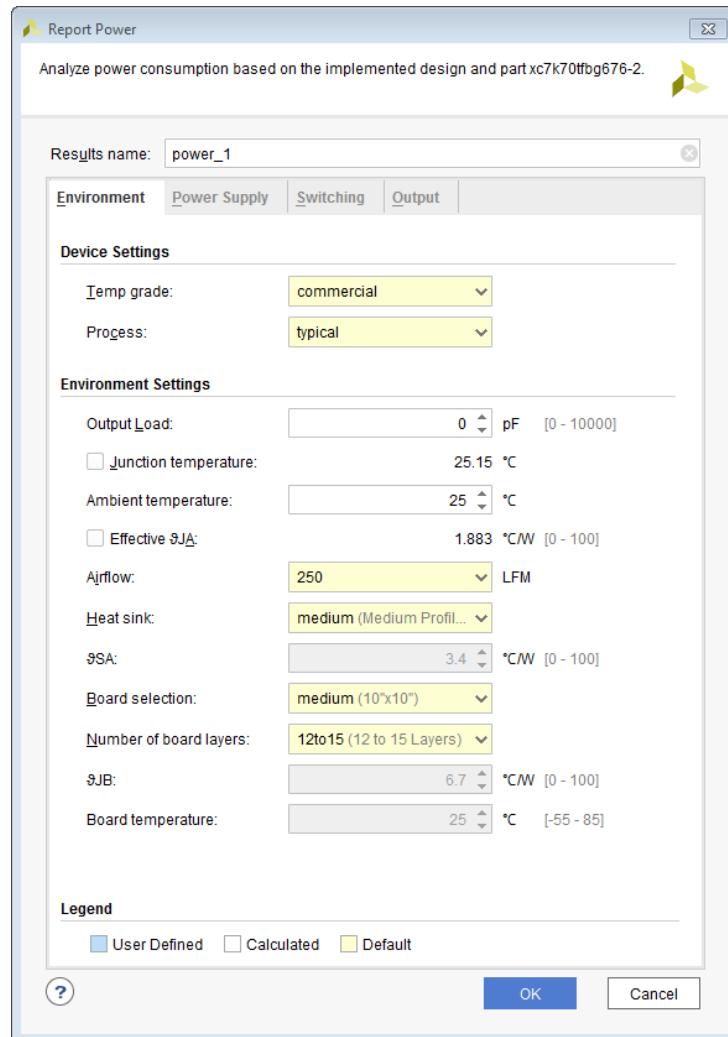
“Power Report”可用于基于设计输入来估算功耗，包括：

- 热量统计数据，例如，结温和环境温度值。

注释：您可使用 `set_operating_condition` 命令的 `-junction_temp` 选项来设置结温。如果不指定温度，软件会基于您的设计输入来为您计算温度。

- 有关板上选择的数据，包括单板层数和板上温度。
- 有关设计所使用的气流和散热器资料的选择数据。
- 报告来自不同电源的 FPGA 电流要求。
- 支持执行详细的配电分析，提供省电策略相关指南，并减少动态功耗、散热功耗或片外功耗。
- 可使用仿真活动文件来提升功耗估算的准确性。

图166：“Report Power”对话框

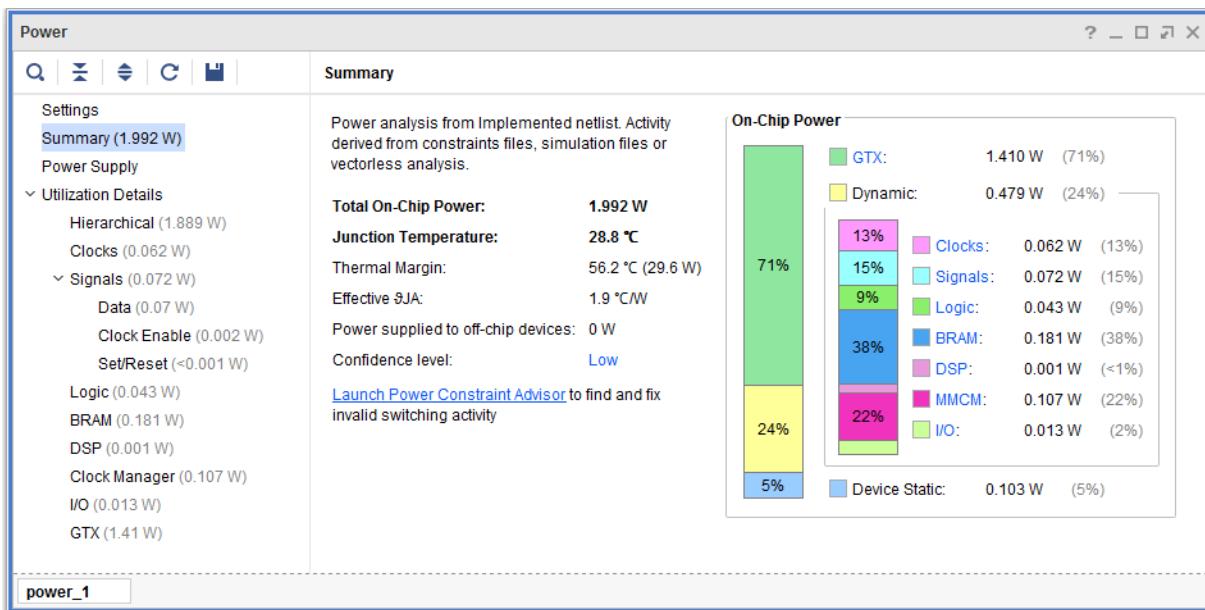


分析功耗报告

使用“功耗报告 (Report Power)”对话框（请参阅下图）可根据以下条件分析功耗：

- 设置
- 总功耗
- 层级
- 电压轨
- 块类型

图 167：功耗报告



如需了解有关功耗报告和分析结果的更多信息，请参阅《Vivado Design Suite 用户指南：功耗分析与最优化》([UG907](#))。

默认情况下，实现流程中完成布线后会生成功耗报告的文本版本。

非工程流程中的功耗报告

在非工程流程中，完成 `link_design` 或 `synth_design` 后即可使用 `report_power`。生成的报告使用可用布局布线来提供更准确的功耗数值。要从 Tcl 控制台或脚本生成此报告，请运行 `report_power`。

RAM 使用情况报告 (Report RAM Utilization)

“RAM 使用情况 (Ram Utilization)” 报告可帮助您分析专用 RAM 块（例如，URAM 和块 RAM）的使用情况以及分布式 RAM 原语。默认情况下，此报告覆盖整个设计，但可通过 `-cell` 开关将其限制于特定层级。此报告可在综合后以及任意实现步骤后生成，但只能通过 Tcl 命令行查看。

“RAM Utilization” 报告对于 Vivado 2019.2 或更高版本所推断的内存最有效。它支持用户将 “RTL 内存阵列 (RTL Memory Array)” 与 FPGA 中的实际物理实现进行比较。

此报告可显示如下内容：

- 每个内存原语的使用情况
- 阵列大小和尺寸（仅限推断）
- 所推断的内存类型（仅限推断）
- 所推断的内存原语的使用情况（仅限推断）
- 所需的内存性能
- 内存的可选流水线使用情况（如适用）
- 始于和止于内存的最差情况逻辑路径

- 功耗效率数据，例如，级联和使能率

如何运行报告

以下语法将以默认模式运行报告，并将内容发送至 `ram_util.rpt` 文件。

```
report_utilization -file ram_util.rpt
```

由于路径信息可能占用大量运行时间，您可选择使用 `-include_path_info` 开关将以下内容添加到报告中：

```
report_ram_utilization -include_path_info
```

为了报告所有内存（包括非推断原语），应使用 `-detail` 开关：

```
report_ram_utilization -detail
```

报告布局

综述

下图所示综述提供了设计中所使用的 RAM 原语概述以及已推断（或已使用 XPM 例化）的原语数量。

图 168：综述

Memory Type	Total	Used	Available	Util %	Inferred %
URAM	181	960	18.85	100.00	
URAM288	181			100.00	
BlockRAM	1047	2160	48.47	100.00	
RAMB36E2	590			100.00	
RAMB18E2	914			100.00	
LUTMs as Distributed RAM	51052	591840	8.63	99.94	
LUTMs as RAM64X1S	197			100.00	
LUTMs as RAM64X1D	88			100.00	
LUTMs as RAM64M8	23992			100.00	
LUTMs as RAM32X1S	4997			100.00	
LUTMs as RAM32X1D	1046			100.00	
LUTMs as RAM32M16	20640			99.84	
LUTMs as RAM256X1D	48			100.00	
LUTMs as RAM128X1D	44			100.00	

“综述”按原语类型进行细分，并且还显示了各原语的总体利用率。分布式 RAM 也以已用内存 LUT (LUTM) 形式来报告，而不是以分布式 RAM 原语总数的形式来报告。这样有助于识别设计中是否包含大量低效 DRAM 原语（通常旧设计或者含大量例化的设计可能出现此问题）。例如，RAM32X1D 上的单比特成本可能高达每比特 2 个 LUT（或者如果采用组合 LUT，则可能为每比特 1 个 LUT），但如果使用所有比特，则 RAM32M16 可能约为每比特 $\frac{1}{2}$ LUT。

内存描述

内存描述表仅适用于推断的 RAM。它详列了推断内存时综合工具所看到的内存。其中详列了总阵列大小、维度、内存类型和时钟周期。如下图所示：

图 169：内存描述表

Memory Name	Array Size	Memory Type	Port A Dimension	Port A Requirement (ns)	Port B Dimension	Port B Requirement (ns)
MSH	2359294	Single CLOCK SP	262144x9	2.00		

注释：如果推断 2 个以上端口，该报告将仅详列 2 个端口。

内存利用率

内存利用率表详述了阵列映射到硬件中的原语的方式。使用效率较低时，可识别原因来自于宽度或深度。

图 170：内存利用率表

3. Memory Utilization												
Memory Name	Port A Dimension	Port B Dimension	Primitive	Available Bits	Used Bits	Bit Utilizaztion %	A Depth	A Depth Avail	A Depth Util%	A Width	A Width	
MEM	262144x9											
MEM_reg_uram_0	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		
MEM_reg_uram_1	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		
MEM_reg_uram_2	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		
MEM_reg_uram_3	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		
MEM_reg_uram_4	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		
MEM_reg_uram_5	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		
MEM_reg_uram_6	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		
MEM_reg_uram_7	32768x9		URAM288E5	294912	294912	100.00	32768	32768	100.00	9		

内存性能

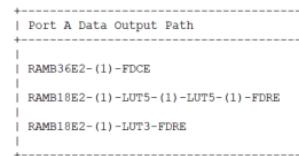
内存性能表可提供有关寄存和级联设置（如适用）的详细信息。

图 171：内存性能表

Memory Name	Primitive	Port A Dimension	Port A Requirement (ns)	Port B Dimension	Port B Requirement (ns)	A Cascade	IREG_A	OREG_A	DOA_REG	B Cascade	IREG
MEM		262144x9									
MEM_reg_uram_0	URAM288E5	32768x9	2.00								
MEM_reg_uram_1	URAM288E5	32768x9	2.00								
MEM_reg_uram_2	URAM288E5	32768x9	2.00								
MEM_reg_uram_3	URAM288E5	32768x9	2.00								
MEM_reg_uram_4	URAM288E5	32768x9	2.00								
MEM_reg_uram_5	URAM288E5	32768x9	2.00								
MEM_reg_uram_6	URAM288E5	32768x9	2.00								
MEM_reg_uram_7	URAM288E5	32768x9	2.00								

如果已指定 `-include_path_info` 开关，将显示额外路径信息，如下图所示。

图 172：-include_path_info



报告的这部分可显示 REFNAME 后接穿过 Port A Data Output Path（数据和奇偶校验位）的最差情况路径的扇出（包含在括号内）。针对内存中每个总线会重复显示此信息。如果未列出任何扇出，则可假定此形状将封装在同一个 slice 内，且扇出为 1。

内存功耗

内存功耗表详列了 Vivado 是否已对内存进行最优化以降低功耗，或者使能信号是通过 POWER (Vcc) 还是通过信号来驱动的。其中还重复列出了级联信息和功耗相关的内存属性。

图 173：内存功耗表

Memory Name	Primitive	Power Gated	Port A Dimension	Port A Enable	Port A WRITE_MODE	A Cascade
MEM			262144x9			
MEM_reg_uram_0	URAM288E5	ENARDEN=NEW	32768x9		POWER	
MEM_reg_uram_1	URAM288E5	ENARDEN=NEW	32768x9		POWER	
MEM_reg_uram_2	URAM288E5	ENARDEN=NEW	32768x9		POWER	
MEM_reg_uram_3	URAM288E5	ENARDEN=NEW	32768x9		POWER	
MEM_reg_uram_4	URAM288E5	ENARDEN=NEW	32768x9		POWER	
MEM_reg_uram_5	URAM288E5	ENARDEN=NEW	32768x9		POWER	
MEM_reg_uram_6	URAM288E5	ENARDEN=NEW	32768x9		POWER	
MEM_reg_uram_7	URAM288E5	ENARDEN=NEW	32768x9		POWER	

非推断原语表

针对例化原语同样可显示通过网表收集的信息，但此类信息更改难度更大。要查看此信息，请运行 `-detail` 开关。在该表中，不存在关联的 RTL 阵列。

第 5 章

执行时序分析

时序分析简介

赛灵思 Vivado® 集成设计环境 (IDE) 提供了多项报告命令，用于验证设计是否满足所有时序约束，以及是否准备好加载到应用开发板上。“Report Timing Summary”为时序验收报告，等同于 ISE® Design Suite 中的 TRCE。“Report Timing Summary”可提供所有时序检查的完整概况，并显示充足的信息以支持您开始对任何时序问题进行分析和调试。如需了解更多信息，请参阅第 1 章：IDE 中的逻辑分析。

您可在窗口中生成此报告、将其写入文件或者打印到日志文件中。当“Report Timing Summary”显示您的设计不满足时序或者缺少某些约束时，您可浏览汇总报告各部分中提供的详细信息并运行更具体的分析。其它时序报告可提供有关特定状况的更多详细信息，并且可通过使用筛选工具和限定范围功能来将分析限定于某些逻辑。

在将时序约束添加到设计前，您必须了解时序分析基础知识以及与之关联的术语。本章探讨了赛灵思 Vivado 集成设计环境 (IDE) 时序引擎所使用的部分关键概念。

术语

- 发送沿 (launch edge) 表示发送数据的源时钟的处于活动状态的时钟沿。
- 捕获沿 (capture edge) 表示捕获数据的目标时钟的处于活动状态的时钟沿。
- 源时钟 (source clock) 也称为发送时钟 (launch clock)。
- 目标时钟 (destination clock) 也称为捕获时钟 (capture clock)。
- 建立要求 (setup requirement) 表示定义最严格的建立约束的发送沿与捕获沿之间的关系。
- 建立关系 (setup relationship) 表示经时序分析工具验证的建立时间检查。
- 保持要求 (hold requirement) 表示定义最严格的保持约束的发送沿与捕获沿之间的关系。
- 保持关系 (hold relationship) 表示经时序分析工具验证的保持时间检查。

时序路径

时序路径是由设计实例之间的连接定义的。在数字化设计中，时序路径由一对时序元件组成，这对时序元件受相同时钟或 2 个不同时钟控制。

常见时序路径

任意设计中最常见的路径为：

- 从输入端口到内部时序单元的路径
- 从时序单元到时序单元的内部路径

- 从内部时序单元到输出端口的路径
- 从输入端口到输出端口的路径

从输入端口到内部时序单元的路径

在从输入端口到时序单元的路径中，数据：

- 在器件外部由开发板上的时钟发送。
- 经延迟后到达器件端口，此延迟称为输入延迟（Synopsys 设计约束 (SDC) 定义）。
- 通过器件内部逻辑传输后到达由目标时钟进行时钟设置的时序单元。

从时序单元到时序单元的内部路径

在从时序单元到时序单元的内部路径中，数据：

- 在器件内部由时序单元发送，该时序单元的时钟由源时钟进行设置。
- 通过部分内部逻辑传输后到达由目标时钟进行时钟设置的时序单元。

从内部时序单元到输出端口的路径

在从内部时序单元到输出端口的路径中，数据：

- 在器件内部由时序单元发送，该时序单元的时钟由源时钟进行设置。
- 传输穿过部分内部逻辑，然后到达输出端口。
- 经过称为输出延迟（SDC 定义）的附加延迟后，由板上时钟捕获。

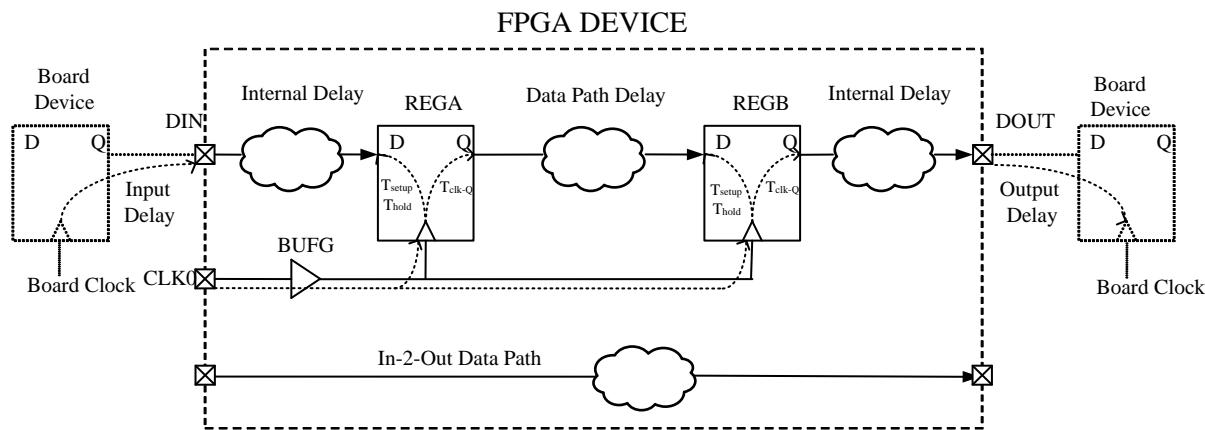
从输入端口到输出端口的路径

在从输入端口到输出端口的路径中，数据无需锁存即可遍历整个器件。此类路径通常也称为输入到输出路径 (in-to-out path)。输入和输出延迟参考时钟可能是虚拟时钟，也可能是设计时钟。

时序路径示例

下图显示了以上描述的路径。在此示例中，设计时钟 CLK0 可用作为 DIN 和 DOUT 延迟约束的开发板时钟。

图 174：时序路径示例



“时序路径 (Timing Path)”部分

每条时序路径均由 3 个部分组成：

- 源时钟路径
- 数据路径
- 目标时钟路径

源时钟路径

源时钟路径是源时钟从源点（通常为输入端口）到发送时序单元的时钟管脚的路径。对于始于输入端口的时序路径，不存在源时钟路径。

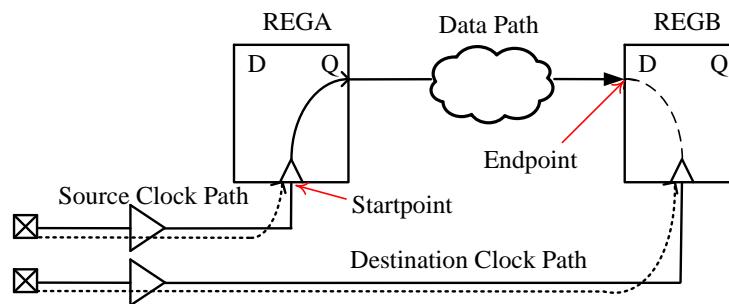
数据路径

数据路径表示在其中传输数据的时序路径（介于路径起点与路径端点之间）。适用如下定义：(1) 路径起点为时序单元时钟管脚或数据输入端口；(2) 路径端点为时序单元数据输入管脚或数据输出端口。

目标时钟路径

目标时钟路径表示目标时钟从源点（通常为输入端口）到捕获时序单元的时钟管脚的路径。对于终止于输出端口的时序路径，不存在目标时钟路径。[目标时钟路径](#)显示了典型时钟路径所包含的 3 个部分。

图 175：典型时序路径



发送沿和捕获沿

在时序单元或端口之间进行传输时，数据：

- 从源时钟的其中一个时钟沿（称为发送沿）发送。
- 由目标时钟的其中一个时钟沿（称为捕获沿）捕获。

在典型的时序路径中，数据在单一时钟周期内的 2 个时序单元之间进行传输。在此情况下：(1) 发送沿发生于 0ns；而 (2) 捕获沿发生于 1 个周期后。

以下章节解释了发送沿和捕获沿如何定义用于时序分析的建立和保持时间关系。

了解时序分析的基础知识

最大和最小延迟分析

时序分析属静态验证，旨在验证在硬件上加载并运行设计后，其时序行为的可预测性。它会将各种制造和环境变化因素组合到延迟模型（按时序极值和极值变化分组）中，并对这些变化加以考量。针对所有建议的角点分析时序即可，针对每个角点，按最消极的条件执行所有检查。例如，以赛灵思 FPGA 为目标的设计必须通过以下 4 项分析：

- 慢速角点 (Slow Corner) 中的最大延迟分析
- 慢速角点 (Slow Corner) 中的最小延迟分析
- 快速角点 (Fast Corner) 中的最大延迟分析
- 快速角点 (Fast Corner) 中的最小延迟分析

根据执行的检查，将使用展现出最消极情况的延迟。因此下列检查与延迟类型始终关联：

- 含建立时间和恢复时间检查的最大延迟
- 保持和移除检查的最小延迟

含建立时间和恢复时间检查的最大延迟

- 针对源时钟路径和数据/复位路径累积延迟，使用给定角点 (corner) 的最差情况延迟（最慢延迟）。
- 针对目标时钟路径累积延迟同样使用该角点 (corner) 的最佳情况延迟（最快延迟）。

保持和移除检查的最小延迟

- 针对源时钟路径和数据/复位路径累积延迟，使用给定角点 (corner) 的最佳情况延迟（最快延迟）。
- 针对目标时钟路径累积延迟同样使用该角点 (corner) 的最差情况延迟（最慢延迟）。

映射到多个角点时，可使用以下检查：

- 建立/恢复（最大延迟分析）
- 保持/移除（最小延迟分析）

建立/恢复（最大延迟分析）

- 源时钟 (Slow_max)，数据路径 (Slow_max)，目标时钟 (Slow_min)
- 源时钟 (Fast_max)，数据路径 (Fast_max)，目标时钟 (Fast_min)

保持/移除（最小延迟分析）

- 源时钟 (Slow_min)，数据路径 (Slow_min)，目标时钟 (Slow_max)
- 源时钟 (Fast_min)，数据路径 (Fast_min)，目标时钟 (Fast_max)

在同一路径上从不混用来自不同角点 (corner) 的延迟进行裕量计算。

大多数情况下，建立或恢复违例发生时存在 Slow 角点延迟，保持或移除违例发生时存在 Fast 角点延迟。但由于偶有例外（尤其是对于 I/O 时序），赛灵思建议您在 2 个角点上都执行 2 项分析。

建立/恢复关系

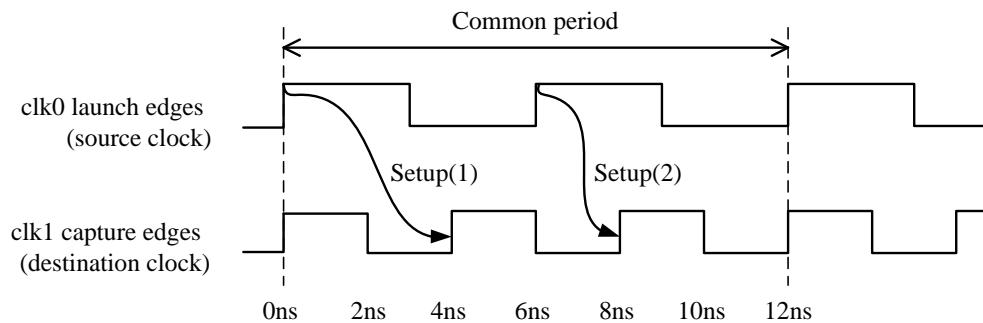
仅对 2 个时钟之间最消极的建立关系执行建立时间检查。默认情况下，此项检查对应于发送沿与捕获沿之间最小正增量。以 2 个触发器之间的路径为例，这 2 个触发器易受其各自时钟的上升沿影响。此路径的发送沿和捕获沿均为时钟上升沿。

其时钟具体定义如下：

- clk0 的周期为 6 ns，首个上升沿位于 0 ns，下降沿位于 3 ns。
- clk1 的周期为 4 ns，首个上升沿位于 0 ns，下降沿位于 2 ns。

如下图所示，存在 2 个唯一的建立关系：Setup(1) 和 Setup(2)。

图 176：建立关系



X13434

从 `clk0` 到 `clk1` 的最小正增量为 2 ns，对应于 `Setup(2)`。“公共周期 (Common Period)” 为 12 ns，对应于 2 个时钟的 2 个同步对齐之间的时间。



提示：这些关系是在考量理想时钟波形时确立的，即在将插入延迟从时钟根应用到触发器时钟管脚之前。



重要提示！如果在 2 个时钟的 1000 个周期内找不到公共周期，那么这 1000 个周期的最差建立关系将用于时序分析。对于此类情况，这 2 个时钟即称为不可扩展的时钟，或者不含公共周期的时钟。分析很可能不对应于最消极的场景。您必须审查这些时钟之间的路径，以评估其有效性，判定是否可将其改为作为异步路径来处理。

当路径要求已知后，即可引入时钟不确定性和建立时间以计算裕量。典型裕量公式为：

数据必需时间（建立） (Data Required Time (setup))	=	捕获沿时间 (capture edge time) + 目标时钟路径延迟 (destination clock path delay) - 时钟不确定性 (clock uncertainty) - 建立时间 (setup time)
数据到达时间（建立） (Data Arrival Time (setup))	=	发送沿时间 (launch edge time) + 源时钟路径延迟 (source clock path delay) + 数据路径延迟 (datapath delay)
裕量（建立） (Slack (setup))	=	数据必需时间 (Data Required Time) - 数据到达时间 (Data Arrival Time)

如上述公式所示，当数据到达时间早于必需时间时，建立裕量为正值。

恢复检查类似于建立检查，但它适用于异步管脚（如预置或清除）。关系建立方式与建立相同，裕量公式也相同（只是使用恢复时间取代建立时间）。

保持/移除关系

保持时间检查（也称为保持关系）直接连接到建立关系。虽然建立时间分析可确保在最消极的场景中仍可安全捕获数据，但保持关系可确保：

- 由建立发送沿所发送的数据不会被位于建立捕获沿之前的活动沿（下图中分别对应于建立沿 S1 和 S2 的 H1a 和 H2a）所捕获。

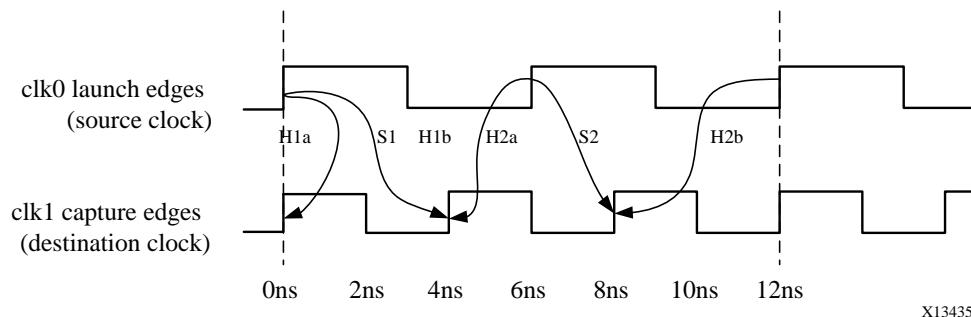
- 由位于建立发送沿之后的下一个活动源时钟沿发送的数据不会被建立捕获沿（下图中分别对应于建立沿 S1 和 S2 的 H2a 和 H2b）所捕获。

保持分析期间，时序引擎仅报告任意 2 个时钟之间最消极的保持关系。最消极的保持关系并非始终与最差建立关系关联。时序引擎必须复查所有可能的建立关系及其对应的保持关系才能识别最消极的保持关系。

仍以建立关系示例中的路径为例。存在 2 项唯一的建立关系。

下图显示了每项建立关系的 2 项保持关系。

图 177：保持关系（按建立关系）



X13435

最高保持要求为 0 ns，对应于源时钟和目标时钟的首个上升沿。

当路径要求已知后，即可引入路径延迟、时钟不确定性和保持时间以计算裕量。典型裕量公式为：

数据必需时间（保持）(Data Required Time (hold)) = 捕获沿时间 (capture edge time) + 目标时钟路径延迟 (destination clock path delay) + 时钟不确定性 (clock uncertainty) + 保持时间 (hold time)

数据到达时间（保持）(Data Arrival Time (hold)) = 发送沿时间 (launch edge time) + 源时钟路径延迟 (source clock path delay) + 数据路径延迟 (datapath delay)

裕量（保持）(Slack (hold)) = 数据到达时间 (Data Arrival Time) - 数据必需时间 (Data Required Time)

如上述公式所示，当新数据到达时间晚于必需时间时，保持裕量为正值。

移除检查类似于保持检查，但它适用于异步管脚（如预置或清除）。关系建立方式与保持相同，裕量公式也相同（只是使用移除时间取代保持时间）。

路径要求

路径要求表示时序路径的捕获沿与发送沿之间的差异。

仍以上一节中的路径和时钟为例，存在如下路径要求：

```
Setup Path Requirement (S1) = 1*T(clk1) - 0*T(clk0) = 4ns
Setup Path Requirement (S2) = 2*T(clk1) - 1*T(clk0) = 2ns
```

对应的保持时间关系为：

- 对应于建立时间 S1

```
Hold Path Requirement (H1a) = (1-1)*T(clk1) - 0*T(clk0) = 0ns
Hold Path Requirement (H1b) = 1*T(clk1) - (0+1)*T(clk0) = -2ns
```

- 对应于建立时间 S2

```
Hold Path Requirement (H2a) = (2-1)*T(clk1) - 1*T(clk0) = -2ns
Hold Path Requirement (H2b) = 2*T(clk1) - (1+1)*T(clk0) = -4ns
```

执行的时序分析仅采用 2 项最消极的要求。在上例中，这 2 项要求分别为：

- 建立时间要求 S2
- 保持时间要求 H1a

时钟相移

时钟相移对应于因时钟路径内的特殊硬件所导致的参考时钟相关的延迟时钟波形。在赛灵思 FPGA 中，时钟相移通常是由 MMCM 或 PLL 原语引入的，前提是这些原语的输出时钟属性 CLKOUT*_PHASE 为非零值。

MMCM/PLL 相移模式

时序分析期间，可通过设置 MMCM/PLL PHASESHIFT_MODE 属性以两种不同方式对时钟相移进行建模，如下表中所述。

表 17：MMCM/PLL PHASESHIFT_MODE 属性

PHASESHIFT_MODE 属性	相移建模	注
WAVEFORM	时钟波形修改	通常，set_multicycle_path -setup 约束用于在往来相移时钟的时钟域交汇路径上调整时序路径要求。
LATENCY	MMCM/PLL 插入延迟	无需额外多周期路径约束。

各赛灵思 FPGA 系列所采用的默认 MMCM/PLL 时钟相移模式不尽相同。但用户可基于 PLL/MMCM 覆盖默认模式。

表 18：默认 MMCM/PLL 时钟相移处理方式

技术	默认 MMCM/PLL 时钟相移处理方式
7 系列	时钟波形修改 (WAVEFORM)
UltraScale	时钟波形修改 (WAVEFORM)
UltraScale+	MMCM/PLL 插入延迟 (LATENCY)



重要提示！ MMCM/PLL PHASESHIFT_MODE 属性不影响器件配置。



重要提示！ 在任意 CLKOUTx 管脚上定义管脚相移并有多个时钟到达 MMCM/PLL 的输入管脚时，PHASESHIFT_MODE=LATENCY 模式无效，并触发 Warning Timing 38-437。在此情况下，MMCM/PLL 应配置为使用 PHASESHIFT_MODE=WAVEFORM 模式。

在两个时钟之间引入偏差以满足时序要求时，PHASESHIFT_MODE=LATENCY 十分便于使用。将时钟相移设置为负值、空值或正值时，调整时序路径要求无需额外的多路径约束。

对于从 7 系列或 UltraScale 移植到 UltraScale+ 的旧设计，如果在 MMCM/PLL 上未设置 PHASESHIFT_MODE 属性，将应用默认行为，并且 MMCM/PLL 时钟相移建模为延迟时延而不是时钟沿相移。在此情况下，需审查旧设计中指定的适用于时钟相移的所有多周期路径约束，并且通常需移除这些约束。通过运行方法检查 (report_methodology) 即可轻松识别这些约束。TIMING-31 用于标记时钟间的多周期路径，其中一个时钟为相移时钟，由 MMCM/PLL 生成 (PHASESHIFT_MODE 设置为 LATENCY)。

Clocking 向导和 High Speed SelectIO 向导都提供了在每个 MMCM/PLL 上强制执行时钟相移建模的选项。PHASESHIFT_MODE 属性自动保存在 IP XDC 内。

时序报告中的相移

正相移将源时钟沿向前移动，导致时钟沿延迟。负相移将源时钟沿向后移动。修改时钟波形导致静态时序分析可能对源时钟和捕获时钟使用不同的时钟沿。

在以下示例中，时钟 clkout0 (周期为 10 ns) 由 MMCM 自动衍生。

- 无相移

```
vivado% set_property CLKOUT0_PHASE 0.000 [get_cells qpll/plle2_adv_inst]
vivado% report_timing
...
(clock clkout0 rise edge) 0.000 0.000 r
...
MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
    -5.411  5.903 r mmcm_inst/mmcinst/mmcme2_adv_inst/
CLKOUT0
...
```

源时钟沿为 0.0 ns。

- 正相移为 12.0 且 PHASESHIFT_MODE=WAVEFORM

```
vivado% set_property CLKOUT0_PHASE 12.000 [get_cells qpll/plle2_adv_inst]
vivado% report_timing
...
(clock clkout0 rise edge) 0.333 0.333 r
...
MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
    -5.411  5.903 r mmcm_inst/mmcinst/mmcme2_adv_inst/CLKOUT0
...
```

源时钟沿发生 0.333 ns (10 ns / 360 * 12.0) 的延迟。

- 正相移为 12.0 且 PHASESHIFT_MODE=LATENCY

```
vivado% set_property CLKOUT0_PHASE 12.000 [get_cells qpll/plle2_adv_inst]
vivado% report_timing
...
(clock clkout0 rise edge) 0.000 0.000 r
...
MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
    -5.078  6.236 r mmcm_inst/mmcinst/mmcme2_adv_inst/
CLKOUT0
...
```

MMCM 插入延迟增加 0.333 ns (10 ns / 360 * 12.0)。源时钟沿为 0.0 ns。

- 负相移为 -15.0 且 PHASESHIFT_MODE=WAVEFORM

```
vivado% set_property CLKOUT0_PHASE -15.000 [get_cells qpll/plle2_adv_inst]
vivado% report_timing
...
(clock clkout0 rise edge) -0.417 -0.417 r
...
MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
-5.411 5.903 r mmcm_inst/mmcinst(CLKOUT0
...
```

源时钟沿向后移动 -0.417 ns (10 ns / 360 * -15.0)。

- 负相移为 -15.0 且 PHASESHIFT_MODE=LATENCY

```
vivado% set_property CLKOUT0_PHASE -15.000 [get_cells qpll/plle2_adv_inst]
vivado% report_timing
...
(clock clkout0 rise edge) 0.000 0.000 r
...
MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
-5.828 5.486 r mmcm_inst/mmcinst(CLKOUT0
...
```

MMCM 插入延迟减少 0.417 ns (10 ns / 360 * -15.0)。源时钟沿为 0.0 ns。

时钟报告中的相移

时钟相移信息在“时钟报告”（report_clocks 命令）中提供。当 MMCM/PLL 时钟发生相移且 MMCM/PLL 的 PHASESHIFT_MODE 属性设置为 LATENCY 时，自动衍生时钟以 S 属性（时延(Latency)模式下的管脚相移）来标记。此外，时钟报告的 Generated Clocks 部分下的时钟详情可显示 MMCM/PLL 插入延迟中计入的管脚相移量。

注释：仅报告对应于自动衍生时钟相移的延迟。来自 MMCM/PLL 块的相移量不包含在自动衍生时钟波形定义中。

在以下示例中，MMCM 的 PHASESHIFT_MODE 属性设置为 LATENCY。自动衍生时钟 clk_out1_clk_wiz_0 针对 MMCM 管脚 CLKOUT0 未定义相移，但时钟 clk_out2_clk_wiz_0 针对 MMCM 管脚 CLKOUT2 已定义 -90 度相移。

```
Attributes
P: Propagated
G: Generated
A: Auto-derived
R: Renamed
V: Virtual
I: Inverted
S: Pin phase-shifted with Latency mode

Clock          Period(ns)   Waveform(ns)   Attributes   Sources
clk_in1        10.000      {0.000 5.000}   P           {clk_in1}
clk_out1_clk_wiz_0 10.000      {0.000 5.000}   P,G,A       {clknetwork/
inst/mmcme3_adv_inst/CLKOUT0}
clk_out2_clk_wiz_0 10.000      {0.000 5.000}   P,G,A,S     {clknetwork/
inst/mmcme3_adv_inst/CLKOUT2}

=====
Generated Clocks
=====
```

```

Generated Clock      : clk_out1_clk_wiz_0
Master Source       : clknetwork/inst/mmcme3_adv_inst/CLKIN1
Master Clock        : clk_in1
Multiply By         : 1
Generated Sources   : {clknetwork/inst/mmcme3_adv_inst/CLKOUT0}

Generated Clock      : clk_out2_clk_wiz_0
Master Source       : clknetwork/inst/mmcme3_adv_inst/CLKIN1
Master Clock        : clk_in1
Multiply By         : 1
Pin Phase Shift(ns) : -2.5 (-90 degrees)
Generated Sources   : {clknetwork/inst/mmcme3_adv_inst/CLKOUT2}

```

时钟偏差和不确定性

偏差和不确定性都会影响建立和保持时间的计算和裕量。

偏差定义

时钟偏差表示目标时钟路径与源时钟路径之间：(1) 从设计中两条路径的公共点；(2) 分别到端点和起点时序单元时钟关键的插入延迟。

在以下公式中：

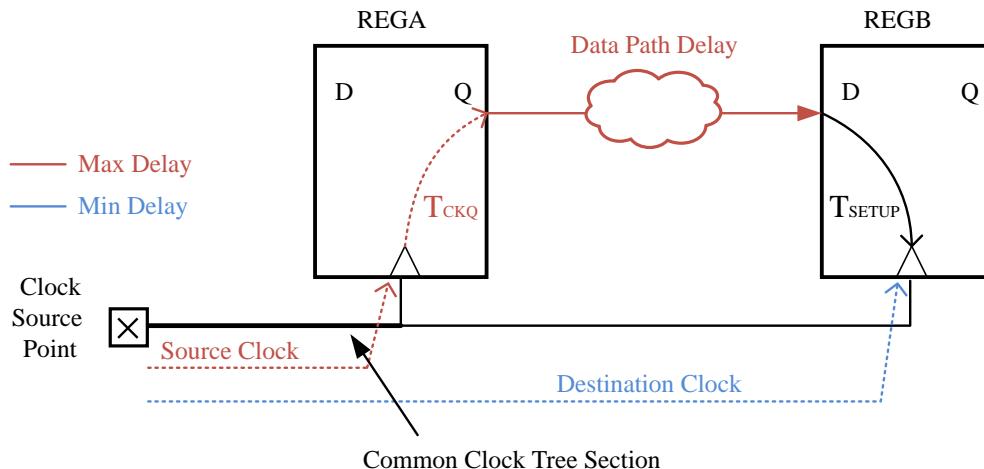
- T_{cj} 表示从公共节点到端点时钟管脚的延迟。
- T_{ci} 表示从公共节点到起点时钟管脚的延迟。

$$T_{skew_i,j} = T_{ci} - T_{cj}$$

时钟消极因素移除 (Clock Pessimism Removal)

典型时序路径报告可显示源时钟路径和目标时钟路径从时钟根到时序单元时钟管脚的延迟详情。如下所述，源时钟和目标时钟采用不同延迟来进行分析，即使在其公共电路上也是如此。

图 178：公共时钟树部分



X13436

此公共部分的延迟差异在偏差计算中引入了额外的消极因素。为避免不现实的裕量计算，通过称为时钟消极因素移除 (CPR) 值的延迟来对此消极因素予以补偿。

```
Clock Pessimism Removal (CPR) = common clock circuitry (max delay - min delay)
```

根据所执行的分析类型，对偏差加上或减去 CPR：

- 最大延迟分析（建立/恢复）
CPR 与目标时钟路径延迟相加。
- 最小延迟分析（保持/移除）
目标时钟路径延迟减去 CPR。

Vivado Design Suite 时序按如下所示方式报告每条时序路径的时钟偏差（在此例中执行保持时间分析）：

- DCD - 目标时钟延迟
- SCD - 源时钟延迟
- CPR - 时钟消极因素移除

```
Clock Path Skew: 0.301ns (DCD - SCD - CPR)
Destination Clock Delay (DCD): 2.581ns
Source Clock Delay (SCD): 2.133ns
Clock Pessimism Removal (CPR): 0.147ns
```

大多数情况下，CPR 准确性在布线前后会发生改变。以包含源时钟与目标时钟为相同时钟的时序路径为例，起点和端点时钟管脚由相同时钟缓存驱动。

布线前，公共点为时钟信号线驱动，即时钟缓存输出管脚。CPR 仅补偿从时钟根到时钟缓存输出管脚的消极因素。

布线后，公共点为器件架构中源和目标时钟路径共享的最后一项布线资源。此公共点不显示在网表中，因此，无法通过从时序报告中减去公共时钟电路延迟差值来直接检索对应 CPR。时序引擎根据不直接向用户公开的器件信息来计算此 CPR 值。

乐观偏差

赛灵思 FPGA 器件可提供高级时钟设置资源，如专用时钟布线树和时钟修改块 (CMB)。部分 CMB 具有使用锁相环电路（存在于 PLL 或 MMCM 原语中）来补偿时钟树插入延迟的功能。补偿的量取决于 PLL 的反馈回路上存在的插入延迟。大部分情况下，PLL（或 MMCM）可驱动多个具有同类型缓存的时钟树，包括反馈回路上的时钟树。由于器件可能较大，所有这些时钟树分支上的插入延迟并不总能与反馈回路延迟相匹配。如果反馈回路延迟大于源或目标时钟延迟，那么由 PLL 驱动的时钟将出现过补偿。在此情况下，CPR 迹象会发生改变，并且可从裕量值中有效移除偏差乐观。其作用是可确保在分析期间，任意时序路径时钟的公共节点上都不存在人为偏差。



建议：请始终在时序分析期间使用 CPR 补偿，以保持裕量准确性和总体时序验收质量。

时钟不确定性

时钟不确定性表示任意成对时钟沿之间可能存在的时间变化总量。不确定性由如下部分组成：计算所得时钟抖动（系统抖动、输入抖动和离散抖动）、某些硬件原语引入的相位误差以及用户在设计约束中指定的任意时钟不确定性 (set_clock_uncertainty)。

对于基准时钟，抖动由 `set_input_jitter` 和 `set_system_jitter` 定义。对于时钟生成器（如 MMCM 和 PLL），该工具会基于其源时钟及其配置上的用户指定的抖动来计算抖动。对于其它生成时钟（例如，基于触发器的时钟分频器），抖动与其源时钟的抖动相同。

用户指定的时钟不确定性将与 Vivado® Design Suite 时序引擎计算所得不确定性相加。对于生成的时钟（例如，从 MMCM、PLL 和基于触发器的时钟分频器生成的时钟），用户在源时钟上指定的不确定性不会通过时钟生成器传输。

如需了解有关抖动和相位误差定义的更多信息，请参阅《Vivado Design Suite 用户指南：使用约束》([UG903](#))。

时钟不确定性有如下 2 个用途：

- 在裕量数值中保留一定量的裕度，用于表示时钟上可能影响硬件功能的任何噪声。由于延迟和抖动数值为保守估值，赛灵思不建议额外增加不确定性以确保硬件功能正常。
- 在 1 个或多个实现步骤内，可对时钟或时钟对相关路径进行过约束。这样可增加 QoR 裕度，以便用于帮助后续步骤在这些路径上达成时序收敛。使用时钟不确定性时，不会修改时钟波形及其关系，因此仍可正确应用其余时序约束。

脉冲宽度检查

脉冲宽度检查属于针对信号波形的规则检查，当波形传输穿越器件并到达硬件原语时执行。这些检查通常对应于原语内部电路所规定的功能限制。例如，DSP 时钟管脚上的最小周期检查可确保驱动 DSP 实例的时钟的运行频率不高于内部 DSP 可承受的频率。

脉冲宽度检查不影响综合或实现。其分析必须在生成比特流之前执行一次，就像 Vivado Design Suite 所提供的所有其它设计规则检查一样。

发生脉冲宽度违例时，原因可能是时钟定义错误（脉冲宽度和周期检查），或者因时钟拓扑错误而导致偏差过大（`max_skew` 检查）。您必须复查目标器件的赛灵思 FPGA 数据手册以了解发生违例的原语的正确操作范围。对于偏差违例，您必须简化时钟树或者将时钟资源布局到更靠近发生违例的管脚的位置。

查看时序路径报告

时序路径报告可提供了解导致时序违例的原因所需的信息。以下章节描述了时序路径报告。

“时序路径汇总 (Timing Path Summary)” 显示了时序路径详情中的重要信息。复查该报告即可了解违例原因，无需分析时序路径。其中包含裕量、路径要求、数据路径延迟、单元延迟、布线延迟、时钟偏差和时钟不确定性的相关信息。它不提供有关单元布局的任何信息。

如需了解用于时序约束和时序分析的术语的相关信息，以及有关裕量和路径要求确定方式的信息，请参阅[了解时序分析的基础知识](#)。

时序路径汇总头文件示例

下图显示了文本报告中时序路径汇总头文件的示例。

图 179：文本报告中的时序路径汇总头文件

```

Slack (MET) : 0.722ns (required time - arrival time)
Source: fftEngine/fftInst/error_reg/C
        (rising edge-triggered cell FDRE clocked by fftClk_0 {rise@0.000ns fall@2.500ns period=5.000ns})
Destination: cpuEngine/iwb_biu/wb_stb_o_reg/D
        (rising edge-triggered cell FDCE clocked by wbClk_4 {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group: wbClk_4
Path Type: Setup (Max at Slow Process Corner)
Requirement: 5.000ns (wbClk_4 rise@10.000ns - fftClk_0 rise@5.000ns)
Data Path Delay: 3.905ns (logic 0.388ns (9.935%) route 3.517ns (90.065%))
Logic Levels: 3 (LUT4=1 LUT6=2)
Clock Path Skew: -0.190ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD): -1.471ns = ( 8.529 - 10.000 )
    Source Clock Delay (SCD): -2.117ns = ( 2.883 - 5.000 )
    Clock Pessimism Removal (CPR): -0.836ns
Clock Uncertainty: 0.172ns ((ISJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter (ISJ): 0.071ns
    Discrete Jitter (DJ): 0.077ns
    Phase Error (PE): 0.120ns

```

下图显示了 Vivado IDE 中时序路径汇总头文件的示例。

图 180：Vivado IDE 中时序路径汇总头文件

Summary	
Name	Path 1
Slack	0.722ns
Source	fftEngine/fftInst/error_reg/C (rising edge-triggered cell FDRE clocked by fftClk_0 {rise@0.000ns fall@2.500ns period=5.000ns})
Destination	cpuEngine/iwb_biu/wb_stb_o_reg/D (rising edge-triggered cell FDCE clocked by wbClk_4 {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group	wbClk_4
Path Type	Setup (Max at Slow Process Corner)
Requirement	5.000ns (wbClk_4 rise@10.000ns - fftClk_0 rise@5.000ns)
Data Path Delay	3.905ns (logic 0.388ns (9.935%) route 3.517ns (90.065%))
Logic Levels	3 (LUT4=1 LUT6=2)
Clock ... Skew	-0.190ns
Clock U...tainty	0.172ns

时序路径头文件信息

时序路径汇总头文件包含以下信息：

- 裕量 (Slack)

裕量为正值表明路径满足衍生自时序约束的路径要求。“Slack”公式取决于所执行的分析。

- 最大延迟分析（建立/恢复）： $slack = data\ required\ time - data\ arrival\ time$
- 最小延迟分析（保持/移除）： $slack = data\ arrival\ time - data\ required\ time$

所需数据和到达时间在时序路径报告的其它小节中进行计算和报告。

- 源 (Source)

路径起点和发送数据的源时钟。起点通常为时序单元的时钟管脚或输入端口。

如果适用，第 2 行可显示原语和时钟管脚的时钟沿敏感性。它还可提供时钟名称和时钟沿定义（波形和周期）。

- 目标 (Destination)

路径端点和捕获数据的目标时钟。端点通常为目标时序单元的输入数据管脚或输出端口。如果适用，第 2 行可显示原语和时钟管脚的时钟沿敏感性。它还可提供时钟名称和时钟沿定义（波形和周期）。

- 路径组 (Path Group)

路径端点所属的时序组。该组通常由目标时钟定义，但归入 `**async_default**` 时序组的异步时序检查（恢复/移除）除外。用户定义的组同样显示在此处。这样便于报告。

- 路径类型 (Path Type)

此路径上执行的分析类型。

- Max：指示用于计算数据路径延迟的最大延迟值，对应于建立和恢复分析。
- Min：指示用于计算数据路径延迟的最小延迟值，对应于保持和移除分析。

该行还可显示报告使用的角点 (corner)：Slow 或 Fast。

- 要求 (Requirement)

当起点和端点由相同时钟控制时或者由无相移的时钟控制时，时序路径要求通常为：

- 1 个时钟周期（针对建立/恢复分析）。
- 0 ns（针对保持/移除分析）。

当路径位于 2 个不同时钟之间时，要求对应于任意源时钟沿和目标时钟沿之间的最小正差值。该值被时序例外约束（例如，多周期路径、最大延迟和最小延迟）所覆盖。

如需了解有关如何从时序约束衍生时序路径要求的更多信息，请参阅[时序路径](#)。

- 数据路径延迟 (Data Path Delay)

通过路径的逻辑部分的累积延迟。除非时钟用作为数据，否则排除时钟延迟。延迟类型对应于“Path Type”行所描述的类型。

- 逻辑级数 (Logic Levels)

路径的数据部分中包含的每种类型的原语的数量，不包括起点和端点单元。

- 时钟路径偏差 (Clock Path Skew)

源时钟的发送沿与目标时钟的捕获沿之间的插入延迟差加上时钟消极因素校正（如果有）。

- 目标时钟延迟 (Destination Clock Delay, DCD)

从目标时钟源点到路径端点的累积延迟。

- 对于最大延迟分析（建立/恢复），使用最小单元和信号线延迟值。
- 对于最小延迟分析（保持/移除），使用最大延迟值。

- 源时钟延迟 (Source Clock Delay, SCD)

从时钟源点到路径起点的累积延迟。

- 对于最大延迟分析（建立/恢复），使用最大单元和信号线延迟值。
- 对于最小延迟分析（保持/移除），使用最小延迟值。

- 时钟消极因素移除 (Clock Pessimism Removal, CPR)

由于源时钟与目标时钟是以不同类型的延迟报告的（即使在公共电路上也是如此），而导致的额外时钟偏差量绝对值。

移除此额外消极因素后，源时钟与目标时钟的公共电路上不再有任何偏差。

对于已布线的设计，最后一个公共时钟树节点通常位于时钟信号线所使用的布线资源内，在路径详情中不予报告。

- 时钟不确定性 (Clock Uncertainty)

任意成对时钟沿之间可能的时间变化总量。

不确定性由如下部分组成：计算所得时钟抖动（系统抖动和离散抖动）、某些硬件原语引入的相位误差以及用户在设计约束中指定的任意时钟不确定性 (`set_clock_uncertainty`)。

用户时钟不确定性是 Vivado IDE 时序引擎计算所得不确定性的补充。

- 总系统抖动 (Total System Jitter, TSJ)

应用于源时钟和目标时钟的系统抖动总和。要全局修改系统抖动，请使用 `set_system_jitter` 约束。虚拟时钟处于理想状态，因此不含任何系统抖动。

- 总输入抖动 (Total Input Jitter, TIJ)

源时钟和目标时钟的输入抖动总和。

要为每个基准时钟单独定义输入抖动，请使用 `set_input_jitter` 约束。Vivado IDE 时序引擎基于生成时钟的主时钟抖动和遍历的时钟资源来计算生成时钟的输入抖动。默认情况下，虚拟时钟处于理想状态，因此不含任何抖动。

如需了解有关时钟不确定性和抖动的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的相应内容。

- 离散抖动 (Discrete Jitter, DJ)

由硬件原语（如 MMCM 或 PLL）引入的抖动量。

Vivado IDE 时序引擎基于这些单元的配置来计算该值。

- 相位误差 (Phase Error, PE)

由硬件原语（如 MMCM 或 PLL）引入的 2 个时钟信号间的相位变化量。

Vivado IDE 时序引擎自动提供该值，并将其与时钟不确定性相加

- 用户不确定性 (User Uncertainty, UU)

由 `set_clock_uncertainty` 约束指定的额外的不确定性。

如需了解有关如何使用此命令的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的相应内容。

根据时序约束、报告的路径和目标器件，在“Timing Path Summary”中可能会显示其它行：

- SLR 间补偿 (Inter-SLR Compensation)

（仅限赛灵思 7 系列 SSI 器件）安全报告跨 SLR 边界的路径所需的额外裕度。

- 输入延迟 (Input Delay)

由输入端口上的 `set_input_delay` 约束指定的输入延迟值。仅限针对始于输入端口的路径才显示该行。

- 输出延迟 (Output Delay)

由输出端口上的 `set_output_delay` 约束指定的输出延迟值。仅限针对止于输出端口的路径才显示该行。

- 时序例外 (Timing Exception)

覆盖路径的时序例外。仅显示优先级最高的例外，因为它是影响时序路径要求的唯一例外。

如需了解有关时序例外及其优先级规则的信息，请参阅[时序路径](#)。

时序路径详情

此报告的第 2 部分提供了有关路径遍历的单元、管脚、端口和信号线的更多详细信息。它分为 3 个部分：

- 源时钟路径

源时钟从其源点到数据路径的起点遍历的电路。本部分针对从输入端口开始的路径不存在。

- 数据路径

数据从起点到端点遍历的电路。

- 目标时钟路径

目标时钟从其源点到数据路径端点时钟管脚遍历的电路。

“源时钟路径”部分和“数据路径”部分结合在一起工作。这两部分始终报告相同类型的延迟：

- 针对建立/恢复分析报告最大延迟

- 针对保持/移除分析报告最小延迟

这两部分共享累积延迟，从数据发送沿时间开始，累积穿过源时钟和数据路径的延迟。最终累积的延迟值称为数据到达时间 (data arrival time)。

目标时钟路径所报告的延迟始终与源时钟和数据路径相反。其初始累积延迟值即在目标时钟源点上发送数据捕获沿的时间。最终累积延迟值称为数据必需时间 (data required time)。

最终报告行汇总了裕量的计算方式。

- 针对最大延迟分析（建立/恢复）

```
slack = data required time - data arrival time
```

- 针对最小延迟分析（保持/移除）

```
slack = data arrival time - data required time
```

文本报告中的时序路径详情

以下是文本报告中的“源时钟 (Source Clock)”、“数据 (Data)”和“目标时钟路径 (Destination Clock Paths)”示例。由于此路径仅覆盖 5 ns 的简单周期约束，因此源时钟发送沿从 0 ns 开始，而目标时钟捕获沿从 5 ns 开始。

图 181：文本报告中的时序路径详情

Location	Delay type	Incr(ns)	Path(ns)	Netlist Resource(s)
(clock fftClk_0 rise edge)				
P23		5.000	5.000 r	
	net (fo=0)	0.000	5.000 r	sysClk (IN)
P23	IBUF (Prop_ibuf_I_0)	0.767	5.767 r	clkini1_buf/0
	net (fo=2, routed)	1.081	6.848	clkgen/sysClk_int
MMCME2_ADV_X0Y0	MMCME2_ADV (Prop_mmcm2_adv_CLKIN1_CLKOUT5)	-7.014	-0.166 r	clkgen/mmcm_adv_inst/CLKOUT5
	net (fo=1, routed)	1.732	1.566	clkgen/fftClk_0
BUFGCTRL_X0Y5	BUFGE (Prop_bufg_I_0)	0.093	1.659 r	clkgen/clkout6_buf/0
	net (fo=1436, routed)	1.224	2.883	fftEngine/fftInst/fftClk
SLICE_X20Y144	FDRE		r	fftEngine/fftInst/error_reg/C
(clock X0Y144 rise edge)				
SLICE_X20Y144	FDRE (Prop_fdre_C_Q)	0.259	3.142 f	fftEngine/fftInst/error_reg/Q
	net (fo=2, routed)	1.764	4.907	cpuEngine/cpu_iwb_adr_o/buffer_fifo/s3_err_i
SLICE_X7Y73	LUT6 (Prop_lut6_I1_0)	0.043	4.950 f	cpuEngine/cpu_iwb_adr_o/buffer_fifo/wb_stb_o_reg_i_6/0
	net (fo=1, routed)	0.650	5.599	cpuEngine/cpu_iwb_adr_o/buffer_fifo/n_0_wb_stb_o_reg_i_6
SLICE_X3Y63	LUT4 (Prop_lut4_I0_0)	0.043	5.642 f	cpuEngine/cpu_iwb_adr_o/buffer_fifo/wb_stb_o_reg_i_2/0
	net (fo=3, routed)	0.669	6.312	cpuEngine/iwb_biu/m0_err_o
SLICE_X0Y55	LUT6 (Prop_lut6_I0_0)	0.043	6.355 r	cpuEngine/iwb_biu/wb_stb_o_reg_i_1/0
	net (fo=2, routed)	0.434	6.788	cpuEngine/iwb_biu/wb_stb_o_0
SLICE_X0Y55	FDCE		r	cpuEngine/iwb_biu/wb_stb_o_reg/D
(clock wbClk_4 rise edge)				
P23		10.000	10.000 r	
	net (fo=0)	0.000	10.000 r	sysClk (IN)
P23	IBUF (Prop_ibuf_I_0)	0.692	10.692 r	clkini1_buf/0
	net (fo=2, routed)	0.986	11.678	clkgen/sysClk_int
MMCME2_ADV_X0Y0	MMCME2_ADV (Prop_mmcm2_adv_CLKIN1_CLKOUT1)	-6.008	5.670 r	clkgen/mmcm_adv_inst/CLKOUT1
	net (fo=1, routed)	1.619	7.289	clkgen/wbClk_4
BUFGCTRL_X0Y3	BUFGE (Prop_bufg_I_0)	0.083	7.372 r	clkgen/clkout2_buf/0
	net (fo=1495, routed)	1.157	8.529	cpuEngine/iwb_biu/wbClk
SLICE_X0Y55	FDCE		r	cpuEngine/iwb_biu/wb_stb_o_reg/C
	clock pessimism	-0.836	7.693	
	clock uncertainty	-0.172	7.521	
SLICE_X0Y55	FDCE (Setup_fdce_C_D)	-0.010	7.511	cpuEngine/iwb_biu/wb_stb_o_reg
required time				
			7.511	
arrival time				
			-6.788	
slack				
			0.722	

Vivado IDE 中的“时序路径详情”

Vivado IDE 中的“时序路径详情 (Timing Path Details)”（如下图所示）所示信息与前图中所示文本报告中显示的信息相同。

图 182：Vivado IDE 中的“时序路径详情”

Source Clock Path				
Delay Type	Incr (ns)	Path (n...)	Location	Netlist Resource(s)
(clock ffcIck_0 rise edge)	(r) 5.000	5.000		
	(r) 0.000	5.000	Site: P23	sysClk
net (fo=0)	0.000	5.000		sysClk
			Site: P23	
IBUF (Prop_ibuf_I_O)	(r) 0.767	5.767	Site: P23	clkIn1_bufI
net (fo=2, routed)	1.081	6.848		clkIn1_bufIO
			Site: MMCM..._ADV_X0Y0	clkgen/mmcmm_adv_inst/CLKIN1
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT5)	(r)...014	-0.166	Site: MMCM..._ADV_X0Y0	clkgen/mmcmm_adv_inst/CLKOUT5
net (fo=1, routed)	1.732	1.566		clkgen/ffcIck_0
			Site: BUFGCTRL_X0Y5	clkgen/clkout6_bufI
BUFG (Prop_bufq_I_O)	(r) 0.093	1.659	Site: BUFGCTRL_X0Y5	clkgen/clkout6_bufIO
net (fo=1436, routed)	1.224	2.883		fftEngine/fftInst/fftClk
FDRE			Site: SLICE_X20Y144	fftEngine/fftInst/error_reg/C
Data Path				
Delay Type	Incr (ns)	Path (n...)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(f) 0.259	3.142	Site: SLICE_X20Y144	fftEngine/fftInst/error_reg/Q
net (fo=2, routed)	1.764	4.907		cpuEngine/cpu_iwb_adr_o/buffer_fifo/s3_err_i
			Site: SLICE_X7Y73	cpuEngine/cpu_iwb_adr_o/buffer_fifo/wb_stb_o_reg_i_6/I1
LUT6 (Prop_lut6_I1_O)	(f) 0.043	4.950	Site: SLICE_X7Y73	cpuEngine/cpu_iwb_adr_o/buffer_fifo/wb_stb_o_reg_i_6/O
net (fo=1, routed)	0.650	5.599		cpuEngine/cpu_iwb_adr_o/buffer_fifo/n_0_wb_stb_o_reg...
			Site: SLICE_X3Y63	cpuEngine/cpu_iwb_adr_o/buffer_fifo/wb_stb_o_reg_i_2/I0
LUT4 (Prop_lut4_I0_O)	(f) 0.043	5.642	Site: SLICE_X3Y63	cpuEngine/cpu_iwb_adr_o/buffer_fifo/wb_stb_o_reg_i_2/O
net (fo=3, routed)	0.669	6.312		cpuEngine/iwb_biu/m0_err_o
			Site: SLICE_X0Y55	cpuEngine/iwb_biu/wb_stb_o_reg_i_1/I0
LUT6 (Prop_lut6_I0_O)	(r) 0.043	6.355	Site: SLICE_X0Y55	cpuEngine/iwb_biu/wb_stb_o_reg_i_1/O
net (fo=2, routed)	0.434	6.788		cpuEngine/iwb_biu/wb_stb_o0
FDCE			Site: SLICE_X0Y55	cpuEngine/iwb_biu/wb_stb_o_reg/D
Arrival Time				
		6.788		
Destination Clock Path				
Delay Type	Incr (ns)	Path (n...)	Location	Netlist Resource(s)
(clock wbClk_4 rise edge)	(r)...000	10.000		
	(r) 0.000	10.000	Site: P23	sysClk
net (fo=0)	0.000	10.000		sysClk
			Site: P23	clkIn1_bufI
IBUF (Prop_ibuf_I_O)	(r) 0.692	10.692	Site: P23	clkIn1_bufIO
net (fo=2, routed)	0.986	11.678		clkgen/sysClk_int
			Site: MMCM..._ADV_X0Y0	clkgen/mmcmm_adv_inst/CLKIN1
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT1)	(r)...008	5.670	Site: MMCM..._ADV_X0Y0	clkgen/mmcmm_adv_inst/CLKOUT1
net (fo=1, routed)	1.619	7.289		clkgen/wbClk_4
			Site: BUFGCTRL_X0Y3	clkgen/clkout2_bufI
BUFG (Prop_bufq_I_O)	(r) 0.083	7.372	Site: BUFGCTRL_X0Y3	clkgen/clkout2_bufIO
net (fo=1495, routed)	1.157	8.529		cpuEngine/iwb_biu/wbClk
FDCE			Site: SLICE_X0Y55	cpuEngine/iwb_biu/wb_stb_o_reg/C
clock pessimism	-0.836	7.693		
clock uncertainty	-0.172	7.521		
FDCE (Setup_fdce_C_D)	-0.010	7.511	Site: SLICE_X0Y55	cpuEngine/iwb_biu/wb_stb_o_reg
Required Time				
		7.511		

使用标准流程时，按 5 个列显示有关路径的信息，使用“增量编译 (Incremental Compile)”时，则显示 6 个列。

- 位置 (Location)

器件上单元或端口所在位置。

- 延迟类型 (Delay Type)

路径遵循的 unisim 原语和特定时序 arc。对于信号线，它会显示扇出 (f_o) 及其状态。信号线可处于以下状态：

- 未布局 (Unplaced)：驱动和负载尚未布局。
 - 已估算 (Estimated)：驱动和/或负载已布局。已部分布线的信号线同样报告为“estimated”。
 - 已布线 (Routed)：驱动和负载均已布局，并且信号线已完全布线。
- 增量 (Incr) (ns) (文本报告) / 延迟 (Delay) (IDE 报告)
与 unisim 原语时序 arc 或信号线关联的增量延迟值。它还可显示约束，例如，输入/输出延迟或时钟不确定性。
- 路径 (Path) (ns) (文本报告) / 累积 (Cumulative) (IDE 报告)
位于每个路径段后的累积延迟。在给定行上，其值为来自前一行的累积值 + 当前行的增量延迟。
- 网表资源 (Netlist Resource) (文本报告) / 逻辑资源 (Logical Resource) (IDE 报告)
遍历的网表对象的名称。
- 管脚复用 (Pin Reuse) (仅限增量编译)
指示路径是否是从参考运行复用所得。适用的值为 ROUTING、PLACEMENT、MOVED 和 NEW。

每个增量延迟都与下列沿感应之一关联：

- r (上升沿)
- f (下降沿)

初始沿感应由用于分析的发送沿或捕获沿判定。它可由路径中任意单元反转，具体取决于时序 arc 的性质。例如，位于反相器输入处的上升沿在输出处会变为下降沿。

沿感应有助于识别在源或目标时钟树上的时钟沿反转所导致的时序路径要求过于苛刻的问题。

时序验收的验证

详细查看时序分析前，了解时序报告中哪部分表明设计已准备好在硬件上运行是很重要的。



重要提示！ 当设计完成布局布线后，时序验收是实现结果分析中的必要步骤。

默认情况下，在 Vivado Design Suite 中使用工程时，运行会自动生成“Report Timing Summary”的文本版本。您还可在内存中加载实现后设计检查点之后以交互方式生成此报告。



重要提示！ “Report Timing Summary”不涵盖总线偏差约束。要报告总线偏差约束，必须在命令行上单独运行 `report_bus_skew` 命令。针对此命令不提供 GUI 支持。

要了解完整的时序验收验证方法，请访问[此链接](#)以参阅《UltraFast 设计方法指南（适用于 Vivado Design Suite）》([UG949](#)) 中的相应内容。

第 6 章

综合分析与收敛技巧

使用细化视图对 RTL 进行最优化

完成任意实现步骤后使用 `report_timing`、`report_timing_summary` 或 `report_design_analysis` 分析时序结果时，您必须审查关键路径结构，了解是否可通过修改 RTL、使用综合属性或者使用其它综合选项来更有效地将其映射到逻辑原语。这对于包含大量逻辑层次的路径尤为重要，因为大量逻辑层次会给实现工具施加压力并限制总体设计性能。

只要发现具有大量逻辑层次的路径，就必须确认路径功能是否需要如此大量的逻辑层次。通常确认逻辑层次的最佳数量不容易，因为它取决于您对设计的了解以及您对总体 RTL 最优化的了解程度。观察综合后经过最优化的网表并识别 RTL 中的问题来源及其改进方法是一项复杂的任务。

在工程模式下，Vivado® IDE 可在经过综合或实现的设计与经过细化的设计之间提供强大的交叉探测机制，从而帮助简化分析。请执行以下操作以对综合后/实现后的设计和细化设计进行交叉探测：

1. 打开内存中综合后/实现后的设计和细化的设计。
2. 选择综合后/实现后设计视图中的时序路径，按“F4”键显示其原理图。
3. 选择“Flow Navigator”窗格中的“Elaborated Design”。这样会同时选中对应于时序路径的 RTL 单元，以便您可按“F4”键打开 RTL 原理图并在细化视图中查看该相同路径或者从端点管脚反向走线回到起点单元。
4. 复查路径遍历的 RTL 逻辑，特别注意运算符或矢量的大小。

示例

在以下示例中，用户编写了一个计数器，如下所示：

图 183：简单计数器 VHDL 示例

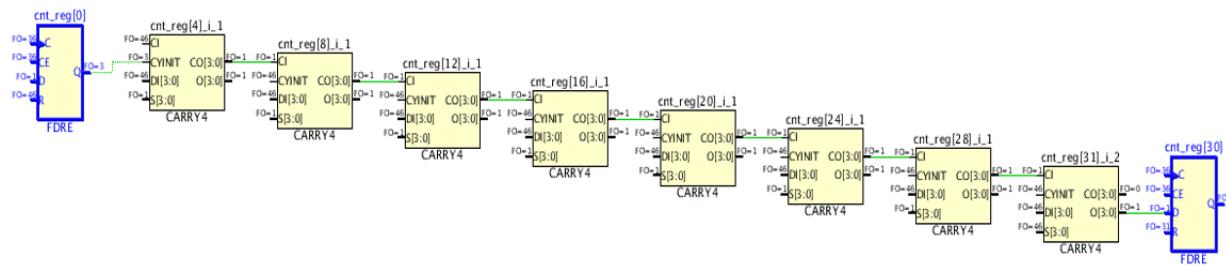
```

signal cnt : integer := 0;

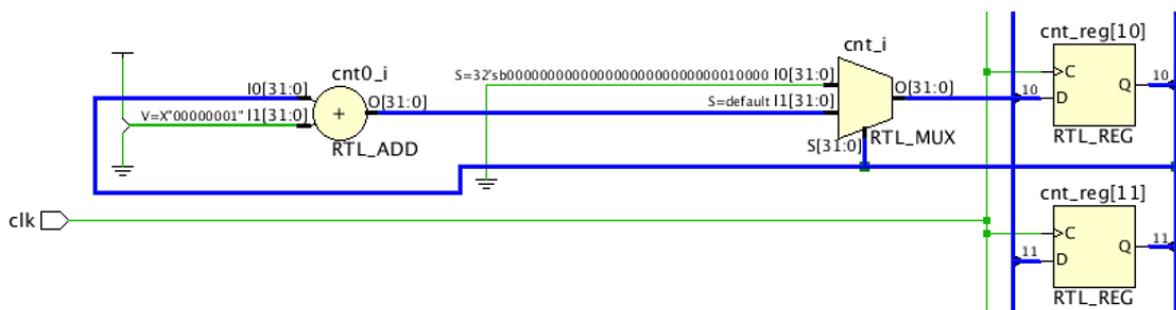
process (clk)
begin
    if(clk'event and clk = '1') then
        if(cnt = 16) then
            cnt <= 0;
        else
            cnt <= cnt + 1;
        end if;
        if(cnt = 8) then
            dout <= din0;
        else
            dout <= din1;
        end if;
    end if;
end process;

```

信号 `cnt` 按从 0 到 16 进行计数，需 5 位矢量用于编码。布线后关键路径原理图如下图所示。端点为 `cnt` 信号的第 30 位。

图 184：`cnt` 计数器布线后关键路径原理图

选中关键路径的起点和端点单元后，只需打开选定单元的原理图并展开从端点管脚返回到起点的逻辑，即可在细化视图中直观展示等效路径，如下图所示。

图 185：细化视图下的 `cnt` 计数器

细化视图显示加法器输入大小限制为 32 位，因为信号 `cnt` 已声明为整数。在此特定示例中，在整个综合最优化过程中均保持使用 32 位运算符。细化视图可明示所发生的变化，您可按如下方式更改 RTL 来获得进一步优化的网表和时序 QoR。随着计数器从 0 递增至 16，您可为信号 `cnt` 定义范围，以强制将加法器输入宽度设置为 5 位而不是 32 位。

图 186：含整数范围的简单计数器 VHDL 示例

```

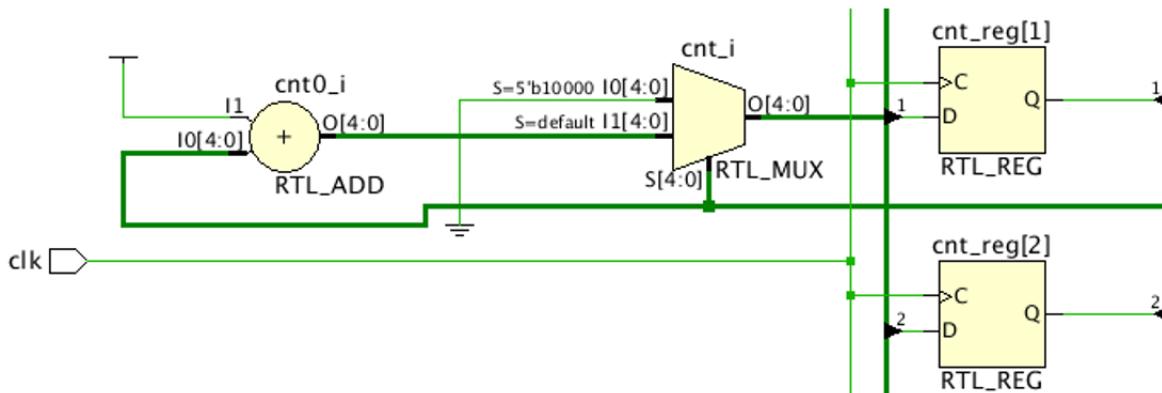
signal cnt : integer range 0 to 16 := 0;

process (clk)
begin
    if(clk'event and clk = '1') then
        if(cnt = 16) then
            cnt <= 0;
        else
            cnt <= cnt + 1;
        end if;
        if(cnt = 8) then
            dout <= din0;
        else
            dout <= din1;
        end if;
    end if;
end process;

```

对 RTL 代码执行的更改后续将影响综合最优化，您可使用细化视图验证其影响，而无需执行整个编译流程：

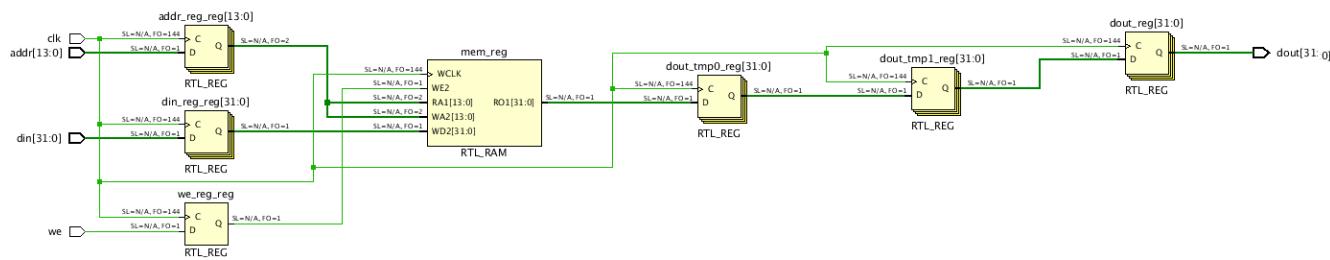
图 187：改进 RTL 后细化视图下的 cnt 计数器



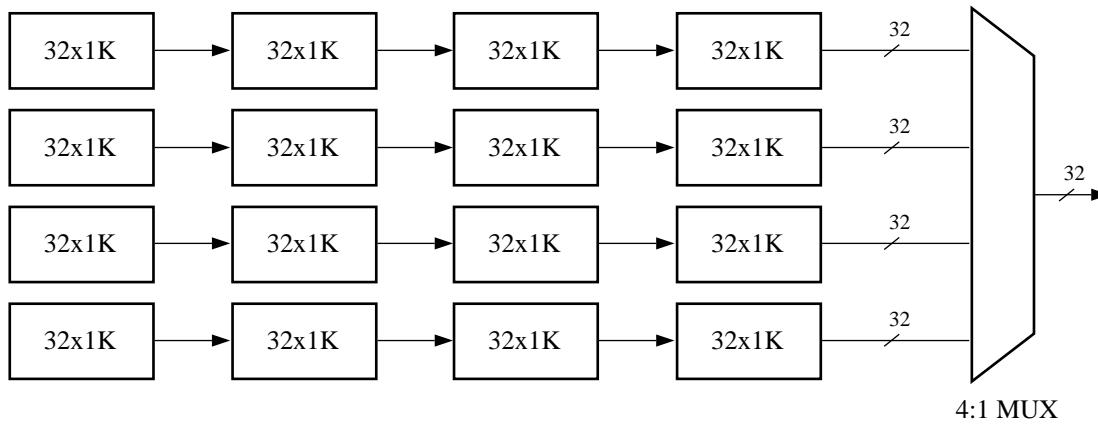
分解深层内存配置，实现功耗与性能平衡

在深层内存配置中，可使用综合属性 RAM_DECOMP 实现更好的内存分解并降低功耗。此属性可在 RTL 中设置。将 RAM_DECOMP 属性应用于内存时，内存是在较宽的原语配置中设置的，而不是在较深且较窄的配置中设置的。

当 CASCADE_HEIGHT 属性与 RAM_DECOMP 属性搭配使用时，综合推断对级联具有更细化的控制权，因此可实现平衡的功耗与性能。此方法需要额外的地址解码逻辑，但可减少任意时间点访问的块 RAM 数量，这有助于降低功耗。下图中的内存配置 ($32 \times 16K$) 显示了设置 RAM_DECOMP 和 CASCADE_HEIGHT 属性时分解内存的方式示例。

图 188： $32 \times 16K$ 内存配置

如果应用属性 RAM_DECOMP = power 和 CASCADE_HEIGHT = 4，那么将按下图所示方式推断 16 RAMB36E2 并对内存进行分解。

图 189：使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 $32 \times 16K$ 内存配置生成的结构

X19321-052517

此处使用的基本原语为 $32 \times 1K$ ，4 个块 RAM 通过内置功能进行级联，组成 $32 \times 4K$ 配置。4 个此类并行结构可创建 1 个深度为 16K 的内存。输出通过多路复用来生成输出数据。

图 190：使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 $32 \times 16K$ 内存配置的 RTL 代码片段

```

module test
(
    input  clk,
    input  we,
    input  [13:0] addr,
    input  [31:0] din,
    output reg [31:0] dout
);

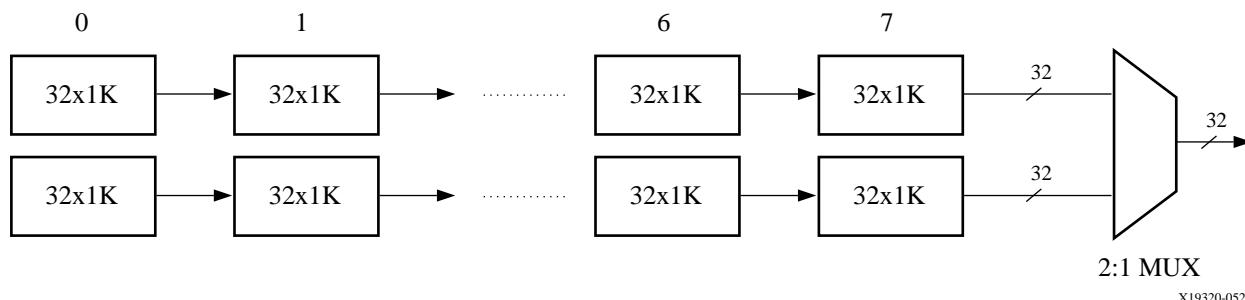
(* ram_style = "block", ram_decomp = "power", cascade_height = 4 *) reg [31:0] mem [(16*1024)-1:0];
reg [13:0] addr_reg;
reg [31:0] dout_tmpp0;
reg [31:0] dout_tmpp1;
reg [31:0] din_reg;
reg        we_reg;

always @(posedge clk)
begin
    addr_reg <= addr;
    din_reg  <= din;
    we_reg   <= we;
    dout_tmpp0 <= mem[addr_reg];
    dout_tmpp1 <= dout_tmpp0;
    dout <= dout_tmpp1;
    if (we_reg)
        mem[addr_reg] <= din_reg;
end

endmodule

```

如果仅应用 RAM_DECOMP = power 属性，则将按下图所示方式推断 16 RAMB36E2 并对内存进行分解。

图 191：使用 RAM_DECOMP 属性的 $32 \times 16K$ 内存配置生成的结构

此处使用的基本原语为 $32 \times 1K$ ，8 个块 RAM 通过内置功能进行级联，组成 $32 \times 8K$ 配置。2 个此类并行结构可创建 1 个深度为 $16K$ 的内存。输出通过多路复用器生成输出数据。多路复用器为 2:1 MUX。

图 192：使用 RAM_DECOMP 属性的 $32 \times 16K$ 内存配置的 RTL 代码片段

```
module test
(
    input  clk,
    input  we,
    input  [13:0] addr,
    input  [31:0] din,
    output reg [31:0] dout
);

(* ram_style = "block", ram_decomp = "power"*) reg [31:0] mem [(16*1024)-1:0];
reg [13:0] addr_reg;
reg [31:0] dout_tmp0;
reg [31:0] dout_tmp1;
reg [31:0] din_reg;
reg         we_reg;

always @(posedge clk)
begin
    addr_reg <= addr;
    din_reg  <= din;
    we_reg   <= we;
    dout_tmp0 <= mem[addr_reg];
    dout_tmp1 <= dout_tmp0;
    dout <= dout_tmp1;
    if (we_reg)
        mem[addr_reg] <= din_reg;
end

endmodule
```

对于这 2 个内存分解示例，总体功耗节省量是相似的，如图 189：使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 $32 \times 16K$ 内存配置生成的结构和图 191：使用 RAM_DECOMP 属性的 $32 \times 16K$ 内存配置生成的结构中所示，因为任意给定时间点仅有 1 个块 RAM 处于活动状态。但在性能方面，4 级深度的级联块 RAM 链（图 189：使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 $32 \times 16K$ 内存配置生成的结构）相比 8 级深度的级联块 RAM 链（图 191：使用 RAM_DECOMP 属性的 $32 \times 16K$ 内存配置生成的结构）提供的性能更好。

当内存深度不为 2 的幂时，优化 RAMB 利用率

以下测试案例可用于观察由综合工具生成的日志文件，了解是否可通过改进 RTL 来帮助工具改进。以下代码片段显示了 VHDL 中深度 40K 宽度为 36 位的内存描述。地址总线需占 16 位。

图 193：40K x 36 位内存 RTL 示例

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity test is
    Port ( clk : in STD_LOGIC;
            addr : in STD_LOGIC_VECTOR (15 downto 0);
            din  : in STD_LOGIC_VECTOR (35 downto 0);
            we   : in STD_LOGIC;
            dout : out STD_LOGIC_VECTOR (35 downto 0));
end test;

architecture rtl of test is
signal addr_reg : STD_LOGIC_VECTOR(15 downto 0);
type mem_type is array (0 to 40959) of STD_LOGIC_VECTOR(35 downto 0);
signal mem : mem_type;
begin

process (clk)
begin
    if (rising_edge(clk)) then
        addr_reg <= addr;
        if(we='1') then
            mem(to_integer(unsigned(addr_reg))) <= din;
        end if;
        dout <= mem(to_integer(unsigned(addr_reg)));
    end if;
end process;

end rtl;
```

通过使用 report_utilization 命令，可以看到综合工具生成 72 个块 RAM，如下图所示。

图 194：利用率报告中由综合生成的块 RAM 数量

Site Type	Used	Fixed	Available	Util%
Block RAM Tile	72	0	1030	6.99
RAMB36/FIFO*	72	0	1030	6.99
RAMB36E1 only	72			
RAMB18	0	0	2060	0.00

如计算针对 40K x 36 配置应推断的块 RAM 数量，所得到的块 RAM 数量将少于综合工具所生成的数量

以下显示了此内存配置的手动计算过程：

- 40K x 36 可细分为 2 个内存：(32K x 36) 和 (8K x 36)
- 需使用基于 MSB 地址位的地址解码器来支持其中一个内存执行读写操作，并选择正确的输出数据。
- 32K x 36 内存可通过 32 个 RAMB 实现： $4 * 8 * (4K \times 9)$
- 8K x 36 内存可通过 8 个 RAMB 实现： $8 * (1K \times 36)$
- 总计需要 40 个 RAMB 才能以最优化方式实现 40K x 36 内存。

为验证是否已推断出 RAMB 的最优数量，综合日志文件包含 1 个会话，其中详列了每个内存的配置方式及其映射到 FPGA 原语的方式。如下图所示，内存深度作为 64K 来处理，这表明深度不等于 2 的幂次方时，无法以最优化方式来加以处理。

图 195：综合日志中的 RAM 配置和映射部分

Start ROM, RAM, DSP and Shift Register Reporting										
Block RAM:										
Module Name	RTL Object	PORT A (Depth x Width)	W	R	PORT B (Depth x Width)	W	R	OUT_REG	RAMB18	Hierarchical Name
test	mem_reg	64 K x 36(READ_FIRST)	W R				Port A 0	72	test/exram_2	

综合工具使用的是 64K x 1（2 个具有级联功能的块 RAM），36 个此类结构（因为 36 位数据）。因此总计有 $36 \times 2 = 72$ 个块 RAM。下图显示的代码片段用于强制综合推断 RAMB 的最优数量。

图 196：最优化的 40 K x 36 位内存 RTL 示例

```

architecture rtl of test is
    signal addr_reg : std_logic_vector(15 downto 0);
    type ram_type_0 is array (0 to 32767) of std_logic_vector(35 downto 0);
    type ram_type_1 is array (0 to 8191) of std_logic_vector(35 downto 0);
    signal RAM_0 : ram_type_0;
    signal RAM_1 : ram_type_1;
    signal dout_0 : std_logic_vector(35 downto 0);
    signal dout_1 : std_logic_vector(35 downto 0);
begin
    process (clk)
    begin
        if clk'event and clk = '1' then
            addr_reg <= addr;
            if we = '1' and addr_reg(15) = '0' then
                RAM_0(to_integer(unsigned(addr_reg(14 downto 0)))) <= din;
            end if;
            dout_0 <= RAM_0(to_integer(unsigned(addr_reg(14 downto 0))));
        end if;
    end process;

    process (clk)
    begin
        if clk'event and clk = '1' then
            if we = '1' and addr_reg(15) = '1' then
                RAM_1(to_integer(unsigned(addr_reg(12 downto 0)))) <= din;
            end if;
            dout_1 <= RAM_1(to_integer(unsigned(addr_reg(12 downto 0))));
        end if;
    end process;

    dout <= dout_1 when addr_reg(15) = '1' else dout_0;
end rtl;

```

最优化 RAMB 输入逻辑以允许输出寄存器推断

以下 RTL 代码片段可从块 RAM（实际上为 ROM）生成关键路径，其中包含多个止于触发器(FF)的逻辑层次。RAMB 单元已在无可选输出寄存器(DOA-0)的情况下完成推断，这给 RAMB 输出路径增加了超过 1 ns 的额外延迟惩罚。

图 197：不含已推断的 RAMB 输出寄存器的内存 RTL 代码

```
module test (input clk, input [3:0] addr, output reg dout, dout_shift);

(* rom_style = "block" *) reg [15:0] mem [0:15];

reg [3:0] addr_reg0;
reg [3:0] addr_reg1;
reg [3:0] addr_reg2;
reg [3:0] addr_reg3;

reg [15:0] dout_mem;

initial
begin
    $readmemh("init.txt",mem);
end

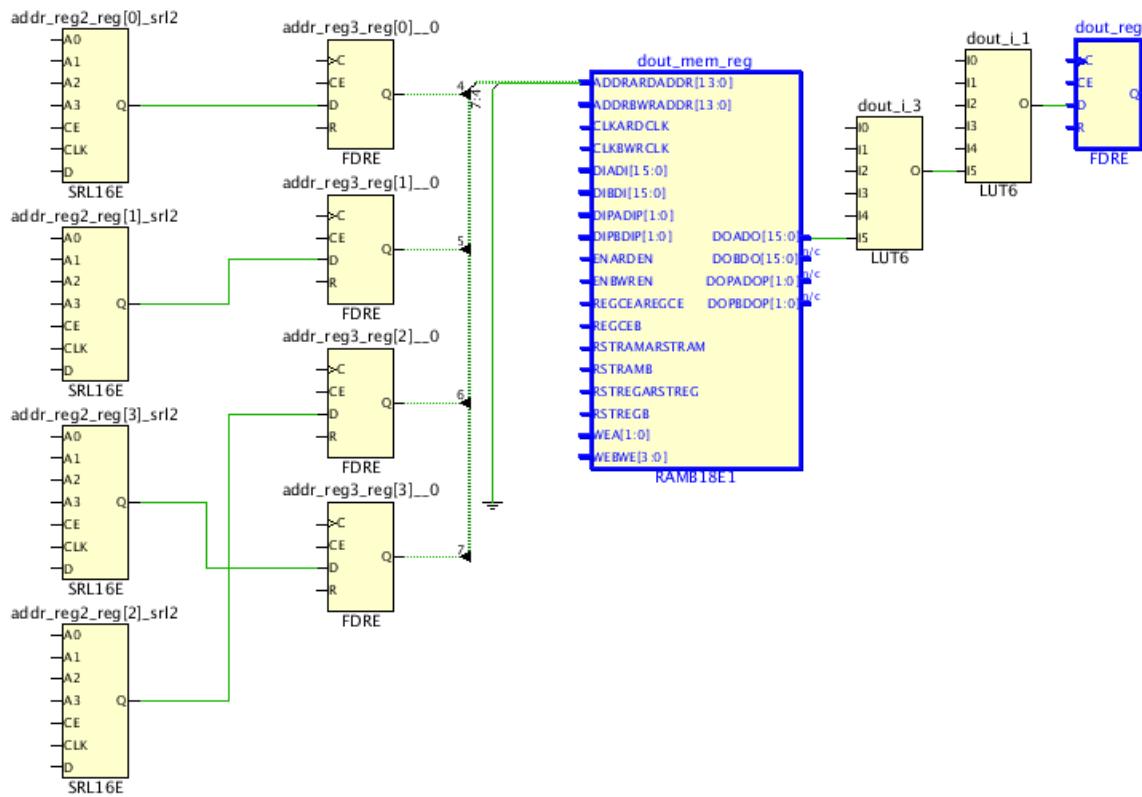
always@(posedge clk)
begin
    addr_reg0 <= addr;
    addr_reg1 <= addr_reg0;
    addr_reg2 <= addr_reg1;
    addr_reg3 <= addr_reg2;
end

always@(posedge clk)
begin
    dout_mem <= mem[addr_reg3];
    dout <= |dout_mem;
    dout_shift <= |addr_reg3;
end

endmodule
```

工具显示的对应以上 RTL 代码的关键路径如下图所示。

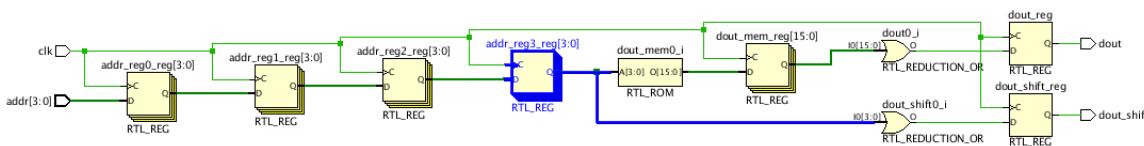
图 198：来自未启用输出寄存器的 RAMB 的关键路径



最好在综合后以及每个实现步骤后复查关键路径以识别需改进哪些逻辑组。对于长路径或者未能以最优方式利用 FPGA 硬件功能的任何路径，请返回 RTL 描述，尝试了解已综合的逻辑未实现最优化的原因，并修改代码以帮助综合工具改进网表。

Vivado 具有强大的嵌入式调试机制，可供您用于开始使用细化视图。细化视图有助于识别问题可能的来源，而无需手动搜索整个 RTL 代码。请参阅下图中所示对应上述 RTL 代码片段的细化视图。

图 199：RTL 代码片段的细化视图



细化视图提供了有关给定测试案例结构效率低下的有效提示。在此例中，问题来自地址寄存器扇出 (addr_reg3_reg)，它用于驱动内存地址和部分胶合逻辑（蓝色高亮）。

由综合工具执行的 RAMB 推断要求 RTL 代码中存在专用地址寄存器，这与当前地址寄存器扇出不兼容。由此导致综合工具对输出寄存器重定时以允许执行 RAMB 推断，而不是使用它来启用 RAMB 可选输出寄存器。

通过复制 RTL 代码中的地址寄存器，使用独立寄存器来驱动内存地址和互连逻辑 | FPGA 逻辑，即可在启用输出寄存器的情况下推断 RAMB。

手动复制后的 RTL 代码和细化视图如下图所示：

图 200：含已复制的地址寄存器的 RTL 代码

```

module test (input clk, input [3:0] addr, output reg dout, dout_shift);

(* rom_style = "block" *) reg [15:0] mem [0:15];

reg [3:0] addr_reg0;
reg [3:0] addr_reg1;
reg [3:0] addr_reg2;
reg [3:0] addr_reg3;
(* KEEP = "true" *) reg [3:0] addr_reg3_dup;

reg [15:0] dout_mem;

initial
begin
    $readmemh("init.txt",mem);
end

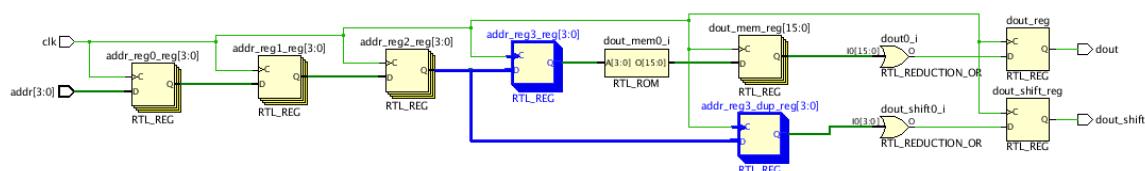
always@(posedge clk)
begin
    addr_reg0 <= addr;
    addr_reg1 <= addr_reg0;
    addr_reg2 <= addr_reg1;
    addr_reg3 <= addr_reg2;
    addr_reg3_dup <= addr_reg2;
end

always@(posedge clk)
begin
    dout_mem <= mem[addr_reg3];
    dout <= |dout_mem;
    dout_shift <= |addr_reg3_dup;
end

endmodule

```

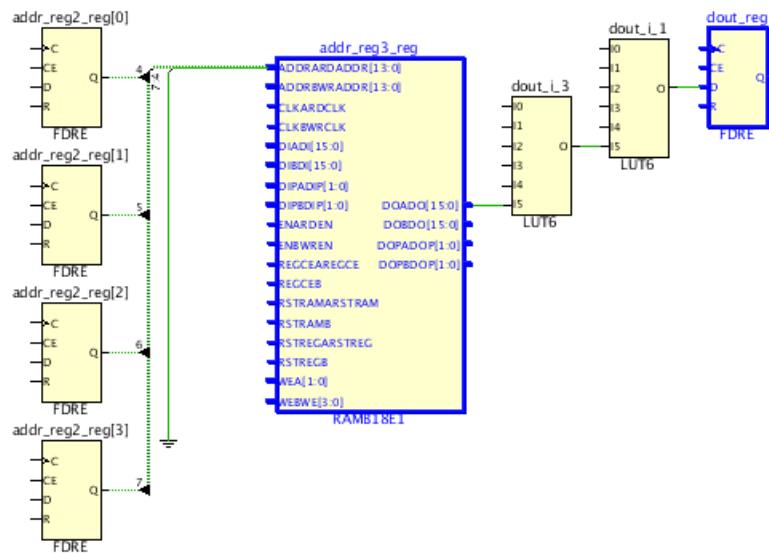
图 201：含已复制的地址寄存器的细化视图



已修改的 RTL 代码的关键路径如下图所示。请注意：

- `addr_reg2_reg` 寄存器已连接到块 RAM 的地址管脚。
- `addr_reg3_reg` 寄存器在块 RAM 中已被吸收。
- RAMB 输出寄存器已启用，由此显著降低了 RAMB 输出上的数据路径延迟。

图 202：已修改的 RTL 代码的关键路径



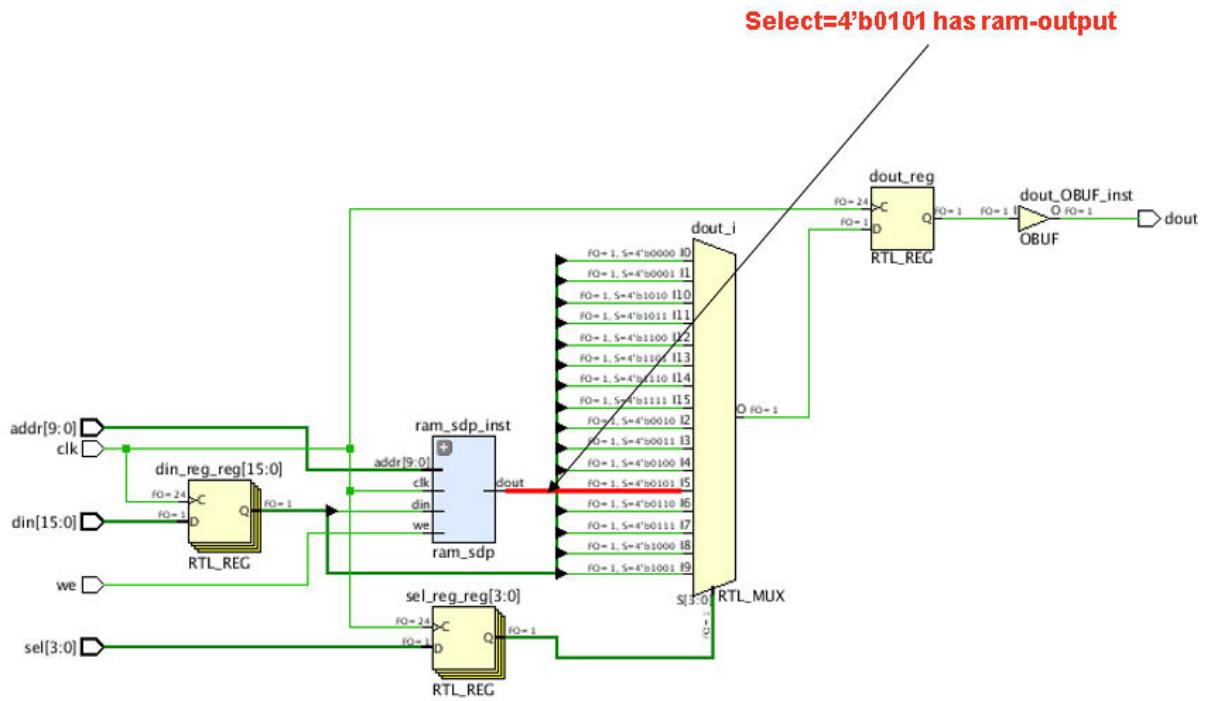
改进 RAMB 输出上的关键逻辑

以下测试用例重点提供了有关通过重构来对关键路径进行改进的信息，例如，将宏（块 RAM）推送到距离目标寄存器更近的位置的情况下。

下图显示了 1 个 16x1 多路复用器，其中仅含 1 个从块 RAM 到多路复用器的输入，其余输入由寄存器馈送。

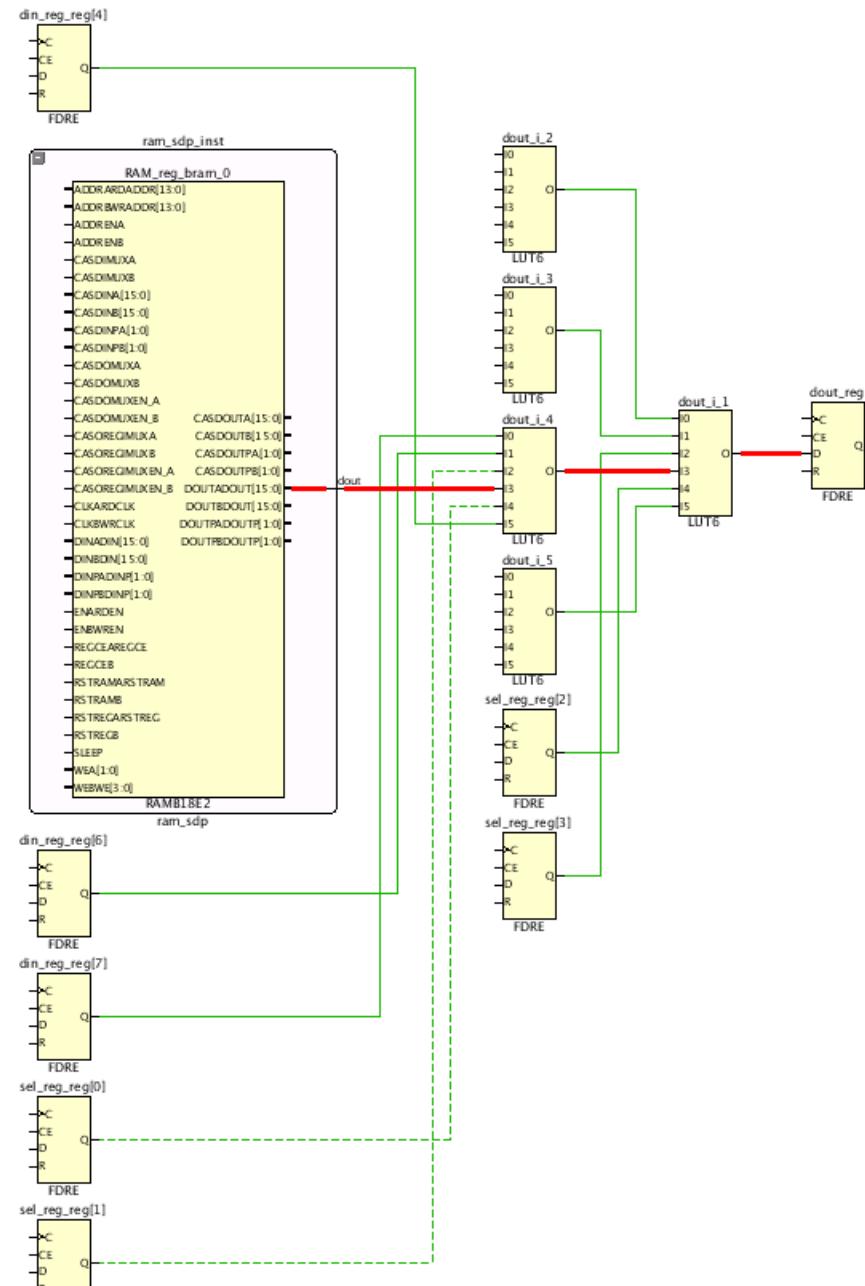
关键路径：块 RAM-> 2 个逻辑层次 -> FF。

图 203：连接到块 RAM 输出的 16x1 多路复用器



下图显示的关键路径中，以红色高亮显示块 RAM 到 FF 的路径。块 RAM->FF 和 FF->FF 都存在 2 个逻辑层次。由于块 RAM CLK->Q 延迟对于块 RAM 更高，因此 RAM->FF 为关键路径。

图 204：关键 RAMB-LUT-FF 路径



下一步，请注意下图中所示 RTL 代码，查看是否能够重构逻辑。

图 205：RTL 代码片段

```
| ram_sdp ram_sdp_inst (.clk(clk), .we(we), .addr(addr), .din(din_reg[5]), .dout(dout_from_ram));  
|  
.always@(posedge clk)  
.begin  
.sel_reg <= sel;  
.din_reg <= din;  
.case(sel_reg)  
. 4'd0: dout <= din_reg[0];  
. 4'd1: dout <= din_reg[1];  
. 4'd2: dout <= din_reg[2];  
. 4'd3: dout <= din_reg[3];  
. 4'd4: dout <= din_reg[4];  
. 4'd5: dout <= dout_from_ram;  
. 4'd6: dout <= din_reg[6];  
. 4'd7: dout <= din_reg[7];  
. 4'd8: dout <= din_reg[8];  
. 4'd9: dout <= din_reg[9];  
. 4'd10: dout <= din_reg[10];  
. 4'd11: dout <= din_reg[11];  
. 4'd12: dout <= din_reg[12];  
. 4'd13: dout <= din_reg[13];  
. 4'd14: dout <= din_reg[14];  
. 4'd15: dout <= din_reg[15];  
.endcase  
.end
```

重构逻辑的最佳方法是将 16x1 多路复用器拆分为 2 个多路复用器来重写上述代码片段。您可将选择值 4'd5 的条件豁免，将其用作为 2x1 多路复用器的启用条件（如下图所示），创建此级联多路复用器结构可生成含 3 个逻辑层次的 FF->FF；但块 RAM->FF 减少至 1 个逻辑层次。这样即可改进块 RAM->FF 路径，从而帮助下游工具实现更好的布局，因为 RAMB 布局比 LUT 和 FF 布局难度更高。总之，对于任意给定设计，减少宏原语（如 RAMB、FIFO 和 DSP）周围的长路径即可改进 QoR 结果。

图 206：用于减少 RAMB 输出逻辑级数的级联多路复用器结构

```
ram_sdp ram_sdp_inst (.clk(clk), .we(we), .addr(addr), .din(din_reg[5]), .dout(dout_from_ram));  
(* KEEP = "true" *) reg dout_nxt;  
  
always@*  
begin  
dout_nxt <= dout;  
case(sel_reg)  
  4'd0: dout_nxt <= din_reg[0];  
  4'd1: dout_nxt <= din_reg[1];  
  4'd2: dout_nxt <= din_reg[2];  
  4'd3: dout_nxt <= din_reg[3];  
  4'd4: dout_nxt <= din_reg[4];  
  //4'd5: dout_nxt <= dout_from_ram;  
  4'd6: dout_nxt <= din_reg[6];  
  4'd7: dout_nxt <= din_reg[7];  
  4'd8: dout_nxt <= din_reg[8];  
  4'd9: dout_nxt <= din_reg[9];  
  4'd10: dout_nxt <= din_reg[10];  
  4'd11: dout_nxt <= din_reg[11];  
  4'd12: dout_nxt <= din_reg[12];  
  4'd13: dout_nxt <= din_reg[13];  
  4'd14: dout_nxt <= din_reg[14];  
  4'd15: dout_nxt <= din_reg[15];  
endcase  
  
end  
  
always@(posedge clk)  
begin  
sel_reg <= sel;  
din_reg <= din;  
if(sel_reg == 4'd5)  
  dout <= dout_from_ram;  
else  
  dout <= dout_nxt;  
end
```

第 7 章

实现分析与收敛技巧

使用 report_design_analysis 命令

当难以实现时序收敛时或者在尝试提升应用的总体性能时，必须在运行综合后以及执行实现流程的任一步骤后审查设计的主要特性。高层次指标的收集相对较为简单，如时序汇总数值 (WNS/TNS/WHS/THS)

(report_timing_summary) 或各项资源利用率数值 (report_utilization、report_clock_utilization、report_high_fanout_nets 和 report_control_sets) 等。但分析和识别设计中具体某一方面对于特定时序路径的影响以及由此造成对总体结果质量 (QoR) 的影响则困难得多。QoR 分析通常要求您同时查看多个全局和局部特性，以确定设计和约束中哪些部分处于欠优化状态，或者哪些逻辑结构不适合目标器件架构和实现工具。

report_design_analysis 命令可用于收集逻辑、时序和物理特性，并合并展示在多个表中以便简化 QoR 根源分析。

注释：report_design_analysis 不会提供时序约束的完整性和正确性方面的报告。要验证时序约束，必须使用 check_timing 和 report_exceptions 命令以及 XDC 和 TIMING Methodology DRC。如需了解有关如何运行这些命令的更多信息，请参阅相应的章节：

- 时序汇总报告 (Report Timing Summary)
- 例外报告 (Report Exceptions)

常见 QoR 问题主要分为 2 类：

- 时序违例
- 拥塞 (Congestion)

时序违例

虽然分析和修复最差时序违例通常有助于提升总体 QoR，但您还必须复查其它关键路径，因为这些路径通常会增加时序收敛困难。您可使用以下命令来报告前 50 条最差的建立时序路径：

```
report_design_analysis -max_paths 50 -setup
```

下图显示了由此命令生成的“建立路径特性 (Setup Path Characteristics)”表格示例。

图 207：建立路径特性 (Setup Path Characteristics)

Paths	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Clock Relationship	Logic Levels	Routes	Logical Path
Path #1 10.000 9.022 0.223(3%) 8.799(97%) -0.498 0.066 Safely Timed 0 1 FDRE FDCE										
Path #2 10.000 9.079 0.223(3%) 8.856(97%) -0.496 0.070 Safely Timed 0 1 FDRE FDCE										
Path #3 10.000 9.079 0.223(3%) 8.856(97%) -0.496 0.070 Safely Timed 0 1 FDRE FDCE										
Path #4 10.000 9.189 0.223(3%) 8.966(97%) -0.385 0.070 Safely Timed 0 1 FDRE FDCE										
Path #5 10.000 9.189 0.223(3%) 8.966(97%) -0.385 0.070 Safely Timed 0 1 FDRE FDCE										
Path #6 10.000 9.156 0.223(3%) 8.933(97%) -0.385 0.070 Safely Timed 0 1 FDRE FDCE										
Path #7 10.000 9.156 0.223(3%) 8.933(97%) -0.385 0.070 Safely Timed 0 1 FDRE FDCE										
Path #8 10.000 8.921 0.223(3%) 8.698(97%) -0.525 0.071 Safely Timed 0 1 FDRE FDRE										
Path #9 10.000 8.899 0.223(3%) 8.676(97%) -0.524 0.071 Safely Timed 0 1 FDRE FDRE										
Path #10 10.000 8.899 0.223(3%) 8.676(97%) -0.524 0.071 Safely Timed 0 1 FDRE FDRE										
Path #11 10.000 8.936 0.223(3%) 8.713(97%) -0.510 0.072 Safely Timed 0 1 FDRE FDSE										
Path #12 10.000 8.936 0.223(3%) 8.713(97%) -0.510 0.072 Safely Timed 0 1 FDRE FDSE										
Path #13 10.000 8.982 0.223(3%) 8.759(97%) -0.440 0.072 Safely Timed 0 1 FDRE FDSE										
Path #14 10.000 9.015 0.223(3%) 8.792(97%) -0.499 0.072 Safely Timed 0 1 FDRE FDCE										

在该表中，可明确识别导致每条路径产生时序违例的各项特性：

- 逻辑延迟百分比过高（逻辑延迟 (Logic Delay)）
 - 逻辑层次是否过多？（逻辑级数 (Logic Levels)）
 - 是否存在阻碍逻辑最优化的任何约束或属性？（Don't Touch 或 Mark Debug）
 - 路径是否包含具有高逻辑延迟的单元，例如 RAMB 或 DSP 等？
 - 当前路径拓扑结构的路径要求是否过于苛刻？（要求 (Requirement)）
- 高信号线延迟百分比（信号线延迟 (Net Delay)）
 - 在路径中是否有任何高扇出信号线？（“高扇出 (High Fanout)” 或 “累积扇出 (Cumulative Fanout)”）
 - 分配给多个 Pblock 的单元布局能否拉开距离？(PBlock)
 - 单元布局能否拉开距离？（“边框大小 (Bounding Box Size)” 或 “时钟区域距离 (Clock Region Distance)”）
 - 对于 SSI 器件，是否存在跨 SLR 边界的信号线？（SLR 交汇 (SLR Crossings)）
 - 在布局看似正确的情况下，是否有 1 个或多个信号线延迟值远高于预期？请参阅[拥塞 \(Congestion\)](#) 部分。
- RAMB 或 DSP 单元中缺少流水线寄存器（而路径中存于此寄存器）
 - 检查路径，确认针对 RAMB 或 DSP 单元是否已启用流水线寄存器
- 高偏差（建立 <-0.5 ns，保持 > 0.5 ns）（时钟偏差 (Clock Skew)）
 - 此路径是时钟域交汇路径吗？（“起点时钟 (Start Point Clock)” 和 “端点时钟 (End Point Clock)”）
 - 时钟是同步时钟还是异步时钟？（时钟关系 (Clock Relationship)）
 - 此路径是否跨多个 I/O 列？（IO 交汇 (IO Crossings)）

为了在赛灵思 Vivado® IDE 中直观显示时序路径及其布局/布线的详细信息，您必须使用以下命令：

```
report_timing -max_paths 50 -setup -input_pins -name worstSetupPaths
```

这些路径按裕量排序，并按“Setup Path Characteristics”表中的相同顺序（如上图所示）显示。

report_design_analysis 还可为最差的 1000 条路径生成“逻辑层次分布 (Logic Level Distribution)”表，以供您用于识别设计中存在的长路径。通常最长的路径首先由布局器加以最优化以满足时序要求，这可能导致较短的路径的布局质量劣化。您必须不断尝试消除较长的路径，以提高整体 QoR。下图显示了仅含 1 个时钟的设计的“Logic Level Distribution”示例。

图 208：“逻辑层次分布 (Logic Level Distribution)” 表

2. Logic Level Distribution	
<hr/>	
+-----+	
End Point Clock Requirement 0 1 2 3 4 5 6 7 8 9 10 11-15 16-20 21-25 26-30 31+	
+-----+	
cpuClk_5 8.000ns 0	
phyClk0_2 8.000ns 0 1 7 140 138 8 198 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
phyClk1_1 8.000ns 0 12 3 93 4 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
usbClk_3 8.000ns 0 10 1 2 13 4 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
+-----+	
* Columns represents the logic levels per end point clock	
** Distribution is for top worst 1000 paths	

根据结果，可通过更改 RTL 或使用不同综合选项来改进网表，或者也可以修改时序约束和物理约束。

拥塞 (Congestion)

report_design_analysis 命令用于报告多个拥塞表，其中显示布局器和布线器发现的拥塞区域。您可在运行布局器和布线器的 Vivado 工具会话内使用以下命令来生成这些表：

```
report_design_analysis -congestion
```

下图显示的拥塞表示例对应于布局器最终拥塞和布线器初始拥塞。

图 209：估算的拥塞表

1. Placer Final Level Congestion Reporting															
Direction	Level	Congestion	Window	Cell Names	Combined LUTs	Avg LUT Input	LUT	LUTRAM	Flop	MUXF	RAMB	URAM	DSP	CARRY	SRL
North	5	118%	(CLEL_R_X44Y659, CLEM_X59Y690)	core_top/crc_top8(58%)	1%	3.94518	73%	33%	17%	18%	42%	NA	0%	5%	0%
East	1	109%	(CLEM_X52Y295, CLEL_R_X52Y296)	core_top/crc_top9(100%)	0%	4.71875	93%	0%	67%	7%	NA	NA	NA	0%	0%
South	5	111%	(CLEM_X44Y658, CLEL_R_X58Y689)	core_top/crc_top8(61%)	1%	3.98173	74%	31%	17%	18%	41%	NA	0%	6%	0%
West	3	104%	(CLEM_X46Y292, CLEL_R_X49Y299)	core_top/crc_top0(100%)	0%	3.91211	73%	0%	73%	9%	NA	NA	NA	9%	0%

2. Router Initial Congestion															
Direction	Type	Level	Window	Combined LUTs	Avg LUT Input	LUT	LUTRAM	Flop	MUXF	RAMB	URAM	DSP	CARRY	SRL	Cell Names
North	Global	5	(CLEM_X44Y630, CLEM_X59Y709)	1%	4.41	89%	16%	18%	19%	68%	NA	0%	4%	0%	core_top/crc_top8(58%)
North	Long	6	(CLEM_X39Y1618, CLEM_X70Y729)	1%	4.33	85%	11%	18%	19%	59%	0%	0%	5%	0%	core_top/crc_top9(59%)
North	Long	6	(CLEM_X39Y634, CLEM_X70Y729)	1%	4.30	85%	12%	18%	19%	59%	0%	0%	5%	0%	core_top/crc_top9(56%)
South	Long	6	(CLEM_X44Y654, CLEM_X75Y717)	1%	4.10	85%	18%	18%	18%	61%	0%	0%	6%	0%	core_top/crc_top9(68%)

针对“模块名称 (Module Names)”所提供的名称对应于每个报告的拼块内存在的层级单元。您可使用以下命令来检索完整名称：

```
get_cells -hier <moduleName>
```

确认拥塞区域内存在的层级单元后，即可使用拥塞缓解技巧来尝试减少总体设计拥塞。

识别设计中最长的逻辑延迟路径

时序路径对应于设计中的逻辑路径。其延迟是单元延迟和信号线延迟的累积值。Vivado® 综合和实现工具由时序驱动，用于对整个编译流程中设计的最差违例路径进行最优化。如果某条路径的累积单元延迟等于或高于时序要求（例如，常见于路径的时钟周期），那么设计在实现后满足时序的概率较低。分析逻辑延迟比简单计算逻辑级数更有效，因为它可显示最差路径在受到估算或已布线的信号线延迟影响前的状态。此分析可生成布局布线前的最差时序路径列表（不含信号线延迟）。

识别在时序方面（而不一定是逻辑级数方面）最差的路径至关重要。例如，未寄存的块 RAM 的时钟输出 (clock to out) 延迟可能极大，而一连串进位链可能具有多个逻辑层次，且每个逻辑层次都含少量延迟。您必须在实现前仔细分析这些路径。这些较长的延迟路径分 3 种典型类别：

- 不使用嵌入式输出寄存器的块 RAM
- 未流水线化的 DSP48
- 较长的逻辑路径

识别这些长路径的最有效方法是综合后运行时序报告，并将布线估算设置为 none。这可通过在 Vivado IDE 的“Timing Report”对话框的“Timer Settings”选项卡中将“互连模型 (Interconnect model)”更改为“无 (none)”，或者在 Tcl 控制台或 shell 中使用以下 Tcl 命令来实现：

```
set_delay_model -interconnect none
```

复查时序结果以识别任何失败路径。如有路径无法满足不含任何布线延迟的时序要求，这些路径将无法通过实际布线来满足时序。这些路径的问题必须立即解决。通常，此类问题必须在 RTL 中修复，但违例也可能源于综合属性缺失或时序约束错误。实现更改后，设计将产生充足的裕量，如下图所示。

图 210：含 0 互连的时序报告

Name	Slack	^1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock	Dest
Constrained Paths (12)														
phyClk1_1 (10)														
↳ Path 121	1.767	1	1	VStat..._i[7]	usbE...J/D	0.823	0.823	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 122	1.786	1	1	LineS..._i[0]	usbE...J/D	0.801	0.801	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 123	1.786	1	1	VStat..._i[3]	usbE...J/D	0.792	0.792	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 124	1.788	1	1	VStat..._i[6]	usbE...J/D	0.803	0.803	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 125	1.795	1	1	LineS..._i[1]	usbE...J/D	0.825	0.825	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 126	1.796	1	1	VStat..._i[0]	usbE...J/D	0.804	0.804	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 127	1.801	1	1	VStat..._i[5]	usbE...J/D	0.799	0.799	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 128	1.809	1	1	VStat..._i[2]	usbE...J/D	0.792	0.792	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 129	1.811	1	1	VStat..._i[4]	usbE...J/D	0.789	0.789	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ Path 130	1.813	1	1	VStat..._i[1]	usbE...J/D	0.787	0.787	0.000	100.0	0.0	8.000	sysClk	phyC	
↳ phyClk0_2 (10)														
↳ cpuClk_5 (10)														

识别高扇出信号线驱动

高扇出信号线常常导致设计实现问题。随着每个 FPGA 系列的裸片大小不断增大，扇出问题也不断增加。具有数千个端点的信号线上的时序通常难以得到满足，如果在路径上存在附加逻辑，或者如果这些信号线是由非时序单元（例如 LUT 或分布式 RAM）驱动的，则尤其如此。

设计师经常通过在特定信号线上使用全局扇出限制或 MAX_FANOUT 属性来解决 RTL 或综合中的高扇出信号线问题。物理最优化 (phys_opt_design) 可基于裕量和布局信息自动复制高扇出信号线驱动，通常可显著改善时序。赛灵思建议您使用结构寄存器 (FD*) 来驱动高扇出信号线，通常在物理最优化期间，结构寄存器较易于复制和重新定位。在综合后以及物理最优化后，查看高扇出信号列表至关重要。用于识别这些信号线的命令为 report_high_fanout_nets。

生成报告后，即可查看穿过高扇出信号线和对应原理图的时序。此报告不会将时钟列为高扇出驱动因素。如果在“驱动类型 (Driver Type)”列中包含 BUFG，则表明此 BUFG 正在驱动逻辑并且可能同时驱动时钟管脚。

```
### Report the high fanout net
report_high_fanout_nets -load_types -max_nets 100
### Report timing through specific high fanout net
report_timing -through [get_nets I_GLOBAL_RST_N_i] -name high_fanout_1
```

以下是通过 phys_opt_design 成功减少扇出的设计示例：

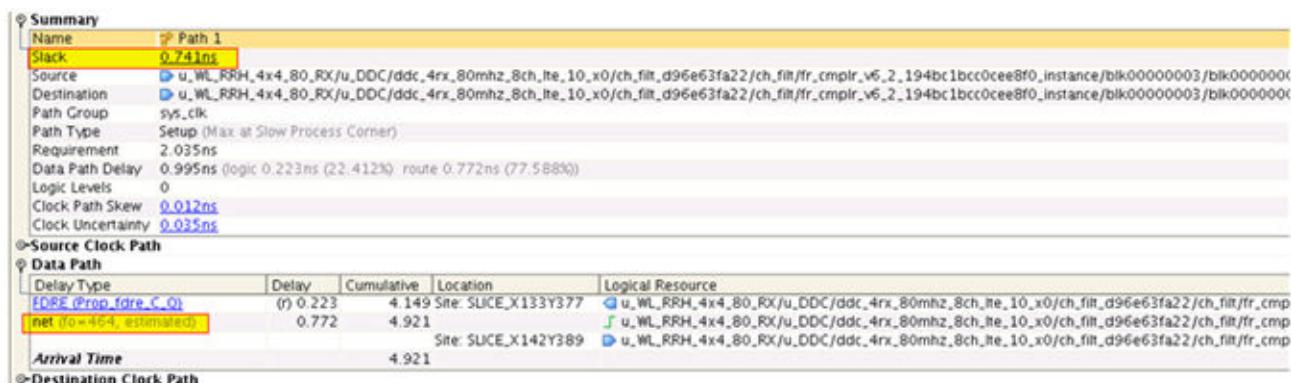
布局后检查点：report_high_fanout_nets

Fanout	Driver Type	Net Name
2945	FDRE	u_WL_RRH_4x4_80_RX/u_DDC/ddc_4rx_80mhz_8ch_lte_10_x0/ch_filt_d96e63fa22/ch_filt/fr_cmplr_v6_2_194bc1bcc0cee8f0_instance/blk00000003/blk00000000

提示：通过将 -timing 和 -load_types 选项与 report_high_fanout_nets 命令搭配使用，还可显示高扇出信号线的延迟和各种负载类型。

物理最优化后，该信号线的“时序报告 (Timing Report)”为：

图 211：时序报告示例



该特定信号线上的扇出从 2945 降低到 464。更重要的是，降低扇出使时序得以改善（在此特定路径上，改善幅度超过 1 ns）。

每个信号线的 FLAT_PIN_COUNT 属性都可指示在整个设计层级中连接到此信号线的叶节点单元数量。使用 get_property 命令可提取 FLAT_PIN_COUNT 属性：

```
get_property FLAT_PIN_COUNT [get_nets my_hfn]
```

提示：您可使用 Tcl 脚本来为穿过任意特定高扇出信号线传输的路径创建额外的报告。

判断保持修复对设计是否存在负面影响

Vivado Design Suite 布线器认为保持时间的修复优先级高于建立时间。这是因为实验室内的设计即使不满足建立时间，只要差距较小则仍可能有效。并且始终可以选择降低时钟频率。但如果存在保持时间违例，则设计几乎不可能正常运行。

大部分情况下，布线器可在不影响建立时间的情况下满足保持时序要求。在某些情况下（主要由于设计或约束中存在的错误），建立时间会受到显著影响。通常导致保持检查错误的原因主要是 `set_multicycle_path` 约束错误（未指定 `-hold`）。其它情况下，保持时间要求过高则是由时钟偏差过大而导致的。在此情况下，赛灵思建议您复查此特定电路的时钟设置架构。欲知详情，请访问[此链接](#)以参阅《UltraFast 设计方法指南（适用于 Vivado Design Suite）》[\(UG949\)](#) 中的相应内容。

如果设计布局后可满足建立时序要求，但布线后不满足建立时间，则可能出现此问题。您可使用 `report_design_analysis` 命令搭配 `-show_all` 选项来查看由于将布线器所添加的绕行布线到修复保持违例所导致的路径延迟。下图显示了 `report_design_analysis` 报告示例中包含“保持修复绕行 (Hold Fix Detour)”列，用于指示布线器由于保持时间修复而添加到时序路径中的延迟 (ps)。

图 212：含“Hold Fix Detour”的“Report Design Analysis”

Paths	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Hold Fix Detour	Combin
Path #1	2.034	1.813	0.134(8%)	1.679(92%)	-0.179	-0.118		467
Path #2	2.034	1.813	0.134(8%)	1.679(92%)	-0.179	-0.118		467
Path #3	2.034	1.813	0.134(8%)	1.679(92%)	-0.179	-0.118		467
Path #4	2.034	1.813	0.134(8%)	1.679(92%)	-0.179	-0.118		467
Path #5	2.034	1.853	0.135(8%)	1.718(92%)	-0.256	-0.117		567
Path #6	2.034	1.811	0.134(8%)	1.677(92%)	-0.179	-0.117		466

 提示：分析估算的布局后保持时序，并识别任何不寻常的保持时间严重违例（超过 500ps）。

如果怀疑保持修复影响时序收敛，可使用以下任一方法来判断：

- [方法 1：不含保持修复情况下的布线](#)
- [方法 2：在最差建立时间失败路径上运行 report_timing -min](#)

方法 1：不含保持修复情况下的布线

1. 将布局后检查点读取到 Vivado Design Suite 中。
2. 添加约束以禁用所有保持检查：

```
set_false_path -hold -to [all_clocks]
```

 注意！此约束仅用于测试。切勿针对将投入量产或者将交付给其它设计人员的设计执行此操作。必须在量产设计前移除此约束。

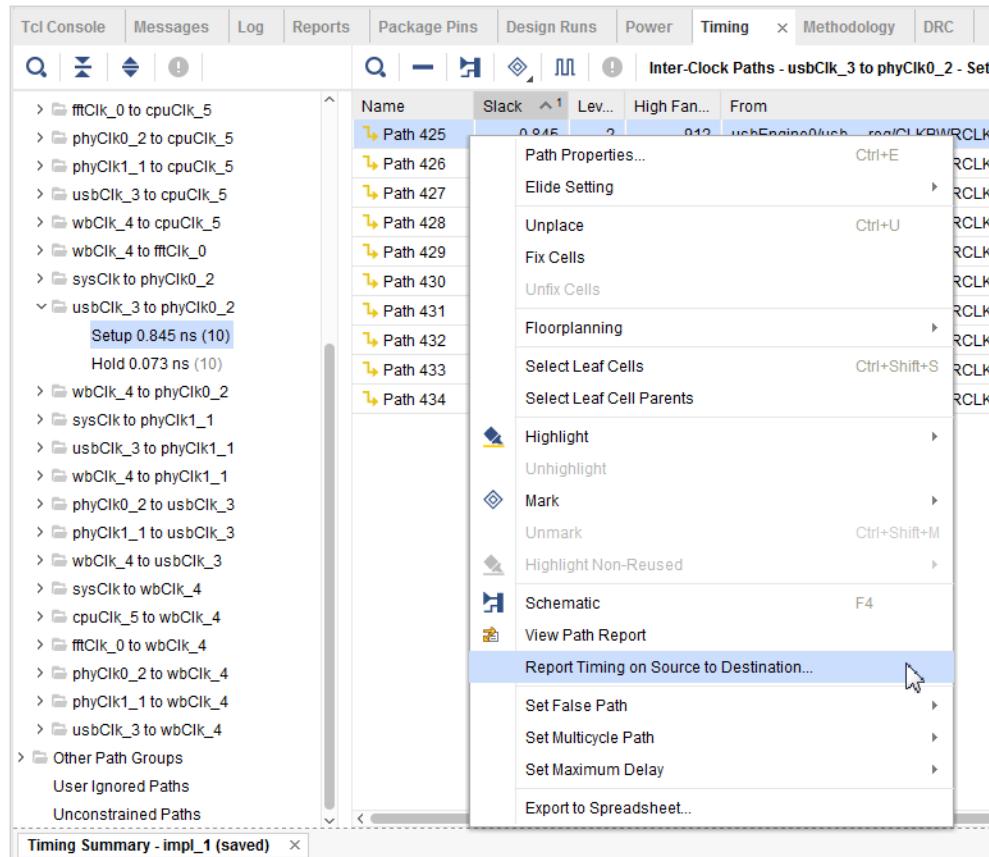
3. 请运行 `route_design` 和 `report_timing_summary`。

如果含保持时间检查的 WNS 与不含该检查的 WNS 之间存在明显差异，则表明保持违例可能过大，而建立路径正受到影响。

方法 2：在最差建立时间失败路径上运行 report_timing -min

请复查该路径的保持时间以判定最差建立时间失败路径是否是由于保持时间修复所导致的。在 Vivado IDE 中，右键单击并单击“Report Timing on Source to Destination”。与执行建立时序分析相反，查看保持时序至关重要。获得保持时间报告后，请验证要求，确保在路径上未添加额外延迟以满足保持时间要求。

图 213：在特定路径上运行“时序 (Timing)”报告



快速分析所有失败路径

`report_timing_summary` 命令是可用于确定设计的所有时序信息的强大工具。有时通过单一报告查看所有失败路径是很有效的。以下命令可从命令行或从 IDE 内部运行。

```
report_timing -max_paths 100 -slack_less_than 0 -name worse_100_setup
```

提示：使用 IDE 时，您可将时序结果导出到电子表格，以便对失败路径执行更全面的分析。

以上命令用于报告前 100 条失败路径。如果现有失败路径不足 100 条，由于使用了 `-slack_less_than 0` 选项，因此仅报告失败路径。通过单一列表来查看失败路径有助于识别失败路径的量级差异。

例如，WNS 可能为 -3 ns，影响少量路径，但列表中下一个 WNS 可能达到 -300 ps 甚至更好。

默认情况下，分析时序失败时，每个端点可看到一条最差时序路径。通常有通病的端点包含大量相似路径。

可将 `-nworst` 选项与 `report_timing` 命令搭配使用，以便复查单一端点的所有最差路径。例如，运行以下命令以查看指向最差情况失败端点的所有路径（假定路径数量少于 100）：

```
report_timing -max_paths 100 -nworst 100
```

复查所有最差路径可能会产生大量数据。为最大限度减少需分析的数据量，可将 `-unique_pins` 选项与 `report_timing` 命令搭配使用，以便仅查看路径中独特的部分。这样即可为通过时序路径的每种唯一管脚组合提供单一路径。例如：

```
report_timing -max_paths 100 -nworst 100 -unique_pins
```

布局规划

本节将探讨布局规划，具体包括：

- [关于布局规划](#)
- [了解布局规划基础知识](#)
- [使用基于 Pblock 的布局规划](#)
- [将特定逻辑锁定到器件站点](#)
- [对堆叠硅片互联 \(SSI\) 器件进行布局规划](#)

关于布局规划

布局规划有助于设计满足时序要求。当设计难以始终如一满足时序要求或者从未满足时序要求时，赛灵思建议您执行布局规划。如果您与设计团队协作并且协作过程中一致性至关重要，那么布局规划同样可以发挥作用。

布局规划可通过减少平均布线延迟来改进建立时间裕量（TNS 和 WNS）。在实现期间，时序引擎致力于解决最差情况建立时间违例和所有保持时间违例。布局规划只能改进建立时间裕量。

当网表采用层级结构时，手动布局规划最为简单。如果综合将整个网表扁平化，那么设计分析会明显变慢。请将综合设置为生成层级网表。对于 Vivado 综合，请使用：

- `synth_design -flatten_hierarchy rebuilt`
- 或
- Vivado 综合默认策略

含交错逻辑路径的大型层级块可能分析难度较大。在较低的次级层级中采用独立逻辑结构的设计更便于分析。最好寄存层级模块的所有输出。走线穿过多个层级块的路径布局分析难度较大。

了解布局规划基础知识

并非每项设计都能始终满足时序。您可能需要为工具提供解决方案指南。布局规划支持您引导工具完成高层次层级布局或详细门电路布局。

您将通过修复最严重的问题或者最常见的问题来最大程度改进结果。例如，如果存在离群路径并且这些路径的裕量明显极差或者具有高层次的逻辑，那么首先需修复这些路径。“Reports” → “Timing” → “Create Slack Histogram”选项可提供离群路径的视图。或者，如果在多个负时序裕量路径中出现相同的时序端点，那么改善其中一条路径可能可为该端点上其它路径实现相同的改善效果。

可考虑利用布局规划通过减少布线延迟或者增加非关键块上的逻辑密度来提高性能。逻辑密度是对应芯片上的逻辑封装紧密程度的指标。

布局规划可帮助您满足更高的时钟频率要求并提升结果的一致性。有多种方法适用于布局规划，每种方法都各有优缺点。

详细的门级布局规划

详细的门级布局规划涉及在器件上的特定站点内对个别叶节点单元进行布局。

详细的门级布局规划的优势

- 详细的门级布局规划适用于手动布线的信号线。
- 详细的门级布局规划可以最大限度发挥器件性能。

详细的门级布局规划的劣势

- 详细的门级布局规划较为耗时。
- 详细的门级布局规划需要具备有关器件和设计的广泛知识。
- 如果网表发生变更，详细的门级布局规划可能需重做。



建议：请将门级布局规划作为最终手段来使用。

信息复用

复用来自满足时序的设计的信息。如果设计无法始终如一满足时序，请使用此流程。要复用信息，请执行以下操作：

1. 打开 2 个实现运行：
 - a. 1 个对应满足时序的运行。
 - b. 另 1 个对应不满足时序的运行。
2. 查看 2 项设计之间的区别。
 - a. 通过 `report_timing_summary` 识别部分失败的时序路径。
 - b. 在满足时序的设计上，以 `min_max` 模式运行 `report_timing` 以对满足时序的设计上的路径进行定时。
3. 比较时序结果：
 - a. 时钟偏差
 - b. 数据路径延迟
 - c. 布局
 - d. 布线延迟



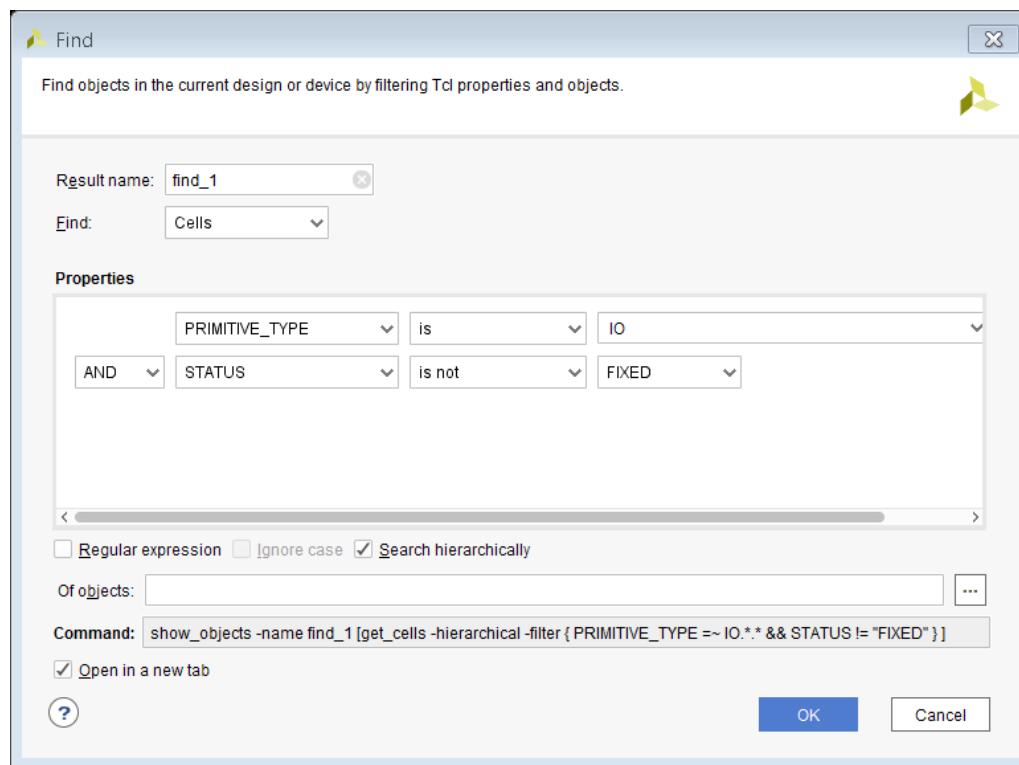
提示：在有多台显示器的计算机上，选择“在新窗口中打开实现 (Open Implementation in New Window)”以在新窗口中打开设计。

4. 如果路径端点之间的逻辑延迟量存在差异，请重新执行综合运行。

复查 I/O 和单元布局

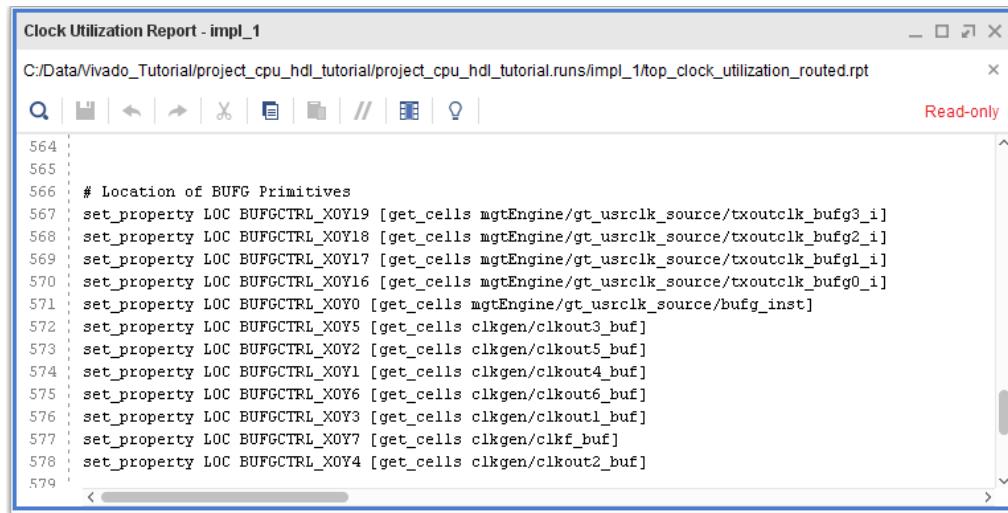
复查设计中的单元布局。比较 2 份 I/O 报告，以复查 I/O 布局和 I/O 标准。确保所有 I/O 均已完成布局。通过简单搜索可以发现不含固定布局的所有 I/O，如下图所示。

图 214: I/O 未固定



如果不同运行间的时钟偏差发生改变，请考虑复用来自满足时序的运行的时钟原语布局。“时钟利用率 (Clock Utilization)” 报告可列出时钟树驱动的布局，如下图所示。

图 215：时钟位置



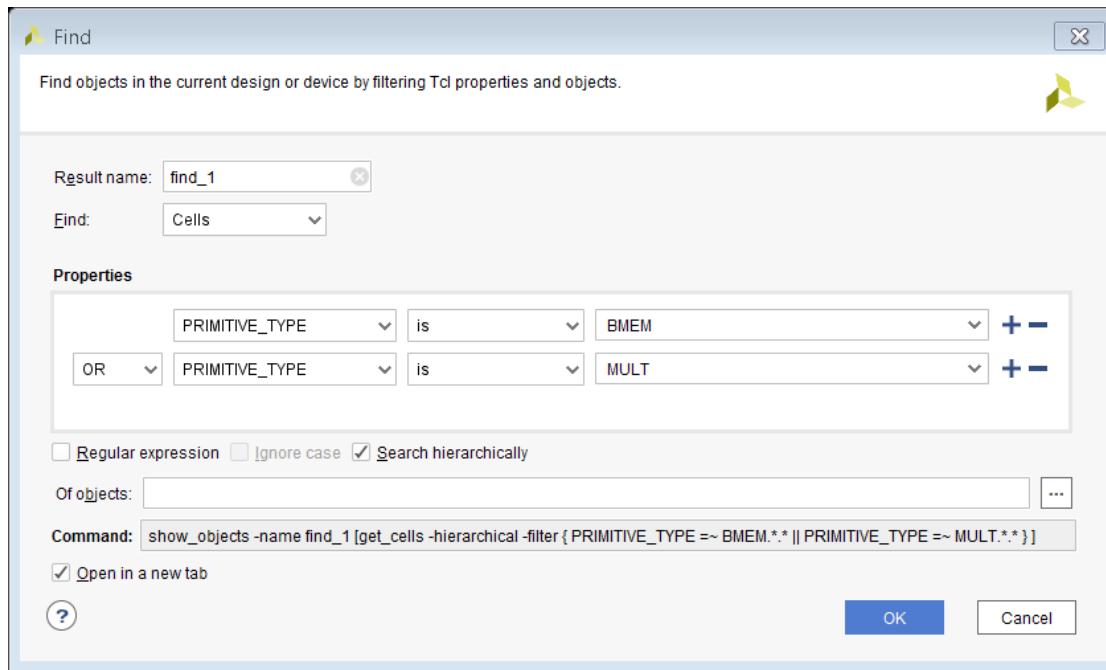
```
Clock Utilization Report - impl_1
C:/Data/Vivado_Tutorial/project_cpu_hdlTutorial/project_cpu_hdlTutorial.runs/impl_1/top_clock_utilization_routed.rpt
Read-only

564
565
566 # Location of BUFQ Primitives
567 set_property LOC BUFGCTRL_XOY19 [get_cells mgtEngine/gt_usrclk_source/txoutclk_bufg3_i]
568 set_property LOC BUFGCTRL_XOY18 [get_cells mgtEngine/gt_usrclk_source/txoutclk_bufg2_i]
569 set_property LOC BUFGCTRL_XOY17 [get_cells mgtEngine/gt_usrclk_source/txoutclk_bufg1_i]
570 set_property LOC BUFGCTRL_XOY16 [get_cells mgtEngine/gt_usrclk_source/txoutclk_bufg0_i]
571 set_property LOC BUFGCTRL_XOY0 [get_cells mgtEngine/gt_usrclk_source/bufg_inst]
572 set_property LOC BUFGCTRL_XOY5 [get_cells clkgen/clkout3_buf]
573 set_property LOC BUFGCTRL_XOY2 [get_cells clkgen/clkout5_buf]
574 set_property LOC BUFGCTRL_XOY1 [get_cells clkgen/clkout4_buf]
575 set_property LOC BUFGCTRL_XOY6 [get_cells clkgen/clkout6_buf]
576 set_property LOC BUFGCTRL_XOY3 [get_cells clkgen/clkout1_buf]
577 set_property LOC BUFGCTRL_XOY7 [get_cells clkgen/clkf_buf]
578 set_property LOC BUFGCTRL_XOY4 [get_cells clkgen/clkout2_buf]
```

LOC 约束可轻松复制到 XDC 约束文件中。

许多设计都已通过复用块 RAM 和 DSP 布局来满足时序。请选择“Edit > Find”以列出实例。

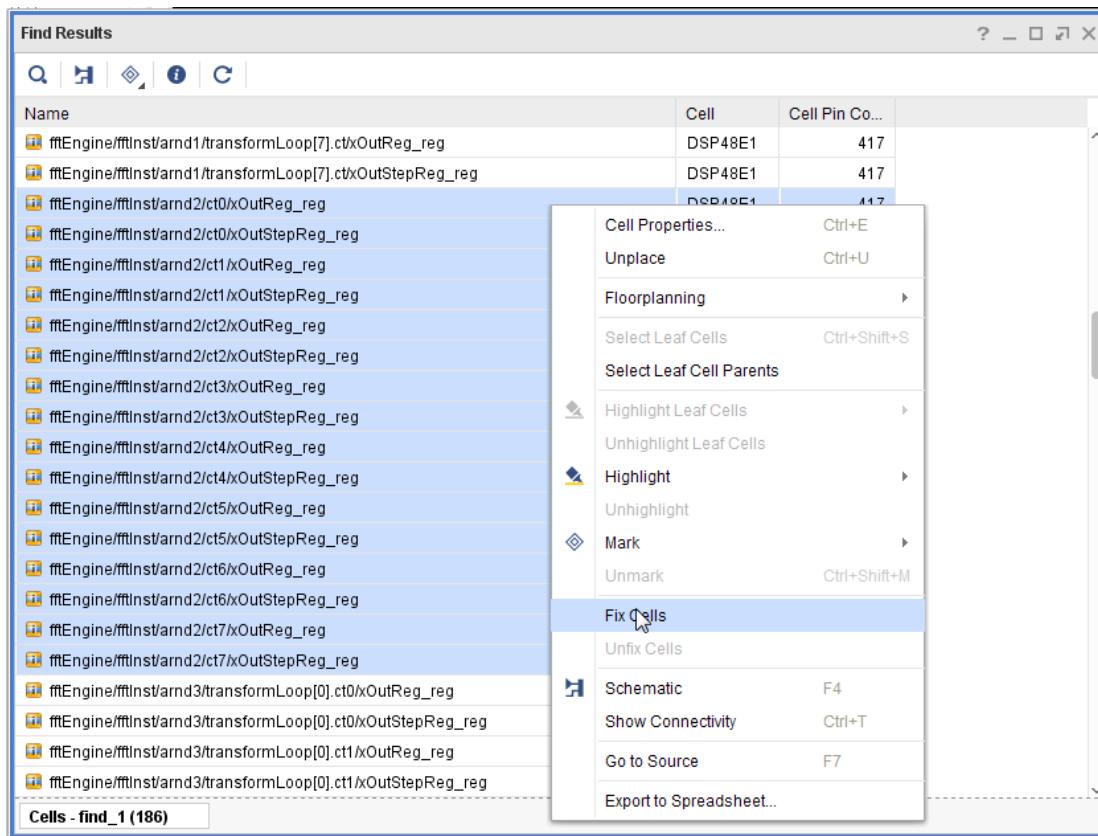
图 216：DSP 或 RAM



添加布局约束

固定逻辑以便向 XDC 添加布局约束。

- 从查找结果中选择宏。
- 右键单击并选择“Fix Cells”（如下图所示）。



建议：在固定布局前，基于层级名称分析布局并将其高亮。

复用布局

I/O、全局时钟资源、块 RAM 宏和 DSP 宏布局十分便于复用。复用此布局有助于减少各网表版本之间的结果差异。这些原语通常具有稳定的名称。布局通常易于维护。



提示：请勿复用通用 slice 逻辑的布局。请勿复用设计中可能更改的部分的布局。

对增量编译复用布局

增量编译支持复用来自上一次运行的布局布线数据。只需在 `place_design` 前引用现有已布局或已布线的 DSP 即可。可复用完整设计、任一层级或单元类型（如，DSP 或块 RAM）。增量编译还可自动处理对设计某些部分执行的更改。

如需了解更多信息，请参阅《Vivado Design Suite 用户指南：实现》(UG904)。

布局规划技巧

对于从未满足时序的设计以及不适合更改网表或约束的设计，可考虑采用门级布局规划。



建议：在考量门级布局规划前，请尝试层次化布局规划。

层次化布局规划

层次化布局规划支持您将 1 个或多个层级布局在片上某个区域内。此区域可向布局器提供全局层面的指导信息，并由布局器执行详细布局。层次化布局规划相比于门级布局规划具有如下优势：

- 层次化布局规划的创建速度比门级布局规划更快。良好的布局规划可改善时序。布局规划不受设计变更影响。
- 层级可充当所有门电路的容器。一般即使网表发生更改，它仍可正常运作。

在层次化布局规划中支持：

- 识别包含关键路径的较低层级。
- 使用顶层布局规划来识别布局位置。
- 通过实现来执行个别单元布局。
- 全面掌握单元和时序路径信息。
- 通常能够有效完成高精度布局。

手动单元布局

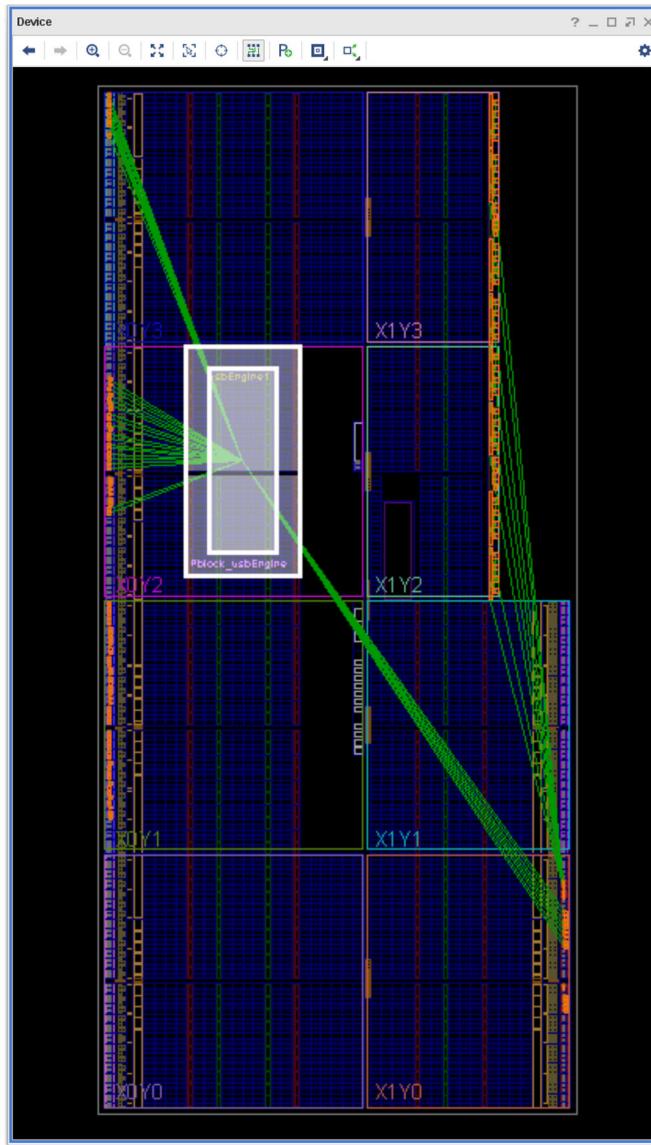
手动单元布局可充分发挥器件性能。使用此技巧时，设计师通常只能将其用于小块设计。可将少量逻辑布局到高速 I/O 接口周围，或者可手动布局块 RAM 和 DSP。手动布局可能比较慢。

所有布局规划技巧都需要大量工程设计时间。可能需要布局规划迭代。如果任意单元名称发生更改，则必须更新布局规划约束。

执行布局规划时，应明确最终管脚输出。最好将 I/O 固定。I/O 可提供锚点作为布局规划的起点。与 I/O 通信的逻辑可向固定管脚移植。

 提示：将与 I/O 通信的块布局到其 I/O 附近。如果管脚输出将块拆分，请考虑修改管脚输出或 RTL。

图 217：将设计拆分的 I/O 组件



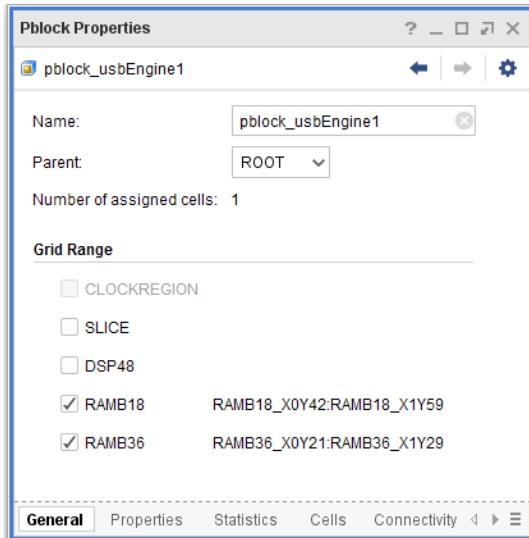
上图所示布局规划可能对时序没有帮助。请考虑将块拆分、更改源代码或者仅约束块 RAM 和 DSP。另请考虑在外部时序要求允许的情况下取消 I/O 寄存器布局。

本部分中提到的 Pblock 以 XDC 约束来表示：

```
create_pblock Pblock_usbEngine
add_cells_to_pblock [get_pblocks Pblock_usbEngine] [get_cells -quiet [list
usbEngine1]]
resize_pblock [get_pblocks Pblock_usbEngine] -add
{SLICE_X8Y105:SLICE_X23Y149}
resize_pblock [get_pblocks Pblock_usbEngine] -add {DSP48_X0Y42:DSP48_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add
{RAMB18_X0Y42:RAMB18_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add
{RAMB36_X0Y21:RAMB36_X1Y29}
```

第 1 行用于创建 Pblock。第 2 行 (add_cells_to_pblock) 用于将层级分配到 Pblock。有 4 种资源类型 (SLICE、DSP48、RAMB18 和 RAMB36)，各有自己的网格。不受网格约束的逻辑可连接至器件内任意位置。要仅约束层级内的块 RAM，请禁用其它 Pblock 网格。

图 218：Pblock 网格



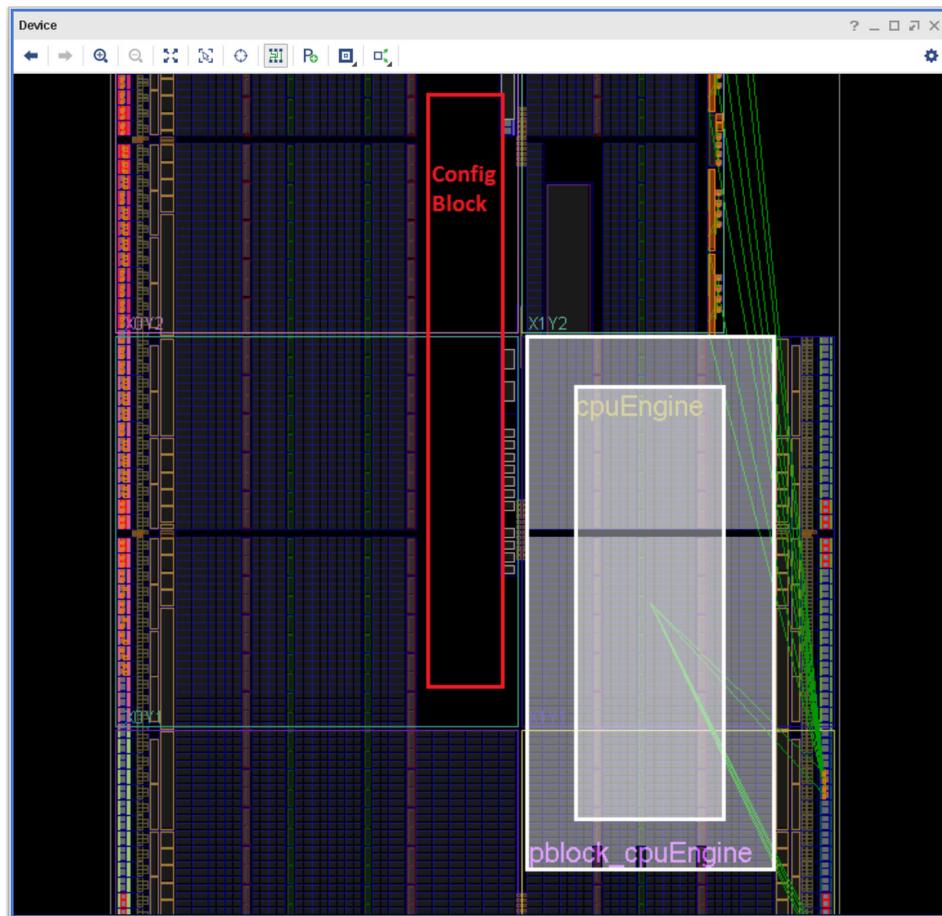
生成的 XDC 命令可用于定义简化的 Pblock：

```
create_pblock Pblock_usbEngine
add_cells_to_pblock [get_pblocks Pblock_usbEngine] [get_cells -quiet [list
usbEngine1]]
resize_pblock [get_pblocks Pblock_usbEngine] -add
{RAMB18_X0Y42:RAMB18_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add
{RAMB36_X0Y21:RAMB36_X1Y29}
```

块 RAM 约束在器件内，但 slice 逻辑可自由布局在器件上任意位置。

 提示：布局 Pblock 时，请注意避免层级布局规划跨越中心配置块。

图 219：避开配置块



使用基于 Pblock 的布局规划

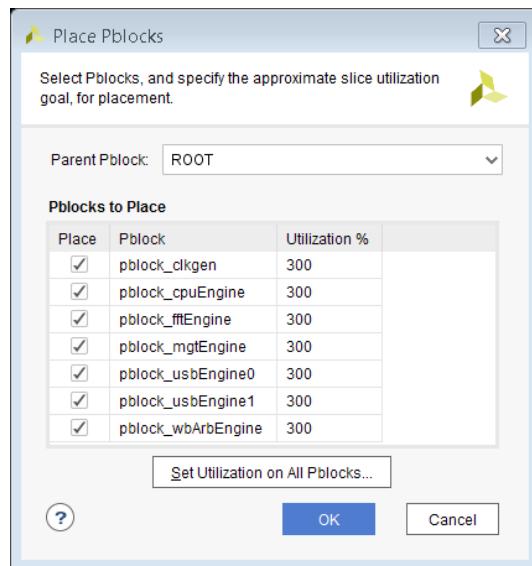
将 RTL 集成到设计中时，有助于将器件内部的设计可视化。以图形化方式查看块之间的互连方式以及综合后的 I/O 管脚输出 (pinout) 有助于您了解自己的设计。

要查看互连方式，请在上层层级内使用 Pblock 生成顶层布局规划。要将顶层 RTL 细分为多个 Pblock，请选择“Tools”→“Floorplanning”→“Auto Create Pblocks”。

要将块布局到器件内，请选择“Tools”→“Floorplanning”→“Place Pblocks”。工具会根据 slice 计数和目标使用情况来设置 Pblock 大小。

分析期间，Pblock 填充可能超过 100%，但在实现期间不会发生此现象。过度填充 Pblock 会导致在器件上其大小减小。这是获取器件顶层块的相对大小概况及其在器件上的分布方式的一项实用技巧。

图 220：“Pblock 布局 (Place Pblocks)” 的“利用率 (Utilization)”

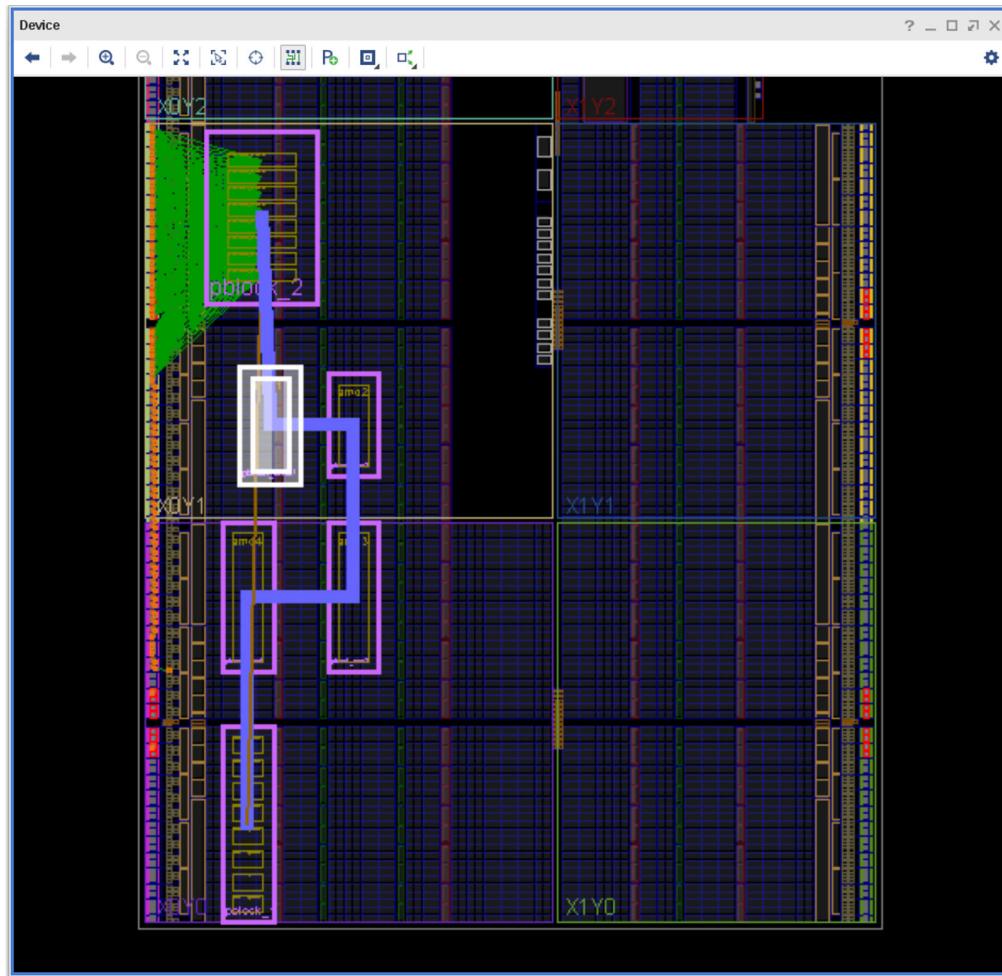


顶层布局规划

顶层布局规划用于显示与 I/O 通信的块（绿线）。连接到 2 个 Pblock 的信号线将捆绑在一起。捆绑可根据共享信号线的数量更改大小和颜色。下图中显示了 2 种顶层布局规划。

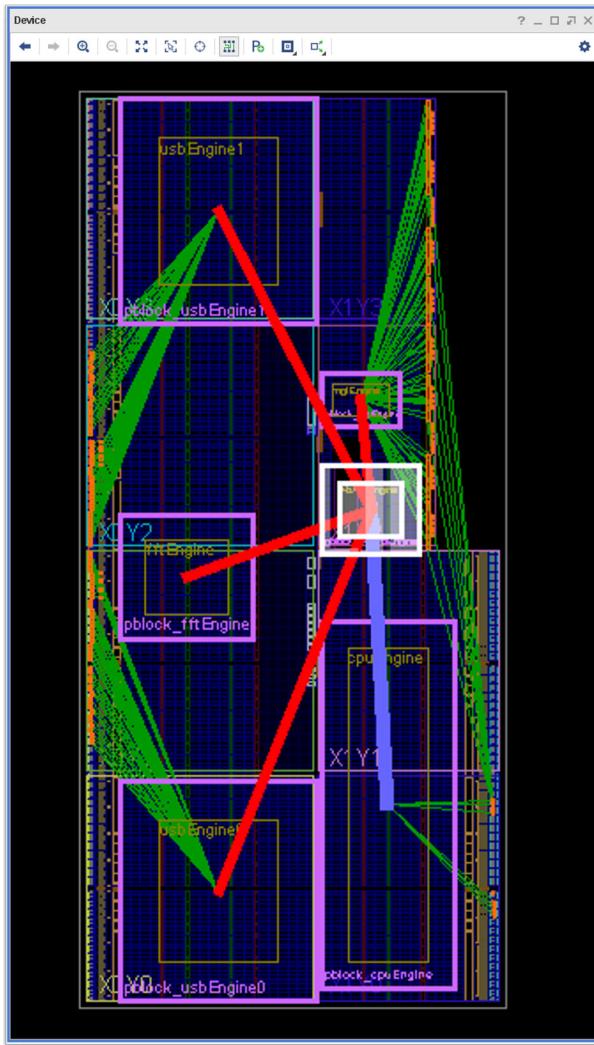
“数据路径顶层布局规划 (Data Path Top Level Floorplan)” 显示了设计的顶层块之间数据流动的方式。每个块都仅与 2 个近邻通信。绿线显示与单个块通信的布局精良的 I/O。

图 221：数据路径顶层布局规划



“控制路径布局规划”显示的设计中的所有块都与中心块进行通信。最大的连接为中心块与最右下角的块之间的连接。中心块必须分布在整个设计周围，才能与所有其它负载进行通信。

图 222：控制路径布局规划



复查布局规划

复查布局规划时，请考量器件资源。Pblock 大小设置不考量如下专用器件资源：

- 块 RAM
- DSP
- MGT
- ClockBuffer

 提示：请复查块，留意布局规划和利用率。

将特定逻辑锁定到器件站点

您可将单元布局在 FPGA 上的特定位置，例如将所有 I/O 端口都布局在赛灵思 7 系列 FPGA 设计上。赛灵思建议您在尝试时序收敛前完成 I/O 布局。

I/O 布局可能影响 FPGA 结构中的单元布局。对结构中的其它单元进行手动布局有助于为时钟逻辑和宏布局提供一致性，从而实现提升实现运行的一致性的目标。

表 19：用于逻辑布局的约束

约束	用法	注
LOC	将门电路或宏布局在特定站点 (site) 上。	SLICE 站点具有子站点（称为 BEL 站点）。
BEL	在 slice 中指定子站点用于基本元件。	

固定单元和非固定单元

固定单元和非固定单元适用于已布局的单元。这两类单元用于描述 Vivado 工具查看设计中已布局的单元的方式。

如需了解有关固定单元和非固定单元的更多信息，请访问[此链接](#)以参阅《Vivado Design Suite 用户指南：实现》([UG904](#)) 中的相应内容。



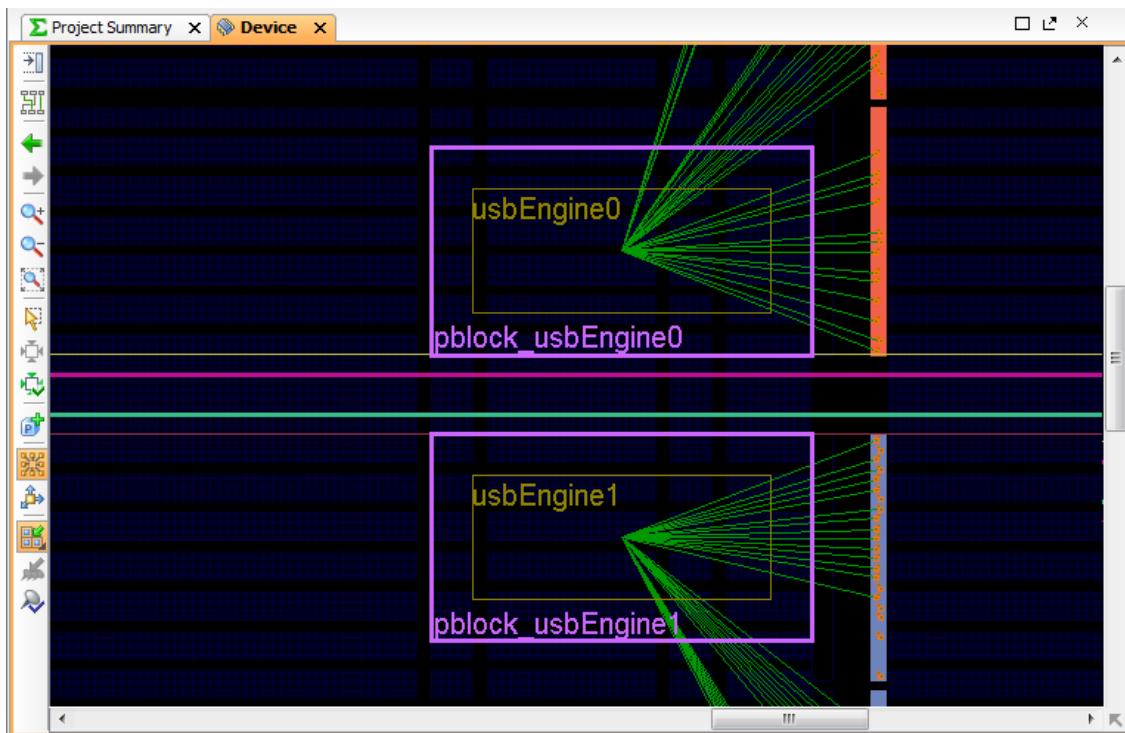
建议：完成 I/O 布局后，请使用层级 Pblock 布局规划作为用户控制的布局的起点。当发现 Pblock 无效时，应使用手动布局逻辑。

对堆叠硅片互联 (SSI) 器件进行布局规划

对于堆叠硅片互联 (SSI) 器件，存在额外的注意事项。SSI 器件是由多个以中介层相连的超级逻辑区域 (SLR) 组成的。中介层连接称为超长线路 (SLL)。当跨 SLR 交汇时会发生延迟损失。

构造设计、生成管脚输出和布局规划时，请时刻留意 SLR。将关键时序路径的逻辑单元保持在单一 SLR 内，从而最大限度减少 SLL 交汇。

图 223：最大限度减少 SLR 交汇



I/O 必须与相关 I/O 接口电路布局在同一个 SLR 内。为 SSI 器件布局逻辑时，也必须仔细考量时钟布局。



建议：让布局器尝试自动将逻辑布局到 SSI 器件内，然后再执行广泛分区。分析自动布局可提供您未考虑到的布局规划方法建议。

时序方法检查

TIMING-1：时钟修改块上的时钟波形无效

在 <CELL_TYPE> 输出 <PIN_NAME> 上指定的时钟 <CLOCK_NAME> 的时钟波形无效，与时钟修改块 (CMB) 设置不匹配。该时钟波形为 <VALUE>。期望的波形为 <VALUE>。

说明

赛灵思 Vivado® Design Suite 会根据传入主时钟的 CMB 设置和特性，在 CMB 输出上自动衍生时钟。如果用户在 CMB 输出上定义生成时钟，那么 Vivado 不会在同一定义点（信号线或管脚）上自动衍生生成时钟。DRC 警告报告称用户定义的生成时钟与 Vivado 原本将自动创建的自动衍生时钟不匹配。这可能导致硬件故障，因为设计的时序约束与器件上所发生的约束不匹配。

解决办法

如果无需用户定义的生成时钟，请移除约束并改为使用自动衍生时钟。如果需要约束，请验证生成时钟约束与自动衍生时钟波形是否匹配，或者修改 CMB 属性以与期望的时钟波形相匹配。如果要强制设置自动衍生时钟的名称，建议使用仅定义 -name 选项的 `create-generated_clock` 约束以及定义该时钟的对象（通常为 CMB 的输出管脚）的名称。请参阅《Vivado Design Suite 用户指南：使用约束》(UG903)，以了解有关创建生成时钟的信息以及自动衍生时钟重命名约束的限制。

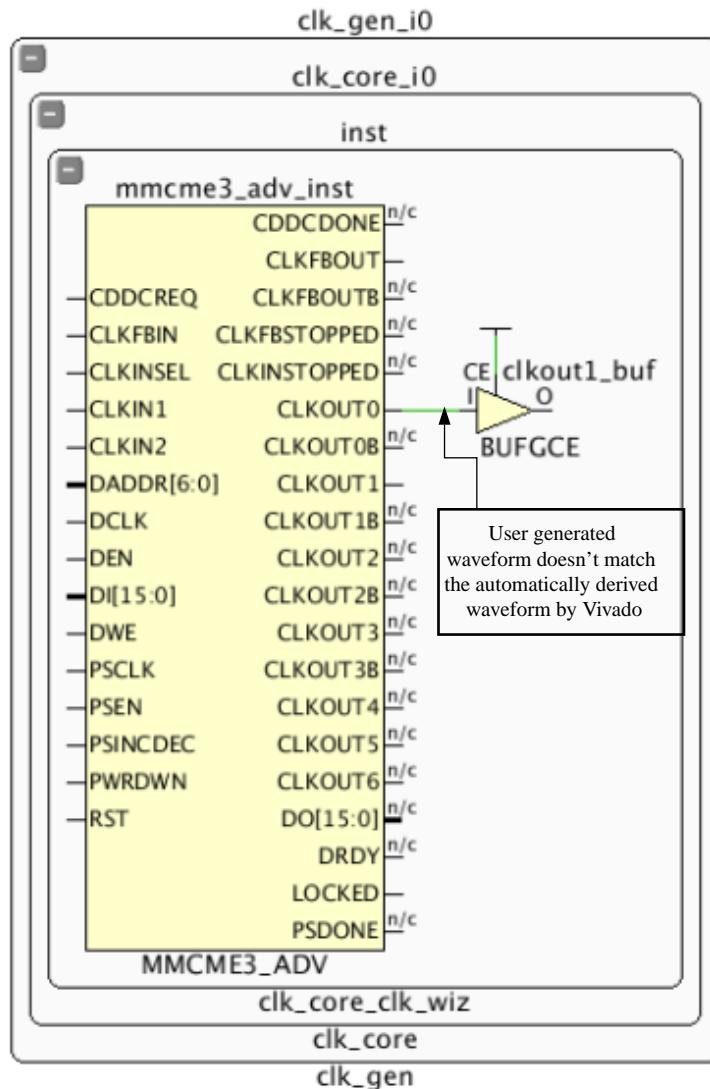
示例

在下图中，在 MMCM 实例管脚 CLKOUT0 上定义了 `create-generated_clock` 约束，但此约束与 Vivado 从 MMCM 属性设置生成的自动衍生波形不匹配。

如需仅对自动衍生时钟进行重命名，请在约束文件中的主时钟定义后使用以下约束：

```
create_generated_clock -name clkName [get_pins clk_gen_i0/clk_core_i0/inst/mmcm3_adv_inst/CLKOUT0]
```

图 224: 时钟修改块上的无效时钟波形



XI5522-111715

TIMING-2：基准时钟源管脚无效

在错误的管脚 <PIN_NAME> 上创建了基准时钟 < CLOCK_NAME >。建议仅在适当的时钟根（不含时序 arc 的输入端口或原语输出管脚）上创建基准时钟。

描述

基准时钟必须在时钟树的源时钟上定义。例如，源时钟可能是设计的输入端口。在逻辑路径中间定义基准时钟时，时序分析准确性可能降低，因为它会忽略位于基准时钟源点之前的插入延迟，从而导致无法正确执行偏差计算。因此，最好不要在内部驱动管脚上创建基准时钟。否则可能导致硬件故障。

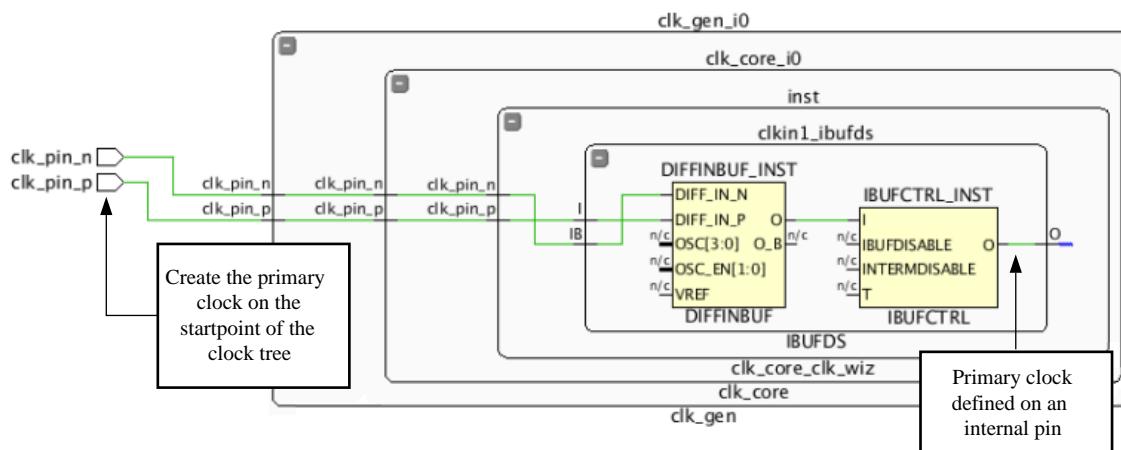
解决办法

修改 `create_clock` 约束以使用实际时钟树源。

示例

在下图中，基准时钟定义 `create_clock` 约束布局在 `IBUFCTRL` 实例的输出管脚上。如果使用 `clk_pin_p` 时钟对输入或输出端口路径进行定时，则裕量将不准确，因为将缺失时钟树插入延迟部分。差分输入缓存的基准时钟定义经布局在顶层端口 `clk_pin_p` 上。

图 225：内部管脚上的基准时钟无效



X15523-111715

TIMING-3：时钟修改块上的基准时钟无效

在时钟修改块的输出管脚或信号线 `<PIN/NET_NAME>` 上会创建基准时钟 `<clock_name>`。

描述

Vivado 会根据 CMB 设置和传入主时钟的特性，在 CMB 输出上自动衍生时钟。如果用户在 CMB 输出上定义基准时钟，那么 Vivado 不会在相同输出上自动衍生时钟。此 DRC 报告显示在 CMB 的输出上已创建基准时钟，导致与传入时钟之间的联系中断，并阻碍时钟插入延迟的正常计算。不建议如此行为，因为它可能导致时序分析不准确和硬件行为错误。

解决办法

修改约束以移除 CMB 上的 `create_clock` 约束。如需强制设置自动生成时钟的名称，赛灵思建议使用 `create_generated_clock` 约束，其中仅含 `-name` 选项和 CMB 输出管脚。请参阅《Vivado Design Suite 用户指南：使用约束》(UG903)，以获取有关创建生成时钟的其它信息。

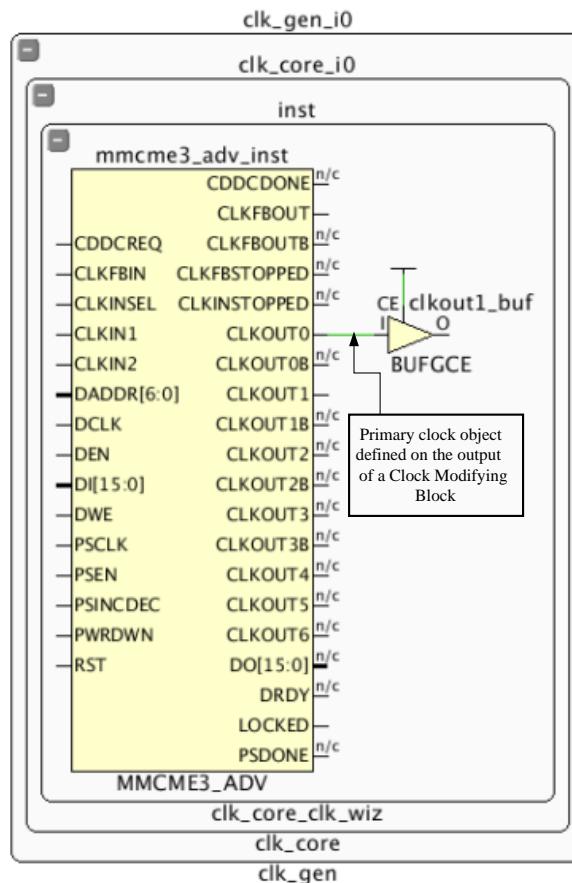
示例

在下图中，在 MMCM 实例管脚 CLKOUT0 上已定义 `create_clock` 约束。这将覆盖由 Vivado 创建的自动衍生时钟，导致与传入时钟之间的所有关系丢失。

如需仅对自动衍生时钟进行重命名，请在约束文件中的主时钟定义后使用以下约束：

```
create_generated_clock -name clkName [get_pins clk_gen_i0/clk_core_i0/inst/mmcme3_adv_inst/CLKOUT0]
```

图 226：时钟修改块上的基准时钟无效



XI5524-111715

TIMING-4：时钟树上的基准时钟重新定义无效

时钟树上的时钟重新定义无效。基准时钟 `<clock_name>` 是在时钟 `<clock_name>` 下游定义的，并覆盖其插入延迟和/或波形定义。

描述

基准时钟必须在时钟树的源时钟上定义。例如，源时钟可能是设计的输入端口。在覆盖传入时钟定义的下游定义基准时钟时，时序分析准确性可能降低，因为它会忽略位于重新定义的基准时钟源点之前的插入延迟，从而导致无法正确执行偏差计算。之所以不建议这样做，是因为这可能导致时序分析错误，从而导致硬件故障。

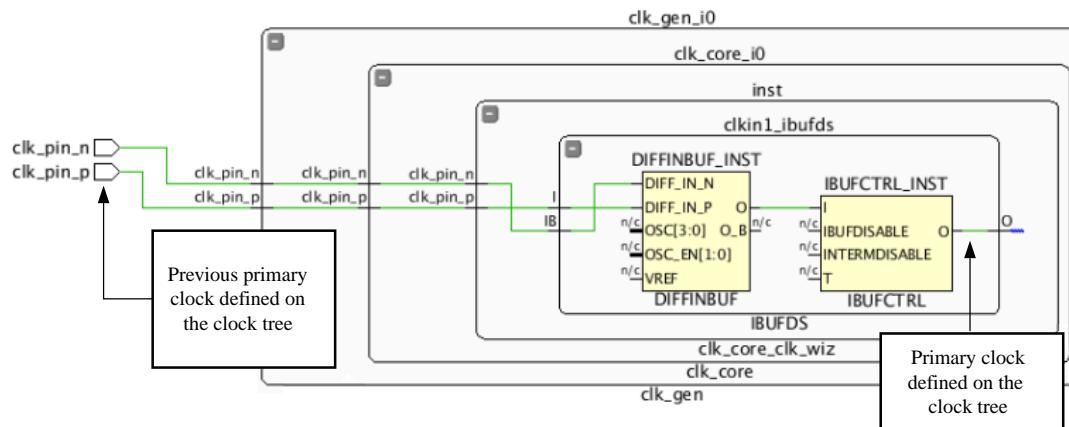
解决办法

移除下游对象上的 `create_clock` 约束，允许传输上游时钟或者创建生成时钟以引用上游基准时钟。

示例

在下图中，在顶层端口 `clk_pin_p` 上已正确定义基准时钟。但 `create_clock` 约束用于在 `IBUFCTRL` 输出上重新定义基准时钟。此新时钟将忽略 `IBUFCTRL` 前的所有延迟。

图 227：时钟树上的基准时钟重新定义无效



X15525-111715

TIMING-5：时钟树上的波形重定义无效

时钟树上的反向波形无效。生成时钟 `<clock_name>` 定义为位于时钟 `<clock_name>` 的下游，并具有波形反向定义（相比于传入时钟）。

描述

应定义与传入时钟相关的生成时钟。DRC 警告报告称生成时钟包含无效定义，例如，相比于传入时钟存在周期不同、相移或反转。

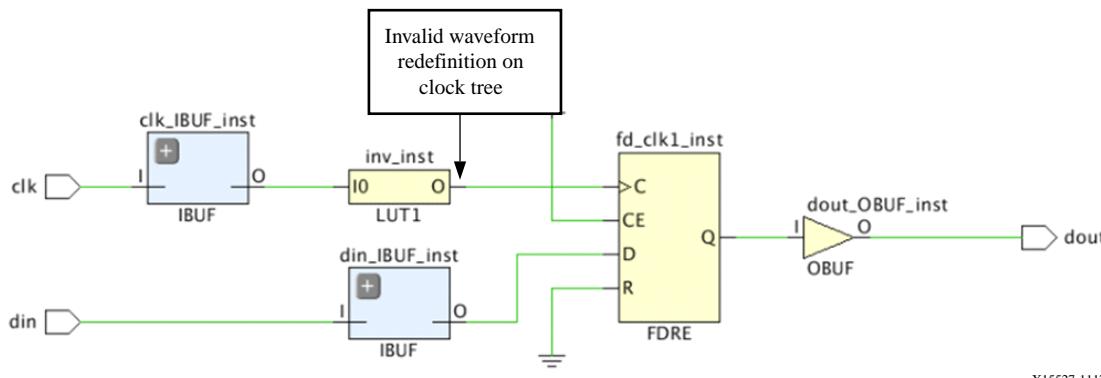
解决办法

修改 `create_generated_clock` 约束以定义与传入时钟定义相匹配的正确波形定义。如需了解有关创建正确的生成时钟约束的详情，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903)。

示例

在下图中，在 LUT1 反相器的输出上创建了 `create_generated_clock`，但未应用 `-invert` 开关。

图 228：时钟树上的波形重定义无效



XI5527-111715

TIMING-6：相关时钟间无公共基准时钟

时钟 `<clock_name1>` 与 `<clock_name2>` 之间相互关联（组合在一起进行定时），但两者间无公共基准时钟。即使满足时序要求，设计仍可能失败。要查找这些时钟之间的时序路径，请运行以下命令：`report_timing -from [get_clocks <clock_name1>] -to [get_clocks <clock_name2>]`。

描述

默认情况下，将报告的 2 个时钟视为相关联并以同步方式对其进行定时，即使这 2 个时钟并非衍生自公共基准时钟且不含已知相位关系也是如此。DRC 警告报告称时序引擎无法保证这些时钟处于同步状态。

解决办法

解决办法取决于 2 个时钟域处于异步还是同步状态。对于异步时钟，时序例外（例如，`set_max_delay -datapath_only`、`set_clock_groups` 或 `set_false_path`）应覆盖 2 个域之间的路径。当这 2 个域之间的所有路径都实现例外完全覆盖时，即可解决 DRC。

示例

对于同步时钟，如果原先 2 个时钟具有相同波形，那么可在 2 个时钟源对象上定义同一个时序时钟（请参阅以下示例）。

示例 1: `create_clock -period 10 -name clk1 [get_ports <clock-1-source> <clock-2-source>]`

如果 2 个时钟波形不同, 那么可将第 1 个时钟定义为基准时钟 (primary clock), 将第 2 个时钟定义为生成时钟, 并将第 1 个时钟指定为主时钟 (master clock) (请参阅以下示例 2)。

示例 2: `create_clock -period 10 -name clk1 [get_ports <clock-1-source>]`

如果时钟相关联, 但时钟周期比率为 2, 那么解决方案是在 1 个时钟源上创建基准时钟, 而在第 2 个时钟源上创建生成时钟:

```
create_generated_clock -source [get_ports <clock-1-source>] -name clk2 -divide_by 2  
[get_ports <clock-2-source>]
```

TIMING-7: 相关时钟间无公共节点

时钟 `<clock_name1>` 与 `<clock_name2>` 之间相互关联 (组合在一起进行定时), 但两者间无公共节点。此设置在硬件中可能失败。要查找这些时钟之间的时序路径, 请运行以下命令: `report_timing -from [get_clocks <clock_name1>] -to [get_clocks <clock_name2>]`。

描述

默认情况下, 将报告的 2 个时钟视为相关联并以异步方式对其进行定时。DRC 警告报告称时序引擎无法保证这些时钟在硬件中同步, 因为它无法确定 2 个时钟树之间的公共节点。

解决办法

解决办法取决于 2 个时钟域处于异步还是同步状态。对于异步时钟, 时序例外 (例如, `set_max_delay -datapath_only`、`set_clock_groups` 或 `set_false_path`) 应覆盖 2 个域之间的路径。

对于同步时钟, 可豁免此 DRC 警告。

在模块的非关联 (OOC) 综合期间报告违例时, 如果已知 2 个时钟在顶层具有公共节点, 那么可通过如下概述的步骤来防止出现 TIMING-7 违例:

1. 在首个输入时钟端口上将其中 1 个时钟定义为基准时钟。
2. 在第 2 个输入时钟端口上将第 2 个时钟定义为生成时钟。此时钟应参考步骤 1 中定义的基准时钟。
3. 在 2 个输入时钟端口上定义 `HD.CLK_SRC` 属性。

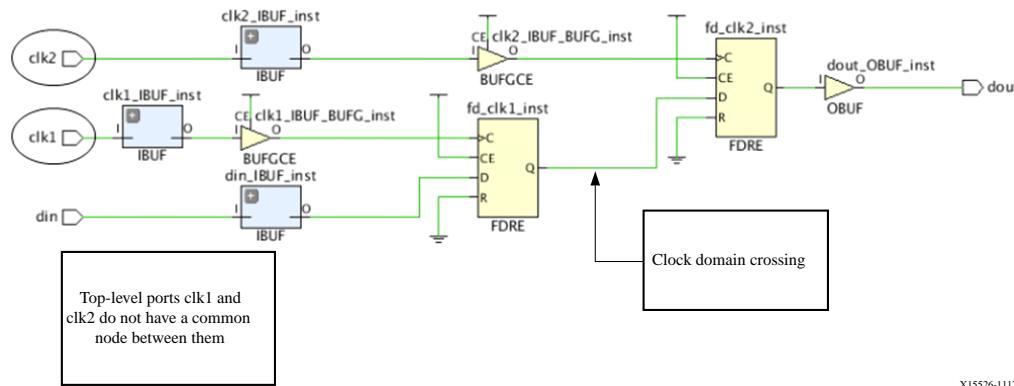
示例

在下图中, 在 `clk1` 与 `clk2` 域之间存在同步时钟域交汇 (CDC)。默认情况下, 在 Vivado 中 `clk1` 和 `clk2` 已判定为同步。但由于 `clk1` 和 `clk2` 为输入端口, 因此这 2 个时钟之间不存在公共节点关系。对此, Vivado Design Suite 无法保证 2 个时钟处于同步。

但是，如果模块以非关联 (OOC) 方式综合，并且 clk1 与 clk2 在顶层具有公共节点，那么 OOC 综合期间可能可通过定义如下约束来防止 TIMING-7 违例：

```
create_clock -period 3.000 [get_ports clk1]
set_property HD.CLK_SRC BUFGCTRL_X0Y2 [get_ports clk1]
create_generated_clock -divide_by 2 -source [get_ports clk1] \
[get_ports clk2]
set_property HD.CLK_SRC BUFGCTRL_X0Y4 [get_ports clk2]
```

图 229：关联时钟间无公共节点



XI5526-111715

TIMING-8：相关时钟间无公共周期

发现时钟 <CLOCK_NAME1> 与 <CLOCK_NAME2> 之间存在关联（组合在一起进行定时），但两者间无公共（可扩展）周期。

说明

默认情况下，将报告的 2 个时钟视为相关联并以异步方式对其进行定时。但时序引擎将 2 个时钟的波形扩展至超过 1000 个周期后无法判定公共周期。在此情况下，这 1000 余个周期的最差建立时间关系将用于时序分析。但是，时序引擎无法确保这是最消极的情况。通常周期比特别小的时钟会发生这种情况。

解决办法

由于波形不允许在两个时钟之间执行安全时序分析，因此建议将这些时钟作为异步时钟来处理。有鉴于此，时序例外应涵盖两个时钟域之间的路径（例如，`set_max_delay -datapath_only`、`set_false_path` 或 `set_clock_groups`）。

TIMING-9：未知 CDC 逻辑

在穿过 `set_false_path`、`set_clock_groups` 或 `set_max_delay -datapath_only` 约束的 2 个时钟域之间检测到 1 个或多个异步时钟域交汇。但在捕获时钟端未找到任何双寄存器逻辑同步器。建议运行 `report_cdc` 以实现完整详细的 CDC 覆盖。同时，请考虑使用 XPM_CDC 以避免出现严重性为“严重 (Critical)” 的问题。

描述

DRC 的用途是确保由时序例外加以约束的时钟间域在设计时包含安全的异步时钟域交汇电路。如需了解有关已确认的安全拓扑结构的更多详情，请参阅[时钟域交汇报告 \(Report Clock Domain Crossings\)](#)。

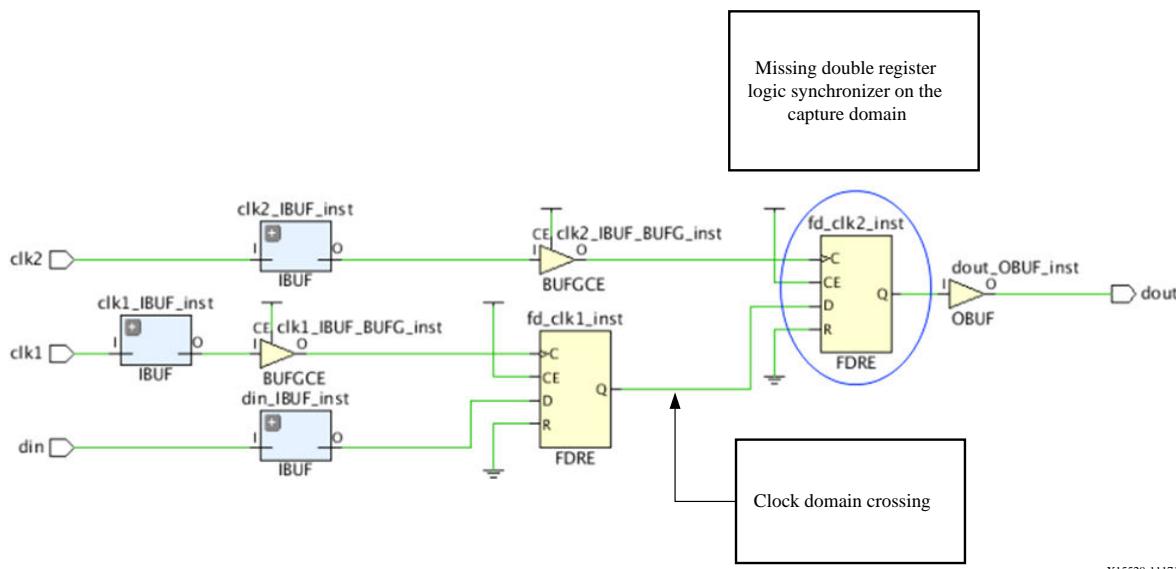
解决办法

建议在相应的设计中使时钟间路径实现正确同步。要实现此目标，请添加至少 1 个双寄存器逻辑同步器。如果在路径上已定义 1 个 FIFO 或高层次协议，则可安全忽略此 DRC。要获取 CDC 违例的详细列表，请运行 `report_cdc`。

示例

在下图中，在 `clk1` 与 `clk2` 之间存在异步时钟域。但 `clk2` 捕获域不包含用于同步数据的双寄存器逻辑同步器。

图 230：缺少同步器



X15528-111715

TIMING-10：同步器上缺失属性

在 2 个时钟域之间已检测到 1 个或多个逻辑同步器，但同步器的 1 个或 2 个寄存器上并未定义 `ASYNC_REG` 属性。建议运行 `report_cdc` 以实现完整详细的 CDC 覆盖

描述

同步器寄存器的 `ASYNC_REG` 属性必须设置为 `TRUE` 才能在综合与实现期间保留经过任意逻辑最优化的单元；并最优化其布局以实现最佳“平均故障间隔时间 (MTBF)”统计数据。

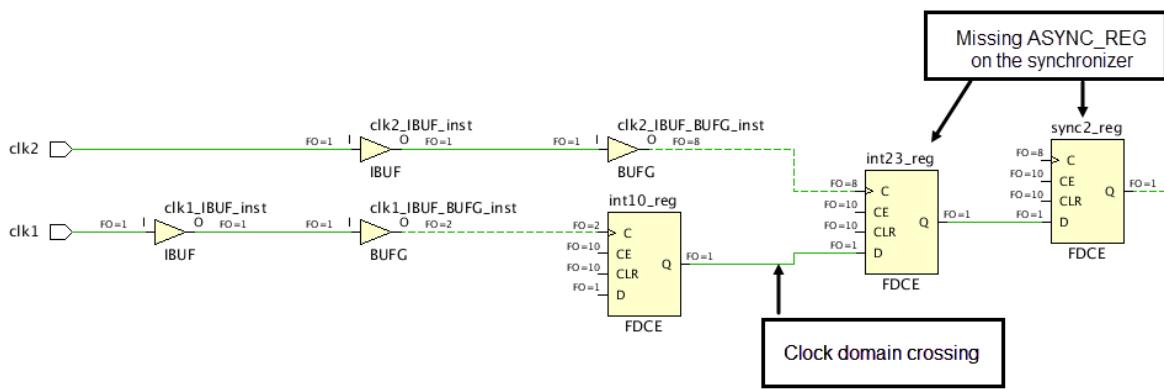
解决办法

解决方案是向逻辑同步器的每个阶段添加 ASYNC_REG 约束。要获取 CDC 违例的详细列表，请运行 report_cdc。要了解有关 ASYNC_REG 约束的更多信息，请参阅《Vivado Design Suite 属性参考指南》(UG912)。只要前 2 个同步器寄存器中至少其一缺少 ASYNC_REG 属性，就会触发 TIMING-10 违例。

示例

在下图中，在 clk1 与 clk2 之间存在异步时钟域，通过双寄存器逻辑同步器已实现正常同步。但对同步器的每个寄存器都必须应用 ASYNC_REG 属性以增大时序裕量并降低 MTBF。

图 231：同步器上缺失属性



X22694-041919

TIMING-11：含“Datapath Only”选项的最大延迟不适用

在 <PIN_NAME> 与 <PIN_NAME> 之间已应用含 -datapath_only 的最大延迟约束。起点和端点属于相同时钟域或者属于 2 个时钟域（前提是这 2 个时钟域可组合在一起安全定时）。仅建议在不存在已知相位关系的时钟间的路径上使用 -datapath_only 选项。在路径端点上找到同步器时，将豁免此 DRC。

说明

含 -datapath_only 选项的 set_max_delay 用于从建立时序裕量计算中移除时钟偏差，并忽略保持时序。set_max_delay -datapath_only 命令用于约束满足以下条件的异步信号时序路径：(1) 无时钟关系；但 (2) 需要最大延迟。不建议在同步路径上使用此约束。

解决办法

解决方案是修改 set_max_delay -datapath_only 约束，以使其避免覆盖同步时序路径。请参阅消息中列出的起点单元和端点单元，以查找关联的 set_max_delay 约束。

TIMING-12: 已禁用“时钟再收敛消极因素移除”

说明

“时钟再收敛消极因素移除 (Clock Reconvergence Pessimism Removal, CRPR)” 模式处于已禁用状态。不建议在此模式下执行时序分析，因为过消极的时钟树延迟可能导致无法实现时序收敛。

CRPR 功能用于消除人为引发的消极因素，此类消极因素源自于时钟网络的公用部分中使用的最大和最小延迟。如果禁用 CRPR，则可能导致难以实现时序收敛。

解决办法

建议启用 CRPR 分析以确保设计包含准确的时序信息。用于启用 CRPR 分析的 Tcl 命令为 config_timing_pessimism -enable。

TIMING-13: 因路径分段而忽略的时序路径

由于管脚 <PIN_NAME> 上的路径分段，某些时序路径不包含在报告中。为防止路径分段，应使用有效起点和端点列表来定义所有最小延迟约束和最大延迟约束。

描述

当时序路径细分为较小的路径以便定时，即发生路径分段。当分别在属于无效起点和端点的管脚上定义最大和最小延迟约束时，时序引擎对穿过节点的时序 arc 进行细分以便使各节点分别成为有效的起点和端点。强烈建议避免进行路径分段，因为它会导致意外后果。这可能导致时序分析不正确或者硬件故障。

解决办法

在 set_max_delay 和 set_min_delay 约束中谨慎选择有效的起点和端点，尽可能避免路径分段。如需了解有关路径分段和使用最小值/最大值延迟约束的更多信息，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903)。

TIMING-14: 时钟树上的 LUT

在时钟树上发现 LUT <CELL_NAME>。不建议在时钟路径上包含 LUT 单元。

描述

时钟路径上的 LUT 可能导致偏差过大，因为时钟必须在穿过结构的常规布线资源上进行布线。除偏差过大外，这些路径更易于受到 PVT 变动的影响。强烈建议尽可能避免使用局部时钟。

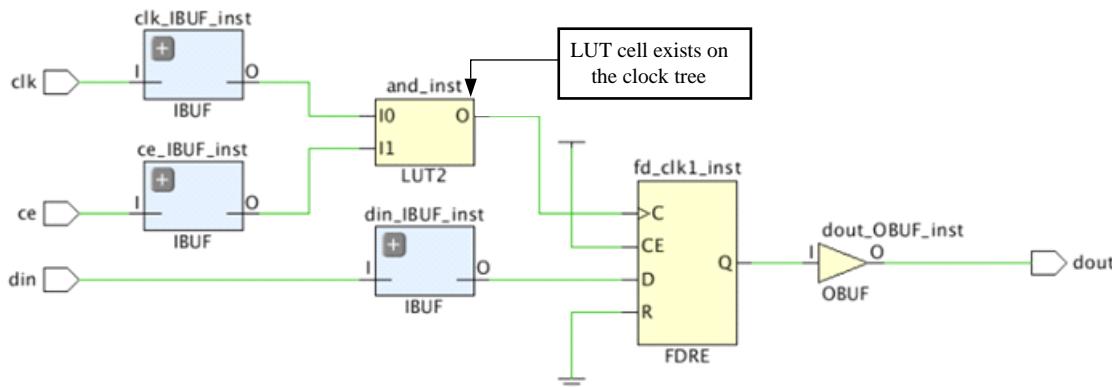
解决办法

解决方案是更改设计，移除位于时钟树上的 LUT。综合可能会在时钟门控和反转等许多情况下出现此状况。对于反转 LUT1 单元，完成 opt_design 后可能会将 LUT 吸收到下游 slice 中。请调查此状况，确保完成 opt_design 后此情况仍有效。

示例

在下图中，使用 LUT 对含时钟使能信号的时钟进行门控。路径上的 LUT 可能导致偏差过大。

图 232：时钟树上的 LUT



TIMING-15：时钟间路径上的保持时间严重违例

在 <CELL_NAME> (由 <CLOCK_NAME> 进行时钟设置) 与 <CELL_NAME> (由 <CLOCK_NAME> 进行时钟设置) 之间存在 <VALUE> ns 的严重的时钟间偏差，此偏差导致存在 <VALUE> ns 的保持时间严重违例。布线期间修复保持时间严重违例可能会影响建立时间裕量，导致时序收敛难度提升。

说明

DRC 警告报告称由于时钟间偏差导致保持时间严重违例，可能导致实现期间难以达成时序收敛。建议对大于 1.0 ns 的严重的时钟间偏差进行调查，以确保约束或设计拓扑结构正确。

解决办法

调查时序路径上的时钟间严重偏差是否应定时，或者是否与未最优化的时序约束有关。如果由于未约束的 CDC 路径而导致发生严重偏差，请添加必要的时序例外。如果由于与时钟数关联的逻辑而导致发生违例，请调查是否可通过改进路径的拓扑结构来更轻松地收敛时序。

TIMING-16：建立时间严重违例

在 <CELL_NAME> (由 <CLOCK_NAME> 进行时钟设置) 与 <CELL_NAME> (由 <CLOCK_NAME> 进行时钟设置) 之间存在 <VALUE> ns 的建立时间严重违例。这些阶段结束时出现的建立时间严重违例可能难以在布局后实现流程期间进行修复，原因可能是 XDC 约束或设计架构未最优化。

说明

此 DRC 警告用于报告在实现期间较难以达成时序收敛的建立时间违例。建议对大于 1.0 ns 的建立时间违例进行调查，以确保约束或设计拓扑结构正确。

解决办法

调查建立时间严重违例是否源于应定时的时序路径，或者违规是否与未最优化的时序约束有关。如果由于未约束的 CDC 路径而导致发生建立违例，请添加必要的时序例外。如果由于存在大量组合逻辑而导致发生违例，请调查是否可通过改进路径的拓扑结构来更轻松地收敛时序。

TIMING-17：未设置时钟的时序单元

时序时钟无法到达时钟管脚 <PIN_NAME>。

说明

DRC 报告可列出不受时序时钟约束的时序单元，此类时序单元会影响针对报告的单元所生成的时序分析。强烈建议正确定义所有时钟以实现最大范围的时序路径覆盖，并保证最高的准确性。否则可能因缺失时序分析而导致硬件故障。

解决办法

解决办法是在驱动未约束的时序单元的时钟树上创建缺失的基准时钟或生成时钟。

TIMING-18：输入或输出延迟缺失

在 <PORT_NAME> 上缺失与 <CLOCK_NAME> 时钟相关的 <INPUT/OUTPUT> 延迟。

描述

IO 时序与包含外部器件的时序路径有关。输入和输出延迟可指定与设计接口处的时钟沿相关的端口路径延迟。强烈建议添加输入/输出延迟约束，以确保 FPGA 接口可满足外部器件的时序要求。

解决办法

添加对应于必需的开发板应用的必需输入和输出延迟约束。

TIMING-19：ODDR 上的生成时钟波形反向

生成时钟 <CLOCK_NAME> 的波形与传入时钟 <CLOCK_NAME> 的波形相比为反向。

描述

前向时钟端口上的生成时钟应定义为与传入时钟相关。DRC 警告报告称，通过对比传入源时钟发现，前向时钟端口上的生成时钟具有无效的波形（例如，波形反向）。这可能导致硬件故障，因为与前向时钟关联的端口的时序分析与器件上所发生的操作不匹配。

解决办法

修改 `create_generated_clock` 约束以定义与传入时钟定义匹配的正确波形。如需了解有关创建正确的生成时钟约束的详情，请参阅《Vivado Design Suite 用户指南：使用约束》([UG903](#))。

TIMING-20：未设置时钟的锁存器

无法正确分析锁存器 <CELL_NAME>，因为时序时钟无法访问其控制管脚 <PIN_NAME>。

说明

此 DRC 报告的是不受时序时钟约束的锁存器单元列表，这会影响生成的时序分析。强烈建议正确定义所有时钟以实现最大范围的时序路径覆盖，并保证最高的准确性。否则后果可能导致时序分析覆盖范围不完整，从而导致硬件故障。

解决办法

解决办法是在时钟树的源端创建主时钟或生成时钟，以驱动锁存器单元上未约束的控制管脚。

TIMING-21：MMCM 上的 COMPENSATION 属性无效

MMCM <CELL_NAME> 包含的与其反馈回路连接相关的 COMPENSATION 属性值无效。如果反馈回路连接至 FPGA 外部，该属性应设置为 EXTERNAL。如果反馈回路位于 FPGA 内部，该属性应设置为 ZHOLD。

说明

MMCM 补偿模式用于定义为输出时钟的延迟补偿配置 MMCM 反馈的方式。根据 MMCM 用例，反馈路径应与特定拓扑结构相匹配。此 DRC 警告报告称 MMCM 用例的拓扑结构与 COMPENSATION 属性值不匹配。这可能导致硬件中因时序分析不匹配而出现意外行为。

解决办法

建议对设计中 MMCM 的 COMPENSATION 属性保留默认值 AUTO。Vivado 集成设计环境 (IDE) 将在电路拓扑结构上自动选择相应的补偿值。如需了解有关补偿属性和输入延迟补偿的其它信息，请参阅特定于您的架构的“时钟资源用户指南”。

TIMING-22：MMCM 上缺少外部延迟

MMCM <CELL_NAME> 具有外部反馈回路，但在 FBOUT 与 FBIN 之间未指定任何外部延迟。建议在使用外部反馈回路连接到管脚 FBOUT 和 FBIN 的 2 个端口之间使用 set_external_delay 指定外部延迟。

描述

如果反馈板载走线与外部组件的走线相匹配，那么可配置 MMCM 以进行外部纠偏。在计算 MMCM 补偿延迟时会使用外部延迟值。这可能导致硬件故障（在 IO 路径上尤其如此），因为 MMCM 补偿的时序分析与器件上所发生的操作不匹配。

解决办法

在外部反馈输入和输出端口之间为已定义的外部走线延迟添加 set_external_delay 约束。如需了解有关 set_external_delay 命令的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》(UG835)。

示例

```
set_external_delay -from <output_port> -to <input_port> <external_delay_value>
```

TIMING-23：发现组合循环

在组合路径上已检测到时序循环。在 <CELL_NAME1> 与 <CELL_NAME2> 之间已禁用时序 arc，以中断时序循环。

描述

当组合逻辑的输出回馈到其输出从而生成时序循环时，就会创建组合时序循环。此循环会无限循环相同路径运行且无法定时，从而导致周期数徒然增加。为解决时序循环，Vivado IDE 会禁用循环中的单元上的时序 arc。

解决办法

如果您不想创建组合反馈循环，请通过修改设计源文件 (RTL) 来纠正问题。但由于本应使用时序循环，请使用 `set_disable_timing` 命令在最合理位置（通常为反馈路径处）中断时序循环，而不是任由 Vivado Timing 在随机位置将其中断。

TIMING-24：仅最大延迟数据路径已被覆盖

时钟 `<clock_name1>` 和 `<clock_name2>` 之间的 `set_clock_groups` 或 `set_false_path` 覆盖 `set_max_delay -datapath_only`（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 `<#>`）。不建议覆盖 `set_max_delay -datapath_only` 约束。将时钟之间的 `set_clock_groups` 或 `set_false_path` 替换为点对点 `set_false_path` 约束。

描述

仅当 `set_max_delay -datapath_only` 约束被时钟间的 `set_clock_groups` 或 `set_false_path` 约束所覆盖时，才会出现此 DRC 警告。如果点对点 `set_false_path` 覆盖 `set_max_delay -datapath_only`，那么将不会生成此 DRC 报告。

解决办法

解决方案是将时钟之间的 `set_clock_groups` 或 `set_false_path` 替换为点对点伪路径约束，以避免错误覆盖 `set_max_delay -datapath_only` 约束。

TIMING-25：千兆位收发器 (GT) 上的时钟波形无效

收发器输出管脚 `<pin_name>` 上或连接到该管脚的信号线上定义的时钟 `<clock_name>` 的波形与收发器不一致，或者缺少参考时钟定义。自动衍生的时钟周期为 `<period>`，用户定义的时钟周期为 `<period>`。

描述

对于 UltraScale 器件，Vivado 会根据 GT 设置和传入主时钟的特性，在 GT 输出上自动衍生时钟。对于 7 系列器件，Vivado 不会自动衍生 GT 时钟；而是由您负责在 GT 输出管脚上创建相应的基准时钟。DRC 警告报告称用户定义的时钟与 Vivado 原本将自动创建的自动衍生时钟不匹配。这可能导致硬件故障，因为设计的时序约束与器件上所发生的约束不匹配。

解决办法

如果无需用户定义的生成时钟，请移除约束并改为使用自动衍生时钟。如果需要约束，请验证生成时钟约束与自动衍生时钟波形是否匹配，或者修改 GT 属性以与期望的时钟波形相匹配。如果要强制设置自动衍生时钟的名称，建议使用仅定义 `-name` 选项的 `create_generated_clock` 约束以及定义该时钟的对象（通常为 GT 的输出管脚）的名称。请参阅《Vivado Design Suite 用户指南：使用约束》(UG903)，以了解有关创建生成时钟的信息以及自动衍生时钟重命名约束的限制。

TIMING-26: 千兆位收发器 (GT) 上时钟缺失

输出时钟管脚 <PIN_NAME> 未定义时钟。在 <PORT_NAME> 输入端口上创建基准时钟，以便 Vivado 自动衍生出缺失的 GT 时钟。

描述

对于 UltraScale 器件，Vivado 会根据 GT 设置和传入主时钟的特性，在 GT 输出上自动衍生时钟。DRC 警告报告称 Vivado 无法自动衍生 GT 的输出时钟，因为输入端口缺少基准时钟。由此导致无法对连接到时钟相关 GT 的下游逻辑进行定时。

解决办法

在建议的 GT 输入端口上创建基准时钟。

TIMING-27: 层级管脚上的基准时钟无效

在错误的内部管脚 <PIN_NAME> 上创建了基准时钟 <CLOCK_NAME>。当基准时钟的驱动管脚具有连接到多个时钟管脚的扇出时，最好不要在层级管脚上创建基准时钟。

说明

如果某个时钟遍历驱动，并在下游层级管脚上定义新时钟，那么该层级管脚的下游单元的时序分析将有别于驱动管脚扇出上的单元的时序分析。如果在驱动时钟与层级管脚时钟之间存在任何同步路径，那么偏差将不准确且时序验收将无效。这可能导致硬件故障。

解决办法

移除层级管脚上的基准时钟定义，或者如果确实需要下游时钟，请使用生成时钟，并改为将驱动时钟指定为主时钟。

TIMING-28: 时序约束引用的自动衍生时钟

自动衍生时钟 <CLOCK_NAME> 在时序约束内部按名称来引用（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <#>）。建议按随附于时钟的管脚名称来引用自动衍生时钟：`get_clocks -of_objects [get_pins <PIN_NAME>]`。

描述

源管脚对象应引用自动衍生时钟。开发期间可能由于对网表或约束进行修改而导致自动衍生时钟名称改变。除非已重命名，否则应不鼓励按名称引用自动衍生时钟，因为修改设计后可能导致后续运行时约束失效。

解决办法

使用 `[get_clocks -of_objects [get_pins <PIN_NAME>]]` 将约束修改为按连接到时钟的管脚名称来引用自动衍生时钟。或者，使用 `create_generated_clock` 约束来强制设置自动衍生时钟的名称。即使某些时序约束已引用自动衍生的时钟，仍可对其进行重命名。如需了解有关使用生成时钟约束来强制设置时钟名称的详细信息，请参阅《Vivado Design Suite 用户指南：使用约束》([UG903](#))。

TIMING-29：多周期路径对不一致

建立和保持多周期路径约束通常应引用相同的 `-start` 对（针对 SLOW-to-FAST 同步时钟）或 `-end` 对（针对 FAST-to-SLOW 同步时钟），请参阅 Vivado IDE 中的“Timing Constraint”窗口中的约束位置 `<#>`。

描述

默认情况下，`set_multicycle_path` 约束用于修改源时钟（针对保持时间）或目标时钟（针对建立时间）相关的路径要求乘数。对于某些用例，路径要求必须根据特定时钟沿倍增。

解决办法

对于建立和保持时间，请修改 `set_multicycle_path` 约束，分别引用目标时钟 (`-end`)（针对 SLOW-to-FAST 同步时钟）和源时钟 (`-start`)（针对 FAST-to-SLOW 同步时钟）。请参阅《Vivado Design Suite 用户指南：使用约束》([UG903](#))，以获取有关正确设置时钟间的多周期路径的更多信息。

TIMING-30：生成时钟的欠优化主时钟源管脚选择

生成时钟 `<CLOCK_NAME>` 包含欠优化的主时钟源管脚选择，时序可能处于消极状态。

描述

虽然 `create_generated_clock` 命令允许您指定任意参考时钟，但是生成时钟应引用在其直接扇入中传输的时钟。此 DRC 警告报告称生成时钟与已定义的主时钟相关联，但该主时钟位于上游比传入主时钟更远处。在此情况下，时序分析可能更消极，并对主时钟与生成时钟之间的路径应用额外的时钟不确定性。这可能导致时序收敛难度略有提升。建议将生成时钟与衍生出生成时钟的主时钟源管脚关联。

解决办法

修改 `create_generated_clock` 约束以参考主时钟源管脚，在设计中生成时钟是从该管脚直接衍生的。

QoR 建议报告 RTL 代码更改示例

TIMING-201：为 RAM 添加输出寄存器

为 RAM 添加输出寄存器可改进 RAM 读取数据路径的时钟输出 (clock to out) 时间。这样可为布局布线工具提供更多的灵活性，使其能够实现最优化的 RAM 布局，并提供将寄存器布局到结构内而不是布局到 RAM 内的选项，从而实现关键路径最优化。

综合工具可轻松推断输出寄存器。此类寄存器必须包含同步复位，或者不含任何复位。

Verilog 代码示例

图 233：更改前

```

module single_sdp_ram #((
    parameter C_DATA_WIDTH = 16,
    parameter C_ADDR_WIDTH = 10) (
    input CLKA,
    input WEA,
    input [C_ADDR_WIDTH-1:0] ADDRA,
    input [C_DATA_WIDTH-1:0] DINA,
    input ENB,
    input [C_ADDR_WIDTH-1:0] ADDRDB,
    output [C_DATA_WIDTH-1:0] DOUTB,
    input CLKDB
);

reg [C_DATA_WIDTH-1:0] ram_i [2**C_ADDR_WIDTH-1:0];
reg [C_DATA_WIDTH-1:0] ram_data = {C_DATA_WIDTH{1'b0}};

always @(posedge CLKA)
    if (WEA)
        ram_i[ADDRA] <= DINA;

always @(posedge CLKDB)
    if (ENB)
        ram_data <= ram_i[ADDRDB];

// 1 clock cycle read latency at the cost of a longer clock-to-out timing
assign DOUTB = ram_data;

endmodule

```

图 234: 更改后

```
module single_sdpram #((
    parameter C_DATA_WIDTH = 16,
    parameter C_ADDR_WIDTH = 10) (
    input CLKA,
    input WEA,
    input [C_ADDR_WIDTH-1:0] ADDR_A,
    input [C_DATA_WIDTH-1:0] DINA,
    input ENB,
    input [C_ADDR_WIDTH-1:0] ADDR_B,
    output [C_DATA_WIDTH-1:0] DOUTB,
    input CLKB
);

reg [C_DATA_WIDTH-1:0] ram_i [2**C_ADDR_WIDTH-1:0];
reg [C_DATA_WIDTH-1:0] ram_data = {C_DATA_WIDTH{1'b0}};

always @ (posedge CLKA)
    if (WEA)
        ram_i[ADDR_A] <= DINA;

always @ (posedge CLKB)
    if (ENB)
        ram_data <= ram_i[ADDR_B];

// 2 clock cycle read latency with improved clock-to-out timing
reg [C_DATA_WIDTH-1:0] doutb_reg = {C_DATA_WIDTH{1'b0}};
always @ (posedge CLKB)
    doutb_reg <= ram_data;

assign DOUTB = doutb_reg;

endmodule
```

VHDL 代码示例

图 235: 更改前

```
-- 2D Array Declaration for RAM signal
type ram_type is array (2**C_ADDR_WIDTH-1 downto 0) of std_logic_vector (C_DATA_WIDTH-1 downto 0);
signal RAM_DATA : std_logic_vector(C_DATA_WIDTH-1 downto 0) ;

...
process(CLKA)
begin
    if(CLKA'event and CLKA = '1') then
        if(WEA = '1') then
            RAM(to_integer(unsigned(ADDRA))) <= DINA;
        end if;
    end if;
end process;

process(CLKB)
begin
    if(CLKB'event and CLKB = '1') then
        if(ENB = '1') then
            RAM_DATA <= RAM(to_integer(unsigned(ADDRB)));
        end if;
    end if;
end process;

-- Read latency of 1 but slower clock to out time
DOUTB <= RAM_DATA;
```

图 236: 更改后

```
-- 2D Array Declaration for RAM signal
type ram_type is array (2**C_ADDR_WIDTH-1 downto 0) of std_logic_vector (C_DATA_WIDTH-1 downto 0);
signal RAM_DATA : std_logic_vector(C_DATA_WIDTH-1 downto 0) ;
signal DOUTB_REG : std_logic_vector(C_DATA_WIDTH-1 downto 0) := (others => '0');
...
process(CLKA)
begin
    if(CLKA'event and CLKA = '1') then
        if(WEA = '1') then
            RAM(to_integer(unsigned(ADDRA))) <= DINA;
        end if;
    end if;
end process;

process(CLKB)
begin
    if(CLKB'event and CLKB = '1') then
        if(ENB = '1') then
            RAM_DATA <= RAM(to_integer(unsigned(ADDRB)));
        end if;
    end if;
end process;

-- Read latency of 2 but faster clock to out time
process(CLKB)
begin
    if(CLKB'event and CLKB= '1') then
        DOUTB_REG <= RAM_DATA;
    end if;
end process;
```

TIMING-202：添加额外流水线以拓宽倍频器

宽型倍频器（即给定架构中至少一个端口大于 DSP slice 所支持的最大宽度）需额外流水线以实现 DSP slice 的最大运行频率。流水线阶段的数量需根据所需宽度更改。

通过在 RTL 中对宽型倍频器输出添加额外阶段，综合即可将其移至最优化位置以大幅简化重新编码。

Verilog 代码示例

图 237: 更改前

```
module wide_multiplier #(parameter DATA_WIDTH=30) (
    input clk,
    input [DATA_WIDTH-1:0] a,
    input [DATA_WIDTH-1:0] b,
    output [2*DATA_WIDTH-1:0] p
);

    reg [DATA_WIDTH-1:0] a_r;
    reg [DATA_WIDTH-1:0] b_r;
    reg [2*DATA_WIDTH-1:0] m_r;

    always @ (posedge clk)
    begin
        a_r <= a;
        b_r <= b;
        m_r <= a_r * b_r;
    end

    assign p = m_r;

endmodule
```

图 238: 更改后

```
module wide_multiplier #(parameter DATA_WIDTH=30) (
    input clk,
    input [DATA_WIDTH-1:0] a,
    input [DATA_WIDTH-1:0] b,
    output [2*DATA_WIDTH-1:0] p
);

reg [DATA_WIDTH-1:0] a_r;
reg [DATA_WIDTH-1:0] b_r;
reg [2*DATA_WIDTH-1:0] m_r;
reg [2*DATA_WIDTH-1:0] m_2r;
reg [2*DATA_WIDTH-1:0] m_3r;
reg [2*DATA_WIDTH-1:0] m_4r;

always @ (posedge clk)
begin
    a_r <= a;
    b_r <= b;
    m_r <= a_r * b_r;
    // Add more pipeline stages after the multiplier
    m_2r <= m_r;
    m_3r <= m_2r;
    m_4r <= m_3r;
end

assign p = m_4r;

endmodule
```

VHDL 代码示例

图 239: 更改前

```
entity wide_multiplier is
  Generic ( DATA_WIDTH : integer := 30);
  Port ( clk : in STD_LOGIC;
         a : in STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
         b : in STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
         p : out STD_LOGIC_VECTOR (2*DATA_WIDTH-1 downto 0));
end wide_multiplier;

architecture Behavioral of wide_multiplier is
begin
  process (clk)
  begin
    if rising_edge (clk) then
      a_r <= SIGNED(a);
      b_r <= SIGNED(b);
      m_r <= a_r * b_r;
    end if;
  end process;
  p <= STD_LOGIC_VECTOR(m_r);

end Behavioral;
```

图 240: 更改后

```
entity wide_multiplier is
  Generic ( DATA_WIDTH : integer := 30);
  Port ( clk : in STD_LOGIC;
          a : in STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
          b : in STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
          p : out STD_LOGIC_VECTOR (2*DATA_WIDTH-1 downto 0));
end wide_multiplier;

architecture Behavioral of wide_multiplier is

signal a_r : SIGNED(DATA_WIDTH-1 downto 0);
signal b_r : SIGNED(DATA_WIDTH-1 downto 0);
signal m_r : SIGNED(2*DATA_WIDTH-1 downto 0);
signal m_2r : SIGNED(2*DATA_WIDTH-1 downto 0);
signal m_3r : SIGNED(2*DATA_WIDTH-1 downto 0);
signal m_4r : SIGNED(2*DATA_WIDTH-1 downto 0);

begin

process (clk)
begin
  if rising_edge (clk) then
    a_r <= SIGNED(a);
    b_r <= SIGNED(b);
    m_r <= a_r * b_r;
    -- Add extra pipes after the multiplier
    m_2r <= m_r;
    m_3r <= m_2r;
    m_4r <= m_3r;
  end if;
end process;
p <= STD_LOGIC_VECTOR(m_4r);

end Behavioral;
```

UTIL-203：使用分布式 RAM 推断的大型 ROM

阵列深度远超 64 位的 RAM 最好推断为块 RAM。默认情况下综合工具会尝试执行此推断，但有时由于编码或约束限制导致此推断无法完成。

无法推断块 RAM 的主要原因是缺少输出寄存器。块 RAM 仅支持同步读取，但分布式 RAM 无此要求。第 2 个原因是读取阵列或 ROM_STYLE 属性时强制要求的资源类型必须通过推断才能得到。

通过简单修改即可改善 LUT 利用率、时序以及拥塞（如果适用）。

Verilog 代码示例

图 241: 更改前

```
module sp_rom (clk, en, addr, dout);
    input clk;
    input en;
    input [5:0] addr;
    output [83:0] dout;

    reg [83:0] data;

    // Combinatorial read process prevents Block RAM usage
    always_comb
    begin
        data <= 84'h11223344;
        case(addr)
            6'b000000: data <= 84'hFF200A; 6'b100000: data <= 84'hFF2222;
            6'b000001: data <= 84'hFF0300; 6'b100001: data <= 84'hFF4001;
            ...
            6'b011111: data <= 84'hFF0102; 6'b111111: data <= 84'hFF400D;
        default: data <= 84'hFF00111;
        endcase
    end

    assign dout = data;

endmodule
```

图 242: 更改后

```
module sp_rom (clk, en, addr, dout);
    input clk;
    input en;
    input [5:0] addr;
    output [83:0] dout;

    reg [83:0] data;

    always @ (posedge clk) // Add an output register to help this ROM get
                           // inferred as block RAM.
    begin
        data <= 84'h11223344;
        if (en)
            case(addr)
                6'b000000: data <= 84'hFF200A; 6'b100000: data <= 84'hFF2222;
                6'b000001: data <= 84'hFF0300; 6'b100001: data <= 84'hFF4001;
                ...
                6'b011111: data <= 84'hFF0102; 6'b111111: data <= 84'hFF400D;
            default: data <= 84'hFF00111;
            endcase
    end

    assign dout = data;

endmodule
```

VHDL 代码示例

图 243: 更改前

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity sp_rom is
    Port ( clk : in STD_LOGIC;
            en : in STD_LOGIC;
            addr : in STD_LOGIC_VECTOR (9 downto 0);
            dout : out STD_LOGIC_VECTOR (15 downto 0));
end sp_rom;

architecture Behavioral of sp_rom is

signal data : STD_LOGIC_VECTOR(15 downto 0);

begin

-- Unregistered ROM read prevents Block RAM usage
process (addr, en)
begin
    if (en = '1') then
        case (TO_INTEGER(UNSIGNED(addr))) is
            when 0 => data <= X"3423";
            when 1 => data <= X"ED77";
            ...
            when 1023 => data <= X"CD34";
            when others => data <= X"0111";
        end case;
    end if;
end process;

dout <= data;

end Behavioral;
```

图 244: 更改后

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity sp_rom is
    Port ( clk : in STD_LOGIC;
            en : in STD_LOGIC;
            addr : in STD_LOGIC_VECTOR (9 downto 0);
            dout : out STD_LOGIC_VECTOR (15 downto 0));
end sp_rom;
|
architecture Behavioral of sp_rom is
|
signal data : STD_LOGIC_VECTOR(15 downto 0);
|
begin
|
-- Register process to enable inference of Block RAM
process (clk)
begin
    if rising_edge (clk) then
        if (en = '1') then
            case (TO_INTEGER(UNSIGNED(addr))) is
                when 0 => data <= X"3423";
                when 1 => data <= X"ED77";
                ...
                when 1023 => data <= X"CD34";
                when others => data <= X"0111";
            end case;
        end if;
    end if;
end process;
|
dout <= data;
|
end Behavioral;
```

UTIL-204：RAM 阵列未能有效使用

当 RAM 阵列地址或数据大小比适合填入 RAM 的理想大小稍大时，综合工具会添加额外 RAM（通常为 RAMB18）以满足所需的额外位数。这可能导致添加的 RAM 无法得到有效利用。

通过将 RAM 分为 2 个阵列，其中一个填入块 RAM，另一个填入分布式 RAM，即可使资源得到最充分的利用，以供综合工具推断。

当 RAM 位于自己的层级内时，此方法最有效。这样即可将单一阵列替换为额外层级以便于对 RAM 阵列进行拆分。

Verilog 代码示例

图 245: 更改前

```
// Single ram instance
single_sdp_ram #((
    .C_DATA_WIDTH(20),
    .C_ADDR_WIDTH(C_ADDR_WIDTH),
    .C_RAM_STYLE("block"))
) single_ram_i (
    .CLKA(CLKA),
    .WEA(WEA),
    .ADDRA(ADDRA),
    .DINA(DINA[19:0]),
    .ENB(ENB),
    .RSTB(RSTB),
    .ADDRB(ADDRB),
    .DOUTB(DOUTB[19:0]),
    .REGCEB(REGCEB),
    .CLKB(CLKB)
);
```

图 246: 更改后

```
// Split RAM array into two instances and target different resources
single_sdp_ram #(
    .C_DATA_WIDTH(18),
    .C_ADDR_WIDTH(C_ADDR_WIDTH),
    .C_RAM_STYLE("block")
) first_ram_i (
    .CLKA(CLKA),
    .WEA(WEA),
    .ADDRA(ADDRA),
    .DINA(DINA[17:0]),
    .ENB(ENB),
    .RSTB(RSTB),
    .ADDRB(ADDRB),
    .DOUTB(DOUTB[17:0]),
    .REGCEB(REGCEB),
    .CLKB(CLKB)
);

single_sdp_ram #(
    .C_DATA_WIDTH(2),
    .C_ADDR_WIDTH(C_ADDR_WIDTH),
    .C_RAM_STYLE("distributed")
) second_ram_i (
    .CLKA(CLKA),
    .WEA(WEA),
    .ADDRA(ADDRA),
    .DINA(DINA[19:18]),
    .ENB(ENB),
    .RSTB(RSTB),
    .ADDRB(ADDRB),
    .DOUTB(DOUTB[19:18]),
    .REGCEB(REGCEB),
    .CLKB(CLKB)
);
```

VHDL 代码示例

图 247: 更改前

```
BRAM_i : SingleRam generic map (
    C_ADDR_WIDTH      => C_ADDR_WIDTH,
    C_DATA_WIDTH       => 20,
    C_RAM_PERFORMANCE => C_RAM_PERFORMANCE,
    C_RAM_STYLE        => "block"
) port map (
    CLKA   => CLKA ,
    WEA    => WEA  ,
    ADDRA  => ADDRA,
    DINA   => DINA(19 downto 0),
    CLKB   => CLKB ,
    ENB    => ENB  ,
    RSTB   => RSTB ,
    REGCEB => REGCEB,
    ADDR_B => ADDR_B,
    DOUTB  => DOUTB(19 downto 0)
);
```

图 248: 更改后

```
BRAM_i : SingleRam generic map (
    C_ADDR_WIDTH      => C_ADDR_WIDTH,
    C_DATA_WIDTH       => 18,
    C_RAM_PERFORMANCE => C_RAM_PERFORMANCE,
    C_RAM_STYLE        => "block"
) port map (
    CLKA    => CLKA ,
    WEA     => WEA  ,
    ADDRA   => ADDRA,
    DINA    => DINA(17 downto 0),
    CLKB    => CLKB ,
    ENB     => ENB  ,
    RSTB    => RSTB ,
    REGCEB  => REGCEB,
    ADDRDB  => ADDRDB,
    DOUTB   => DOUTB(17 downto 0)
);

DRAM_i : SingleRam generic map (
    C_ADDR_WIDTH      => C_ADDR_WIDTH,
    C_DATA_WIDTH       => 2,
    C_RAM_PERFORMANCE => C_RAM_PERFORMANCE,
    C_RAM_STYLE        => "distributed"
) port map (
    CLKA    => CLKA ,
    WEA     => WEA  ,
    ADDRA   => ADDRA,
    DINA    => DINA(19 downto 18),
    CLKB    => CLKB ,
    ENB     => ENB  ,
    RSTB    => RSTB ,
    REGCEB  => REGCEB,
    ADDRDB  => ADDRDB,
    DOUTB   => DOUTB(19 downto 18)
);
```

参考设计文件

请从赛灵思网站下载与本附录关联的[参考设计文件](#)。

附加资源与法律提示

赛灵思资源

如需了解答复记录、技术文档、下载以及论坛等支持性资源，请参阅[赛灵思技术支持](#)。

解决方案中心

如需了解设计周期各阶段有关器件、软件工具和 IP 等的技术支持，请参阅[赛灵思解决方案中心](#)。相关专题包括设计辅助、建议和故障排除提示等。

Documentation Navigator 与设计中心

赛灵思 Documentation Navigator (DocNav) 提供了访问赛灵思文档、视频和支持资源的渠道，您可以在其中筛选搜索信息。打开 DocNav 的方法：

- 在 Vivado® IDE 中，单击“Help” → “Documentation and Tutorials”。
- 在 Windows 中，单击“Start” → “All Programs” → “Xilinx Design Tools” → “DocNav”。
- 在 Linux 命令提示中输入“docnav”。

赛灵思设计中心提供了根据设计任务和其他话题整理的文档链接，您可以使用链接了解关键概念以及常见问题解答。访问设计中心：

- 在 DocNav 中，单击“Design Hub View”标签。
- 在赛灵思网站上，查看[设计中心](#)页面。

注释：如需了解更多有关 DocNav 的信息，请参阅赛灵思网站上的 [Documentation Navigator](#)。

参考资料

以下技术文档是非常实用的补充资料，可配合本指南一起使用：

1. 《Vivado Design Suite 用户指南：使用 Vivado IDE》 ([UG893](#))
 2. 《Vivado Design Suite 用户指南：使用 Tcl 脚本》 ([UG894](#))
 3. 《Vivado Design Suite Tcl 命令参考指南》 ([UG835](#))
 4. 《Vivado Design Suite 用户指南：系统级设计输入》 ([UG895](#))
 5. 《Vivado Design Suite 用户指南：使用约束》 ([UG903](#))
 6. 《UltraFast 设计方法指南（适用于 Vivado Design Suite）》 ([UG949](#))
 7. 《Vivado Design Suite 用户指南：实现》 ([UG904](#))
 8. 《Vivado Design Suite 用户指南：功耗分析与最优化》 ([UG907](#))
 9. 《7 系列 FPGA 时钟资源用户指南》 ([UG472](#))
 10. 《Vivado Design Suite 属性参考指南》 ([UG912](#))
 11. [所有 Vivado Design Suite 文档](#)
-

培训资料

赛灵思提供多种多样的培训课程和 QuickTake 视频，可帮助用户进一步了解有关本文档中提出的概念。使用以下链接获取相关培训资料：

1. [使用 Vivado Design Suite 设计 FPGA 1](#)
 2. [使用 Vivado Design Suite 设计 FPGA 2](#)
 3. [使用 Vivado Design Suite 设计 FPGA 3](#)
 4. [使用 Vivado Design Suite 设计 FPGA 4](#)
 5. [Vivado Design Suite QuickTake 视频教程](#)
 6. [Vivado Design Suite QuickTake 视频：高级时钟约束与分析](#)
 7. [Vivado Design Suite QuickTake 视频：分析实现结果](#)
 8. [Vivado Design Suite QuickTake 视频：时序分析控制](#)
 9. [Vivado Design Suite QuickTake 视频：跨时钟域检查 - CDC 分析](#)
-

请阅读：重要法律提示

本文向贵司/您所提供的信息（下称“资料”）仅在对赛灵思产品进行选择和使用时参考。在适用法律允许的最大范围内：（1）资料均按“现状”提供，且不保证不存在任何瑕疵，赛灵思在此声明对资料及其状况不作任何保证或担保，无论是明示、暗示还是法定的保证，包括但不限于对适销性、非侵权性或任何特定用途的适用性的保证；且（2）赛灵思对任何因资料发生的或与资料有关的（含对资料的使用）任何损失或赔偿（包括任何直接、间接、特殊、附带或连带损失或赔偿，如数据、利润、商誉的损失或任何因第三方行为造成的任何类型的损失或赔偿），均不承担责任，不论该等损失或者赔偿是何种类或性质，也不论是基于合同、侵权、过失或是其他责任认定原理，即便该损失或赔偿可以合理预见或赛灵思事前被告知有发生该损失或赔偿的可能。赛灵思无义务纠正资料中包含的任何错误，也无义务对资料或产

品说明书发生的更新进行通知。未经赛灵思公司的事先书面许可，贵司/您不得复制、修改、分发或公开展示本资料。部分产品受赛灵思有限保证条款的约束，请参阅赛灵思销售条款：<https://china.xilinx.com/legal.htm#tos>；IP 核可能受赛灵思向贵司/您签发的许可证中所包含的保证与支持条款的约束。赛灵思产品并非为故障安全保护目的而设计，也不具备此故障安全保护功能，不能用于任何需要专门故障安全保护性能的用途。如果把赛灵思产品应用于此类特殊用途，贵司/您将自行承担风险和责任。请参阅赛灵思销售条款：<https://china.xilinx.com/legal.htm#tos>。

关于与汽车相关用途的免责声明

如将汽车产品（部件编号中含“XA”字样）用于部署安全气囊或用于影响车辆控制的应用（“安全应用”），除非有符合 ISO 26262 汽车安全标准的安全概念或冗余特性（“安全设计”），否则不在质保范围内。客户应在使用或分销任何包含产品的系统之前为了安全的目的全面地测试此类系统。在未采用安全设计的条件下将产品用于安全应用的所有风险，由客户自行承担，并且仅在适用的法律法规对产品责任另有规定的情况下，适用该等法律法规的规定。

版权声明

© Copyright 2019 赛灵思公司版权所有。Xilinx、赛灵思标识、Alveo、Artix、Kintex、Spartan、Versal、Virtex、Vivado、Zynq 本文提到的其它指定品牌均为赛灵思在美国及其它国家的商标。“PCI”、“PCIe”和“PCI Express”均为 PCI-SIG 拥有的商标，且经授权使用。所有其它商标均为各自所有方所属财产。