查看所有版本

Vivado Design Suite 用户指南

设计分析与收敛技巧

UG906 (v2024.2) 2024 年 12 月 19 日

目录

第 1 章

简介

本文档涵盖了如何驱动 AMD Vivado™ Design Suite 来分析和改善您的设计，其中详解了下列主题：

- 使用 Vivado 集成设计环境 (IDE) 来查看消息、设计网表和交叉探测
- 方法论和 DRC 豁免
- 分析时序报告
- 生成所有网表、时序和设计收敛报告
- 智能设计运行、QoR 建议和 ML 策略

按设计进程浏览内容

AMD 自适应计算文档按一组标准设计进程进行组织，以便帮助您查找当前开发任务相关的内容。您可以在设计中心页面上访问 AMD Versal™ 自适应 SoC 设计进程。您还可以使用设计流程助手来更深入了解设计流程，并找到特定于预期设计需求的内容。本文档涵盖了以下设计进程：

- 硬件、IP 和平台开发：为硬件平台创建 PL IP 块、创建 PL 内核、功能仿真以及评估 AMD Vivado™ 时序收敛、资源使用情况和功耗收敛。还涉及为系统集成开发硬件平台。

第 2 章

IDE 中的交互设计分析

以下章节提供了 AMD Vivado™ Design Suite IDE 中的设计分析简介。IDE 与依赖工程模式的用户密切相关，但其中许多功能特性也适用于依赖非工程模式的用户。本章涵盖了下列技巧：

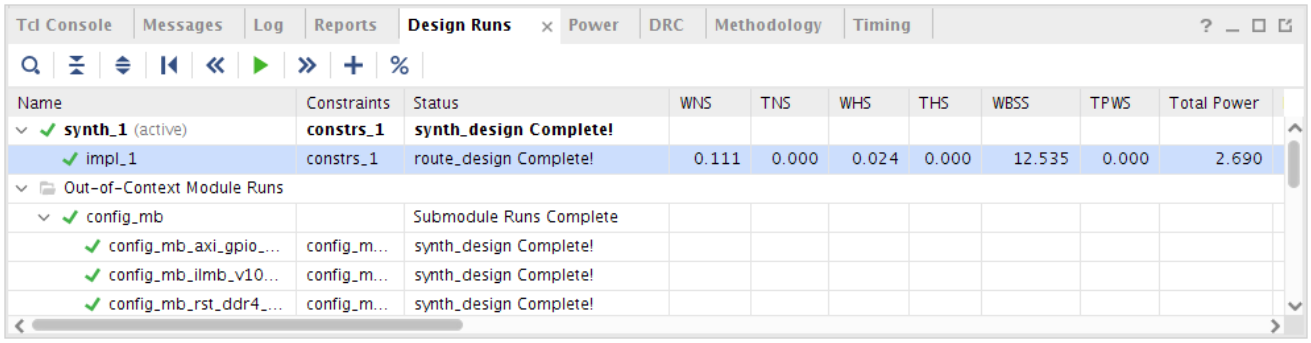
- 使用“Design Runs”（设计运行）窗口进行快速分析
- 在不同窗口之间进行交叉探测
- 方法论分析
- 布局布线分析

使用“Design Runs”窗口

“Design Runs”（设计运行）窗口是很实用的分析起点。它可显示当前运行状态。如需了解更多信息，请参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的“使用‘Schematic’窗口”。

“Design Runs”窗口显示运行状态：“running”（正在运行）、“finished cleanly”（无错完成）或“finished with errors”（已完成但有错误）。

图 1： **“Design Runs”窗口**



“Design Runs”窗口列显示如下内容：

- 运行名称。
- 目标部件。
- 与运行关联的约束集。
- 运行策略。
- 运行的最后完成步骤的状态。
- 运行进度。

- 运行的开始时间。
- 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 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“Timing Summary”（时序汇总）报告启动分析。

  注释：WBSS 表示 `report_bus_skew` 报告的“Worst Bus Skew Slack”（最差总线偏差时序裕量）。
- 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 的使用率。
- 总功耗估算。
- 运行策略简介。
- 方法论检查违例。
- 可用的 QoR 建议。
- 设计运行的增量模式。

选择运行时，右键单击菜单允许执行各项操作，例如，打开运行以便进一步分析，设置运行功能特性（如增量编译或 QoR 建议）。

IDE 中的逻辑分析

逻辑分析特性

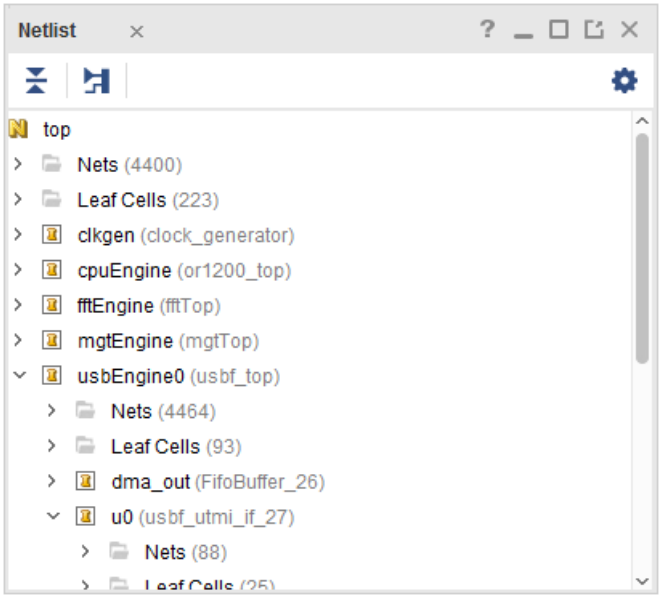
本章讨论逻辑分析特性，包括：

- 使用“Netlist”窗口
- 使用“Hierarchy”窗口
- 使用“Schematic”窗口
- 使用“Find”对话框搜索对象
- 分析使用率统计数据
- 使用 DRC 报告
- 方法论分析

使用“Netlist”窗口

“Netlist”（网表）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 2：**“Netlist”窗口**



根据综合设置，网表层级与原始 RTL 可能匹配，也可能不存在层级。通常，默认情况下综合在对逻辑进行优化时会保留大部分用户层级。这会为实现生成最优的网表。

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。在跨层级边界进行优化后，可能会丢失某些管脚和层级。

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

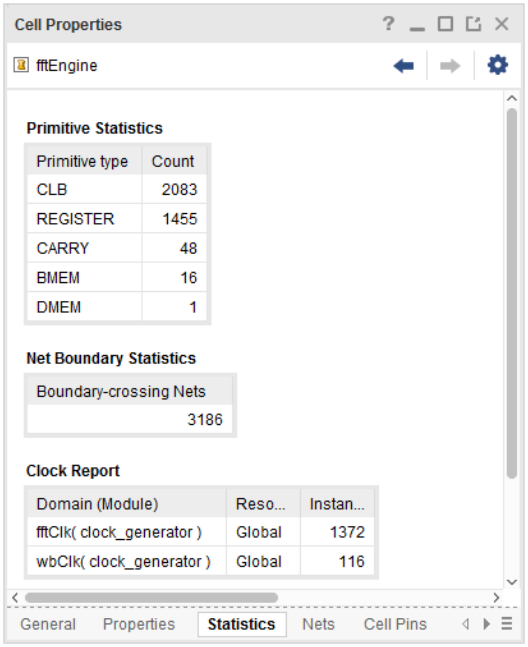
展开“hierarchy”文件夹即可显示该层次的“Nets”、“Leaf Cells”和“hierarchies”。单元旁的图标可显示有关设计状态的信息。

如需了解更多信息，请参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的“使用‘Netlist’窗口”。

选定“hierarchy”的“Cell Properties”（单元属性）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“Statistics”（统计数据）按钮可显示使用率统计数据，包括：

- 整个层级分支的原语使用率，按更高层次存储桶加以分组
- 跨层级边界的信号线数量
- 每个时钟，包括该时钟是否位于全局布线上，以及该时钟在当前层级分支中的负载

图 3： **“Cell Properties”窗口**



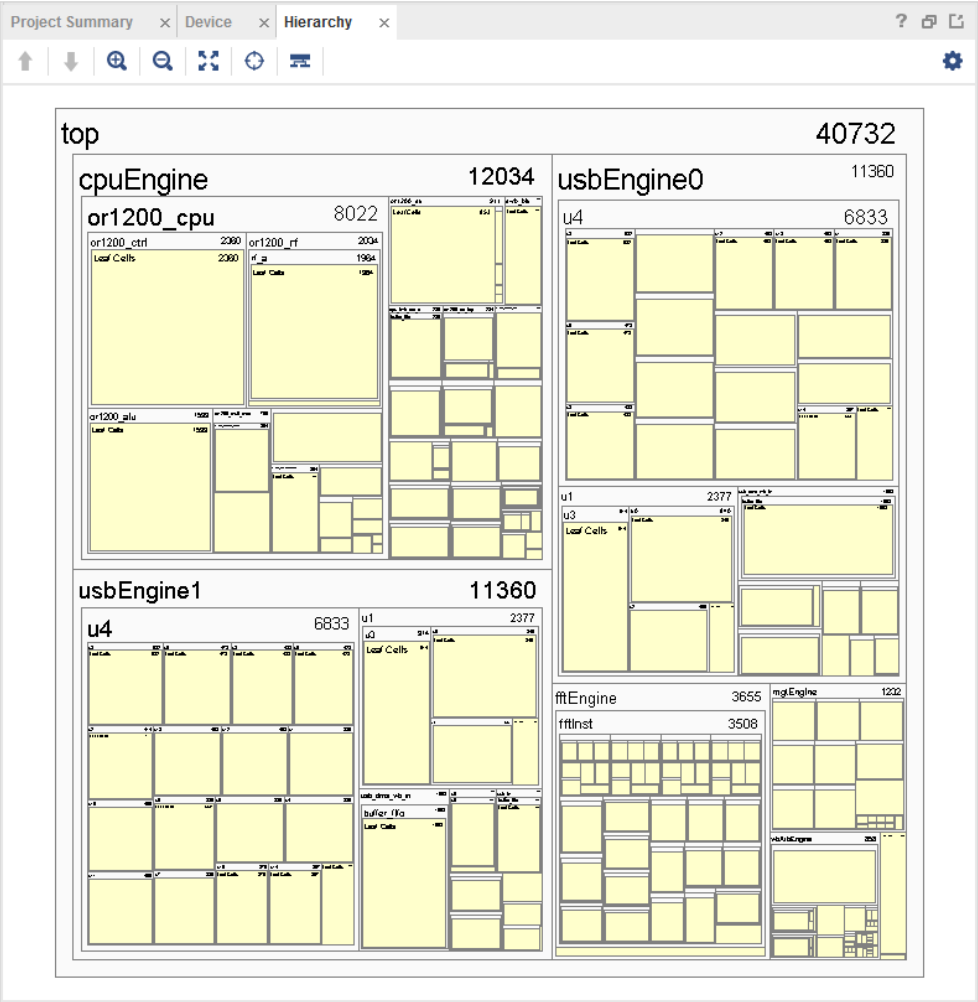
如对设计进行布局规划，那么针对 Pblock 将显示类似的属性。

使用“Hierarchy”窗口

此窗口用于以物理方式浏览层级以了解资源使用率。要打开“Hierarchy”（层级）窗口，请选择“Tools > Show Hierarchy”（工具 > 显示层级），或者从“Netlist”（网表）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 4：“Hierarchy”窗口



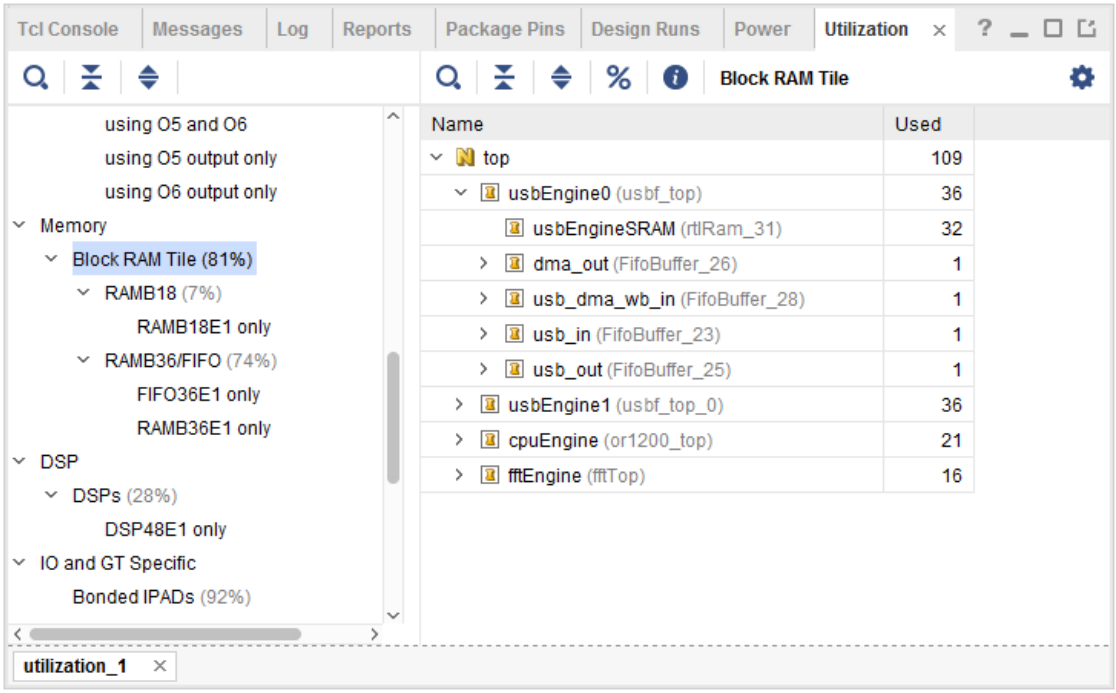
上图显示 `cpuEngine`、`usbEngine0` 和 `usbEngine1` 包含设计中的大部分逻辑，并且使用的资源数量都几乎相同。

使用“Utilization”报告

“Utilization Report”（使用率报告）可根据资源类型细分设计使用率。左侧面板提供按资源类型汇总的使用率，右侧面板按层级显示使用率。

要查看“Utilization Report”，请选择“Reports”→“Report Utilization”（报告 > 使用率报告）。下图显示了“Utilization Report”。

图 5：**Utilization Report**



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率。

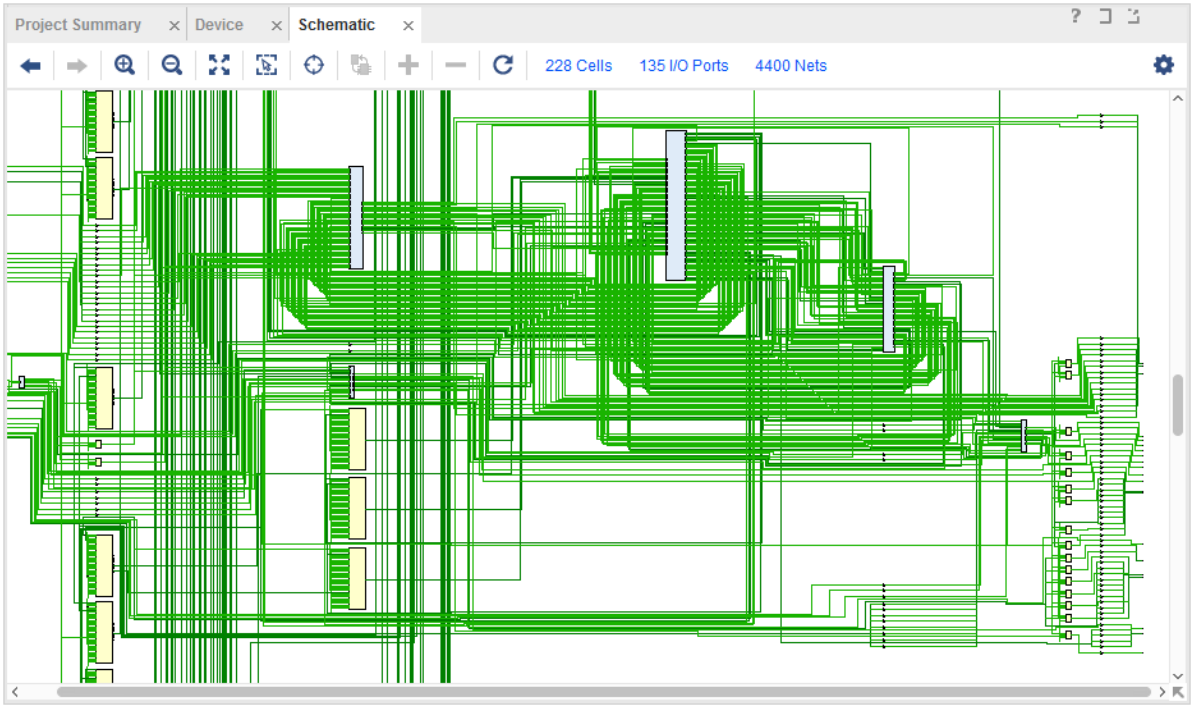
使用“Schematic”窗口

“Schematic”（原理图）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。
- 追踪并扩展逻辑椎。
- 分析设计。
- 更准确了解设计内部发生的状况。

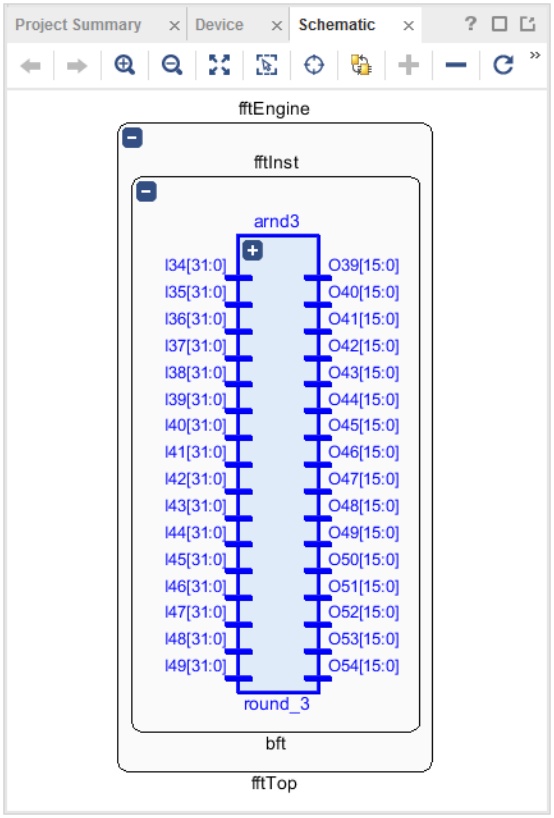
在“Elaborated Design”（细化设计）中的 RTL 层级上可查看工具对代码的解释。在“Synthesize Design”（综合设计）和“Implemented Design”（实现设计）中，可查看由综合工具生成的网表。要打开原理图，请选择“Tools > Schematic”（工具 > 原理图）。如果不选择任何项，则各单元、层级和连接会显示在设计顶层，如下图所示。

图 6：**顶层原理图**



提示：如果您首先选择单元、信号线、管脚或端口对象，然后再创建原理图，那么会创建一个更简单的原理图，其中仅包含选中的对象。

图 7：**含选定单层级的原理图**



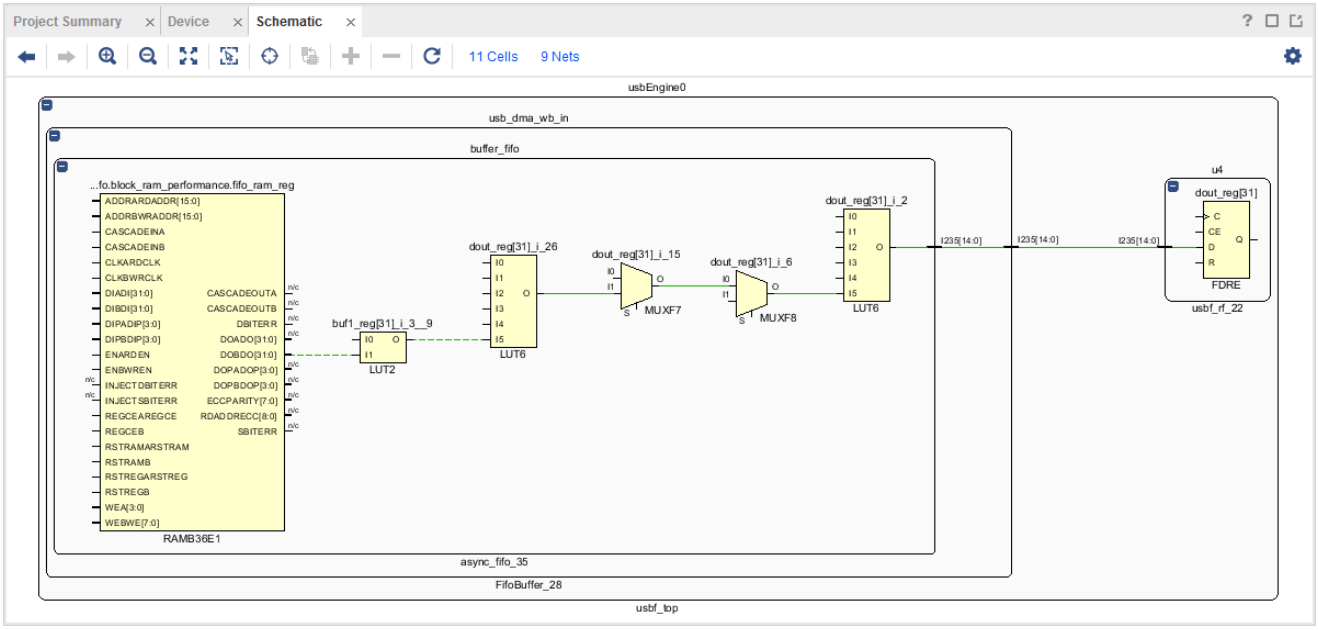
您可通过多种方式来追踪原理图：

- 单击左上方“+”（加号）图标，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，切换上一个与下一个原理图视图。

如需了解有关原理图的更多信息，请参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的“使用‘Schematic’窗口”。

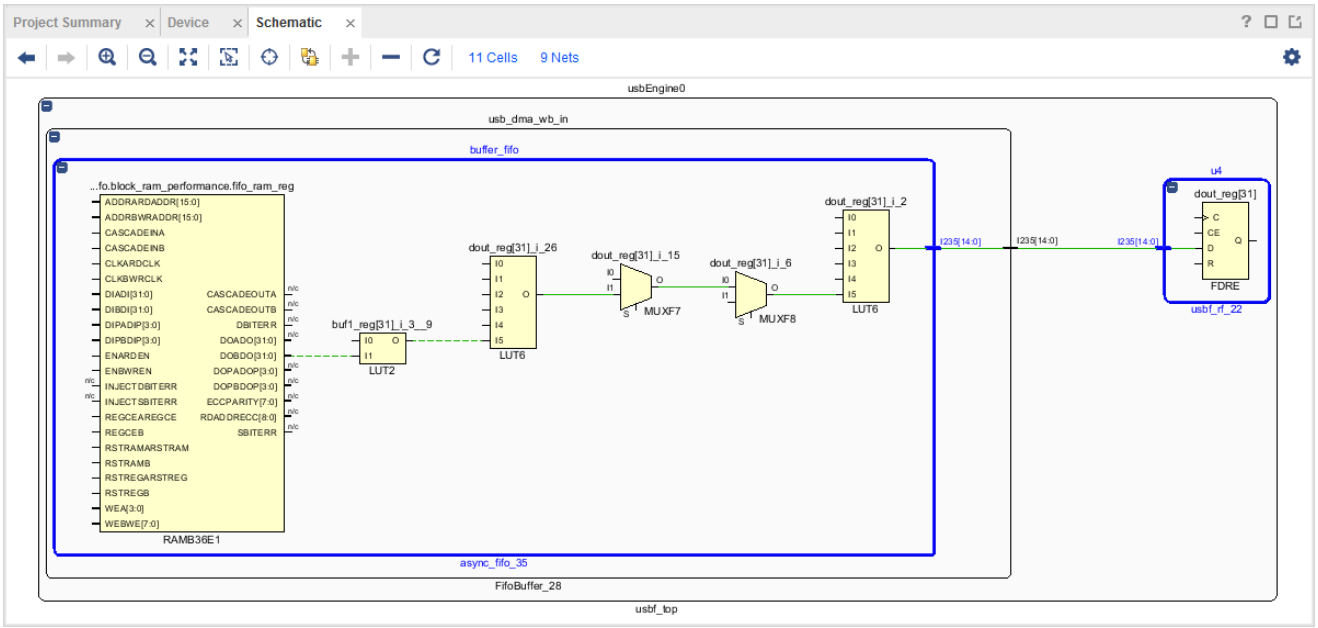
完成实现后，原理图是将时序路径中的单元可视化的最简单方法。选择路径，然后即可打开原理图，其中包含来自该路径的单元和信号线。

图 8：**含时序路径的原理图**



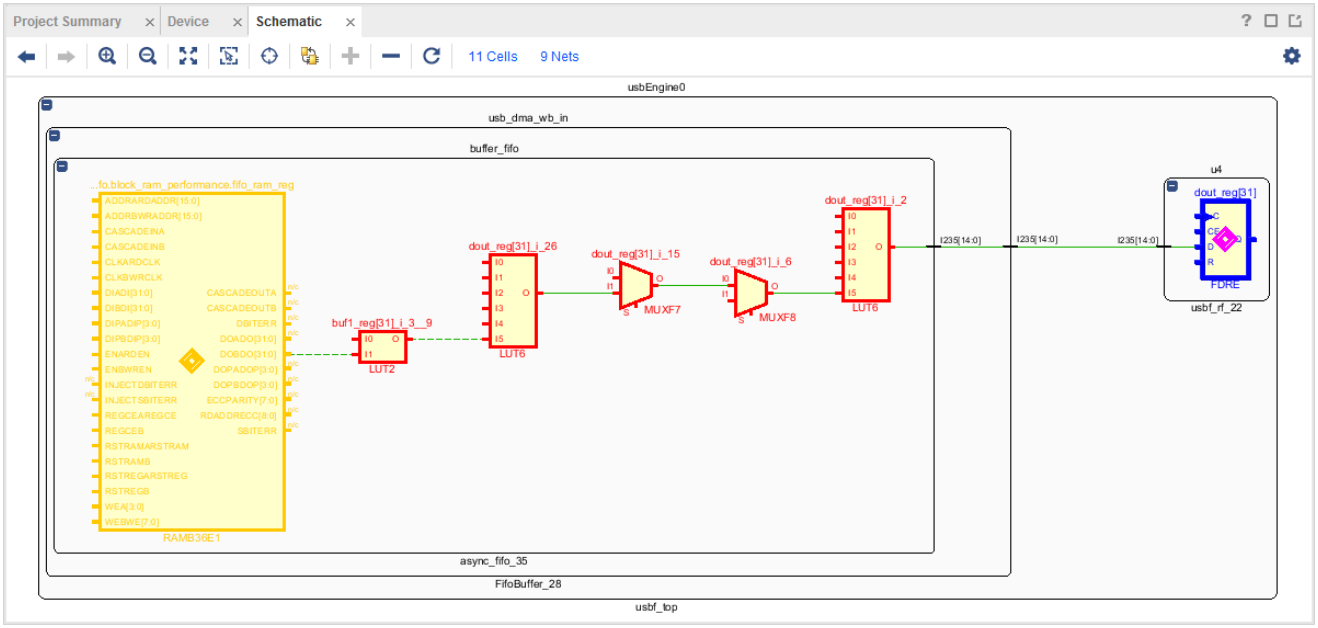
若要识别原理图中选定单元的相关层级分级，请从弹出菜单中选择“Select Leaf Cell Parents”（选择叶节点单元的父单元）。

图 9：**含选定父级原语的时序路径**



检查原理图时，选择“Highlight”（高亮）以及“Mark”（标记）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）以便于跟踪来自原始路径的逻辑和添加的逻辑。

图 10：**含时序路径标记的原理图**

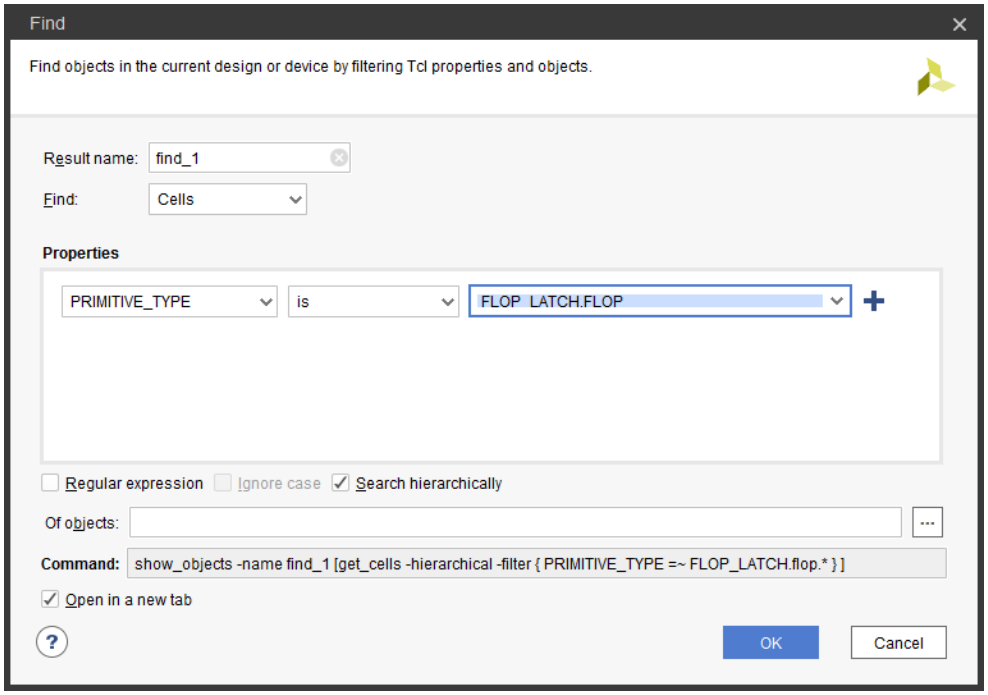


使用“Find”对话框搜索对象

AMD Vivado™ IDE 包含强大的查找和搜索功能。要打开“Find”（查找）对话框，请选择“Edit” → “Find”（编辑 > 查找）。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 11：**“Find”对话框**



查找条件

“Find”（查找）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 12：**显示搜索条件的“Find”对话框**

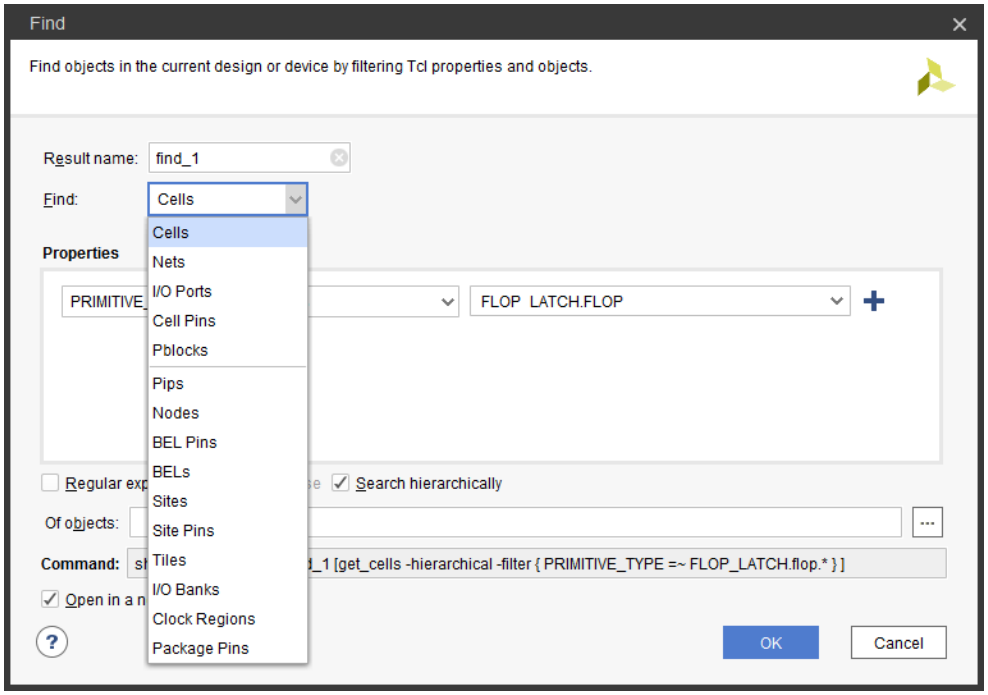
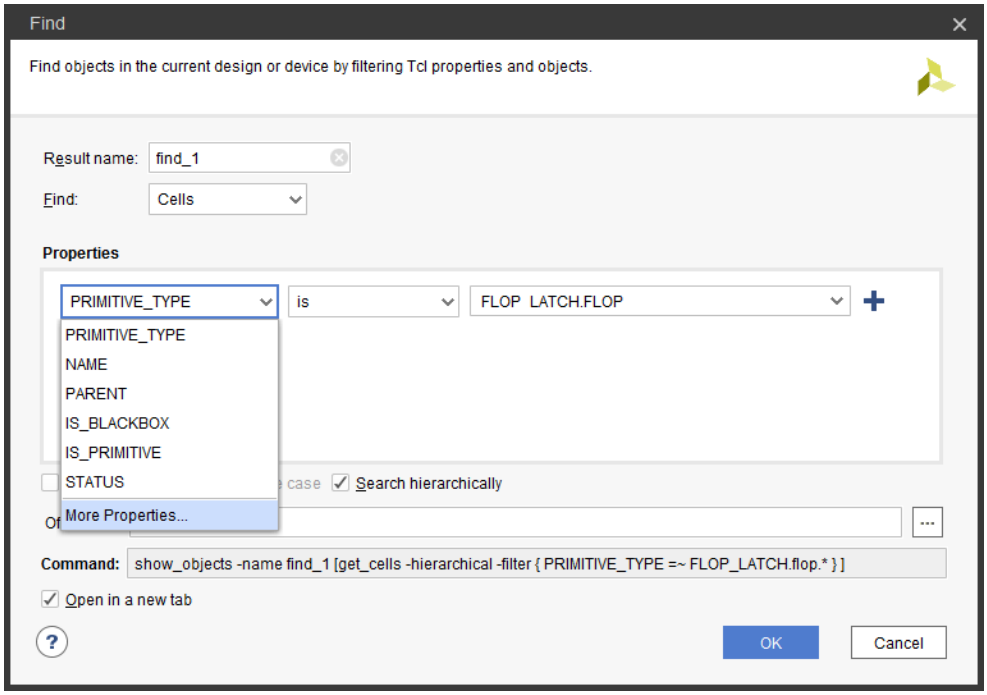


图 13：**显示“Properties”选项的“Find”对话框**



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“Properties”（属性）下拉选项旁边的“+”（加号）。
3. 添加其他搜索条件。
4. 将其他条件与逻辑运算符（AND、OR）配合一起使用。

查找示例

选择“Edit”→“Find”（编辑 > 查找）以执行查找，例如：

- 所有未布局 I/O：

  “Find:”Cells，“Properties:”Primitive is IO + AND STATUS is UNPLACED

- 所含扇出超出 10,000 的所有信号线：

  “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 AMD 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`。

方法论分析

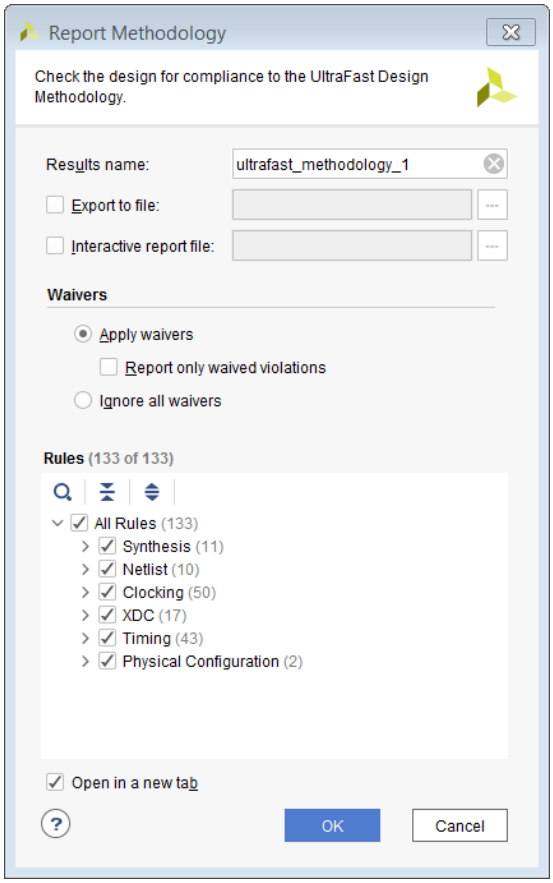
鉴于方法论的重要性，AMD Vivado™ 工具提供了 `report_methodology` 命令，专用于检查方法论 DRC 的合规性。DRC 类型因设计进程阶段而异。RTL lint 样式的检查在细化的 RTL 设计上运行；基于网表的逻辑和约束检查在综合后设计上运行；而实现和时序检查则在实现后设计上运行。

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

```
report_methodology
```

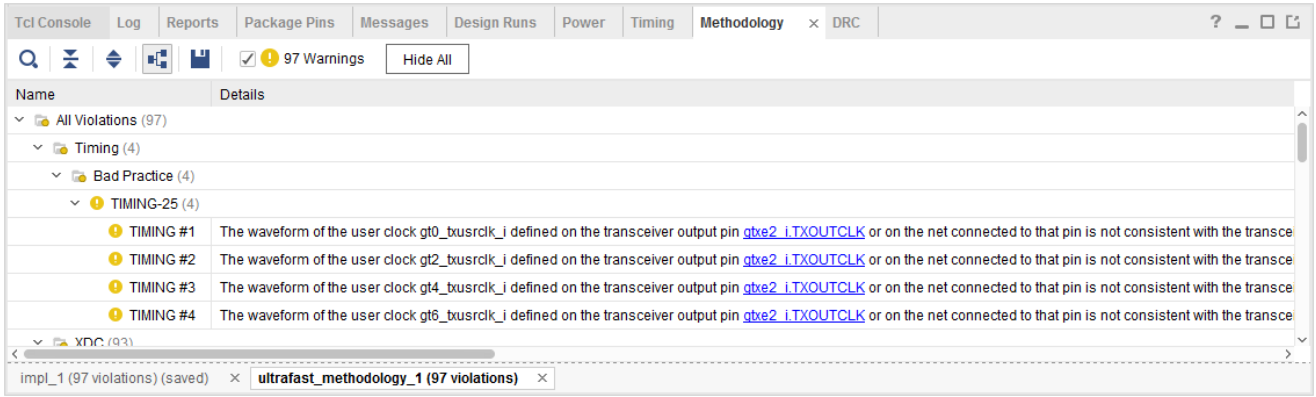
要从 IDE 运行上述检查，请打开待确认设计，然后在工程模式下的 Flow Navigator 中运行 Report Methodology 命令，或通过“Reports” → “Report Methodology”（报告 > 方法论报告）来运行该命令。这样将显示如下图所示对话框。

图 14：**“Report Methodology”对话框**



在“Methodology”（方法论）窗口中会列出所有违例（如有），如下图所示。

图 15：**方法论违例**



如需了解有关运行设计方法论 DRC 的更多信息，请参阅《Vivado Design Suite 用户指南：系统级设计输入》(UG895) 中的“运行方法检查”。

注释：建议解决所有方法论违例，尤其要重点关注“Critical Warning”（严重警告），因为它们会影响时序收敛和验收质量。

布局分析

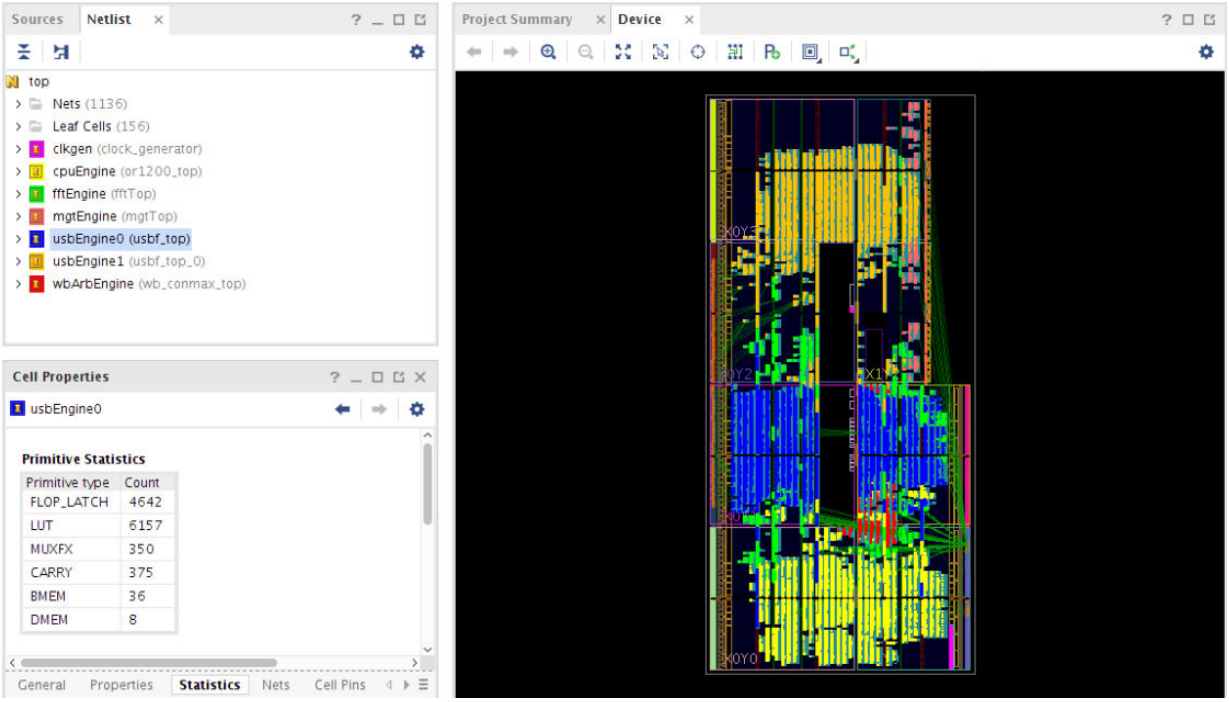
高亮显示布局

审查设计布局的另一途径是分析单元布局。“Highlight Leaf Cells”（高亮叶节点单元）命令有助于开展此项分析。

1. 在“Netlist”（网表）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”（高亮叶节点单元 > 选择颜色）。
3. 如果要选择多个层级，请选择“Cycle Colors”（周期颜色）。

在“Device”（器件）窗口中，构成层级单元的叶节点单元会以颜色编码标示。

图 16：**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。例如，您可观测 `usbEngine0` 的下列属性（蓝色）：

- 它使用一定数量的块 RAM 和 DSP48 单元。
- 它位于芯片的中央时钟区域。
- 它与设计内的其他逻辑 (`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。

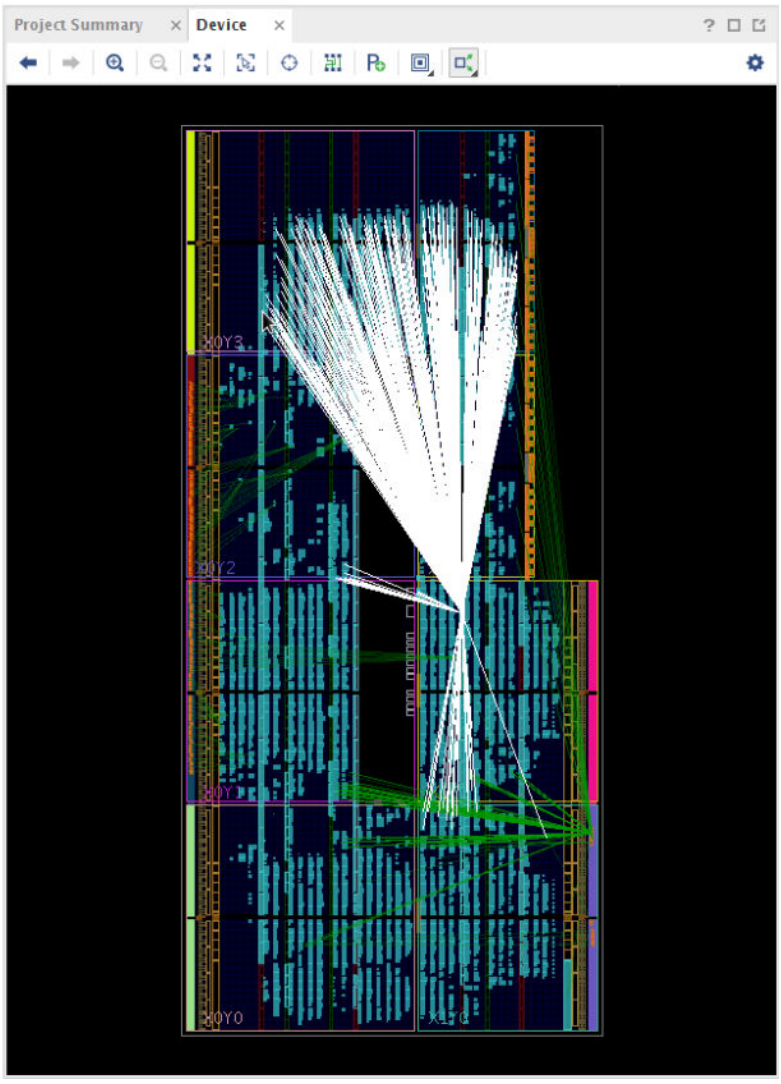
显示连接

根据连接来分析设计是很实用的方法。运行“Show Connectivity”（显示连接）复查受输入、块 RAM 或 DSP bank 驱动的所有逻辑的布局。“Show Connectivity”首先会选择单元或信号线，然后通过添加已连接的单元或信号线来扩展选择。

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。综合编译指示会在存储器推断期间阻止综合在块 RAM 内对输出触发器进行布局。

图 17：**显示连接**



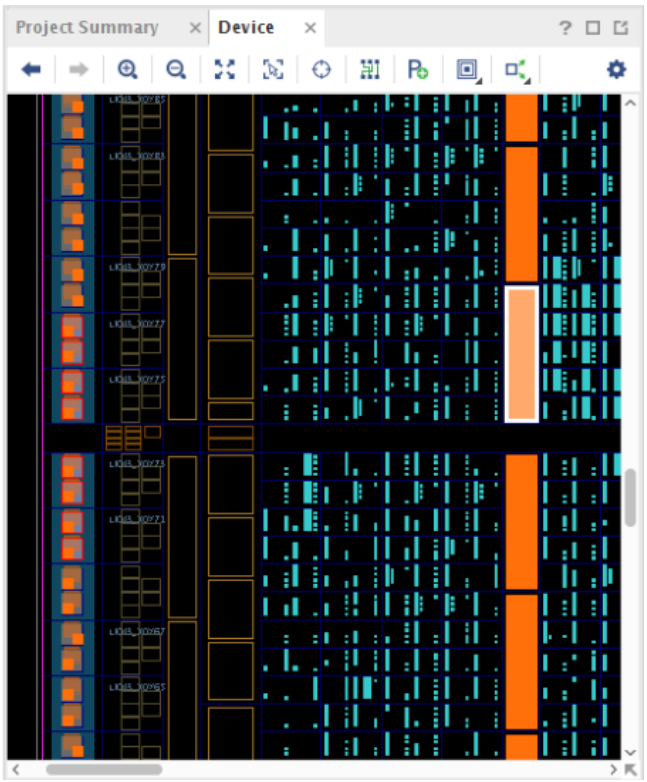
固定逻辑和非固定逻辑

AMD 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 18：**固定布局和非固定布局**



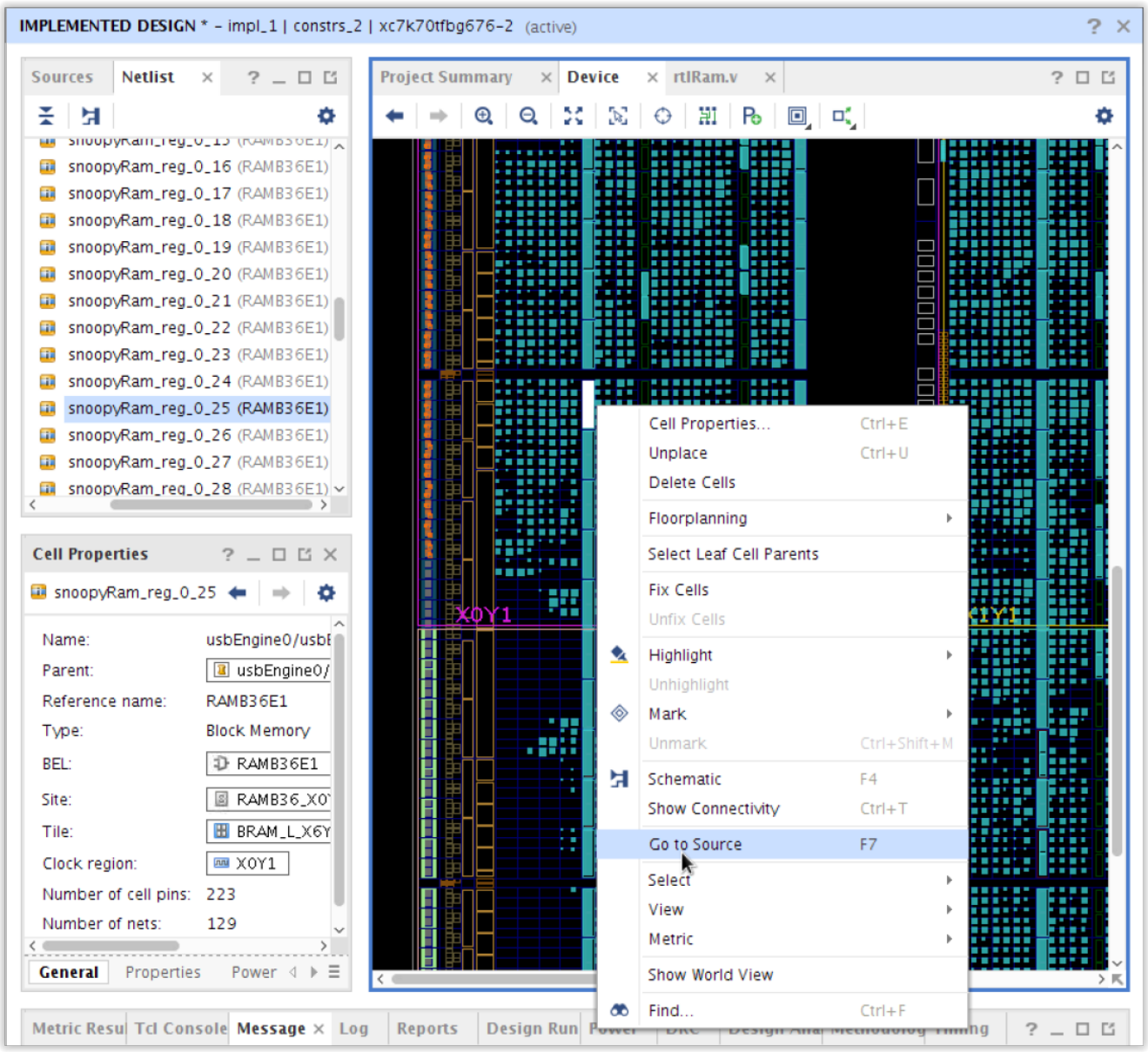
对 RTL 源文件进行交叉探测

对于使用 Vivado 综合进行综合的设计，当网表设计进入存储器后，即可对源文件进行反向交叉探测。

要执行交叉探测，请执行以下操作：

1. 选择门电路。
2. 在弹出菜单中选择“Go to Source”（转至源文件），如下图所示。

图 19：**向源文件进行反向交叉探测**



使用交叉探测判定网表门电路中涉及的源文件。由于综合变换的性质，无法对设计中每个门电路的源文件进行反向交叉探测。

查看指标

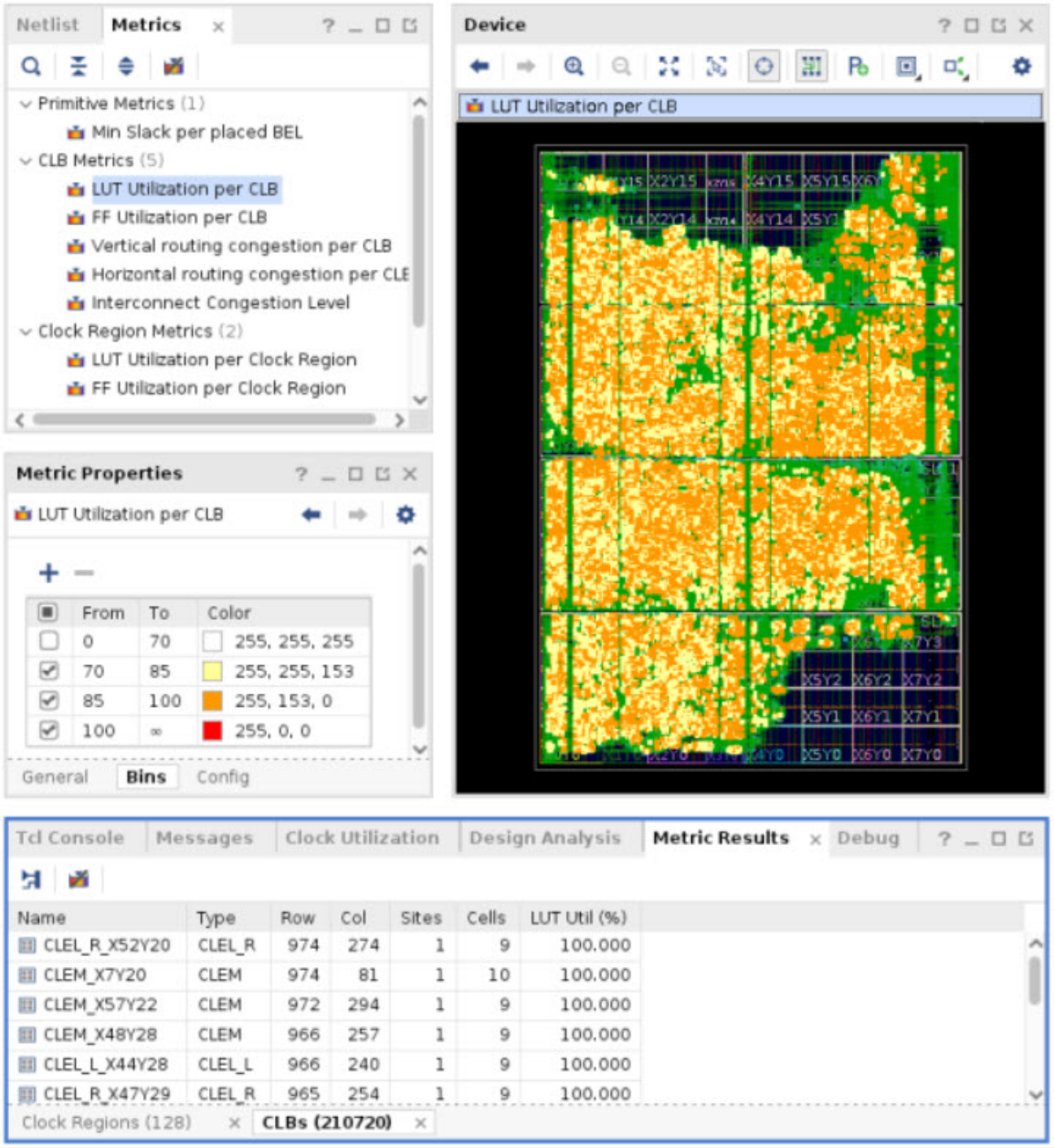
Vivado Design Suite 提供了各项指标，以便告知您有关器件内部的逻辑和布线的信息。这些指标提供了一种交互式的设计分析方法，以取代静态报告。

要激活指标，请确保已打开设计。选择“Window” → “Metrics”（窗口 > 指标）。要启用指标，请右键单击它并选择“Show”（显示）。可一次性添加多个指标。要禁用指标，请右键单击它并选择“Hide”（隐藏）以将其从“Device”（器件）视图中移除。选择“Hide All Metrics”（隐藏所有指标）即可单击隐藏所有已启用的指标。

下图显示了“LUT Utilization per CLB”（各 CLB 的 LUT 使用率）指标：

- “Metrics”（指标）窗口会显示可用指标列表，使用右键单击菜单即可显示这些指标。
- “Metric Properties”（指标属性）窗口显示了用于对给定指标进行分级的各种颜色分类。您可修改指标分类，包括启用或禁用各指标分类或者更改颜色。可添加新的分类，也可以移除现有分类。
- 在“Device”（器件）窗口中，指标分类被覆盖，以显示指标级别具有更高影响的物理位置。
- “Metric Results”（指标结果）窗口允许您搭配其他设计元素来选择并使用交叉-探测。

图 20：**“LUT Utilization per CLB”指标**



不含布局的网表设计中的指标

如果存在 Pblock，则适用以下指标。这些指标与布局无关。

- “LUT Utilization per Pblock”（各 Pblock 的 LUT 使用率）：该指标会根据 LUT 在 Pblock 所含 slice 中的布局方式的预测结果，来对 Pblock 进行颜色编码。
- “FF Utilization per Pblock”（各 Pblock 的 FF 使用率）：该指标根据在 Pblock 所含 slice 中对 FF 进行封装的方式预测，对 Pblock 进行颜色编码。

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 Utilization per CLB”（各 CLB 的 LUT 使用率）：基于已布局的 LUT 使用率对各 slice（分片）进行颜色编码。
- “FF Utilization per CLB”（各 CLB 的 FF 使用率）：基于已布局的 FF 使用率对各 slice 进行颜色编码。
- “Vertical Routing Congestion per CLB”（各 CLB 的垂直布线拥塞）：基于垂直布线使用率的最佳情况估算对互连结构进行颜色编码。
- “Horizontal Routing Congestion per CLB”（各 CLB 的水平布线拥塞）：基于水平布线使用率的最佳情况估算对互连结构进行颜色编码。

对于 UltraScale+ 和更新的架构：

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

时序指标

时序指标能以物理表现形式来显示设计时序问题。根据穿过 BEL 的时序路径上的最差负时序裕量 (WNS) 值，对每个 BEL 都会加以颜色编码。

使用率指标

使用率指标表示已使用的资源百分比，按 CLB 或时钟区域使用情况来分级。这些指标可用于表示 LUT 和 FF 使用率。

拥塞指标

根据目标器件系列，应使用不同的拥塞指标。

7 系列和 UltraScale 器件系列的拥塞指标

对于 7 系列和 UltraScale 器件，下列指标使用与布线器的拥塞估算相似的方法：

- Vertical Routing Congestion per CLB（各 CLB 的垂直布线拥塞）
- Horizontal Routing Congestion per CLB（各 CLB 的水平布线拥塞）

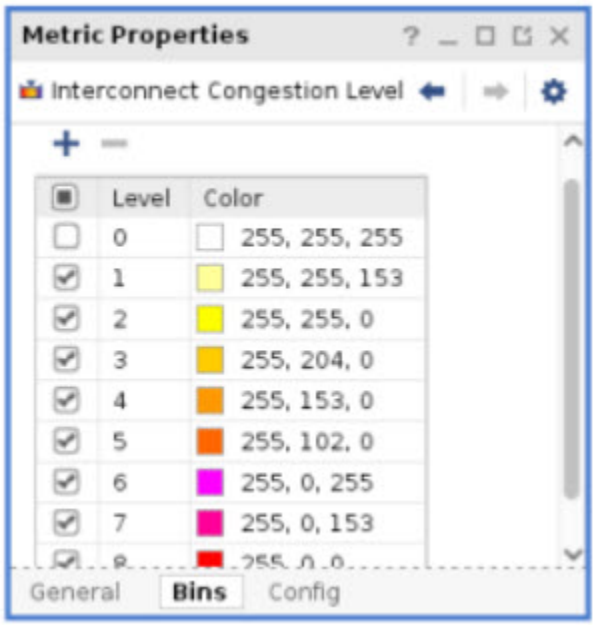
这两项指标都显示基于需求的模型，用于估算两个点之间的布线要求。当需求超过 100 时，会使用相邻拼块中的布线。拥塞与裸片内需求持续超过 100% 的面积成正比。

注释：虽然此方法对于新器件系列仍可用，但不建议使用。

UltraScale+ 和更新的器件系列的拥塞指标

对于 UltraScale+ 和更新的器件系列，请使用“Interconnect Congestion Level”（互连拥塞等级）指标。该指标简化了显示方式并改善了准确性，与布线器估算拥塞的方式更加匹配。该指标基于拥塞等级来进行颜色分类。

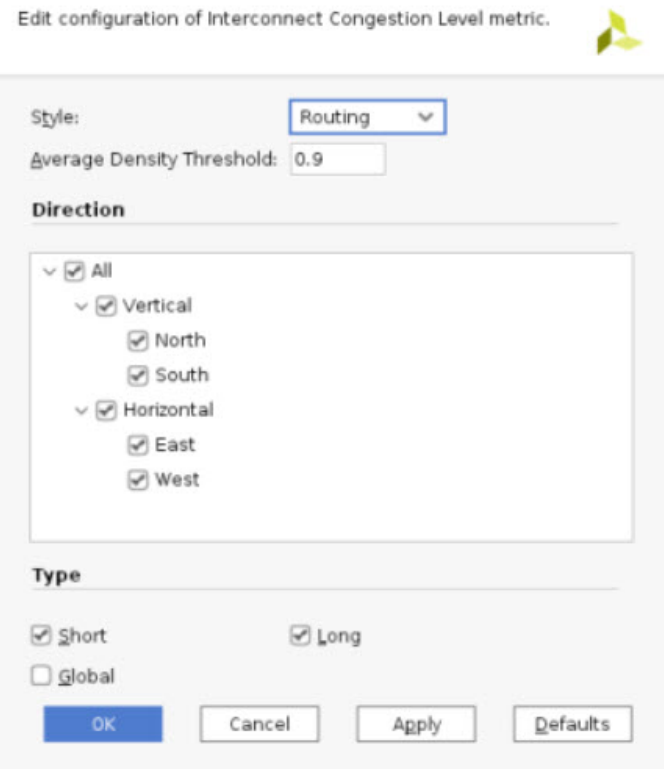
图 21：**互连拥塞等级指标属性**



在配置窗口中，指标视图可配置为：

- 显示估算的布线拥塞（影响布局器）和/或实际布线拥塞（影响布线器）
- 仅显示高于互连使用率阈值（0.9 = 90% 使用率）的项
- 下列特定布线类型的任意组合：“Short”（短）、“Long”（长）或“Global”（全局）
- 布线遍历的方向的任意组合

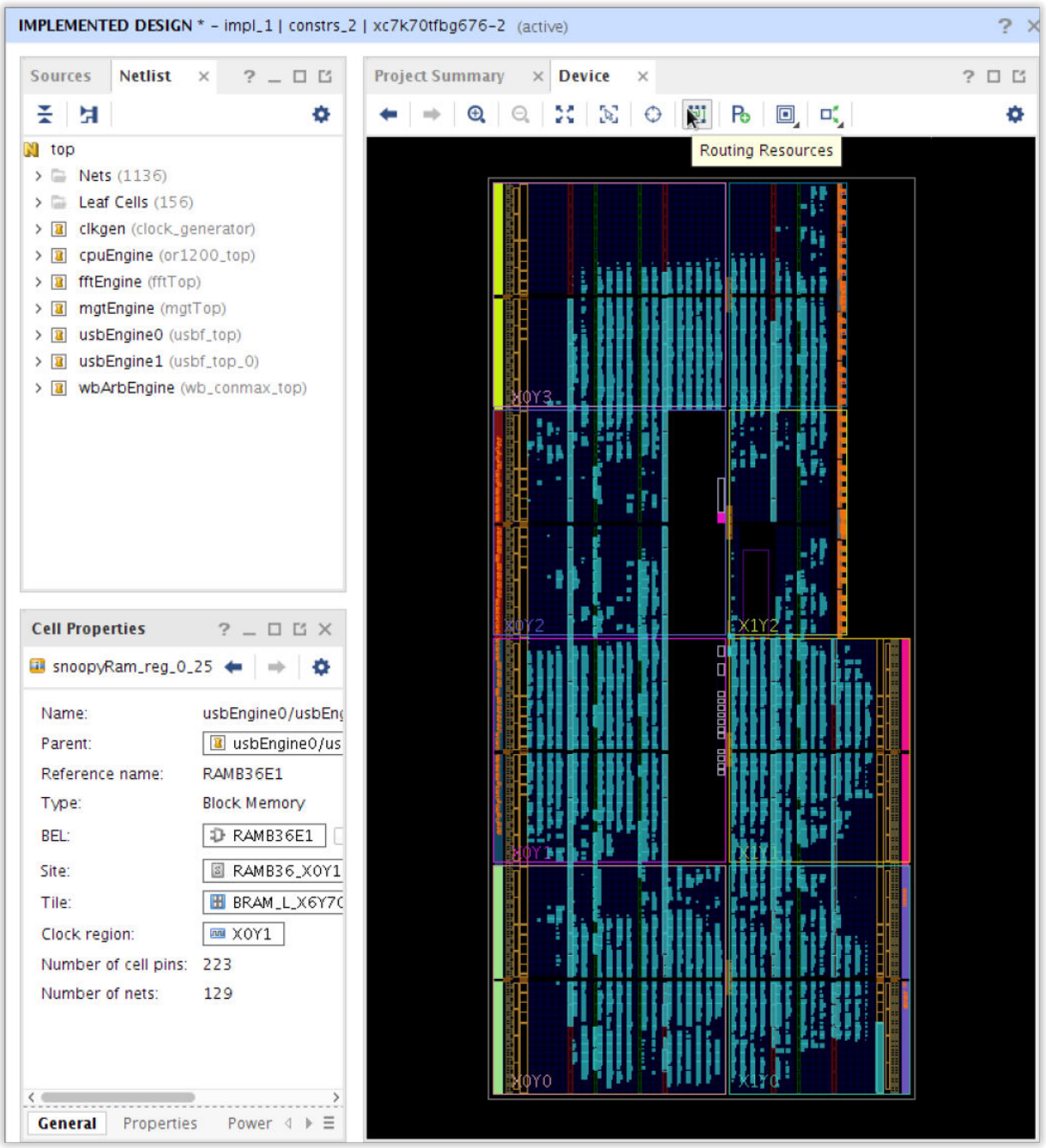
图 22：**互连拥塞等级配置**



布线分析

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

图 23：**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”（器件）窗口的 2 种可视化方式有利于最大限度降低运行时间和存储器占用，同时显示各种规模的设计的细节。

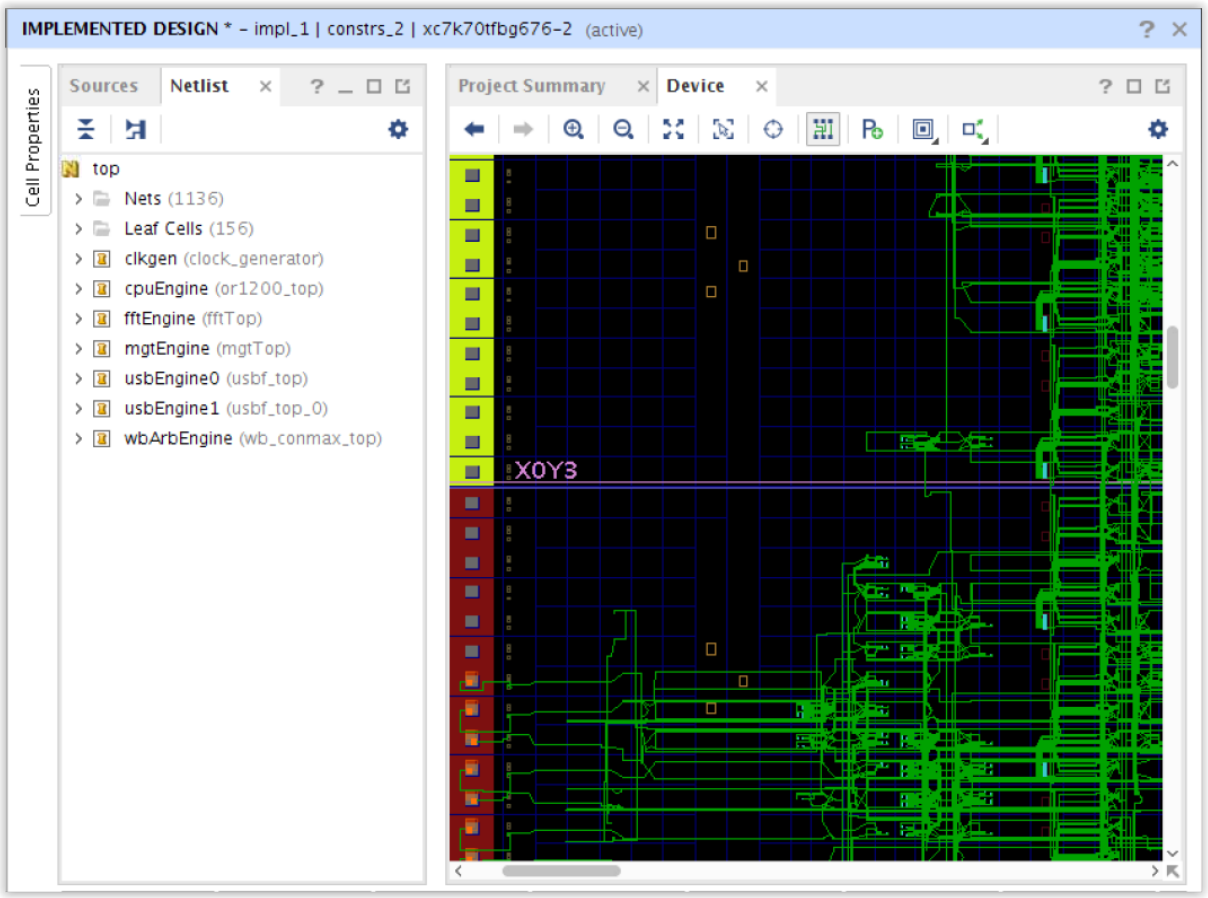
显示缩小的布局布线

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。

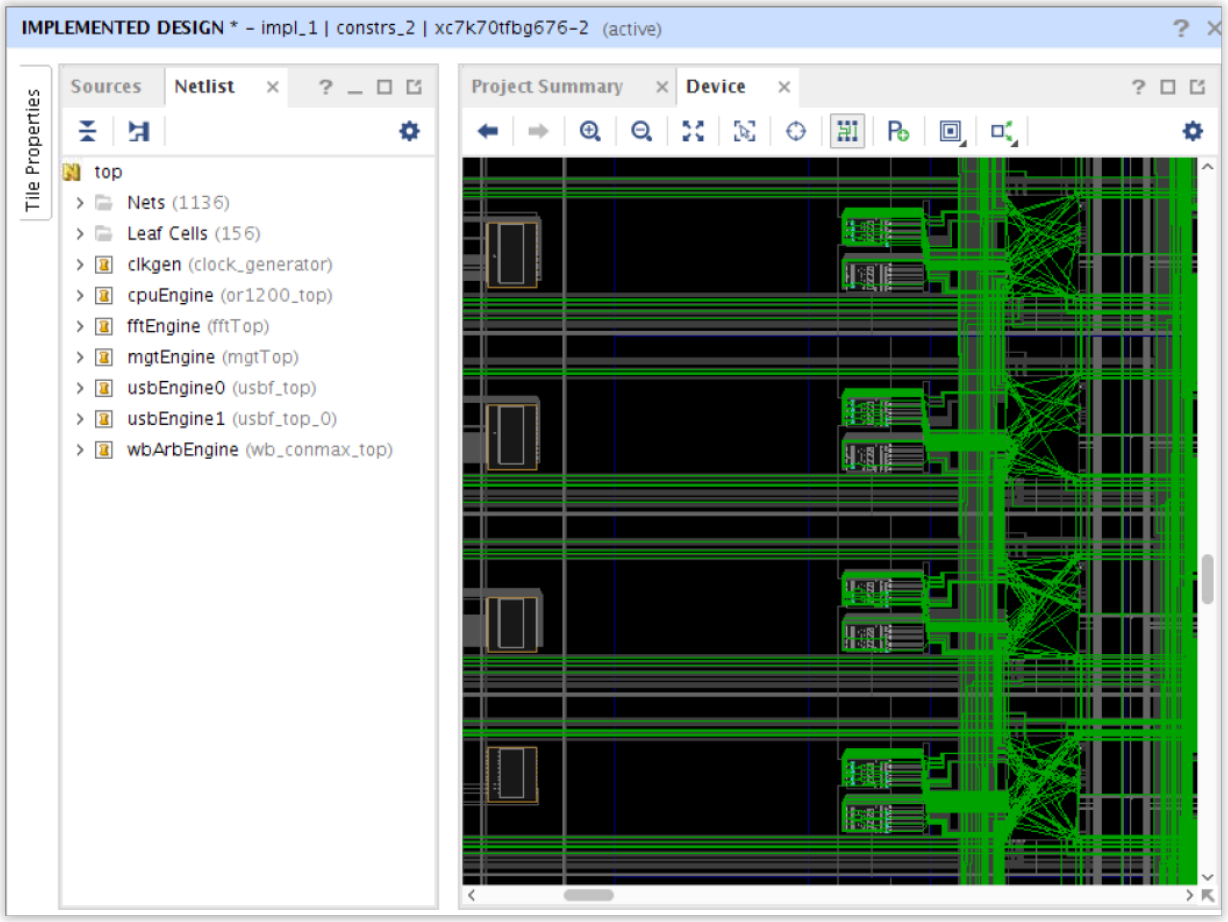
图 24：**抽象视图**



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。

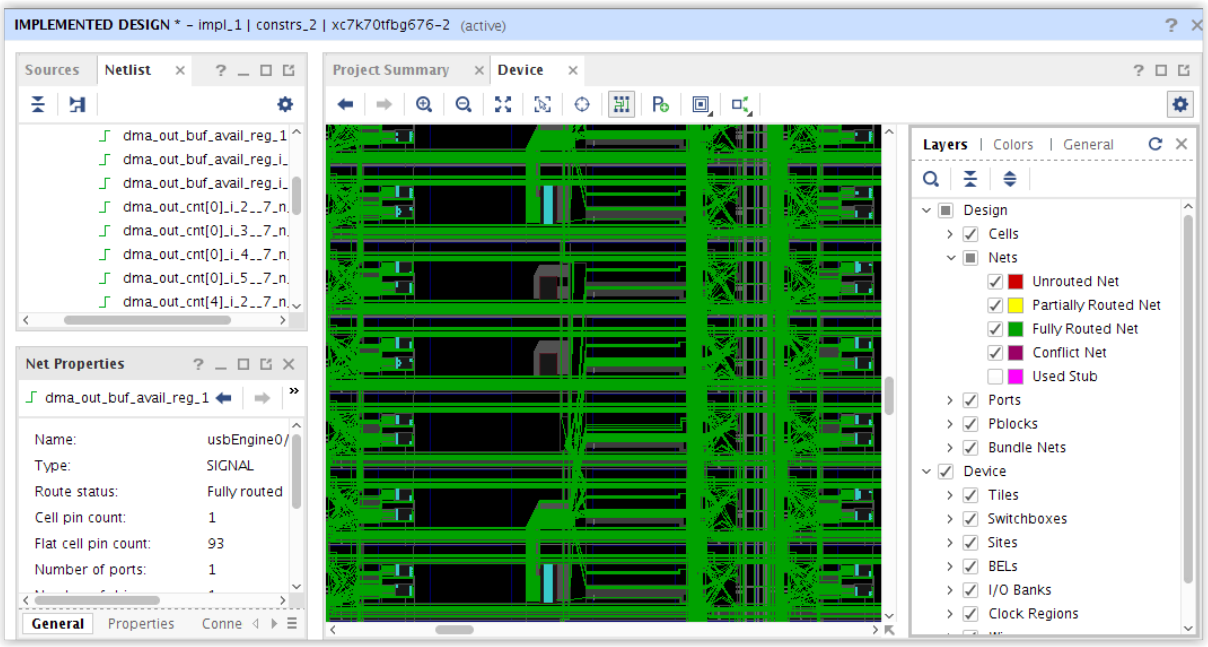
图 25：**详情视图**



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“Device Options”（器件选项）滑出菜单来控制。

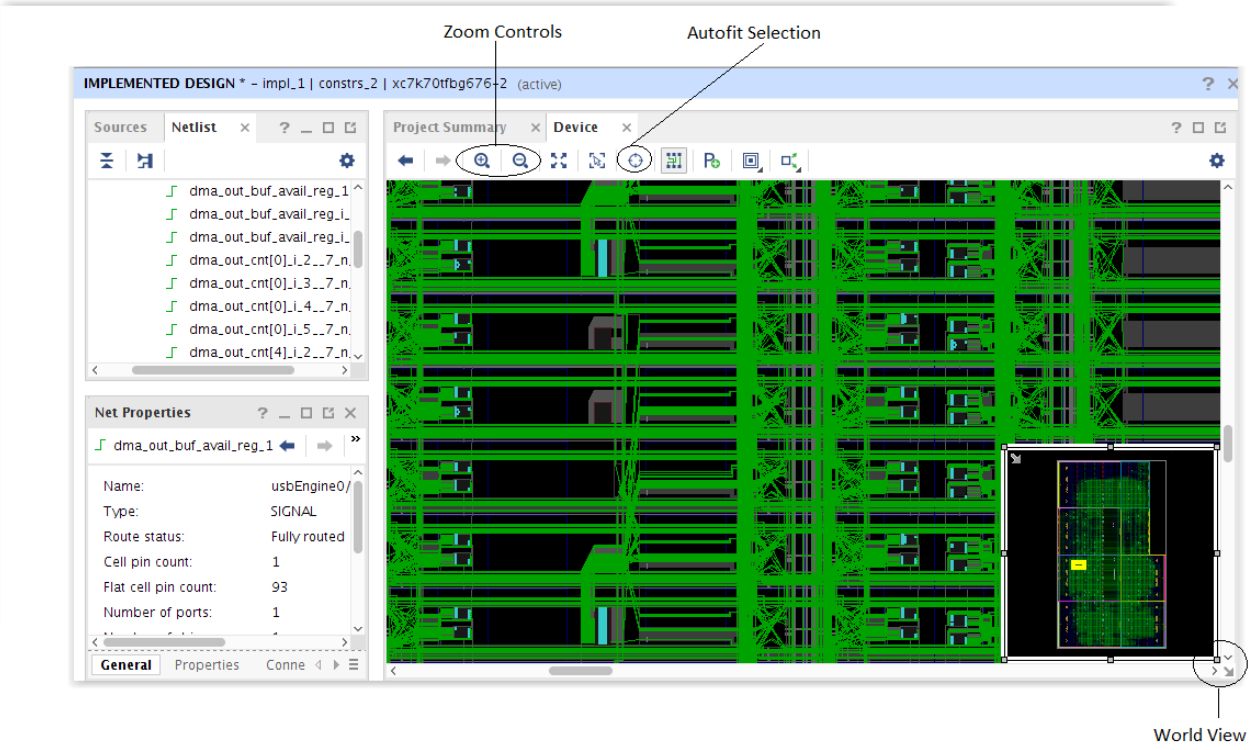
图 26：**“Device”窗口层次**



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 27：**浏览“Device”窗口**



使用以下工具浏览“Device”（器件）窗口：

- “Zoom Controls”（缩放控件）：标准“Zoom In”（放大）、“Zoom Out”（缩小）和“Zoom Full”（缩放至全屏）工具。
- “Auto-fit Selection”（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。“Autofit Selection”对交叉探测特别有用。
- “World”视图：“World”（全景）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。同时使用“Ctrl”和鼠标滑轮可在光标位置缩放视图。

数据流分析

由于高带宽总线存在宽度限制，因此总线的布局可能会对布线拥塞、时序和功耗产生影响。数据流分析旨在通过简化网表以实现轻松导航，同时保留关键布局锚点，从而帮助促进高带宽总线上的数据移动检查。

创建数据流设计时，会根据原始的完整设计网表生成一个精简的网表，并显示在数据流查看器中。该数据流网表会保留用户指定的宽度以上的总线信号线，并删去低于此阈值的总线信号线以及标量信号线和更低级别的单元。

创建数据流设计

要创建数据流设计，请使用以下任一选项：

- 在流菜单中，选中“Open Dataflow Design”（打开数据流设计）。对于该选项，必须有一个检查点或运行已处于打开状态。
- 在 Flow Navigator 中，选中“Open Synthesized Design”（打开已综合的设计）或“Open Implemented Design”（打开已实现的设计）下的“Open Dataflow Design”。该选项会根据需要妥善完成数据流。网表会基于处于活动状态的运行来生成。
- 在 Tcl 接口中，使用以下 Tcl 命令：`create_dataflow_design`。必须先打开设计，然后才能发出此命令。

图 28：**“Flow”菜单下的“Open Dataflow Design”**

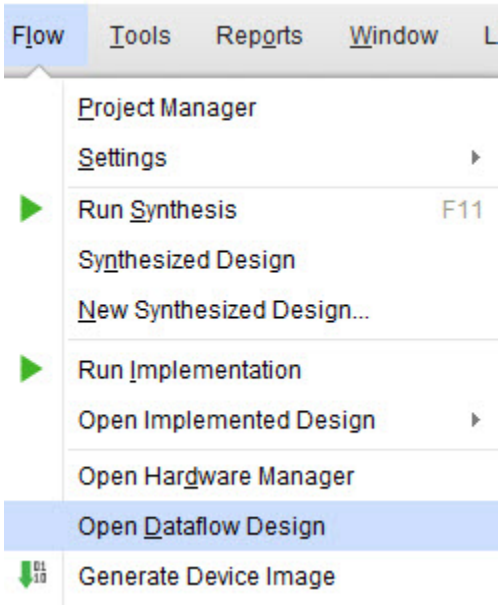
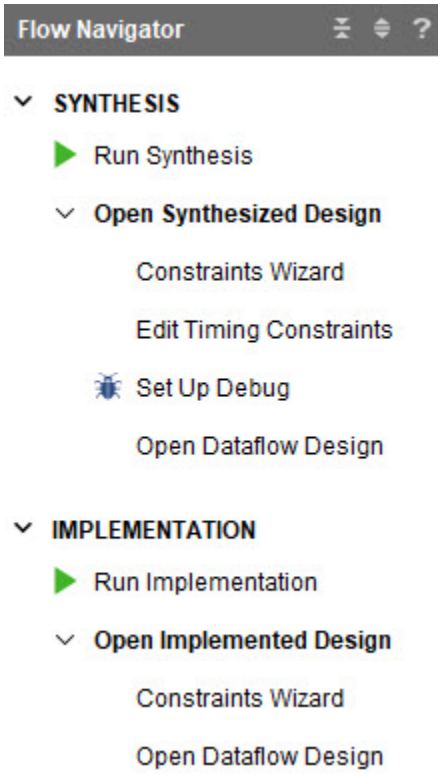


图 29：**Flow Navigator**



当设计已布局或已布线之后，会复制保留在数据流网表中的单元位置，并在器件视图中显示。

注释：网表创建后执行的位置更改都不会反映在网表中。要反映这些更改，需要重新生成数据流网表。

通过增大为 `-min_bus_width` 选项指定的值，可进一步简化所创建的网表。该选项的默认值是 16，将其设置为较大的值会移除较小的总线，从而创建更简化的网表。以下是有关如何使用开关的示例：

```
create_dataflow_design -min_bus_width 128
```

数据流网表生成

数据流网表是原始设计网表的一个子集。生成数据流网表时，会删去以下各项：

- 标量信号线
- 低于 `-min_bus_width` 阈值（默认值为 16）的总线信号线
- 寄存器
- 对数据路径不会产生重大影响的 LUT

数据流网表由以下各项组成：

- 等于或高于 -min_bus_width 阈值的总线信号线
- 影响布局的重要单元。其中包括：
  - BlockRAM

  - UltraRAM
  - LUTRAM
  - DSP slice
  - 硬核 IP
  - 千兆位收发器
  - NoC
- 层级单元
- 时钟信号线

浏览数据流网表

查看数据流设计的主要浏览方法是使用原理图功能。您可以使用与父设计中相同的方法来探索设计。这些方法包括但不限于：

- 选择一个对象，右键单击并选择一个原理图（或按 F4），用所选对象创建一个新的画布。
- 将“Netlist”（网表）窗口中的对象拖放到现有的原理图画布上。
- 双击管脚对象会将信号线展开至其负载（如果是输出管脚）或展开至源（如果是输入管脚）。
- 右键单击管脚对象，然后选择“expand cone to...”（将椎展开至...），可将信号线展开至其负载或源。
- 双击单元会将连接到该单元的所有信号线展开至其源或负载。

为了简化浏览，以下功能与父设计浏览不同：

- 从原理图视图中移除了内部原语。宏级别以上都可查看。这对加快 DSP 和 LUTRAM 单元的浏览速度尤其有帮助。
- 选中 DSP 链或 RAM 级联链等级联单元时，Vivado 会选中 DFV（数据流查看器）内部链中的所有原语。这有助于快速浏览到链的输出，并移至下一个单元。

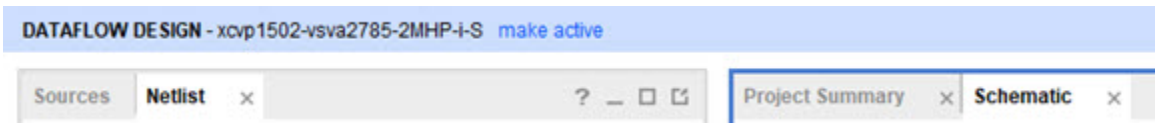
在数据流查看器中的窗口之间交叉探测

选中数据流设计时，处于活动状态的网表是数据流网表，因此：

- “Netlist”（网表）窗口会显示数据流网表
- 任何打开的原理图窗口都会显示数据流网表的元素
- “Hierarchy”（层级）窗口的大小因数据流网表中的原语数量而定
- “Edit” → “Find”（编辑 > 查找）等任何其他项也适用于数据流网表

注释：Vivado IDE 顶部的蓝色功能区表明所选的设计。选中数据流设计时，蓝色功能区包含“DATAFLOW DESIGN”，如下图所示。

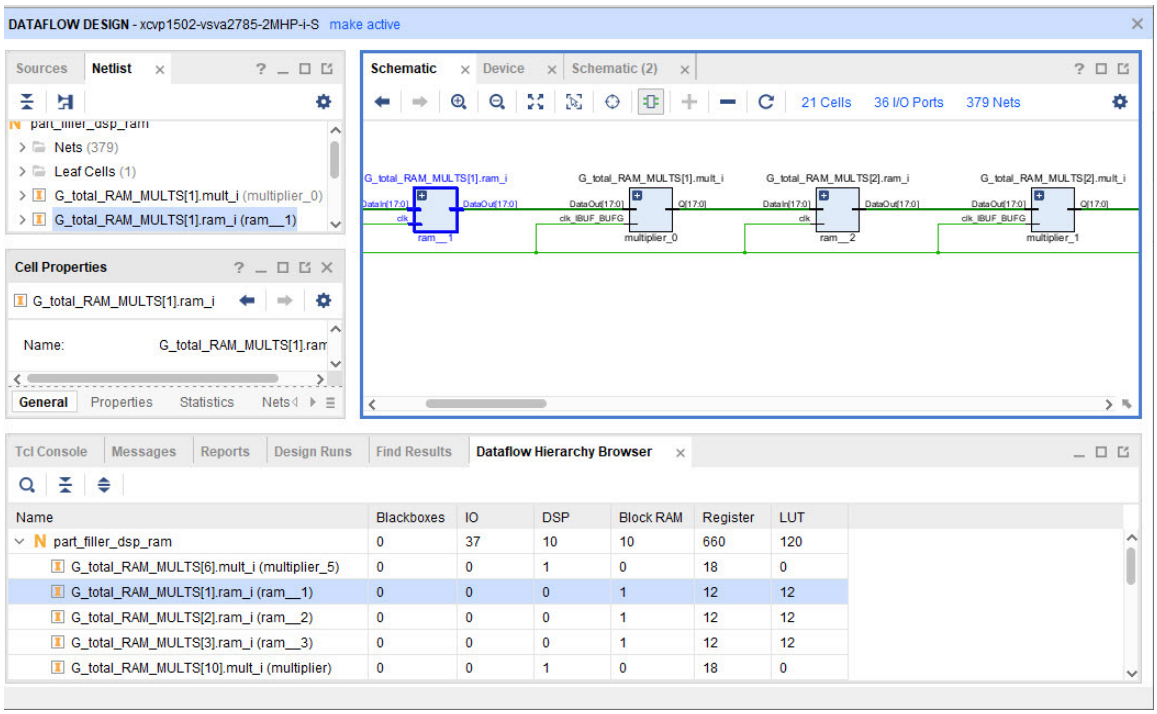
图 30：**包含 DATAFLOW DESIGN 的蓝色功能区**



在任一窗口中选中某个单元时，在所有窗口中都会选中该单元。通过这种交叉探测，您可以在适当的情况下选中某个对象，然后切换窗口以进行进一步调查。

“Hierarchy Browser”（层级浏览器）显示了来自原始网表的资源（如寄存器、LUT 和硬核 IP）的使用率。在数据流查看器中，由于已移除寄存器和 LUT，因此有时层级会看起来很小，但实际上资源要求可能会很高。如果在另一个窗口中选中层级单元，则会在“Dataflow Hierarchy Browser”（数据流层级浏览器）中显示资源使用情况，如下所示。这有助于快速查看层级单元内的实际使用率，并降低切换设计的要求：

图 31：**数据流设计交叉探测**



在父设计与子设计之间交叉探测

出于各种原因，有必要在源自数据流设计的设计到父设计之间进行交叉探测。例如，可能需要：

- 针对父网表进行交叉探测，查看从数据流网表中删去的详细信息。执行此操作可以观察如下项：
  - 被删去的单元
  - 关键布局单元之间的流水线阶段数
  - 查看数据流网表中不可用的单元的布局
- 运行数据流查看器中不可用的分析，例如：
  - 时序报告
  - 使用 `report_design_analysis` 执行的拥塞分析

单元交叉探测是最有效的交叉探测方法。具体方法是选择一个设计中的某个单元对象，然后切换设计以进行交叉探测。创建网表时，会重命名众多信号线，因此某一设计中的某一条信号线与数据流设计中的某一条信号线之间通常没有相关性。

执行从数据流设计到原始网表的交叉探测时，数据流查看器中的所有叶节点单元都包含在父网表中，因为数据流网表是父网表的子集。这样能成功执行交叉探测。但恰恰是出于同样原因，执行从父网表到作为其子集的数据流设计的交叉探测则不太可能成功。请避免尝试从叶节点单元（如寄存器或 LUT）进行交叉探测，而应改用保留在数据流网表中的单元（如块 RAM、UltraRAM、DSP 和硬核 IP）来进行交叉探测。

在设计之间切换的最简单方法是使用屏幕顶部的“Design Switch”（设计切换）图标。这样可以快速切换到父设计或先前使用的数据流设计。此图标仅在创建数据流设计后才可用。根据是从其他设计切换到数据流设计还是从数据流设计切换到其他设计，此图标会发生改变。

以下呈现了不同的“切换设计”图标：

-  表示从其他设计切换到 DFV 进行交叉探测。
-  表示从 DFV 切换到其他设计进行交叉探测。

数据流路径

`get_dataflow_paths` 命令会返回数据流网表中的路径列表，这些路径包含两个或更多个单元以及这些单元彼此之间的连接。该命令允许您将数据流设计分析锁定为目标单元以及与其连接的单元。

返回的对象可与 `show_objects` 命令一起使用，以在 Vivado IDE 中显示路径列表。您可选中此命令，并与其他窗口（如原理图、网表和器件）进行交叉探测。以下是将此命令与 `show_objects` 命令结合使用的一个示例：

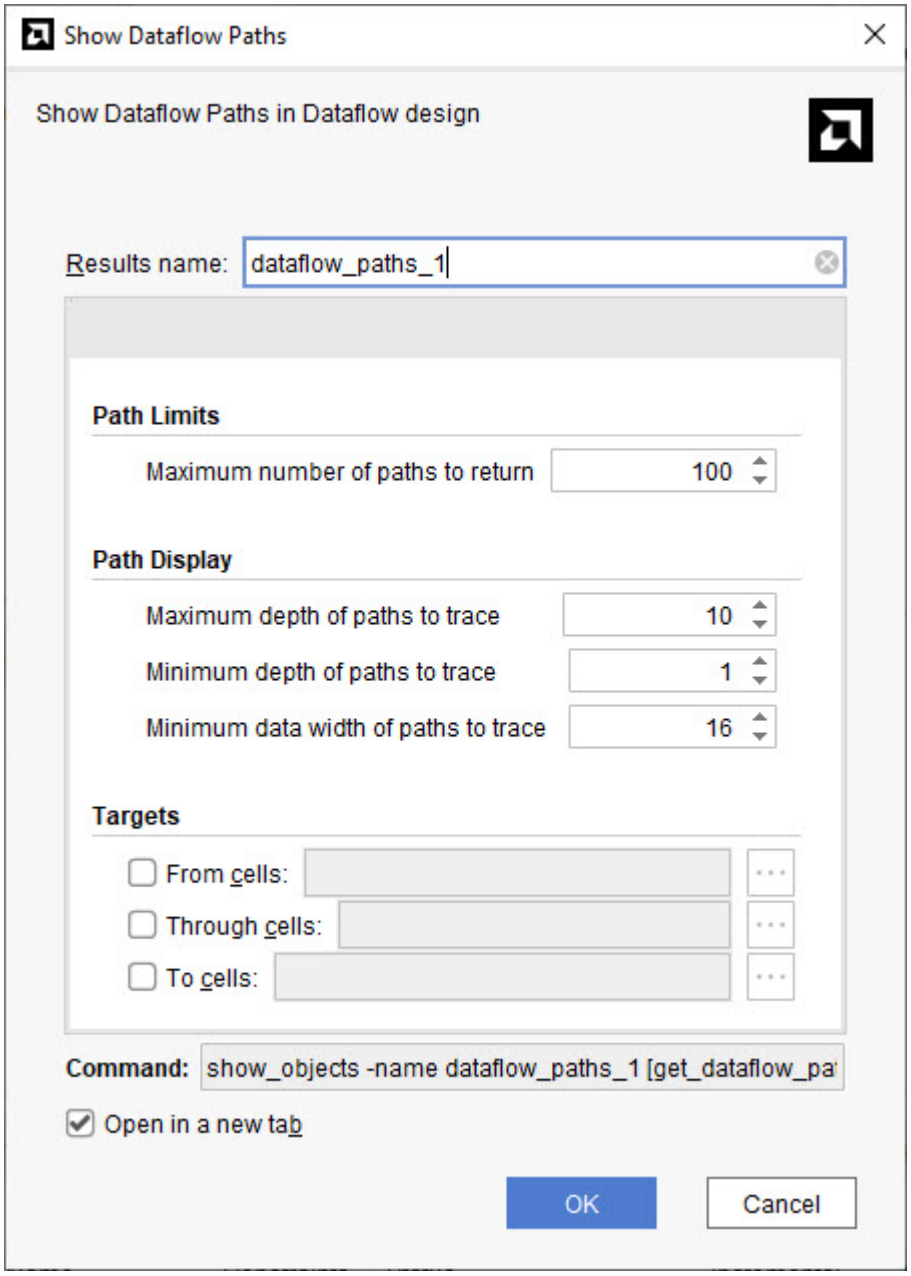
```
show_objects -name dataflow_paths_1 [get_dataflow_paths]
```

您也可以使用“Tools”→“Show Dataflow Paths”（工具 > 显示数据流路径）菜单在 Vivado IDE 中生成数据流路径。在此窗口中可执行下列操作：

- 控制该命令返回的路径数量
- 控制要追踪的单元的最小和最大数量
- 控制数据流路径内总线的最小数据宽度
- 将始于、止于或穿过特定单元的路径设定为目标

也可通过以下方式从原理图窗口中打开“Dataflow Paths”（数据流路径）对话框：右键单击叶节点单元，然后选择“Show Dataflow Paths”→“From/Through/To Cell”（显示数据流路径 > 始于、止于或穿过单元），如下图所示。这样即可在该对话框中预填充选定单元的名称。

图 32： **“Show Dataflow Paths”窗口**



`get_dataflow_paths` 命令的工作方式如下：扫描数据流网表以查找路径，这些路径目标为最大深度 max_depth（默认值为 10）的开关；然后该命令会返回若干条路径，数量不超过 max_paths 所指定的值（默认值为 100）。如果路径数超过 max_paths 值，那么返回的优先顺序如下：

- 首先返回深度等于 max_depth 的路径，然后按深度顺序返回深度小于该值的路径
- 按起点的字母顺序返回

通过 Tcl 搭配使用以下开关有助于确保将返回的路径锁定为用户指定的对象：

```
-from <cells>
-to <cells>
-through <cells>
```

`-from / -to / -through` 允许您指定单元列表，并将返回的路径列表锁定为指定的对象。`-from` 开关允许用户指定起点，`-to` 开关允许用户指定终点，而 `-through` 则允许用户指定中间点列表。

```
-max_depth <integer>
-min_depth <integer>
-max_paths <integer>
-min_width <integer>
```

默认情况下，`-max_depth` 开关用于指定可搜索的最长路径。首先返回最长路径。此项的默认值为 10。`-min_depth` 允许用户指定最小深度。仅当未达到 max_paths 时才允许使用此项。`-max_paths` 开关允许您修改返回的最大路径数。默认值为 100。`-min_width` 开关用于指定将显示的最小总线宽度。返回结果中将不包含低于该值的总线的任何部分。默认值：1。

数据流设计中禁用的命令

数据流查看器仅用于对经数据流优化的网表进行网表分析。因此，有大量命令并不受数据流设计的支持。这些命令包括但不限于：

- 所有时序命令
  - `report_timing`、`report_clock_interaction`、`create_clock`、`set_max_delay` 以及任何时序约束或时序报告
- 所有 ECO 命令
  - `create_cell`、`create_net`、`create_pin` 和 `connect_net` 等。
- 实现命令
  - `synth_design`、`opt_design`、`place_design`、`phys_opt_design`、`route_design` 和 `write_device_image`
- 报告命令
  - `report_high_fanout_nets`、`report_control_sets`、`report_clock_utilization`、`report_ram_utilization`、`report_design_analysis`、`report_qor_suggestions` 和 `report_qor_assessment`

第 3 章

查看报告和消息

AMD 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。

本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“Results”（结果）窗口区域中的“Messages”（消息）窗口和“Reports”（报告）窗口中。

运行以下任一命令时，该工具会启动新进程：

- Run Synthesis
- 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“Tools”（工具）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

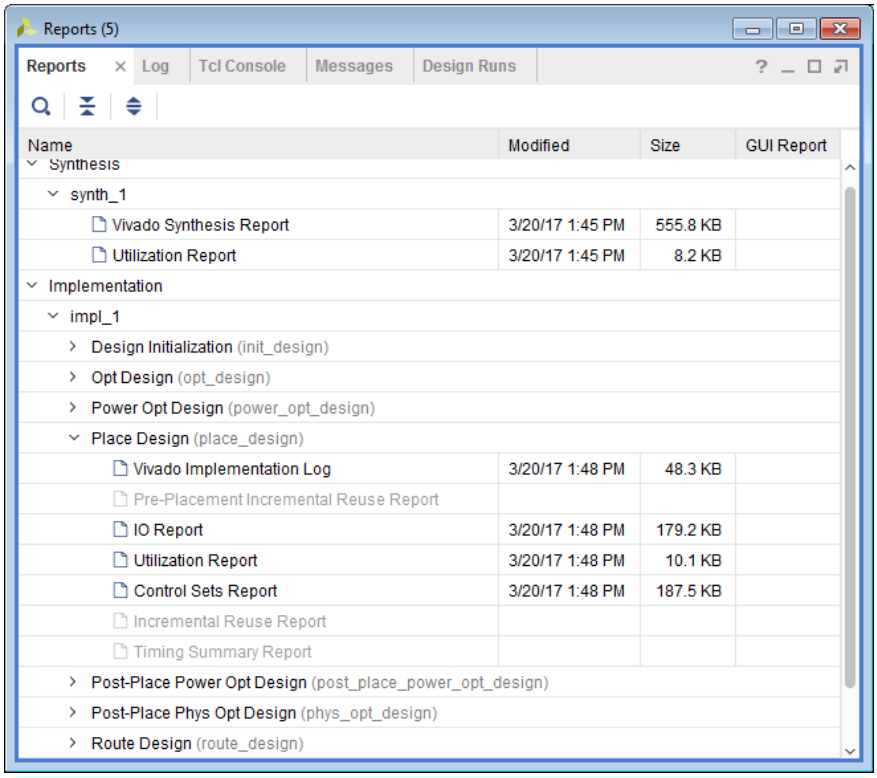
消息可提供有关设计特定元素或者有关工具流程中发生的错误的简洁状态声明。

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

使用“Reports”窗口

正在运行的综合与实现的报告显示在“Reports”（报告）窗口中。选择“Run Properties”（运行属性）窗口的“Reports”选项卡即可查看“Design Runs”（设计运行）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 33： **“Reports”窗口**



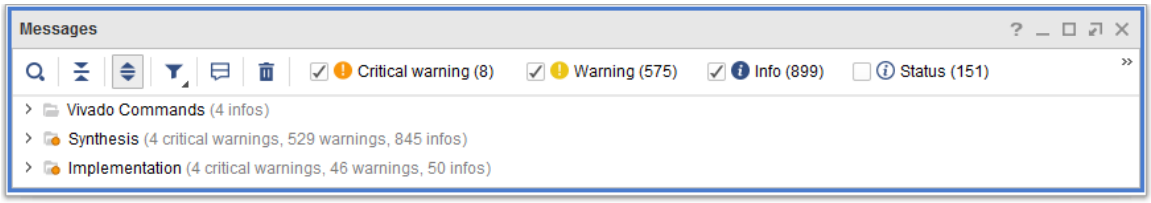
使用“Messages”窗口

消息分为两种类型：

- 存储在磁盘上的消息
- 存储在存储器中的消息

Vivado 集成设计环境 (IDE) 按创建消息时的操作在“Messages”（消息）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 34： **“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。单击链接即可查看来源。

提示：使用弹出菜单可将消息复制粘贴到其他窗口或文档。

每条消息都带有消息 ID 和消息严重性标签。

- “Message ID”（消息 ID）：消息 ID 用于识别不同消息，以便对其进行分组和排序。
- “Message Severity”（消息严重性）：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对设计进行细化、综合和实现。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。

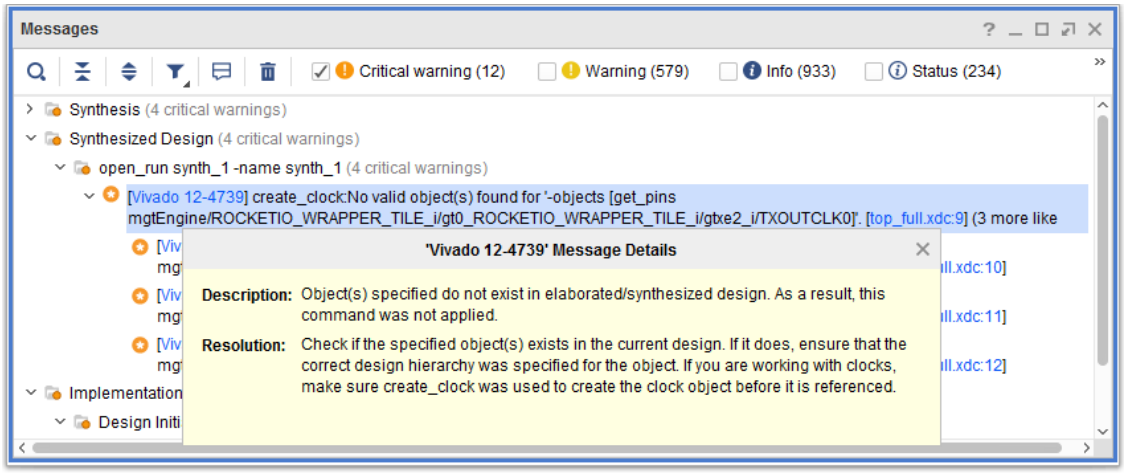
表 1：**消息严重性**

严重性	消息
Status（状态）	用于描述设计处理的一般状态。
Info（参考）	有关设计处理的工艺和反馈的一般状态。
Warning（警告）	由于未按期望方式应用约束或规范，可能导致设计结果不甚理想。
Critical Warning（严重警告）	某些用户输入或约束未应用，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（错误）	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存储器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和严重警告。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其他时钟也同样失败。

图 35：**审查“Error”和“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“Messages”（消息）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，请参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的 set_msg_config。

Vivado 生成的消息

本节旨在探讨 Vivado 生成的消息。

综合 log 日志

Vivado 综合 log 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
- 含多个驱动程序的信号线
- 未驱动的层级管脚
- 最优化信息
- 黑盒
- 最终原语计数
- 单元使用率（按层级）
- 运行时间和存储器使用率

重要提示！请复查此报告或“Messages”（消息）选项卡中显示的“Errors”（错误）、“Critical Warnings”（严重警告）和“Warnings”（警告）。“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 log 日志

Vivado 实现 log 日志包括如下内容：

- 有关位置、网表和所使用的约束的信息。

- 逻辑优化任务。默认情况下，该工具会运行逻辑优化例程来降低使用率并提高设计性能。
- 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存储器。

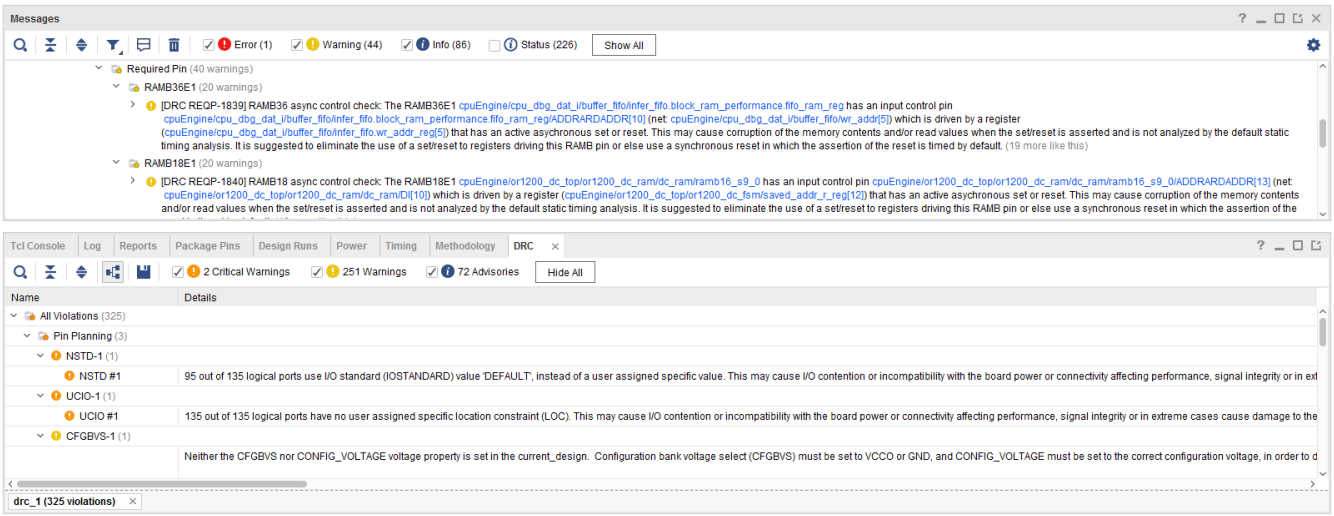
请复查此报告或“Messages”（消息）选项卡相应部分中显示的“Errors”（错误）、“Critical Warnings”（严重警告）和“Warnings”（警告）。“Placer”（布局器）会生成警告，这些警告在流程后续可能升级为“Errors”。如果使用单步运行，那么 log 日志将仅包含最后一步的结果。

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

使用 DRC 报告

设计规则检查 (DRC) 用于检查设计并报告常见问题。执行实现期间，工具也会运行 DRC。DRC 随布局布线不断完善。

图 36：**显示严重警告和错误**



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

设计早期阶段中的“Critical Warnings”在后续实现流程中会转变为“Errors”（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“Report DRC”（DRC 报告）步骤会在报告中将未约束的 I/O 列为“Critical Warning”。布线后设计 DRC 报告也会报告此类“Critical Warnings”。您必须审查报告，因为在 `write_bitstream` 中，“Critical Warning”DRC 会提升到错误级别严重性，并阻止生成比特流。请尽早审查 DRC 报告以判断设计中哪些领域需要修改。

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。如果将强制性 DRC 作为实现命令（如 `opt_design`、`place_design`、`phys_opt_design`、`route_design` 或 `write_bitstream`）的前提条件来运行，那么同样会滤除其中已豁免的 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_*` 命令返回的其他第一类对象，例如，管脚、单元、信号线、Pblock 和 site（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！与其他约束类似，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：Vivado 命令 `rename_net`、`rename_cell`、`rename_port` 和 `rename_pin` 不会对豁免内的设计元素进行重命名。如果豁免引用上述任一命令重命名的网表元素，则会变为无效，因为它仍引用该设计元素的原始名称。

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，请参阅《Vivado Design Suite 用户指南：使用 Tcl 脚本》(UG894) 中的“创建定制设计规则检查 (DRC)”。

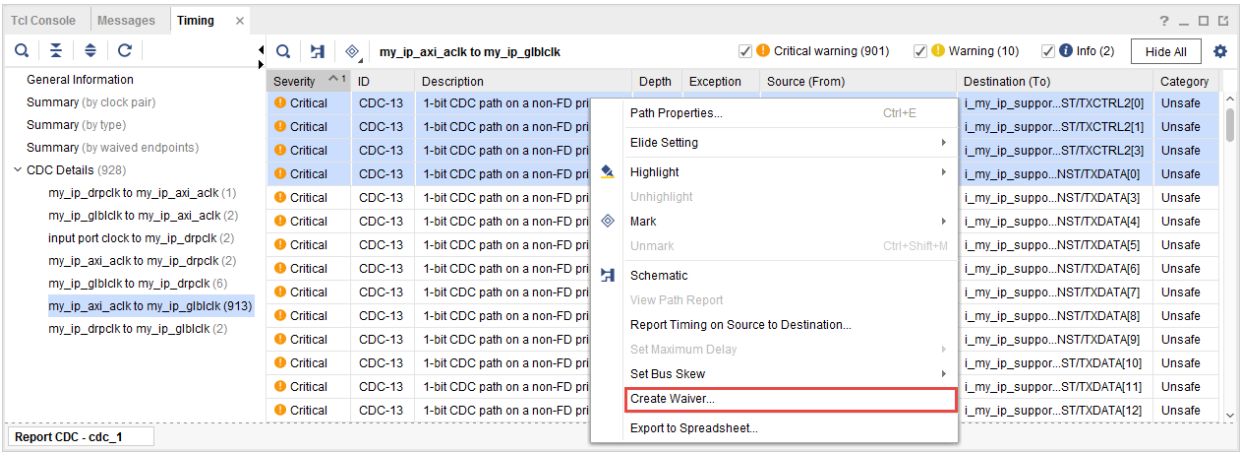
创建豁免

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

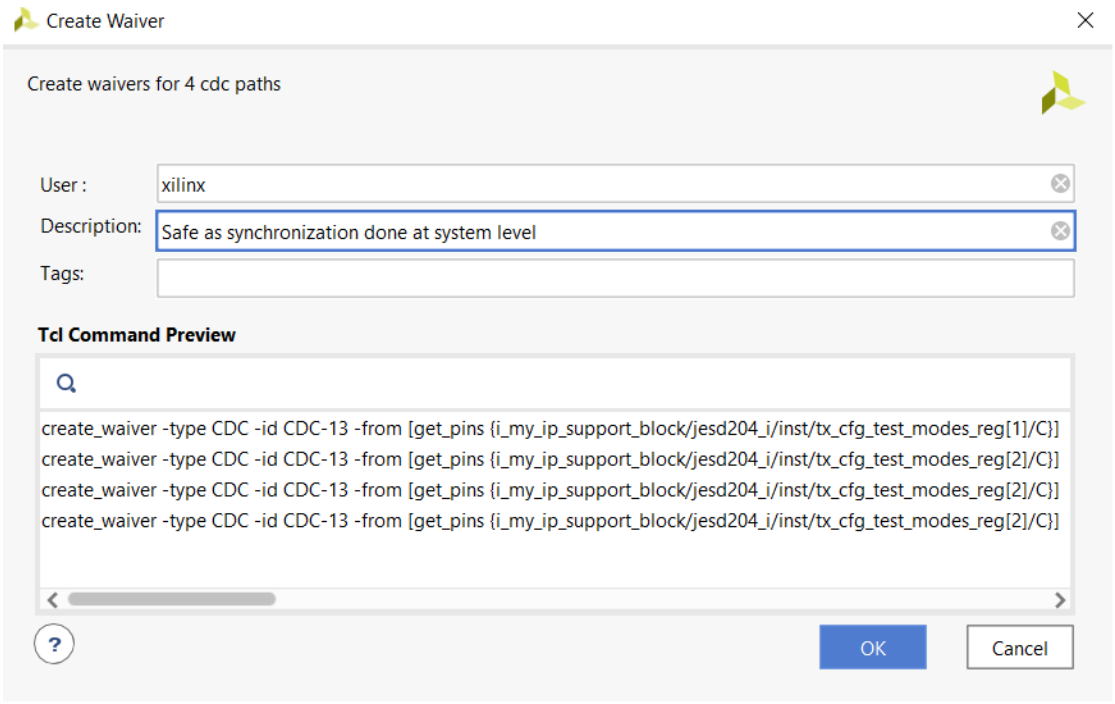
在“Report DRC”（DRC 报告）、“Report Methodology”（方法论报告）或“Report CDC”（CDC 报告）GUI 结果窗口中均可直接创建豁免。要从结果窗口中豁免违例，请选中一项或多项违例，右键单击上下文菜单并从中选择“Create Waiver”（创建豁免）。在下图中，已选中 4 个 CDC 违例并为其创建 4 项豁免。

图 37：**从 CDC 违例创建豁免**



单击“Create Waiver”会打开以下小组件。

图 38：**Create Waiver GUI**

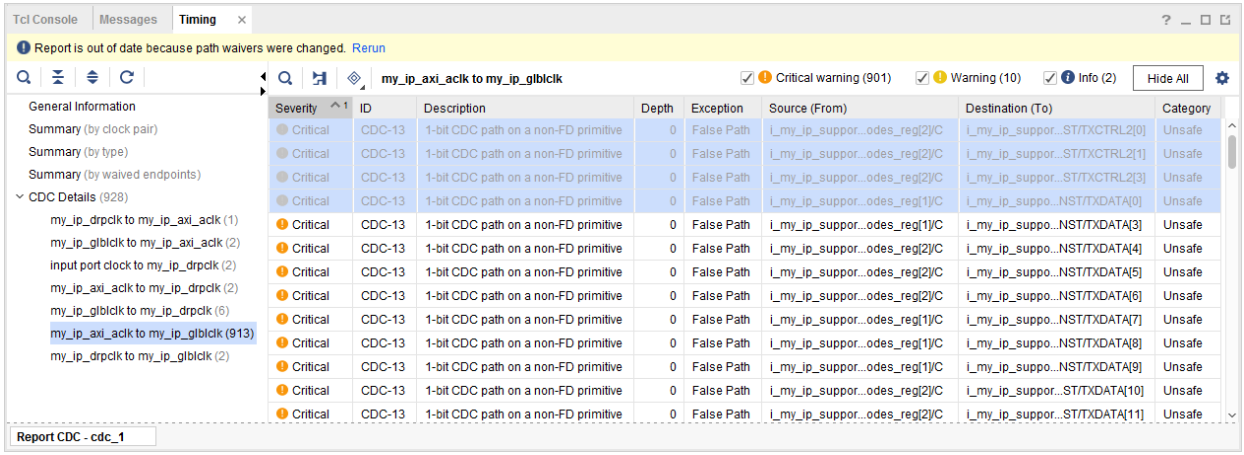


即使 Vivado 工具会填充用户名，但此“User”字段仍可编辑。“Description”（描述）为必填项，建议提供详细信息以供设计团队复查。“Tags”（标签）字段为可选字段，可用于通过关键字字符串或列表来提供额外描述。它主要用于记录，可用于搜索 XDC 文件或者使用 `get_waivers` 命令筛选豁免等。该对话框包含 GUI 发送给 Tcl 控制台的 Tcl 命令预览。

提交“Create Waiver”（创建豁免）窗口后，针对豁免的每项违例，GUI 将向 Tcl 控制台发送一条 `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。豁免是基于原始违例对象创建的，但创建的豁免从不引用该对象。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，将从新结果窗口中滤除已豁免的违例。

图 39：**创建豁免后已禁用的行**



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>]
```

“description”（描述）为必填字段。不指定“user”（用户）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_methodology_violations` 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。如需获取有关使用模式和通配符的更多信息，请参阅 创建 DRC 豁免和 Methodology 豁免。

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（方法论）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）的其他豁免。

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”（描述）：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。如果创建的豁免所设置的类型错误，则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 作用域限定机制，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 时复原。创建限定作用域的豁免时，建议使用命令行选项 `-scope` 以确保限定通配符作用域。

如果豁免所含实参与已存在的其他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。为减少存储器占用和运行时间，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）不含其他实参，因为消息为通用消息而非专用消息。其他 DRC 和 Methodology 违例可能包含多个字符串和不同类型的对象。

注释：如果违例不引用任何字符串或对象（例如，TIMING-9 和 TIMING-10），不建议将其豁免。

违例内的字符串是通过豁免内的 `-string` 来指定的。器件或设计对象（管脚、单元、信号线、Pblock 和 site）是以 `-objects` 选项来指定的。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：

- 使用 `get_*` 命令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。
- 指定对象列表以代替单一对象。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 2：**通配符关键字**

对象	通配符
单元	*CELL
信号线	*NET
管脚	*PIN
端口	*PORT
Site	*SITE
拼块	*TILE
BEL	*BEL
封装 bank	*PKGBANK
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。创建限定作用域的豁免时，`-scope` 可确保管脚和单元的通配符与位于比该作用域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”（CDC 报告）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 3：**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 读写潜在冲突	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-6	多位，使用 ASYNC_REG 属性同步	Warning	Unsafe
CDC-3	1 位，使用 ASYNC_REG 属性同步	Info	Safe

表 3：**CDC 规则优先级** (续)

规则 ID	CDC 拓扑结构	严重性	类别
CDC-8	异步复位，使用缺失的 ASYNC_REG 属性同步	Warning	Safe
CDC-2、5	1 位和多位 CDC，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他严重性为“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”（CDC 报告）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

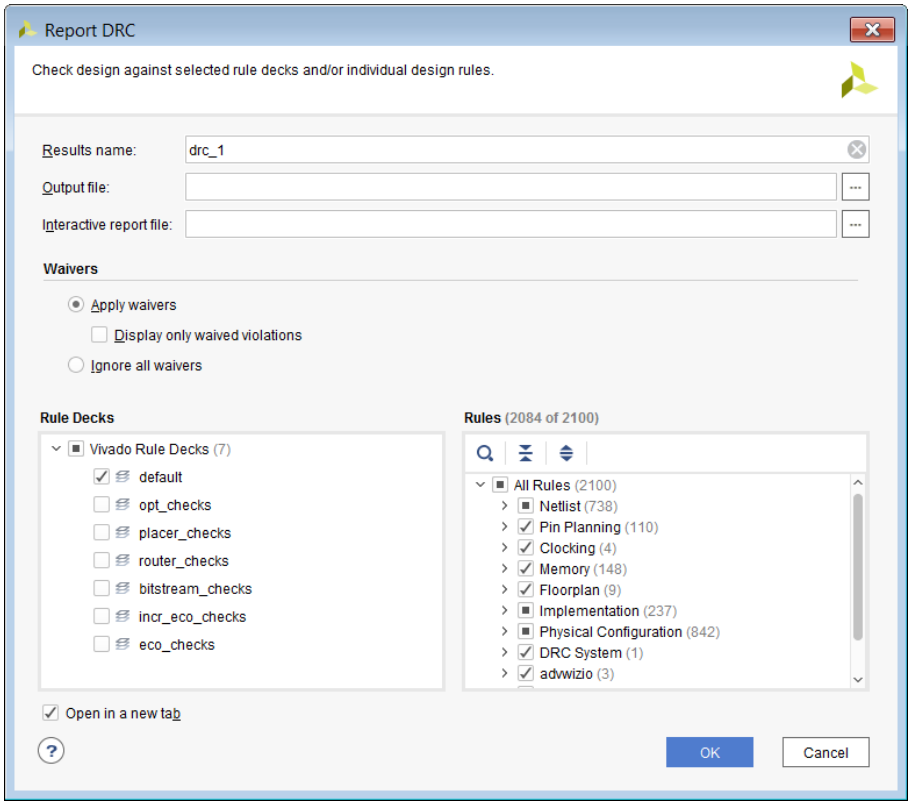
建议验证豁免的违例是否符合预期。必须在定义豁免后且最终比特流之前执行此操作。

“Report CDC”（CDC 报告）、“Report DRC”（DRC 报告）和“Report Methodology”（方法论报告）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”（豁免）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 40：**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 41：**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、DRC 或 Methodology 信息已过期：

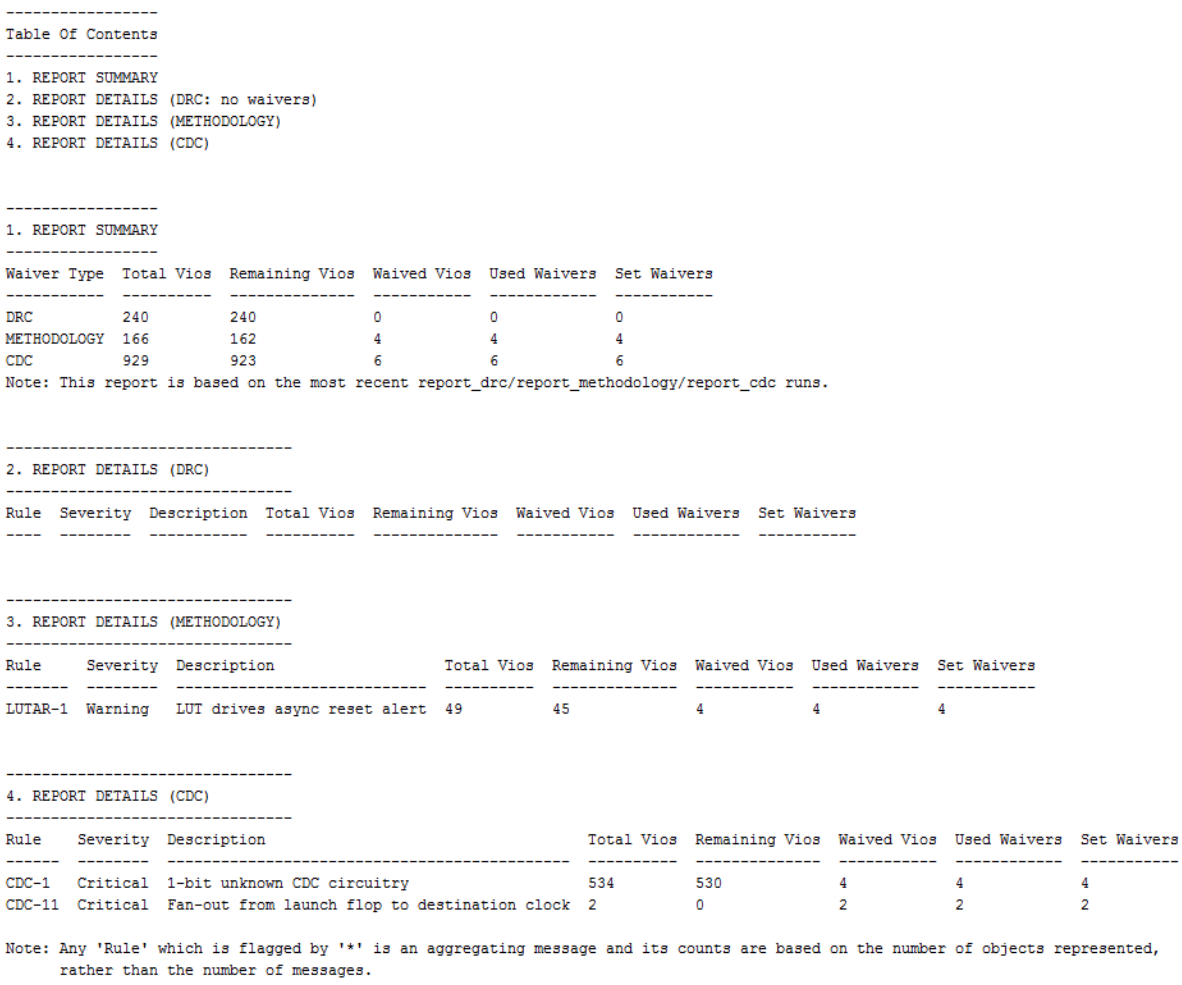
```
WARNING: [Vivado_Tcl 4-972] Waiver counts for 'CDC' will be invalid because
report_cdc has not been run since waivers were changed; please run the
report_cdc
command.
WARNING: [Vivado_Tcl 4-972] Waiver counts for 'DRC' will be invalid because
report_drc has not been run since waivers were changed; 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：

- “Total Vios”（违例总数）：应用豁免前的违例总数。将报告的违例数量（不含违例）。按端点数量计算多比特规则数量。
- “Remaining Vios”（剩余违例数）：应用豁免后的违例数。没有任何违例获得豁免时，此数值与“Total Vios”相同。按端点数量计算多比特规则数量。
- “Waived Vios”（已豁免的违例数）：已豁免的违例数。没有任何违例获得豁免时，此数值为 0。按端点数量计算多比特规则数量。
- “Used Waivers”（已用豁免数）：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。
- “Set Waivers”（已落实的豁免数）：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”（汇总）表会报告设计中的所有违例。

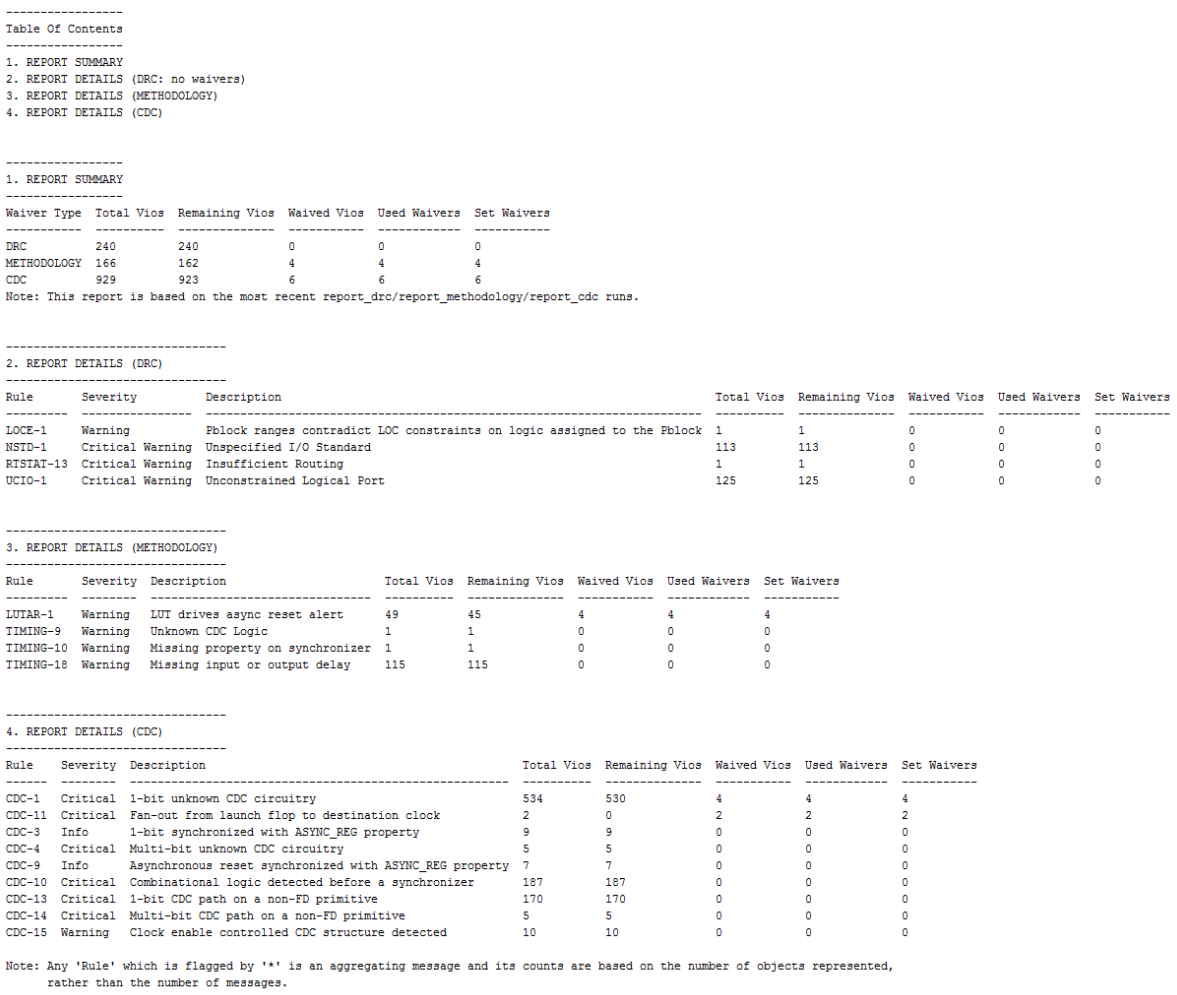
图 42：**report_waivers 默认报告**



```
-----------------
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
----  --------  -----------  ----------  --------------  -----------  ------------  -----------


---------------------------------
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


---------------------------------
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

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 43：**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
RTSTAT-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。如果不应出现不匹配的豁免，则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，并从检查点复原。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 AMD 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 AMD 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。AMD 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 AMD IP 豁免的差异。

表 4：**导出豁免**

Vivado 命令	导出用户豁免	导出 AMD IP 豁免
write_xdc	支持	支持
write_waivers	支持	不支持

其他豁免命令

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 命令不返回 AMD IP 豁免。

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 AMD IP 豁免。

可配置报告策略

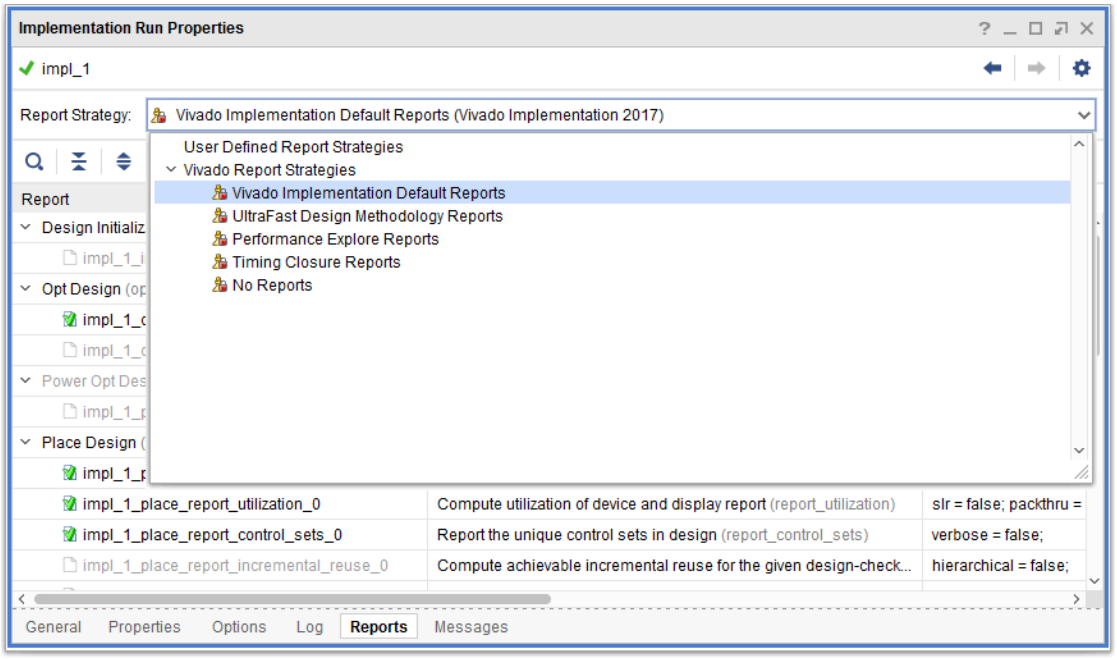
“Configurable Report Strategies”（可配置报告策略）支持在 Vivado 工程模式下运行综合与实现的每个步骤之后选择要运行的报告命令。根据设计阶段、设计复杂性和用户首选项，需自动生成一组不同的报告以供频繁查看。默认情况下，有多项预定义的综合和实现报告策略可供使用。此外，Vivado IDE 支持创建新的报告策略，并将这些策略与用户首选项和任何其他 Vivado IDE 设置保存在一起。

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)。要设置其他报告策略，请执行以下操作：

1. 在“Design Runs”窗口中，选择运行。
2. 选择“Run Properties”（运行属性）窗口中的“Reports”（报告）选项卡。
3. 从“Report Strategy”（报告策略）下拉列表中选择策略。

图 44：**选择对应“Implementation Run”（实现运行）的“Report Strategy”**



“Flow”（流程）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 5：**流程和报告策略**

流程	报告策略	注释
综合	Vivado 综合默认报告	仅在综合结束后运行使用率报告
	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《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中建议的所有报告
	性能浏览报告	与默认报告相同，完成 `phys_opt_design` 后运行附加时序报告
	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

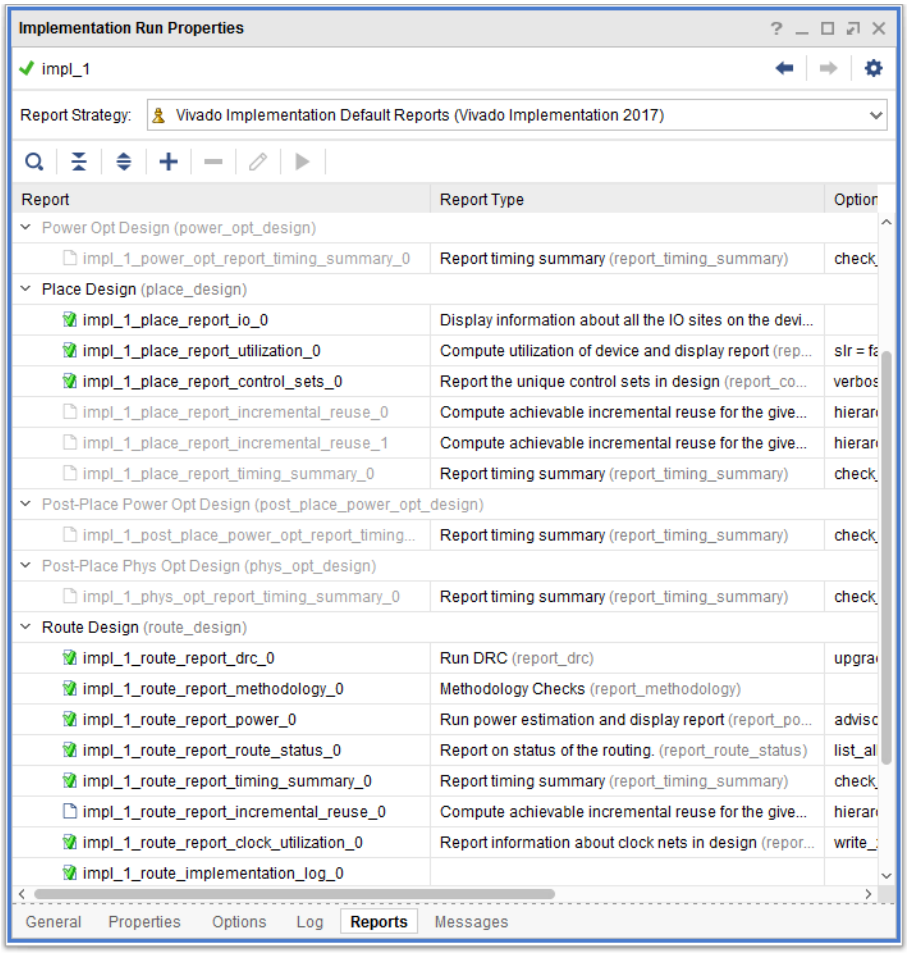
启动运行前，由于以下某一原因，部分报告可能为灰显：

- 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“Incremental Compile”（增量编译）运行中启用

图 45：**查看生成的运行报告**



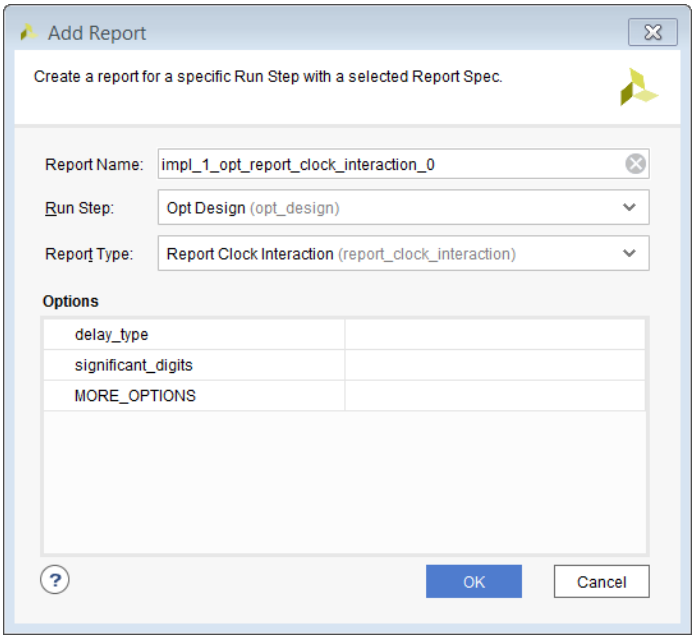
提示：设置旧工程属性 `ENABLE_OPTIONAL_RUNS_STA` 时，仅生成少量时序汇总报告。AMD 保留在未来版本中删除该属性的权力。示例：`set_property ENABLE_OPTIONAL_RUNS_STA 1 [current_project]`

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“Run Properties”（运行属性）窗口中的任意现有报告来决定要添加、删除还是编辑报告：

- “Add Report (+)”（添加报告）：选择“Run Step”（运行步骤）和“Report Type”（报告类型），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 46：**向“Run Report Strategy”添加报告**



- “Delete Report (-)”（删除报告）：此按钮用于从“Run Report Strategy”（运行报告策略）删除选定报告。此操作不可撤销。
- “Edit Report ( )”（编辑报告）：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。

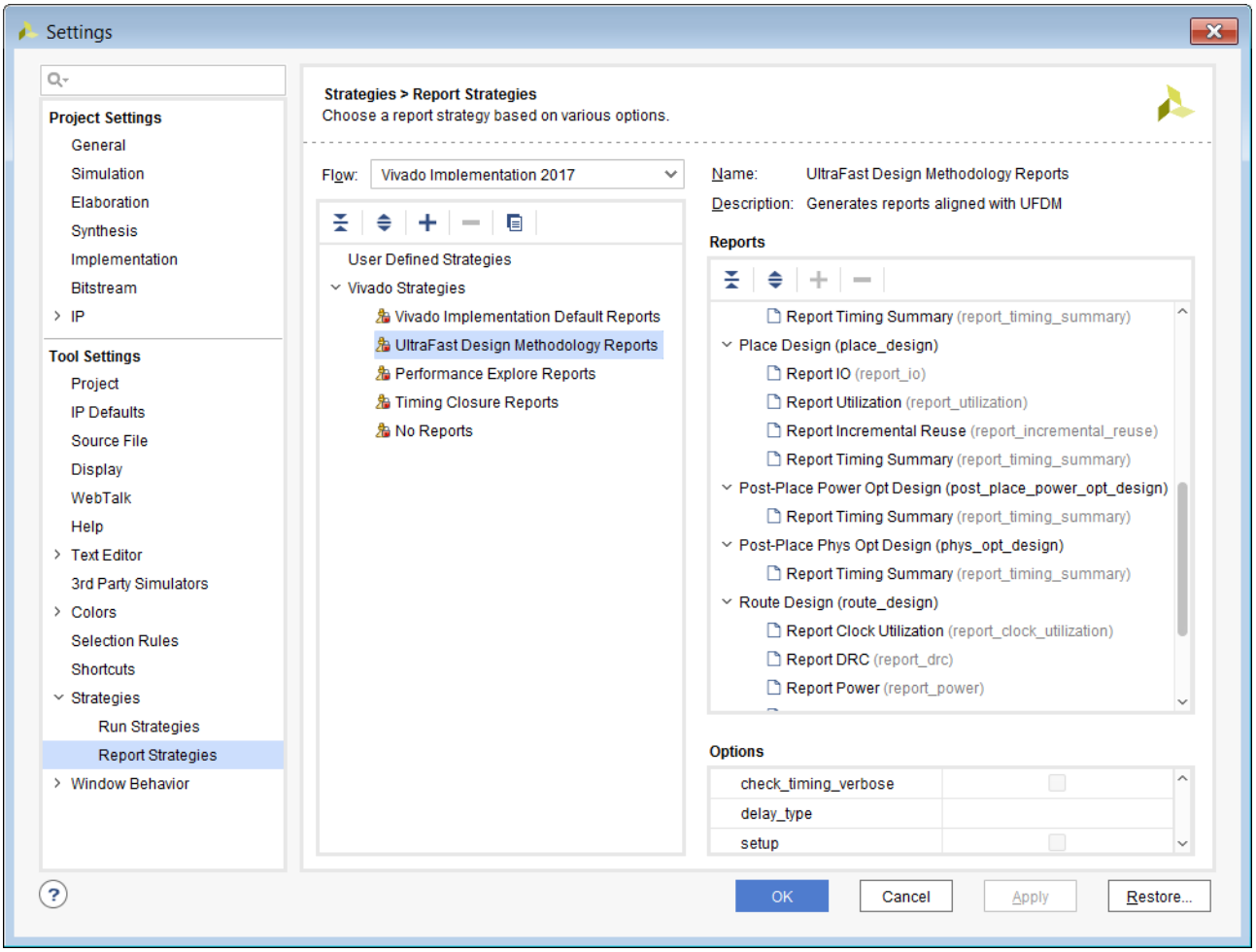
重要提示！运行完成后生成新报告时，Vivado 会在后台打开对应流程步骤的检查点，以生成报告文件。此操作会阻止大部分 Vivado IDE 功能（包括 Tcl 控制台）直至成功生成报告为止。生成报告可能需要几分钟到 1 小时时间，具体取决于您的设计大小以及报告的复杂性。

对特定运行的“Report Strategy”（策略报告）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”（工程设置）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。

创建新的策略报告

新“Report Strategies”（策略报告）必须在 Vivado IDE 的“Project Settings”（工程设置）窗口中的“Tool Settings”（工具设置）的“Strategies”（策略）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”（运行策略）与“Report Strategy”的默认关联均无法修改。

图 47： **“Settings”窗口的“Report Strategies”配置**



要创建新的策略报告：

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“Flow”（流程）、“Synthesis”（综合）或“Implementation”（实现）。
   c. 提供描述（可选）。
   d. 单击“OK”（确定）。

   或：
2. 选择现有策略、单击“copy”（复制）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“-”按钮可移除报告。
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 5：流程和报告策略。

第 4 章

时序分析

AMD 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”（时序汇总报告）属于时序验收报告。“Report Timing Summary”可提供所有时序检查的完整概览，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 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 log 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。其他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作用域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 Vivado IDE 时序引擎所使用的部分关键概念。

术语

- “launch edge”（发送沿）表示发送数据的源时钟的处于活动状态的时钟沿。
- “capture edge”（捕获沿）表示捕获数据的目标时钟的处于活动状态的时钟沿。
- “source clock”（源时钟）也称为“launch clock”（发送时钟）。
- “destination clock”（目标时钟）也称为“capture clock”（捕获时钟）。
- “setup requirement”（建立时间要求）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。
- “setup relationship”（建立时间关系）表示经时序分析工具验证的建立时间检查。
- “hold requirement”（保持时间要求）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。
- “hold relationship”（保持时间关系）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。

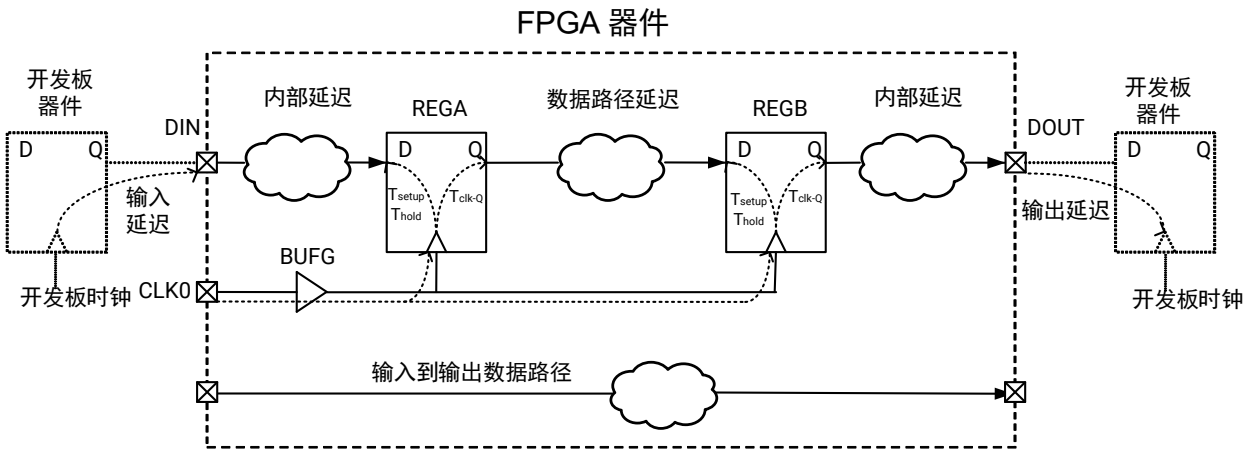
从输入端口到输出端口的路径

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 48：**时序路径示例**



“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。

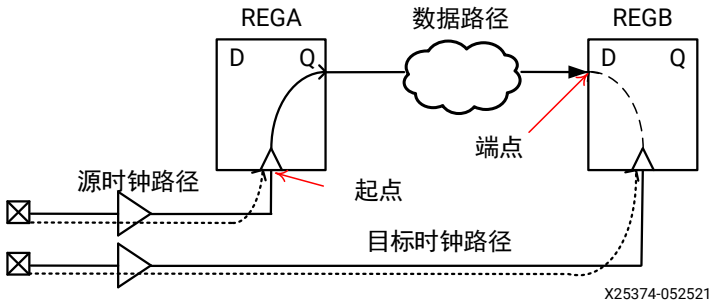
数据路径

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 49：**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 0 ns；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。

时序分析关键概念

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中并按时序角及其变化量加以分组，将所有这些要素一并纳入考量范围。针对所有建议的时序角分析时序即可，针对每个角，按最消极的条件执行所有检查。例如，以 AMD FPGA 为目标的设计必须通过以下 4 项分析：

- 慢速角 (Slow Corner) 中的最大延迟分析
- 慢速角 (Slow Corner) 中的最小延迟分析
- 快速角 (Fast Corner) 中的最大延迟分析
- 快速角 (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 (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 (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 (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 (corner) 的最差情况延迟（最慢延迟）。

映射到多个角时，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时序角 (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慢速 (Slow) 角延迟，保持或移除违例发生时存在快速 (Fast) 角延迟。但由于偶有例外（尤其是对于 I/O 时序），AMD 建议您在 2 个角上都执行 2 项分析。

建立/恢复关系

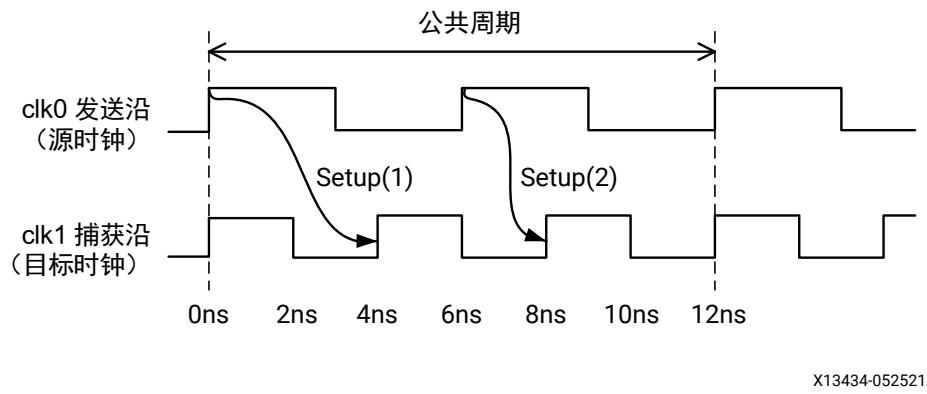
仅对 2 个时钟之间最消极的建立关系执行建立时间检查。默认情况下，此项检查对应于发送沿与捕获沿之间最小正增量。以 2 个触发器之间的路径为例，这 2 个触发器易受其各自时钟的上升沿影响。此路径的发送沿和捕获沿均仅为时钟上升沿。

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 50：**建立关系**



从 `clk0` 到 `clk1` 的最小正增量为 2 ns，对应于 Setup(2)。“Common Period”（公共周期）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，也就是在将插入延迟从时钟根应用于触发器时钟管脚之前确立的。

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：

Data Required Time (setup)（数据必需时间（建立））	**=**	capture edge time（捕获沿时间） + destination clock path delay（目标时钟路径延迟） - clock uncertainty（时钟不确定性） - setup time（建立时间）
Data Arrival Time (setup)（数据到达时间（建立））	=	launch edge time（发送沿时间） + source clock path delay（源时钟路径延迟） + datapath delay（数据路径延迟）
Slack (setup)（裕量（建立））	=	Data Required Time（数据必需时间）- Data Arrival Time（数据到达时间）

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）与建立关系直接相连。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：

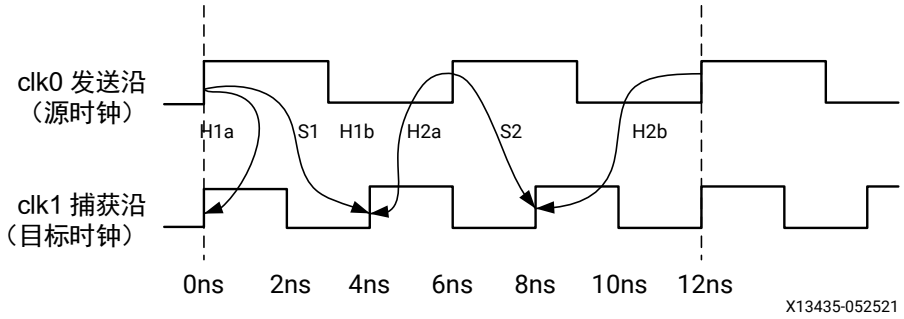
- 由建立发送沿所发送的数据不会被位于建立捕获沿之前的活动沿（下图中分别对应于建立沿 S1 和 S2 的 H1a 和 H2a）所捕获。
- 由位于建立发送沿之后的下一个活动源时钟沿发送的数据不会被建立捕获沿（下图中分别对应于建立沿 S1 和 S2 的 H2a 和 H2b）所捕获。

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 51：**保持关系（按建立关系）**



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：

> Data Required Time (hold) = capture edge time + destination clock path delay + clock uncertainty + hold time，即，数据必需时间（保持）= 捕获沿时间 + 目标时钟路径延迟 + 时钟不确定性 + 保持时间
>
> Data Arrival Time (hold) = launch edge time + source clock path delay + datapath delay，即，数据到达时间（保持）= 发送沿时间 + 源时钟路径延迟 + 数据路径延迟
>
> Slack (hold) = Data Arrival Time - Data Required Time，即，裕量（保持）= 数据到达时间 - 数据必需时间

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

时钟相移对应于延迟时钟波形，此波形与因时钟路径内的特殊硬件所导致的参考时钟相关。在 AMD 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 6：**MMCM/PLL PHASESHIFT_MODE 属性**

PHASESHIFT_MODE 属性	相移建模	注释
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 AMD 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 7：**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)
AMD UltraScale™	时钟波形修改 (WAVEFORM)
AMD UltraScale+™	MMCM/PLL 插入延迟 (LATENCY)
AMD Versal™ 自适应 SoC	MMCM/PLL 插入延迟 (LATENCY)

重要提示！ MMCM/PLL PHASESHIFT_MODE 属性不影响器件配置。它仅影响静态时序分析引擎的裕量计算。由于在 WAVEFORM/LATENCY 模式之间切换时可能需要调整时序约束，因此设计时序可能也会受到影响。

重要提示！ 在任何 `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，并会触发“Warning Timing 38-437”警告。在此类情况下，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时，使用 PHASESHIFT_MODE=LATENCY 尤显便利。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周期路径约束。

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，则会应用默认行为，并且 MMCM/PLL 时钟相移作为延迟时延而不是时钟沿相移来进行建模。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 Wizard 和 High Speed SelectIO Wizard 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/mmcm_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/mmcm_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/mmcm_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/mmcm_adv_inst/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/mmcm_adv_inst/
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“Clock Report”（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：报告中仅含对应于自动衍生时钟相移的延迟。在自动衍生时钟波形定义中不包含来自 MMCM/PLL 块的相移量。

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 并未给 MMCM 管脚 CLKOUT0 定义相移，但时钟 clk_out2_clk_wiz_0 已为 MMCM 管脚 CLKOUT2 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管脚的插入延迟。

在以下公式中：

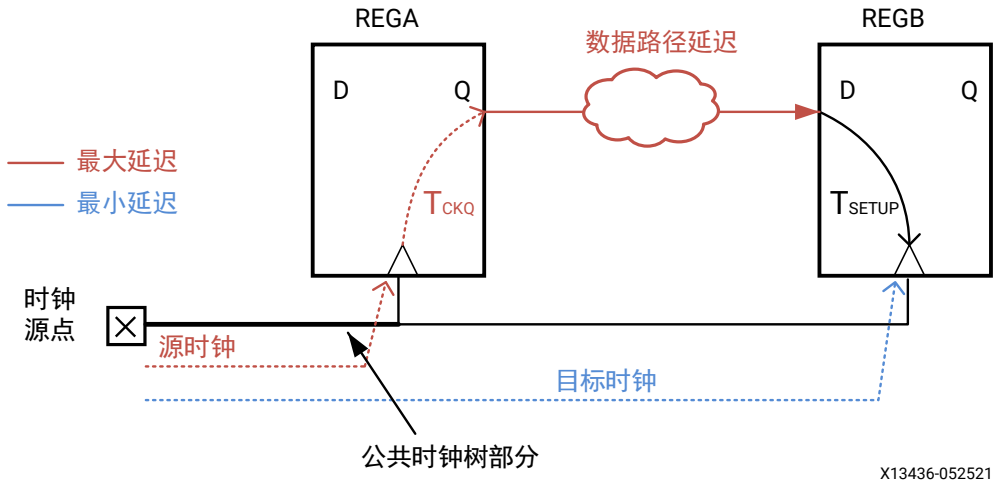
- `Tcj` 表示从公共节点到端点时钟管脚的延迟。
- `Tci` 表示从公共节点到起点时钟管脚的延迟：

```
Tskewi,j = Tcj- Tci
```

时钟消极因素移除

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 52：**公共时钟树部分**



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。以所含源时钟和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冲器驱动。

布线前，公共点为时钟信号线驱动，即时钟缓冲器输出管脚。CPR 仅补偿从时钟根到时钟缓冲器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

AMD 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冲器的时钟树，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 符号会发生改变，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。对于其他生成时钟（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 AMD 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，AMD 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，在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 所提供的所有其他设计规则检查一样。

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）。您必须查看目标器件的 AMD 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。

“Timing Path Summary”（时序路径汇总）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 时序分析关键概念。

时序路径汇总头文件示例

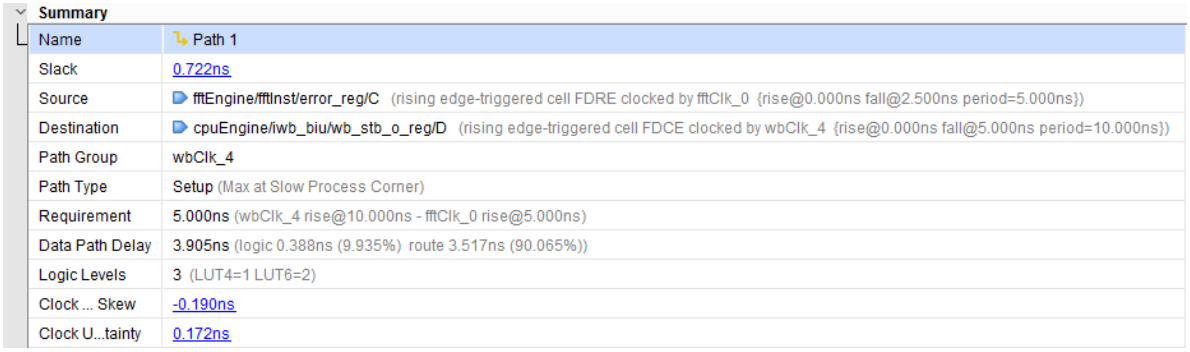
下图显示了文本报告中时序路径汇总头文件的示例。

图 53：**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  ((TSJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter     (TSJ):    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 54：**Vivado IDE 中时序路径汇总头文件**



时序路径汇总头文件信息

时序路径汇总头文件包含以下信息：

- “Slack”（裕量）

  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他小节中进行计算和报告。

- “Source”（源）

  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）。

- “Destination”（目标）

  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）。

- “Path Group”（路径组）

  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。

- “Path Type”（路径类型）

  此路径上执行的分析类型。

  - “Max”（最大值）：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”（最小值）：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时序角 (corner)：“Slow”（慢速角）或“Fast”（快速角）。

- “Requirement”（要求）

  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。

- “Data Path Delay”（数据路径延迟）

  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。

- “Logic Levels”（逻辑级数）

  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。

- “Clock Path Skew”（时钟路径偏差）

  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）。

- “Destination Clock Delay (DCD)”（目标时钟延迟 (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。

- “Source Clock Delay (SCD)”（源时钟延迟(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。

- “Clock Pessimism Removal (CPR)”（时钟消极因素移除 (CPR)）

  表示由于以下原因而导致的额外时钟偏差量绝对值：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）。

  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。

- “Clock Uncertainty”（时钟不确定性）

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。

- “Total System Jitter (TSJ)”（总系统抖动 (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。

- “Total Input Jitter (TIJ)”（总输入抖动 (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，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的“时钟时延、抖动和不确定性”章节。

- “Discrete Jitter (DJ)”（离散抖动 (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。

- “Phase Error (PE)”（相位误差 (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

- “User Uncertainty (UU)”（用户不确定性 (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，请参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的 set_clock_uncertainty 部分。

根据时序约束、报告的路径和目标器件，在“Timing Path Summary”中可能会显示其他行：

- “Inter-SLR Compensation”（SLR 间补偿）

  （仅限 AMD SSI 器件）安全报告跨 SLR 边界的路径所需的额外裕度。

- “Input Delay”（输入延迟）

  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。

- “Output Delay”（输出延迟）

  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。

- “Timing Exception”（时序例外）

  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 例外报告。

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：

- “Source Clock Path”（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。对于从输入端口开始的路径，不含这部分内容。

- “Data Path”（数据路径）

  数据从起点到端点遍历的电路。

- “Destination Clock Path”（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。

“Source Clock Path”部分和“Data Path”部分搭配在一起使用。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“data arrival time”（数据到达时间）。

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“data required time”（数据必需时间）。

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“Source Clock”（源时钟）、“Data”（数据）和“Destination Clock Paths”（目标时钟路径）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。

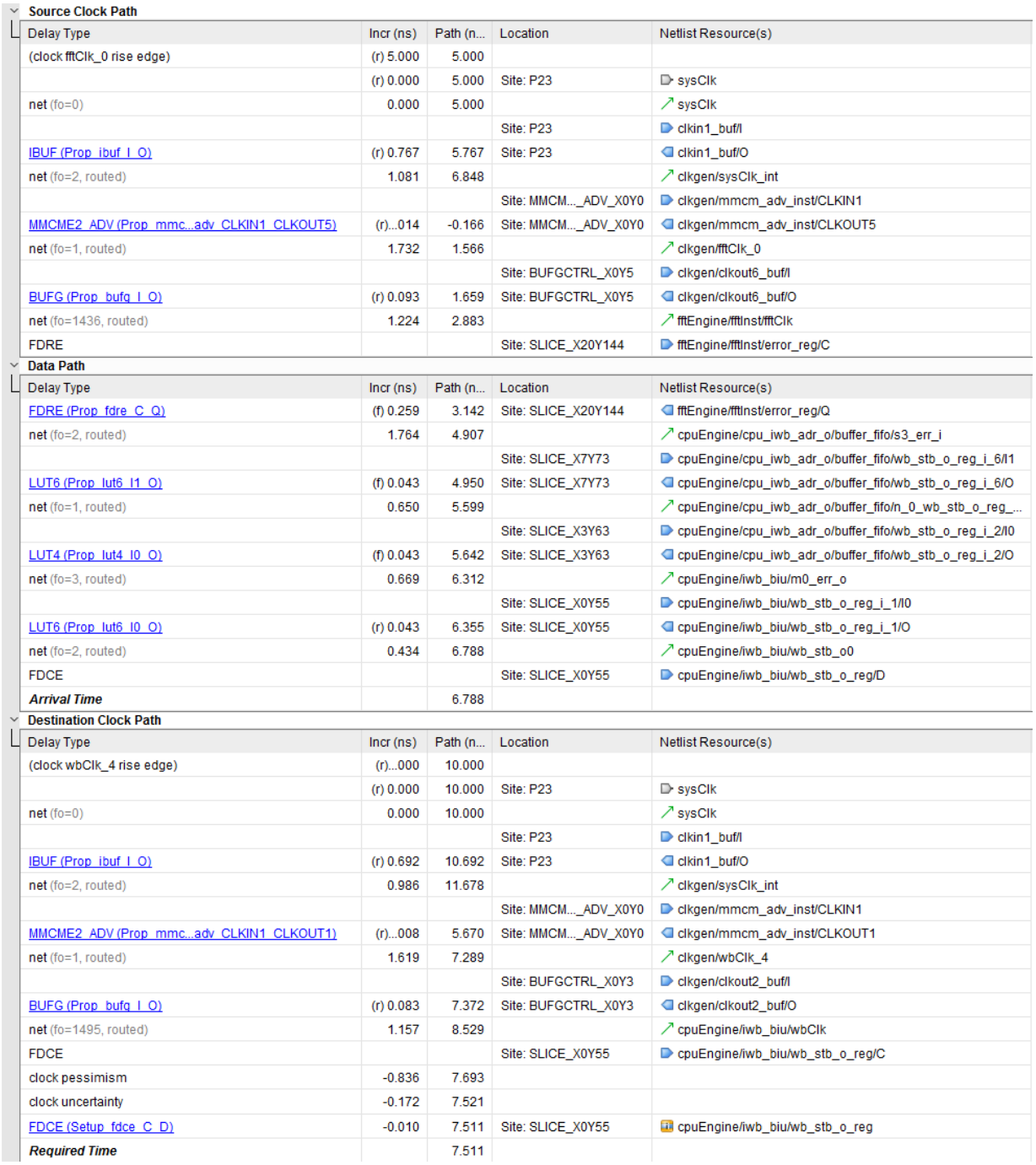
图 55：**文本报告中的时序路径详情**

Location	Delay type	Incr(ns)	Path(ns)		Netlist Resource(s)
	(clock fftClk_0 rise edge)				
		5.000	5.000	r	
P23		0.000	5.000	r	sysClk (IN)
	net (fo=0)	0.000	5.000		sysClk
P23	IBUF (Prop_ibuf_I_O)	0.767	5.767	r	clkin1_buf/O
	net (fo=2, routed)	1.081	6.848		clkgen/sysClk_int
MMCME2_ADV_X0Y0	MMCME2_ADV (Prop_mmcme2_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	BUFG (Prop_bufg_I_O)	0.093	1.659	r	clkgen/clkout6_buf/O
	net (fo=1436, routed)	1.224	2.883		fftEngine/fftInst/fftClk
SLICE_X20Y144	FDRE			r	fftEngine/fftInst/error_reg/C
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_O)	0.043	4.950	f	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_i_6
SLICE_X3Y63	LUT4 (Prop_lut4_I0_O)	0.043	5.642	f	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
SLICE_X0Y55	LUT6 (Prop_lut6_I0_O)	0.043	6.355	r	cpuEngine/iwb_biu/wb_stb_o_reg_i_1/O
	net (fo=2, routed)	0.434	6.788		cpuEngine/iwb_biu/wb_stb_o0
SLICE_X0Y55	FDCE			r	cpuEngine/iwb_biu/wb_stb_o_reg/D
	(clock wbClk_4 rise edge)	10.000	10.000	r	
P23		0.000	10.000	r	sysClk (IN)
	net (fo=0)	0.000	10.000		sysClk
P23	IBUF (Prop_ibuf_I_O)	0.692	10.692	r	clkin1_buf/O
	net (fo=2, routed)	0.986	11.678		clkgen/sysClk_int
MMCME2_ADV_X0Y0	MMCME2_ADV (Prop_mmcme2_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	BUFG (Prop_bufg_I_O)	0.083	7.372	r	clkgen/clkout2_buf/O
	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“Timing Path Details”

如下图所示，Vivado IDE 中的“Timing Path Details”（时序路径详情）所显示的信息与前图中所示文本报告中显示的信息相同。

图 56：**Vivado IDE 中的“Timing Path Details”**



Source Clock Path

Delay Type	Incr (ns)	Path (n...	Location	Netlist Resource(s)
(clock fftClk_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	clkin1_buf/I
IBUF (Prop_ibuf_I_O)	(r) 0.767	5.767	Site: P23	clkin1_buf/O
net (fo=2, routed)	1.081	6.848		clkgen/sysClk_int
			Site: MMCM..._ADV_X0Y0	clkgen/mmcm_adv_inst/CLKIN1
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT5)	(r)...014	-0.166	Site: MMCM..._ADV_X0Y0	clkgen/mmcm_adv_inst/CLKOUT5
net (fo=1, routed)	1.732	1.566		clkgen/fftClk_0
			Site: BUFGCTRL_X0Y5	clkgen/clkout6_buf/I
BUFG (Prop_bufg_I_O)	(r) 0.093	1.659	Site: BUFGCTRL_X0Y5	clkgen/clkout6_buf/O
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_buf/I
IBUF (Prop_ibuf_I_O)	(r) 0.692	10.692	Site: P23	clkin1_buf/O
net (fo=2, routed)	0.986	11.678		clkgen/sysClk_int
			Site: MMCM..._ADV_X0Y0	clkgen/mmcm_adv_inst/CLKIN1
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT1)	(r)...008	5.670	Site: MMCM..._ADV_X0Y0	clkgen/mmcm_adv_inst/CLKOUT1
net (fo=1, routed)	1.619	7.289		clkgen/wbClk_4
			Site: BUFGCTRL_X0Y3	clkgen/clkout2_buf/I
BUFG (Prop_bufg_I_O)	(r) 0.083	7.372	Site: BUFGCTRL_X0Y3	clkgen/clkout2_buf/O
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“Incremental Compile”（增量编译）时，则显示 6 个列：

- 位置

  表示器件上单元或端口所在位置。

- Delay Type（延迟类型）

  表示路径遵循的 unisim 原语和特定时序弧。对于信号线，它会显示扇出 (`fo`) 及其状态。信号线可处于以下状态：

- “Unplaced”（未布局）：驱动和负载尚未布局。
- “Estimated”（估算）：驱动和/或负载已布局。已部分布线的信号线同样报告为“estimated”。
- “Routed”（已布线）：驱动和负载均已布局，并且信号线已完全布线。

- Incr(ns) (text report) / Delay (IDE report)，即，增量（文本报告）/延迟（IDE 报告）

  表示与 unisim 原语时序弧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。

- Path(ns) (text report) / Cumulative (IDE report)，即，路径（文本报告）/累积（IDE 报告）

  表示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。

- Netlist Resource(s) (text report) / Logical Resource (IDE report)，即，网表资源（文本报告）/逻辑资源（IDE 报告）

  表示遍历的网表对象的名称。

- Pin Reuse (Incremental Compile only)（管脚复用（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）

初始时钟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相，具体取决于时序弧的性质。例如，位于反相器输入处的上升沿在输出处会变为下降沿。

时钟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”（时序汇总报告）的文本版本。您还可在存储器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论，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“分析并解决时序违例”。

第 5 章

报告

本章探讨了各种类型的报告：

- 常规报告 涵盖了各种基于网表的报告以及功耗和 DRC。
- 时序报告 涵盖了与时序约束相关的所有报告。
- 时序收敛报告 涵盖了设计收敛阶段使用的报告。虽然这其中也包括基于时序的分析和基于网表的分析，但重点在于两者交互所导致的时序收敛困难。

常规报告

Report Utilization

“Report Utilization”（使用率报告）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使用率。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 命令用法的详细信息，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 AMD UltraScale™ 系列和 AMD UltraScale+™ 系列。每份报告都包含用于运行的器件和以下方面的使用率：

- 网表逻辑
  - LUT
  - MuxFx
  - 寄存器
  - slice
  - LUT（作为存储器）
  - LUT 触发器对
  - LUT（作为逻辑）
  - 进位逻辑

  每个类别中都可能包含其他项。
- CLB 分布。这提供了布局后网表逻辑如何映射到物理站点的详细信息。
- BLOCKRAM
  - BlockRam
  - UltraRAM
  - FIFO

- 算术
  - 包含的 DSP 资源
- I/O 资源
- 时钟资源
- 特定的器件资源。例如：
  - STARTUPE2
  - XADC
- 原语类型计数（按使用率排序）
- 黑盒
- 例化网表
- SLR 交汇使用率

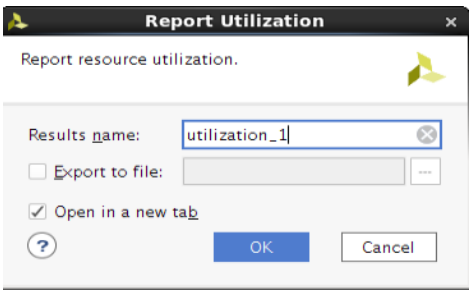
从 Tcl 控制台运行此报告时，其中可包含使用 `-cells` 选项时的特定层级单元的使用率。从 AMD Vivado™ IDE 运行此报告时，此信息会显示在交互表格中。

当逻辑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“Report Utilization”

要从 Vivado IDE 生成“Utilization Report”，请选择“Reports”→“Report Utilization”（报告 > 使用率报告）。

图 57：**“Report Utilization”对话框**



“Results Name”字段

在“Report Utilization”（使用率报告）对话框顶部的“Results Name”（结果名称）字段中指定结果窗口的名称。

等效的 Tcl 命令：

```
report_utilization -name utilization_1
```

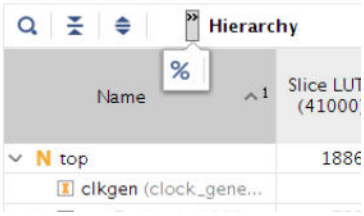
下图显示了详细使用率报告。

图 58：**Report Utilization**

Name	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	
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使用率数值或使用率百分比。

图 59：**使用率报告百分比切换**



显示特定单元的使用率

选择 `-cells` 选项时，生成报告会显示指定单元及其子单元的使用率。

```
-cells {cell_name_list}
```

可从目标单元级别排除特定单元：

```
-exclude_cells {cell_name_list}
```

显示特定 Pblock 的使用率

选择 `-pblocks` 选项时，使用率报告反映的是指定 Pblock 的规格。仅允许指定一个父级 Pblock。可用资源可反映父级 Pblock 范围，且使用的资源细分为父级 Pblock、子级 Pblock 和未分配的 Pblock。此细分允许您对指定父级 Pblock 资源的竞争要求进行评估。仅支持在 Tcl 模式下使用这些命令行选项。

```
-pblocks {Pblock}
-exclude_child_pblocks
-exclude_non_assigned
```

指定这些选项时，报告中会包含另两个与指定 Pblock 相关的表格。在汇总信息中会打印与父级 Pblock 和任意子级 Pblock 相关的属性以及有关其 SLR 布局的部分信息。在第二个表格中会打印时钟区域统计数据。

图 60：**Pblock 汇总报告**

```
1. Pblock Summary
-----------------
+-------+--------------+-------------------------------+--------------------+-----------------+-----------------+
| Index |    Parent    |             Child             | EXCLUDE_PLACEMENT | CONTAIN_ROUTING | SLR(s) Covered |
+-------+--------------+-------------------------------+--------------------+-----------------+-----------------+
| 1     | pblock_arnd4 |                               |                  0 |               0 |           SLR0 |
| 2     |              | pblock_transformLoop[0].ct_1 |                  0 |               0 |           SLR0 |
| 3     |              | pblock_transformLoop[1].ct   |                  0 |               0 |           SLR0 |
+-------+--------------+-------------------------------+--------------------+-----------------+-----------------+

2. Clock Region Statistics
--------------------------
+-------------+--------------------+
| CLOCKREGION | Pblock Sites in CR |
+-------------+--------------------+
| X1Y3        |            100.00% |
+-------------+--------------------+
```

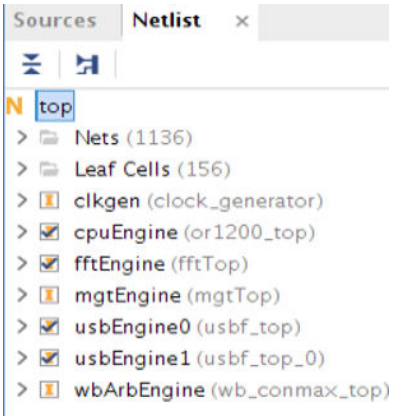
布局前和布局后的使用率数值因 LUT 组合以及未分配单元（布局前无法考量）而异。

使用 `-pblocks` 时，使用率表包含以下如下列：

- “Parent”（父级）：仅分配到父级 Pblock。
- “Child”（子级）：仅分配到子级 Pblock。
- “Non-Assigned”（未分配）：表示在指定 Pblock 定义的区域中已使用，但未分配至指定 Pblock 或其子级 Pblock 的资源总量。
- “Used”（已用）：指定 Pblock 定义的区域内已用资源总量
- “Fixed”（固定）：指定 Pblock 已定义区域内 LOC 约束已固定的资源总量
- “Prohibited”（禁止）：已定义的区域内由于 PROHIBIT 约束而导致禁止使用的资源。
- “Available”（可用）：指定 Pblock 定义的区域内可用的资源总量。
- “Util%”（使用率）：“Used”/“Available”

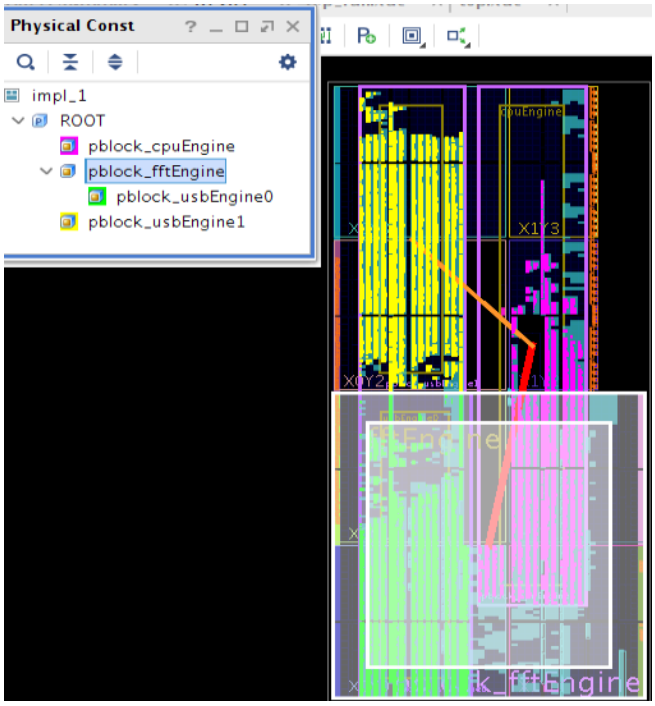
以下示例有助于您更全面理解此报告。下图显示了设计示例层级。

图 61：**设计示例层级**



下图显示了 Pblock 矩形。每个 Pblock 内来自布线后网表的资源都会高亮显示。

图 62：**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，请运行 `report_utilization`（不含任何选项）。

图 63：**顶层使用率报告**

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 64：**Pblock pblock_usbEngine1 的使用率报告**

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“Used”和“Available”）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“Non-Assigned”（未分配）。

```
report_utilization -pblocks pblock_fftEngine
```

图 65：**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 O5 output only	12	11	0	0			
using O6 output only	974	4651	3986	9588			
using O5 and O6	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 O5 output only	1	0	0	1			
using O6 output only	0	0	0	0			
using O5 and O6	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 66：**不含未分配资源的使用率报告**

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 O5 output only	12	11	11			
using O6 output only	974	4651	5613			
using O5 and O6	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 O5 output only	1	0	1			
using O6 output only	0	0	0			
using O5 and O6	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 8：**含 Pblock 分配的报告表**

大小写	标题	描述	报告
1	整个器件报告（根 Pblock）：`report_utilization`	EXCLUDE_PLACEMENT 对使用率报告无影响。	“Util%”：“Used”/“Available”。
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。	“Non-Assigned”（未分配）：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。 “Fixed”（固定）：父级 Pblock 边界内已固定的单元总数。 “Used”：父级 Pblock 边界内已布局的“Parent”+“Child”+“Non-Assigned”单元数量。 “Available”（可用）：父级 Pblock 边界内物理资源总数。 “Util%”：“Used”/“Available”。
		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。	“Non-Assigned”（未分配）：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。 “Fixed”（固定）：报告区域内已固定的单元总数。 “Used”（已用）：父级 Pblock 单元数（不包括子级 Pblock 单元数）。 “Available”（可用）：报告区域内物理资源总数。 “Util%”：“Used”/“Available”。
3	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。	“Available”：Pblock 物理资源的并集。 “Util%”：“Used”/“Available”。

显示 SLR 使用率

选择 `-slr` 选项时，生成的报告会显示 SLR 相关使用率。SLR 使用率表包含以下四张表：

- “SLR Connectivity”（SLR 连接）
- “SLR Connectivity Matrix”（SLR 连接矩阵）
- “SLR CLB Logic and Dedicated Block Utilization”（SLR CLB 逻辑和专用块使用率）
- “SLR IO Utilization”（SLR IO 使用率）

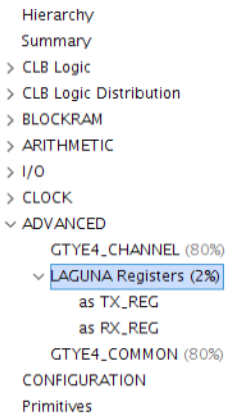
默认情况下，运行 `report_utilization` 时也会显示这些表。“SLR Connectivity”表可显示用于 SLR 每侧的 TX 和 RX 方向的 LAGUNA 寄存器。

图 67：**GUI 中的“SLR Connectivity”报告**

Connectivity	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 寄存器的使用总量。

图 68：**GUI 中的“LAGUNA”寄存器报告**



“SLR CLB Logic and Dedicated Block Utilization”表可显示每个 SLR 的资源使用率。

图 69：**每个 SLR 中的使用率**

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

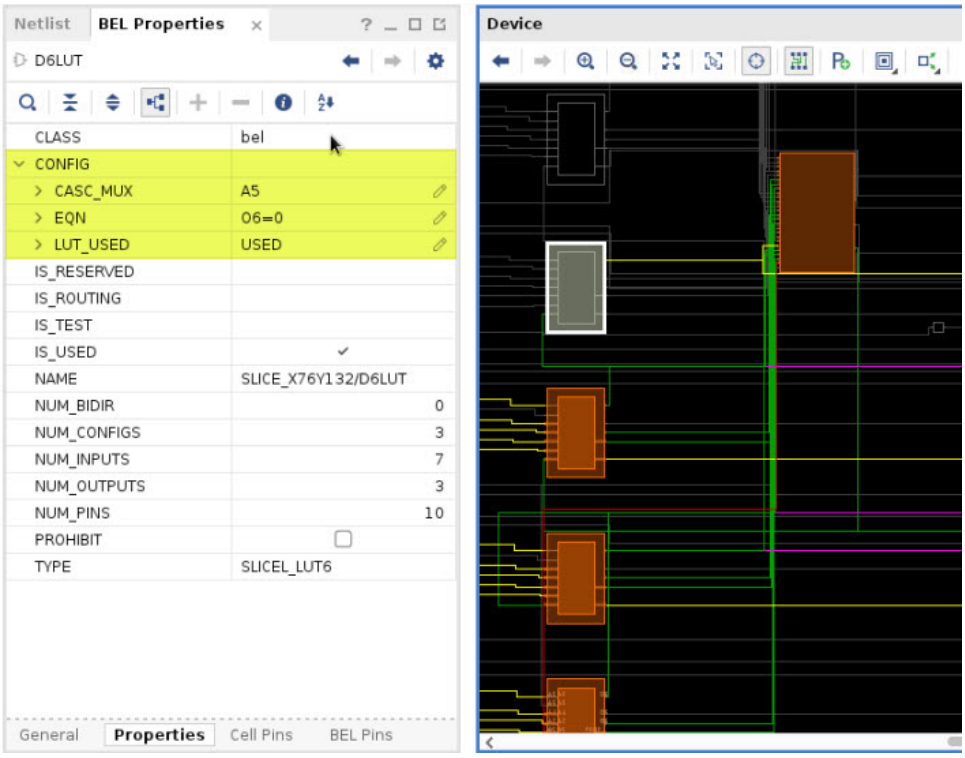
选择以下选项时，可将报告限制为显示部分特定层级的相关信息。根据层级报告使用率时，可指定要报告的层级深度。默认深度为 0，即默认情况下 `-hierarchical` 仅报告顶层相关信息。

```
-hierarchical
-hierarchical_depth <args>
-hierarchical_percentage
```

显示 Packthru

Packthru（封装直通）是已用的 BEL，其中不含关联的网表单元。通常添加这些 Packthru 的目的是为了能以合规方式布线到无法以其他方式完成布线的各元件，或者为了简化布线。Packthru LUT 将使用一项输出和最多一项输入。如果需要，那么此类 LUT 的其他部分仍可用。仅当此 site 位置无需满足任何其他用途时，才能将寄存器用作为 packthru。

在以下示例中，packthru LUT 配置为将逻辑 0 从该 LUT 的 O6 输出驱动到 LOOKAHEAD BEL 内。O5 输出空闲可用，可添加更小的 LUT 以使用此 BEL。




```
-packthru
```

显示自定义表格报告

选中以下选项时，可自定义报告以便仅显示某些类型的资源及层级深度。

```
-spreadsheet_table <args>
-spreadsheet_depth
```

显示禁止信息和已固定信息

生成文本版本报告时，会在此报告的每一行中添加有关禁止的站点 (site) 和已固定的单元的信息。在 Tcl 控制台中运行 `report_utilization` 且不使用 `-name` 选项时，会发生此操作。

禁止会更改可用资源的计算方式：Available Resources = Total Resources – Prohibit，即，可用资源 = 总资源 – 禁止的资源。

如果单元的 `IS_LOC_FIXED` 属性设为 1，即表示单元已固定。在单元上设置 `LOC` 属性时，会自动发生此操作。

Report I/O

“I/O Report”（I/O 报告）用于替代 AMD Design Suite PAD 文件。“I/O Report”可列出：

- “Pin Number”（管脚编号）：表示器件中的所有管脚
- “Signal Name”（信号名称）：表示分配给管脚的用户 I/O 的名称
- “Bank Type”（bank 类型）：表示 I/O 所在的 bank 类型，类型包括：“High Range”（高量程）、“High Performance”（高性能）、“Dedicated”（专用）等

- “Pin Name”（管脚名称）：表示管脚名称
- “Use”（用途）：表示 I/O 使用类型，类型包括：“Input”（输入）、“Output”（输出）、“Power/Ground”（功耗/接地）、“Unconnected”（未连接）等
- “I/O Standard”（I/O 标准）：表示用户 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”（I/O 端口）窗口。

- “I/O Bank Number”（I/O bank 编号）：表示管脚所在的 I/O bank
- “Drive (mA)”（驱动）：表示驱动强度（以毫安为单位）
- “Slew Rate”（压摆率）：表示缓冲器的压摆率配置，值为 Fast 或 Slow
- “Termination”（终端）：表示片上/片外终端设置
- “Voltage”（电压）：表示各管脚的值，包括 VCCO、VCCAUX 和相关管脚
- “Constraint”（约束）：如果管脚已由用户约束，则显示为“Fixed”
- “Signal Integrity”（信号完整性）：表示管脚的信号完整性

Report Clock Utilization

“Clock Utilization”（时钟使用率）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使用率。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使用率。“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
- 与每条时钟信号线关联的时序时钟名称和周期
- 每个时钟区域的时钟设置和互连结构负载使用率
- 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”（时钟使用率）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板级原理图。

“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：

- 时钟原语使用率
- 全局时钟资源
- 全局时钟源详情
- 局部时钟资源
- 时钟区域使用率详情
- 全局时钟布局详情

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：

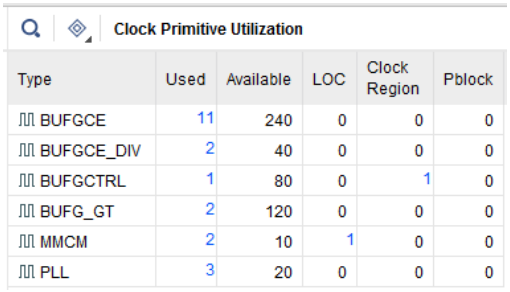
- 针对时钟缓冲器驱动的每条信号线分配唯一全局 ID“`g<n>`”
- 针对时钟生成器（例如，MMCM 或输入缓冲器）分配唯一源 ID“`src<n>`”。
- 对于不使用全局时钟资源布线的时钟信号分配唯一局部 ID“`<n>`”。

“Global Source ID”（全局源 ID）和“Local ID”（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”（时钟原语使用率）表可显示每种时钟原语类型及其物理约束的使用率汇总信息。

图 70：**Report Clock Utilization -“Clock Primitive Utilization”表**



Clock Primitive Utilization

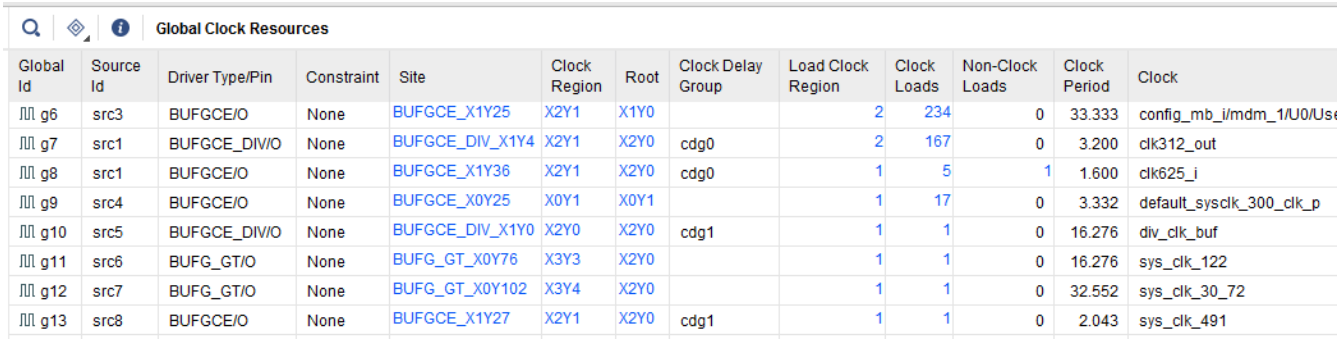
Type	Used	Available	LOC	Clock Region	Pblock
BUFGCE	11	240	0	0	0
BUFGCE_DIV	2	40	0	0	0
BUFGCTRL	1	80	0	1	0
BUFG_GT	2	120	0	0	0
MMCM	2	10	1	0	0
PLL	3	20	0	0	0

注释： “Clock Region”（时钟区域）约束不适用于 7 系列器件。

“Global Clock Resources”表

“Global Clock Resources”（全局时钟资源）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 71：**Report Clock Utilization -“Global Clock Resources”表**



Global Clock Resources

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 9：**“Global Clock Resources”表详情**

列	描述
Global Id（全局 ID）	表示全局时钟信号线唯一 ID
Source Id（源 ID）	表示生成原语（连接到时钟缓冲器）的时钟 ID
Driver Type/Pin（驱动程序类型/管脚）	表示连接到时钟信号线的原语管脚

表 9：**“Global Clock Resources”表详情** (续)

列	描述
Constraint（约束）	表示用户物理约束，其中包含应用于时钟缓冲器的最高优先级。优先级规则如下所示： 1. LOC 2. CLOCK_REGION* 3. PBLOCK * 不适用于 7 系列。
Site（站点）	表示由用户或者由 Vivado 实现工具所设置的时钟缓冲器位置。
Clock Region（时钟区域）	表示缓冲器所在的器件时钟区域。 不适用于 7 系列。
Root（根）	表示时钟信号线 CLOCK_ROOT 所在的时钟区域。 不适用于 7 系列。
Clock Delay Group（时钟延迟组）	表示由用户指定的时钟信号线组的名称，用于强制由 AMD Vivado™ 实现工具进行布线匹配。 不适用于 7 系列。
Load Clock Region（负载时钟区域）	表示时钟信号线负载所在的时钟区域数量。
Clock Loads（时钟负载）	表示已通过时钟管脚负载连接的单元数量。
Non-Clock Loads（非时钟负载）	表示非时钟管脚负载（例如，FDCE/CE 管脚）的数量。
Clock Period（时钟周期）	表示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。
Clock（时钟）	表示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”。
Driver Pin（驱动程序管脚）	表示时钟信号线驱动管脚的逻辑名称。
Net（信号线）	连接到时钟驱动管脚的时钟信号段的逻辑名称。
GCLK_DESKEW	识别该时钟上是否已启用校准。（仅限 Versal）
CLOCK_EXPANSION_WINDOW	布局后报告 CLOCK_EXPANSION_WINDOW 属性。（仅限 Versal）
Clock Low Fanout（时钟低扇出）	识别是否已应用 CLOCK_LOW_FANOUT 属性。

“Global Clock Source Details”表

“Global Clock Source Details”（全局时钟源详情）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 MMCM (src0/src1) 的每个输出到时钟缓冲器的连接。src1 的输出 CLKOUT0 用于驱动 2 个全局时钟：g7 和 g8。

图 72：**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 10：**“Global Clock Source Details”表中包含以下列**

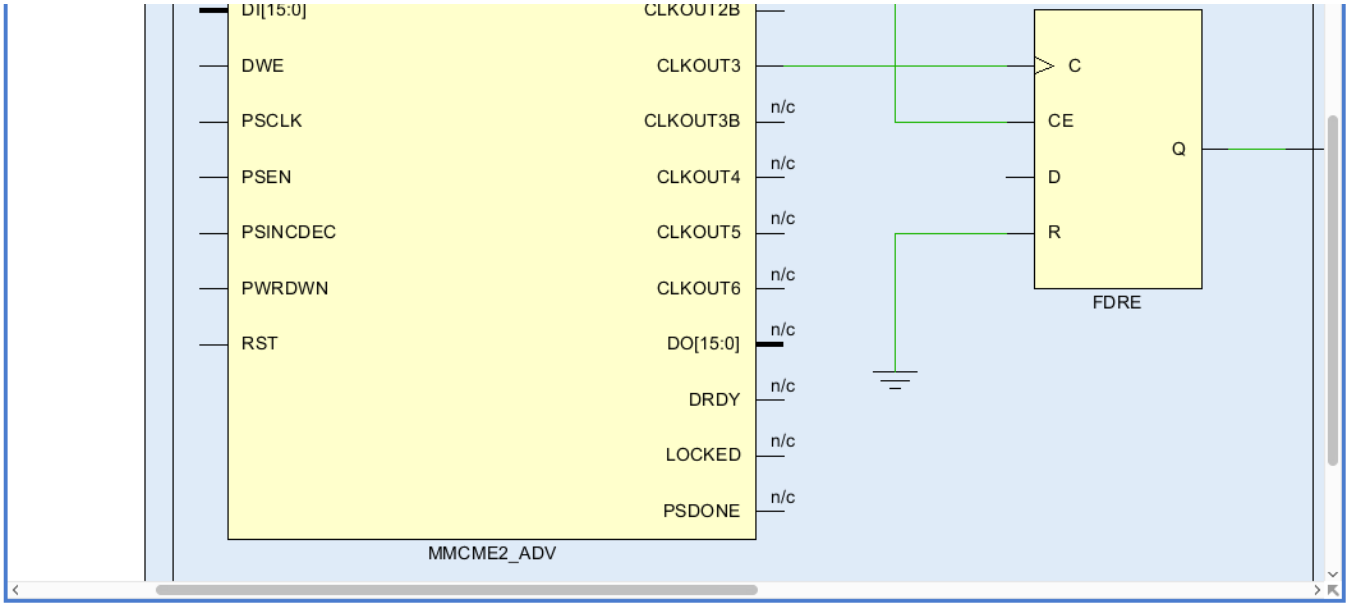
列	描述
Source Id（源 ID）	表示生成原语的时钟的 ID。
Global Id（全局 ID）	表示由全局时钟源管脚驱动的全局时钟 ID。
Driver Type/Pin（驱动程序类型/管脚）	表示生成时钟的输出原语管脚。
Constraint（约束）	表示用户物理约束，其中包含应用于时钟缓冲器的最高优先级。优先级规则如下所示： 1. `LOC` 2. `PBLOCK`
Site（站点）	表示由用户或 Vivado 实现工具所设置的全局时钟源位置。
Clock Region（时钟区域）	表示时钟源所在的器件时钟区域。
Clock Loads（时钟负载）	表示已连接到全局时钟源管脚的时钟管脚负载数量。
Non-Clock Loads（非时钟负载）	表示非时钟管脚负载（例如，FDCE/CE 管脚）的数量。
Source Clock Period（源时钟周期）	表示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。
Clock（时钟）	表示由全局时钟源管脚生成的时序时钟的名称。如果在同一个时钟信号线上有多个时钟在传输，则报告“Multiple”。
Driver Pin（驱动程序管脚）	表示全局时钟源管脚的逻辑名称。
Net（信号线）	表示连接到全局时钟源管脚的时钟信号线段的逻辑名称。

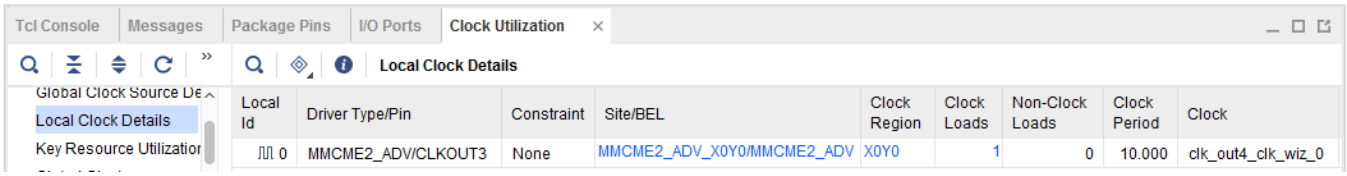
“Local Clock Details”表

仅当设计中发现局部时钟时，才会报告“Local Clock Details”（局部时钟详情）表。局部时钟表示以常规互连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冲器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 73：**Report Clock Utilization - 局部时钟示例**

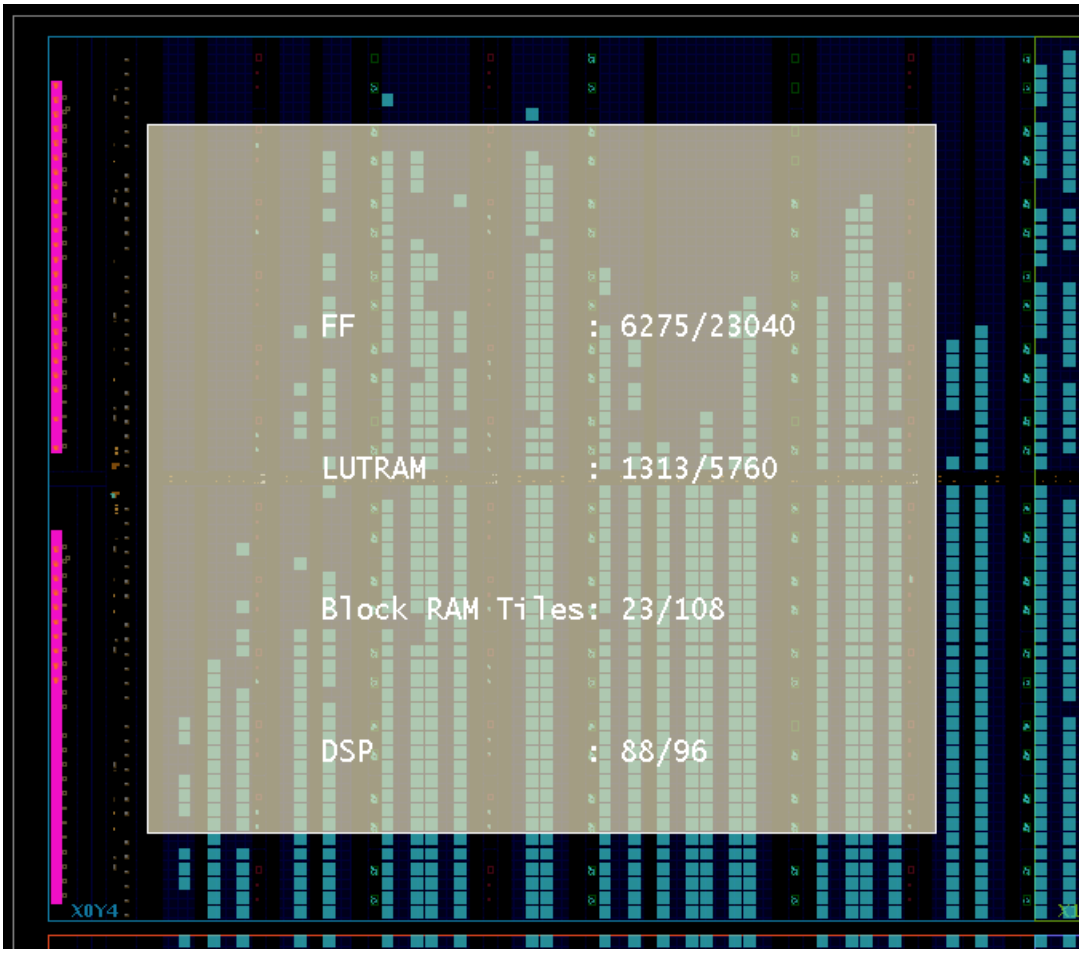




“Clock Regions”表

“Clock Regions”（时钟区域）部分仅适用于 UltraScale 器件系列，其中包含的几张表提供了每个时钟区域的原语和布线资源使用率。在“Clock Utilization”（时钟使用率）窗口中，“Show Metrics In Device Window”（在器件窗口中显示指标）按钮  可用于选择要在“Device”（器件）窗口中的每个时钟区域上显示的资源类型，如下图所示。

图 74：**Report Clock Utilization -“Device”窗口中的时钟区域资源使用率指标**



“Clock Regions”表包括：

- “Clock Primitives”（时钟原语）：每个时钟区域内的每种时钟原语类型的使用率。
- “Load Primitives”（负载原语）：每个时钟区域内非时钟时序原语的使用率。

对于“Clock Primitives”和“Load Primitives”表，“Global Clock”（全局时钟）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。

- “Global Clock Summary”（全局时钟汇总）：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率（按时钟区域划分），如下图所示。该表仅显示在文本报告中。

图 75：**Report Clock Utilization -“Global Clock Summary”示例**

```
6. Clock Regions : Global Clock Summary
---------------------------------------

+----+----+----+----+----+
|    | 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 |
+----+----+----+----+----+
```

- “Routing Resource Utilization”（布线资源使用率）：按类型和时钟区域显示全局时钟布线使用率。

“Key Resource Utilization”表

“Key Resource Utilization”（关键资源使用率）表仅适用于 7 系列器件，它等同于对应 UltraScale 器件的所有“Clock Regions”（时钟区域）表的组合。“Global Clock Summary”（全局时钟汇总）表同样只能以文本报告方式显示。

“Global Clocks”表

“Global Clocks”（全局时钟）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“Global ID”（全局 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的时钟区域，在这些区域中可对每条全局时钟信号线进行布线。

列描述与“Clock Primitive Utilization”（时钟原语使用率）、“Global Clock Resources”（全局时钟资源）和“Global Clock Source Details”（全局时钟源详情）表中的列描述相同。

对于 UltraScale 器件，如果在任一时钟区域内，时钟信号线无需驱动任何负载即可完成布线，那么其“Global ID”以“+”字符来标记，如下图所示。

图 76：**Report Clock Utilization -“Clock Region Cell Placement”示例**

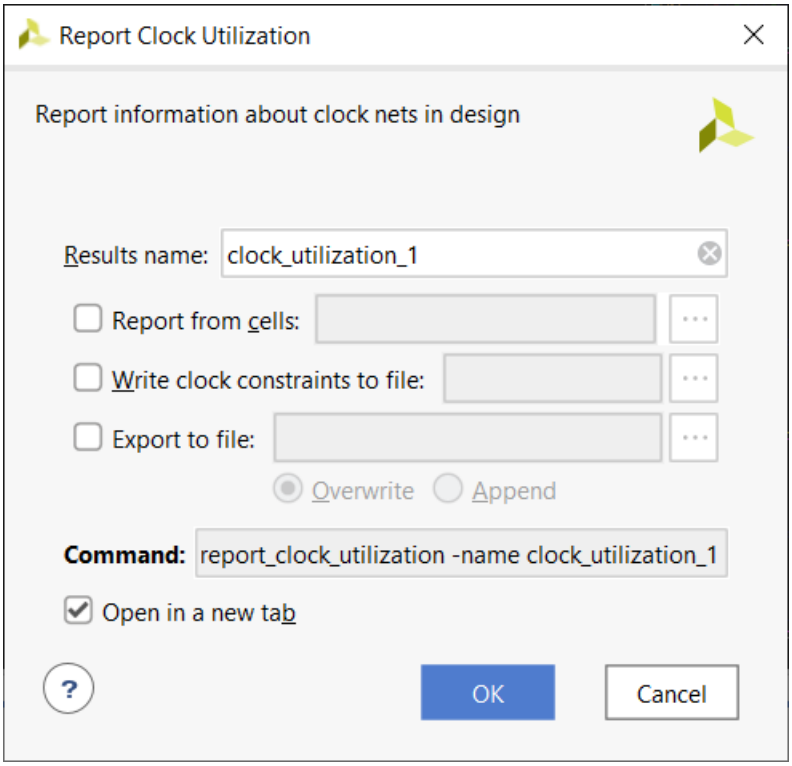
Clock Region Cell Placement

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

生成“Clock Utilization Report”

要在 Vivado IDE 中生成“Clock Utilization Report”，请选中“Reports”→“Report Clock Utilization”（报告 > 时钟使用率报告）。

图 77：“Report Clock Utilization”对话框



等效的 Tcl 命令：

```
report_clock_utilization -name clock_utilization_1
```

“Results Name”字段

在“Report Clock Utilization”（时钟使用率报告）对话框的“Results Name”（结果名称）字段中，指定报告的图形窗口的名称。

等效的 Tcl 选项：

```
-name <windowName>
```

Show Clock Roots Only

选中该选项时，“Global Clock Resources”（全局时钟资源）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
```

Write Clock Constraints to File

选择该选项并指定新约束文件的名称，这样即可导出时钟源并加载对应于存储器中的设计布局信息的物理约束。

等效的 Tcl 选项：

```
-write_xdc <filename>
```

Export to File

除生成 GUI 报告外，您还可通过选择“Export to file”（导出至文件）并在右侧字段中指定文件名来将结果写入文件。单击“Browse”（浏览）按钮可选择不同目录。

等效的 Tcl 选项：

```
-file <arg>
```

选择“Overwrite”（覆盖）选项，即可用新的分析结果覆盖现有文件。选择“Append”（追加）可追加新结果。

等效的 Tcl 选项：

```
-append
```

Report DRC

布线器运行前，本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“Critical Warnings”（严重警告）。特定检查的严重性可能会在后续流程中提升。

Report DRC 可运行常用“Design Rule”（设计规则）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 缓冲器、时钟缓冲器和综合后设计包含的其他原语。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 和其他布局。
- 对 UltraScale、GT、IODELAY 和其他原语上的属性连线执行基本检查。
- 检查连接，例如，检查由同步信号驱动的某些管脚。

  注释：不会针对非关联设计运行连接检查
- 在考量所有可用布局布线的情况下运行相同的 DRC。
- DRC 具有 4 种安全级别：Info（参考）、Warning（警告）、Critical Warning（严重警告）和 Error（错误）。“Critical Warnings”和“Errors”当前不会阻止设计流程。

如果存在严重程度级别为错误的 DRC 违例，则应停止流程。

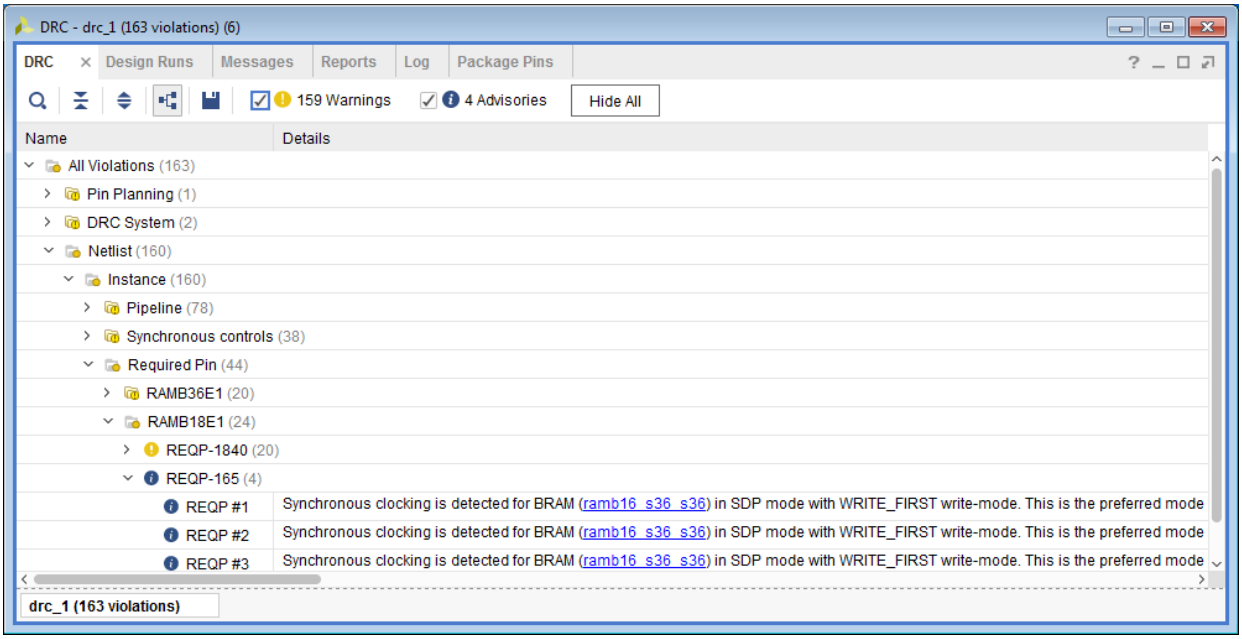
实现流程的步骤同样会运行 DRC，这些步骤可能在关键点停止运行此流程。布局器和布线器会检查导致阻止布局的问题。根据阶段，某些消息的严重性级别可能较低。这些 DRC 标记条件不会阻止 `opt_design`、`place_design` 或 `route_design` 完成，但可能会导致硬件故障。

Report DRC 会检查用户是否约束了所有设计端口的封装管脚位置和 I/O 标准。如果这些约束不存在，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”。如果没有这些约束，这些工具就不会生成比特流。

在流程初期降低严重性可便于您在判定最终管脚分配 (pinout) 前运行设计直至完成实现迭代。您必须运行比特流生成以实现完整的 DRC 验收。

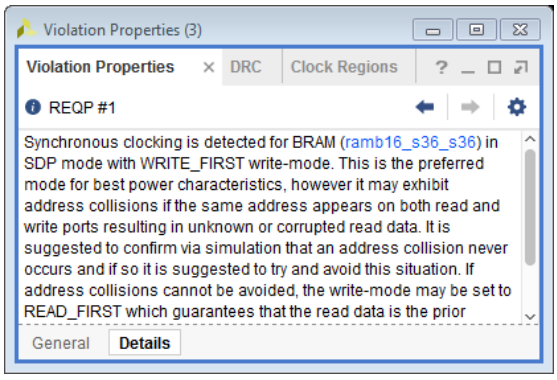
下图显示了 Report DRC 的 Vivado IDE 图形用户界面。

图 78：**DRC 报告**



单击任一 DRC 可打开属性并显示详细消息。请查看“Properties”（属性）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 79：**“Violation Properties”对话框**



DRC 报告为静态报告。您必须重新运行 Report DRC 才能使其反映设计变更。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`。

Report Route Status

“Route Status Report”（布线状态报告）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

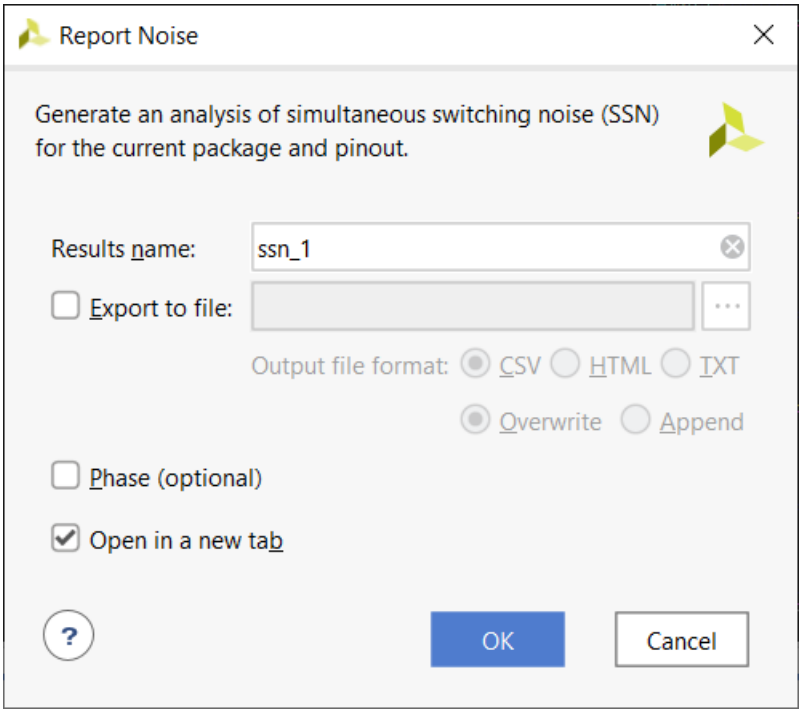
以下是已完全布线的设计的“Report Route Status”示例：

```
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 :
    ------------------------------------------- :------------ :
```

Report Noise

“Report Noise”（噪声报告）命令用于为 AMD 执行同步开关噪声 (SSN) 计算。“Noise”（噪声）报告会在 Vivado IDE 的下方窗格区域中的新选项卡内打开。您可将结果导出至 CSV 文件或 HTML 文件。

图 80：**“Report Noise”对话框**



“Noise”报告包含 4 个部分：

- “Noise Report”的“Summary”部分
- “Noise Report”的“Messages”部分
- “Noise Report”的“I/O Bank Details”部分
- “Noise Report”的“Links”部分

“Noise Report”的“Summary”部分

“Noise Report”（噪声报告）的“Summary”（汇总）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

“Noise Report”的“Messages”部分

“Noise Report”（噪声报告）的“Messages”（消息）部分包含报告期间生成的消息的详细列表。

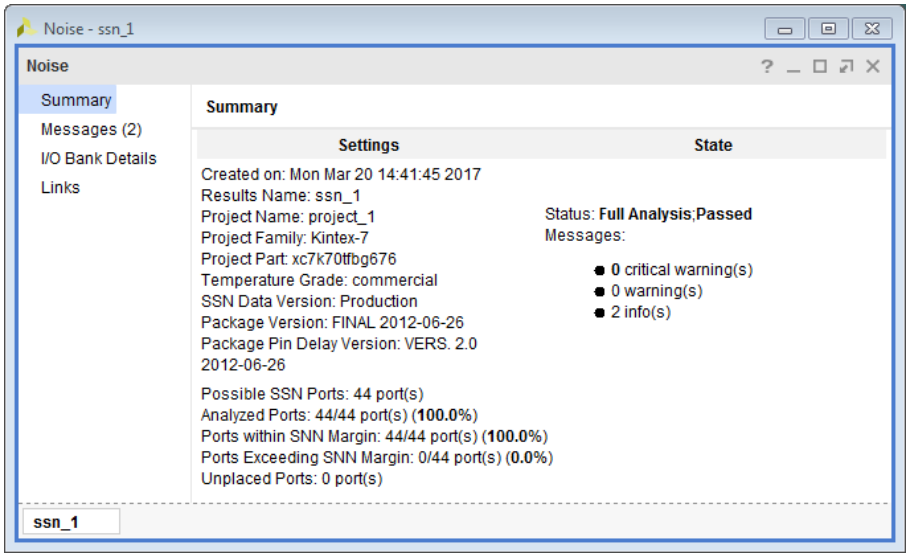
“Noise Report”的“I/O Bank Details”部分

“Noise Report”（噪声报告）的“I/O Bank Details”（I/O Bank 详情）部分包含“Pins”（管脚）、“Standards”（标准）和“Remaining Margin”（剩余裕度）的列表。

“Noise Report”的“Links”部分

“Noise Report”（噪声报告）的“Links”（链接）部分包含指向位于 https://adaptivesupport.amd.com/s/?language=en_US 的文档的链接。

图 81：**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
```

Report Power

在布线后会生成“Power Report”（功耗报告），它基于当前器件工作条件和设计的切换率来报告功耗详情。功耗分析要求网表已完成综合或设计已完成布局布线。

- `set_operating_conditions` 命令用于设置工作条件。
- `set_switching_activity` 命令用于定义切换活动。

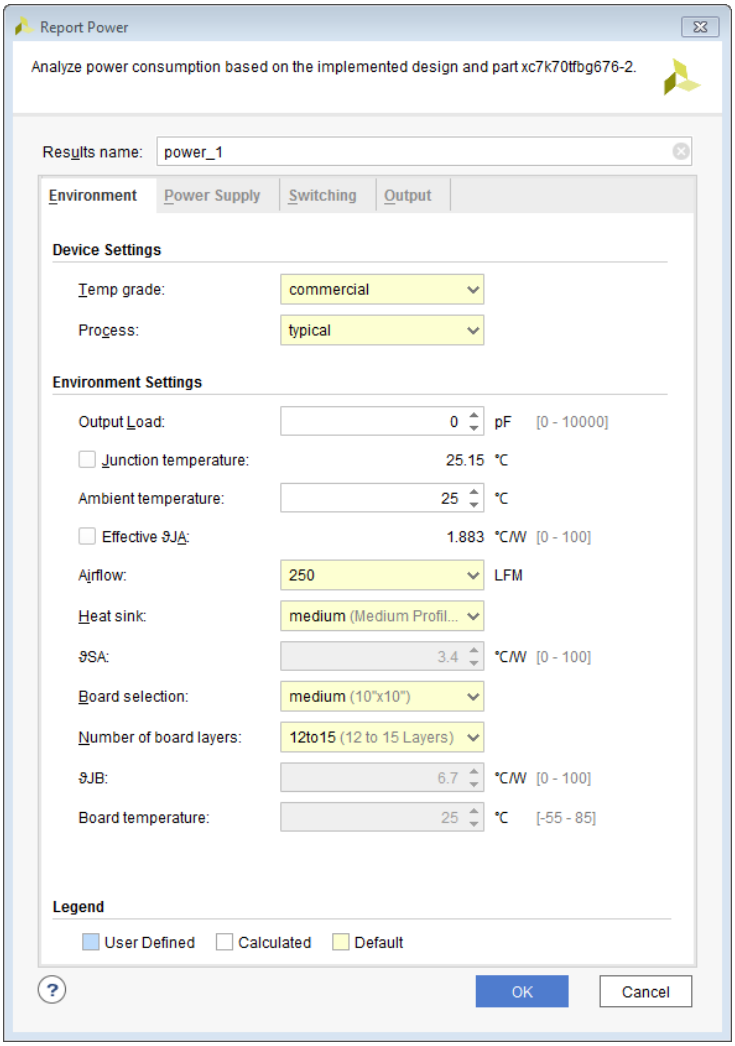
当“综合后设计”或“实现后设计”打开时，即可使用“Report Power”命令。

“Power Report”可用于基于设计输入来估算功耗，包括：

- 散热性能统计数据，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。
- 有关开发板选项的数据，包括开发板层数和板上温度。
- 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 82：“Report Power”对话框

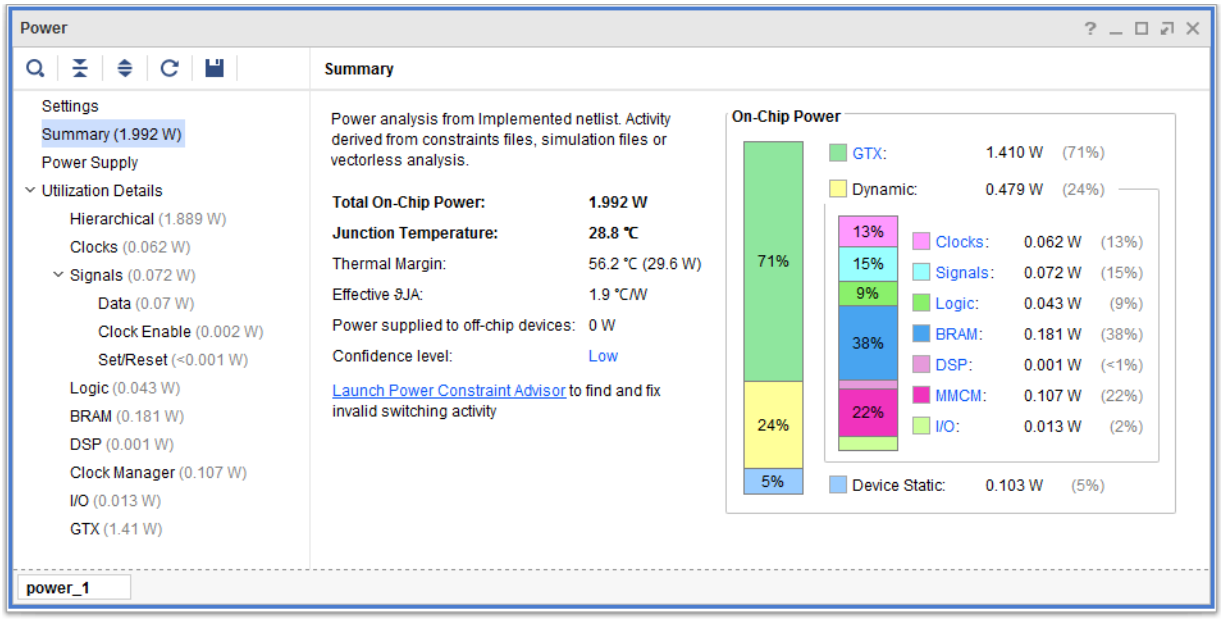


分析功耗报告

“Report Power”（功耗报告）对话框（请参阅下图）可用于根据以下条件分析功耗：

- 设置
- 总功耗
- 层级
- 电压轨
- 块类型

图 83：**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：功耗分析与优化》(UG907)。

默认情况下，实现进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`。

Report RAM Utilization

“Ram Utilization”（RAM 使用率）报告可帮助您分析专用 RAM 块（例如，URAM 和块 RAM）以及分布式 RAM 原语的使用率。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”报告对于由 Vivado 综合推断所得存储器最有效，因为您可将“RTL Memory Array”（RTL 存储器阵列）与 FPGA 中的实际物理实现进行比较。

此报告可显示如下内容：

- 每个存储器原语的使用率
- 阵列大小和维度（仅限推断）
- 存储器的类型
- 存储器原语的使用率
- 所需的存储器性能
- （可选）存储器的流水线使用率（如适用）
- 始于和止于存储器的最差情况逻辑路径

- 功耗效率数据，例如，级联和使能率

此报告也可以 CSV 格式生成。如需对大量数据进行管理和排序，那么首选此方法。

运行 RAM 使用率报告

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_ram_utilization -file ram_util.rpt
```

以下语法将生成此报告和 CSV 文件 `ram_util.csv`。

```
report_ram_utilization -csv ram_util.csv -file ram_util.rpt
```

为了报告所有存储器（包括基于 LUTRAM 的存储器），必须使用 `-include_lutram` 开关：

```
report_ram_utilization  -include_lutram
```

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
```

Report Control Sets

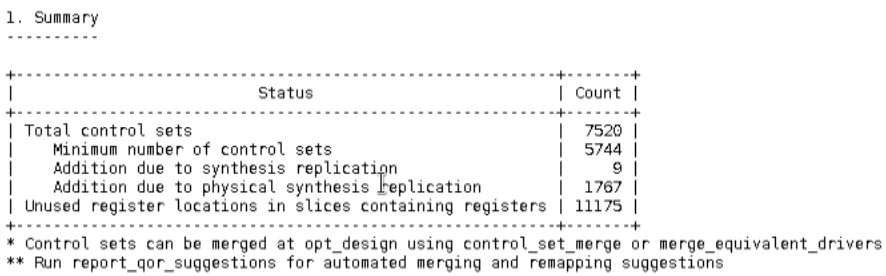
控制集表示时钟信号、时钟使能信号和置位/复位信号的唯一组合。每个 slice 都支持有限数量的控制集，以供位于其中的触发器组合使用。根据所使用的架构，部分控制集允许在 slice 内共享。用户应熟悉目标系列的“Configurable Logic Block”（可配置逻辑块架构）才能理解兼容性规则。

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“Control Sets Report Summary”（控制集报告汇总）示例。请遵循《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中有关控制集数量的建议进行操作。

图 84：**控制集汇总表**



1. Summary

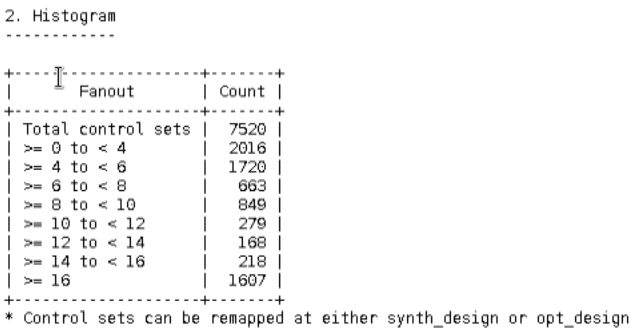
Status	Count
Total control sets	7520
Minimum number of control sets	5744
Addition due to synthesis replication	9
Addition due to physical synthesis replication	1767
Unused register locations in slices containing registers	11175

* Control sets can be merged at opt_design using control_set_merge or merge_equivalent_drivers
** Run report_qor_suggestions 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。以下提供了直方图汇总报告以供概览：

图 85：**控制集直方图表**



2. Histogram

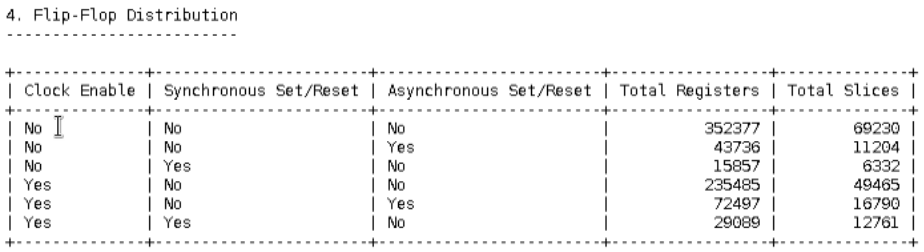
Fanout	Count
Total control sets	7520
>= 0 to < 4	2016
>= 4 to < 6	1720
>= 6 to < 8	663
>= 8 to < 10	849
>= 10 to < 12	279
>= 12 to < 14	168
>= 14 to < 16	218
>= 16	1607

* 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“Flip Flop Distribution”（触发器分布）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 86：**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：

- “Clock Signal”（时钟信号）：逻辑时钟信号名称
- “Enable Signal”（使能信号）：逻辑时钟使能信号名称
- “Set/Reset Signal”（置位/复位信号）：逻辑置位/复位信号名称
- “Slice Load Count”（分片负载计数）：包含连接到控制集的单元的唯一 slice 数量
- “BEL Load Count”（BEL 负载计数）：连接到控制集的单元数量

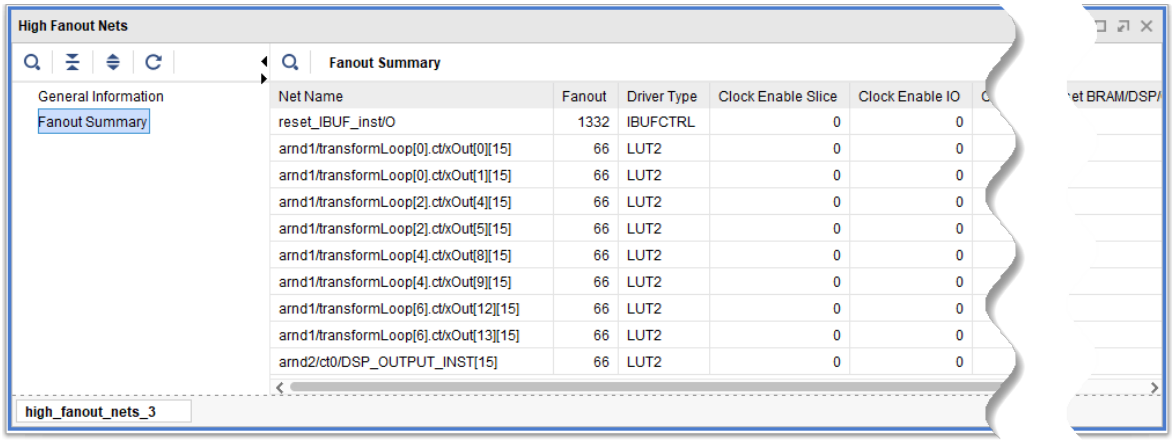
Report High Fanout Nets

`report_high_fanout_nets` 命令用于分析网表并报告具有最高扇出的信号线。此命令可在综合后、布局后或布线后网表上运行。但在布局前，此报告无法传递时钟区域和 SLR 信息。此报告可显示以下信息：

- “Fanout”（扇出）
- “Driver Type”（驱动程序类型）
- “Load Types”（负载类型）
- “Clock Regions”（时钟区域，如适用）
- “SLRs”（SLR，如适用）

下图显示了所生成报告的示例。

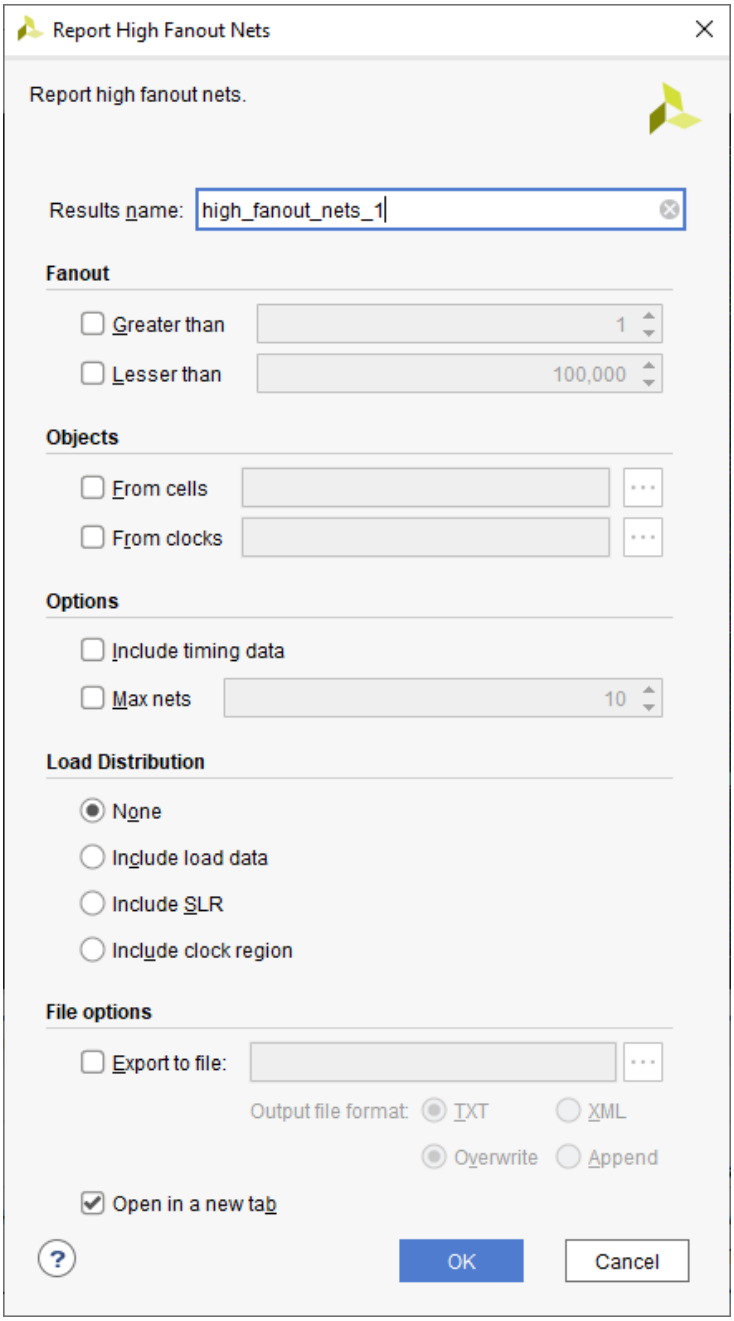
图 87：**“High Fanout Net”报告**



生成“High Fanout Net”报告

要在 Vivado IDE 中打开的设计上生成高扇出信号线报告，请选择“Reports” → “Report High Fanout Nets”（报告 > 高扇出信号线报告）。

图 88：“Report High Fanout Nets”对话框



等效的 Tcl 命令：

```
report_high_fanout_nets -name hfn_1
```

高扇出信号线报告选项

名称

在位于“Report High Fanout Nets”（高扇出信号线报告）对话框顶部的“Results Name”（结果名称）字段中，指定报告的图形窗口。

等效的 Tcl 选项：

```
-name <windowName>
```

“Fanout”（扇出）

最大和最小限制可设为仅报告含指定扇出的信号线。

等效的 Tcl 命令：

```
-fanout_greater_than 1
```

```
-fanout_lesser_than 100000
```

“Objects”（对象）

可基于对象来限制报告。您可将报告作用域限定于特定层级单元。选中此项时，所有源文件均位于限定作用域的单元中。您也可以报告时钟。默认不报告时钟，但可在报告中使用开关来指定。

等效的 Tcl 命令：

```
-cells [get_cells <hierarchical cell>]
```

```
-clocks [get_clocks <clock object>]
```

“Max Nets”（最大信号线数量）

默认报告 10 条信号线。指定 `-max_nets` 开关即可更改该值。在报告中也可添加信号线上的最差情况裕量。

等效的 Tcl 命令：

```
-max_nets <n>
```

```
-timing
```

“Load Distribution”（负载分布）

在报告中也可添加负载类型、时钟区域和 SLR。如需访问全部项，则必须多次运行此命令。

等效的 Tcl 命令：

```
-load_types
```

```
-clock_regions
```

```
-slr
```

“Histogram”（直方图）

可显示直方图，其中包含给定存储器中的风扇计数和信号线数量。如果存在大量扇出较低的信号线，该直方图可用于判定这些信号线是否会造成困难。该表只能以文本模式查看。该表示例如下图所示。

图 89：**直方图表格**

1. Histogram

Fanout	Nets	%
1	3801	63.58
2	1306	21.84
3	376	6.28
4	121	2.02
5-10	293	4.90
11-50	48	0.80
51-100	32	0.53
101-500	0	0.00
>500	1	0.01
ALL	5978	100.00

等效的 Tcl 命令：

```
-histogram
```

注释：在每条信号线的时序报告中均可显示扇出，并且可在信号线上使用 FLAT_PIN_COUNT 属性来判定扇出。

要解决高扇出问题，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“优化高扇出信号线”和“关键路径上的高扇出”。

时序报告

Report Timing

综合后，可根据需要在流程中随时查阅“Report Timing”（时序报告）以查看特定时序路径，对“Report Timing Summary”（时序汇总报告）中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“Pulse Width”（脉冲宽度）报告。

从 Tcl 控制台或从 GUI 运行时，可使用 `-cells` 选项将此时序报告限定为一个或多个层级单元。限定报告作用域后，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”

如果设计已加载到存储器中，则可以从“Clock Interaction Report”（时钟交互报告）或“Report Timing Summary”（时序汇总报告）路径列表运行“Report Timing”（时序报告），也可从菜单中运行。

从菜单运行“Report Timing”

如需从菜单运行“Report Timing”，请选中“Reports”→“Timing”→“Report Timing”（报告 > 时序 > 时序报告）。

从“Clock Interaction Report”运行“Report Timing”

要从“Clock Interaction Report”（时钟交互报告）运行“Report Timing”（时序报告），请执行以下操作：

1. 选中一个“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“Paths List”运行“Report Timing”

要从“Paths List”（路径列表）运行“Report Timing”（时序报告），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：

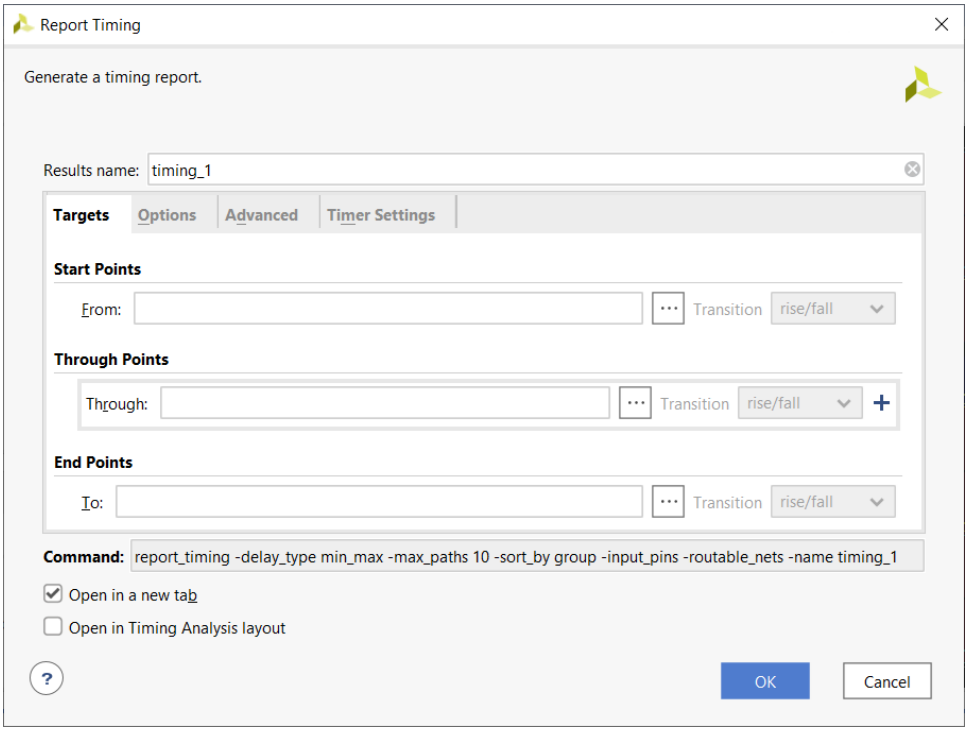
- 位于对话框底部的“Command”（命令）字段中，以及
- 执行之后在 Tcl 控制台中

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”（时序汇总报告）选项相同，但增加了部分筛选选项。

“Report Timing”对话框

“Targets”选项卡

图 90：**“Report Timing”对话框：“Targets”选项卡**



“Report Timing”（时序报告）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）：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”（上升沿/下降沿）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)”（穿越点）：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”（穿越）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）：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”（报告）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。

“Options”选项卡

“Options”（选项）选项卡包含下列选项：

- Reports
- Path Limits
- Path Display

Reports

- “Path delay type”（路径延迟类型）：请参阅 “Report”部分。
- “Do not report unconstrained paths”（不报告未约束的路径）：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”（时序报告）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`

Path Limits

- “Number of paths per group”（各组的路径数量）：请参阅 Report Timing Summary。
- “Number of paths per endpoint”（各端点的路径数量）：请参阅 Report Timing Summary。
- “Limit paths to group”（将路径限制到组）：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`

Path Display

- “Display paths with slack greater than”（显示裕量大于指定值的路径）：基于路径的裕量值显示报告的路径。

  等效的 Tcl 选项：`-slack_greater_than`
- “Display paths with slack less than”（显示裕量小于指定值的路径）：请参阅 Report Timing Summary。
- “Number of significant digits”（有效位数）：请参阅 Report Timing Summary。
- “Sort paths by”（路径排序依据）：按组（默认）或者按裕量显示报告的路径。按组排序时，报告中会包含每个组和每种分析类型 (`-delay_type min/max/min_max`) 的 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，含最差的违例的组排在最高。

  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`

“Advanced”选项卡

“Advanced”（高级）选项卡与 Report Timing Summary 包含相同的选项。

“Timer Settings”选项卡

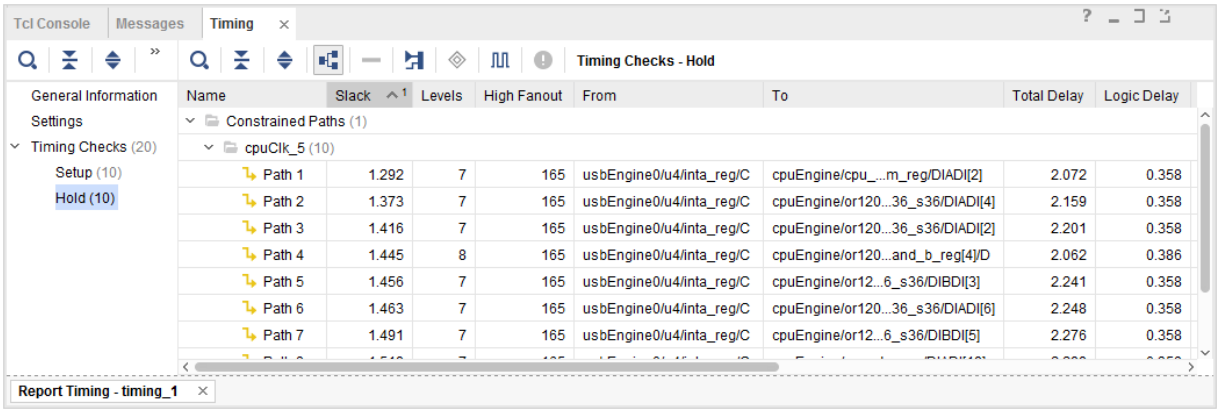
“Timer Settings”（定时器设置）选项卡与 Report Timing Summary 包含相同的选项。

复查时序路径详情

单击“OK”运行报告命令后，将打开一个新窗口。这样您即可复查其中内容。在其中可查看执行选定的每种类型 (`min/max/min_max`) 的分析之后所报告的 N 条最差路径。

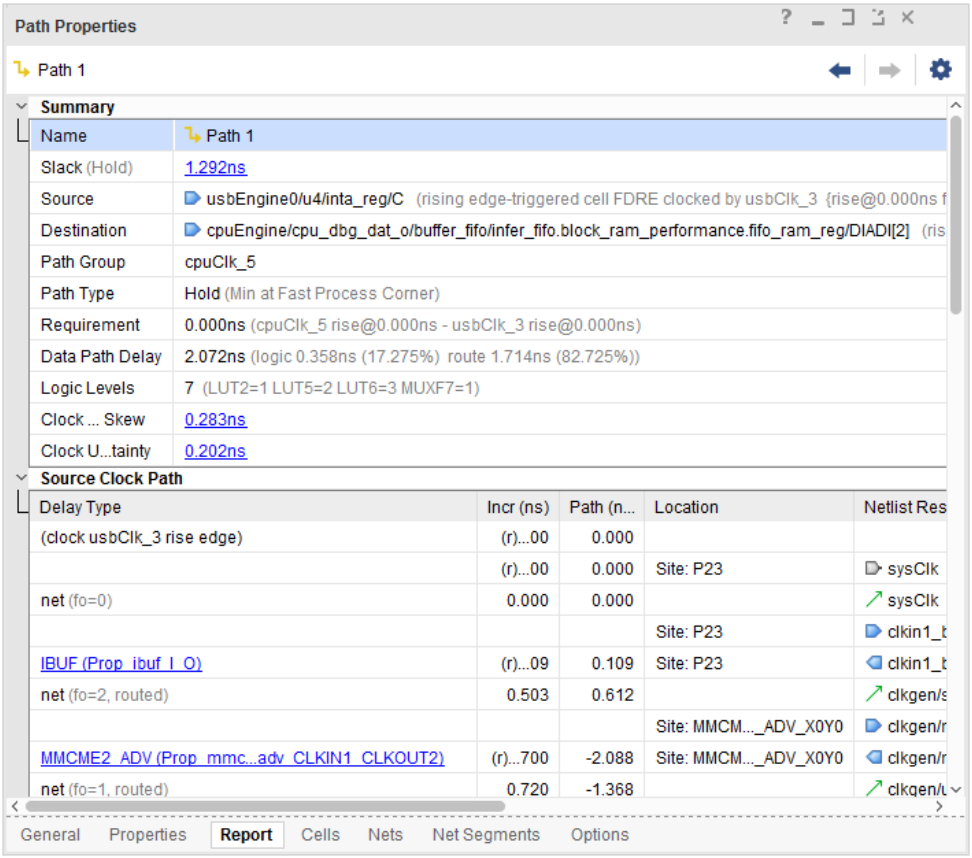
下图显示的“Report Timing”（时序报告）窗口中已选中最小和最大分析（SETUP 和 HOLD），且 N=4。

图 91：**时序报告路径列表**



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

图 92：**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 4 章：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：

- 查看时序路径板级原理图。
- 在选定路径的相同起点和端点上重新运行时序分析。
- 在“Device”（器件）和“Schematic”（板级原理图）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”（仅显示失败路径）按钮。

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“Timing Summary”（时序汇总）报告文件。

如果在存储器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 > 综合）
- “Flow Navigator” → “Implementation”（Flow Navigator > 实现）
- “Reports” → “Timing” → “Report Timing Summary”（报告 > 时序 > 时序汇总报告）

等效的 Tcl 命令为 `report_timing_summary`。

如需了解有关 `report_timing_summary` 选项的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的 report_timing_summary。

在综合后设计中，AMD 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“Route Status”（布线状态）报告。

从 Tcl 控制台或从 GUI 运行此时序汇总报告时，可使用 `-cells` 选项将其限定于 1 个或多个层级单元。限定报告作用域后，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。

从 Tcl 控制台运行时，此报告的第一部分提供了来自最新的 `report_methodology` 运行的方法论违例汇总信息。从 GUI 运行 `report_timing_summary` 时，这部分名为“Methodology Summary”（方法论汇总）。如果运行 `report_timing_summary` 之前尚未运行 `report_methodology`，则这部分为空。如果上一轮 `report_methodology` 运行已实现了任何设计更改，那么违例汇总信息可能并未提供最新信息。

“Report Timing Summary”对话框

在 Vivado IDE 中，“Report Timing Summary”（时序汇总报告）对话框包含以下选项卡：

- “Options”选项卡
- “Advanced”选项卡
- “Timer Settings”选项卡

位于“Report Timing Summary”对话框顶部的“Results name”（结果名称）字段用于指定在“Results”（结果）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”（器件）和“Schematic”（板级原理图）窗口以及设计源文件。

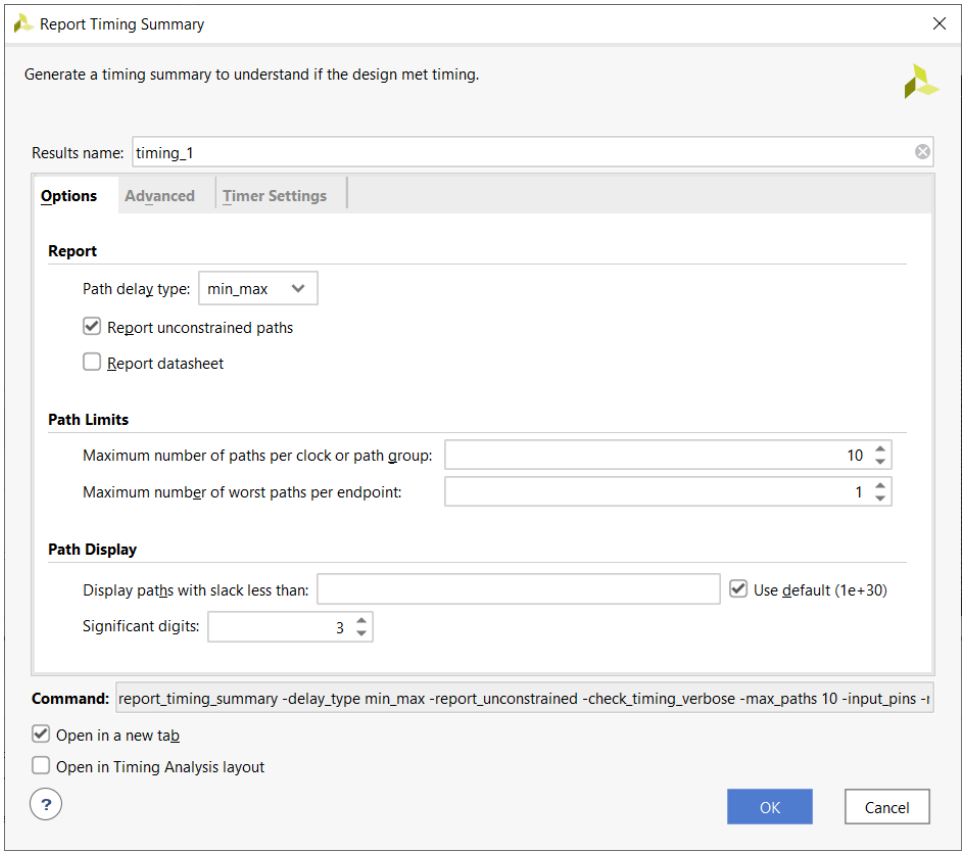
如果该字段留空，那么报告将返回 Tcl 控制台，且在“Results”窗口中不会打开图形化版本的报告。

等效的 Tcl 选项：`-name`

“Options”选项卡

“Report Timing Summary”（时序汇总报告）对话框中的“Options”（选项）选项卡如下图所示。

图 93：“Report Timing Summary”对话框：“Options”选项卡



“Report”部分

“Report Timing Summary”（时序汇总报告）对话框的“Options”（选项）选项卡的“Report”（报告）部分包含以下内容：

- “Path delay type”（路径延迟类型）：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`

- “Report unconstrained paths”（报告未约束路径）：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`

- “Report datasheet”（数据手册报告）：

  生成本章中的 Report Datasheet 中所定义的设计数据手册。

  等效的 Tcl 选项：`-datasheet`

“Path Limits”部分

“Report Timing Summary”（时序汇总报告）对话框中“Options”（选项）选项卡的“Path Limits”（路径限制）部分包括：

- “Maximum number of paths per clock or path group”（每个时钟组或路径组的最大数量）：控制每对时钟或每个路径组报告的路径的最大数量。

  等效的 Tcl 选项：`-max_paths`

- “Maximum number of worst paths per endpoint”（每个端点的最差路径的最大数量）：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`

“Path Display”部分

“Report Timing Summary”（时序汇总报告）对话框的“Options”（选项）选项卡的“Path Display”（路径显示）部分包括：

- “Display paths with slack less than”（显示裕量小于指定值的路径）：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`

- “Significant digits”（有效位数）：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”（时序汇总报告）对话框底部的全部 3 个选项卡通用的控件：

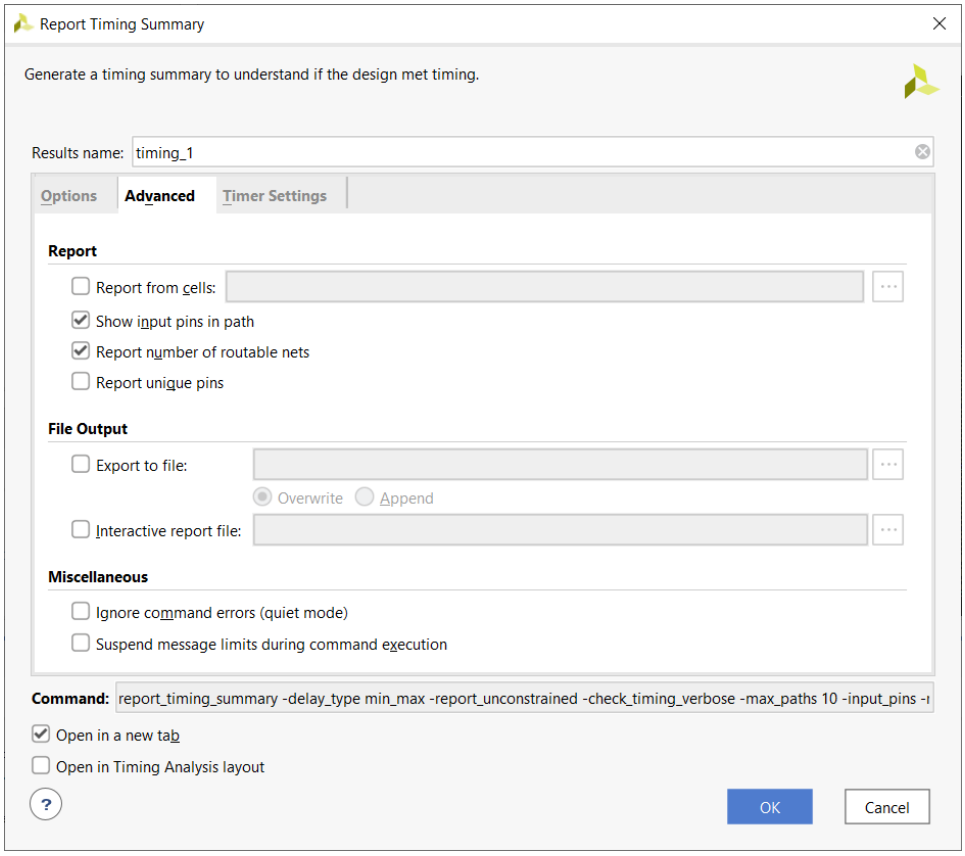
- “Command”（命令）：显示等效于“Report Timing Summary”对话框中指定的各种选项的 Tcl 命令行。
- “Open in a New Tab”（在新选项卡中打开）：在新选项卡中打开结果，或替换“Results”（结果）窗口中打开的最后一个选项卡。
- “Open in Timing Analysis layout”（在时序分析布局中打开）：将当前视图布局复位为“Timing Analysis”（时序分析）视图布局。

  如需了解有关视图布局的更多信息，请参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的“布局选择器”。

“Advanced”选项卡

“Report Timing Summary”（时序汇总报告）对话框中的“Advanced”（高级）选项卡如下图所示。

图 94：“Report Timing Summary”对话框：“Advanced”选项卡



“Report”部分

- “Report from cell”（基于单元的报告）：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`

- “Show input pins in path”（显示路径中的输入管脚）：显示用于路径的单元输入管脚。

  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。

- “Report unique Pins”（唯一管脚报告）：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`

“File Output”部分

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

  等效的 Tcl 选项：`-file`

- “Overwrite”（覆盖）或“Append”（追加）：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`

- “Interactive report file”（交互式报告文件）：将结果以 AMD RPX 格式写入指定的文件中。RPX 文件是一个包含所有报告信息的交互式报告，可在 Vivado Design Suite 中使用 `open_report` 命令将其重新加载到存储器中。

“Miscellaneous”部分

- “Ignore command errors”（忽略命令错误）：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`

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

  等效的 Tcl 选项：`-verbose`

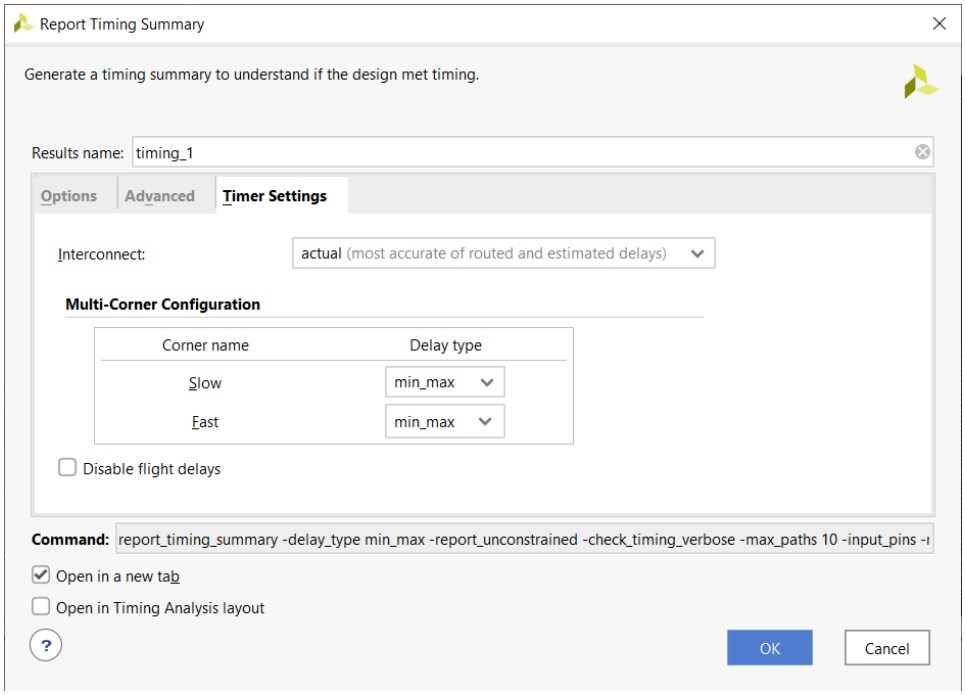
“Timer Settings”选项卡

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 会话内运行的其他时序相关命令，但综合和实现命令除外。

定时器设置不保存为工具首选项。每个新会话都会复原默认值。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。

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

图 95：**“Report Timing Summary”对话框：“Timer Settings”选项卡**



“Interconnect”**设置**

该选项用于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”。

- “Estimated”（估算）：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，基于驱动程序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程序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`。
- “Actual”（实际）：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`。
- “None”（无）：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`

“Multi-Corner Configuration”设置

指定要针对指定时序角分析的路径延迟类型。有效值包括 `none`、`max`、`min` 和 `min_max`。选择 `none` 为指定时序角禁用的时序分析。

建议：两个时序角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`

Disable Flight Delays

不将封装延迟添加到 I/O 延迟计算中。

等效的 Tcl 命令：`config_timing_analysis`

“Timing Summary Report”详情

“Timing Summary Report”（时序汇总报告）包含下列部分：

- “General Information”部分
- “Timer Settings”部分
- “Design Timing Summary”部分
- “Clock Summary”部分
- “Methodology Summary”部分
- “Check Timing”部分
- “Intra-Clock Paths”部分
- “Inter-Clock Paths”部分
- “Other Path Groups”部分
- “User-Ignored Paths”部分
- “Unconstrained Paths”部分

“Timing Summary Report”中包含的全面信息类似于 Vivado IDE 中的多项报告（时钟交互报告、脉冲宽度报告、时序报告和检查时序）所提供的信息，以及仅限 Tcl 中才可用的某些报告 (`report_clocks`)。但“Report Timing Summary”还包含自己特有的信息，例如，“Unconstrained Paths”（未约束路径）。

“General Information”部分

“Timing Summary”（时序汇总）报告的“General Information”（常规信息）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

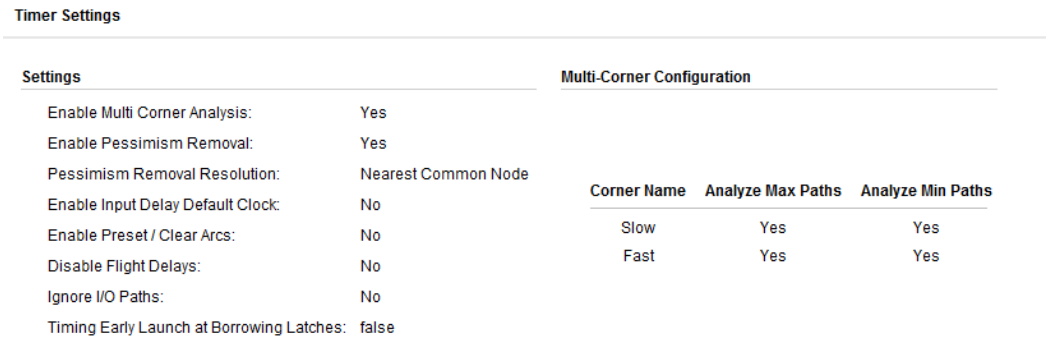
“Timer Settings”部分

“Timing Summary”（时序汇总）报告的“Timer Settings”（定时器设置）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：

- “Enable Multi-Corner Analysis”（启用多角分析）：针对每个时序角启用此分析，即“Multi-Corner Configuration”（多角配置）。
- “Enable Pessimism Removal”（启用消极因素移除）和“Pessimism Removal Resolution”（消极因素移除解决办法）：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。
- “Enable Input Delay Default Clock”（启用输入延迟默认时钟）：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。
- “Enable Preset / Clear Arcs”（启用预置/清除时序弧）：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。
- “Disable Flight Delays”（禁用飞行延迟）：为 I/O 延迟计算禁用封装延迟。

图 96：**“Timing Summary”报告：“Timer Settings”部分**



Timer Settings

Settings

Enable Multi Corner Analysis:	Yes
Enable Pessimism Removal:	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

Multi-Corner Configuration

Corner Name	Analyze Max Paths	Analyze Min Paths
Slow	Yes	Yes
Fast	Yes	Yes

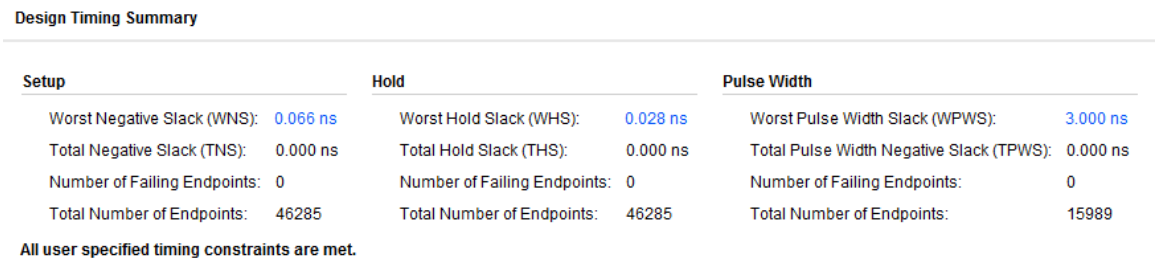
要获取有关默认定时器设置以及如何对其进行更改的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 的 config_timing_analysis 中提供的 `config_timing_analysis`。

“Design Timing Summary”部分

“Timing Summary”（时序汇总）报告的“Design Timing Summary”（设计时序汇总）部分（如下图所示）可提供设计时序汇总信息，并将所有其他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 97：**Design Timing Summary**



“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”（设计时序汇总）部分的“Setup”（建立时间）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)”（最差负时序裕量）：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)”（总体负时序裕量）：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。
  - 负值，前提是存在违例。
- “Number of Failing Endpoints”（故障端点数）：含违例 (WNS < 0 ns) 的端点总数。
- “Total Number of Endpoints”（端点总数）：已分析的端点总数。

“Hold”区域（最小延迟分析）

“Design Timing Summary”（设计时序汇总）部分的“Hold”（保持时间）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)”（最差保持时序裕量）：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)”（总体保持时序裕量）：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。
  - 负值，前提是存在违例。
- “Number of Failing Endpoints”（故障端点数）：含违例 (WHS < 0 ns) 的端点总数。
- “Total Number of Endpoints”（端点总数）：已分析的端点总数。

“Pulse Width”区域（管脚开关限制）

“Design Timing Summary”（设计时序汇总）部分的“Pulse Width”（脉冲宽度）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
- 最大偏差（在同一叶节点单元的两个时钟管脚之间，例如 PCIe® 或 GT [仅限 UltraScale 器件]）。

报告值为：

- “Worst Pulse Width Slack (WPWS)”（最差脉冲宽度时序裕量）：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)”（总体脉冲宽度时序裕量）：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”（故障端点数）：含违例 (WPWS < 0 ns) 的管脚总数。
- “Total Number of Endpoints”（端点总数）：已分析的端点总数。

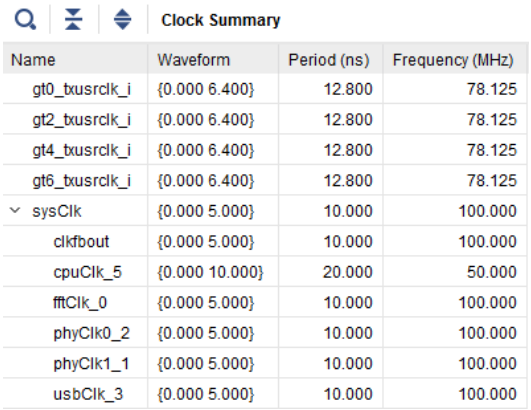
“Clock Summary”部分

“Timing Summary”（时序汇总）报告的“Clock Summary”（时钟汇总）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 98：**“Timing Summary”报告：“Clock Summary”部分**



Clock Summary

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
fftClk_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

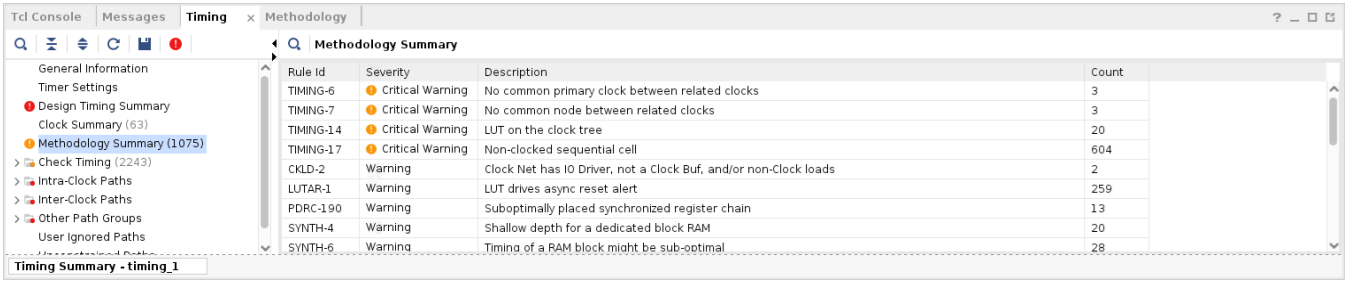
“Methodology Summary”部分

“Timing Summary”（时序汇总）报告的“Methodology Summary”（方法论汇总）部分包含方法论违例表格。在“Methodology Summary”右侧报告的是违例总数。类别名称前的图标表示需要复查的“Error”（错误）或“Critical Warning”（严重警告）。红色图标表示最高严重性“Error”，橙色图标表示“Critical Warning”。“Warning”（警告）或“Advisory”（建议）违例无图标。

注释： “Report Timing Summary”并不会运行“Report Methodology”。它仅报告最近在设计的存储器中或者通过重新加载检查点运行 `report_methodology` 后，通过跟踪发现的违例汇总。要以独立报告形式生成详细的方法论违例列表，请执行以下任一操作：

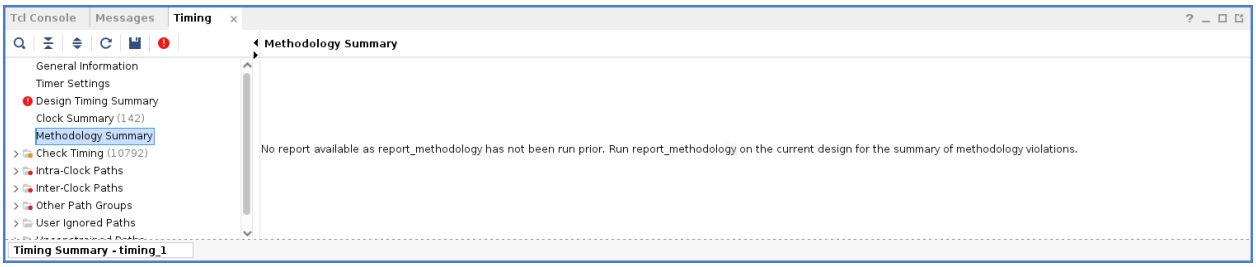
- 从菜单中选中“Reports”→“Report Methodology”（报告 > 方法论报告）。
- 运行 Tcl `report_methodology` 命令。

图 99：**Methodology Summary**



如果运行“Report Timing Summary”前未运行 `report_methodology`，则会显示一条消息用于说明无报告可用，如下图所示。

图 100：**未运行“Report Methodology”时的“Methodology Summary”**



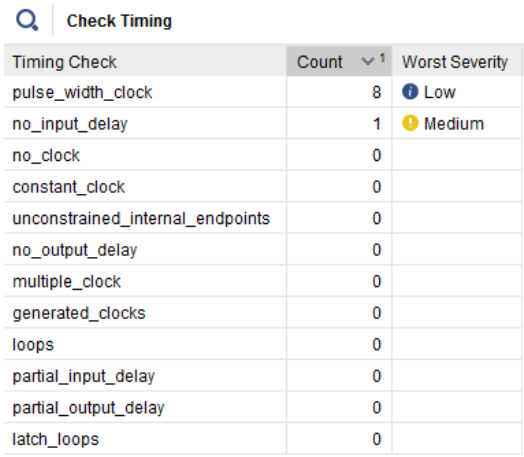
如需了解有关“Report Methodology”的更多信息，请参阅 方法论分析。

“Check Timing”部分

“Timing Summary”（时序汇总）报告的“Check Timing”（检查时序）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 101：**“Timing Summary”报告：“Check Timing”部分**



Check Timing

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”（报告 > 时序 > 检查时序）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。

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一起定时，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（多路复用器），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弧时，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。

“Intra-Clock Paths”部分

“Timing Summary Report”（时序汇总报告）的“Intra-Clock Paths”（时钟内部路径）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 102：**“Timing Summary”报告：“Intra-Clock Paths”部分**

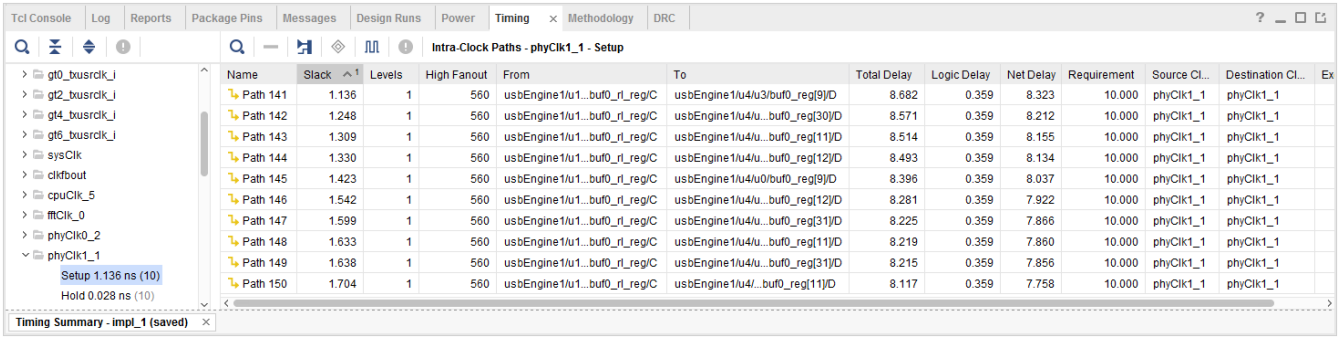
Intra-Clock Paths

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_txusrclk_i	rise - rise	9.887	0.000	0	218	rise - rise	0.135	0.000	0	218	6.000	0.000	0	154
gt2_txusrclk_i	rise - rise	9.668	0.000	0	218	rise - rise	0.095	0.000	0	218	6.000	0.000	0	154
gt4_txusrclk_i	rise - rise	8.853	0.000	0	218	rise - rise	0.123	0.000	0	218	6.000	0.000	0	154
gt6_txusrclk_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。

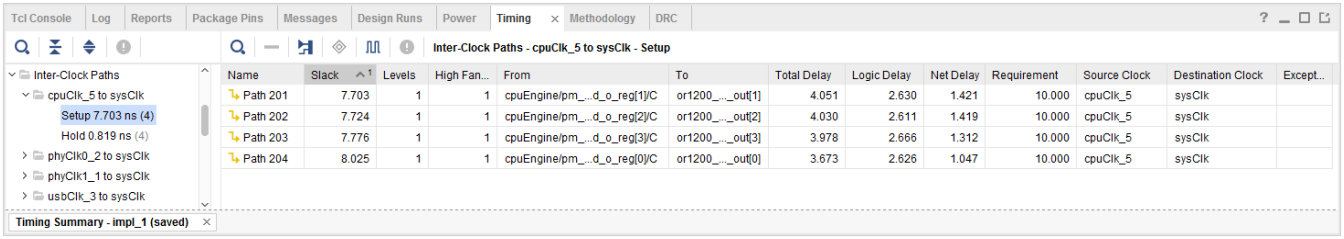
图 103：**“Timing Summary”报告：“Intra-Clock Paths”详细信息**



“Inter-Clock Paths”部分

与“Intra-Clock Paths”（时钟内部路径）部分相似，“Timing Summary”（时序汇总）报告的“Inter-Clock Paths”（时钟间路径）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 104：“Timing Summary”报告：“Inter-Clock Paths”详细信息

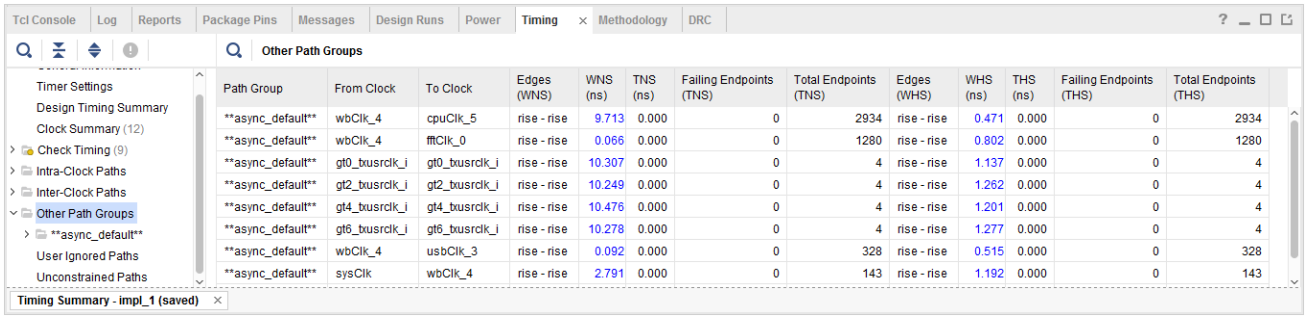


要查看详细信息，请单击左侧索引面板中“Inter-Clock Paths”下的名称。例如，您可查看有关每个时钟的裕量和违例汇总信息以及有关 SETUP、HOLD 或脉冲宽度检查的 N 条最差路径的详细信息。N 值定义方法如下：在命令行上使用 `-max_paths`，或者按时钟或路径组的最大路径数 (GUI) 来义。

“Other Path Groups”部分

“Timing Summary”（时序汇总）报告中的“Other Path Groups”（其他路径组）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。要访问该表，请在左侧面板中选择“Other Path Groups”。

图 105：“Timing Summary”报告：“Other Path Groups”部分

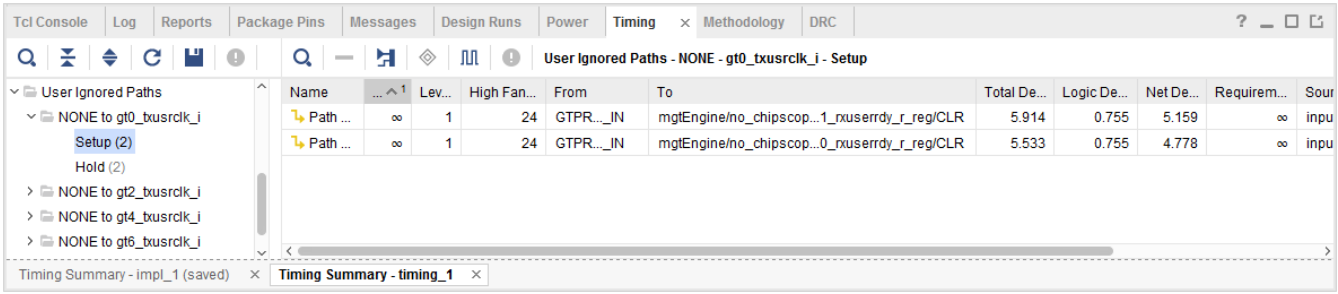


提示：**async_default** 是由 Vivado IDE 时序引擎自动创建的路径组。其中包括以异步时序检查结束的所有路径，例如，恢复和移除。这 2 项检查分别在“SETUP”类别和“HOLD”类别下报告，这 2 个类别分别对应于最大延迟分析和最小延迟分析。使用 `group_path` 创建的所有组也同样显示在此部分下。每个路径组中都可包含源时钟与目标时钟的任意组合。

“User-Ignored Paths”部分

“Timing Summary”（时序汇总）报告中的“User-Ignored Paths”（用户忽略的路径）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。

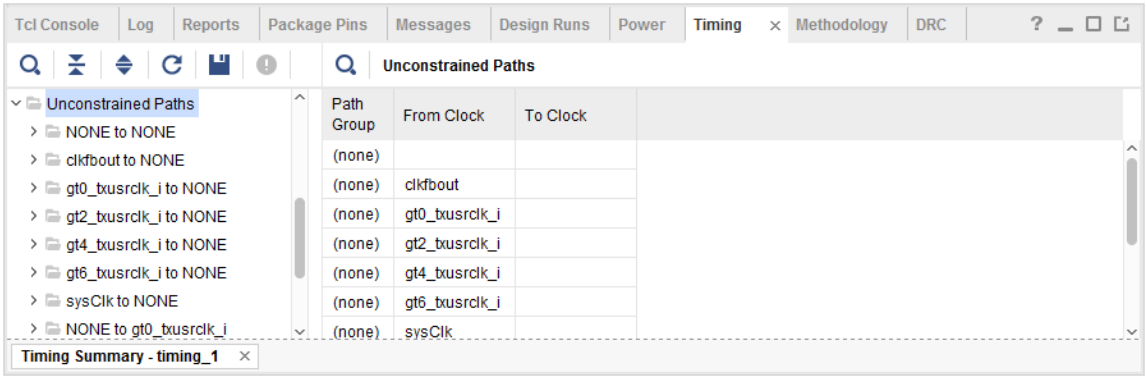
图 106：“Timing Summary”报告：“User-Ignored Paths”部分



“Unconstrained Paths”部分

“Timing Summary”（时序汇总）报告中的“Unconstrained Paths”（未约束的路径）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 107：“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”（建立时间）、“Hold”（保持时间）和“Pulse Width”（脉冲宽度）小节，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“Path Properties”（路径属性）窗口（“Report”（报告）视图）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 4 章：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：

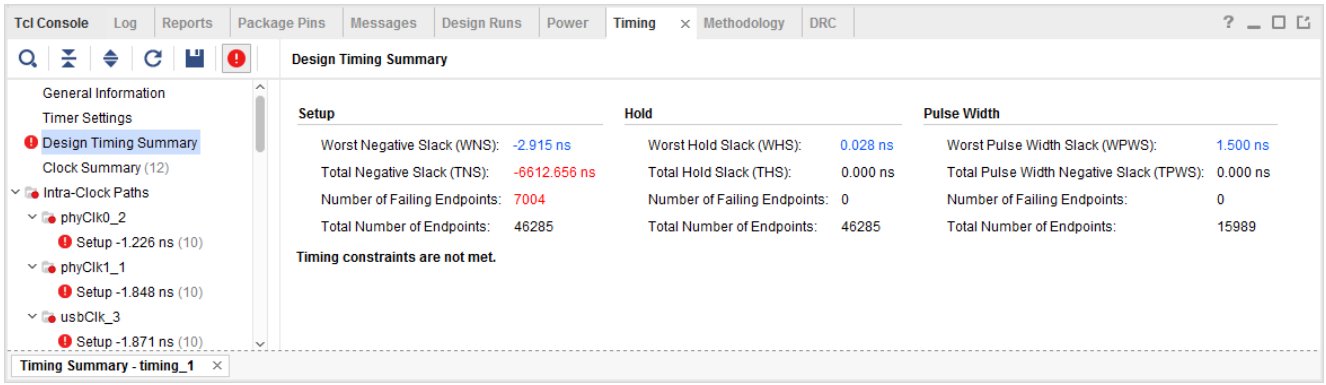
1. 在正确的面板内右键单击路径。
2. 从弹出菜单中选择下列选项之一：
   - “Schematic”（板级原理图）：打开路径的板级原理图。
   - “Report Timing on Source to Destination”（报告从源到目标的时序）：在同一路径上重新运行时序分析。
   - “Highlight”（高亮）：在“Device”（器件）窗口与“Schematic”窗口中高亮显示该路径。

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”（仅显示失败路径）按钮 。

下图显示了“Timing Summary”（时序汇总）窗口，其中仅显示失败路径。

图 108： **“Timing Summary”报告：违例路径筛选**



Report Bus Skew

“Bus Skew”（总线偏差）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，可使用 `-cells` 选项将总线偏差报告限定为一个或多个层级单元。限定报告作用域后，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”（总线偏差报告）GUI 中不可用。

运行“Report Bus Skew”

在命令行中通过 `report_bus_skew` Tcl 命令或者从 GUI 中的“Reports”→“Timing”→“Report Bus Skew”（报告 > 时序 > 总线偏差报告）下可获取总线偏差报告。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）。

“Bus Skew Report Summary”部分

“Bus Skew Report Summary”（总线偏差报告汇总）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”：报告后期引用的约束 ID。此信息可便于在报告中搜索该约束。

- “From”（来源）：针对 `set_bus_skew -from` 选项提供的模式。
- “To”（目标）：针对 `set_bus_skew -to` 选项提供的模式。
- “Corner”（时序角）：计算得到最差总线偏差的时序角（慢速角 (Slow) 或快速角 (Fast)）。
- “Requirement”（要求）：总线偏差目标值。
- “Actual”（实际值）：约束所覆盖的所有路径中计算所得的最差总线偏差。
- “Slack”（裕量）：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
```

注释：总线偏差违例 (WBSS) 表示在异步总线的多个不同的位之间存在的偏差超出预期。这可能导致在目标时钟域中捕获该总线上的错误数据。在此类情况下，总线的不同位可能会反映处于不同时钟周期的源时钟域所发送的状态。布线后如果留有一个或多个 WBSS 违例，请尝试执行不同的布线或布局指令。为了保证硬件稳定性，建议确保设计中未残留总线偏差违例。

“Bus Skew Report Per Constraint”部分

“Bus Skew Report Per Constraint”（总线偏差报告（按约束））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. “Per Constraint”（按约束）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：

- “From Clock”（源时钟）：起点时钟域。
- “To Clock”（目标时钟）：端点时钟域。
- “Endpoint Pin”（端点管脚）：报告的路径中所含的端点管脚。
- “Reference Pin”（参考管脚）：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。
- “Corner”（时序角）：用于计算此端点的最差偏差的快速/慢速 (Fast/Slow) 角。
- “Actual”（实际值）：计算所得偏差。偏差是 Endpoint Pin 的相对延迟减去 Reference Pin 的相对延迟减去相对 CRPR 的差值。
- “Slack”（裕量）：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。

图 109：**总线偏差报告（按约束）**

```
2. Bus Skew Report Per Constraint
---------------------------------

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

From Clock   To Clock   Endpoint Pin      Reference Pin     Corner  Actual(ns)  Slack(ns)
----------   ---------  ----------------  ----------------  ------  ----------  ---------
clk1         clk2       datac_reg[1]/D    datac_reg[2]/D    Slow         1.107     -0.107
clk1         clk2       datac_reg[2]/D    datac_reg[1]/D    Slow         1.107     -0.107
clk1         clk2       datac_reg[3]/D    datac_reg[2]/D    Slow         0.986      0.014
clk1         clk2       datac_reg[0]/D    datac_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。但由于总线偏差分析是在相同时钟沿上完成的，因此并不包含时钟不确定性。而是改为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。目标时钟的发送时间始终为 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。当时钟或数据路径跨多个 SLR 时，裕量计算期间会应用适当的 SLR 间补偿以防止出现不必要的消极因素。随后，在总线偏差头文件中会报告此类补偿。

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：

图 110：**详细路径示例**

```
Slack (MET) :             0.536ns  (requirement - actual skew)
  Endpoint Source:        datal_reg[1]/C
                            (rising edge-triggered cell FDRE clocked by clk1)
  Endpoint Destination:   datac_reg[1]/D
                            (rising edge-triggered cell FDRE clocked by clk2)
  Reference Source:       datal_reg[2]/C
                            (rising edge-triggered cell FDRE clocked by clk1)
  Reference Destination:  datac_reg[2]/D
                            (rising edge-triggered cell FDRE clocked by clk2)
  Path Type:              Bus Skew (Max at Slow Process Corner)
  Requirement:            1.000ns
  Endpoint Relative Delay:    1.068ns
  Reference Relative Delay:  -0.100ns
  Relative CRPR:              0.704ns
  Actual Bus Skew:            0.464ns  (Endpoint Relative Delay - Reference Relative Delay - Relative CRPR)

Endpoint path:
    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
                         (clock clk1 rise edge)       0.000     0.000 r
                         propagated clock network latency
                                                      5.389     5.389
    SLICE_X47Y53         FDRE                         0.000     5.389 r  datal_reg[1]/C
    SLICE_X47Y53         FDRE (Prop_fdre_C_Q)         0.269     5.658 r  datal_reg[1]/Q
    SLICE_X46Y53         net (fo=1, estimated)        0.432     6.090 r  datac_reg[1]/D
  -------------------------------------------------------------------    -------------------

                         (clock clk2 rise edge)       0.000     0.000 r
                         propagated clock network latency
                                                      5.040     5.040
                         clock pessimism              0.000     5.040
    SLICE_X46Y53         FDRE (Setup_fdre_C_D)       -0.018     5.022    datac_reg[1]
  -------------------------------------------------------------------
                         data arrival                           6.090
                         clock arrival                          5.022
  -------------------------------------------------------------------
                         relative delay                         1.068

Reference path:
    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
                         (clock clk1 rise edge)       0.000     0.000 r
                         propagated clock network latency
                                                      5.016     5.016
    SLICE_X47Y53         FDRE                         0.000     5.016 r  datal_reg[2]/C
    SLICE_X47Y53         FDRE (Prop_fdre_C_Q)         0.216     5.232 r  datal_reg[2]/Q
    SLICE_X46Y53         net (fo=1, estimated)        0.274     5.505 r  datac_reg[2]/D
  -------------------------------------------------------------------    -------------------

                         (clock clk2 rise edge)       0.000     0.000 r
                         propagated clock network latency
                                                      5.414     5.414
                         clock pessimism              0.000     5.414
    SLICE_X46Y53         FDRE (Hold_fdre_C_D)         0.191     5.605    datac_reg[2]
  -------------------------------------------------------------------
                         data arrival                           5.505
                         clock arrival                          5.605
  -------------------------------------------------------------------
                         relative delay                        -0.100
```

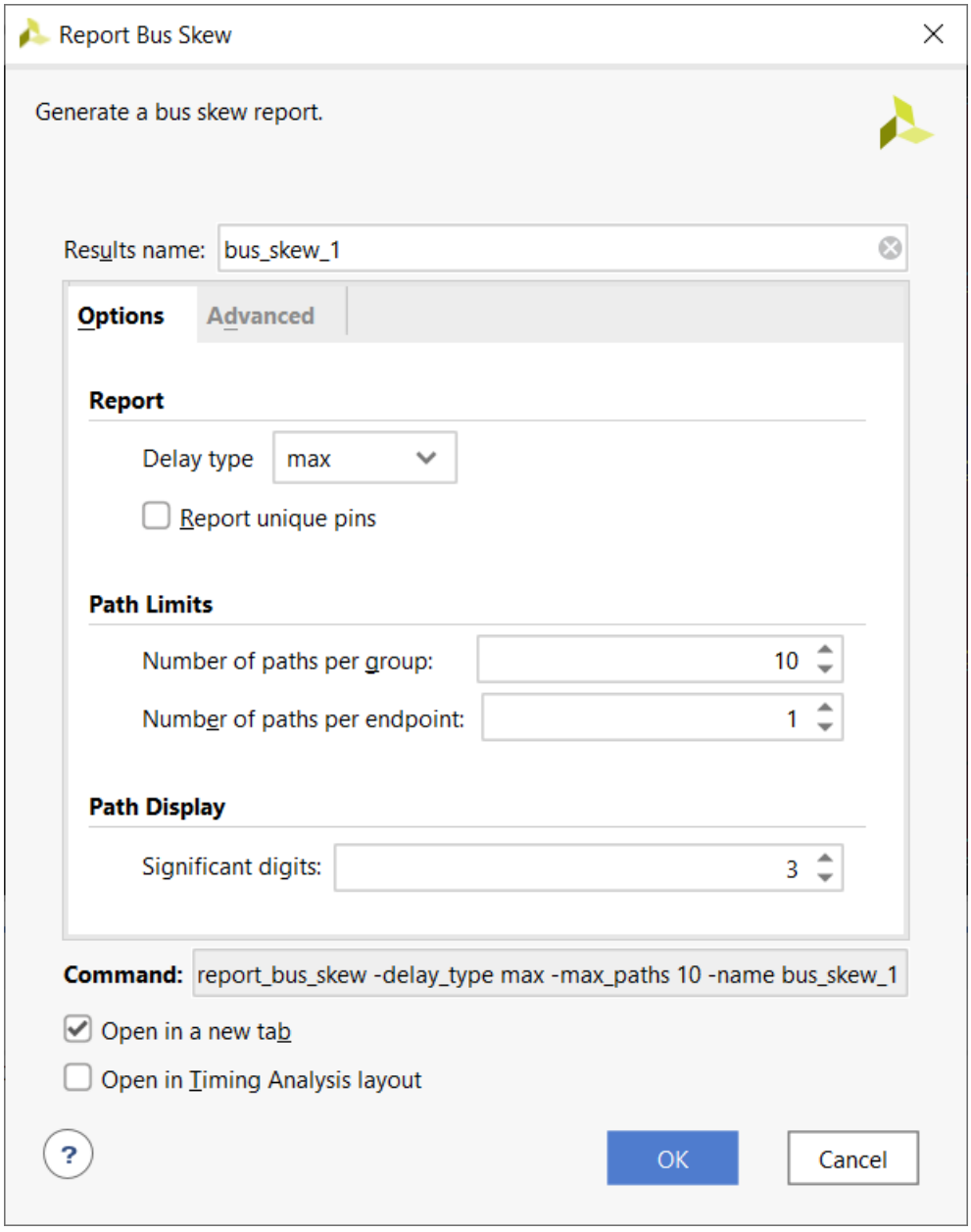
“Report Bus Skew”对话框

在 AMD Vivado™ IDE 中，要打开“Report Bus Skew”对话框，请选中“Reports” → “Timing” → “Report Bus Skew”（报告 > 时序 > 总线偏差报告）。

“Report Bus Skew”对话框：“Options”选项卡

“Report Bus Skew”（总线偏差报告）对话框中的“Options”（选项）选项卡如下图所示。

图 111：**“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
- Path Limits
- Path Display

报告

- “Delay type”（延迟类型）：请参阅 “Report”部分。
- “Report unique Pins”（唯一管脚报告）：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`。

Path Limits

- “Number of paths per group”（各组的路径数量）：请参阅 Report Timing Summary。
- “Number of paths per endpoint”（各端点的路径数量）：请参阅 Report Timing Summary。

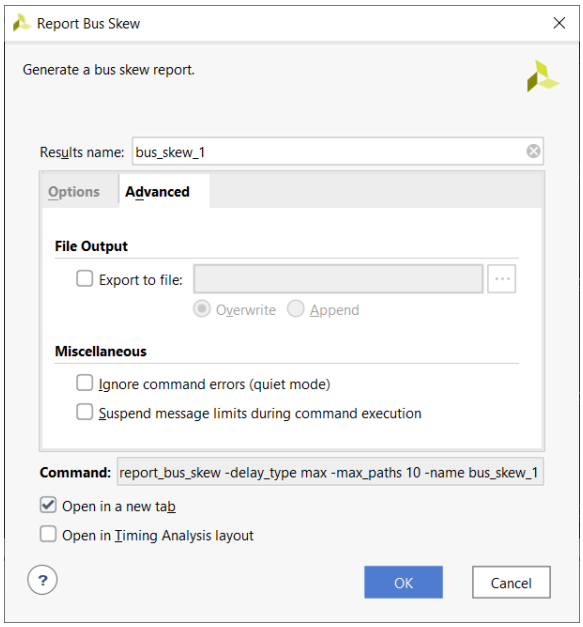
Path Display

- “Number of significant digits”（有效位数）：请参阅 Report Timing Summary。

“Report Bus Skew”对话框：“Advanced”选项卡

下图显示了“Report Bus Skew”（报告总线偏差）对话框中的“Advanced”（高级）选项卡。

图 112：**“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：

- File Output
- Miscellaneous

File Output

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

  等效的 Tcl 选项：`-file`。

- “Overwrite”（覆盖）或“Append”（追加）：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`。

Miscellaneous

- “Ignore command errors”（忽略命令错误）：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`。

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

  等效的 Tcl 选项：`-verbose`。

“Timing Summary Report”详情

“Bus Skew Report”（总线偏差报告）包含下列部分：

- “General Information”部分
- “Summary”部分
- “Set Bus Skew”部分

“General Information”部分

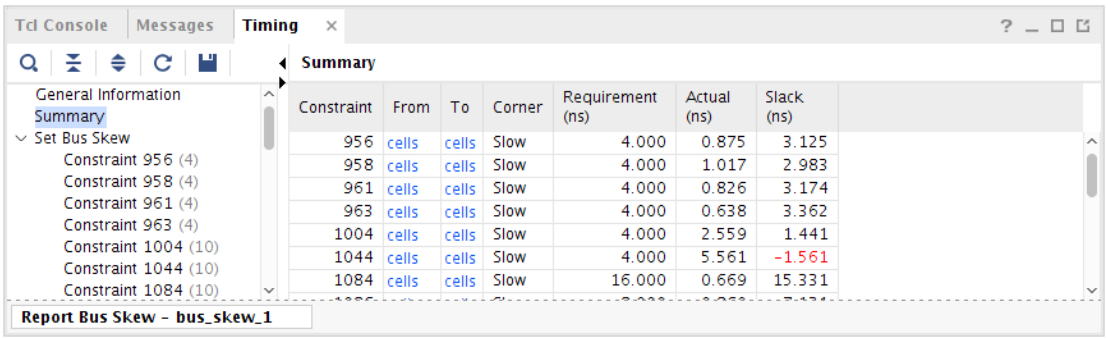
“Timing Summary”（时序汇总）报告的“General Information”（常规信息）部分可提供如下内容的相关信息：

- 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

“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。

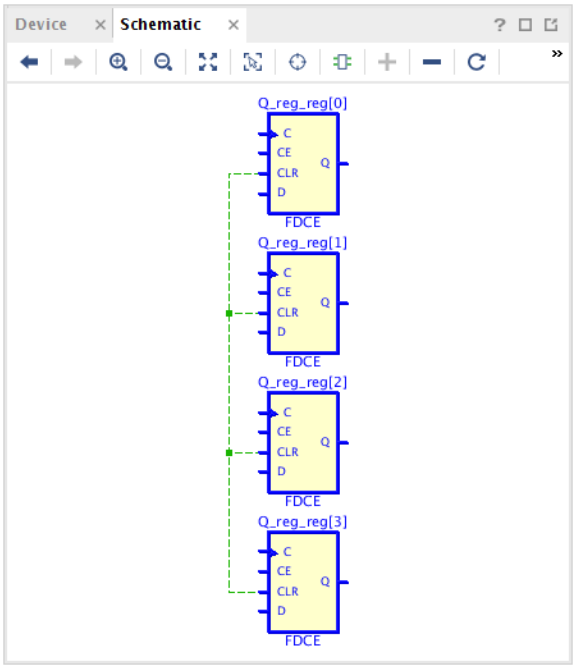
图 113：**Report Bus Skew：“Summary”部分**



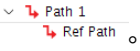
表格中的“Constraint”（约束）列表示时序约束位置编号，与 Timing Constraints Editor (TCE)（时序约束编辑器）中报告的位置编号相匹配。

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。对象可以是单元或管脚。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 键打开板级原理图。

图 114：**总线偏差端点的板级原理图示例**



“Set Bus Skew”部分

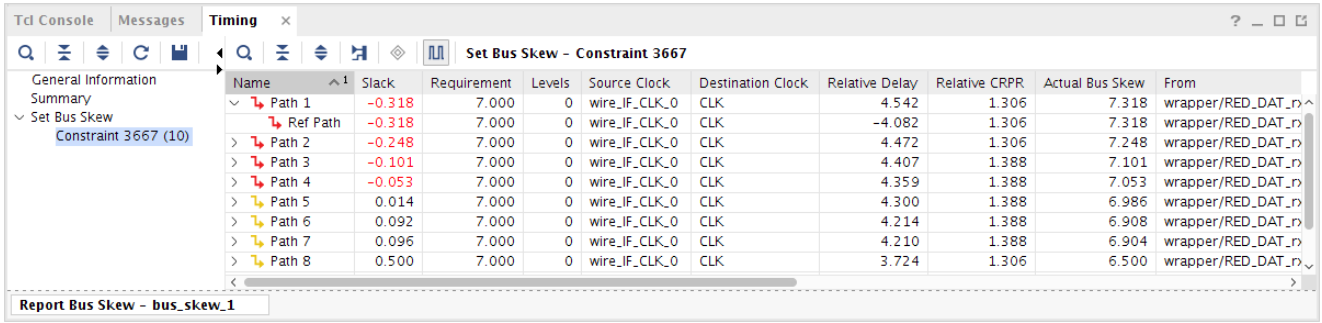
此部分提供了有关每项“Bus Skew”（总线偏差）约束的详细时序路径。针对每个时序路径端点都存在 1 条关联的可扩展参考路径 

。

总线偏差计算方法为：

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

图 115：**首个端点及其参考路径的详细路径示例**



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“Property”（属性）窗格中查看详细的时序路径报告。可通过单击板级原理图图标并按 F4 键来生成路径和/或参考路径的板级原理图（可一并选择端点路径与参考路径）。

Report Clock Networks

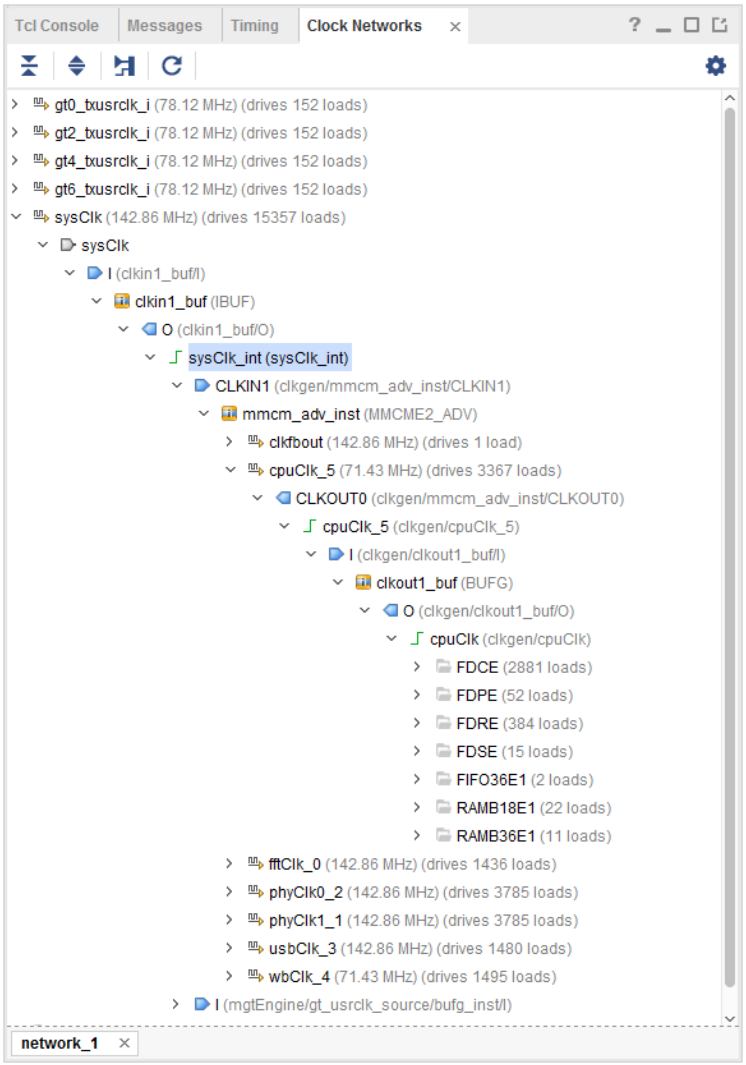
“Report Clock Network”（时钟网络报告）命令可从以下位置运行：

- AMD 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 116：**时钟网络**



时钟树：

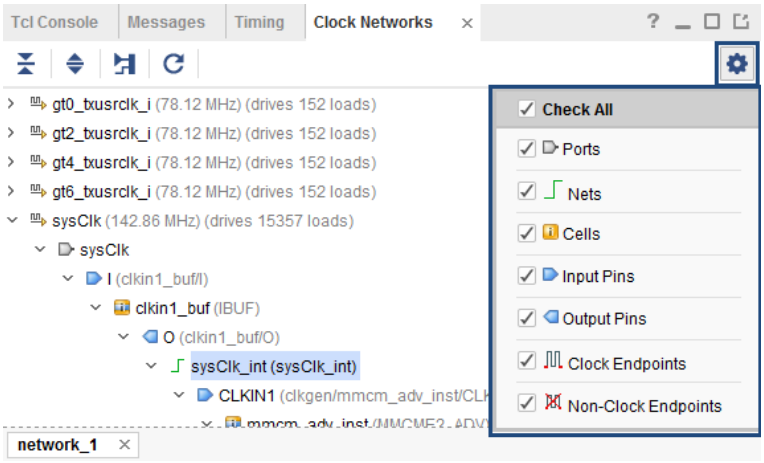
- 显示用户定义的时钟或工具自动生成的时钟。
- 报告从 I/O 端口到负载的时钟。

  注释：完整的时钟树仅在报告的 GUI 表单中详细展示。此报告的文本版本仅显示时钟根的名称。
- 可用于查找驱动其他 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“Ports”（端口）、“Nets”（信号线）、“Instances”（实例）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 117：**时钟网络筛选工具**



要查看时钟路径的板级原理图，请执行以下操作：

1. 选择时钟树中的对象。
2. 运行“Trace to Source”（追踪源文件）弹出命令。

Report Clock Interaction

要查看“Clock Interaction Report”（时钟交互报告），请选择以下任一项：

- “Reports” → “Timing” → “Report Clock Interaction”（报告 > 时序 > 时钟交互报告）
- “Flow Navigator ” → “Synthesis” → “Report Clock Interaction”（Flow Navigator > 综合 > 时钟交互报告）
- “Flow Navigator” → “Implementation” → “Report Clock Interaction”（Flow Navigator > 实现 > 时钟交互报告）

等效的 Tcl 命令：`report_clock_interaction -name clocks_1`

从 Tcl 控制台运行时，可使用 `-cells` 选项将交互报告限定为一个或多个层级单元。限定报告作用域后，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 AMD Vivado™ IDE 中，“Report Clock Interaction”（时钟交互报告）对话框包含以下选项卡：

- “Results Name”字段
- “Command”字段
- “Open in a New Tab”复选框
- “Options”选项卡
- “Timer Settings”选项卡

“Results Name”字段

“Report Clock Interaction”（时钟交互报告）对话框顶部的“Results name”（结果名称）字段用于指定打开的图形报告的名称。

等效的 Tcl 选项：`-name`

“Command”字段

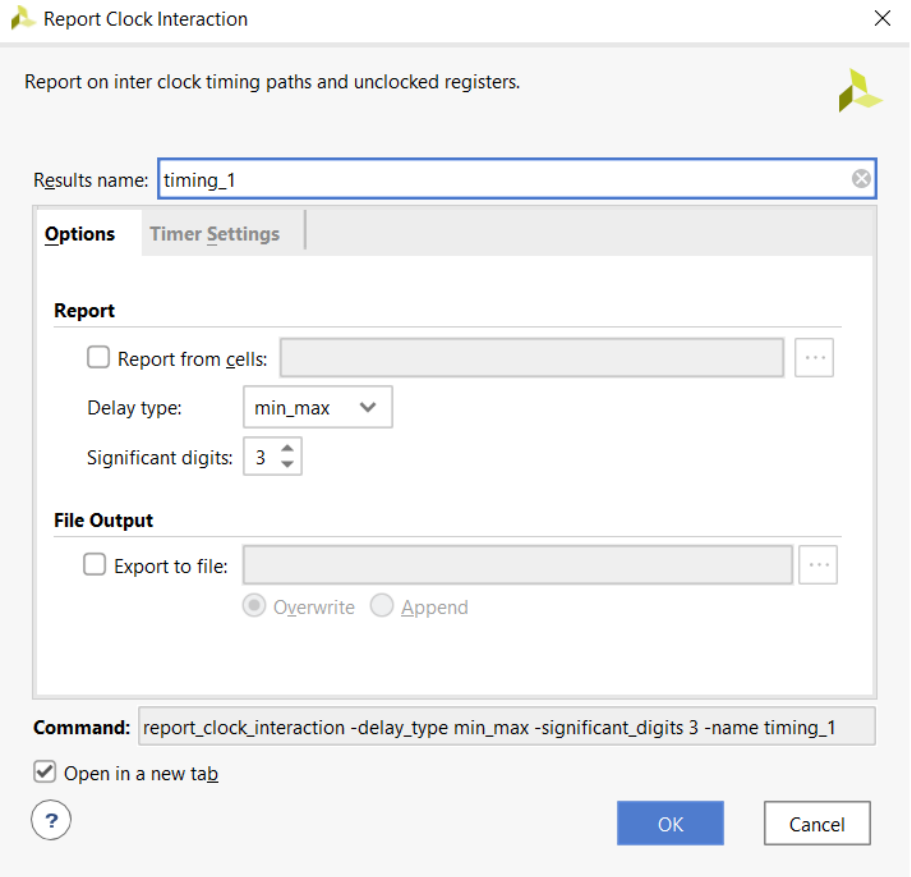
“Command”（命令）字段用于显示 Tcl 命令行，等效于“Report Clock Interaction”（时钟交互报告）对话框中指定的各种选项。

“Open in a New Tab”复选框

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

“Options”选项卡

图 118：“Report Clock Interaction”：“Options”选项卡



“Report Clock Interaction”（时钟交互报告）对话框的“Options”（选项）选项卡包含以下部分：

- “Report from Cells”字段
- “Delay Type”字段
- “Significant Digits”字段
- “File Output”部分

“Report from Cells”字段

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`

“Delay Type”字段

“Delay Type”（延迟类型）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`

“Significant Digits”字段

“Significant Digits”（有效位数）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`

“File Output”部分

“File Output”（文件输出）部分包括：

- “Write Results to File”：使用“Write Results to File”（将结果写入文件）字段将结果写入一个指定的文件。在 Vivado IDE 中，此报告显示在“Clock Interaction”（时钟交互）窗口中。

  等效的 Tcl 选项：`-file`

- “Overwrite/Append”：选择“Overwrite/Append”（覆盖/追加）选项按钮，以确定在将报告写入文件时：(1) 覆盖指定的文件，还是 (2) 向现有报告追加新信息。

  等效的 Tcl 选项：`-append`

“Timer Settings”选项卡

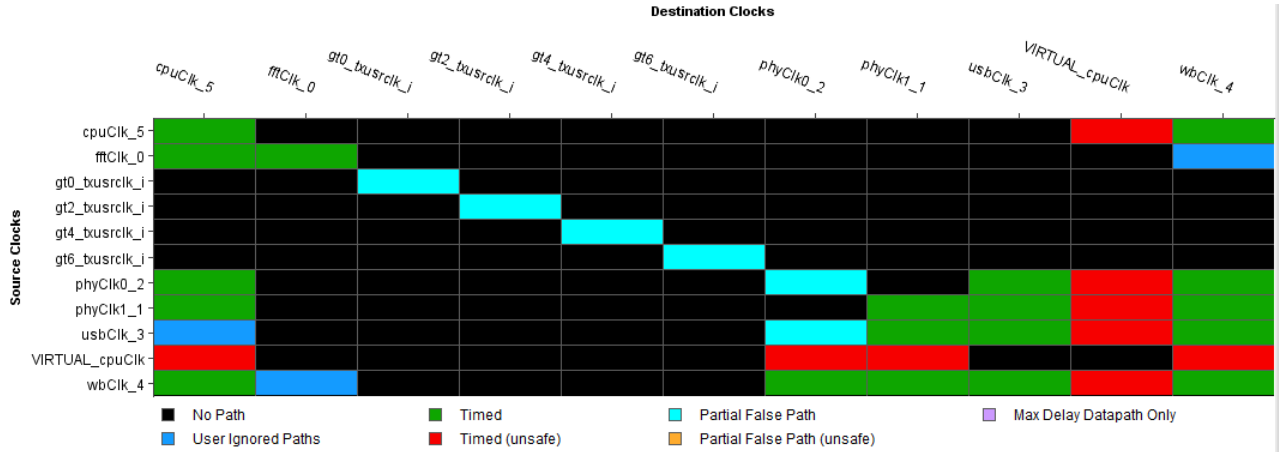
如需了解有关该选项卡的详细信息，请参阅 “Timer Settings”选项卡。

“Clock Interaction Report”详情

“Clock Interaction”（时钟交互）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 119：**Report Clock Interaction**



Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Path Req (WNS)	Edges (WHS)	WHS (ns)	THS (ns)	Clock Pair Classification	Inter-Clock Constraints
26	phyClk1_1	cpuClk_5	rise - rise	1.901	0.000	0	61	7.000	rise - rise	1.040	0.000	Clean	Timed
31	fftClk_0	cpuClk_5	rise - rise	1.063	0.000	0	61	7.000	rise - rise	1.488	0.000	Clean	Timed
34	VIRTUAL_c...	cpuClk_5	rise - rise	-1.372	-196.991	532	834		rise - rise	1.089	0.000	Virtual Clock	Timed (unsafe)
13	wbClk_4	fftClk_0				0	1635					Ignored	Asynchronous Groups
33	fftClk_0	fftClk_0	rise - rise	0.673	0.000	0	8320	7.000	rise - rise	0.046	0.000	Clean	Timed
1	gt0_txusrclk_i	gt0_txusrclk_i	rise - rise	9.923	0.000	0	222	12.800	rise - rise	0.135	0.000	Clean	Partial False Path
2	gt2_txusrclk_i	gt2_txusrclk_i	rise - rise	9.734	0.000	0	222	12.800	rise - rise	0.095	0.000	Clean	Partial False Path
3	gt4_txusrclk_i	gt4_txusrclk_i	rise - rise	8.856	0.000	0	222	12.800	rise - rise	0.123	0.000	Clean	Partial False Path

矩阵颜色编码

矩阵拼块按颜色编码。矩阵的颜色由“Tools” → “Settings” → “Colors” → “Clock Interaction Chart”（工具 > 设置 > 颜色 > 时钟交互图表）下定义的图形编辑器的背景颜色确定，或者通过选择“Clock Interactions”（时钟交互）选项卡上的齿轮图标来确定。如需了解更多信息，请参阅《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893) 中的“指定颜色”。如需隐藏图例，请单击矩阵左侧工具栏上的“?”按钮。

- “No Path”（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。
- “Timed”（已定时）- 绿色：源时钟与目标时钟存在同步关系，并且一起安全定时。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。
- “User Ignored Paths”（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。如果运行交互报告仅仅是为了进行保持时间分析 (`-delay_type min`) 并且 `set_max_delay -datapath_only` 约束已覆盖源时钟和目标时钟，那么由于衍生自仅最大延迟数据路径的隐式伪路径，“Clock Pair Classification”（时钟对分类）会报告为“ignored”（忽略）（GUI 类别为“User Ignored Paths”），“Inter-Clock Constraints”（时钟间约束）报告为“Auto Generated False Path”（自动生成的伪路径）。
- “Partial False Path”（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。
- “Timed (Unsafe)”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，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的“异步时钟组”。
- “Partial False Path (Unsafe)”（部分伪路径（不安全））- 橙色：此类别与“Timed (Unsafe)”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。

- “Max Delay Datapath Only”（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。

Clock Pair Classification

“Clock Pair Classification”（时钟对分类）列提供了有关两个时钟之间缺少公共基准时钟、缺少公共节点、缺少公共相位以及缺少公共周期的信息，以及是否存在虚拟时钟的信息。

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。

- “Ignored”（已忽略）：当“Clock Group”（时钟组）、“False Path”（伪路径）或“Max Delay Datapath Only”（仅最大延迟数据路径）完全覆盖时钟对时，将忽略分析。

  注释：当“Max Delay Datapath Only”覆盖时钟对时，在建立时间分析期间，“Inter-Clock Constraints”（时钟间约束）将报告为“Max Delay Datapath Only”，在保持时间分析 (`-delay_type min`) 期间，则报告为“Auto Generated False Path”（自动生成的伪路径）。

- “Virtual Clock”（虚拟时钟）：至少一个时钟是虚拟时钟，并且不适用公共基准时钟或公共节点检查。
- “No Common Clock”（无公共时钟）：两个时钟不具有公共基准时钟。
- “No Common Period”（无公共周期）：两个时钟的周期不可扩展。
- “Partial Common Node”（部分公共节点）：两个时钟显示为同步，但一小部分交汇路径不具有公共节点，并且无法安全定时。
- “No Common Node”（无公共节点）：两个时钟显示为同步，但交汇路径无公共节点。
- “No Common Phase”（无公共相位）：两个时钟不存在已知的相位关系。
- “Clean”（清除）：以上条件均不适用。

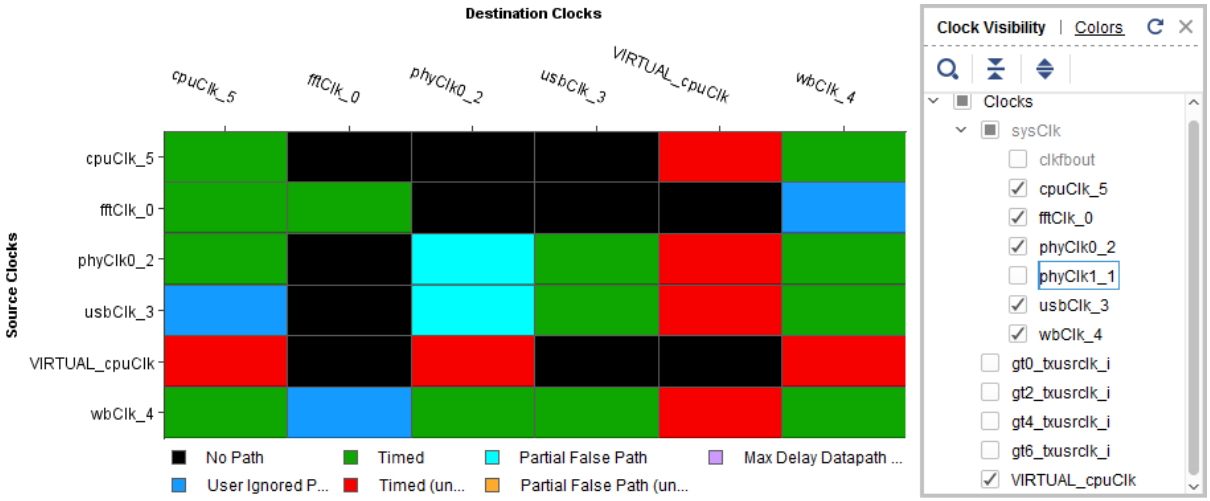
时钟筛选

如需筛选“Clock Interaction”（时钟交互）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“Clock Visibility”（时钟可视性）。
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“Hide Unused Clocks”（隐藏不使用的时钟）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 120：**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概览。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 “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

多次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”：当前显示的源/目标时钟对的数字 ID。
- “Source Clock”（源时钟）：作为路径起点的时钟域。
- “Destination Clock”（目标时钟）：作为路径终点的时钟域。
- “Edges (WNS)”（时钟沿 (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。
- “WNS (Worst Negative Slack)”（最差负时序裕量 (WNS)）：针对跨越指定时钟域的各条路径计算所得的最差时序裕量。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。
- “Total Negative Slack”（总体负时序裕量）：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。
- “Failing Endpoints”（故障端点）：未能满足时序的交汇路径中的端点数。违例总和对应于 TNS。
- “Total Endpoints (TNS)”（端点总数 (TNS)）：交汇路径中的端点总数。

- “Path Req (WNS)”（路径要求 (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。

- “Clock Pair Classification”（时钟对分类）：提供有关时钟对之间的公共节点和公共周期的信息。按优先级从高到低排序分为：“Ignored”（已忽略）、“Virtual Clock”（虚拟时钟）、“No Common Clock”（无公共时钟）、“No Common Period”（无公共周期）、“Partial Common Node”（部分公共节点）、“No Common Node”（无公共节点）、“No Common Phase”（无公共相位）和“Clean”（无错）。请参阅 Clock Pair Classification。

- “Inter-Clock Constraints”（时钟间约束）：显示源时钟与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：

- “Edges (WHS)”（时钟沿 (WHS)）：用于计算最差保持时序裕量的时钟沿。

- “WHS (Worst Hold Slack)”（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所得的最差时序裕量。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。

- “THS (Total negative Hold Slack):”（总体负保持时序裕量 (THS)：）：跨越最小延迟分析（保持/移除）的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。

- “Failing Endpoints (THS)”（故障断点 (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。

- “Total Endpoints (THS)”（端点总数 (THS)）：最小延迟分析（保持/移除）的交汇路径中的端点总数。

- “Path Req (WHS)”（路径要求 (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 4 章：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“Export to Spreadsheet”（导出到电子数据表）命令可将表格导出到 XLS 文件，以便在电子数据表中使用。

Report Pulse Width

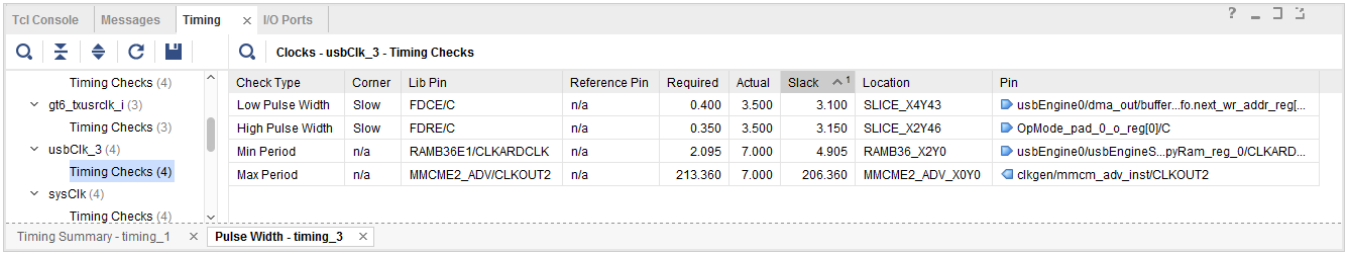
下图所示“Pulse Width Report”（脉冲宽度报告）用于检查设计是否满足每个实例时钟管脚的最小周期、最大周期、高脉冲时间和低脉冲时间要求。它还会检查已实现的设计中的相同实例（例如，PCIe® 时钟）的 2 个时钟管脚之间的最大偏差要求是否已得到满足。脉冲宽度裕量公式不包含抖动或时钟不确定性。

等效的 Tcl 命令：`report_pulse_width`

从 Tcl 控制台运行时，可使用 `-cells` 选项将脉冲宽度报告限定为一个或多个层级单元。限定报告作用域后，报告中仅包含单元内部的管脚。该选项在“Report Pulse Width”（脉冲宽度报告）GUI 中不可用。

注释：AMD Integrated Software Environment (ISE) Design Suite 实现将此检查称为“Component Switching Limits”（组件切换限制）。

图 121：**Report Pulse Width**



Report Datasheet

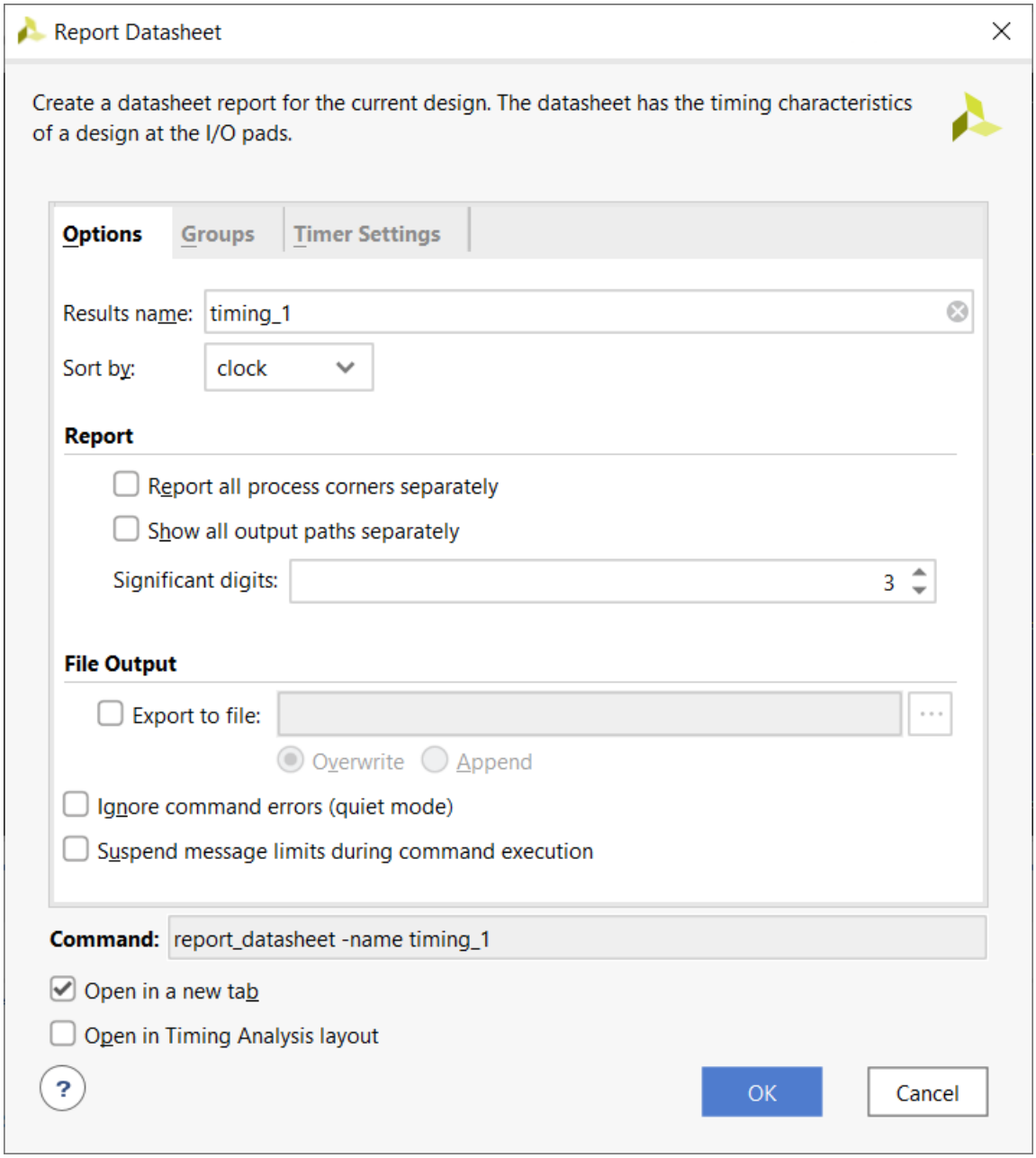
“Report Datasheet”（数据手册报告）命令用于报告系统级集成中使用的 FPGA 操作参数。

“Report Datasheet”对话框

在 AMD Vivado™ IDE 中，选择“Reports” → “Timing” → “Report Datasheet”（报告 > 时序 > 数据手册报告）即可打开“Report Datasheet”对话框。请参阅下图。

“Report Datasheet”对话框：“Options”选项卡

图 122：**“Report Datasheet”对话框：“Options”选项卡**



“Report Datasheet”（数据手册报告）对话框的“Options”（选项）选项卡包含以下内容：

- “Results name”（结果名称）：指定 Report Datasheet 命令返回的结果的名称。报告将在 Vivado IDE 的“Timing”（时序）窗口中以指定名称打开。

  等效的 Tcl 选项：`-name`

- “Sort by”（排序依据）：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`

- “Report all process corners separately”（单独报告所有工艺角）：报告当前设计中所有已定义的工艺角的数据。

  等效的 Tcl 选项：`-show_all_corners`

- “Significant digits”（有效位数）：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`

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

  等效的 Tcl 选项：`-file`

- “Overwrite”（覆盖）或“Append”（追加）：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`

- “Ignore command errors”（忽略命令错误）：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`

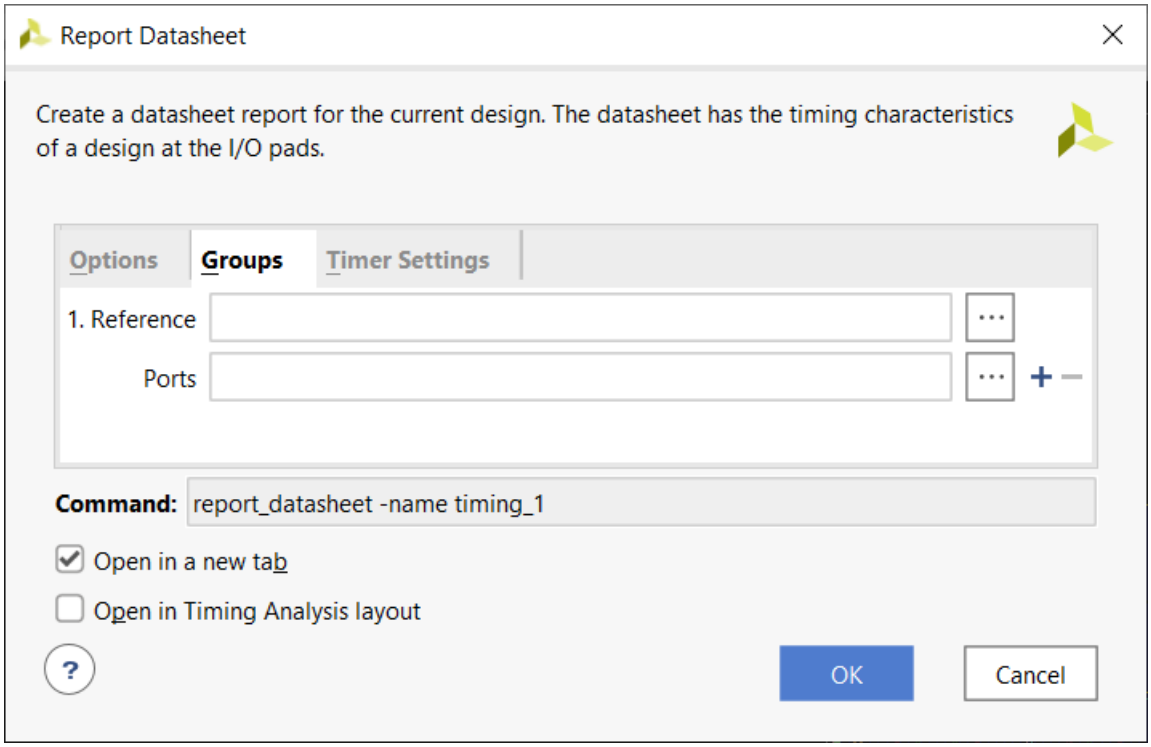
- “Suspend message limits”（暂挂消息限制）：临时覆盖所有消息限制。返回来自此命令的所有消息。

  等效的 Tcl 选项：`-verbose`

- “Command”（命令）：显示等效于“Report Datasheet”对话框中指定的各种选项的 Tcl 命令行。
- “Open in a new tab”（在新选项卡中打开）：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。
- “Open in Timing Analysis layout”（在时序分析布局中打开）：将当前视图布局复位为“Timing Analysis”（时序分析）视图布局。

“Report Datasheet”对话框：“Groups”选项卡

图 123：**“Report Datasheet”对话框：“Groups”选项卡**



“Report Datasheet”（数据手册报告）对话框的“Groups”（组）选项卡支持您通过指定参考端口和要报告的其他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：

- “Reference”（参考）：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`
- “Ports”（端口）：定义要报告的其他端口。
  - 请注意“Ports”字段右侧的“+”和“-”（加号与减号）按钮。
  - “+”（加号）按钮指定多个组，每个组都有自己的参考时钟端口，允许您定义一组新的端口组，包括一个新的参考端口。
  - “-”（减号）按钮可根据需要删除其他端口组。

“Report Datasheet”对话框：“Timer Settings”选项卡

如需了解有关该选项卡的详细信息，请参阅 “Timer Settings”选项卡。

“Datasheet Report”详情

通用信息

这部分提供了有关设计和 AMD 器件的详细信息以及报告时的工具环境信息。

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

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。

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。

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。

Setup/Hold for Input Buses

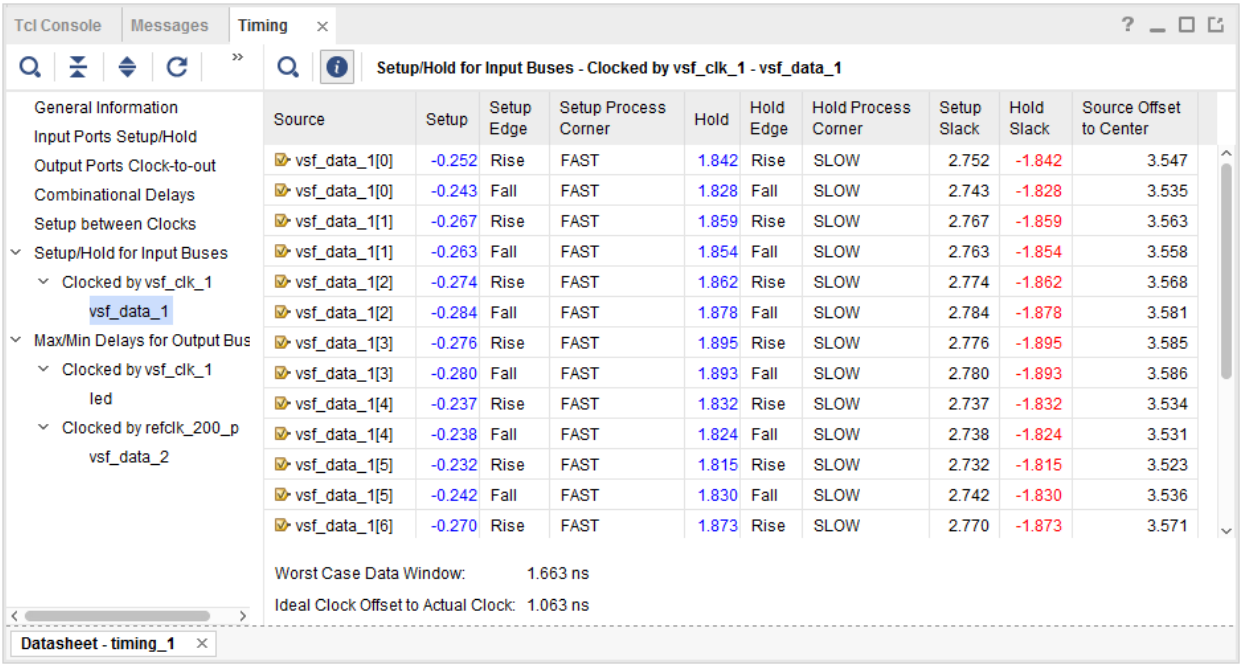
输入总线是自动推断所得，可显示其最差情况下的建立时间和保持时间要求。整个总线的最差情况数据窗口是最大建立时间和保持时间值的总和。如果输入端口受到约束，则将同时报告裕量。

针对已定义 IDELAY 的输入时钟，报告将显示最优抽头点。最优抽头点可用于配置 IDELAY 以实现平衡的建立时间和保持裕量。

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时间和保持时间定义。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 124：**输入总线的建立和保持延迟**



Source	Setup	Setup Edge	Setup Process Corner	Hold	Hold Edge	Hold Process Corner	Setup Slack	Hold Slack	Source Offset to Center
vsf_data_1[0]	-0.252	Rise	FAST	1.842	Rise	SLOW	2.752	-1.842	3.547
vsf_data_1[0]	-0.243	Fall	FAST	1.828	Fall	SLOW	2.743	-1.828	3.535
vsf_data_1[1]	-0.267	Rise	FAST	1.859	Rise	SLOW	2.767	-1.859	3.563
vsf_data_1[1]	-0.263	Fall	FAST	1.854	Fall	SLOW	2.763	-1.854	3.558
vsf_data_1[2]	-0.274	Rise	FAST	1.862	Rise	SLOW	2.774	-1.862	3.568
vsf_data_1[2]	-0.284	Fall	FAST	1.878	Fall	SLOW	2.784	-1.878	3.581
vsf_data_1[3]	-0.276	Rise	FAST	1.895	Rise	SLOW	2.776	-1.895	3.585
vsf_data_1[3]	-0.280	Fall	FAST	1.893	Fall	SLOW	2.780	-1.893	3.586
vsf_data_1[4]	-0.237	Rise	FAST	1.832	Rise	SLOW	2.737	-1.832	3.534
vsf_data_1[4]	-0.238	Fall	FAST	1.824	Fall	SLOW	2.738	-1.824	3.531
vsf_data_1[5]	-0.232	Rise	FAST	1.815	Rise	SLOW	2.732	-1.815	3.523
vsf_data_1[5]	-0.242	Fall	FAST	1.830	Fall	SLOW	2.742	-1.830	3.536
vsf_data_1[6]	-0.270	Rise	FAST	1.873	Rise	SLOW	2.770	-1.873	3.571

注释：以下 Tcl 命令可用于指定最优抽头点：

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

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，其他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。

Max/Min Delays for Groups

对于“Source Synchronous Output Interfaces”（源同步输出接口），前向时钟需要相关的输出偏差。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”（输出总线的最大/最小延迟），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 (`rldiii_ck_n[0]`) 相关的多个位（如，`rldiii_a[0-19]`、`rldiii_ba[0-3]`、`rldiii_ref_n`、`rldiii_we_n`）分组在一起，可使用以下命令：

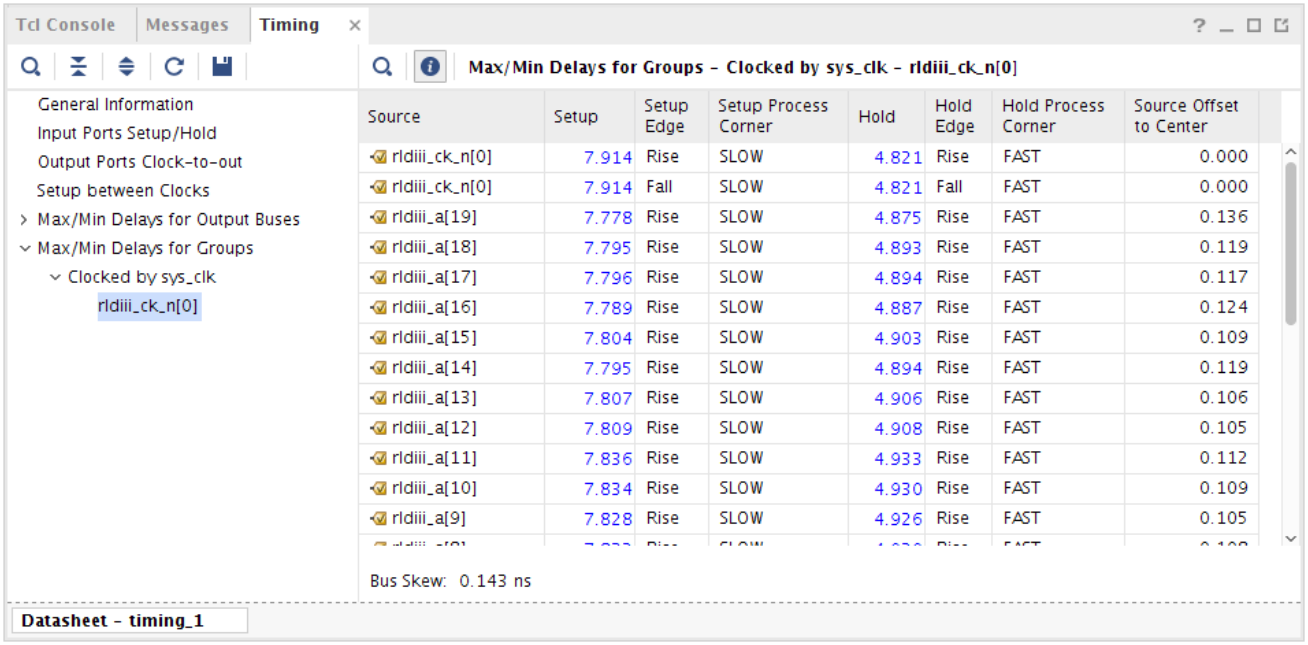
```
report_datasheet -group [get_ports {rldiii_ck_n[0] rldiii_a[*] rldiii_ba[*]
rldiii_ref_n rldiii_we_n}] -name timing_1
```

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时序角的最差情况数据。

下图显示了对应此示例的报告数据手册。

图 125：**数据手册的最大/最小延迟报告示例**



例外报告

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

- 在设计中已置位并且影响时序分析的所有时序例外
- 在设计中已置位但由于被其他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他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，请参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的 report_exceptions。如需了解有关时序例外优先级顺序的更多信息，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的“XDC 优先级”。

`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

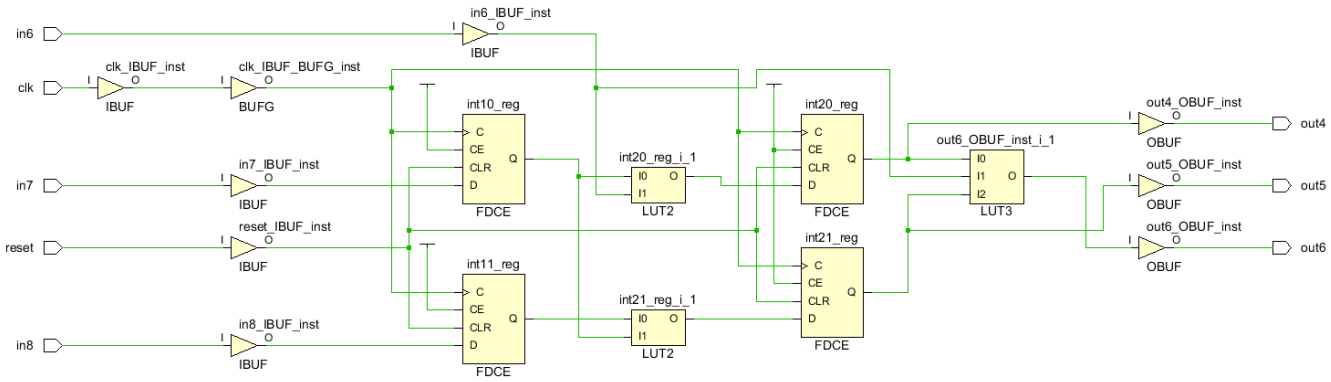
注释：即使“Clock Groups”（时钟组）严格意义上不属于时序例外，但仍在 `report_exceptions` 命令的覆盖范围内，因为时钟组可能覆盖其他时序例外。

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

本示例描述了对设计中的部分时序例外进行处理的方式，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 126：**对应时序例外的完全约束设计示例**



“Report Exception”（例外报告）命令的第一种工作模式为 `report_exceptions`。

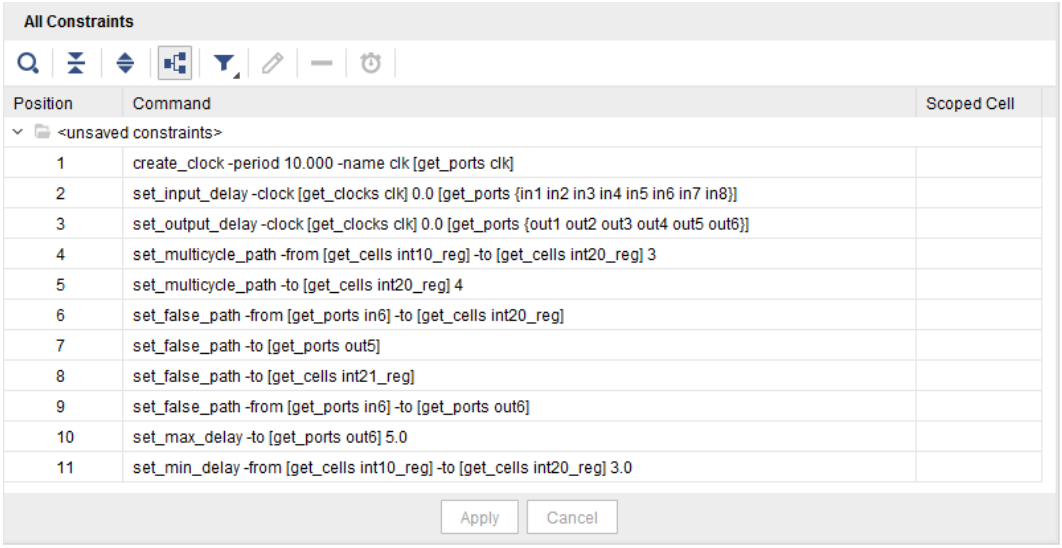
1. 选择“Window”→“Timing Constraints”（窗口 > 时序约束）。
2. 在“Timing Constraints”窗口中，为设计添加以下时序例外：

```
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 127：**显示时序约束变更的“Timing Constraints”窗口**



下图显示了实际例外报告 (`report_exceptions`)。

图 128：**Report Exception**

```
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
```

“Exceptions Report”（例外报告）包含以下信息：

- “Position”（位置）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 11：**“Setup”/“Hold”列命令约定**

简称	时序例外
`cycles=`	`set_multicycle_path`
`false`	`set_false_path`
`max=`	`set_max_delay`
`max_dpo=`	`set_max_delay -datapath_only`
`min=`	`set_min_delay`
`clock_group=`	`set_clock_group`

- 当约束的某一部分被另一个时序例外覆盖时，“Status”（状态）列将报告 1 条消息。下表显示了“Status”列的命名约定：

  表 12：**“Status”列命名约定**

简称	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”列中报告该时钟组。

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）的某些部分被多周期约束位置 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]`) 覆盖。
- 时序约束位置 10 (`set_max_delay 5 -to [get_ports out6]`) 的某些部分被伪路径位置 9 (`set_false_path -from [get_ports in6] -to [get_ports out6]`) 覆盖。

报告被覆盖的作用域内的时序例外

“Report Exception”命令的第 2 种操作模式是报告被覆盖的限定作用域内的时序例外。此报告是使用 `-scope_override` 命令行选项生成的。

```
report_exceptions -scope_override
```

注释：此模式仅可用于来自 Tcl 控制台的文本报告。

命令行选项 `-scope_override` 用于报告被顶层约束部分或全部覆盖的限定作用域内的时序例外。例如，它可标记被顶层用户约束覆盖的 IP 约束。但该选项不会报告被其他限定作用域的约束（来自相同或不同作用域）所覆盖的限定作用域的约束。

对于每个被覆盖的时序例外，此报告都包括以下信息：

- 约束位置编号
- `-from`/`-through`/`-to` 命令行选项的模式
- 覆盖建立时间和/或保持时间的约束的类型
- 约束的作用域
- 部分或全部覆盖此约束的顶层约束的位置编号列表

下图显示了“Exceptions Report”（例外报告）。

图 129：**Exceptions Report**

Exceptions Report

Position	From	Through	To	Setup	Hold	Scope	Status
53	*	*	[get_pins -hier *cdc_to*/D]	false	false	pfm_top_i/static_region/brd_mgmt_scheduler/board_management/psreset_board_control/U0	Partially overridden path by CG 62
59	*	*	[get_pins -hier *cdc_to*/D]	false	false	pfm_top_i/static_region/brd_mgmt_scheduler/embedded_schduler/psreset_scheduler/U0	Partially overridden path by CG 62

报告已忽略的时序例外

“Report Exception”（例外报告）命令的第二种操作模式如下所示：

```
report_exceptions -ignored
```

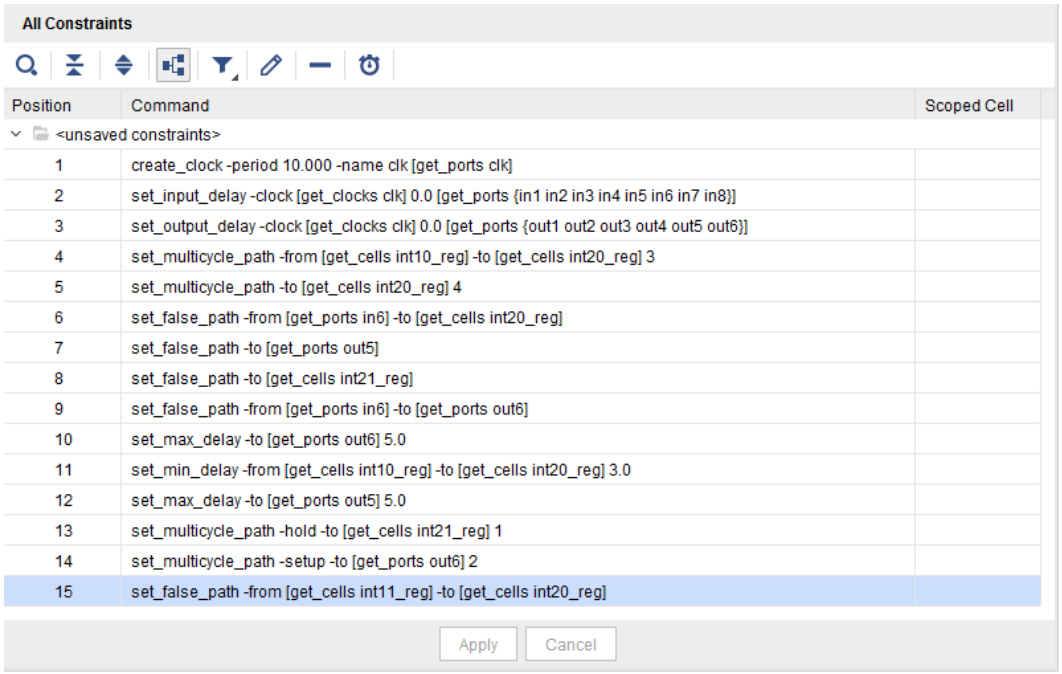
为便于演示，请在前述时序例外的基础上添加以下时序例外：

```
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]
```

所有这些例外均已被先前章节中的时序例外（报告影响时序分析的时序例外）所覆盖，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”（时序约束）窗口如下图所示。

图 130：**“Timing Constraints”窗口**



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

图 131：**Exceptions Report**

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： “Status”（状态）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 工具可生成已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 132：**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 `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 133：**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”（例外报告）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“Max Delay”（最大延迟）约束或“Min Delay”（最小延迟）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“False Path”（伪路径）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一个或多个单元从 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 134：**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”（例外报告）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) 存储器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 False Path（伪路径）约束上的 `report_exceptions -write_valid_exceptions`。

图 135：**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”（例外报告）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，以减少耗用的存储器和运行时间。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，不会滤除无效的起点和端点。

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 136：**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

“Report Exceptions”对话框

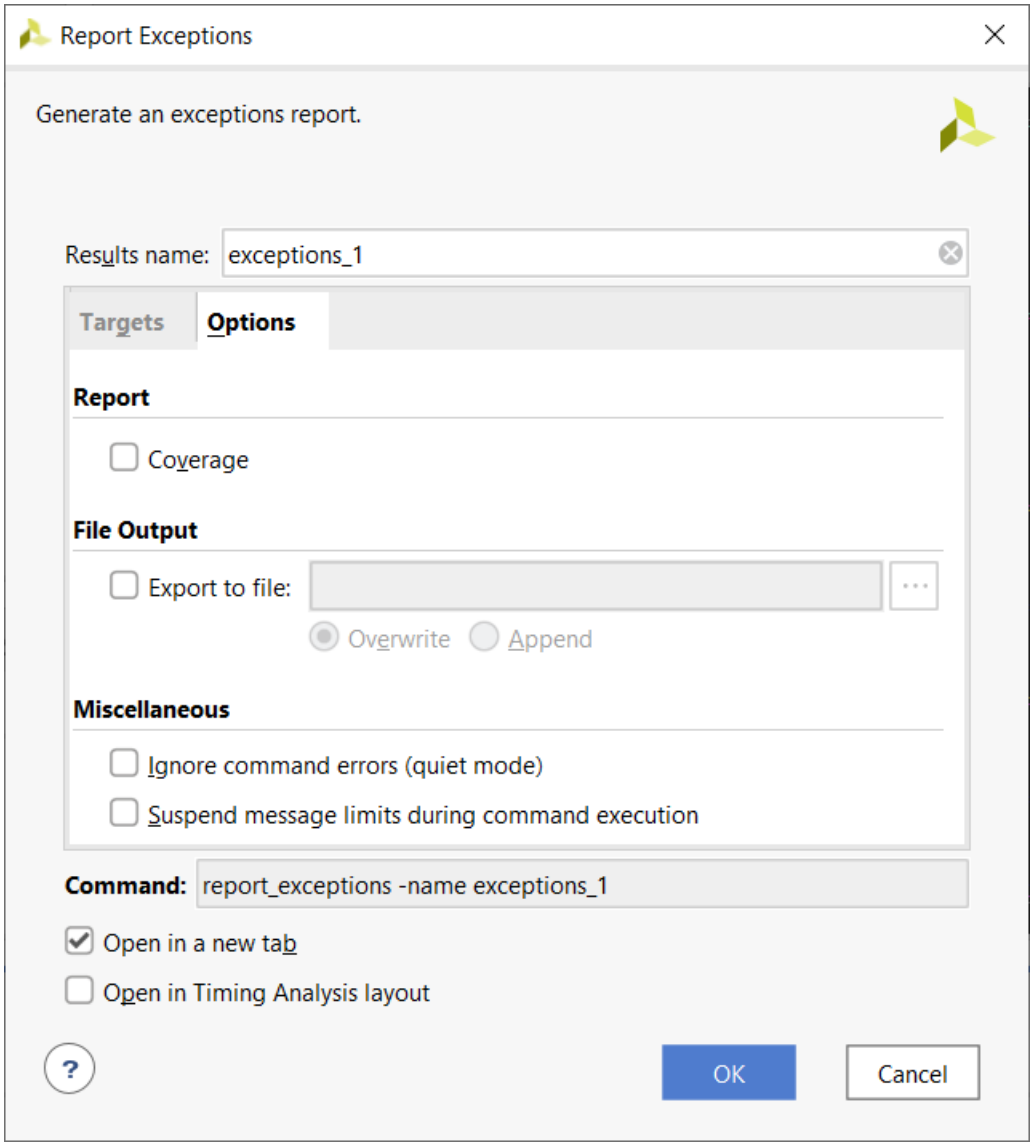
在 AMD Vivado™ IDE 中，选择“Reports” → “Timing” → “Report Exceptions”（报告 > 时序 > 例外报告）即可打开“Report Exceptions”对话框。

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”（例外报告）的时间可能比通过 Tcl 控制台运行 `report_exceptions` 的时间更长。

“Report Exceptions”对话框：“Options”选项卡

“Report Exceptions”（例外报告）对话框中的“Options”（选项）选项卡如下图所示。

图 137：**“Report Exceptions”对话框：“Options”选项卡**



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

“Report”部分

“Coverage”（覆盖范围）：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`

“File Output”部分

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

  等效的 Tcl 选项：`-file`

- “Overwrite”（覆盖）或“Append”（追加）：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`

“Miscellaneous”部分

- “Ignore command errors”（忽略命令错误）：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`

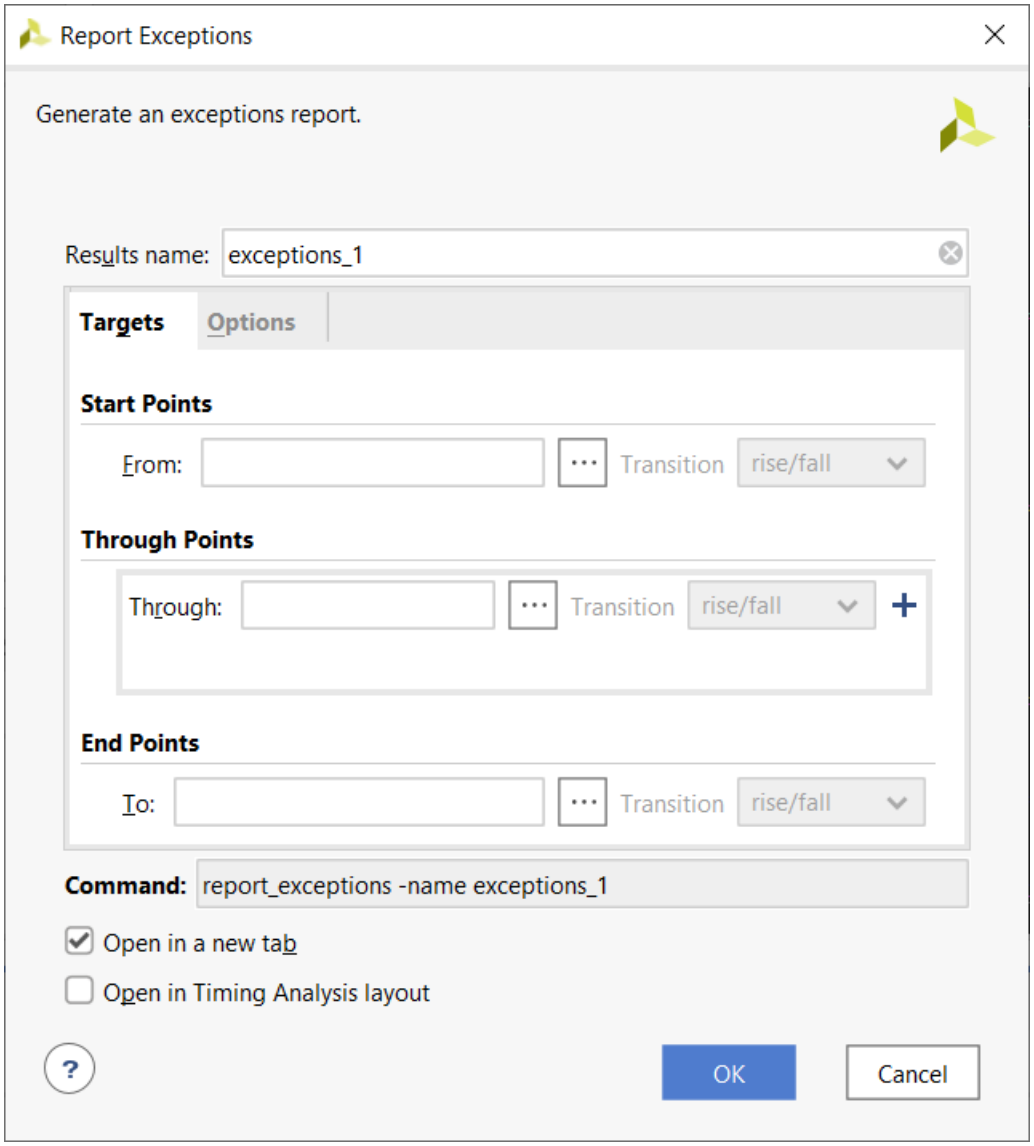
- “Suspend message limits during command execution”（命令执行期间暂挂消息限制）：临时覆盖所有消息限制并返回所有消息。

  等效的 Tcl 选项：`-verbose`

“Report Exceptions”：“Targets”选项卡

“Report Exceptions”（例外报告）对话框中的“Targets”（目标）选项卡如下图所示。

图 138：**“Report Exceptions”：“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：

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

“Exceptions Report”详情

“Exceptions Report”（例外报告）包含以下几个部分：

“General Information”部分

“Exceptions Report”（例外报告）的“General Information”（常规信息）部分提供了有关以下内容的信息：

- 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

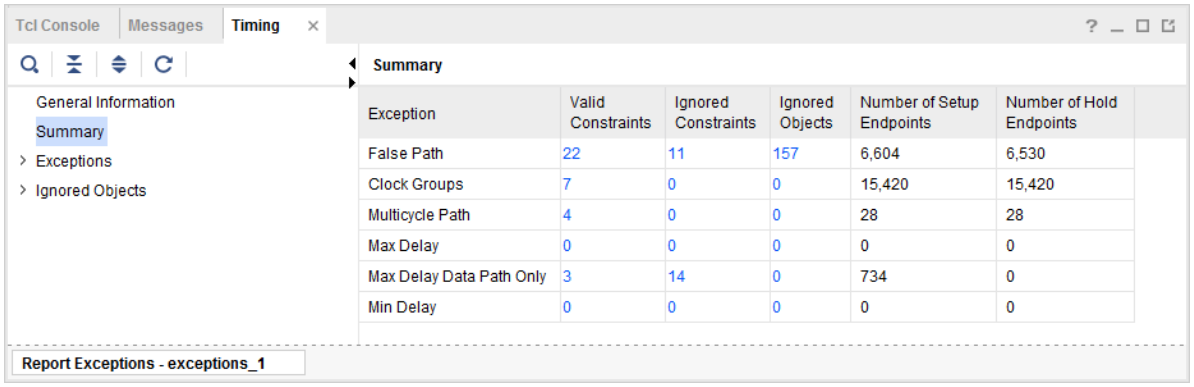
“Summary”部分

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“Exceptions”（例外）部分或“Ignored Objects”（已忽略的对象）部分的超链接。“Valid Constraints”（有效约束）和“Ignored Constraints”（已忽略的约束）链接至同一个“Exceptions”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他约束所覆盖，则忽略例外。

图 139：**Report Exceptions：“Summary”部分**

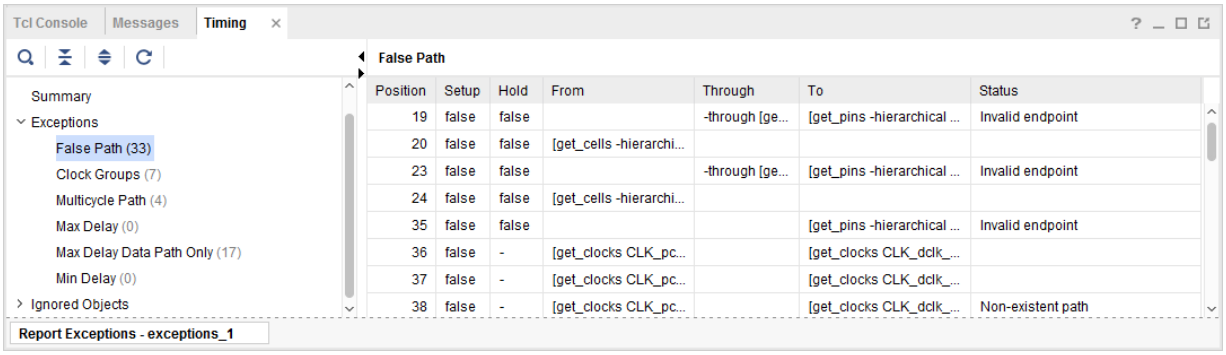


“Exceptions”部分

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“Summary”（汇总）表的超链接。详情表的格式取决于在 GUI 中是否已选中“Coverage”（覆盖率）选项。

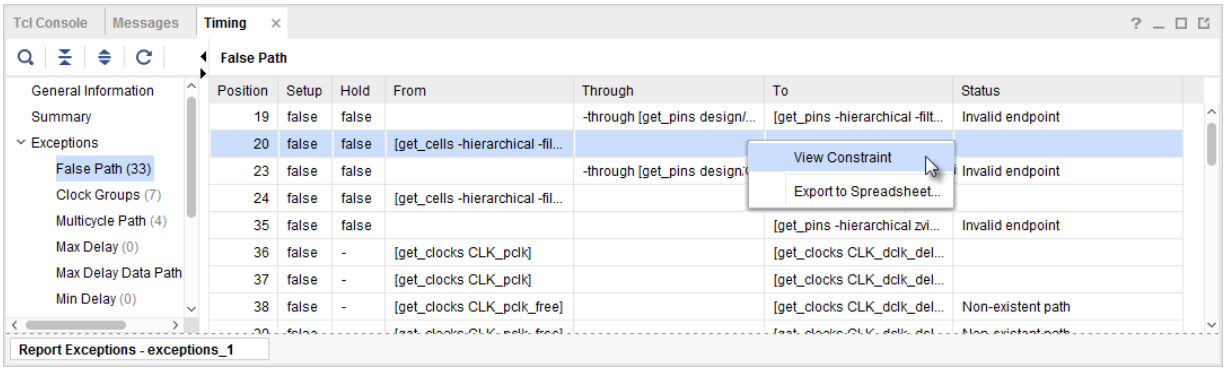
以下是未选中“Coverage”选项的详情表示例。

图 140：**“Report Exceptions”：未选中“Coverage”的详情表**



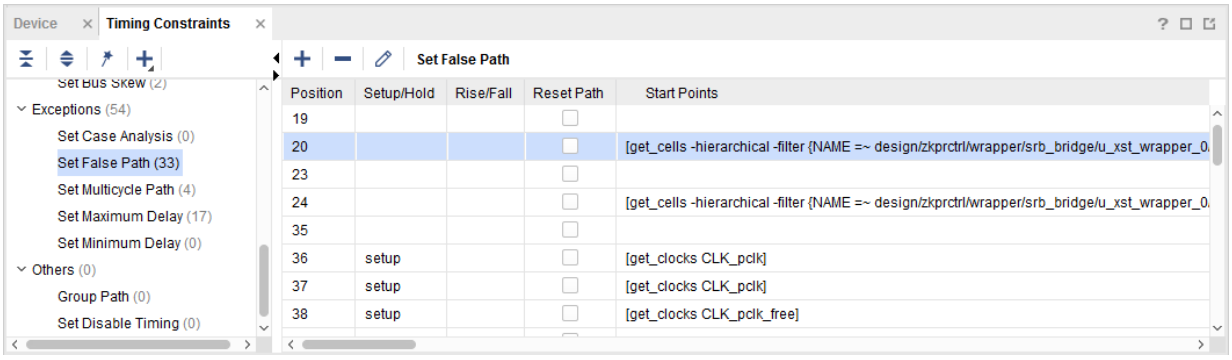
表格中的“Position”（位置）列表示与 Timing Constraints Editor (TCE) 所报告的位置编号相匹配的时序约束位置编号。您可双击任一行以重定向至 TCE 内选中的约束。或者也可以右键单击该行并从弹出菜单中选择“View Constraint”（查看约束）。

图 141：**“Report Exceptions”上下文菜单**



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

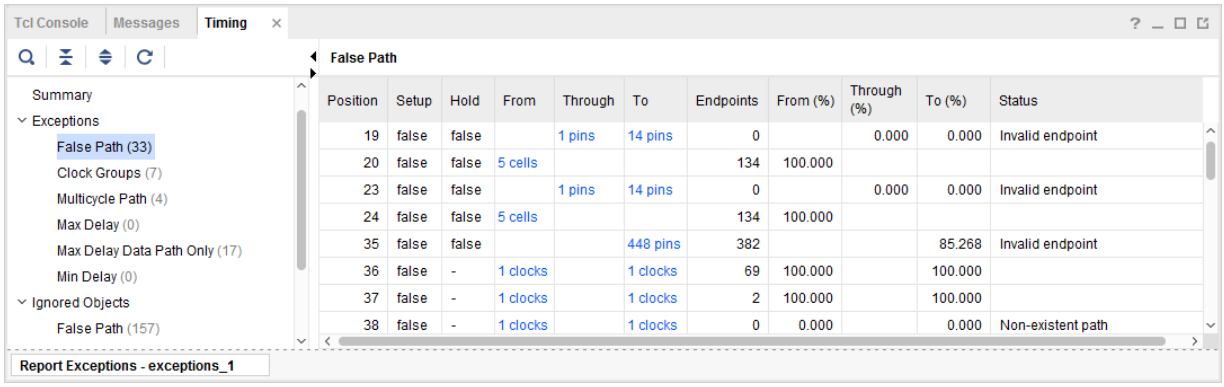
图 142：**Timing Constraint Editor 内的时序例外**



“From”（来源）、“Through”（穿过）和“To”（目标）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 143：**“Report Exceptions”：选中“Coverage”的详情表**



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。选择对象后，可使用“F4”键打开板级原理图。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“Status”（状态）列可报告约束状态，例如，“Invalid endpoint”（无效端点）、“Partially overridden path”（路径被部分覆盖）、“Non-existent path”（路径不存在）或者“Totally overridden”（完全覆盖）。在命令行上运行 `report_exception` 时，同样可报告这些状态：

- “Non-existent path”（不存在的路径）：例外被视为无效（不影响时序分析）。
- “Totally overridden”（完全覆盖）：例外被视为无效（不影响时序分析）。

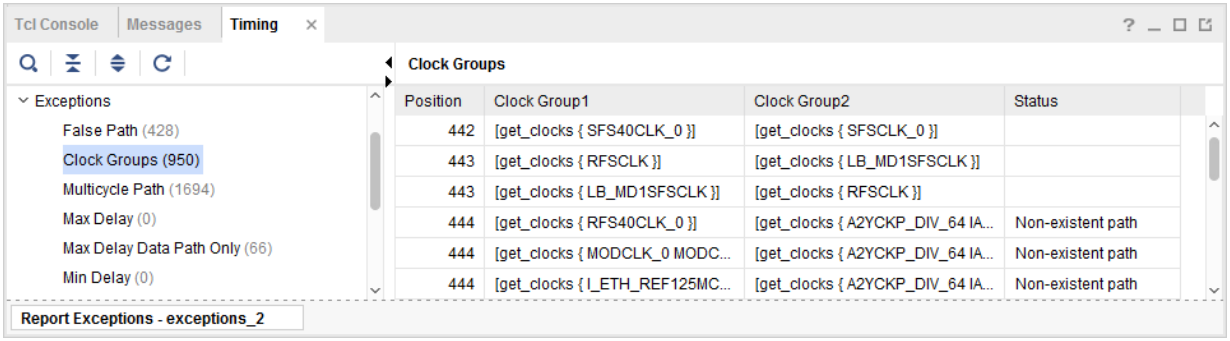
注释：覆盖率按如下顺序计算：“From”、“Through”和“To”。针对某一层次计算的覆盖率取决于上一层次。当给定层次计算所得覆盖率为 0% 时，所有后续层次都继承此覆盖率 0%。

注释：覆盖率为 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 144：**时钟组的详情表**



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：

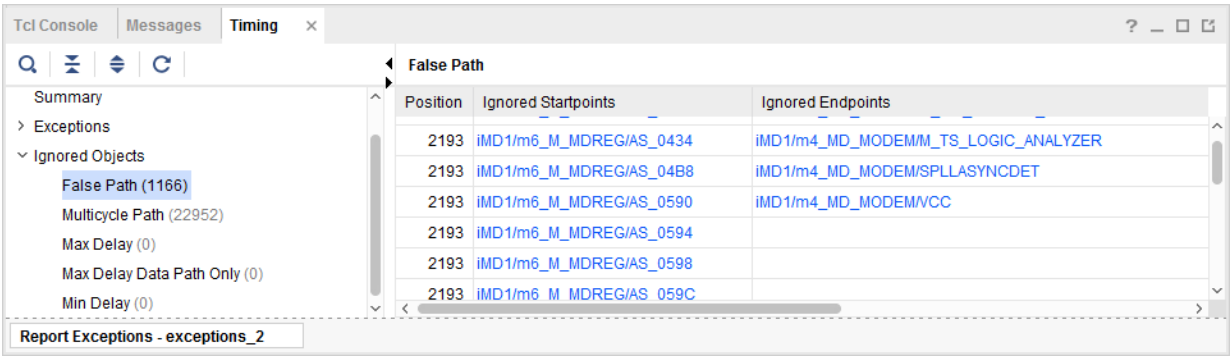
```
set_clock_groups -physically_exclusive -group RFSCLK -group LB_MD1SFSCLK
```

约束跨 2 行，因为从时钟 RFSCLK 到时钟 LB_MD1SFSCLK 之间存在部分时序路径，而在时钟 LB_MD1SFSCLK 到时钟 RFSCLK 之间同样存在部分时序路径。

“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 145：**“Report Exceptions”：“Ignored Objects”**



表中的“Position”（位置）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“View Constraint”（查看约束）。

表中的“Ignored Startpoints”（已忽略的起点）列和“Ignored Endpoints”（已忽略的端点）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。选择后，可在“Property”（属性）页面中查看属性，并通过“F4”键打开板级原理图。

Report Clock Domain Crossings

“Clock Domain Crossings (CDC)”（时钟域交汇）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“Clock Interaction”（时钟交互）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 控制台运行时，可使用 -cells 选项将 CDC 报告限定为一个或多个层级单元。如果 CDC 报告已限定作用域，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。限定作用域选项在“Report CDC”（CDC 报告）GUI 中不可用。

概述

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”（CDC 报告）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“Cross Domain Crossing (CDC)”（时钟域交汇 (CDC)）和时钟间时序分析的上下文中，术语“safe”（安全）、“unsafe”（不安全）和“endpoints”（端点）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，其他时序报告会将该总线报告为多个时序端点。

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“Report Clock Domain Crossings”

从 Vivado IDE 运行“Report CDC”（CDC 报告）时，默认情况下会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”（按时钟对汇总）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 log 日志文件。

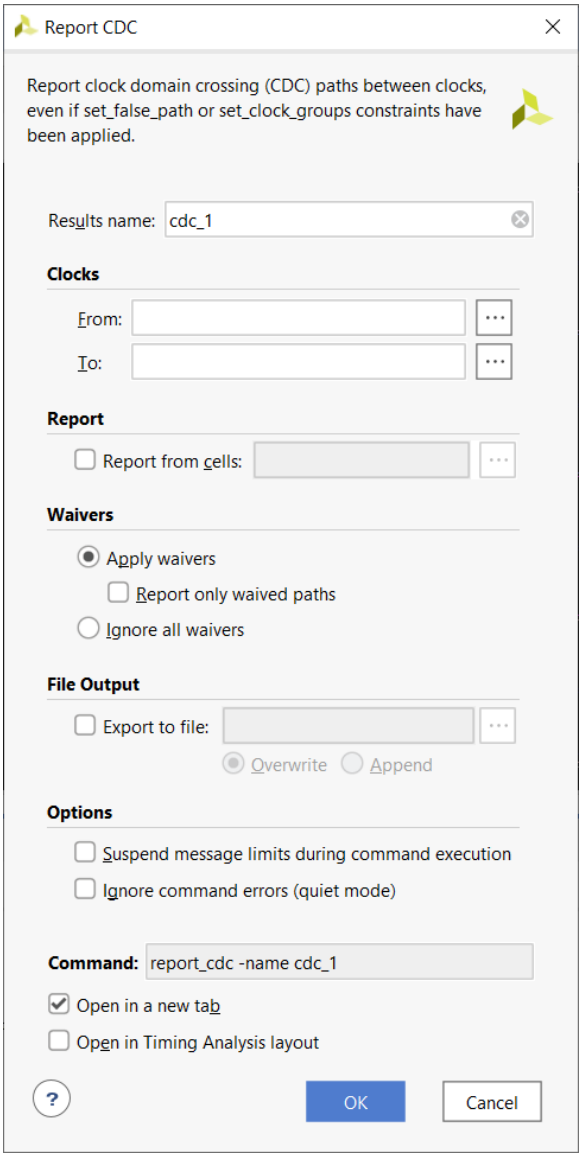
要在 Vivado IDE 中运行“Report Clock Domain Crossings”（时钟域交汇报告），请选择“Reports”→“Timing”→“Report CDC”（报告 > 时序 > CDC 报告）。

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：

- “Results Name”字段
- “Clocks”字段 (From/To)
- “File Output”字段
- “Options”字段

图 146：**“Report CDC”对话框**



“Results Name”字段

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

等效的 Tcl 选项：`-name <windowName>`

“Clocks”字段 (From/To)

“Clocks”（时钟）下的“To”（目标）和“From”（源）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”（CDC 报告），将其限定于特定时钟并生成更通俗易懂的报告。单击右侧“Browse”（浏览）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>`

“File Output”字段

“File Output”（文件输出）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`

“Options”字段

“Options”（选项）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”（CDC 报告）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 13：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 拓扑结构的简化板级原理图 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 14：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。例如，由于 CDC-15 的优先级高于 CDC-10，在寄存器上检测到的安全的 CDC-15 会屏蔽相同寄存器的管脚 D 上的不安全的 CDC-10。

注释：使用以下命令行选项即可覆盖此默认行为：

```
-all_checks_per_endpoint
```

此选项会强制工具报告应用于端点的所有“Info”（参考）、“Warning”（警告）和“Critical”（严重）检查，而不考虑规则优先级。但如果在寄存器上检测到至少一条安全规则，就不会报告相同寄存器上的不安全规则。

表 13：**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 ASYNC_REG 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 ASYNC_REG 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 ASYNC_REG 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 ASYNC_REG 属性。

表 13：**CDC 规则与描述** (续)

CDC 拓扑结构	CDC 规则	严重性	描述
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 ASYNC_REG 属性。
	CDC-9	Info	1 个异步复位已同步，并且存在 ASYNC_REG 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 原语上已检测到 CDC。
CE 控制的 CDC	CDC-15	Warning	时钟使能控制的 CDC。
多路复用器控制的 CDC	CDC-16	Warning	多路复用器控制的 CDC。
多路复用器数据保持 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 HARD_SYNC 原语同步。
LUTRAM-to-FD CDC	CDC-26	Warning	LUTRAM 读取/写入存在潜在冲突。

表 14：**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
多路复用器数据保持 CDC	CDC-17
多路复用器控制的 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

- 按类型汇总
- 详情报告

各汇总部分提供了需复查并且可能需更改设计的问题概览。这些部分可用于浏览严重性最高的违例，在“Detailed Report”（详情报告）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“Summary by clock pair”（按时钟对汇总）部分。

按时钟对汇总

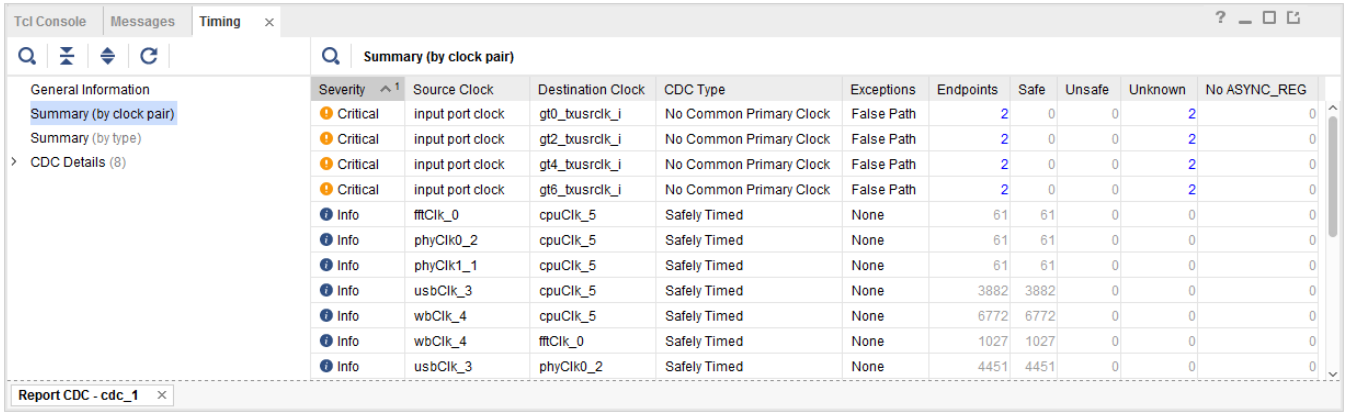
在“Summary (by clock pair)”（按时钟对汇总）部分中，提供了有关 2 个时钟之间的 CDC 路径数量以及这些路径中找到的最严重问题的严重性的实用信息。该表包含以下几个列：

- “Severity”（严重性）：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：“Info”（参考）、“Warning”（警告）或“Critical”（严重）。
- “Source Clock”（源时钟）：显示 CDC 源时钟的名称。
- “Destination Clock”（目标时钟）：显示 CDC 目标时钟的名称。
- “CDC Type”（CDC 类型）：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：
  - “Safely Timed”（安全定时）：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。
  - “User Ignored”（用户忽略）：`set_false_path` 或 `set_clock_groups` 已涵盖所有 CDC 路径。
  - “No Common Primary Clock”（无公共基准时钟）：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。
  - “No Common Period”（无公共周期）：CDC 时钟处于异步状态，在 2 个不含公共周期的时钟间至少 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 时序分析关键概念。
  - “No Common Phase”（无公共相位）：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。
- “Exceptions”（例外）：应用于 CDC 的时序例外（如果有）包括：
  - “None”（无）：CDC 路径上不存在下列任何时序例外：“Clock Group”（时钟组）、“False Path”（伪路径）或“Max Delay Datapath Only”（仅最大延迟数据路径）。`report_cdc` 不报告其他时序例外，如“Multicycle Paths”（多周期路径）、“Min Delay”（最小延迟）和“Max Delay”（最大延迟）。
  - “Asynch Clock Groups”（异步时钟组）：`set_clock_groups -asynchronous` 例外已应用于 CDC 时钟。
  - “Exclusive Clock Groups”（互斥时钟组）：`set_clock_groups -exclusive` 例外已应用于 CDC 时钟。
  - “False Path”（伪路径）：`set_false_path` 例外已应用于往来 CDC 时钟的路径或所有 CDC 路径。
  - “Max Delay Datapath Only”（仅最大延迟数据路径）：`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，而所有其他 CDC 路径均已因 `set_false_path` 约束而被忽略时，才会报告“Max Delay Datapath Only”。
  - “Partial Exceptions”（部分例外）：`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。

- “Endpoints”（端点）：CDC 路径端点总数。这是处于“Safe”（安全）、“Unsafe”（不安全）和“Unknown”（未知）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。
- “Safe”（安全）：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
- “Unsafe”（不安全）：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
- “Unknown”（未知）：处于未知状态的 CDC 路径端点数量。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）。
- “No ASYNC_REG”（无 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 147：**“Summary (by clock pair)”部分**

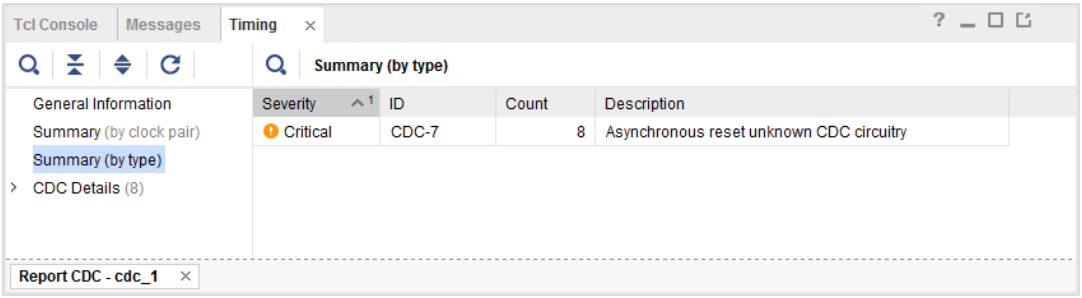


Severity	Source Clock	Destination Clock	CDC Type	Exceptions	Endpoints	Safe	Unsafe	Unknown	No ASYNC_REG
Critical	input port clock	gt0_txusrclk_i	No Common Primary Clock	False Path	2	0	0	2	0
Critical	input port clock	gt2_txusrclk_i	No Common Primary Clock	False Path	2	0	0	2	0
Critical	input port clock	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	fftClk_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	fftClk_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”（按类型汇总）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 148： **“Summary by Type”表**



“Summary by Type”表包含以下列：

- “Severity”（严重性）：报告 CDC 规则的严重性，值为：Info（参考）、Warning（警告）或 Critical（严重）。
- ID：CDC 规则的唯一标识号，详见 表 13：CDC 规则与描述。
- “Count”（计数）：整个报告中出现的 CDC 规则数量。
- “Description”（描述）：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：

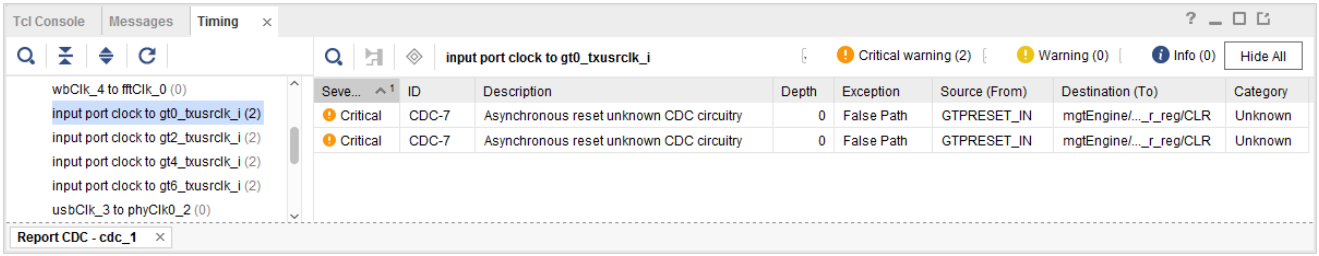
- Critical：此严重性对应于含未知或不安全的 CDC 结构的 CDC 路径。您必须复查每条路径，并通过修改 RTL 来修复结构或者将问题豁免。默认情况下，使用 Vivado IDE 时会生成路径详情，前提是在命令行上将 `-details` 选项与 `report_cdc` 命令配合使用。
  - 在交汇信号线上存在一些组合逻辑，或者在交汇信号线的扇入中发现若干源时钟。这可能导致“Mean Time Between Failures (MTBF)”（平均故障间隔时间）特性劣化。
  - 在交汇信号线上存在通向相同目标时钟域的扇出。这可能导致数据一致性问题。
- Warning：此严重性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。
- Info：此严重性表明 CDC 结构全部安全且已完成正确约束。

详情报告

通过查看“Report CDC”（CDC 报告）中的“CDC Details”（CDC 详情）部分即可查看此报告的详情。您可以使用详情报告来查看所选路径的板级原理图（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板级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 Detailed Report”示例如下图所示。

图 149：**CDC 详情报告**



“CDC Detailed Report”表包含以下几个列：

- “Severity”（严重性）：报告 CDC 规则的严重性，值为：Info（参考）、Warning（警告）或 Critical（严重）。
- ID：CDC 规则的唯一标识号，详见 表 13：CDC 规则与描述。
- “Description”（描述）：CDC 规则的简短描述。
- “Depth”（深度）：已找到的同步器阶段数量（仅适用于同步器拓扑结构）。
- “Exception”（例外）：应用于 CDC 路径的时序例外。
- “Source (From)”（源）：CDC 时序路径起点。
- “Destination (To)”（目标）：CDC 时序路径端点。
- “Category”（类别）：显示为 Safe（安全）、Unsafe（不安全）、Unknown（未知）。
- “Source Clock (From)”（源时钟）：源时钟名称。

  注释：仅当您单击“CDC Details”（“Timing-Report CDC”窗口的左列）时，才会显示此列
- “Destination Clock (To)”（目标时钟）：目标时钟的名称。

  注释：仅当您单击“CDC Details”（“Timing-Report CDC”窗口的左列）时，才会显示此列

> **重要提示！**CDC 报告可标记某些 AMD 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 AMD IP 产品指南。

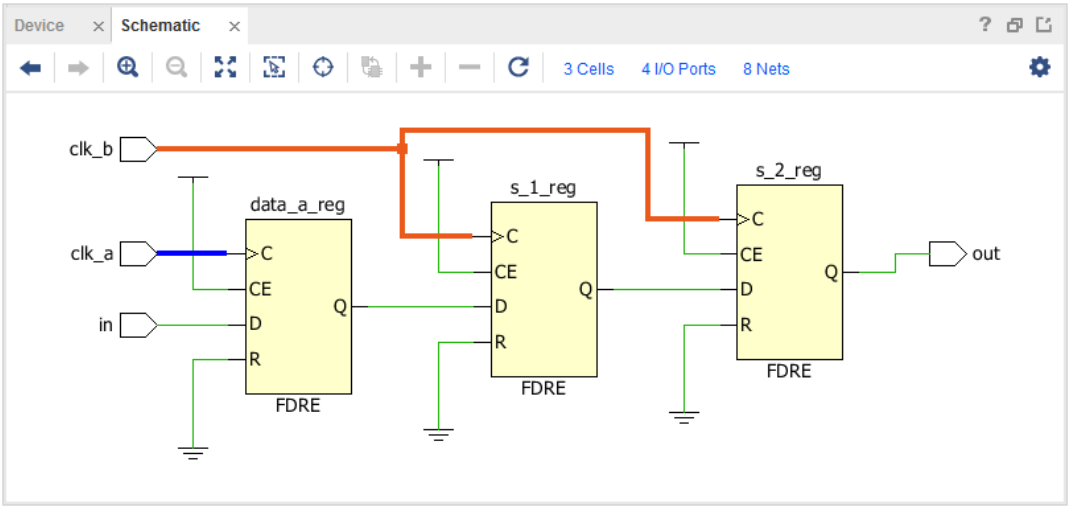
CDC 拓扑结构的简化板级原理图

以下部分展示了 CDC 拓扑结构的简化板级原理图以及简要说明。在所有板级原理图中，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 150：**单比特同步器的简化拓扑结构**

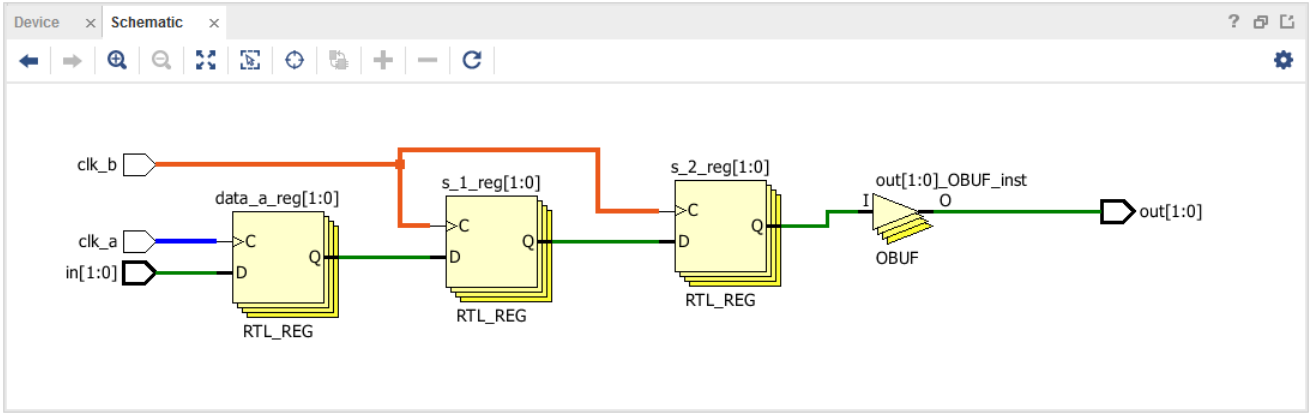


如果触发器的 CLEAR 或 PRESET 管脚同时连接到异步源，那么此同步器在报告中仅显示为单比特同步器而非异步复位同步器。

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 151：**位宽为 2 的多比特同步器**



如果 CDC 总线的某些位不匹配相同 CDC 规则，那么该总线将以单个位或总线分段的形式来报告，这些单个位或总线分段都具有匹配相同 CDC 规则的连续索引。

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，鉴于工具无法判定拓扑结构是否足以满足设计，CDC 规则 CDC-6 归类为“Warning”（警告）。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，则应改为使用其他同步器拓扑结构，例如，CE 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` 以安全方式对这两个信号的断言无效进行时序约束。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 152：**基于 CLEAR 的异步复位同步器**

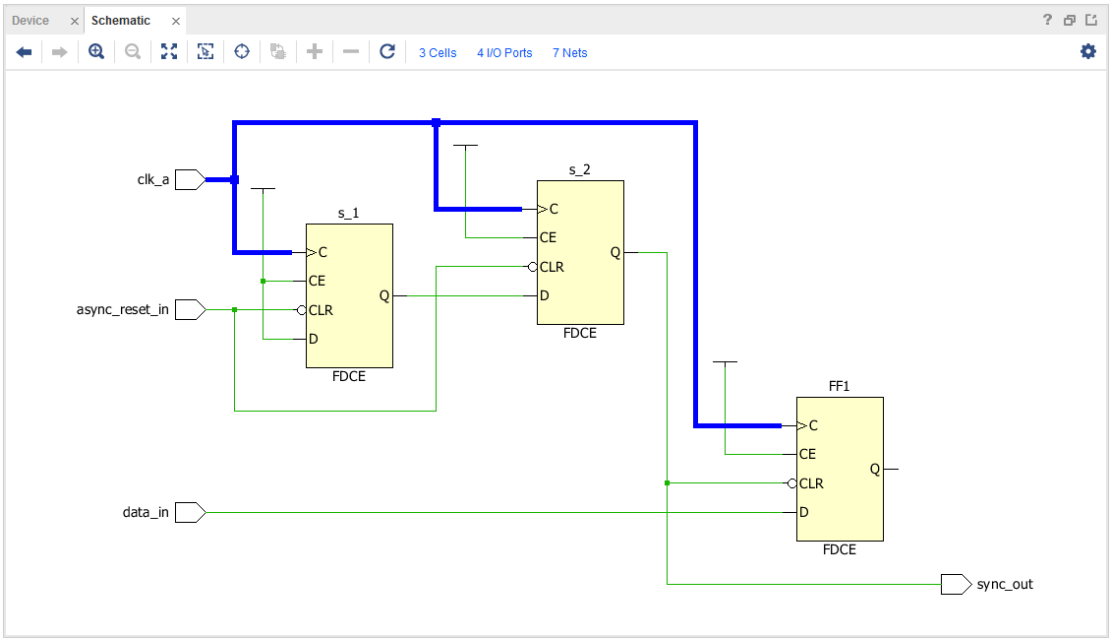
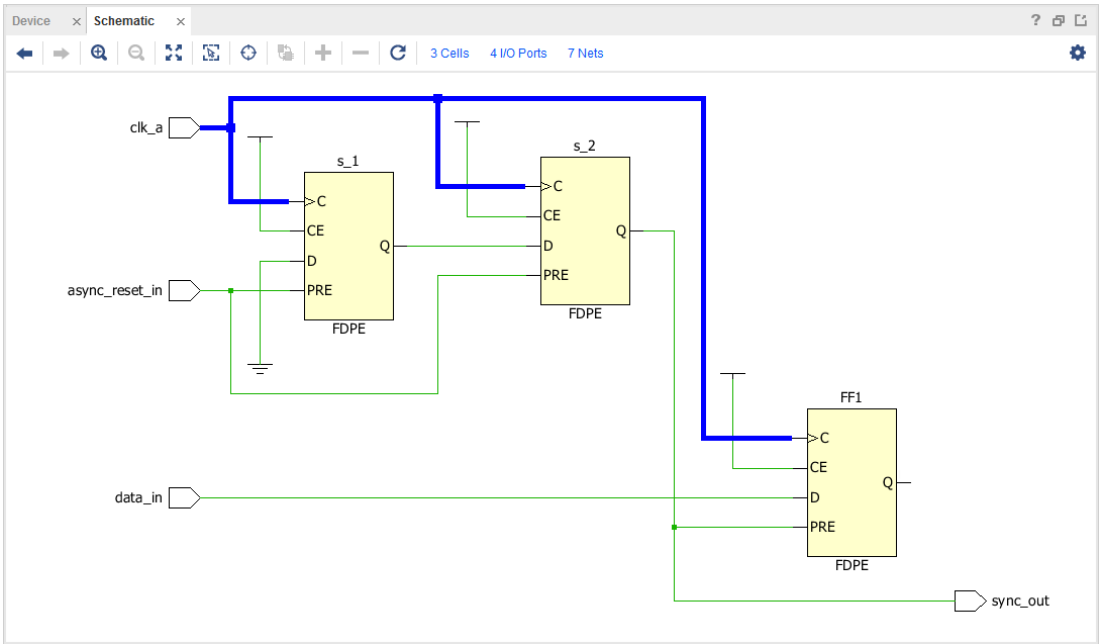


图 153：**基于 PRESET 的异步复位同步器**



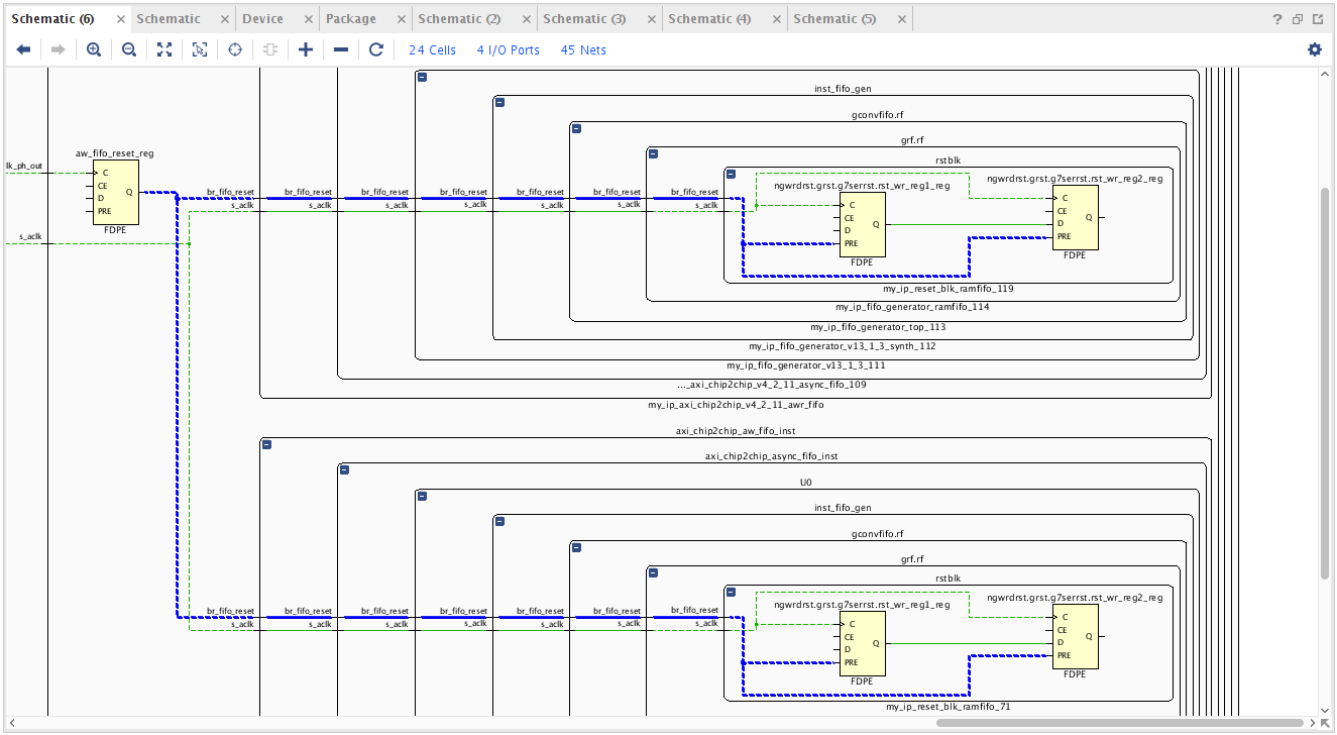
通常建议避免在目标时钟域内包含多个复位信号的同步。这意味着从源时钟域到目标时钟域的复位不应存在任何扇出。此建议可防止目标时钟域在不同时间解复位从而导致设计处于未知状态。不遵循此建议会导致从发送触发器到目标时钟出现严重的 CDC-11 扇出违例。

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解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。

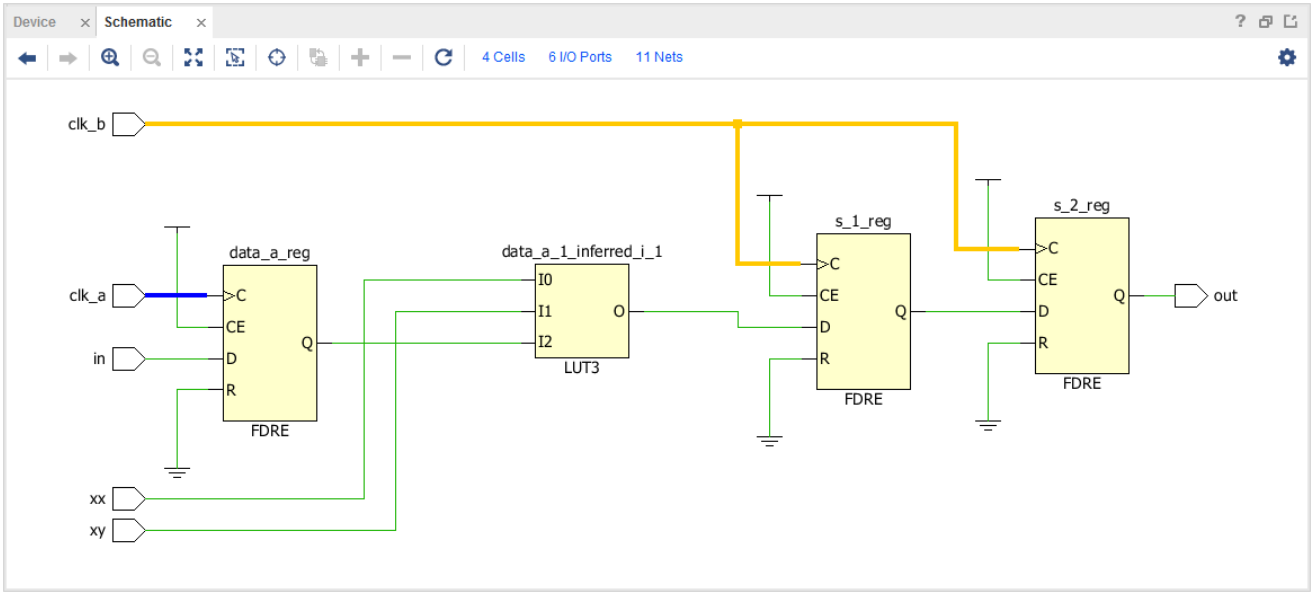
图 154：**安全复位同步示例**



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 个逻辑函数，由 LUT3 表示。

图 155：**组合逻辑简化示例**

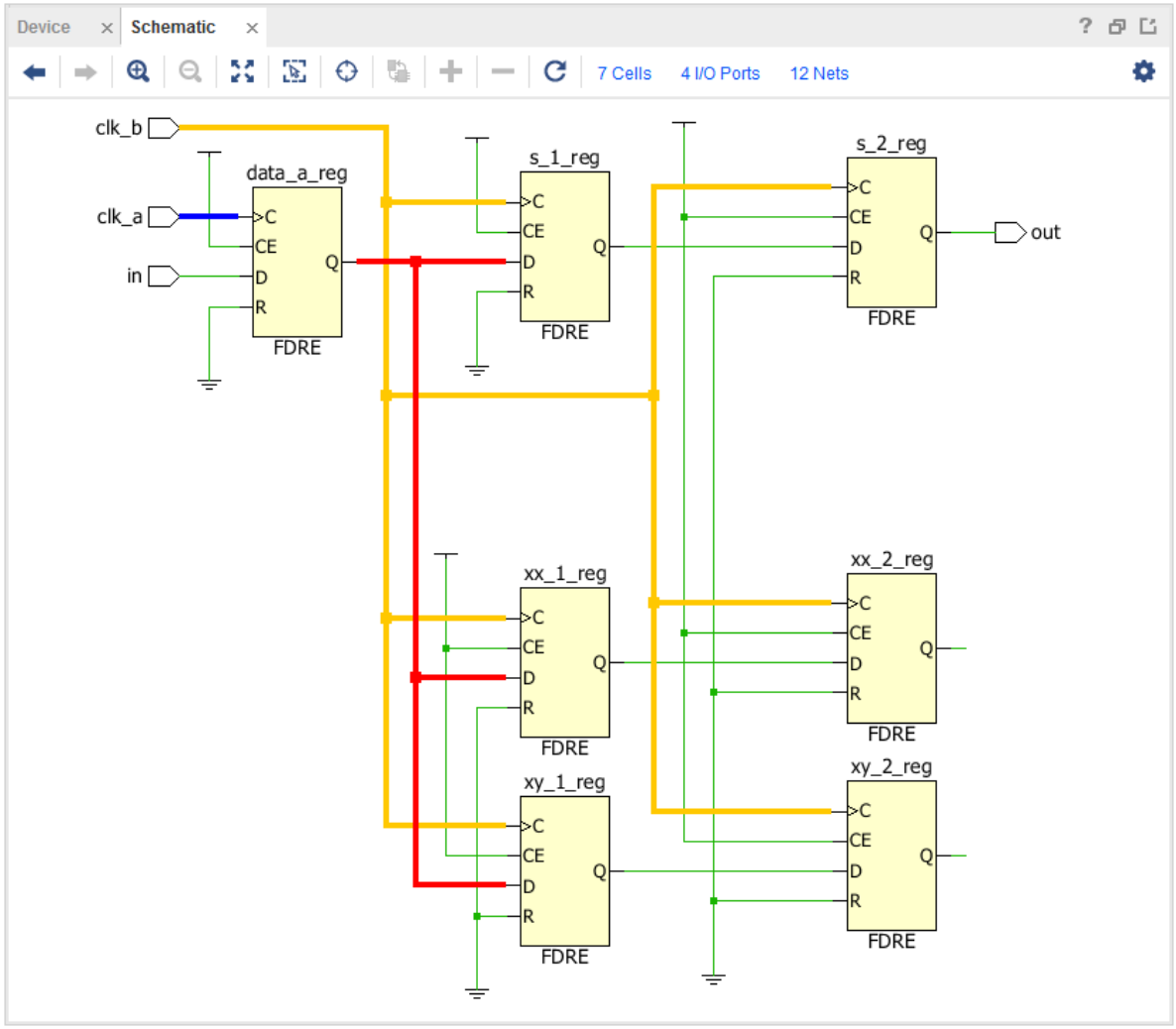


原先不推荐采用此结构，因为在组合逻辑的输出上可能出现毛刺，同步器会捕获此类毛刺并传输至设计下游其余部分。

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限制，但并不具备周期精确性。

图 156：**经简化的扇出示例**

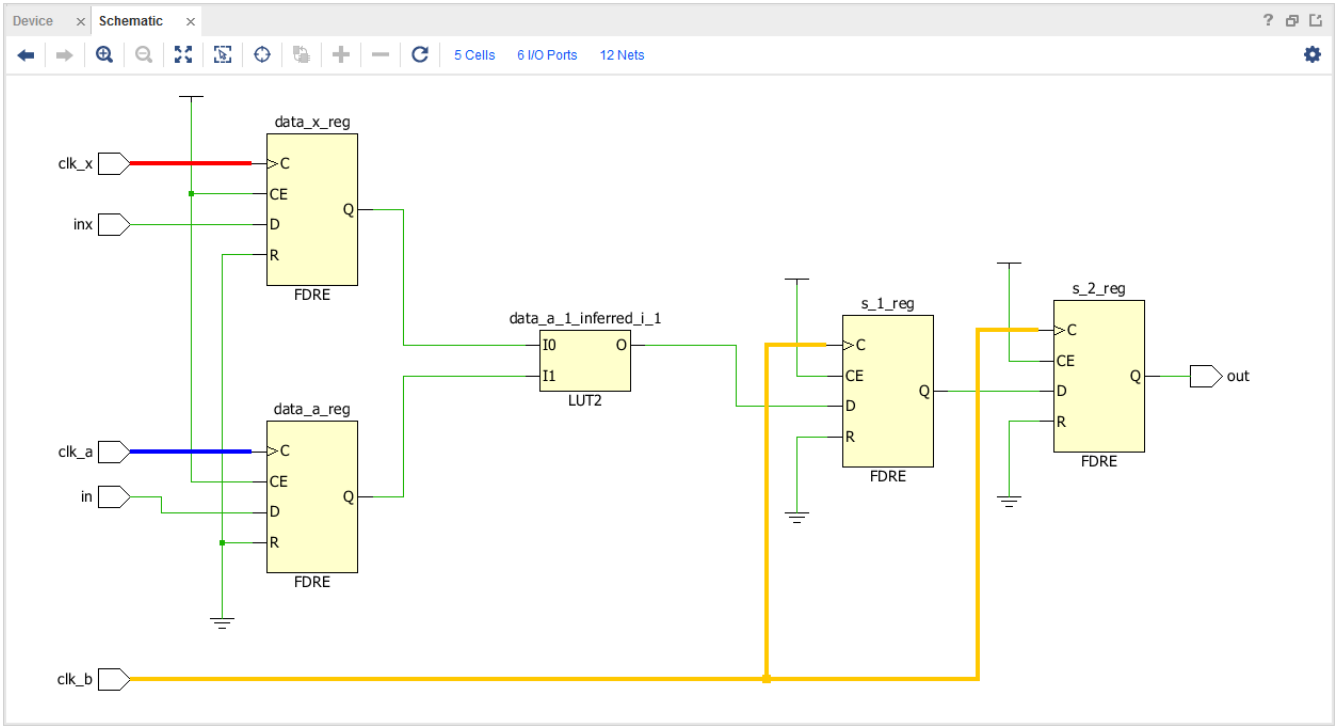


注释： N 到 N 个不同时钟域的扇出并不构成 CDC 问题，不会触发 CDC-11 违例。请参阅 异步复位同步器 部分，以获取复位信号上的安全扇出示例。

多时钟扇入

在下图所示“Multi-Clock Fanin”（多时钟扇入）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或 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。

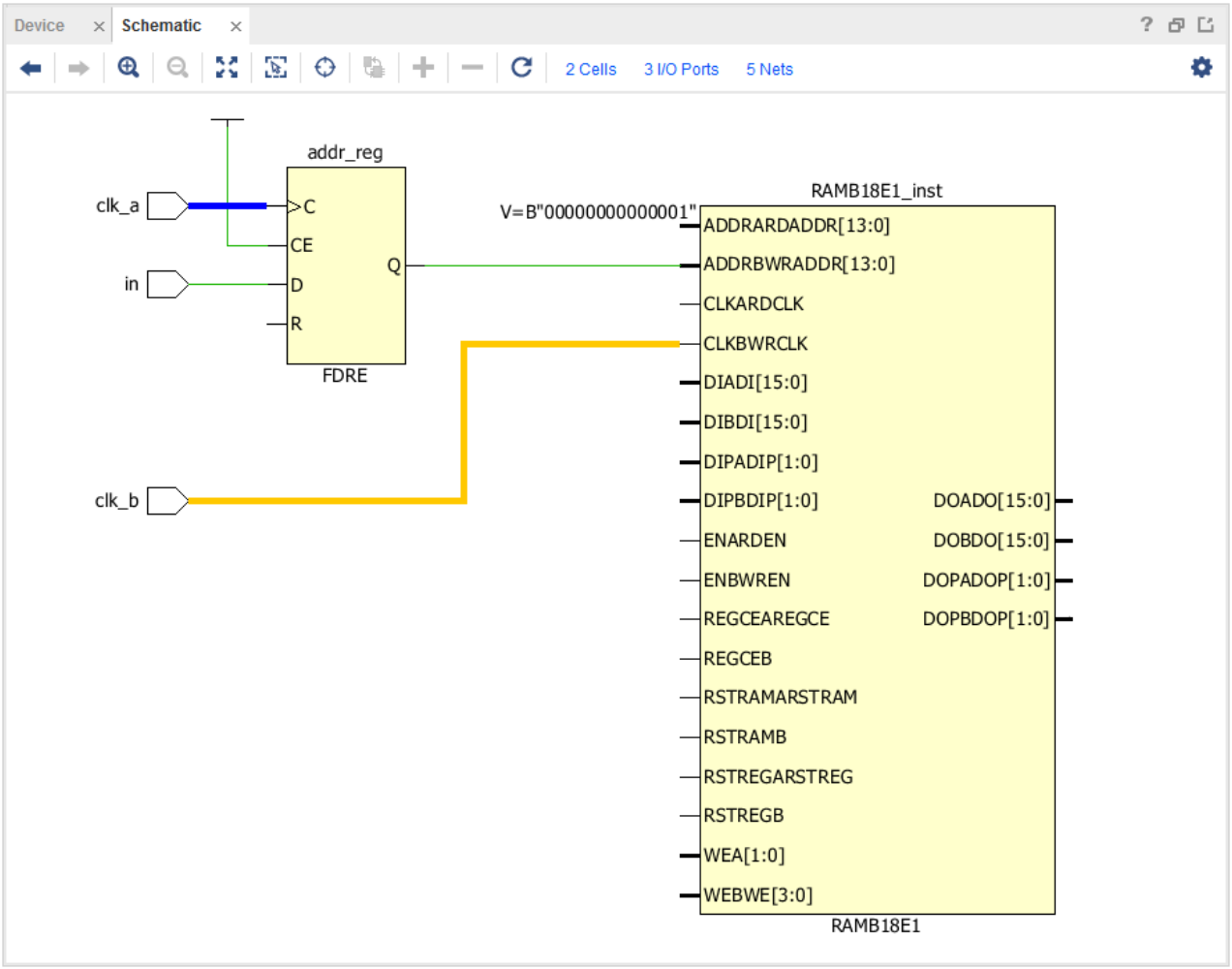
图 157：**多时钟扇入示例**



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 158：**非 FD 原语示例**

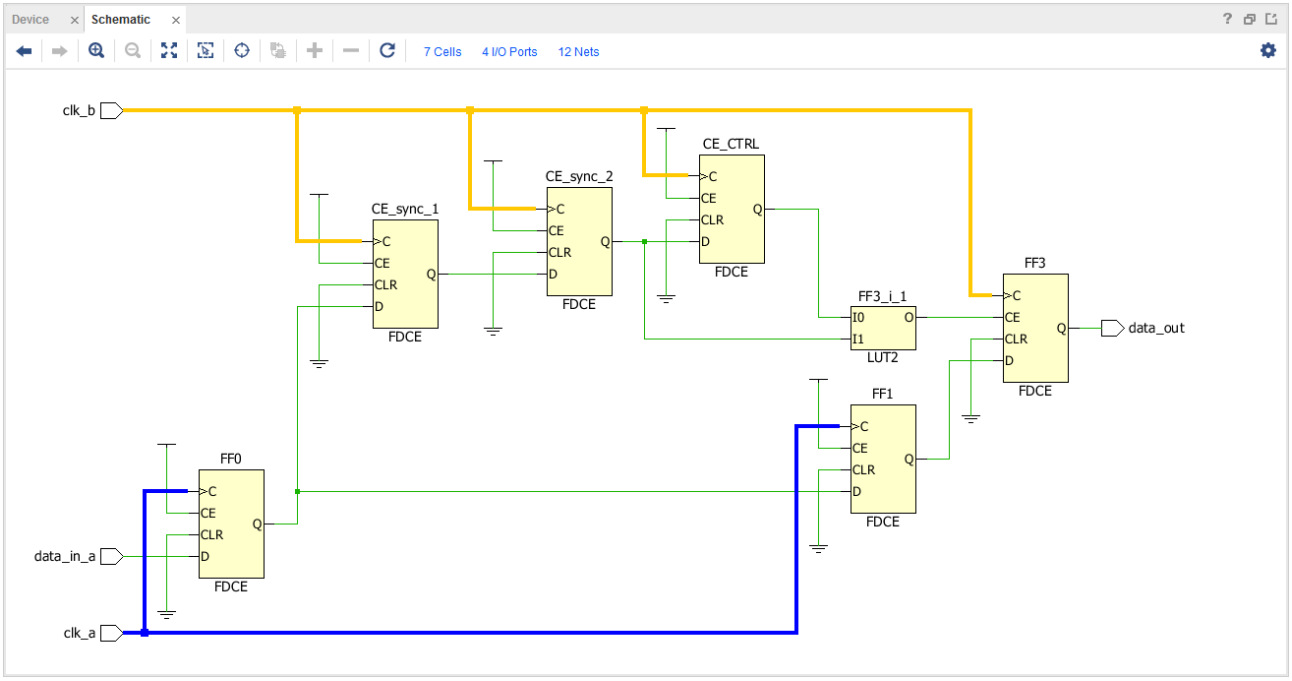


注释：此规则不包含由 CDC-18 检测和覆盖的 `HARD_SYNC` 宏。

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 159：**CE 控制的 CDC 示例**

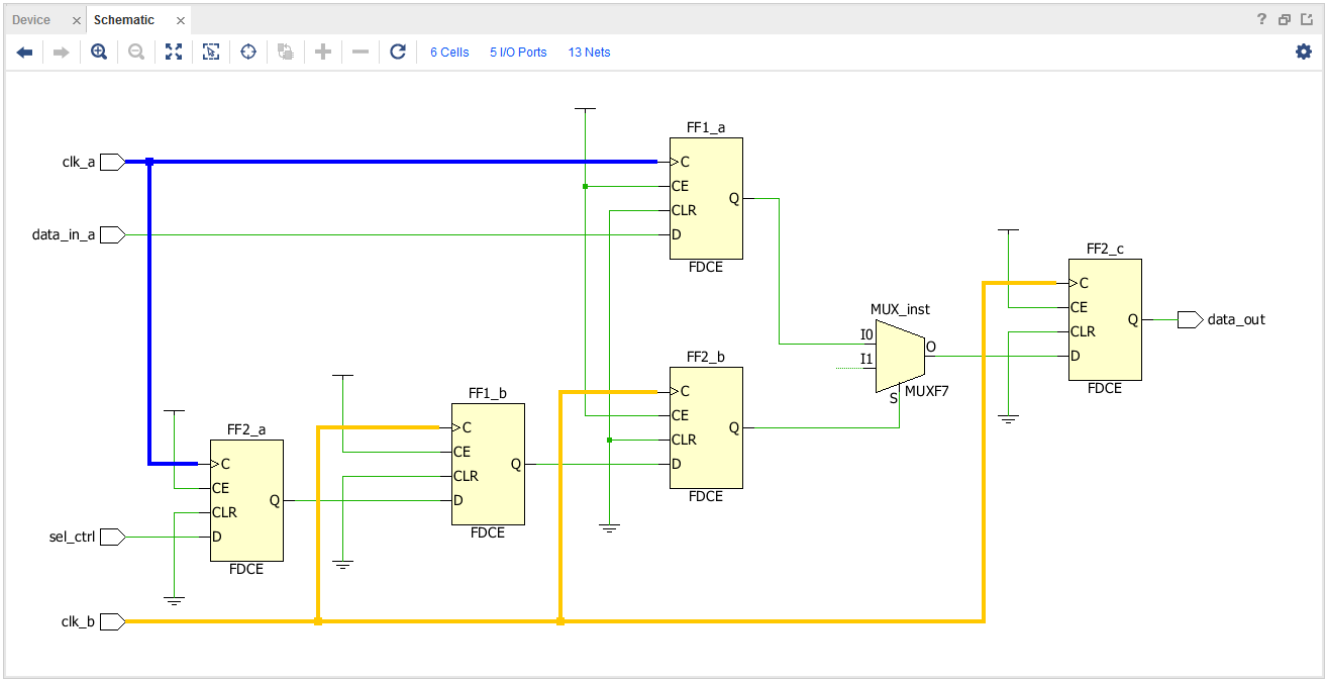


CDC 引擎仅检查连接到 FF3/CE 的信号是否同样由 `clk_b` 发送。在驱动 CE 管脚的电路上，对于时钟使能信号的同步方式不存在限制，前提是将其作为安全的 CDC 路径单独报告即可。此外，您还负责约束从 `clk_a` 域到 FF3 的时延，这通常是通过 `set_max_delay -datapath_only` 约束来完成的。

多路复用器控制的 CDC

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

图 160：**多路复用器控制的 CDC 示例**

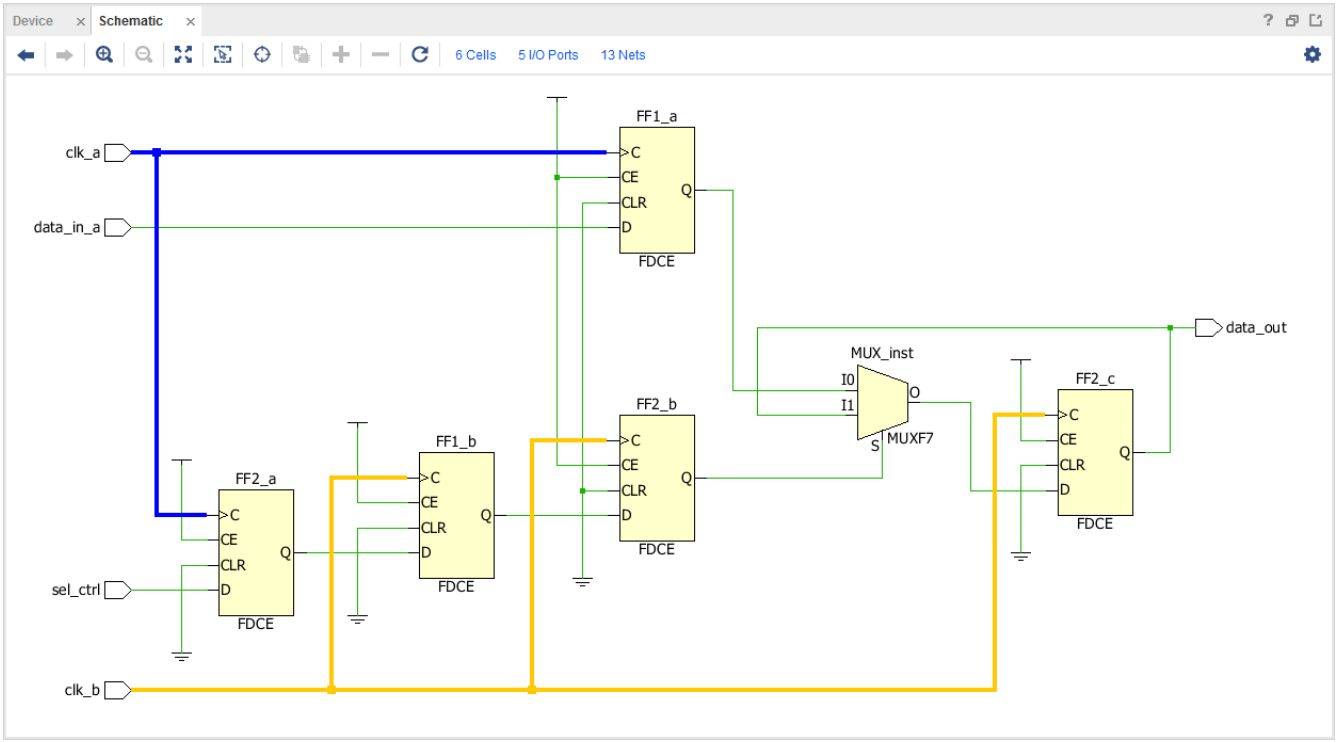


与 CE 控制的 CDC 类似，对于选择信号的同步方式并无限制，前提是此信号单独报告为安全，并且用户负责约束 `FF2_c` 上的交汇延迟。

多路复用器数据保持 CDC

在下图所示的多路复用器数据保持 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 161：**多路复用器数据保持 CDC 示例**



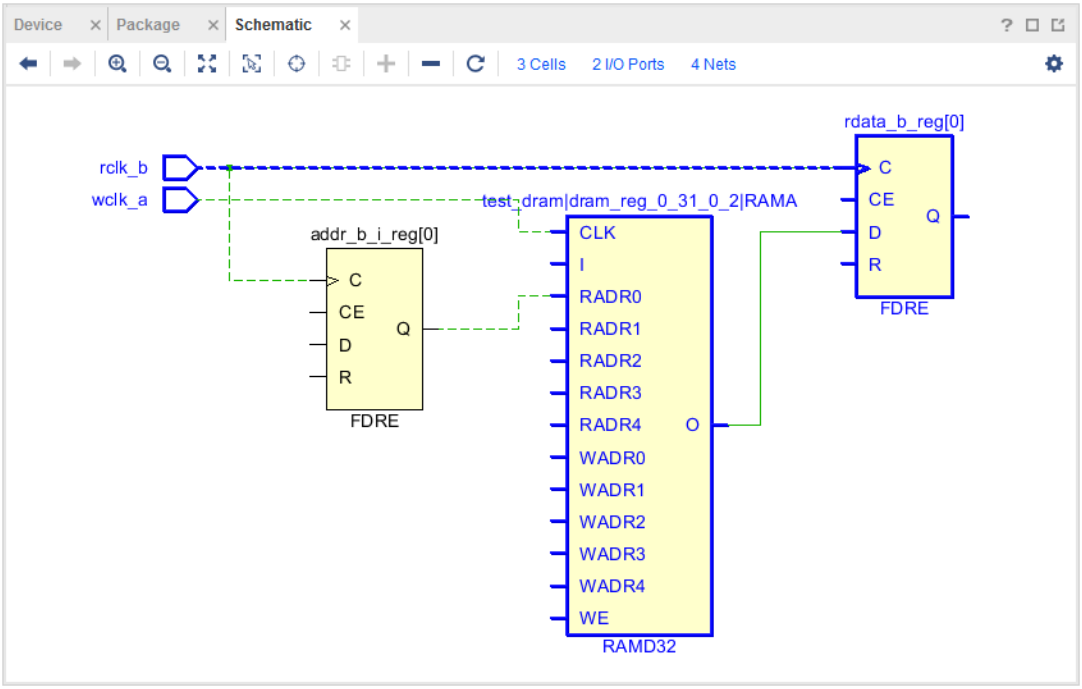
与 CE 控制的 CDC 类似，对于多路复用器选择信号的同步方式并无限制，前提是此信号单独报告为安全，并且用户负责约束 `FF2_c` 上的交汇延迟。

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，AMD 的 FIFO Generator IP 具有防止任意读写冲突的内置逻辑。

图 162：**LUTRAM 读写潜在冲突**



时序收敛报告

Report QoR Assessment

`report_qor_assessment` 命令会生成报告以提供下列信息：

- 评估得分，用于指示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
- 使用率和性能指标汇总信息
- 对于 QoR 至关重要的方法论检查汇总信息（仅限在文本版本中提供）
- 有关 ML 策略可用性的信息

总体评估汇总

此汇总包含 QoR 评估得分和流程指南信息。

评估得分可以预测设计在实现流程中给定时间点达成时序目标的可能性。在流程中越早运行该命令，可节省的编译时间更多，因此收益就越大。虽然准确性略有牺牲，得分比最终布线后得分高，但差值应不超过 1。此得分是通过分析一组复杂的设计指标来生成的，这些指标包括 UltraFast 方法论、器件使用率、控制集、时钟设置、建立裕量和保持裕量等。此外，其中还考量了器件特有的特性和设计阶段。例如，执行 `synth_design` 后，将对时钟设置网表结构进行仔细检查；在 `place_design` 时钟偏差随着准确性增加而具有更大的权重之后以及在 `route_design` 之后，将考虑其他新因素，例如设计能否完全布线。评分范围为从 1 至 5。如果得分低于 5，请使用 `report_qor_suggestions` 来改善得分。

在下表中提供了评分详细信息：

表 15：**QoR 评估报告评分**

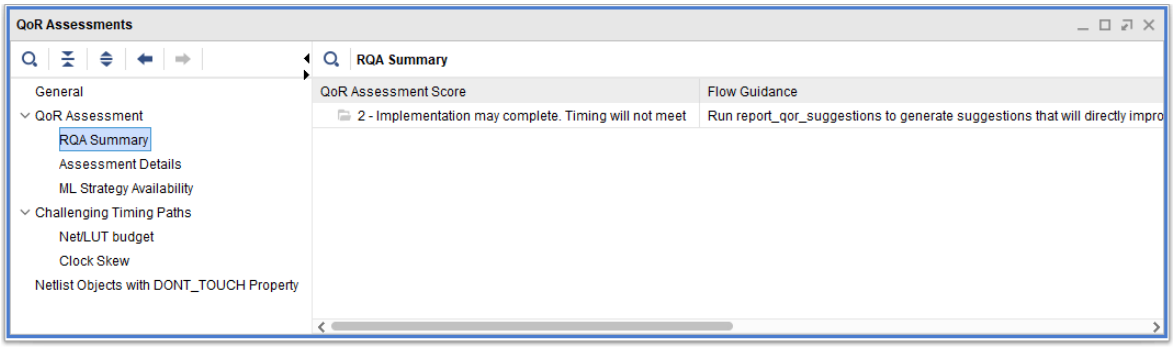
得分	含义
1	设计将可能无法完成实现。
2	设计将能够完成实现，但无法满足时序。
3	设计将可能无法满足时序。
4	设计将可能满足时序。
5	设计将能够满足时序。

流程指南属于总体评估汇总的一部分。它会根据设计的当前状态进行动态更新。它可提供以下相关信息：

- 您是否需要解决方法论问题
- 使用 QoR 建议是否改善设计
- 使用 ML 策略还是增量编译

下图显示了“QoR Assessment Score”（QoR 评估得分）为 2 的设计。

图 163：**总体评估汇总**

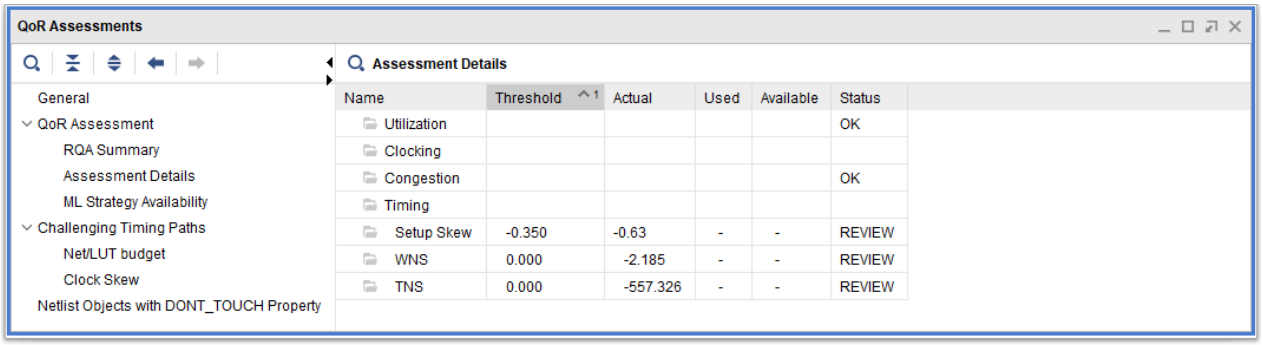


QoR Assessment Details

“QoR Assessment Details”（QoR 评估详情）表如下图所示，其中提供了便利的设计概览，着重显示奠定 RQA 评分基础的以下领域的问题。

- Utilization（使用率）
- Netlist（网表）
- Clocking（时钟设置）
- Congestion（拥塞）
- Timing（时序）

图 164：**QoR Assessment Details**



该表显示了分为 5 个类别的设计特性。每个类别中如无任何子项标记为 REVIEW，则该类别标记为 OK。如有子项标记为 REVIEW，则会显示时序失败的项及其阈值和当前值。阈值并非硬性限制，可超出阈值限制，但可能导致难以达成时序收敛。如果阈值超出过多或者有众多类别均超出其阈值，则需特别留意。标有 * 的项并不直接参与评分，但对于设计是否能满足时序而言这些项至关重要，故而必须仔细检查。

使用率检查是在 SLR 级别和 Pblock 级别对整个器件执行的检查。运行 `report_qor_suggestions` 有助于降低使用率。

网表检查是针对网表结构和非时序约束执行的检查。这些检查会识别具有 DONT_TOUCH 属性的项、驱动程序剖析信息欠佳的高扇出信号线以及可能给实现工具增加困难的其他设计功能特性。

时钟设置可显示建立时间路径或保持时间路径上时钟偏差是否过高。失败的时钟偏差路径会被自动添加到 Vivado IDE 中。在文本模式下，添加 `-csv_output_dir <directory>` 即可生成 CSV 格式的时序路径。运行 `report_qor_suggestions` 可以给众多时钟偏差问题提供自动修复。

拥塞会查看网表中的剖析信息，寻找可能造成布线拥塞的问题。拥塞区域信息在布局前不可用，但有部分网表项可用。您可先运行布局布线来评估拥塞，而后再修复这些项。运行 `report_qor_suggestions` 可生成相关建议，以拥塞区域内的单元为目标来减少拥塞。

时序会查看每个时钟组中 100 条最差的路径。它会查看：

- WNS、TNS、WHS 和 THS，判定设计是否有可能达成时序收敛。
- 信号线预算检查的是可布线的信号线，其中将添加保守的信号线延迟，而不是添加估算的延迟。
- LUT 预算检查的是 LUT，将延迟替换为保守的 LUT 延迟，而不是使用估算的延迟。

LUT 和信号线预算检查都允许使用低于理想值的估算值。通过解决超出裕量的路径中的问题，以减少设计流程中后续出现的问题数量。请参阅 Vivado IDE 中的“Challenging Timing Paths”（时序收敛困难的路径）部分，或者生成 CSV 文件以查看有关这些文件的更多信息。

在已布线的设计上，通过检查其他功能特性即可使用“last mile”（最后一步）指令查看设计是否收敛，该指令是在“Intelligent Design Runs”（智能设计运行）功能特性内部使用的指令。它会基于最差情况时序路径内涉及的 WNS、WHS、路径前后裕量和原语，检查时序路径是否能满足时序。

方法论检查

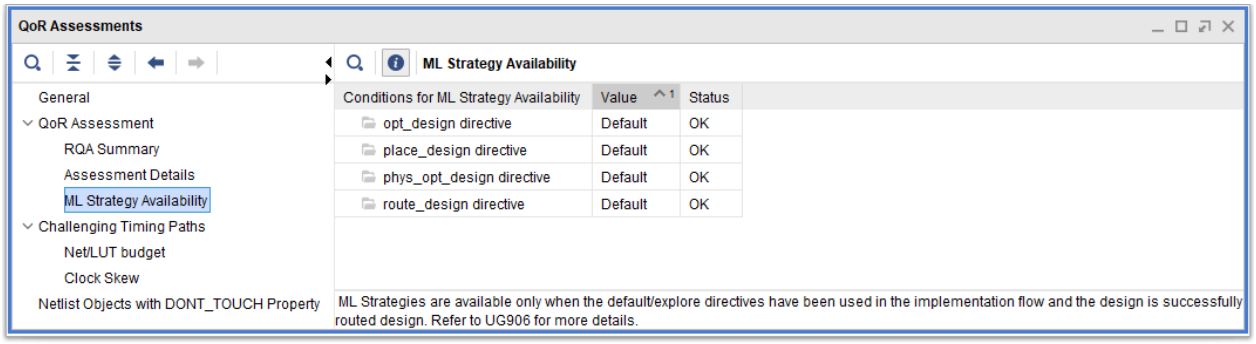
使用 `report_methodology` 运行有限数量的方法论检查以奠定坚实的基础，确保 QoR 建议有效。如果已生成方法论检查，那么除非设计中存在变更，否则就会复用缓存的结果。如需运行方法论检查，则会导致运行时间增加。方法论检查可使用 `-exclude_methodology_checks` 开关来禁用。

注释：在 Vivado IDE 中，不生成本节。而是改为使用“Report Methodology”（方法论报告）来访问方法论信息。默认情况下，对于大部分实现报告策略都会生成此信息。

ML Strategy Availability

如果参考运行中尚未运行所需的实现运行，那么 `report_qor_suggestions` 不会生成 ML 策略。“ML Strategy Availability”（ML 策略可用性）表（如下图所示）会核对每一个必需的实现步骤。

图 165：**“ML Strategy Availability”**



流程要求如下：

- `opt_design` 命令必须搭配 `Explore` 或 `Default` 指令一起运行。

  注释：允许多次调用 `opt_design`，但最后一次调用必须满足此条件。
- 剩余实现指令必须全部设为 `Default` 或全部设为 `Explore`。不允许混用搭配这些实现步骤。
- 必须启用 `phys_opt_design` 命令。
- 设计必须完成布线。
- 此设计必须为 AMD UltraScale™ 或 AMD UltraScale+™ 设计。

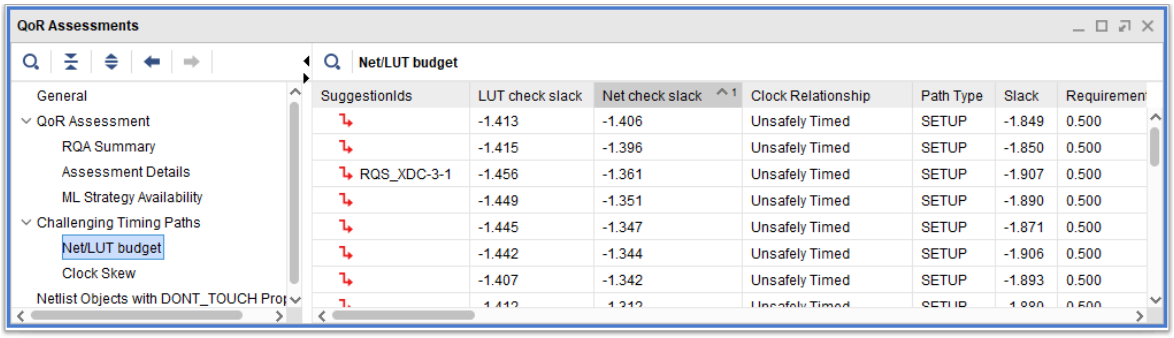
Challenging Timing Paths

“Challenging Timing Paths”（时序收敛困难的路径）部分列出了“Assessment Details”（评估详情）部分中未能通过检查的时序路径的关键属性。默认情况下，该命令会对每个时钟组中的 100 条失败的路径进行评估。它会分析下列因素：

- 信号线预算
- LUT 预算
- 时钟偏差

下图显示了“Net/LUT Budget”（信号线/LUT 预算）报告的示例。

图 166： **“Net/LUT Budget”报告**



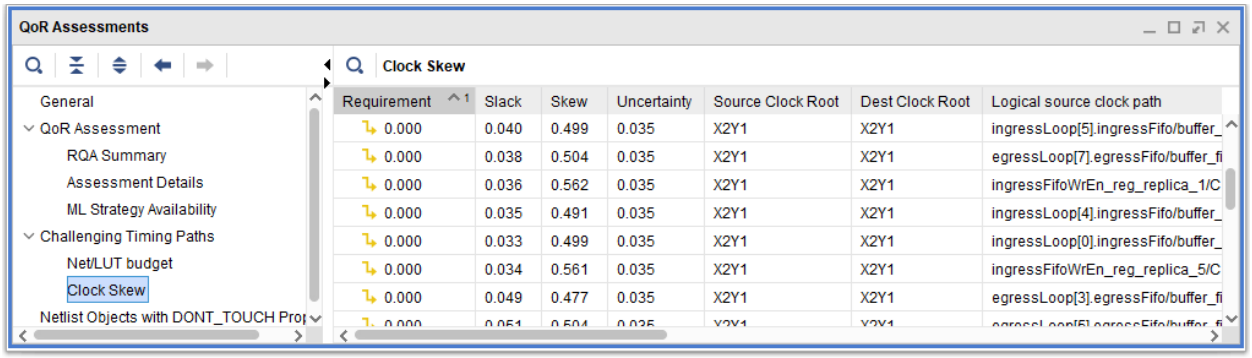
对于这些检查，估算的信号线或 LUT 延迟会被替换为设计所期望的典型值，并对新预算进行计算。如果路径始于或者止于块 RAM、DSP 或其他硬核块，则会给这些路径添加惩罚。如果无法利用时钟树偏差来改善裕度，则会有其他惩罚。在“LUT Check Slack”（LUT 检查裕量）列和“Net Check Slack”（信号线检查裕量）列中，会显示基于受惩罚的路径所得的新裕量。

在“SuggestionsID”（建议 ID）列中，会显示路径相关的 QoR 建议。对于不存在建议的项，应进行调查并且可能需要重新编码。如果存在建议，应用这些建议可能即可解决问题，而无需进行代码编辑。

“Clock Skew”（时钟偏差）部分用于报告与时钟偏差相关的项，如下图所示：

- 偏差值
- 源时钟和目标时钟名称
- 源时钟和目标时钟上的时钟根
- 不确定性

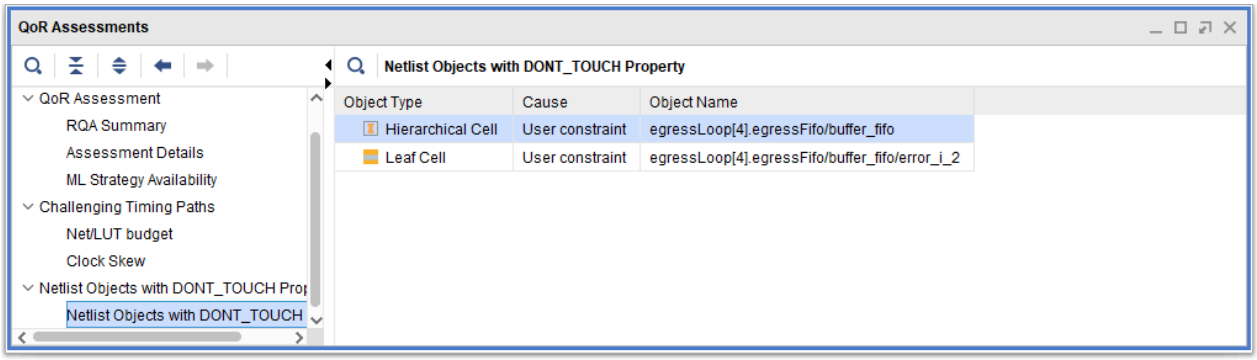
图 167： **“Clock Skew”报告**



含 DONT_TOUCH 属性的网表对象

含 DONT_TOUCH 属性的网表对象可能阻止执行有助于改善设计性能的最优化。RQA 报告的以下部分显示了设有 DONT_TOUCH 属性的对象：

图 168：**含 DONT_TOUCH 属性的网表对象**



RQA 报告显示了以下对象及其原因：

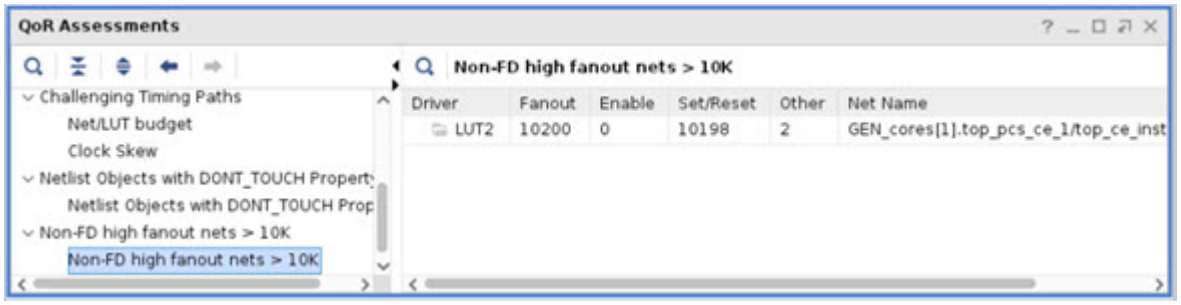
- 层级单元
- 叶节点单元
- 信号线

注释：DONT_TOUCH 属性可防止工具对路径进行最优化，利用 Vivado 自动使用其他属性的特性也可以添加该属性。如需移除 DONT_TOUCH 属性，请谨慎操作。例如，DFX 流程使用 `DONT_TOUCH` 来防止静态模块与可配置模块之间发生跨边界最优化，因此不应移除。相比之下，因 MARK_DEBUG 而添加的 DONT_TOUCH 属性对于流程并不重要，但它表示如果进行最优化，那么此信号不可用于硬件探测。

非 FD 高扇出信号线 > 10K

本节中报告的信号所含扇出大于 10K 但不受触发器驱动。驱动高扇出的信号线具有的驱动程序剖析信息应允许在不影响路径前时序的前提下执行复制。这样即可支持将复制的单元更自由地布局在负载旁。如有源约束，那么复制的单元的布局将聚集在一起，从而降低此技巧的有效性。

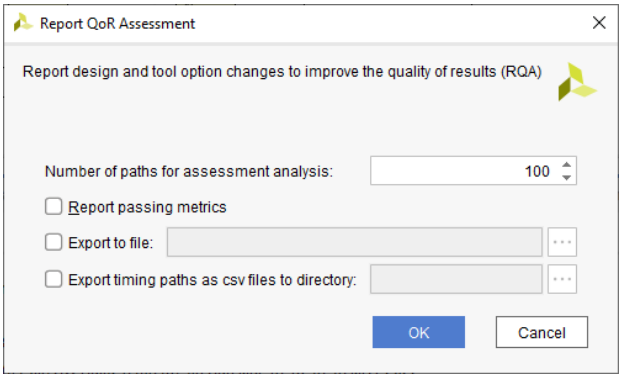
在详情部分中会捕获更多详细信息，如负载类型、驱动程序剖析信息和信号线名称等。但不捕获有关此信号的时序信息。如果该路径是伪路径，或者具有极低的要求，则可忽略。



生成 QoR 评估报告

在 Vivado IDE 中打开设计并单击“Reports” → “Report QoR Assessment”（报告 > QoR 评估报告）即可访问 `report_qor_assessment` 命令。

图 169：**“Report QoR Assessment”对话框**



Tcl 控制台中的等效命令如下：

```
report_qor_assessment -name report_qor_assessment
```

要更改时序路径限制的默认值 100，请更改“Number of paths for suggestion analysis”（建议分析的路径数），如下图所示。等效 Tcl 命令使用 `-max_paths <N>` 开关，其中 N 是整数：

```
report_qor_assessment -max_paths <N>
```

“Report passing metrics”（报告合格指标）选项会展开详情表，以显示已检查但合格的所有指标。等效 Tcl 命令使用 `-full_assessment_details` 开关：

```
report_qor_assessment -full_assessment_details
```

相比于文本报告，生成支持性 CSV 文件的选项有助于改善浏览大量数据的能力。要生成支持性 CSV 文件，请选中该框并指定目录。CSV 文件用于：

- 显示 RQA 评分中涉及的时序路径，以及是否有建议可用于帮助解决相关问题
- 显示包含 DONT_TOUCH 属性（可能阻碍优化）的单元和信号线

等效 Tcl 命令使用 `-csv_output_dir` 开关：

```
report_qor_assessment -csv_output_dir <directory>
```

其他命令选项

- `-exclude_methodology_checks`：默认情况下，方法论检查是必需的（或者，如果先前已运行此命令，则会从内部高速缓存中捕获其检查结果）。该选项允许完全省略方法论检查。如果先前未曾运行此命令，那么该选项可以节省编译时间。由于方法论检查在整个实现运行过程中大部分情况下处于静态，因此建议在实现运行的多个阶段中调用 `report_qor_assessment` 时使用该选项。

自动终止运行

在基于工程的实现运行中运行 QoR 评估功能时，会生成 1 到 5 之间的评分，用于指示设计达成时序收敛的可能性是较低还是较高。

对于预测发现质量较低的运行，也可以提前终止，相较于继续运行直至结束，这样可以提前清空服务器资源。在以下情况下会提前终止运行：

- 当 MIN_RQA_SCORE 运行属性置位时。例如，此项设为 3 时，所含 RQA 评分为 1 或 2 的设计将被终止。
- 完成 opt_design 后，在“Timing Closure Report Strategy”（时序收敛报告策略）中调用 report_qor_assessment 命令时。如需额外增加调用，建议采用定制报告策略。

在运行 `impl_1` 时将 MIN_RQA_SCORE 属性设置为 1 的示例如下所示：

```
set_property MIN_RQA_SCORE 3 [get_runs impl_1]
```

注释：该功能特性在非工程模式下并没有直接关联。基于脚本生成 RQA 评分的简单方法是调用 `report_qor_assessment`，然后调用 `report_design_analysis -qor_summary -json <filename>.json`。

Report QoR Suggestions

`report_qor_suggestions` 命令是处理 QoR 建议对象时使用的主要命令。QoR 建议对象会创建命令和属性来改善设计的时序性能（欲知详情，请参阅 QoR 建议）。

`report_qor_suggestions` 命令可执行两项任务：

1. 报告 QoR 建议对象
2. 生成新 QoR 建议对象

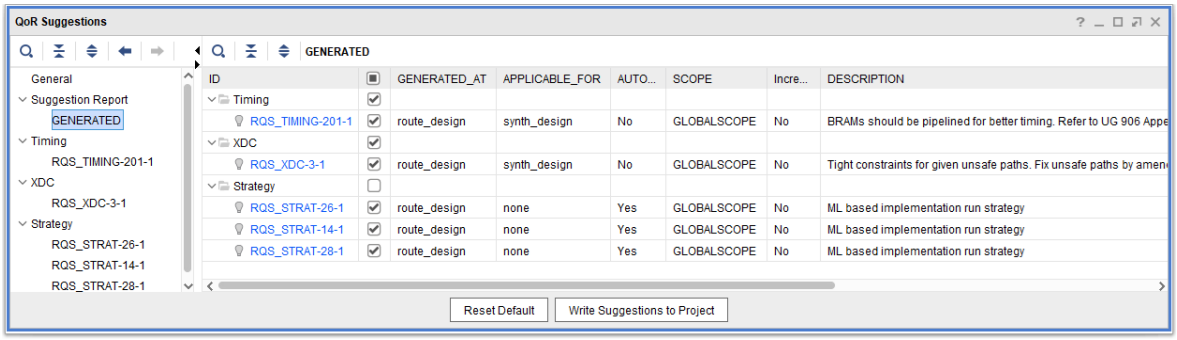
在综合后的任意阶段都能在设计上运行此命令。

此节中还提供了有关使用 `write_qor_suggestions` 命令从报告创建 RQS 文件的部分详细信息。

“QoR Suggestions”报告

“QoR Suggestions”（QoR 建议）报告拆分为汇总部分和详情部分，汇总部分中每项建议位于顶部，有关建议的详情则位于报告的下半部分。下表显示了所生成报告的示例。

图 170：**report_qor_suggestions 报告示例**



在此报告的“Suggestion Report”（建议报告）下提供了所有建议的列表。这些建议分为 4 个类别来呈现。这些建议按如下方式成对显示：

- “GENERATED”（生成的建议）和“EXISTING”（现有建议）：
  - “Generated”建议是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”（已应用的建议）和“FAILED TO APPLY”（应用失败的建议）：
  - “Applied”建议是已启用并且已通过 APPLICABLE_FOR 阶段的建议。这些建议均已得到成功应用。
  - “Failed to apply”建议已启用并且已通过 APPLICABLE_FOR 阶段，但尚未成功应用。请检验现有 log 日志文件，了解尚未应用这些建议的原因。“Applied”建议是已启用并且已通过 APPLICABLE_FOR 阶段的建议。这些建议均已得到成功应用。

报告的下半部分包含有关生成的建议的详细信息。它拆分为以下类别，`report_qor_suggestions` 根据这些类别来分析设计：

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

通过观察 GENERATED 建议可知，详情部分应提供充分的信息，以供您推断报告这些建议的原因。可通过“GENERATED”建议的详情部分进行交叉探测。以下交叉探测方法非常实用：

- 选中对象会在其他窗口（例如，“Device”视图）中高亮这些对象
- 按“F4”可显示选定对象的原理图
- 右键单击对象即可生成时序报告

通过观察 EXISTING 建议可知，可能对象已修改且不存在（例如，`opt_design` 可能从网表中移除对象）。因此，选择“EXISTING”建议时，交叉探测有时不可用。

对于每项建议，报告中包含额外的列，以提供有关如何使用这些建议的实用信息。下表显示了这些列的详细信息。

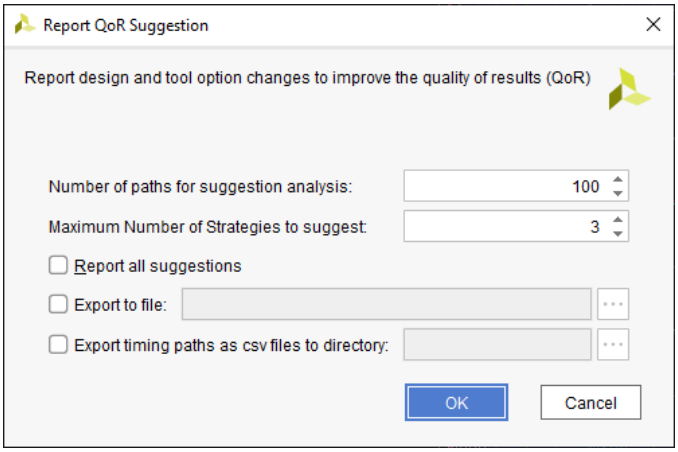
表 16：**附加信息列**

属性	值	描述
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	RQS 文件或 `current_run`（前提是在当前运行中生成这些建议）。	建议的来源。
AUTOMATIC	Yes 和 No	描述建议是由 Vivado 工具自动执行还是需手动执行。
SUGGESTION_SCOPE	GLOBALSCOPE，OOC 顶层模块	用于启用 OOC 综合以自动限定综合建议的作用域。

生成 QoR 建议报告

在 AMD Vivado™ IDE 中使用“Reports”（报告）下拉菜单中的“Report QoR Suggestions”（QoR 建议报告）即可访问 `report_qor_suggestions` 命令。

图 171：**“Report QoR Suggestions”对话框**



Tcl 控制台中的等效命令如下：

```
report_qor_suggestions -name qor_suggestions_1
```

要更改时序路径限制的默认值 100，请更改“Number of paths for suggestion analysis”（建议分析的路径数），如下图所示。这样即可增加建议数量，但这些建议仍将应用于尚未最优化的时序路径。等效的 Tcl 命令行选项如下：

```
-max_paths <N>
```

要更改生成的 ML 策略数量，请更改“Maximum Number of Strategies to suggest”（最大策略建议数），如以下对话框所示。等效的 Tcl 命令行选项如下：

```
-max_strategies <N>
```

要扩展分析以报告不违反阈值条件的建议，请选中“Report all suggestions”（报告所有建议）。行为如下：

- 时序建议：无论是否满足时序，都会提供有关时序路径的建议。
- 使用率建议：提供有关非关键资源的建议。
- 拥塞建议：提供有关在布线后阶段已满足时序的设计的建议。

等效的 Tcl 命令行选项如下：

```
-report_all_suggestions
```

要生成支持性 CSV 文件以显示失败的时序路径及其关联的建议，请选中此复选框并指定目录。CSV 文件能够大幅简化时序路径的浏览，较文本报告中的表格更便于管理。此外还会生成第二个文件，其中包含 DONT_TOUCH 报告。等效的 Tcl 命令行选项如下：

```
-csv_output_dir <directory>
```

注释：DONT_TOUCH 属性可防止工具对路径进行最优化，利用 Vivado 自动使用其他属性的特性也可以添加该属性。如需移除 DONT_TOUCH 属性，请谨慎操作。例如，DFX 流程使用 DONT_TOUCH 来防止静态模块与可配置模块之间发生跨边界最优化，因此不应移除。相比之下，因 MARK_DEBUG 而添加的 DONT_TOUCH 属性对于流程并不重要，但它表示如果进行最优化，那么此信号不可用于硬件探测。

其他命令选项

- `-of_objects <suggestion objects>`：启用特定建议的报告。在此模式下运行时，`report_qor_suggestions` 不会生成新建议。此命令可快速执行，读取 RQS 文件后，此命令可用于查看其中包含的建议。其使用示例如下所示：

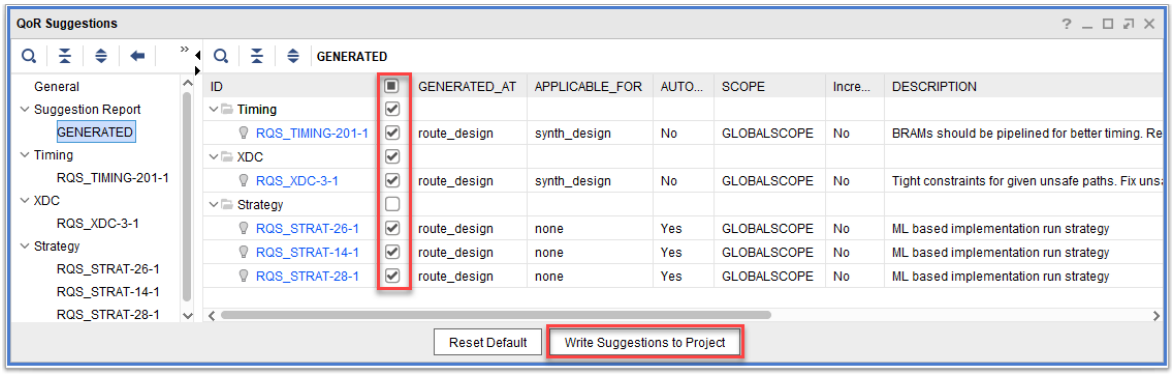
```
report_qor_suggestions -of_objects [get_qor_suggestions <objectNames>]
```

- `-cells <cellName>`：为执行的分析更改顶层单元。默认值为设计顶层。

写入建议对象文件

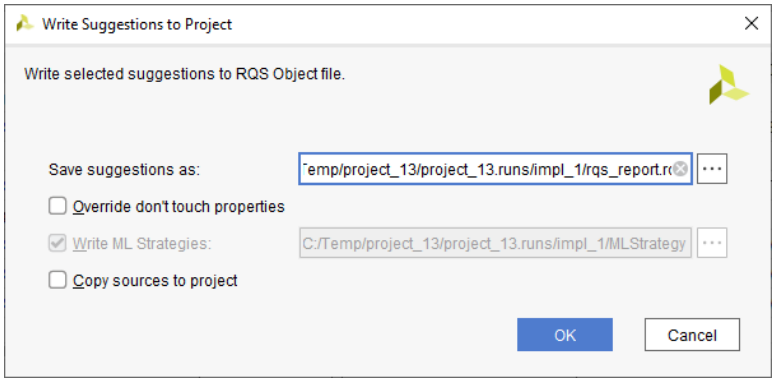
创建 QoR 建议报告后，必须生成包含建议的 RQS (`.rqs`) 文件，此文件可馈送到建议运行中。为此，请选中要包含在运行中的建议，然后写入 QoR 建议文件。如下图中所示。

图 172：**选择/写入建议**



下图显示了写入建议时的对话框。

图 173：**“Write Suggestions to Project”对话框**



等效的 Tcl 命令如下所示：

```
write_qor_suggestions filename.rqs
```

部分建议要求您确认必须覆盖 DONT_TOUCH 属性才能为其提供授权。等效的 Tcl 选项如下所示：

```
-disable_dont_touch
```

策略建议的处理方法与标准建议不同。写入 ML 策略允许创建多个策略 RQS 文件。如需了解更多详情，请参阅 策略建议。等效的 Tcl 选项如下所示：

```
-strategy_dir <directory>
```

Report Design Analysis

“Design Analysis”（设计分析）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“Report Design Analysis”

您可从 Tcl 控制台或 Vivado IDE 运行“Report Design Analysis”（设计分析报告）。“Report Design Analysis”可生成的报告分 3 种类别：

- “Timing”（时序）：用于报告时序路径的时序和物理特性。
- “Complexity”（复杂性）：用于分析设计的布线复杂性和 LUT 分布情况。
- “Congestion”（拥塞）：用于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”（报告 > 设计分析报告）。

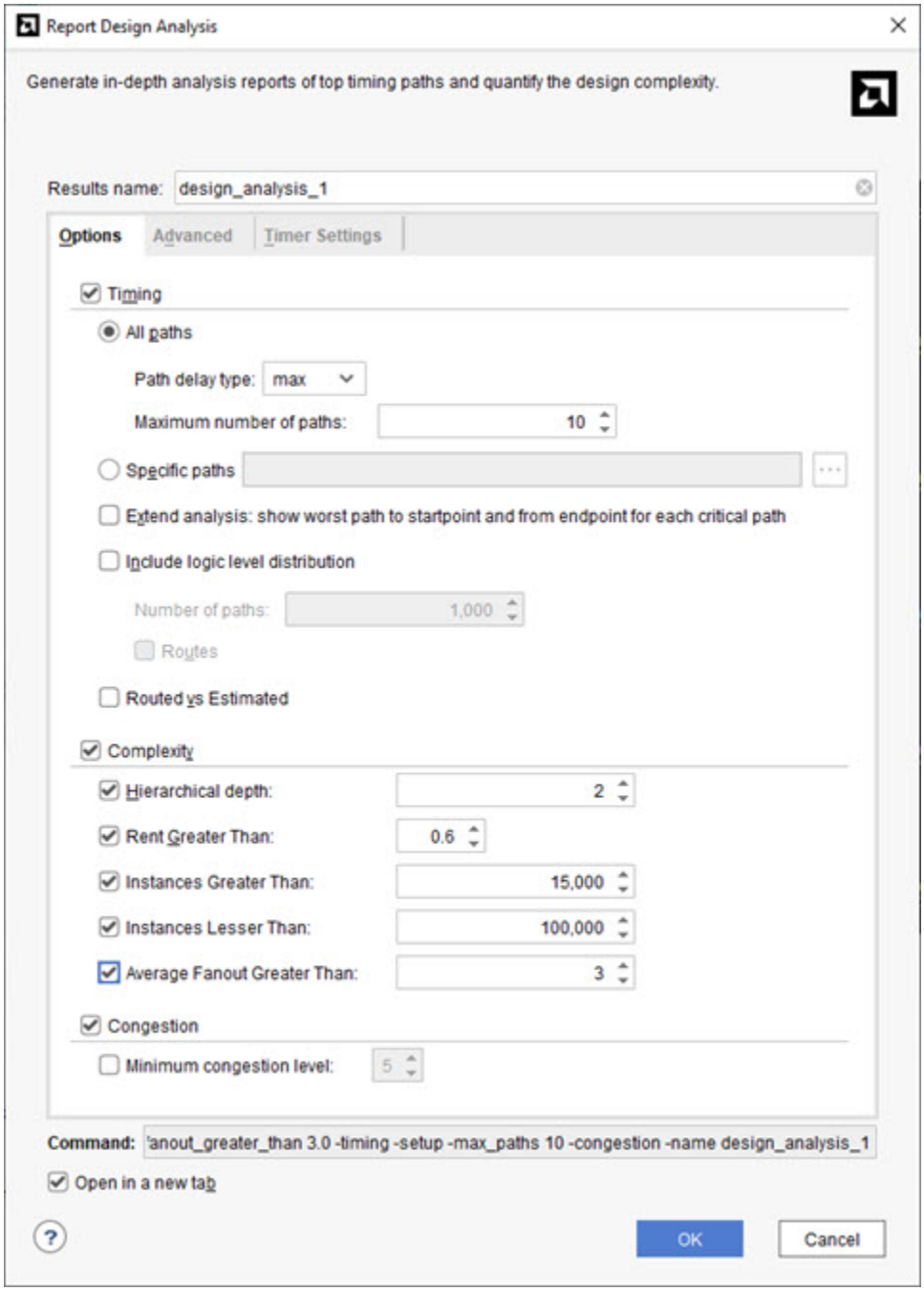
等效的 Tcl 命令：`report_design_analysis -name design_analysis_1`

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）中包含：

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

图 174：**“Report Design Analysis”对话框**



“Results Name”字段

在位于“Report Design Analysis”（设计分析报告）对话框顶部的“Results Name”（结果名称）字段中，指定报告的图形窗口的名称。

等效的 Tcl 选项：`-name <windowName>`

“Options”选项卡

在上图中所示的“Options”（选项）选项卡中，包含下列字段：

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

“Timing”字段

“Timing”（时序）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“All Paths”（所有路径）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“Specific Paths”（特定路径）选项，则将针对指定路径对象执行分析。单击右侧“Browse”（浏览）按钮，打开搜索对话框，协助找出路径对象。如需了解有关 `get_timing_paths` 的更多信息，请参阅《Vivado Design Suite Tcl 命令参考指南》(UG835) 中的 get_timing_paths。

等效的 Tcl 选项：`-of_timing_paths <args>`

选择“Extend Analysis”（扩展分析）选项，对感兴趣的每条路径执行扩展分析，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。

选中该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。如果要分析的是特定路径，则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>`

“Complexity”字段

“Complexity”（复杂性）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。这可作为布线拥塞的早期指示。这部分命令容易受编译时间影响，在大型设计上尤其如此。请参阅 复杂性报告。

等效的 Tcl 选项：`-complexity`

选中“Hierarchical Depth”（层级深度）选项时，可选择顶层单元之下层级中的多个层次以计算 Rent 系数。默认设为顶层，可通过在“Advanced”（高级）选项卡中指定 `-cells` 选项来调整为更低的层次。使用 cells 选项将减少所分析的层级数，并加速计算。

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

“Rent Greater Than”（Rent 下限）选项用于将报告中显示的信息局限于所含 Rent 值高于指定层级及其子级的模块。通常所含值低于该值的模块不会出现拥塞。默认值为 0.6。

等效的 Tcl 选项：`-rent_greater_than <arg>`

“Instances Greater Than”（实例数下限）选项用于限制将在其中执行 Rent 计算的层级单元的数量。通常，较小的模块即使所含 Rent 系数较高，也不会出现拥塞。增大该值通常会缩短命令的编译时间。默认值为 15000。

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

“Instances Lesser Than”（实例数上限）选项用于限制将在其中执行 Rent 计算的层级单元的数量。通常为了减少拥塞而执行的操作会增大面积，因此若要在层级单元上执行这些操作，这些单元大小最好不超过必要的大小。减小该值通常会缩短命令的编译时间。默认值为 100000。

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

“Average Fanout Greater Than”（平均扇出下限）选项会限制在其中执行 Rent 计算的层级单元的数量，方法是计算模块内信号的平均扇出，如果低于阈值，则不执行 Rent 计算。通常，通过将较高的 Rent 系数与较高的平均扇出相结合，即可估算得到最适合用于预测拥塞的模块。增大该值通常会缩短命令的编译时间。默认值为 3.0。

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

“Congestion”字段

“Congestion”（拥塞）字段用于切换 `-congestion` Tcl 的 ON（开启）和 OFF（关闭）。

选择“Minimum congestion level”（最低拥塞等级）选项，指定在设计中显示布线器拥塞状况的最低拥塞等级。如果不指定，默认最低拥塞等级为 5。该值的取值范围在 3 到 8 之间。

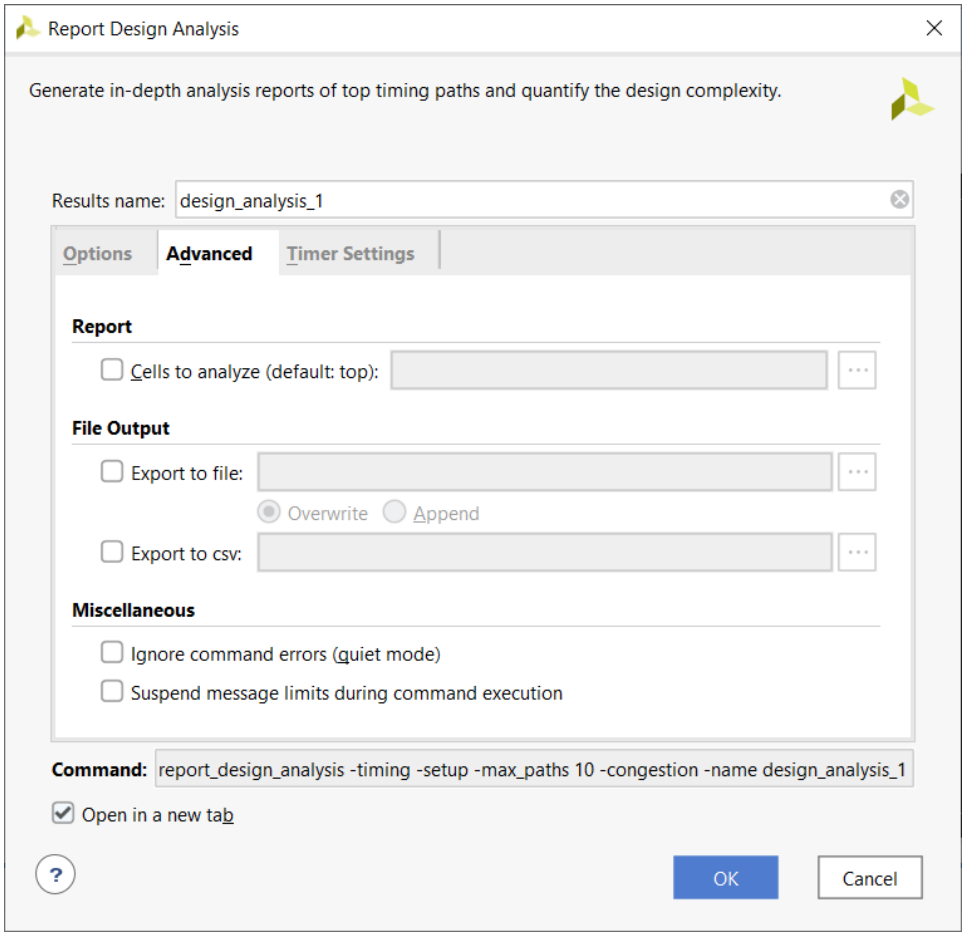
等效的 Tcl 选项：`-min_congestion_level <args>`

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。

“Advanced”选项卡

“Advanced”（高级）选项卡如下图所示。

图 175：**“Advanced”选项卡**



在“Advanced”选项卡中提供了以下字段：

- “Report”（报告）：选中“Advanced”选项卡中的“Cells to Analyze”（待分析的单元）选项即可指定要使用的分层单元。单击右侧“Browse”（浏览）按钮即可打开搜索对话框，并查找单元对象。该选项用于将时序和复杂性分析限制在报告范围内。
- “File Output”（文件输出）：生成 GUI 报告，并将结果写入文件。请选择“Export to file”（导出到文件）并在右侧字段中指定文件名。单击“Browse”（浏览）按钮可选择不同目录。

  等效的 Tcl 选项：`-file <arg>`

  选择“Overwrite”（覆盖）选项，即可用新的分析结果覆盖现有文件。

  选择“Append”（追加）可追加新结果。

  等效的 Tcl 选项：`-append`

- “Miscellaneous”（其他）：“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。

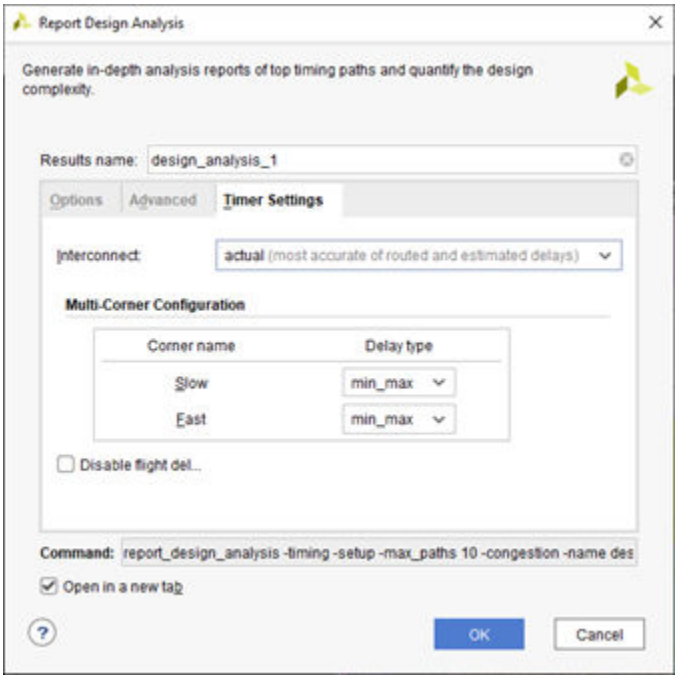
  等效的 Tcl 选项：`-quiet/-verbose`

“Timer Settings”选项卡

在下图所示“Timer Settings”（定时器设置）选项卡中，包含下列字段和选项。

- “Interconnect”选项
- “Multi-Corner Configuration”字段
- “Disable Flight Delays”选项

图 176： **“Timer Settings”选项卡**



“Interconnect”选项

您可选择时序路径分析中使用的互连模型：

- “actual”（实际值）：此模型为已布线的设计提供最准确的延迟。
- “estimated”（估算值）：此模型包含基于设计布局和连接到器件（实现前）的方式所估算的互连延迟。即使设计已完全布线，仍可指定估算延迟。
- “none”（无）：时序分析中不包含互连延迟。仅应用逻辑延迟。这有助于识别路径中逻辑延迟超出或者占用大量时序路径要求的区域。

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)。

“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)。

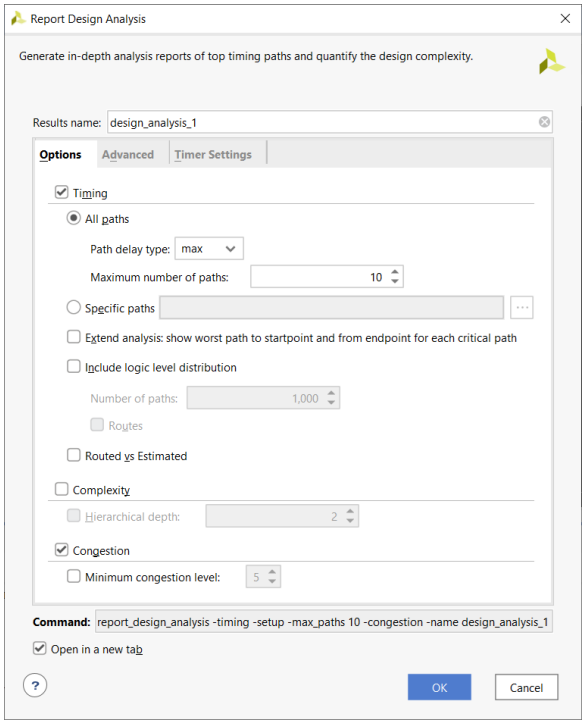
“Disable Flight Delays”选项

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 177：**“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。

- `csv <filename>.csv`：用于搭配时序路径选项生成 CSV 文件。对大量路径进行排序时，该选项很有用。
- `-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的并列对比结果。报告中的“Timing Category”（时序类别）中的某些字段带有“Estimated”或“Routed”前缀以便比较。
- `-return_timing_paths`：返回时序路径对象，以支持对位于指定时钟域内的特定逻辑层次的路径进行进一步分析。必须同时搭配 `-end_point_clock` 选项和 `-logic_levels` 选项使用。
- `-end_point_clock <arg>`：用于将逻辑层次分布报告部分限制为具有指定端点时钟的时序路径。
- `-logic_levels <arg>`：该选项可限制逻辑层次发送到逻辑层次直方图算法的时序路径。只能指定单个值。

- `-min_level <arg>`：该选项可将所含逻辑级数或布线数量小于指定值的所有时序路径都组合到单个分箱内。指定 <arg> 值后，必须至少为其传递值 1。
- `-max_level <arg>`：该选项可将所含逻辑级数或布线数量大于指定值的所有时序路径都组合到单个分箱内。指定 <arg> 值后，该值必须大于 -min_level 的值。

以下复杂性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界框的复杂性分析。例如：

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

注释：在左括号“`{`”与边界框起始位置之间需添加 1 个空格，如以上示例中所示。

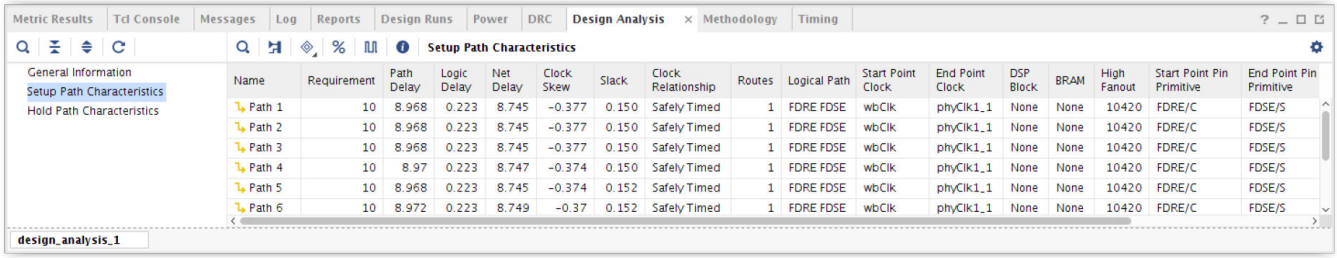
时序路径特性报告

下图显示了在“Timing Mode”（时序模式）下运行“Report Design Analysis”（设计分析报告）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。在 Vivado IDE 中选中“Reports”→“Report Design Analysis”（报告 > 设计分析报告）或者使用以下 Tcl 命令来生成报告：

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

提示：要创建保持路径特性报告，请在“Report Design Analysis”对话框的“Options”选项卡中选中“Path delay type: min”（路径延迟类型：最小值），或将 `-hold` 添加到 Tcl 命令中。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)。

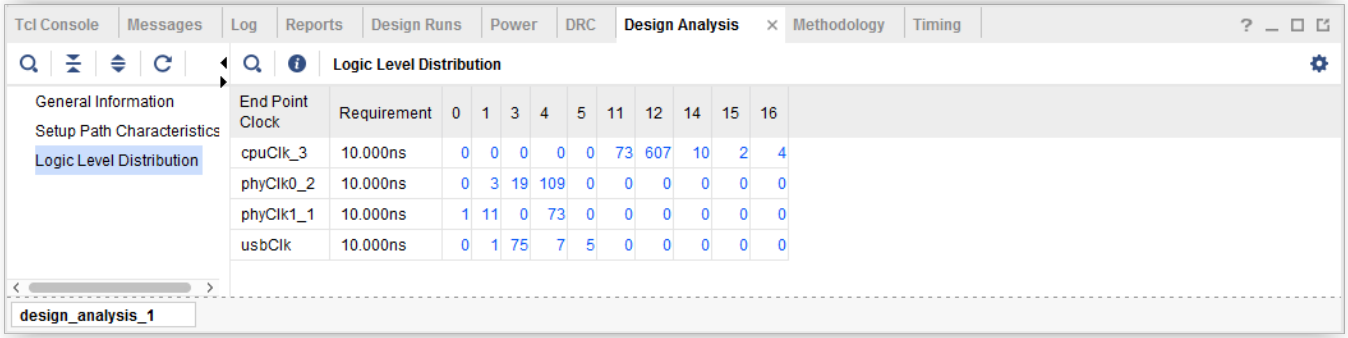
图 178：**建立路径特性示例**



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“Logic Level Distribution”（逻辑层次分布）表。针对“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，生成该表需选择“Report Design Analysis”对话框“Options”选项卡中的“Include logic level distribution”（包含逻辑层次分布）。“Logic Level Distribution”表的示例如下图所示。

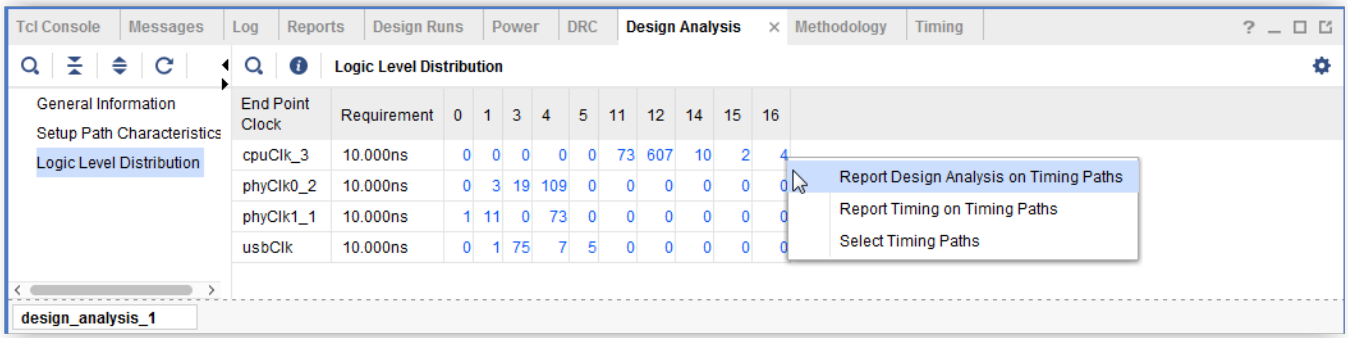
图 179：**“Logic Level Distribution”报告示例**



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。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。

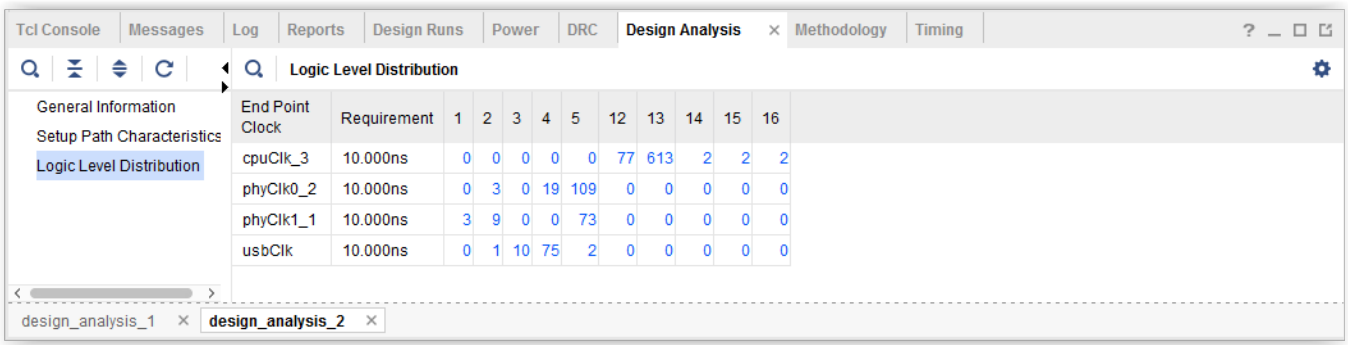
图 180：**选定路径的“Report Design Analysis”**



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

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 181：**使用 -routes 的“Logic Level Distribution”报告示例**



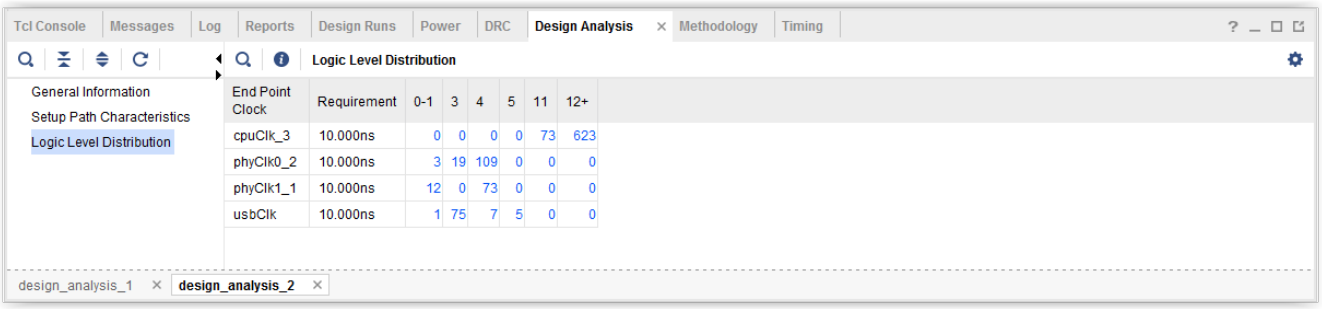
End Point Clock	Requirement	1	2	3	4	5	12	13	14	15	16
cpuClk_3	10.000ns	0	0	0	0	0	77	613	2	2	2
phyClk0_2	10.000ns	0	3	0	19	109	0	0	0	0	0
phyClk1_1	10.000ns	3	9	0	0	73	0	0	0	0	0
usbClk	10.000ns	0	1	10	75	2	0	0	0	0	0

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` 会分别使用以下 bin 文件生成报告：0-2、3、4、5、11 和 12+。

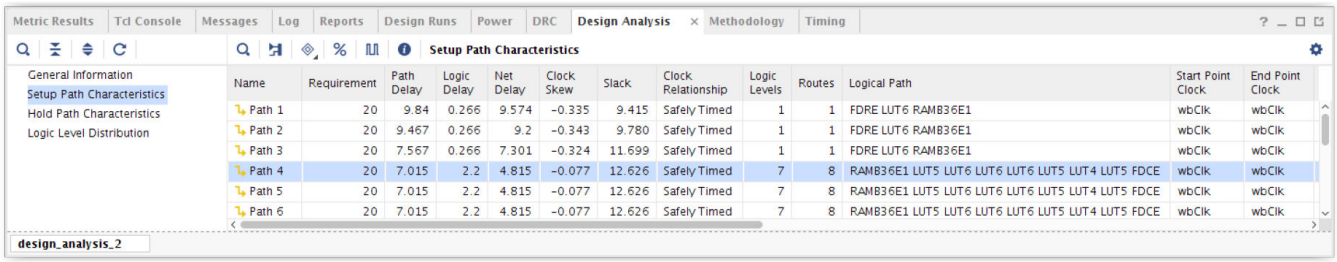
图 182：**使用 -min_level 和 -max_level 的“Logic Level Distribution”报告示例**



分析特定路径

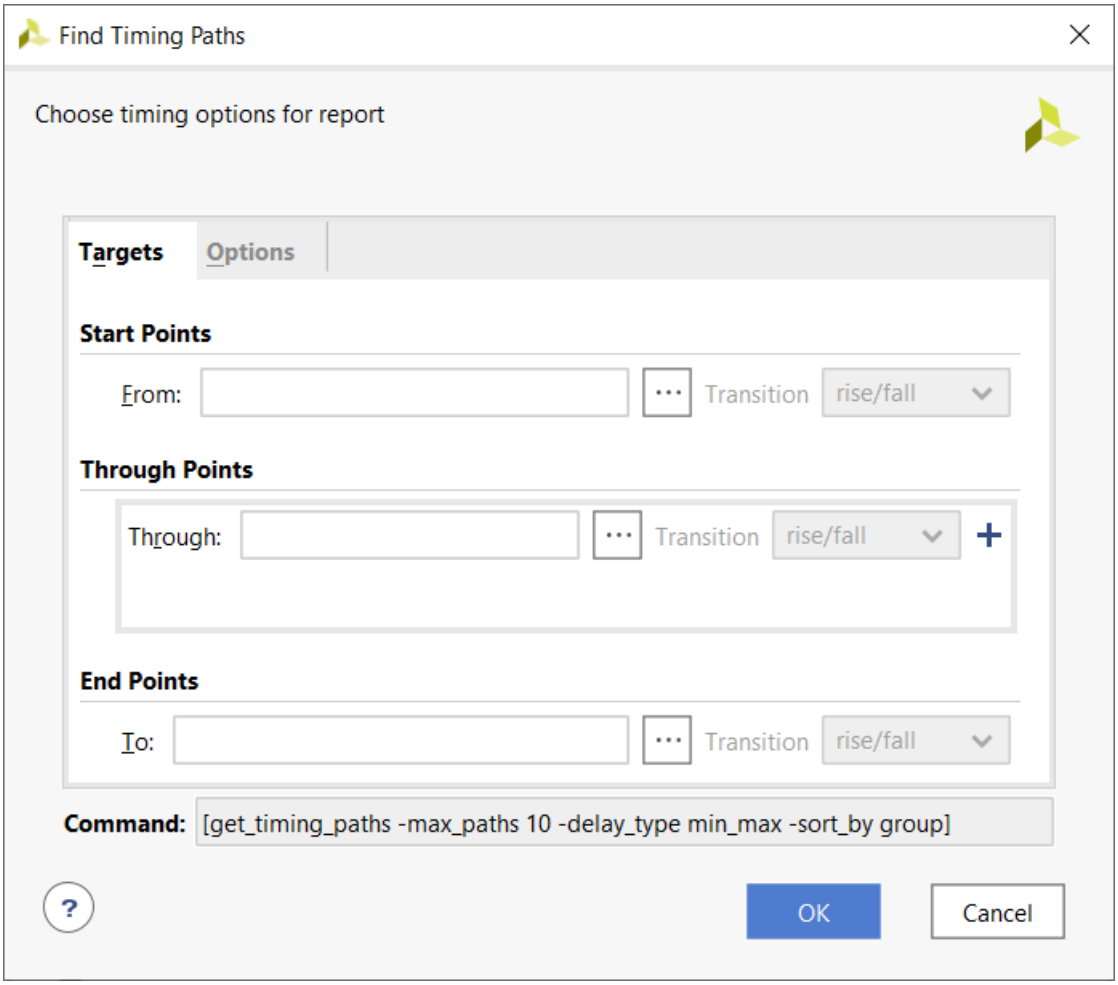
分析特定路径 显示了来自“Report Design Analysis”（设计分析报告）的报告示例，此示例采用“Timing Mode”（时序模式）并选定特定路径。

图 183：**特定时序路径特性示例**



在本例中，“Path Characteristics”（路径特性）和“Logic Level Distribution”（逻辑层次分布）表（如果已选中）已限制为指定路径。要指定路径，在“Report Design Analysis”对话框“Specific paths”（指定路径）选择部分单击右侧的“Browse”（浏览）按钮。这样会打开“Find Timing Paths”（查找时序路径）对话框（如下图所示）。

图 184：**“Find Timing Paths”对话框**



结合前后最差路径执行最差路径分析

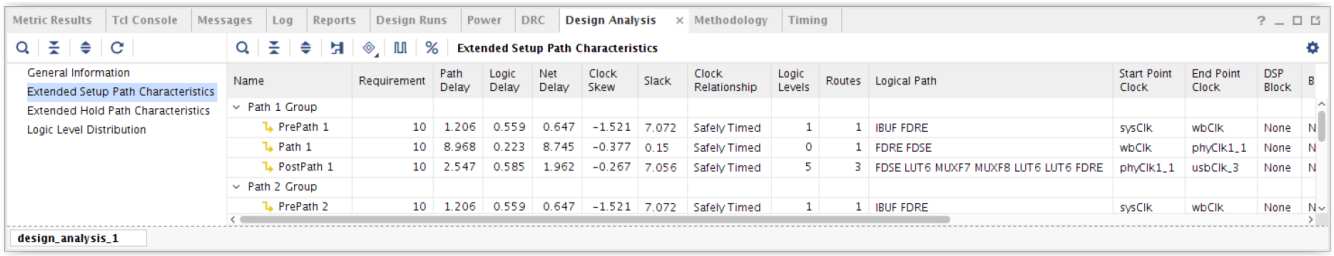
下图显示了“Timing Mode”（时序模式）下已选中“Extend analysis”（扩展分析）选项情况下的“Report Design Analysis”（设计分析报告）的报告示例。

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

“Path Characteristics”（报告特性）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 185：**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
Path 1 Group														
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 MUXF7 MUXF8 LUT6 LUT6 FDRE	phyClk1_1	usbClk_3	None	N
Path 2 Group														
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

- “Timing Analysis”（时序分析）：“Path Type”（路径类型）和“Requirement”（要求）详列了时序分析类型（SETUP 或 HOLD）和时序路径要求。“Slack”（裕量）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。“Timing Exception”（时序例外）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 1 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。
- “Datapath”（数据路径）：“Path Delay”（路径延迟）、“Logic Delay”（逻辑延迟）和“Net Delay”（信号线延迟）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“High Fanout”（高扇出）和“Cumulative Fanout”（累积扇出）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 系列和 UltraScale 器件 LUT 延迟报告的差异。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进行报告。因此，7 系列器件的 `Net Delay/Logic Delay` 比值大于 UltraScale 器件的比值。

- “Clocks”（时钟）：有关时序路径时钟的“Start Point Clock”（起点时钟）、“End Point Clock”（端点时钟）、“Clock Relationship”（时钟关系）和“Clock Skew”（时钟偏差）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“Clock Relationship”（时钟关系）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“Safely Timed”（已安全定时）。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“No Common Primary Clock”（无公共基准时钟）、“No Common Period”（无公共周期）、“No Common Node”（无公共节点）或“No Common Phase”（无公共相位）。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，请检查级联时钟缓冲器的时钟树结构。在 7 系列器件中，请检查源时钟和目标时钟的不同时钟缓冲器类型。在 UltraScale 器件中，可能需要检查时钟信号线的布局和布线，因为这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“Timing Characteristics”（时序特性）。

第 2 类：逻辑

- 路径：“Start Point Pin Primitive”（起点管脚原语）、“End Point Pin Primitive”（端点管脚原语）、“Start Point Pin”（起点管脚）、“End Point Pin”（端点管脚）、“Logic Levels”（逻辑层次）、“Logical Path”（逻辑路径）和“Routes”（布线）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。

  请检查端点管脚（例如，`CLR`、`PRE`、`RST` 和 `CE`），这些管脚可能包含在控制信号的高扇出信号线（例如，异步复位和时钟使能信号）中。另请检查单元类型，因为部分原语（如块 RAM 和 DSP）具有比其他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。您可使用此信息来快速检查造成大量逻辑层次的主要原因是 `LUT` 还是 `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互连结构。

  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，或者使用其他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。如果路径来自不含输出寄存器且含有数个逻辑层次的 RAMB 或 DSP，那么在此类路径上满足时序满足时序则更为困难。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 Crossings”（IO 交汇）和“SLR Crossings”（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导致了特定模块内多个实现运行之间发生时序问题，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：过度的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 间采用流水打拍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：
  - “High Fanout”（高扇出）用于显示数据路径中所有信号线的最高扇出，而“Cumulative Fanout”（累积扇出）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。由于依靠布局来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：`phys_opt_design -force_replication_on_nets <netName>`
  - 如果“Hold Fix Detour”（保持修复绕行）断言有效，则表示数据路径上布线已延迟，以满足路径保持时间要求。如果路径无法满足建立时间要求，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）。

    注释：在布线器中的特定保持时间修复阶段，会将 `HOLD_DETOUR` 属性置位。其他阶段可能会影响保持时序，但该属性将不予置位；例如，如果由于拥塞而导致存在布线绕行。

第 4 类：属性

- LUT 组合：“Combined LUT Pairs”表示路径中存在成对组合的 LUT。虽然成对组合的 LUT 可降低逻辑使用率，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”（标记调试）和“Dont Touch”（禁止触碰）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。
  - 默认情况下，设置 `MARK_DEBUG` 属性会同时设置 `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：“Fixed Loc, Fixed Route”（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他路径共享信号线，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。请务必谨记，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：Dynamic Function eXchange 设计

对于 Dynamic Function eXchange (DFX) 设计：逻辑路径中的每个单元都带有前缀，用于识别此单元是属于可重配置分区 (RP#:) 还是属于设计的静态区域 (S:)。

- “DFX Path Type”（DFX 路径类型）：用于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。
- “DFX Boundary Nets”（DFX 边界信号线）：用于报告静态区域或可重配置模块 (RM) 之间或者网表中两个 RM 之间的时序路径的交汇次数。
- “Boundary Fanout”（边界扇出）：用于报告位于 PPLOC 的边界路径扇出到其下游负载的行为。

复杂性报告

复杂性报告 (Complexity Report) 可显示顶层设计和/或层级单元的各类型叶节点单元的 Rent 指数 (Rent Exponent)、平均扇出 (Average Fanout) 以及分布方式。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，并且这些分组与其他分组同样连接紧密。这通常可理解为布线资源使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“Complexity Mode”（复杂性模式）来运行“Report Design Analysis”（设计分析报告）：

- 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 17：**在“Complexity Mode”下运行“Report Design Analysis”的选项**

Tcl 选项	描述
`-complexity`	强制以“Complexity Mode”来运行报告。
`-cells <arg>`	限制所考量的层级单元的范围，仅限位于指定层级中指定层次下的单元。
`-hierarchical_depth <arg>`	指定层级中的层次数，这些指定的层次均位于要报告的合格单元之下。
`-av_fanout_greater_than`	某一层级单元内的信号线的最小平均扇出。
`-instances_greater_than`	某一层级单元必须包含的最小实例数。
`-instances_lesser_than`	某一层级单元必须包含的最大实例数。
`-rent_greater_than`	仅报告超出指定 Rent 值得层级单元。

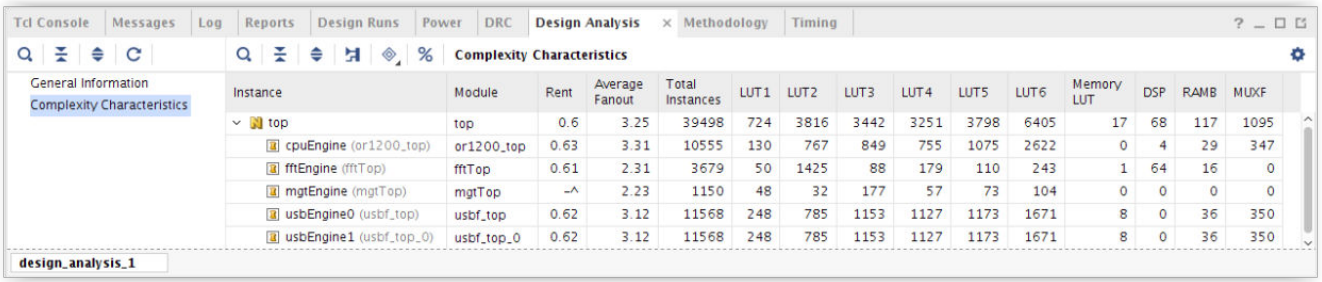
分析顶层设计复杂性

下图显示的示例报告来自“Complexity Mode”（复杂性模式）下的“Report Design Analysis”（设计分析报告），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 186：**顶层和层级深度为 1 的层级的复杂性分析**



Instance	Module	Rent	Average Fanout	Total Instances	LUT1	LUT2	LUT3	LUT4	LUT5	LUT6	Memory LUT	DSP	RAMB	MUXF
top	top	0.6	3.25	39498	724	3816	3442	3251	3798	6405	17	68	117	1095
cpuEngine (or1200_top)	or1200_top	0.63	3.31	10555	130	767	849	755	1075	2622	0	4	29	347
fftEngine (fftTop)	fftTop	0.61	2.31	3679	50	1425	88	179	110	243	1	64	16	0
mgtEngine (mgtTop)	mgtTop	-^	2.23	1150	48	32	177	57	73	104	0	0	0	0
usbEngine0 (usbf_top)	usbf_top	0.62	3.12	11568	248	785	1153	1127	1173	1671	8	0	36	350
usbEngine1 (usbf_top_0)	usbf_top_0	0.62	3.12	11568	248	785	1153	1127	1173	1671	8	0	36	350

注释：可通过右键单击某一表格标题来启用列筛选，以简化报告的导航。如果报告的层级数量过高，那么可组合使用列筛选功能来缩小报告范围。

读取和解释复杂性报告

前例所示的“Complexity Characteristics”（复杂性特性）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“Total Instances”（实例总数）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他因素还会限制布局解空间，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”表时需要考虑的其他事项：

- 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。
- 如果分层实例所含 Rent 指数较低，那么与其他因素（例如，布局时有大量数据流流经该模块）相结合就可能导致拥塞。

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优化，这样可减少 LUT6 的使用，从而降低网表密度。
- 综合时，减少 LUT 输入量。
- 尝试在综合中禁用 LUT 组合。
- 在实现期间使用“Congestion Strategy”（拥塞策略）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。
- 利用 QoR 建议和智能设计运行来自动解决拥塞问题。

拥塞报告

“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“Congestion Mode”（拥塞模式）下运行“Report Design Analysis”（设计分析报告），必须在“Report Design Analysis”对话框的“Options”（选项）选项卡中指定“Congestion”（拥塞）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：

- Placer Final Congestion Reporting
- “Router Initial Congestion”报告
- “SLR Net Crossing”报告

“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等级的所有窗口。其中各列具体定义如下：

- “Direction”（方向）：发生拥塞的资源的方向，分为：North（北）、South（南）、West（西）或 East（东）。
- “Congestion Level”（拥塞等级）：CLB 拼块中的最大拥塞等级。
- “Congestion”（拥塞）：指示定义的窗口中估算的布线资源使用率。该值可大于 100%。
- “Congestion Window”（拥塞窗口）：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。

- “Cell Names”（单元名称）：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。

- Avg LUT Input：这是窗口内 LUT 的平均 LUT 输入。

- COMBINED LUTs %：指示窗口内组合 LUT 的百分比。
- LUT usage %：窗口内 LUT 使用率百分比。
- LUTRAM usage %：窗口内 LUTRAM 使用率百分比。
- Flop usage %：窗口内 FD（含 LD）使用率百分比。
- MUX usage %：窗口内 MUXF 使用率百分比。
- RAMB usage %：窗口内 RAMB 使用率百分比。
- URAM usage %：窗口内 URAM 使用率百分比。
- DSP usage %：窗口内 DSP 使用率百分比。
- CARRY usage %：窗口内 CARRY 使用率百分比。
- SRL usage %：窗口内 SRL 使用率百分比。

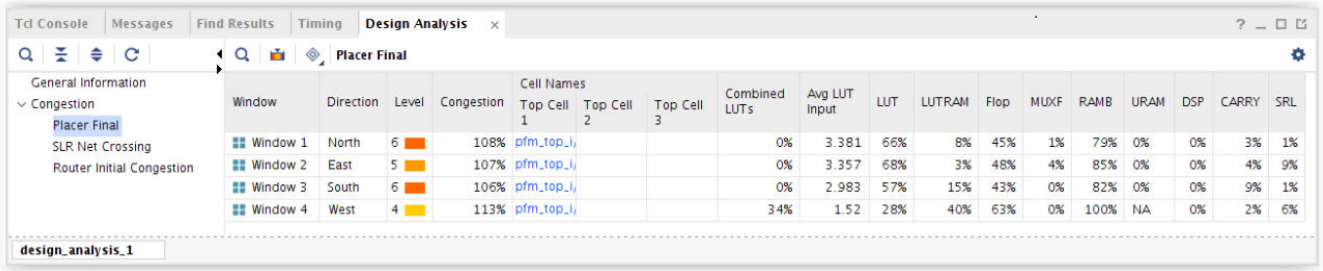
Placer Final Congestion Reporting

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

- 如果 LUT 使用率较高，请检验实例的“Complexity”（复杂性）报告中是否存在较高百分比的 LUT6。
- 如果拥塞区域中 RAMB 或 DSP 使用率较高，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。资源使用率 (%) 表明了位于拥塞区域内的资源类型。

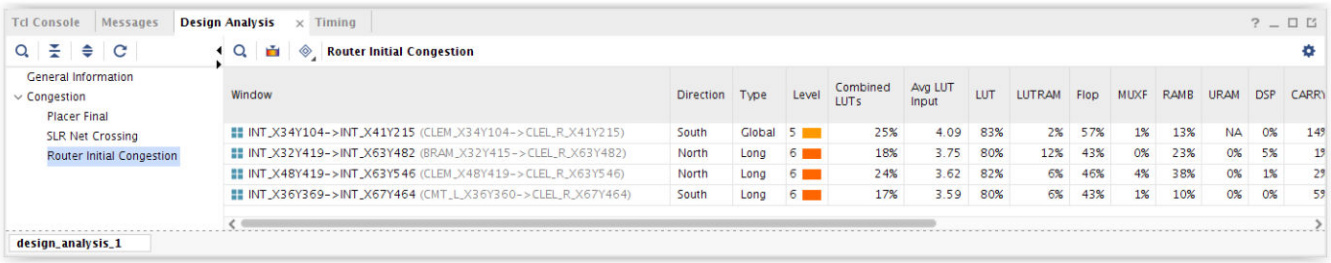
图 187：**“Placer Final Congestion Reporting”表示例**



“Router Initial Congestion”报告

仅当已运行布线器时，“Router Initial Congestion”（布线器初始拥塞）才可用，对于 7 系列 FPGA，该报告名为“Initial Estimated Router Congestion”（初始估算布线器拥塞）。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 188：**“Router Initial Congestion”报告表格示例**



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使用率。
- 短拥塞则考虑给定方向的所有其他互连使用率。

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使用率增加，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：

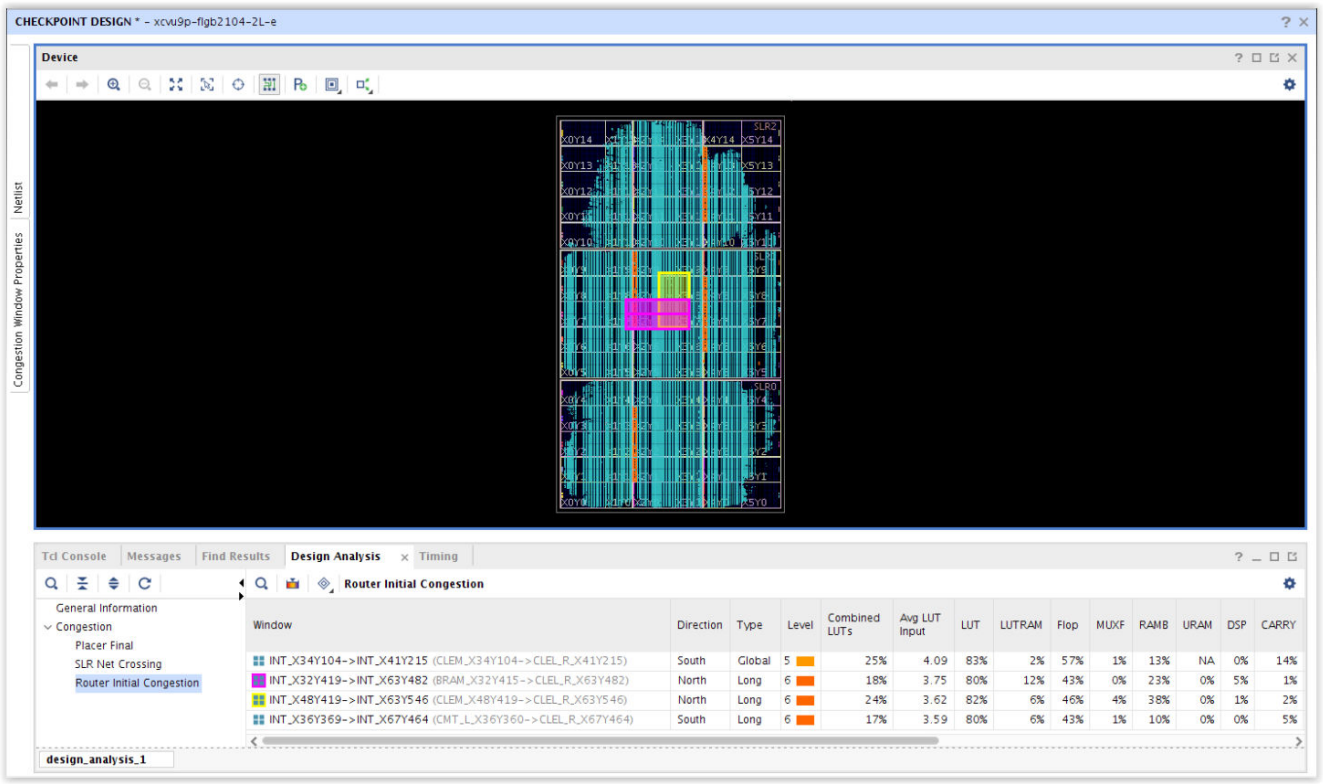
- 当拥塞等级大于 6 时，设计满足时序要求的概率很低，且可能在布线期间失败。
- 当拥塞等级为 4 或 5 时，请识别位于拥塞区域内的模块。您可对这些模块应用拥塞缓解技巧，或者使用其他指令（例如，*SpreadLogic*）重新运行布局器。
- 当拥塞等级不超过 3 时，拥塞可能不会导致问题，除非设计的时序预算非常紧凑。

在上图所示的“Router Initial Congestion”示例中，报告的区域的拥塞等级不小于 5。要以更低的拥塞阈值生成拥塞报告，请使用 `-min_congestion_level` 开关。默认最低拥塞等级为 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他区域（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 189： **“Router Initial Congestion”报告表格示例**

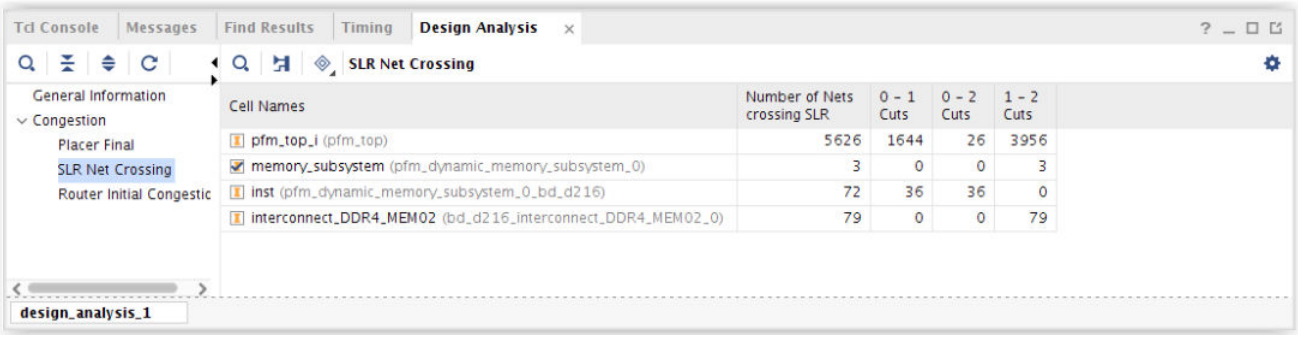


“SLR Net Crossing”报告

“SLR Net Crossing”（SLR 信号线交汇）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 190： **“SLR Net Crossing”报告表示例**



分析设计的“SLR Net Crossing”报告表的“Congestion”（拥塞）和“Timing QoR”（时序 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. 如果在使用各布局指令执行多轮实现运行期间，跨越 SLR 的特定模块持续遇到时序问题，请尝试减少使用 Pblock 以将模块约束到单一 SLR。

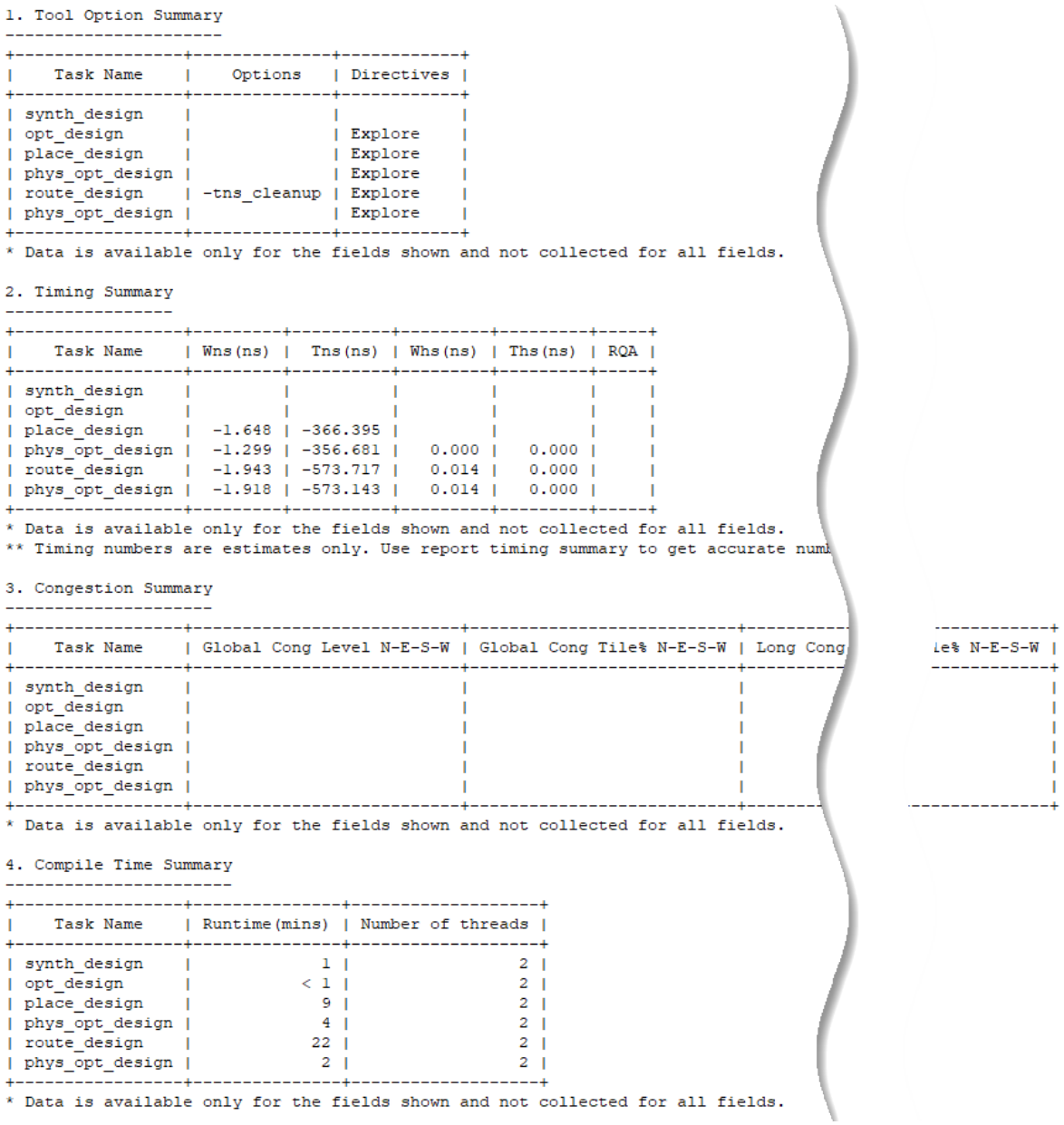
设计 QoR 汇总

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。该选项可按两种格式生成：基于文本的报告或 JSON 格式。

要生成基于文本的格式，请运行以下命令：

```
report_design_analysis -qor_summary
```

图 191：**设计分析 QoR 汇总报告**



1. Tool Option Summary

Task Name	Options	Directives
synth_design		
opt_design		Explore
place_design		Explore
phys_opt_design		Explore
route_design	-tns_cleanup	Explore
phys_opt_design		Explore

* Data is available only for the fields shown and not collected for all fields.

2. Timing Summary

Task Name	Wns(ns)	Tns(ns)	Whs(ns)	Ths(ns)	RQA
synth_design					
opt_design					
place_design	-1.648	-366.395			
phys_opt_design	-1.299	-356.681	0.000	0.000	
route_design	-1.943	-573.717	0.014	0.000	
phys_opt_design	-1.918	-573.143	0.014	0.000	

* Data is available only for the fields shown and not collected for all fields.
** Timing numbers are estimates only. Use report timing summary to get accurate numb

3. Congestion Summary

Task Name	Global Cong Level N-E-S-W	Global Cong Tile% N-E-S-W	Long Cong	le% N-E-S-W
synth_design				
opt_design				
place_design				
phys_opt_design				
route_design				
phys_opt_design				

* Data is available only for the fields shown and not collected for all fields.

4. Compile Time Summary

Task Name	Runtime(mins)	Number of threads
synth_design	1	2
opt_design	< 1	2
place_design	9	2
phys_opt_design	4	2
route_design	22	2
phys_opt_design	2	2

* Data is available only for the fields shown and not collected for all fields.

此汇总报告也可以 JSON 格式生成。采用 JSON 格式时，该表作为单个表格平铺展示，以便于解析。此报告中仅限这部分能按此格式输出，但它更便于对部分关键设计指标进行解析。要以 JSON 格式生成汇总报告，请运行以下命令：

```
report_design_analysis -qor_summary -json <json filename>
```

使用“Report Design Analysis”

常见 QoR 问题主要分为 2 类：

- 时序违例

- 拥塞

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，但您还必须复查其他关键路径，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“Setup Path Characteristics”（建立路径特性）表格示例。

图 192：**建立路径特性**

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 FDPE
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）
  - 逻辑层次是否过多？（逻辑级数）
  - 是否存在阻碍逻辑最优化的任何约束或属性？（勿触，标记调试）
  - 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）
- 高信号线延迟百分比（信号线延迟）
  - 在路径中是否有任何高扇出信号线？（高扇出，累积扇出）
  - 分配给多个 Pblock 的单元布局能否拉开距离？(Pblocks)
  - 单元布局能否拉开距离？（边界框大小，时钟区域距离）
  - 对于 SSI 器件，是否存在跨 SLR 边界的信号线？（SLR 交汇）
  - 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 部分。
- RAMB 或 DSP 单元中缺少流水线寄存器（而路径中存在此寄存器）
  - 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）
  - 此路径是时钟域交汇路径吗？（起点时钟，端点时钟）
  - 时钟是同步时钟还是异步时钟？（时钟关系）

- 此路径是否跨多个 I/O 列？（IO 交汇）

为了在 AMD 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“Logic Level Distribution”（逻辑层次分布）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 193：**“Logic Level Distribution”表**

2. Logic Level Distribution

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	0	0	0	0	0	0	0	0	0	0	285	24	0	0	0
phyClk0_2	8.000ns	0	7	140	138	8	198	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
usbClk_3	8.000ns	0	10	1	2	13	4	8	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

`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 194：**估算的拥塞表**

1. Placer Final Level Congestion Reporting

Direction	Type	Level	Window	Combined LUTs	Avg LUT Input	LUT	LUTRAM	Flop	RAMB	URAM	DSP	LOOKAHEAD8	SRL	Cell Names
West	Short	5	(CLE_W_CORE_X63Y189,CLE_E_CORE_X80Y220)	12%	3.288	92%	0%	8%	0%	NA	NA	37%	0%	top(100%)

2. Router Initial Congestion

Direction	Type	Level	Window	Combined LUTs	Avg LUT Input	LUT	LUTRAM	Flop	RAMB	URAM	DSP	LOOKAHEAD8	SRL	Cell Names
South	Short	5	(CLE_E_CORE_X64Y152,CLE_E_CORE_X87Y199)	14%	3.468	93%	0%	8%	0%	0%	NA	35%	0%	top(100%)
South	Short	5	(CLE_E_CORE_X64Y160,CLE_E_CORE_X79Y191)	13%	3.509	94%	0%	7%	0%	NA	NA	36%	0%	top(100%)

针对“Module Names”（模块名称）所提供的名称对应于每个报告的 tile（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。

第 6 章

综合分析与收敛技巧

本章讲解了如何使用细化视图来分析综合结果，并提供了示例。

使用细化视图对 RTL 进行优化

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他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。如需了解有关“Linter”工具的更多信息，请参阅《Vivado Design Suite 用户指南：综合》(UG901)。

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并不容易，因为它取决于您对设计的了解以及您对总体 RTL 优化的了解程度。观察综合后经过优化的网表并确定 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 可在综合后或实现后的设计与细化后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. 打开存储器中综合后/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”键显示其板级原理图。
3. 选择 Flow Navigator 窗格中的“Elaborated Design”（细化设计）。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 板级原理图，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

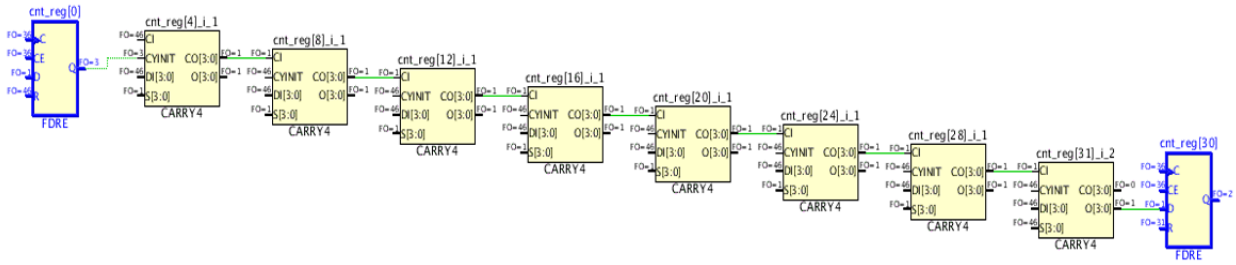
以下示例是一个计数器：

图 195：**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;
```

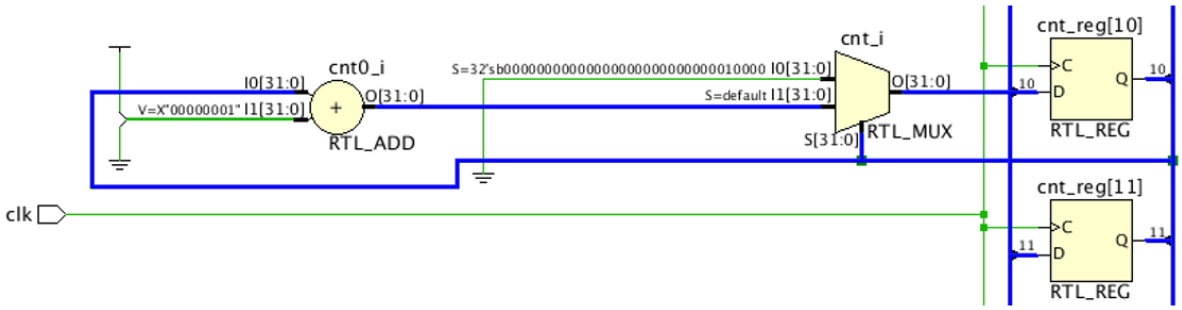
信号 `cnt` 按从 0 到 16 进行计数，需 5 位矢量用于编码。布线后关键路径的板级原理图如下图所示。端点为 `cnt` 信号的第 30 位。

图 196：**cnt 计数器布线后关键路径的板级原理图**



选中关键路径的起点和端点单元后，只需打开选定单元的板级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 197：**细化视图下的 cnt 计数器**



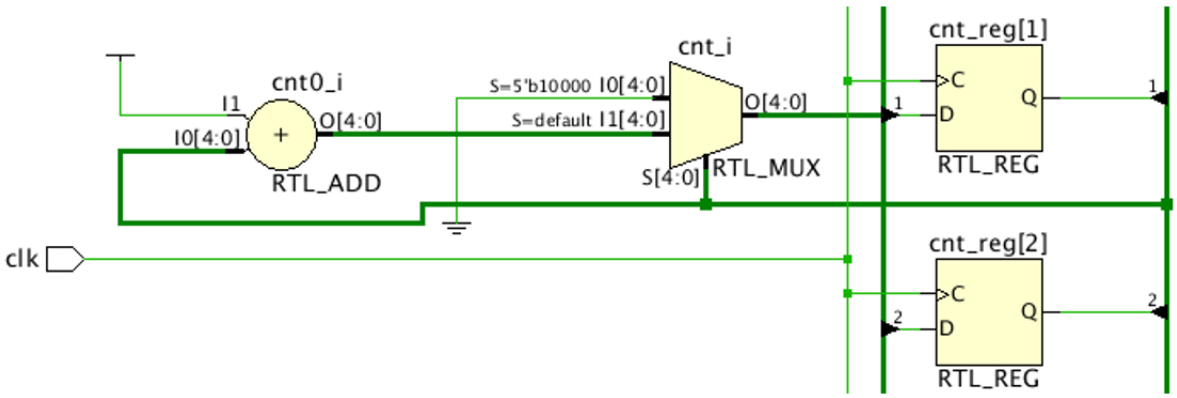
细化视图显示加法器输入大小限制为 32 位，因为信号 `cnt` 已声明为整数。在此特定示例中，在整个综合优化过程中均保持使用 32 位运算符。细化视图可明示所发生的情况，您可按如下方式更改 RTL 来获得进一步优化的网表和时序 QoR。随着计数器从 0 递增至 16，您可为信号 `cnt` 定义范围，以强制将加法器输入宽度设置为 5 位而不是 32 位。

图 198：**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会影响综合优化，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 199：**改进 RTL 后细化视图下的 cnt 计数器**

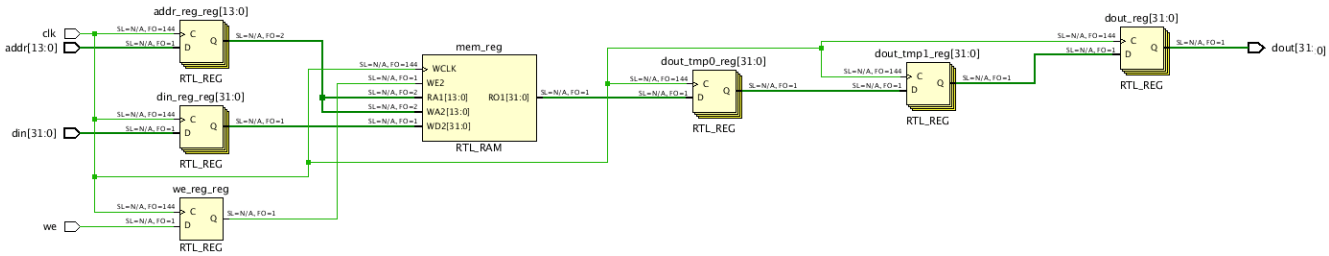


分解深层存储器配置，实现功耗与性能平衡

在深层存储器配置中，可使用综合属性 RAM_DECOMP 实现更好的存储器分解并降低功耗。此属性可在 RTL 中设置。将 RAM_DECOMP 属性应用于存储器时，存储器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。

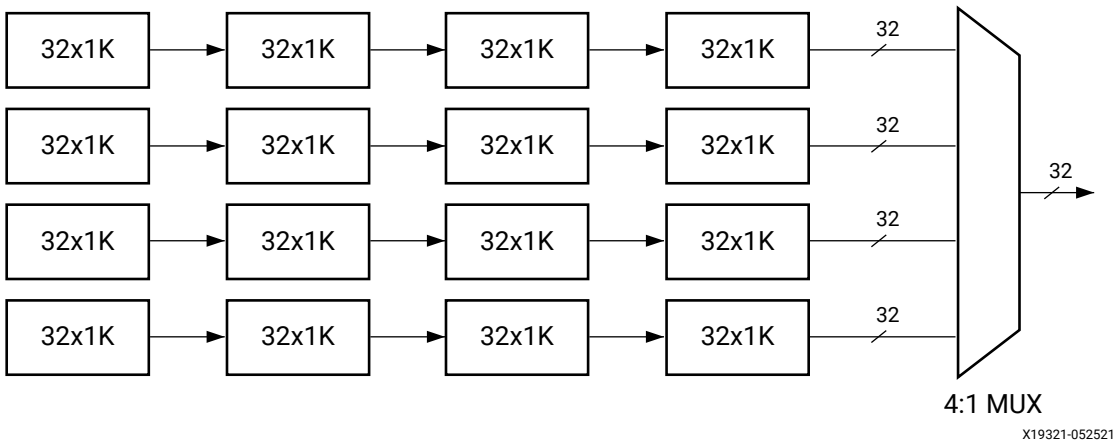
当 CASCADE_HEIGHT 属性与 RAM_DECOMP 属性搭配使用时，综合推断对级联具有更细化的控制权，因此可实现平衡的功耗与性能。此方法需要额外的地址解码逻辑，但可减少任意时间点访问的块 RAM 数量，这有助于降低功耗。下图中的存储器配置 (32 × 16K) 显示了设置 RAM_DECOMP 和 CASCADE_HEIGHT 属性时分解存储器的方式示例。

图 200：**32 × 16K 存储器配置**



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 并对存储器进行分解。

图 201：**使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 32 × 16K 存储器配置生成的结构**



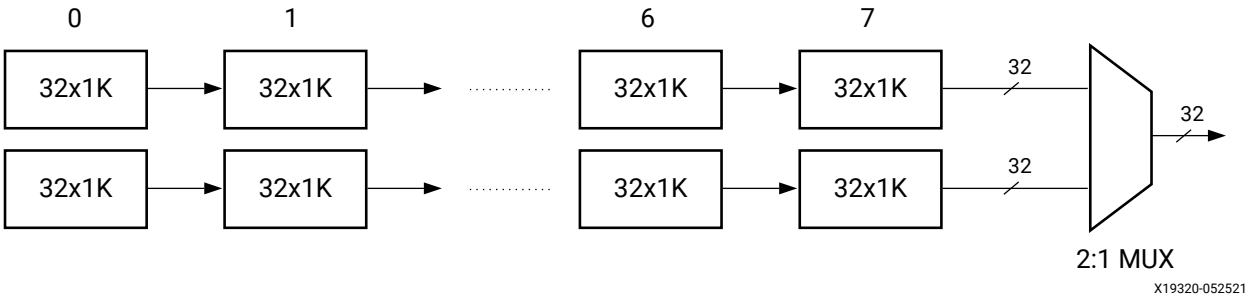
此处使用的基本原语为 32 × 1K，4 个块 RAM 通过内置功能进行级联，组成 32 × 4K 配置。4 个此类并行结构可创建 1 个深度为 16K 的存储器。输出通过多路复用来生成输出数据。

图 202：**使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 32 × 16K 存储器配置的 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_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
```

如果仅应用 RAM_DECOMP = power 属性，则将按下图所示方式推断 16 RAMB36E2 并对存储器进行分解。

图 203：**使用 RAM_DECOMP 属性的 32 × 16K 存储器配置生成的结构**



此处使用的基本原语为 32 × 1K，8 个块 RAM 通过内置功能进行级联，组成 32 × 8K 配置。2 个此类并行结构可创建 1 个深度为 16K 的存储器。输出通过多路复用来生成输出数据。多路复用器为 2:1 MUX。

图 204：**使用 RAM_DECOMP 属性的 32 × 16K 存储器配置的 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 个存储器分解示例，总体功耗节省量是相似的，如 图 201：使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 32 × 16K 存储器配置生成的结构 和 图 203：使用 RAM_DECOMP 属性的 32 × 16K 存储器配置生成的结构 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 201：使用 RAM_DECOMP 和 CASCADE_HEIGHT 属性的 32 × 16K 存储器配置生成的结构）相比 8 级深度的级联块 RAM 链（图 203：使用 RAM_DECOMP 属性的 32 × 16K 存储器配置生成的结构）提供的性能更好。

当存储器深度不为 2 的幂时，优化 RAMB 使用率

以下测试案例可用于观察由综合工具生成的 log 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 位的存储器描述。地址总线需占 16 位。

图 205：**40K x 36 位存储器 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 206：**使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。

以下显示了此存储器配置的手动计算过程：

- 40K x 36 可细分为 2 个存储器：(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存储器执行读写操作，并选择正确的输出数据。
- 32K x 36 存储器可通过 32 个 RAMB 实现：4 * 8 * (4K x 9)
- 8K x 36 存储器可通过 8 个 RAMB 实现：8 * (1K x 36)
- 总计需要 40 个 RAMB 才能以最优化方式实现 40K x 36 存储器。

为验证是否已推断出 RAMB 的最优数量，综合 log 日志文件包含 1 个会话，其中详列了每个存储器的配置方式及其映射到 FPGA 原语的方式。如下图所示，存储器深度作为 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 207：**综合 log 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 | RAMB36 | Hierarchical Name |
+------------+------------+-----------------------+---+---+-----------------------+---+---+---------+--------+--------+------------------+
|test        | mem_reg    | 64 K x 36(READ_FIRST)  | W | R |                       |   |   | Port A  | 0      | 72     | test/extram__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 x 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 208：**最优化的 40 K x 36 位存储器 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;
```

优化 RAMB 输入逻辑以允许输出寄存器推断

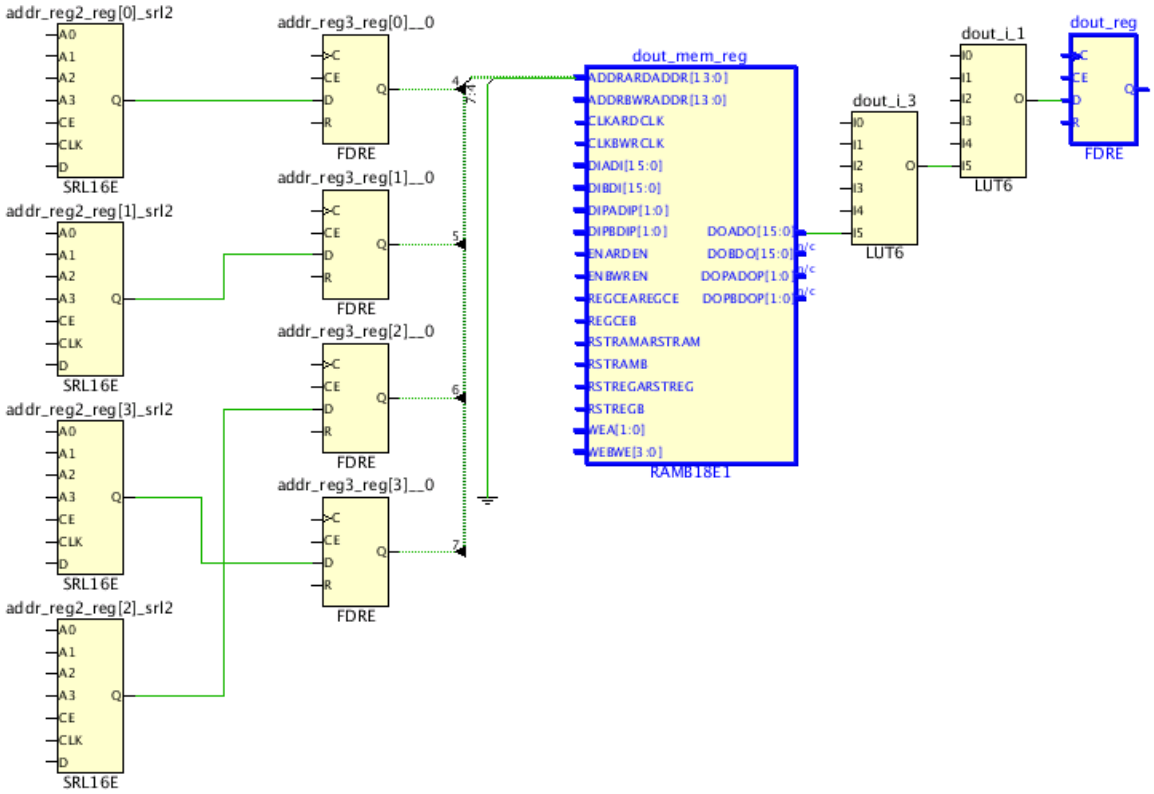
以下 RTL 代码片段可从块 RAM（实际上为 ROM）生成关键路径，其中包含多个止于触发器 (FF) 的逻辑层次。RAMB 单元已在无可选输出寄存器 (DOA-0) 的情况下完成推断，这给 RAMB 输出路径增加了超过 1 ns 的额外延迟惩罚。

图 209：**不含已推断的 RAMB 输出寄存器的存储器 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。

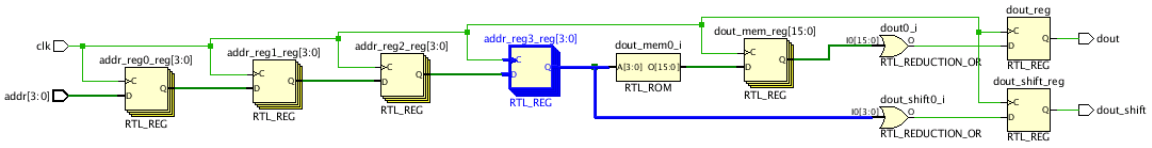
图 210：**来自未启用输出寄存器的 RAMB 的关键路径**



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。如有任何路径过长，或者未能以最优方式利用 FPGA 硬件功能，请返回 RTL 描述，尝试了解已综合的逻辑未实现最优的原因，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 211：**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)，它用于驱动存储器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进行重定时以允许执行 RAMB 推断，而不是使用它来启用 RAMB 可选输出寄存器。

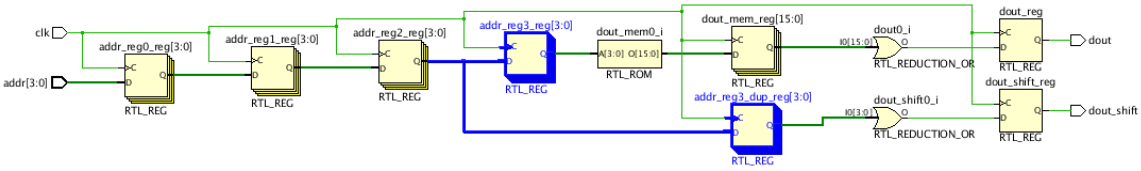
通过复制 RTL 代码中的地址寄存器，使用独立寄存器来驱动存储器地址和互连逻辑或 FPGA 逻辑，即可在启用输出寄存器的情况下推断 RAMB。

手动复制后的 RTL 代码和细化视图如下图所示：

图 212：**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
```

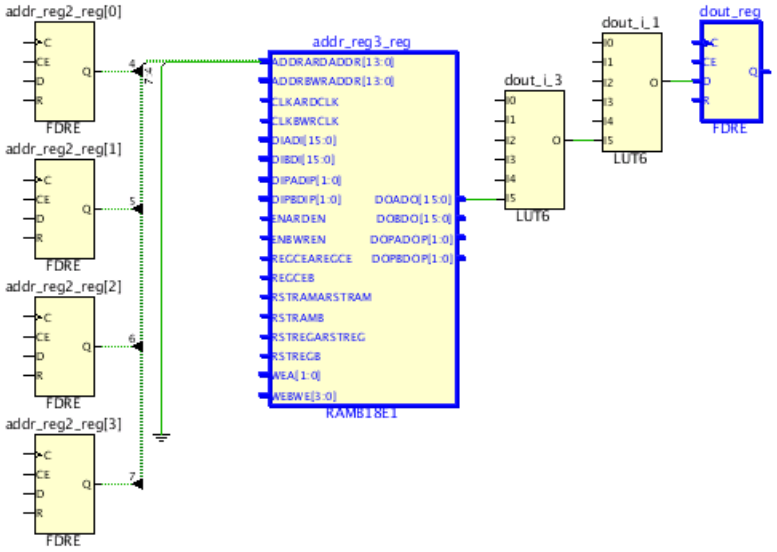
图 213：**含已复制的地址寄存器的细化视图**



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 214：**已修改的 RTL 代码的关键路径**



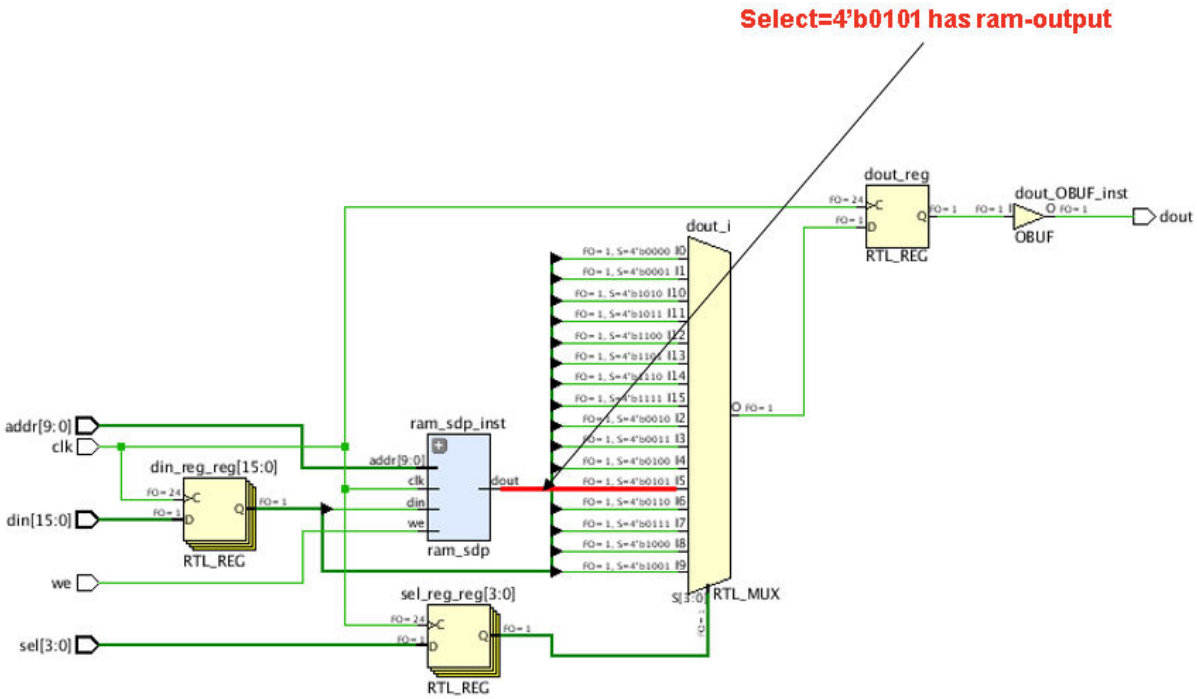
改进 RAMB 输出上的关键逻辑

以下测试用例以将宏（块 RAM）推送至距离目标寄存器更近的位置为例，重点提供了有关通过重构来对关键路径进行改进的信息。

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。

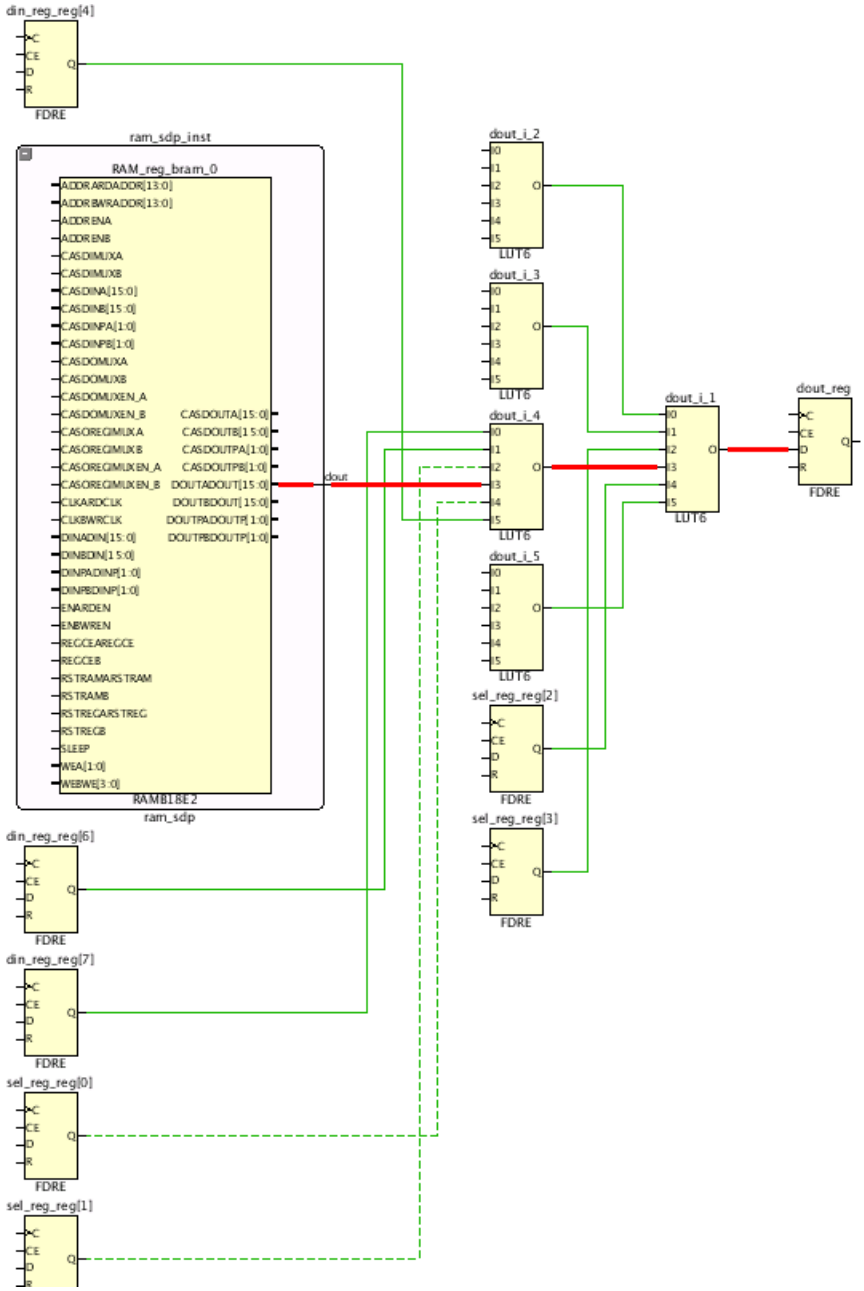
关键路径：块 RAM-> 2 个逻辑层次 -> FF。

图 215：**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 216：**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 217：**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、URAM 和 DSP）周围的长路径即可改进 QoR 结果。

图 218：**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

本章涵盖了可用于时序收敛的技巧。此处所述技巧应视为是对《UltraFast 设计方法时序收敛快捷参考指南》(UG1292) 和《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中所描述的时序收敛技巧的补充。

- 智能设计运行是自动时序收敛流程，用于解决复杂的时序收敛问题，几乎无需用户具备任何使用知识即可使用。
- QoR 建议对象流程能通过应用属性来为用户自动增强 QoR。
- ML 策略流程有助于用户为给定设计选择最优的工具选项。
- 布局规划是一种复杂的技巧，能为布局器提供指导信息以改善布局器结果，从而帮助改善时序路径和拥塞。
- 判断设计是否存在导致无法满足保持时间要求的问题，是决定时序收敛策略的关键部分。

智能设计运行

智能设计运行 (IDR) 是一种特殊类型的实现运行，它使用复杂流程来尝试达成时序收敛。由于 IDR 可能较为激进，因此预计编译时间可达标准运行的约 3.5 倍。

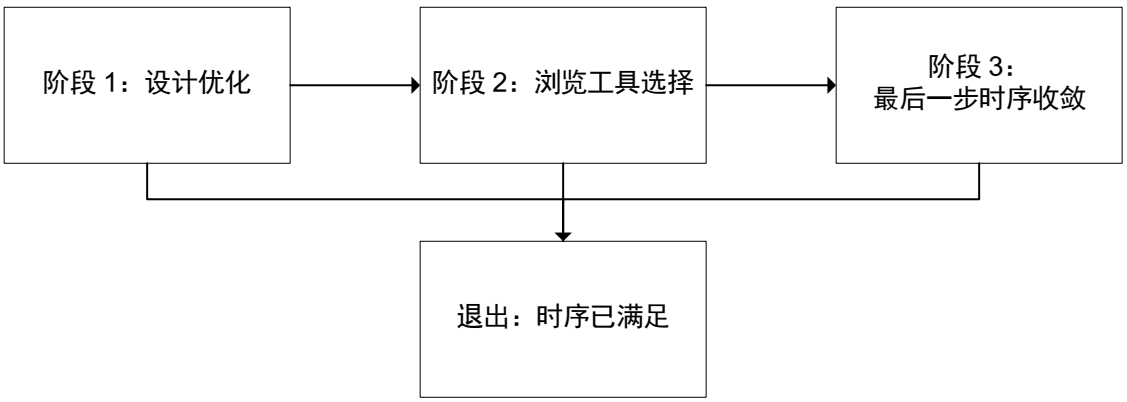
IDR 围绕复杂的时序收敛功能特性展示了一个简单的用户界面，对于大部分设计，它所达成的结果与 FPGA 专家不相上下。

注释： “智能设计运行”对于 AMD Versal™ 架构不可用。请尝试改用“自动 QoR 建议”流程来执行 自动生成和应用建议 操作。

概述

适用于时序收敛的 Intelligent Design Run（智能设计运行）是一种激进的时序收敛实现，其运行的唯一目的就是达成时序收敛。在此过程中功耗和编译时间均不予考虑，但凭借节省使用率可以达成一定程度的功耗最优化。它拆分为三个阶段，如下简化图所示。

图 219：**IDR 概述**



此流程为完全自动化流程，无需用户控制可运行的阶段。在尝试使用 IDR 达成时序收敛前，设计中应不含任何方法论问题。运行 `report_methodology` 并修复或者豁免所有严重警告和警告。

以下提供了每个阶段的详细信息。

- 阶段 1：设计最优化：在设计最优化阶段，会生成并应用 QoR 建议。此阶段的编译时间通常高达标准实现运行的编译时间的 2.5 倍。原因如下：
  - 为了生成准确数据以供分析，必须在布局后和布线前运行实现工具。为了应用建议，设计运行必须复位，并且必须重新运行实现。
  - 支持先达成 QoR 建议的影响效果，而后再开展新分析并生成新建议，这样即可避免过高估计设计问题的影响，从而最大化 QoR 的影响。
- 阶段 2：工具选项探索：此阶段使用 ML 策略来预测要使用的最佳工具选项。
- 阶段 3：最后一步时序收敛：此阶段利用布线后 `phys_opt_design`、使用“Last Mile”指令进行增量实现以及利用增量 QoR 建议来达成时序收敛。要进入此阶段，设计的 RQA 得分必须不低于 3，并且 WNS 在 -0.250 到 0.000 之间。如果不满足这些条件，则会跳过此阶段且退出流程。

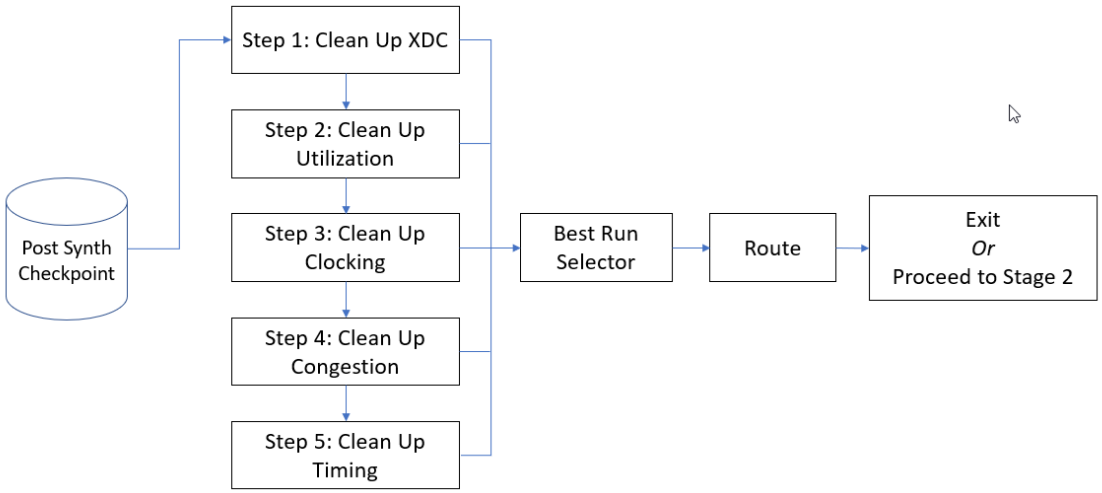
满足以下任一条件时，流程就会退出：

- 在任意阶段，如果满足时序并且设计已完全布线。
- 在阶段 1 中，如果：
  - 设计未能通过初始时序检查。
  - 设计未能通过初始使用率检查。
  - 方法论检查结果为失败，则激活工具退出选项，并且失败的方法论检查包含下列 ID：TIMING-6/7/8/13。
  - 设计布线失败。
  - 不存在预测的 ML 策略。
- 在阶段 2 中，如果不满足 Last Mile 条件。
- 在阶段 3 结束时，如果 Last Mile 算法已穷尽，且无法再进行进一步提升。

阶段 1：设计优化

“Design Optimization”（设计优化）阶段拆分为多个按顺序执行的步骤。下图显示了这些步骤。

图 220：**设计优化步骤**



在任意给定步骤中，可以运行多条实现命令（例如，`opt_design`、`place_design` 和 `route_design`），并且可生成 QoR 建议。对于每个步骤，都有一个目标建议列表；如果在目标建议列表上出现任何生成的建议，设计就会复位到所需的设计阶段，以便成功应用该建议。如果目标建议列表上针对给定步骤没有任何建议，则会跳过此步骤。

设计优化步骤的详细信息如下所述。

- 清理 XDC：检查设计中是否有任何原因会导致生成实现错误和无法修复的时序问题。如果发现错误，流程会退出。在此阶段不生成也不应用任何建议。
- 清理使用率：寻找可以降低使用率而不产生时序惩罚的建议。如果在流程早期可以检测到并修复任何其他与使用率无关的建议，则可应用这些建议。
- 清理时钟设置：运行设计直至 `place_design` 以生成准确的时钟偏差时序数值。如果存在建议，那么该流程会复位。

  注释：如果在“清理使用率”和“清理时钟设置”阶段没有任何建议，则会报告一个名为“First Pass”（首次直通或首通）的特殊阶段。此阶段会用作为基线参考，以便与后续阶段进行对比。为缩短编译时间，如果有建议，则不会生成此特殊阶段。
- 清理拥塞：在此步骤中，运行有限的一部分布线器之后，会生成拥塞以便更准确地查看设计中的拥塞。如果存在建议，则会加以应用。

  注释：生成此拥塞信息时，log 日志文件显示 `route_design` 失败，但由于此时并非旨在进行完整布线，因此可忽略此失败。
- 清理时序：此阶段会基于源自先前阶段的检查点的时序失败的时序路径生成 QoR 建议，并重新运行布局器。

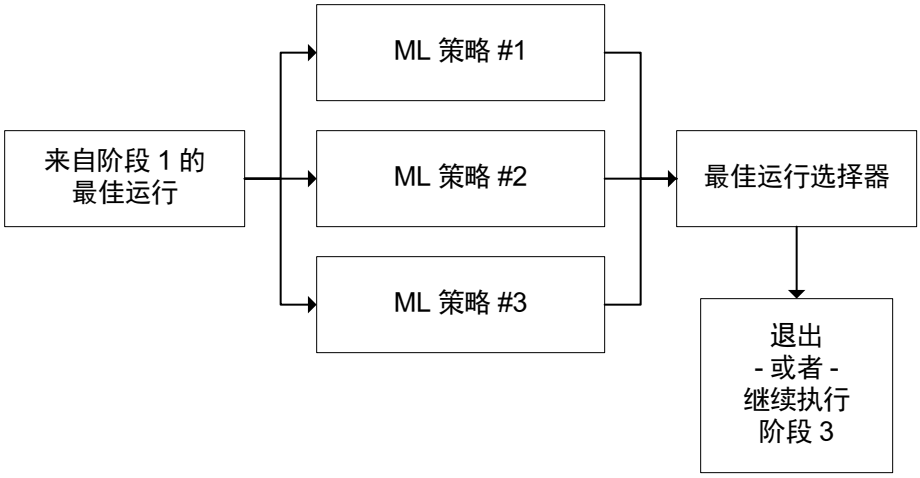
在阶段 1 设计优化末，会判定是退出还是利用最佳运行继续阶段 2 工具选项探索。

为检验 IDR 对设计所做的修改，QoR 建议报告会显示每个步骤包含的各项 GENERATED 建议和 APPLIED 建议。此外还会生成多个检查点，这些检查点可从设计运行目录轻松访问。

阶段 2：工具选项探索

在“工具选项探索”阶段，目标是通过各种工具选项获得最大的 QoR 回报。为此，需使用 ML 策略预测的工具选项来执行 3 轮实现运行。如下图所示：

图 221：**工具探索实现运行**



QoR 建议将沿用阶段 1 中的最佳运行。如果阶段 1 中的后续步骤显示 QoR 劣化，则将丢弃其建议。通过运行 3 轮 ML 策略运行（有 3 轮以上运行可用），那么来自任一轮运行的 QoR 波动都会被平滑掉。

ML 策略阶段使用标准实现运行，只要计算资源允许，即可并行运行。因此，并行使用时，此阶段的编译时间约为一轮实现运行。完成实现运行后，最佳运行选择器会判定要继续使用的运行。流程行为如下：

- 如果满足时序，则退出。
- 如果满足“Last Mile Timing Closure”（最后一步时序收敛）条件，则继续执行阶段 3。
- 如果不满足“Last Mile Timing Closure”条件，则退出。

注释：要使用多个远程主机并行运行阶段 2，请参阅《Vivado Design Suite 用户指南：实现》(UG904) 的“附录 A”章节。

阶段 3：最后一步时序收敛

“Last Mile Timing Closure”（最后一步时序收敛）阶段会从前两个阶段中的任一阶段提取最佳实现运行结果，并尝试在此基础上达成时序收敛。此阶段的 QoR 增益相比于编译时间可能较小。设计必须以满足“Last Mile Timing Closure”要求才能运行此阶段。

“Last Mile”指令会从现有已布线检查点继续执行，并尝试作用于失败的路径。在约 20% 的设计中会达成时序收敛，其中 WNS < -0.100 ns。

“Last Mile Timing Closure”阶段的目标是在设计上达成时序收敛。这相比于默认工具流程稍有不同，默认工具流程的目的是尽可能达成最佳 WNS 和时序收敛的 WHS。算法必须在尝试改善时序与不显著更改布局布线结果之间达成平衡。为此，可使用“Last Mile”增量指令和 QoR 建议来收敛时序。在此过程中，将复用来自参考运行的含 APPLIED 属性的建议以及已设置 INCREMENTAL_FRIENDLY 属性的建议。完成布线后，可运行 `phys_opt_design` 来进一步尝试收敛时序。

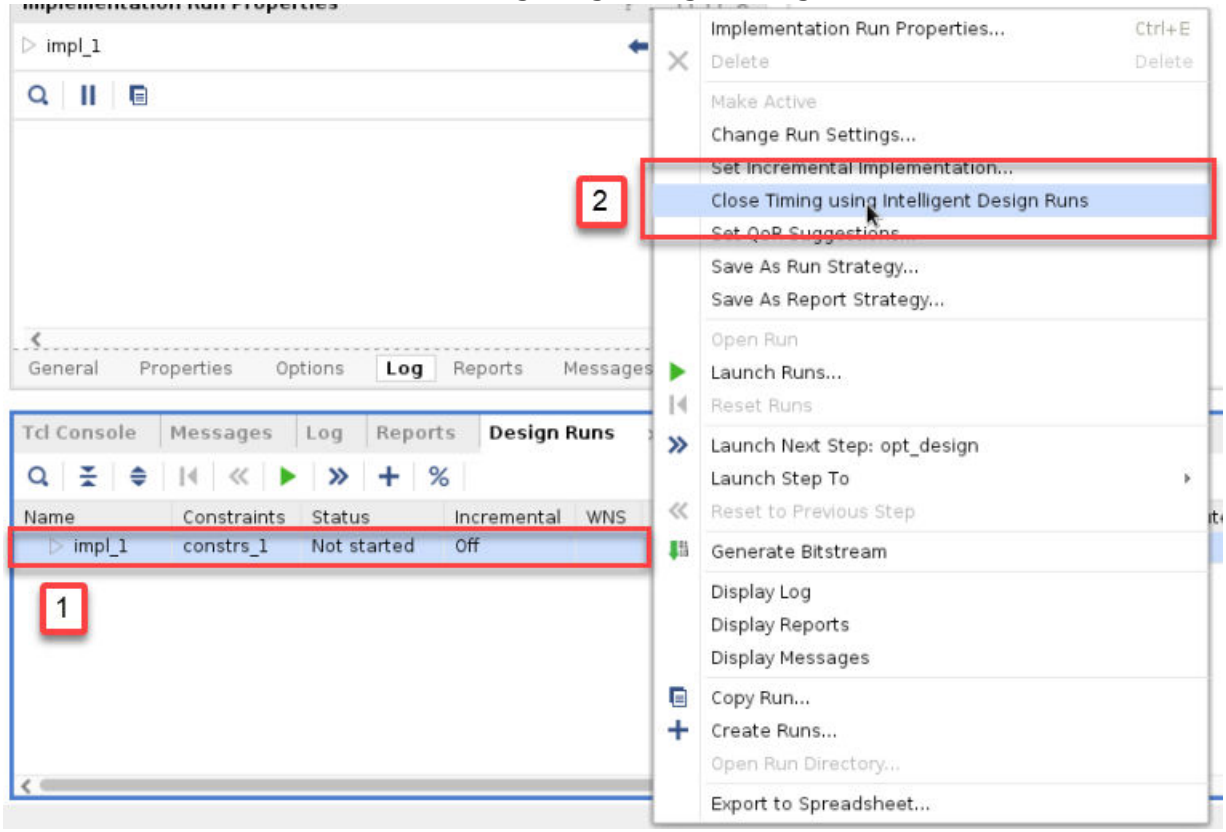
为了从阶段 1 或 2 进入“Last Mile Timing Closure”阶段，设计必须满足以下条件：

- 运行已完全完成布线
- WNS > -0.250
- RQA 得分不低于 3

创建智能设计运行

智能设计运行 (IDR) 是基于标准实现运行创建的。在“Design Runs”（设计运行）窗口中，右键单击实现运行，然后选择“Close Timing using Intelligent Design Runs”（使用智能设计运行收敛时序），如下图所示。

图 222：**“Close Timing Using Intelligent Design Runs”命令**



创建智能设计运行的等效 Tcl 命令如下所示：

```
create_run -flow {Vivado IDR Flow 2021} -parent_run <synth runName> <idr
runName>
set_property REFERENCE_RUN impl_1 [get_runs <idr runName>]
```

REFERENCE_RUN 属性用于从实现运行复制 Tcl 挂钩。在每个实现的运行阶段都会应用 Tcl 挂钩。例如，如果存在 `opt_design` 前的 Tcl 挂钩，那么每次调用 `opt_design` 命令之前都会执行此挂钩。当运行复位时会检验该属性，以便提取对实现运行 Tcl 挂钩执行的后续更改。如果要将 Tcl 挂钩添加到 IDR，请首先创建实现运行、添加 Tcl 挂钩，然后创建新的 IDR。

注释：当前不支持 `init_design` 前后的 Tcl 挂钩。

由于指令受 IDR 控制，从具有相同网表、相同约束和相同 Tcl 挂钩的运行创建 IDR 是没有价值的。因此存在如下限制：基于任意给定实现运行，只能创建一个 IDR。如需多个 IDR，请更改综合选项以创建不同网表或者修改布局规划。

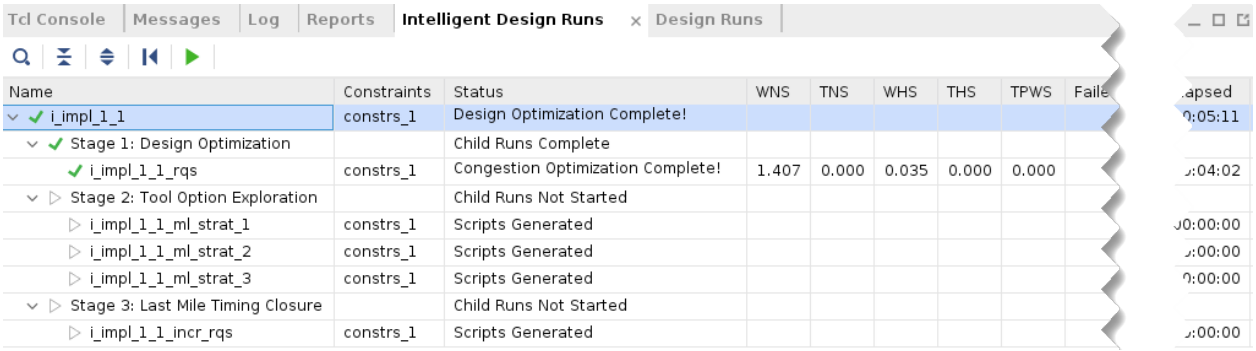
注释：无法从综合运行直接创建 IDR 运行。

“Intelligent Design Runs”窗口中的流程控制

“Intelligent Design Runs”（智能设计运行）窗口可提供两项功能，如下图所示：

- 上下文相关的右键单击菜单，用于访问流程控制和设计分析选项
- 允许访问诸如 WNS、TNS、WHS 和 THS 等指标，以便执行顶层和子层 IDR 运行

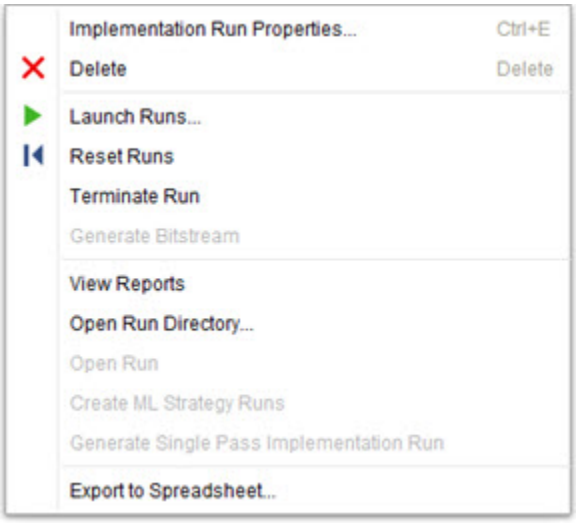
图 223：**“Intelligent Design Runs”选项卡**



这些指标表示 IDR 阶段的最佳运行指标。在整个流程中，这些指标会定期更新。

右键单击菜单与上下文相关，以选定的流程阶段为目标。右键单击顶层即可显示下列选项（这些选项是右键单击任意下层阶段的超集）：

图 224：**流程控制和设计分析选项菜单**



菜单选项如下所述。

- “Implementation Run Properties”（实现运行属性）：打开 IDR 运行的运行属性。这是正常运行的缩减版本。

- “Delete”（删除）：删除运行。

  注释：基于任意给定实现运行只能创建一轮 IDR 运行。要创建第二轮运行，必须先删除第一轮运行，或者应创建第二轮实现运行，然后从中创建 IDR。

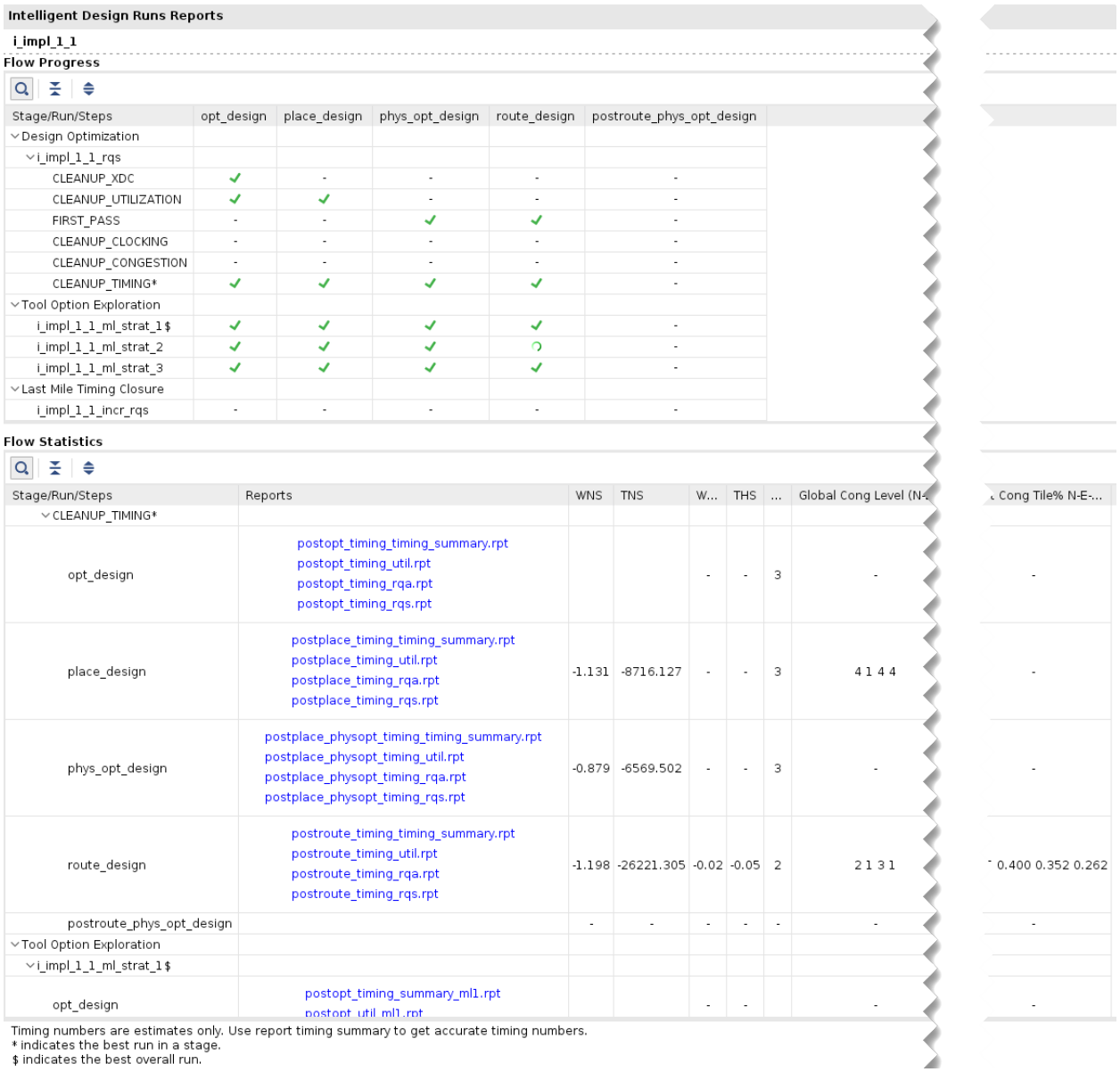
- “Launch Runs”（启动运行）：启动运行。
- “Reset Runs”（复位运行）：将 IDR 复位，并删除所有文件。
- “Terminate Run”（终止运行）：终止所选运行，但不删除运行目录中的文件。该选项仅在运行过程中可用。
- “Generate Bitstream”（生成比特流）：如果运行尚未启动，则启动运行直至比特流生成。如果 IDR 已完成但比特流尚未生成，则该选项会从已完成的 IDR 的最佳运行的已布线的检查点生成比特流。该选项在运行过程中不可用。
- “View Reports”（查看报告）：打开“Intelligent Design Runs Reports”（智能设计运行报告）窗口。
- “Open Run Directory”（打开运行目录）：打开运行目录。该选项可用于访问中间检查点和文本报告。
- “Open Run”（打开运行）：从选定的设计分析阶段打开所选运行或者最佳运行。仅在已布线的检查点上可用。
- “Create ML Strategy Runs”（创建 ML 策略运行）：完成阶段 1 后，如果设计仍有时序失败，那么该选项会变为可用。ML 策略会自动创建并就绪，可立即搭配任意 APPLIED QoR 建议使用。选中该选项会创建 3 轮运行，等效于运行阶段 2。如果设计发生更改，那么相比于使用包含“Last Mile”（最后一步）的单通运行，首选使用该选项，因为它更适合处理更大的更改。
- “Generate Single Pass Implementation Run”（生成单通实现运行）：创建标准实现运行，它会设置 RQS 文件和增量检查点（如需），由此创建的结果与在 IDR 中可达成的结果相同。该选项仅在成功完成 IDR 后才可用。

Intelligent Design Run Reports

“Intelligent Design Run Reports”（智能设计运行报告）窗口分为两个部分，如下所示：

- “Flow Progress”（流程进度）：
  - 用于显示已运行的阶段以及当前正在运行的阶段。
  - 指明最佳整体运行 (“$”) 和某一阶段内的最佳运行 (“*”)。
- “Flow Statistics”（流程统计数据）：
  - 用于显示有关所收集的设计时序和拥塞信息的统计数据。
  - 包含指向 IDR 中生成的报告的超链接。

图 225：**“Intelligent Design Run Reports”窗口**



Intelligent Design Runs Reports

i_impl_1_1

Flow Progress

Stage/Run/Steps	opt_design	place_design	phys_opt_design	route_design	postroute_phys_opt_design
Design Optimization					
i_impl_1_1_rqs					
CLEANUP_XDC	✓	-	-	-	-
CLEANUP_UTILIZATION	✓	✓	-	-	-
FIRST_PASS	-	-	✓	✓	-
CLEANUP_CLOCKING	-	-	-	-	-
CLEANUP_CONGESTION	-	-	-	-	-
CLEANUP_TIMING*	✓	✓	✓	✓	-
Tool Option Exploration					
i_impl_1_1_ml_strat_1$	✓	✓	✓	✓	-
i_impl_1_1_ml_strat_2	✓	✓	✓	◌	-
i_impl_1_1_ml_strat_3	✓	✓	✓	✓	-
Last Mile Timing Closure					
i_impl_1_1_incr_rqs	-	-	-	-	-

Flow Statistics

Stage/Run/Steps	Reports	WNS	TNS	W...	THS	...	Global Cong Level (N-	Cong Tile% N-E-...
CLEANUP_TIMING*								
opt_design	postopt_timing_timing_summary.rpt postopt_timing_util.rpt postopt_timing_rqa.rpt postopt_timing_rqs.rpt			-	-	3	-	-
place_design	postplace_timing_timing_summary.rpt postplace_timing_util.rpt postplace_timing_rqa.rpt postplace_timing_rqs.rpt	-1.131	-8716.127	-	-	3	4 1 4 4	-
phys_opt_design	postplace_physopt_timing_timing_summary.rpt postplace_physopt_timing_util.rpt postplace_physopt_timing_rqa.rpt postplace_physopt_timing_rqs.rpt	-0.879	-6569.502	-	-	3	-	-
route_design	postroute_timing_timing_summary.rpt postroute_timing_util.rpt postroute_timing_rqa.rpt postroute_timing_rqs.rpt	-1.198	-26221.305	-0.02	-0.05	2	2 1 3 1	0.400 0.352 0.262
postroute_phys_opt_design		-	-	-	-	-	-	-
Tool Option Exploration								
i_impl_1_1_ml_strat_1$								
opt_design	postopt_timing_summary_ml1.rpt postopt_util_ml1.rpt			-	-		-	-

Timing numbers are estimates only. Use report timing summary to get accurate timing numbers.
* indicates the best run in a stage.
$ indicates the best overall run.

在 IDR 中会捕获以下数据：

- RQA 得分（在整个过程中捕获）
- 拥塞（在布局后和初始布线阶段收集所得）
- 时序（仅限在布局后、执行物理最优化后以及布线后阶段内收集）

注释：如果尚未运行任何阶段或者尚未收集特定指标的任何数据，则会显示连字符 (-)。

注释：收集的时序数值通常为估算值，可能与运行 `report_timing_summary` 后所得的数值略有不同。

生成的报告是固定的。用户不直接控制生成的报告。如需额外报告，应使用 Tcl 挂钩来添加。或者，可在运行目录中打开检查点之后生成额外报告。在顶层运行目录中会自动生成等效的文本报告。其固定名称为 `idr_flow_summary.rpt`。

智能设计运行后续步骤

由于智能设计运行需要较长时间才能完成，每次设计更改时连续运行不太现实。因此，当智能设计运行完成后，需要将从运行中习得的结果整合到主工程运行中。具体有两种方法：右键单击顶层的“Intelligent Design Run”（智能设计运行），然后选择以下任一方法：

1. Generate Single Pass Implementation Run（生成单通实现运行）
2. Generate ML Strategy Runs（生成 ML 策略运行）

单通运行会创建一轮与智能设计运行等效的运行，它会复制在最佳运行中得到的结果，但会采用一种更节省编译时间的方式来运行。此轮运行是根据以下要素动态构成的：

- 已改进设计的 QoR 建议
- 根据“阶段 1 实现策略”或“阶段 2 ML 预测的策略”的运行结果，取其中最佳运行来生成的实现命令指令。
- 如果已运行“阶段 3：最后一步”并产生最佳结果，那么在设计布线后，会将利用增量更改的一组额外命令作为 Tcl 进程添加到 Tcl 挂钩中。

尝试以相同输入网表复制相同结果时，单通运行非常有用。需要时序收敛构建时尤其如此，因为它可帮助用户以最小的工作量完成设计收敛。然而，这个流程在应对设计更改时可能表现欠佳。此外，当包含“最后一步”阶段时，它会在不合适的情况下增加编译时间。

进行设计更改时，“ML 策略”运行可提供更稳健的流程。使用“ML 策略”运行时将会：

- 包含已改进设计的所有 QoR 建议
- 运行 3 轮实现运行，每轮各从排名前 3 的预测实现策略中取一项策略来运行。这样可在计算与 QoR 重新运行之间取得良好平衡

并且运行的完成速度也更快，因为其中排除了向实现运行添加额外步骤的“最后一步”流程。进行设计更改时，这通常更适用于迭代。如果后续要重新添加“最后一步”，那么应在完全布线的 DCP 文件中调用以下脚本来复制“最后一步”流程。

```
package require Vivado 1.2022.1

namespace eval ::xilinx::designutils {
    namespace export last_mile
}

proc ::xilinx::designutils::last_mile {{write_intermediate_files 0}
{write_final_files 0}} {
# Vivado    : Supported 2023.1 and later
#
# Families : Versal, Ultrascale+, Ultrascale
#
# Summary  : Run last mile flow. This is equivalent to
#            running stage 3 in IDR.
#            The last mile flow will run intensive
#            algorithms trying to close out timing
#            on a fully routed design.
#            It is recommended to run post route
#            phys_opt_design -directive Explore before
#            running last mile.
#            A call to this script can be added as a
#            TCL_POST hook to post route
#            phys_opt_design -directive Explore.
#
#            The requirements to run this are:
#              i)   A fully routed open design
```

```
#               ii)  When called a check will be made to
#                    see if the QoR Assessment Score of 3
#                    or 4 is achieved, if not it will exit.
#               iii) A WNS >= -0.250
#
# Arguments:
#
# write_intermediate_files -
# When set to 1, writes out intermediate DCP and timing
# reports after:
#  i)   preplace phys_opt_design,
#  ii)  place_design
#  iii) postplace phys_opt_design
#
# write_final_files -
# Writes reports and DCPs after the final stage
# route_design
# when set to 1.
#  Project users:     It is recommended to leverage
#                     project infrastructure to write
#                     reports
#  Non-Project users: Recommended to add custom
#                     write_checkpoint and reporting
#                     commands after this script is
#                     executed.
#
# Return Value:
# 1 - last mile completed successfully
#
# Categories: xilinxtclstore, designutils
set top [get_property TOP [current_design]]

# Entry Checks
puts "-I: Checking last mile entry criteria is met..."
if {[llength [current_design -quiet]] == 0} {
    puts "-E: Last mile requires an open design that is\
    fully routed"
    return -code 2 "Error: Last mile requires an open\
    design that is fully routed"
}
if {[report_route_status -boolean_check ROUTED_FULLY] == 0} {
    puts "-E: Last mile requires a design that is fully\
    routed"
    return -code 2 "Error: Last mile requires a design\
    that is fully routed"
}
# actual value is 3
if {[set score [get_assessment_score]] < 3} {
    puts "-E: Last mile requires a design has an QoR\
    Assessment Score of 3 or greater.\
    Design Score: $score"
    return -code 2 "Error: Last mile requires a design\
    has an QoR Assessment Score of 3 or greater.\
    Design Score: $score"
}
# actual value is -0.250
if {[set slack [get_property SLACK [get_timing_paths -setup]]] < -0.250} {
    puts "-E: Last mile requires a design a WNS >= -0.250\
    Design Slack: $slack"
    return -code 2 "Error: Last mile requires a design a\
    WNS >= -0.250. Design Slack: $slack"
}
puts "-I: Passed last mile entry criteria checks"
```

```
# QoR Suggestion Generation
puts "-I: Generating QoR Suggestions for Last Mile"
report_qor_suggestions -file ${top}_qor_suggestions_lastmile.rpt
if {[llength [get_qor_suggestions -filter {INCR_FRIENDLY} -quiet ] ] > 0} {
    puts "-I: Enabling generated incremental friendly\
    QoR Suggestions..."
    write_qor_suggestions -of_objects \
    [get_qor_suggestions -filter {INCR_FRIENDLY}] \
    -force -file ${top}_qor_suggestions_lastmile.rqs
    set_property ENABLED 1 [get_qor_suggestions \
    -filter {INCR_FRIENDLY}]
} else {
    puts "-I: No QoR Suggestions found for last mile\
    flow. Flow will continue.."
}

# Tool Flow Commands
phys_opt_design -clock_opt -retime -lut_opt
if {$write_intermediate_files != 0} {
    report_timing_summary -delay_type min_max \
    -report_unconstrained -check_timing_verbose \
    -max_paths 10 \
    -input_pins -routable_nets \
    -rpx ${top}_timing_lastmile_preplace_physopt.rpx \
    -file ${top}_timing_lastmile_preplace_physopt.rpt
    write_checkpoint -force \
    ${top}_lastmile_preplace_physopt.dcp
}
place_design -directive LastMile
if {$write_intermediate_files != 0} {
    report_timing_summary -delay_type min_max \
    -report_unconstrained \
    -check_timing_verbose \
    -max_paths 10 \
    -input_pins \
    -routable_nets \
    -rpx ${top}_timing_lastmile_placed.rpx \
    -file ${top}_timing_lastmile_placed.rpt
    write_checkpoint -force \
    ${top}_lastmile_placed.dcp
}
phys_opt_design -directive Explore
if {$write_intermediate_files != 0} {
    report_timing_summary -delay_type min_max \
    -report_unconstrained \
    -check_timing_verbose \
    -max_paths 10 \
    -input_pins \
    -routable_nets \
    -rpx ${top}_timing_lastmile_postplace_physopt.rpx \
    -file ${top}_timing_lastmile_postplace_physopt.rpt
    write_checkpoint -force \
    ${top}_lastmile_postplace_physopt.dcp
}
route_design -directive Explore
if {$write_final_files != 0} {
    report_timing_summary -delay_type min_max \
    -report_unconstrained -check_timing_verbose \
    -max_paths 10 -input_pins -routable_nets \
    -rpx ${top}_timing_lastmile_routed.rpx \
    -file ${top}_timing_lastmile_routed.rpt
    write_checkpoint -force \
```

```
    ${top}_lastmile_routed.dcp
}
phys_opt_design -directive Explore
phys_opt_design -directive Explore
if {$write_final_files != 0} {
    report_timing_summary -delay_type min_max \
    -report_unconstrained -check_timing_verbose \
    -max_paths 10 -input_pins -routable_nets \
    -rpx ${top}_timing_lastmile_postroute_physopt.rpx \
    -file ${top}_timing_lastmile_postroute_physopt.rpt
    write_checkpoint -force \
    ${top}_lastmile_postroute_physopt.dcp
}

return 1
}
```

面向非工程用户的智能设计运行建议

要使用智能设计运行功能特性，需要 Vivado 工程。这是因为需要进行运行管理。以下指示信息解释了创建综合后工程的最简单方法。这些信息适用于以下流程的用户：

- 非工程实现运行
- 使用较低版本的 Vivado 或第三方综合工具进行综合

访问智能设计运行功能特性的最简单方法是给综合后工程添加单个完整的设计 DCP 源文件。这样即可提供完整的网表和所有设计约束。创建此工程时，可按照 创建智能设计运行 部分中的步骤启动 IDR。

1. 从现有实现运行生成单一检查点。要执行此操作，请在实现运行 Tcl 脚本中找到 `opt_design` 调用，并在此阶段之前写入一个检查点。下面给出了 1 个示例：

   ```
   write_checkpoint -force <PreOptDesign>.dcp
   opt_design -directive Explore ; ## FOR EXAMPLE ONLY ##
   ```

   最早的完整设计检查点是在 `opt_design` 之后写入的，无需用户干预。但理想情况是在运行 `opt_design` 之前写入检查点。在此情况下，应使用 `opt_design` 之前的 Tcl 挂钩来写入检查点。在 Tcl 脚本中添加 `write_checkpoint` 行，如果存在冲突，则在 `init_design` (`link_design`) 之后或者在 `opt_design` 之前插入该行：

   ```
   add_files -fileset utils_1 -norecurse ./test.tcl
   set_property STEPS.INIT_DESIGN.TCL.POST [get_files ./test.tcl -of
   [get_fileset utils_1] ] [get_runs <ImplRun>]
   ```

   或

   ```

   set_property STEPS.OPT_DESIGN.TCL.PRE [get_files ./test.tcl -of
   [get_fileset utils_1] ] [get_runs <ImplRun>]
   ```

2. 如有检查点可用，请创建综合后工程，可使用“New Project”Wizard（新建工程向导）来轻松创建该工程。创建工程的等效 Tcl 代码如下所示：

   ```
   create_project <ProjectName> <ProjectDirectory> -part <PartName>
   set_property design_mode GateLvl [current_fileset]
   add_files -norecurse <PreOptDesign>.dcp
   ```

如需了解有关工程设置的更多信息，请参阅《Vivado Design Suite 用户指南：设计流程概述》(UG892) 中的“使用工程模式”。

受支持的家族和设计流程

下表汇总了该版本中 IDR 支持的器件家族和设计流程。

表 18：**IDR 支持汇总**

项目	支持状态
支持的家族	UltraScale、UltraScale+
流程	工程模式
DFX	支持（不包括 Versal 最后一步操作）
Vitis	不支持

QoR 建议

凭借以下策略，QoR 建议可用于改善设计满足时序的能力：

- 在命令中添加开关，如 `opt_design`
- 在单元和信号线之类的设计对象中添加属性
- 完整的实现策略

`report_qor_suggestions` 命令可在 AMD Vivado™ IDE 或基于文本的报告中生成报告。它可用于：

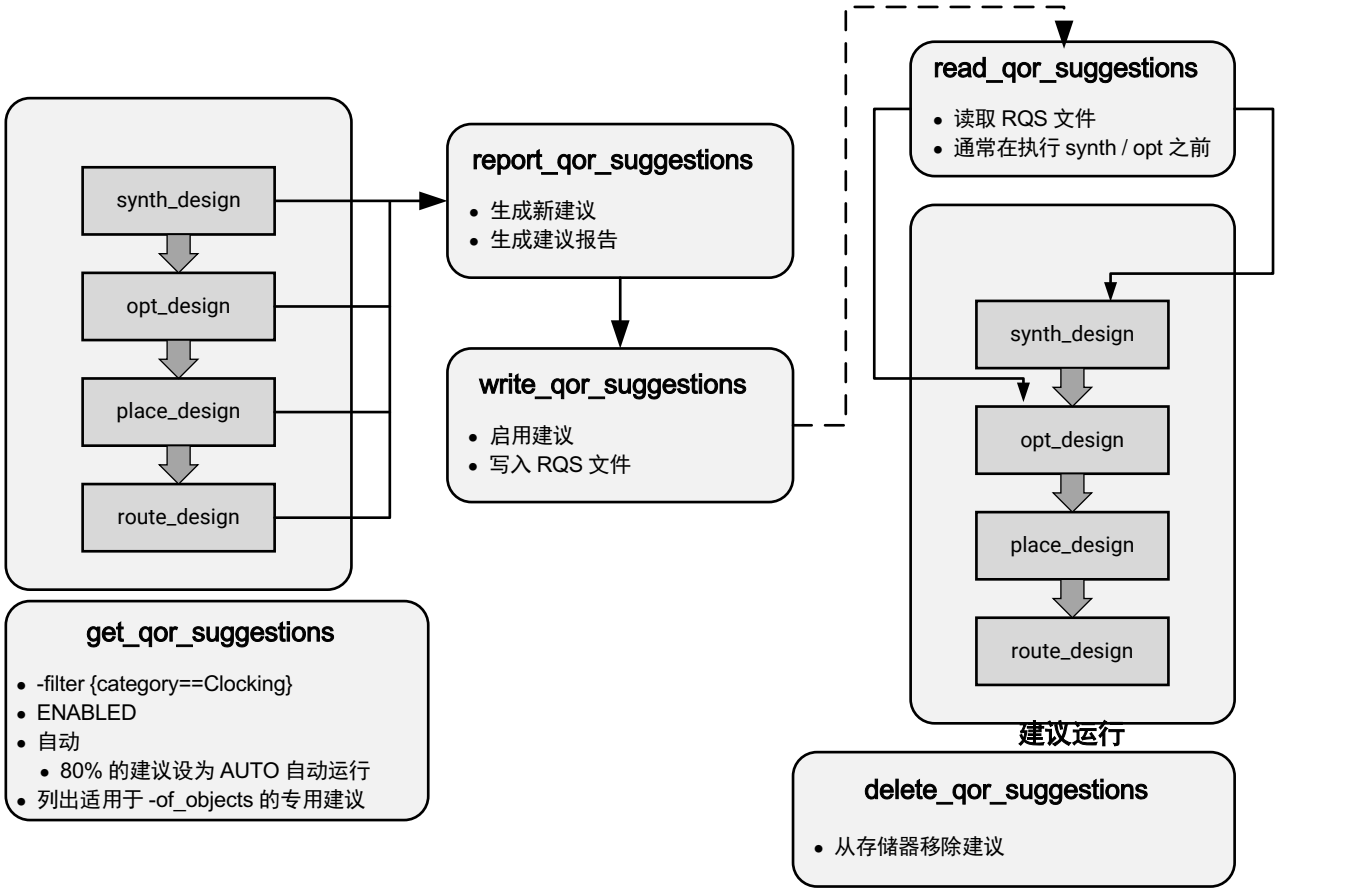
- 生成和查看有关存储器中当前设计的新建议
- 查看使用 `read_qor_suggestions` 命令读入的现有建议

综合后，随时可对存储器中加载的设计运行 `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
- XDC
- 网表
- 使用率
- 拥塞
- 时序
- 策略

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 226：**建议流程**



生成新建议前，必须将设计加载到存储器中。在综合后的任意阶段均可运行 `report_qor_suggestions`。返回的建议在报告中按重要性从高到低进行排序并列示。

它仅报告改善设计 QoR 所需的建议。有时，必须先获取布局或布线信息，而后才能发出建议。此外，存在一些限制用于确保仅生成包含必要的设计更改的建议。

- 网表建议是基于网表分析来生成的。这些建议用于识别导致后续流程中时序失败的网表结构，但并不会直接观察时序路径，因此可在时序收敛的设计上生成。
- 时钟设置建议通常需在布局之后生成，但布局前已有准确信息可用的情况下则例外。这些建议需要获取失败的时序路径，但有少数例外情况。
- 时序建议是通过检验每个时钟组中前 100 条失败的时序路径来生成的。
- 生成使用率建议的前提是它判定建议的目标资源已过度使用，并且生成的建议不会导致关键资源增加。这些建议可在任意设计阶段报告。
- 拥塞仅在布局后才会报告。如果设计已完成布线并且时序已满足，则不会报告拥塞建议，因为已证明这些建议对时序收敛没有影响。
- 最后一个类别是“策略”，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。

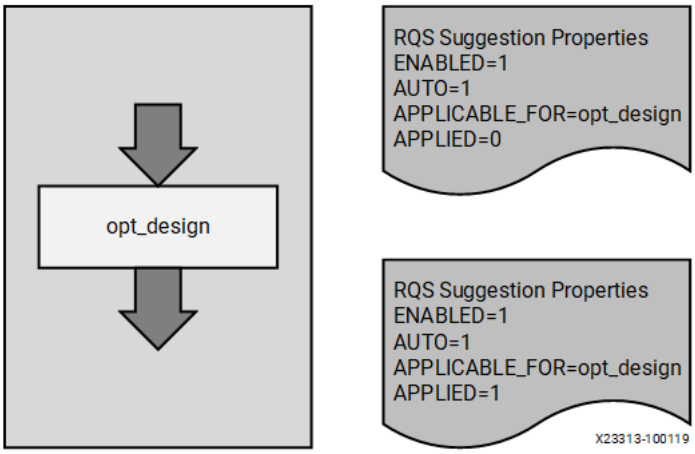
执行建议

满足以下条件时，在建议运行轮次期间执行建议：

- 这些建议处于已启用 (ENABLED) 状态。
- 必须运行 APPLICABLE_FOR 阶段。
- 这些建议必须设置为 AUTO。

执行建议时，APPLIED 设置将会更新，如下图所示：

图 227：**建议执行**



在实现流程中，如果未将建议中的某个属性正确应用于关联的单元或信号线，则 FAILED_TO_APPLY 将被设置为 1。如果部分应用某项建议，则将生成新的建议，此建议将被拆分为已应用的建议和未能应用的建议。

注释：如果实现工具在后续流程中拒绝 FAILED_TO_APPLY 属性，则该属性将不会置位。仅当给定属性无法应用于对象时，FAILED_TO_APPLY 才会置位。该属性的应用过程不接受跟踪。

如果 APPICABLE_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，请务必将此建议写入 RQS 文件以便后续继续使用此建议。

所有相关命令

有 5 条相关命令可用于处理 QoR 建议对象：

表 19：**所有相关命令**

命令	功能
report_qor_suggestions	生成新建议。 报告现有建议。
write_qor_suggestions	将建议对象写入文件。在此进程中，将自动启用 (ENABLED) 建议。
read_qor_suggestions	从文件中读取建议对象。

表 19：**所有相关命令** (续)

命令	功能
get_qor_suggestions	返回 QoR 建议对象。
delete_qor_suggestions	从存储器中移除 QoR 建议。
get_qor_checks	返回 Vivado 可生成的所有建议的列表。

管理建议

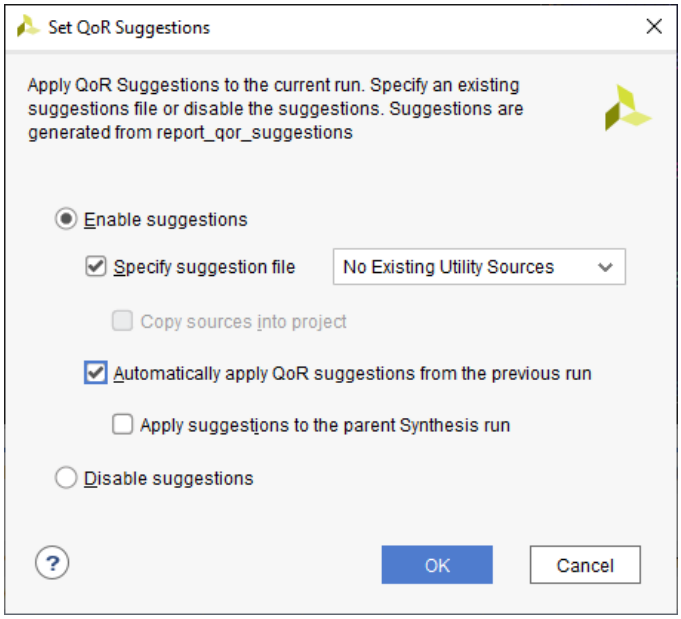
工程模式下的一些建议管理功能既有助于简化建议文件的管理，也有助于提高一般情况下的性能。在非工程流程下也可以复制这些方法。

工程模式

在工程模式下，建议写入 RQS 文件后，会将此文件自动添加到工程 (`utils_1`) 中的实用工具源文件集内。请将此文件存储于除运行目录外的其他目录中，因为当运行复位后，将删除此文件。每轮运行都应采用不同的独立目录来存储各轮运行的专用建议文件。推荐采用如下位置：`<project_dir>/<project>.srcs/utils_1/<run_name>`。

在工程内的“Design Runs”（设计运行）窗口中右键单击运行，然后选择“Set QoR Suggestions”（设置 QoR 建议）。在综合运行和实现运行中可能都需要添加此操作。

图 228：**“Set QoR Suggestions”窗口**



选择“Enable Suggestions”（启用建议）。这样您即可选择建议文件和/或执行自动运行。所选建议文件如果尚未添加到 `utils_1` 文件集中，则会执行添加。

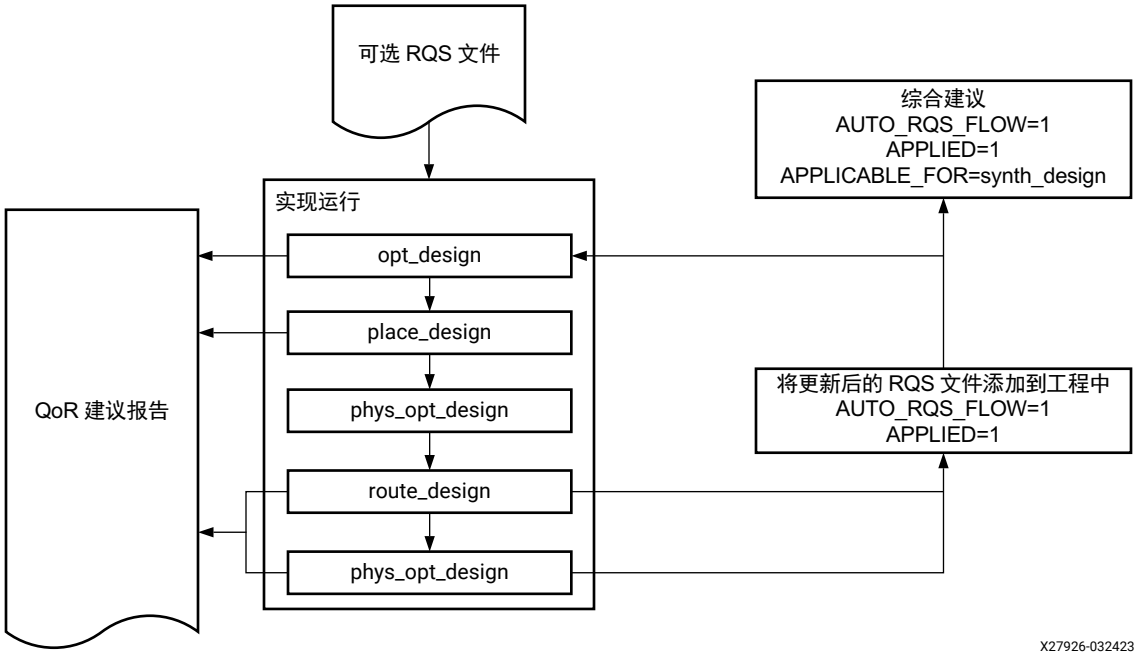
以下代码片段中显示了此流程的等效 Tcl 命令：

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

自动生成和应用建议

自动 QoR 建议流程会为基于工程的运行生成并应用 QoR 建议。对于易于更改的设计而言，这是生成和应用 QoR 建议的最简单的方法。启用自动 QoR 建议时，会按下图所示调整标准实现流程。

图 229：**含自动 QoR 建议的实现流程**



初始运行可设为使用用户所选的 RQS 文件，也可以在不使用 RQS 文件的情况下运行。如果选择用户 RQS 文件，那么此文件会保留在原处直至生成新建议为止。此文件中的任何建议都将写入 `<top>_routed.rqs` 文件，并移除保留原始文件的要求。

如果已启用 `report_qor_suggestions` 命令，那么在 `opt_design`、`place_design` 以及 `route_design` 或布线后 `phys_opt_design` 之后运行该命令。

大多数建议会在实现运行结束时写入 RQS 文件中，以便在下一次实现运行时应用。可写入 RQS 文件的建议包括：

- 含有属性 AUTO_RQS_FLOW==1 的新建议
- 含有属性 APPLIED==1 的旧建议

此外还包括 RQS_CLOCK-1 建议，该建议可在 `opt_design` 处生成，且在同一轮运行中应用。

在运行命令的每个点的“Reports”（报告）选项卡中提供了 `report_qor_suggestions` 报告，可用于检查这些建议。

当运行复位后，会将建议从实现运行目录复制到另一处位置，并将其添加到 `utils_1` 文件集中。默认情况下，此位置位于源文件目录中，但也可修改为任意其他位置。

建议也可以应用于父综合（如果存在）。仅限单一子实现运行可提供综合建议。如选中多个子实现运行，则使用最新的运行。

在第一类运行对象上使用了以下属性：

表 20：**自动 QoR 建议流程运行属性**

属性	值	描述
RQS_FILES	RQS 文件名	允许您输入用户 RQS 文件。
AUTO_RQS	· 1：开启 · 0：关闭	启用自动 RQS 流程。
AUTO_RQS.DIRECTORY	<directory>	允许指定替代目录，以便将 RQS 文件复制到其中。
AUTO_RQS.SUGGESTION_RUN	生成 RQS 文件的实现运行名称	应用于综合父运行。

以下代码片段中显示了在 Tcl 中使用这些属性的示例：

```
set_property AUTO_RQS 1 [get_runs impl_1]
set_property RQS_FILES C:/temp/test.rqs [get_runs impl_1]
set_property AUTO_RQS.DIRECTORY C:/project_name/sources/rqs/impl_1
[get_runs impl_1]
set_property AUTO_RQS.SUGGESTION_RUN impl_1 [get_runs synth_1]
```

自动 QoR 建议和 ML 策略

如果在开始运行前存在 ML 策略，那么可使用 RQS_FILES 属性来设置 RQS 文件。在流程结束时，会将这些策略建议添加到新的 RQS 文件中。

如果不存在 ML 策略，则会在建议生成进程中，在 `<run_name>/MLStrategy` 目录中为用户自动生成这些策略。

注释：仅限 AMD UltraScale™ 和 AMD UltraScale+™ 支持 ML 策略流程。

自动 QoR 建议和增量编译

启用增量流程后，在后续步骤中只能应用对增量友好的新建议。这会限制建议可以执行的更改量，因此可能难以大幅改善时序。

如果在 `opt_design` 处应用建议，并且随后发起增量流程，则可应用影响更大的建议。由于此建议可能导致时序劣化，因此可能需要将其禁用。

如果自动增量流程由于参考检查点质量欠佳而决定使用默认流程，则可应用所有建议。

注释：仅限 UltraScale 和 UltraScale+ 支持的 QoR 建议和增量流程。

禁用不需要的建议

由于对自动添加到建议文件的建议缺少足够的控制，如果建议导致时序结果变差，那么可在 `opt_design` 之前的步骤中禁用 Tcl 脚本。以下代码片段提供了禁用 `RQS_TIMING-1` 建议的示例：

```
set_property ENABLED 0 [get_qor_suggestions RQS_TIMING-1*]
```

非工程模式

在此模式下创建 RQS 文件的过程与工程模式相同。即，编写建议文件，随后必须使用 `read_qor_suggestions` 将此文件添加到运行中。`read_qor_suggestions` 命令应在 `synth_design` 或 `opt_design` 之前运行。

此流程的等效 Tcl 命令如下所示：

```
read_vhdl <some_file>.vhd
read_qor_suggestions all_enabled_suggestions.rqs
synth_design -top <top> -part <part>
opt_design
...
route_design
report_qor_suggestions -file design_rqs_routed.rpt
write_qor_suggestions -force all_enabled_suggestions.rqs
```

为了减少实验，请找出最适合当前设计的具体建议。这样您即可专注于找出能普遍给各种设计带来良好回报的建议上。以下命令可在 `opt_design` 后启用建议，并在最终设计阶段之后将建议写入某个文件，以便在下一轮设计中回收利用。这是通过专注于 AUTO_RQS_FLOW 属性所生成的建议来达成的：

```
opt_design
report_qor_suggestions -file opt_report_qor_suggestions.rpt
set_property ENABLED 1 [get_qor_suggestions -filter
{GENERATED_AT==opt_design && SUGGESTION_SOURCE=="Current Run" && ENABLED &&
APPLICABLE_FOR==place_design && AUTO_RQS_FLOW}]
place_design
...
# Final command either route_design or phys opt design
report_qor_suggestions -file route_report_qor_suggestions.rpt
write_qor_suggestions -of_objects [get_qor_suggestion -filter {APPLIED ||
(AUTO_RQS_FLOW && APPLIED==0)}] -file postroute.rqs
```

注释：加载设计后，才能运行 `report_qor_suggestions` 命令。

增量流程中的 RQS

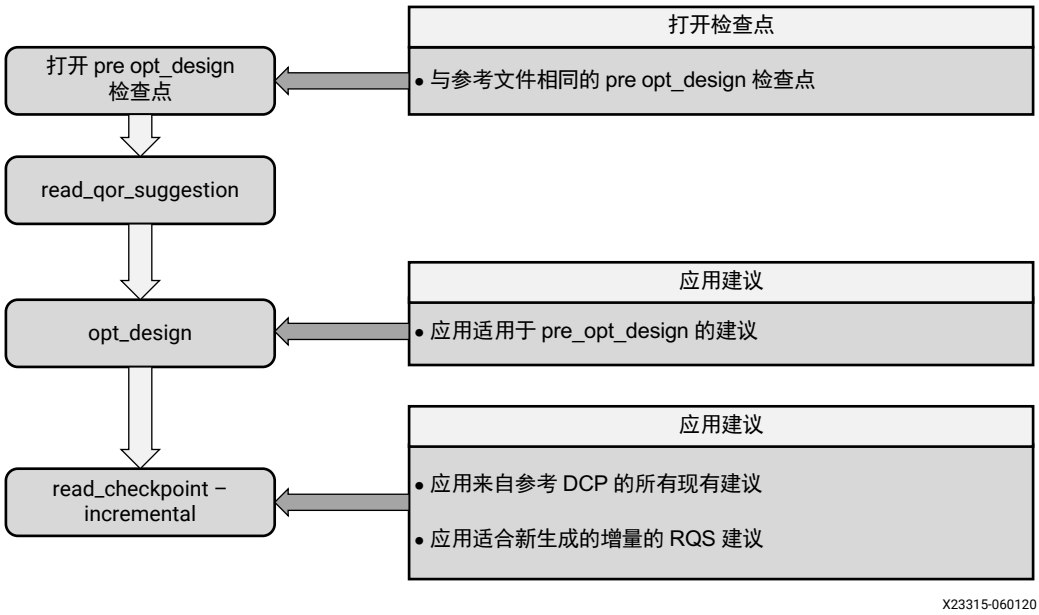
当设计非常接近时序收敛（通常 WNS 小于 -250 ps）时，可启用增量流程并包含 RQS 建议。这样即可利用增量流程和 RQS 建议来实现时序收敛并节省迭代时间。`report_qor_assessment` 用于在“Flow Guidance”（流程指南）部分中指示何时使用此流程。

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。

注释：仅限 AMD UltraScale™ 和 AMD UltraScale+™ 器件家族支持此流程。

流程其余部分将为您自动应用。Vivado 会通过区分新生成 (GENERATED) 的建议以及参考运行中已应用 (APPLIED) 的建议，来明确应用于流程中各阶段的具体建议。如下图所示：

图 230：**增量流程**



如果运行增量流程，那么对于参考中已应用的建议，无论是否已启用，都会从参考 DCP 中读取并应用这些建议。之所以忽略 ENABLED 属性，因为至关重要的是应尽可能复制参考检查点。

下一步会应用来自 RQS 文件的适合增量流程的建议。在 `read_qor_suggestions` 期间会自动启用这些建议，并且必须启用后才有效。在 `read_checkpoint -incremental` 期间（而不是在 APPLICABLE_FOR 阶段中）会应用这些建议。因此，在此阶段后不应读取或启用建议，否则会被忽略。RQS 文件中的任何不适合增量流程的新建议都将被忽略（将要应用的是在参考中已应用的现有不适合增量流程的建议）。

应用适用于 `opt_design` 的建议时，请谨慎处理。因为在执行此操作时，流程尚未意识到自身处于增量模式下，故而无法自动管理这些建议。对于参考中已应用的现有建议，您必须确保在增量运行中也同样应用这些建议，并且不应用任何新建议。如果倾向于应用这些建议，那么应更新参考。

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，在启动下一轮增量运行前，必须将所有建议（而不只是适合增量流程的建议）都导出至 RQS 文件。

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 RQA 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。
- 这些新生成建议必须从参考检查点生成。此项检查可确保建议不会影响时序已解决的路径（例如，布线后 `phys_opt_design`）。

运行该流程所需的命令示例如下所示：

- 参考：

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

- 增量流程：

```
# RQS-Incremental Run:
open_checkpoint <pre_opt.dcp>
read_qor_suggestions ./post_route.rqs
# opt_design directive must be same as the reference run
opt_design -directive {same directive as reference run}
read_checkpoint -incremental reference_routed.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.dcp
```

自动删除建议

为了防止建议过多累积，Vivado Design Suite 会对建议进行自动管理。生成新建议时，它将删除与先前生成的建议相同的建议。

以 Tcl 或文本格式来查看建议

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_qor_suggestions`。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 中写出建议，请使用以下命令：

```
write_qor_suggestions -tcl_output_dir <outputDir>
```

通过运行此命令即可将 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。

对于为实现命令所生成的 Tcl 脚本，最好手动管理这些脚本与设计约束的集成。对于综合 Tcl 脚本，在工程模式下可将其添加到设置的约束中（将文件筛选工具更新为 Tcl 以进行查看），或者在非工程模式下可使用 `read_xdc -unmanaged <Tcl file>` 访问这些脚本。

当对象显示在 Tcl 中时，您应负责维护 Tcl 以移除不再需要的对象，并且仅应用这些对象要应用的建议。您必须将任何新生成的 Tcl 脚本追加到现有 Tcl 脚本的末尾。

`report_qor_suggestions` 将不再报告使用 Tcl 输入的建议。

调试 QoR 建议

有时，有必要更深入地研究建议，以了解其产生原因。通常，应调查以下几个方面：

- 使用率和网表建议
  - 这通常需要调查建议所适用的网表中的单元。这可以通过选择对象并打开板级原理图来完成。

- 时钟和时序建议
  - 这通常需要调查与建议相关的时序路径。时序报告构成 QoR 建议报告的一部分。
- 拥塞建议
  - 这通常需要调查单元及其位置。

要获取建议中目标对象的单元列表，请按照以下步骤写出 Tcl 文件。编写 Tcl 建议文件时，每项建议都会在文件中使用其 ID 进行标记。用户可从中复制建议的 [get_cells <>] 或 [get_nets <nets>] 节。以下是一个 Tcl 示例，可用于进一步探索对象：

```
set cells <cells from Tcl file> report_timing -from $cells
report_design_analysis -of [get_timing_paths -from $cells -max_paths 100] -
name -RQS show_objects -name RQS_objs $cells select_objects [get_sites -of
$cells]
```

受支持的系列和流程

下表汇总了该版本中 QoR 建议支持的器件系列和设计流程。

表 21：**QoR 建议支持汇总**

项目	支持状态
支持的系列	UltraScale、UltraScale+ 和 Versal 自适应 SoC
流程	工程模式和非工程模式
DFX	支持
IP OOC 综合	不支持
Vitis	不支持

支持的建议

Vivado 可生成 100 余项 QoR 建议，其中 80 余项建议能自动执行，无需用户对约束或 RTL 进行任何编辑。要查看可生成的所有建议的列表，请运行 `get_qor_checks` 命令。您可检查随附于返回的对象的属性。此命令返回的对象并非生成的建议，无法写入这些对象本身。

`-family` 开关可用于仅显示适用于某一系列的建议。以下示例演示了如何查看当前打开的设计系列可用的建议：

```
get_qor_checks -family [get_property FAMILY [get_parts [get_property PART
[current_design]]]]
```

`-filter` 开关允许用于基于对象属性进行筛选。可供筛选的实用属性有：

- CATEGORY
- AUTO
- INCR_FRIENDLY

此命令的输出应使用 Tcl 来处理。以下示例演示例如如何查看所有 AUTO 检查的 ID 和描述：

```
foreach sugg [lsort -dict [get_qor_checks -filter {AUTO==1}]] {
  set ID [get_property ID $sugg]
  set DESCRIPTION [get_property DESCRIPTION $sugg]
  puts "[format %-16s $ID]: $DESCRIPTION "
}
```

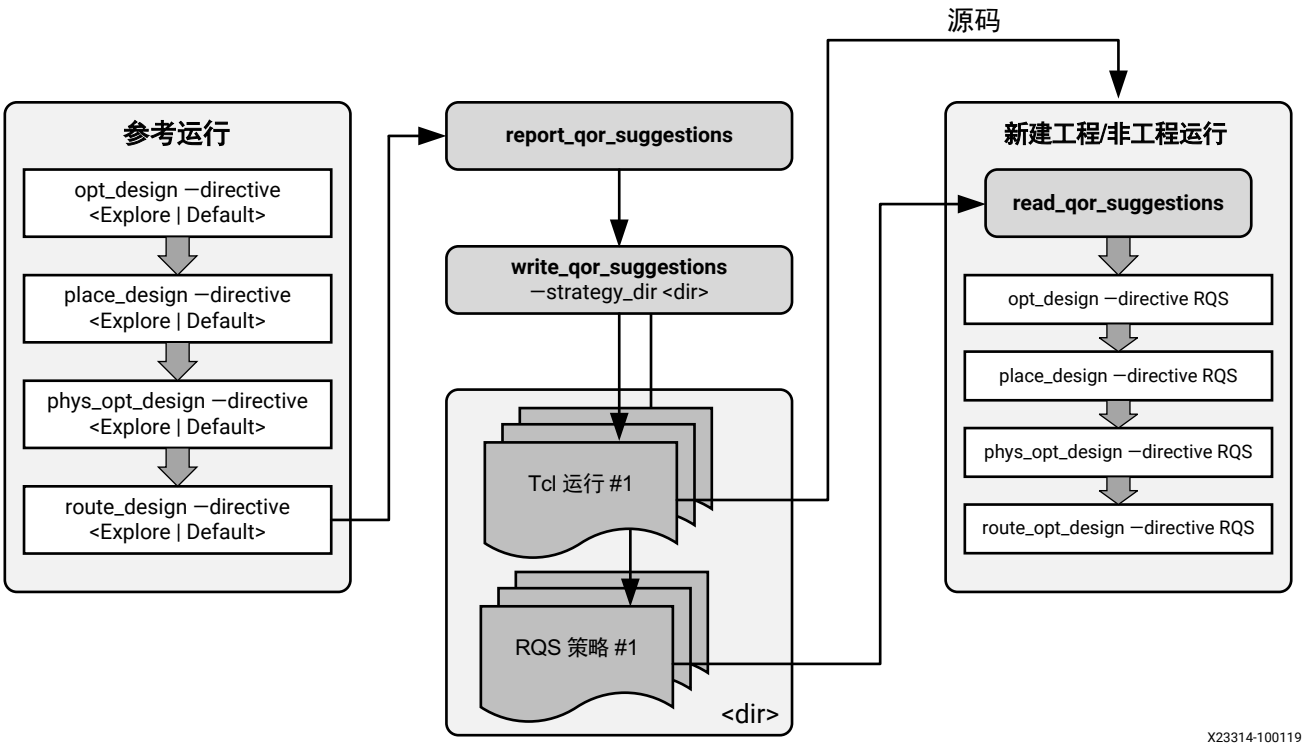
策略建议

注释：仅限 UltraScale 架构和 UltraScale+ 架构支持策略建议。

典型时序收敛策略需运行大量实现策略并选取其中最佳的策略以供在实验室内应用。ML 策略同样可选，且只需您运行 3 项策略即可达成类似的 QoR 收益。这些策略使用机器学习来检验布线后设计的各项功能特性，以便预测相同设计上不同策略的性能。在 `report_qor_suggestions`（和 `write_qor_suggestions`）所生成的 RQS 文件中会捕获最佳的 3 项策略，以供后续应用。这样即可显著降低服务器功耗。

如果在实现命令上，指令设为 RQS，那么此命令会为该指令和其他工具命令选项引用 RQS 文件。此流程如下图所示：

图 231：**策略建议流程**



此流程包含 4 个关键点：

1. `report_qor_suggestions` 命令必须在使用 `Default` 或 `Explore` 指令生成的完全布线的设计上运行。如需了解相关要求的完整详细信息，请参阅 ML Strategy Availability。

2. `write_qor_suggestions -strategy_dir <dir>` 命令会在指定的目录中生成所需的 RQS 文件。默认情况下，可生成 3 项策略。生成的每项策略各有一个 RQS 文件，其中包含所有建议对象以及策略建议对象。使用 `write_qor_suggestions -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，请使用以下命令增加数量：

   ```
   report_qor_suggestions -max_strategies <n>
   ```

3. 生成的 RQS 文件必须读入新的实现运行。
4. 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。

策略建议的工程应用

在工程模式下，在下列步骤中会捕获生成和应用建议的进程：

1. 生成包含策略的 QoR 建议文件（以及其他可选 QoR 建议）。
2. 创建新的实现运行、读取 QoR 建议文件并将指令设置为 RQS。

在 IDE 中集成了多种方式用于完成步骤 1。这些方式因使用的流程以及添加到 RQS 文件中的其他 QoR 建议而异。下表描述了每一项功能特性的工作方式及其对于非策略 QoR 建议的影响。

表 22：**生成 ML 策略的过程**

流程	RQS 文件创建	其他 QoR 建议
标准实现运行	右键单击“Design Runs”（设计运行）窗口。	默认会添加 APPLIED 建议。 可选添加新生成的 AUTO 建议。
启用自动 RQS	自动。	默认会添加 APPLIED 建议。 默认会添加新的 AUTO 建议。
来自未完成的设计的 QoR 建议报告	编写建议期间，会自动编写包含选定建议的策略。	由用户逐一选中。
智能设计运行	自动	包含来自阶段 1 IDR 的 APPLIED 建议。 等同于阶段 2 运行。
布线后 Tcl 挂钩中的 QoR 建议报告	手动	可由用户逐一选中。使用 `-strategy_dir ./MLStrategy`。

以上每个选项都会在步骤 2 中生成一种方式来创建和设置运行。在每个项中，都会在运行目录中创建 `MLStrategy` 目录。其中包含 3 个 RQS 文件。看到这些文件时，会在“Design Runs”右键单击菜单中启用 `create_rqs_runs` 选项。

运行该选项时，它会自动创建 3 个新的实现运行，并将其与参考运行相连。每次参考运行仅允许 3 项 ML 策略。如果要重新创建这些策略，请首先将其删除，然后重新运行。次要设计更改应该无需重新生成这些策略。

以下显示了从实现运行 `impl_1` 创建新 ML 策略的等效 Tcl 命令：

```
create_rqs_runs -reference_run [get_runs impl_1]
```

策略建议的非工程应用

在非工程模式下，`-strategy_dir` 目录中提供了 1 个 Tcl 脚本示例。此脚本显示了 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 `opt_design` 阶段之前中加载到存储器中。其中不包含任何检查点报告或写入操作。

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，AMD 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平铺，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。

您可修复最严重的问题或者最常见的问题，从而最大程度改进结果。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”（报告 > 时序 > 创建裕量直方图）命令可以提供离群路径视图。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他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 site（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：请将详细的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：

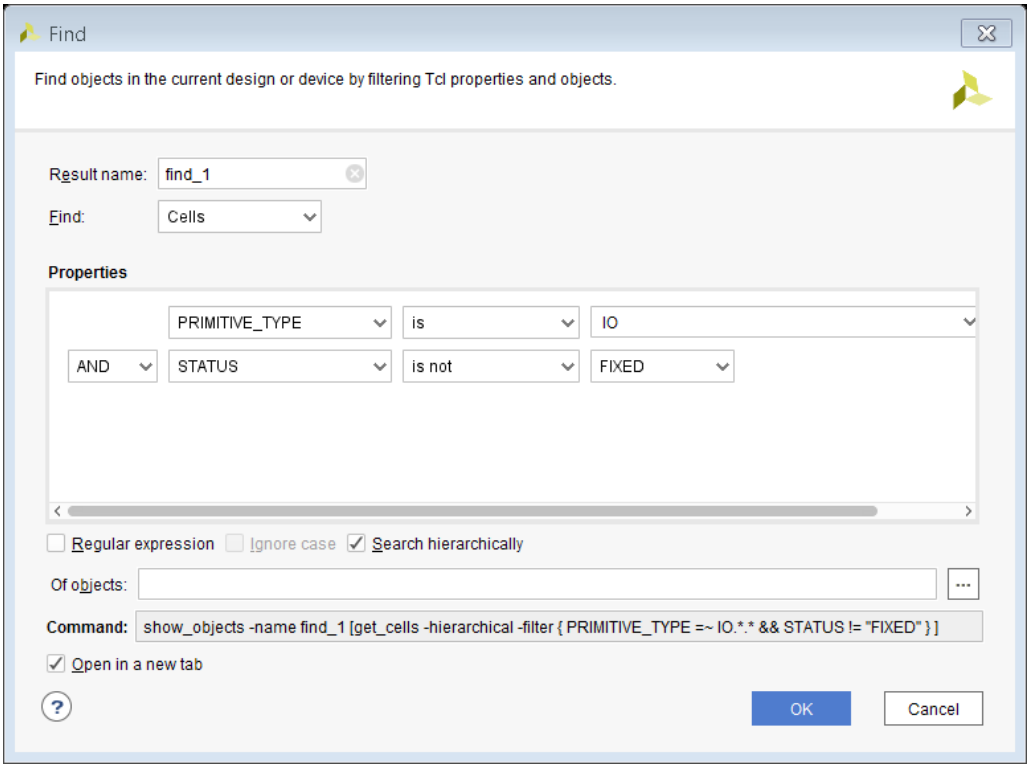
1. 打开 2 个实现运行：
   a. 1 个对应满足时序的运行。
   b. 另 1 个对应不满足时序的运行。

   提示：在有多台显示器的计算机上，选择“Open Implementation in New Window”（在新窗口中打开实现）以在新窗口中打开设计。
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`，这样即可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
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

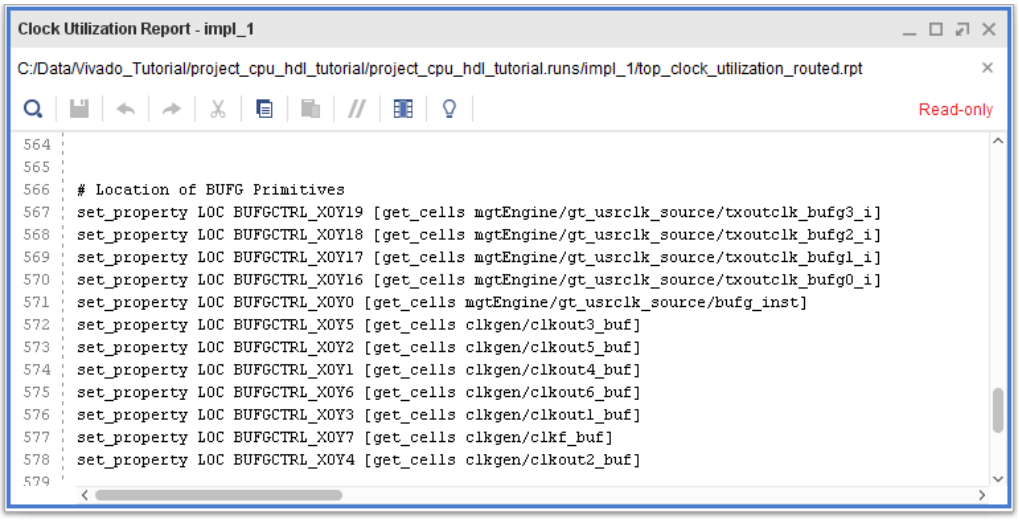
复查设计中的单元布局。比较 2 份 I/O 报告，以复查 I/O 布局和 I/O 标准。确保所有 I/O 均已完成布局。通过简单搜索可以发现不含固定布局的所有 I/O，如下图所示。

图 232：**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。“Clock Utilization”（时钟使用率）报告可列出时钟树驱动程序的布局，如下图所示。

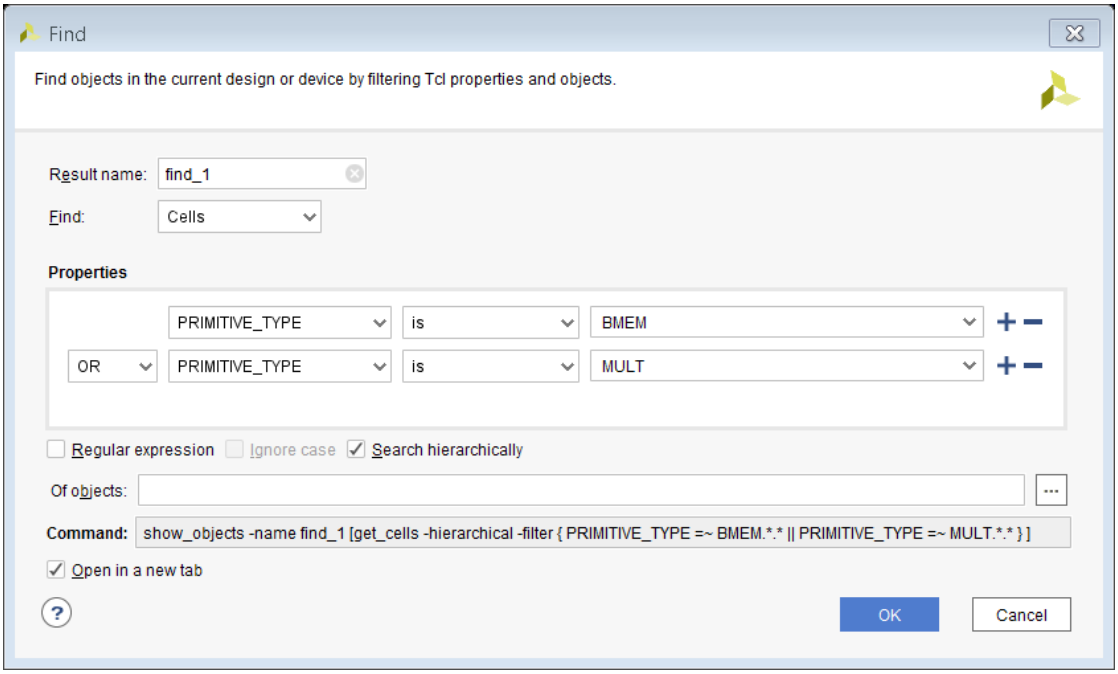
图 233：**时钟位置**



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”（编辑 > 查找）以列出实例。

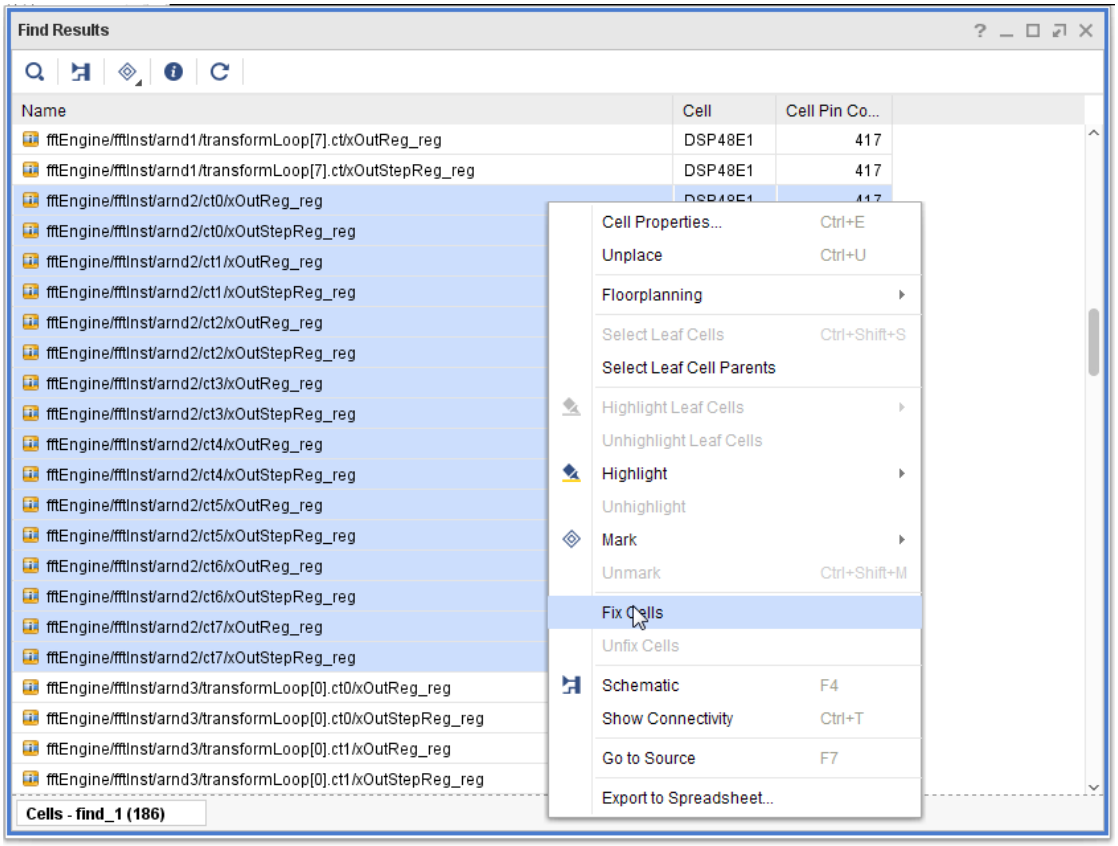
图 234：**DSP 或 RAM**



添加布局约束

固定逻辑以便向 XDC 添加布局约束。

1. 从查找结果中选择宏。
2. 右键单击并选中“Fix Cells”（固定单元），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网表版本升级所导致的变化。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 DCP 即可完成复用设置。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，请尝试分层布局规划。

分层布局规划

分层布局规划支持您将一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。分层布局规划相比于门级布局规划具有如下优势：

- 分层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。

在分层布局规划中支持：

- 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

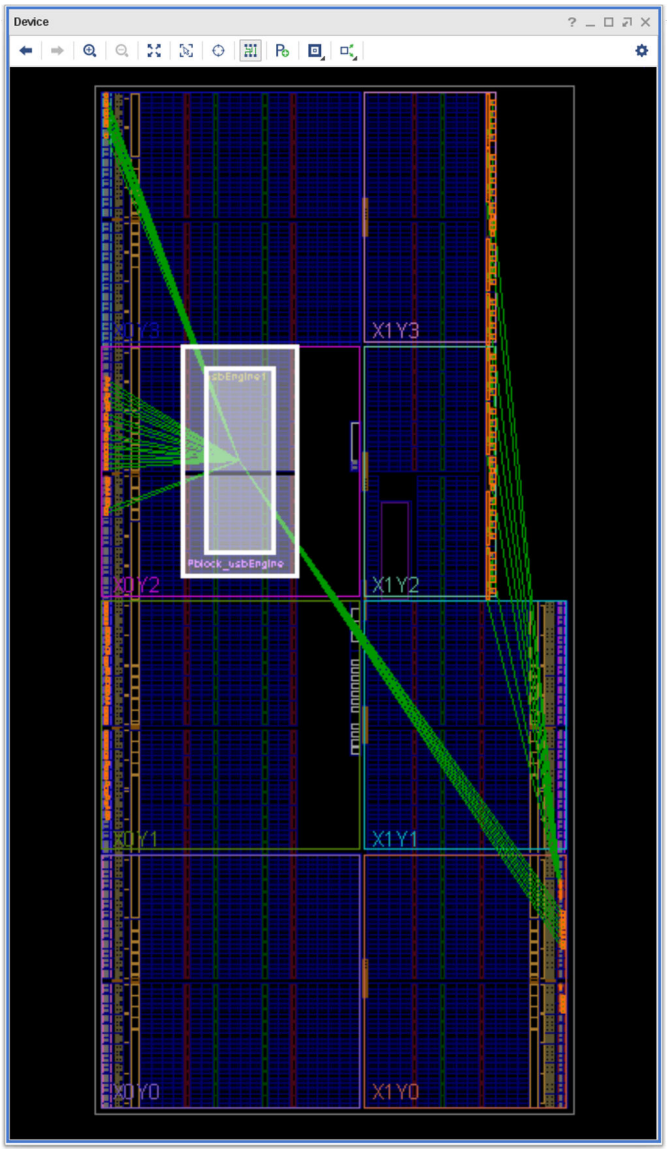
手动单元布局可充分发挥器件性能。使用此技巧时，设计师通常只能将其用于小块设计。可将少量逻辑布局到高速 I/O 接口周围，或者可手动布局块 RAM 和 DSP。手动布局可能比较慢。

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，应明确最终管脚分配。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。如果管脚分配导致块被拆分，请考虑修改管脚分配或 RTL。

图 235：**导致设计拆分的 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：

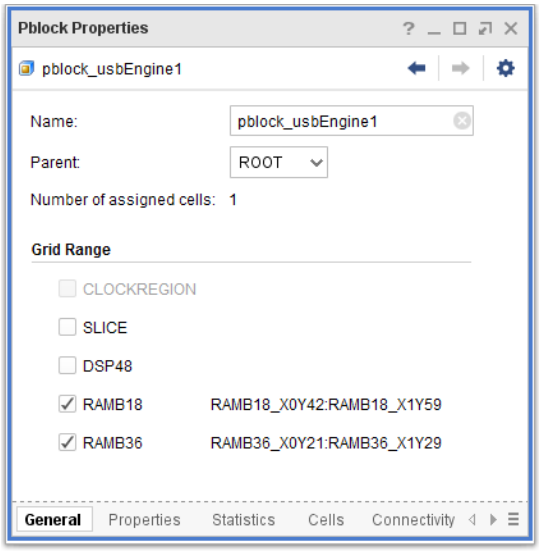
```
create_pblock Pblock_usbEngine
add_cells_to_pblock [get_pblocks Pblock_usbEngine] [get_cells -quiet [list
usbEngine1]]
resize_pblock [get_pblocks Pblock_usbEngine] -add
{SLICE_X8Y105:SLICE_X23Y149}
resize_pblock [get_pblocks Pblock_usbEngine] -add {DSP48_X0Y42:DSP48_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add
{RAMB18_X0Y42:RAMB18_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add
{RAMB36_X0Y21:RAMB36_X1Y29}
```

注释：默认情况下，在布局规划阶段后的整个实现流程中，都将 Pblock 视为软核 Pblock。要强制使用硬核 Pblock，应将 Pblock 属性 IS_SOFT 设为 0：

```
set_property IS_SOFT 0 [get_pblocks Pblock_usbEngine]
```

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，请禁用其他 Pblock 网格。

图 236：**Pblock 网格**



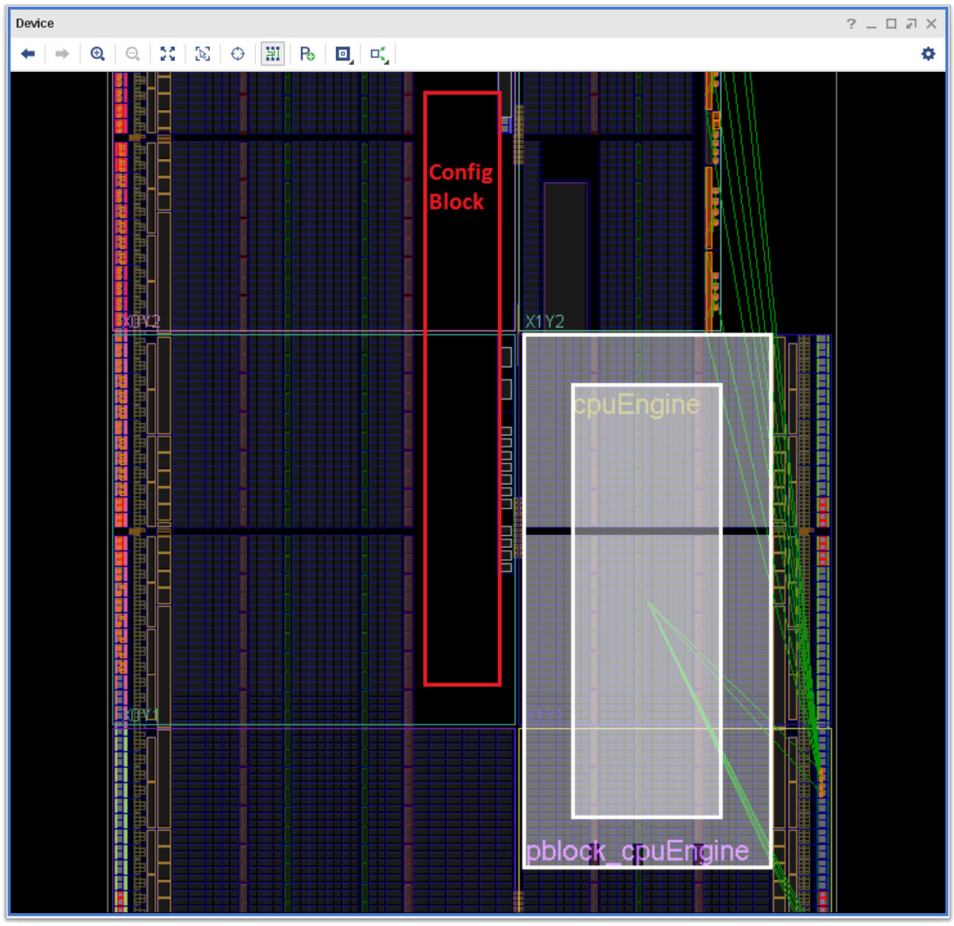
生成的 XDC 命令可用于定义简化的 Pblock：

```
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（分片）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 237：**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 管脚分配 (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。

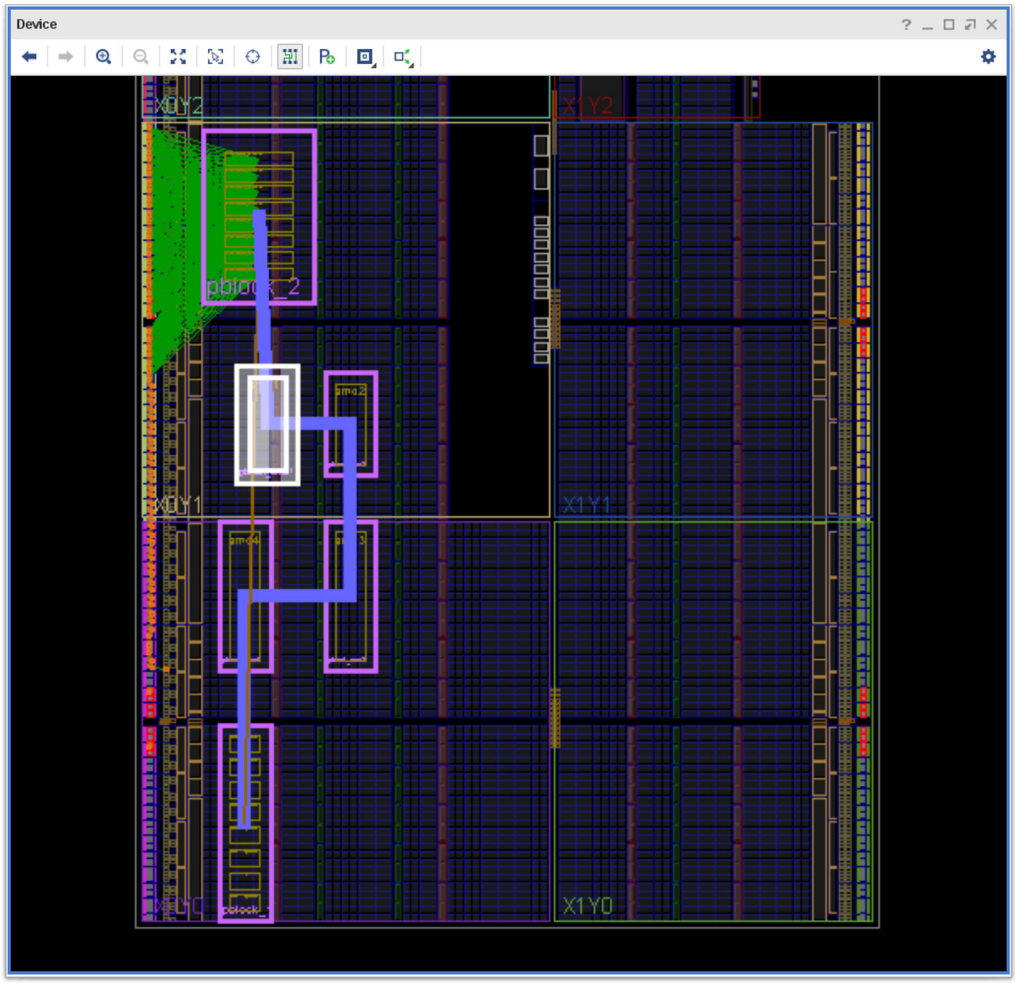
分析期间，Pblock 填充可能超过 100%，但在实现期间不会发生此现象。过度填充 Pblock 会导致在器件上其大小减小。这项实用的技巧旨在帮助您概览设计顶层块的相对大小及其在器件上的占用方式。

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。

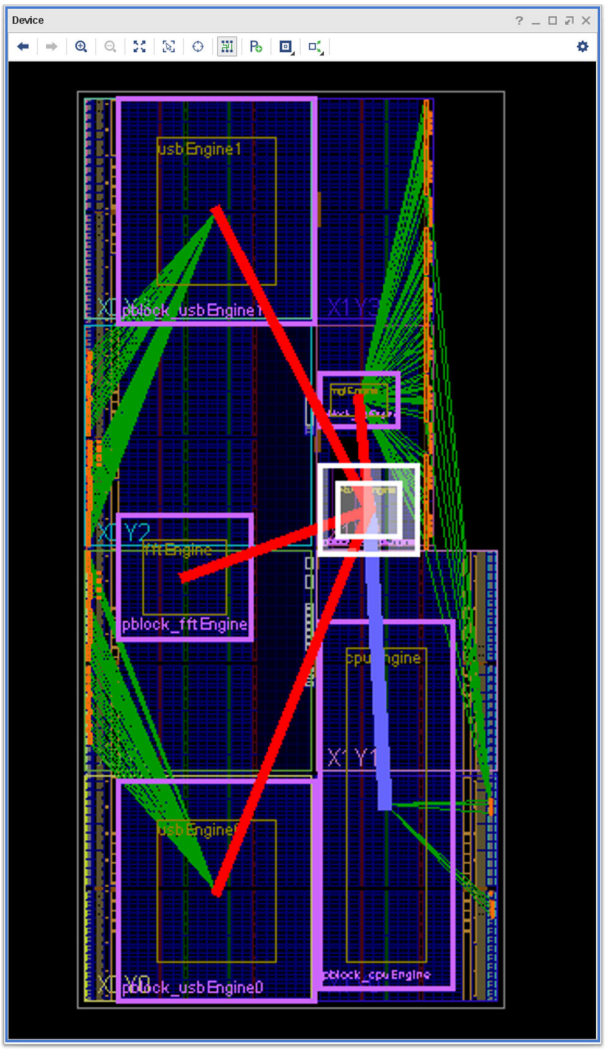
“Data Path Top Level Floorplan”（数据路径顶层布局规划）显示了设计的顶层块之间数据流动的方式。每个块都仅与 2 个近邻通信。绿线显示与单个块通信的布局精良的 I/O。

图 238：**Data Path Top Level Floorplan**



“Control Path Floorplan”（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，才能与所有其他负载进行通信。

图 239：**Control Path Floorplan**



分析使用率统计数据

实现问题的常见原因之一是未考量显式和隐式物理约束。例如，管脚分配 (pinout) 在逻辑布局上变为显式物理约束。slice（分片）逻辑在大部分器件中都是一致的。但如下专用资源表示的是隐式物理约束，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
- 千兆位收发器
- DSP slice
- 块 RAM
- 时钟管理块，如 MMCM
- 时钟缓冲器，如 BUFG

在为设计的其余部分设计接口时，大量耗用这些专用资源的块可能必须围绕器件分散排列并采用物理约束布局布线。此外，Pblock 为显式物理约束，用于为指定逻辑定义允许的布局区域。通过搭配使用以下方法来分析器件上的块资源使用率：

- 使用率报告
- 网表属性
- Pblock 属性

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 AMD 7 系列 FPGA 设计上。AMD 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 互连结构中的单元布局。对互连结构中的其他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，目的是提升实现运行的一致性。

表 23：**用于逻辑布局的约束**

约束	用途	注释
LOC	将门电路或宏布局在特定 site（站点）上。	SLICE 站点具有子站点（称为 BEL 站点）。
BEL	在 slice（分片）中指定用于基本元件的子站点。	

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，请参阅《Vivado Design Suite 用户指南：实现》(UG904) 中的“修改布局”。

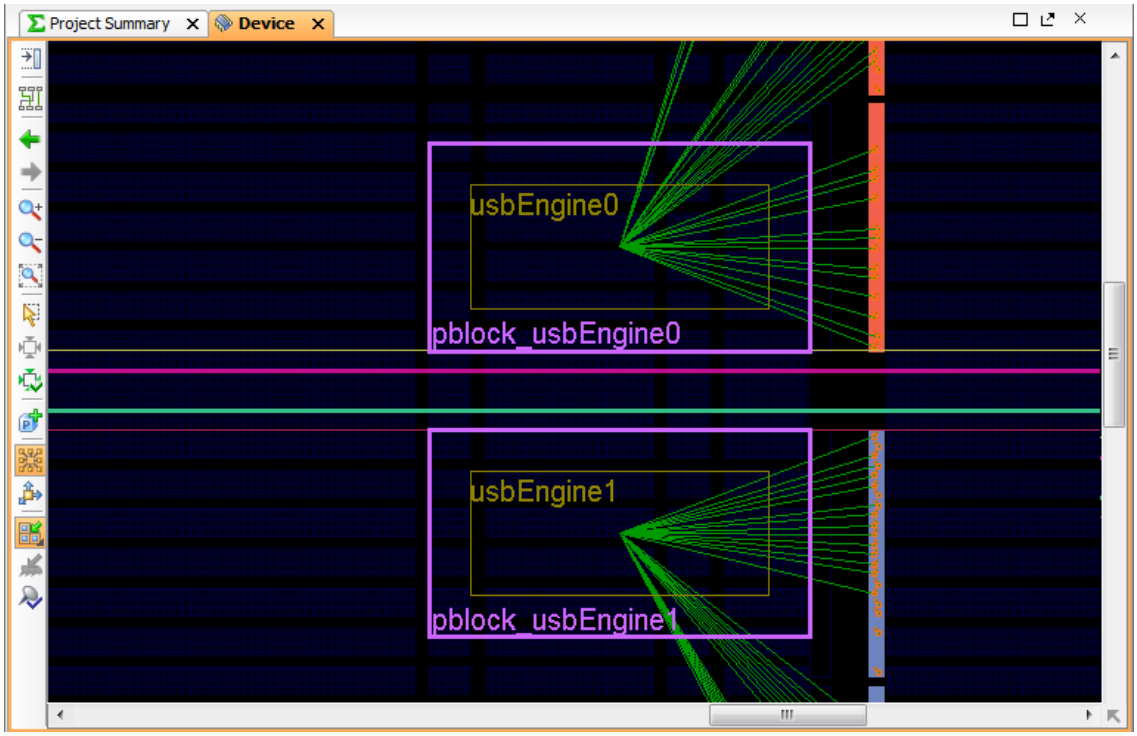
建议：完成 I/O 布局后，请使用层级 Pblock 布局规划作为用户控制的布局的起点。当发现 Pblock 无效时，应使用手动布局逻辑。

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) 部件，存在额外的注意事项。SSI 部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 交汇时会发生延迟惩罚。

构造设计、生成管脚分配和布局规划时，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 240：**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 部件布局逻辑时，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 部件内，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。

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。如果存在保持时序违例，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。通常导致保持时间检查错误的原因主要是 `set_multicycle_path` 约束不正确（未指定 `-hold`）。其他情况下，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，AMD 建议您复查此特定电路的时钟设置架构。如需了解更多信息，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“识别时序违例的根源”。

如果设计可满足布局后建立时序要求，但无法满足布线后建立时序要求，则可能出现此问题。为了修复保持时间违例，布线器会添加布线绕行，使用 `report_design_analysis` 命令搭配 `-show_all` 选项即可查看由此所导致的路径延迟。下图显示的 report_design_analysis 报告示例中包含“Hold Fix Detour”（保持修复绕行）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 241：**含“Hold Fix Detour”的“Report Design Analysis”**

Paths	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Hold Fix Detour	Combir
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 500 ps）。

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：在失败的最差建立时间路径上运行 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他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。

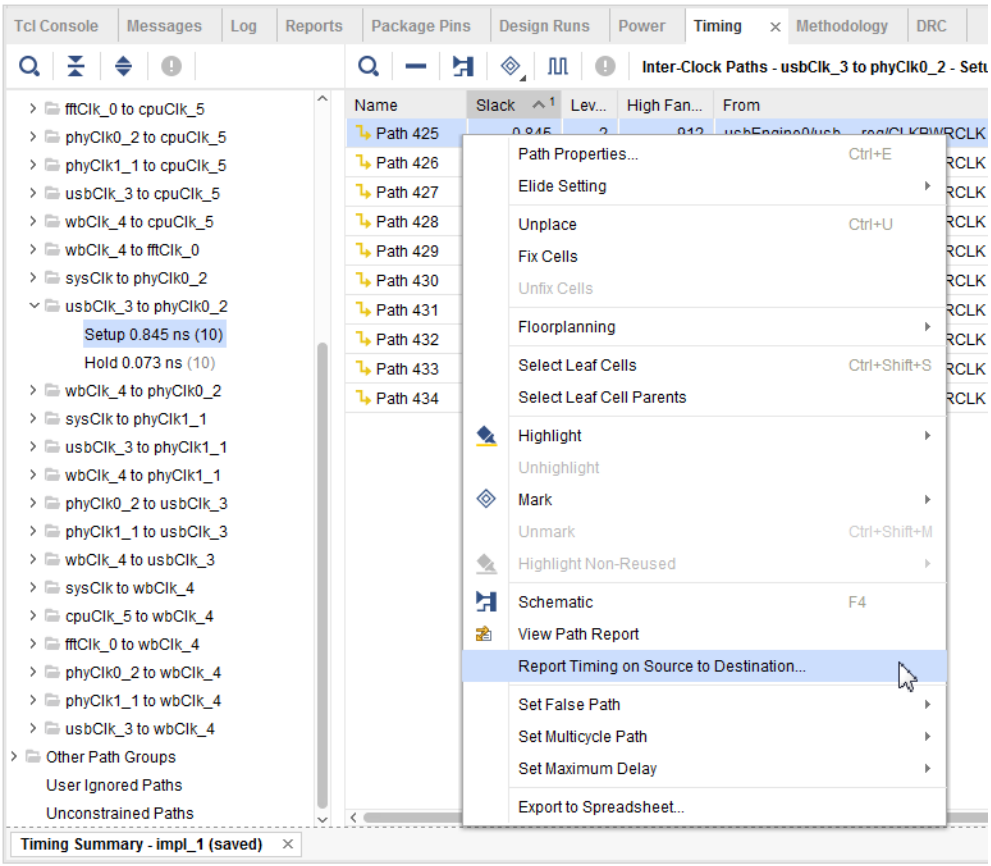
3. 请运行 `route_design` 和 `report_timing_summary`。

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：在失败的最差建立时间路径上运行 report_timing -min

请复查该路径的保持时间以判定失败的最差建立时间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”（报告从源到目标的时序）。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 242：**在特定路径上运行时序报告**

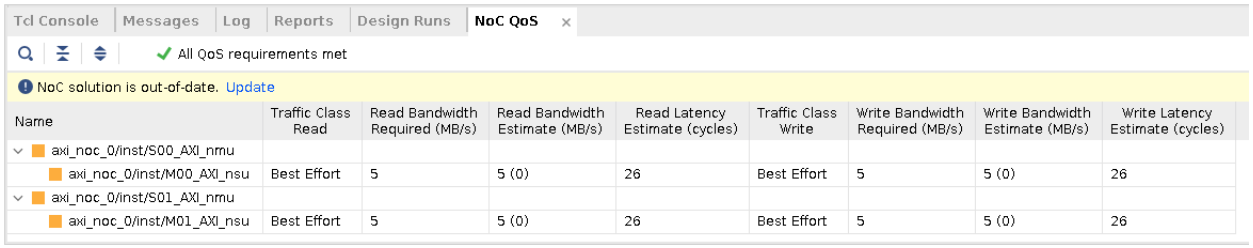


第 8 章

在 Versal 器件中执行 NoC 服务质量分析

AMD Vivado™ 中的服务质量 (QoS) 用于将片上网络 (NoC) 编译器生成的当前 NoC 解决方案估算所得 QoS 与 AXI NoC IP 和/或 AXI4-Stream NoC IP 中指定的 QoS 要求进行对比。一旦 NoC 解决方案过时，就需要调用 NoC 编译器并生成新的 NoC 解决方案以更新 QoS 报告。QoS 报告将在其功能区内显示为已过期（请参阅下图）。单击“Update”（更新）链接将调用 NoC 编译器以重新生成 NoC 解决方案。

图 243：**已过期的 Vivado NoC QoS 报告**



Name	Traffic Class Read	Read Bandwidth Required (MB/s)	Read Bandwidth Estimate (MB/s)	Read Latency Estimate (cycles)	Traffic Class Write	Write Bandwidth Required (MB/s)	Write Bandwidth Estimate (MB/s)	Write Latency Estimate (cycles)
axi_noc_0/inst/S00_AXI_nmu								
axi_noc_0/inst/M00_AXI_nsu	Best Effort	5	5 (0)	26	Best Effort	5	5 (0)	26
axi_noc_0/inst/S01_AXI_nmu								
axi_noc_0/inst/M01_AXI_nsu	Best Effort	5	5 (0)	26	Best Effort	5	5 (0)	26

注释：在 IP integrator 中，可使用 `validate_bd_design` Tcl 命令来更新 NoC 解决方案，在实现工具/流程中，则可使用 `update_noc_qos` Tcl 命令。

可能导致 NoC 解决方案过期的操作包括：

- 在 IP integrator 中：
  - 在 IP integrator 中更新 AXI NoC IP QoS 要求
  - 更改 IP integrator 块设计
  - 在 NoC 视图中更改 NoC 主/从单元 (NMU/NSU) 分配
- 在实现工具/流程中：
  - 取消 NMU/NSU 实例布局
  - 影响 NMU/NSU 布局的 XDC 物理约束
  - 更改连接到 NMU/NSU 的逻辑信号线

对于 NoC 中指定的每项连接，NoC QoS 报告都会为读写传输事务提供如下信息报告：

- “Traffic Class”（流量类）：IP integrator 中的 NoC 上指定的用于连接的流量类
- “Bandwidth Required”（必需带宽）：IP integrator 中的 NoC 上指定的用于连接的带宽 (MB/s)
- “Bandwidth Estimate”（带宽估算）：NoC 编译器为当前 NoC 解决方案估算的带宽 (MB/s)
- “Latency Estimate”（时延估算）：NoC 编译器为当前 NoC 解决方案估算的结构时延（以 NoC 时钟周期数为单位）

重要提示！对于 QoS 报告中的时延估算，有 2 个重要事项需要注意。“Latency Estimate”属于结构时延，而非动态时延。它报告的是传输事务的最短往返时间。“Latency Estimate”是在 QoS 报告中以 NoC 时钟周期数来估算的，而 AMD Performance Traffic Generator IP 在仿真中报告的时延是以 AXI 时钟周期数来计算的。由于 Vivado 中的 QoS 报告采用的是结构估算，因此您的设计必须利用实际流量激励来确认后才能完成设计验收。

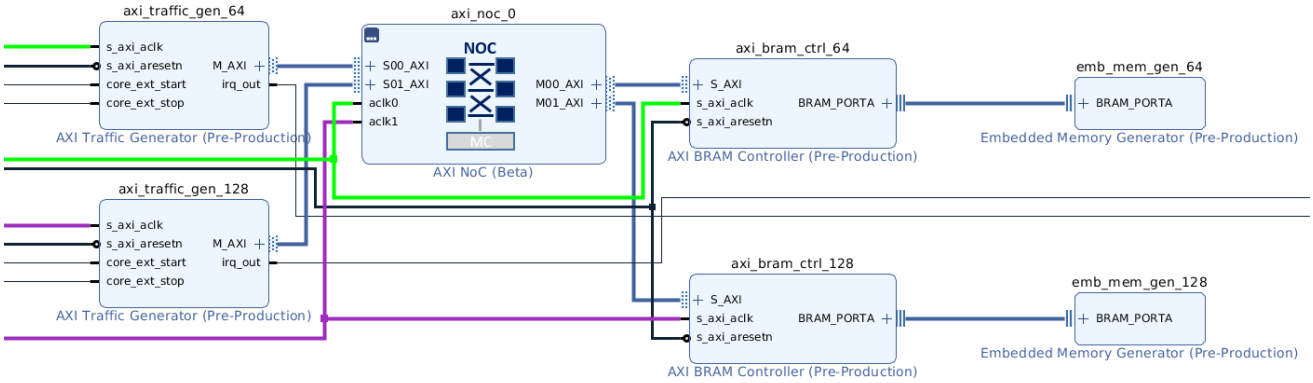
Vivado NoC QoS 报告示例

以下示例显示了 IP integrator 设计中包含 2 个 AXI Traffic Generator 和 2 个 AXI 块 RAM 控制器以及关联嵌入式存储器的部分。此示例用于演示您如何调整自己的数据宽度和 PL 时钟频率以实现相同带宽。`axi_traffic_gen_64`、`axi_bram_ctrl_64` 和 `emb_mem_gen_64` 均采用 64 位数据宽度，并连接至 200 MHz 时钟（绿色高亮）。`axi_traffic_gen_128`、`axi_bram_ctrl_128` 和 `emb_mem_gen_128` 均采用 128 位数据宽度，并连接至 100 MHz 时钟（紫色高亮）。

`axi_traffic_gen_64` 通过 NoC 连接至 `axi_bram_ctrl_64`，而 `axi_traffic_gen_128` 则通过 NoC 连接至 `axi_bram_ctrl_128`。

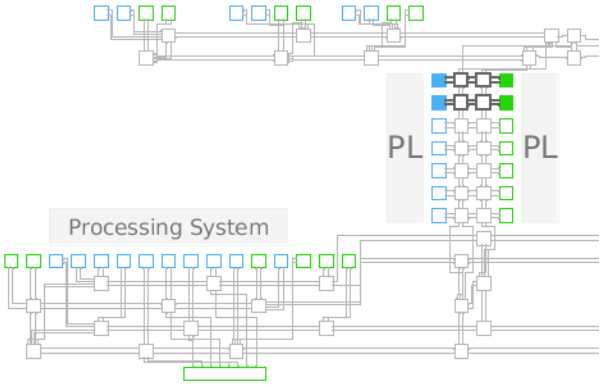
每条连接上的读写所需带宽均已在 NoC 上设置为 1000 MB/秒。

图 244：**IP integrator 块设计示例**



在 IP Integrator 中确认设计后的初始 NoC 解决方案如下图所示，每条 NoC 连接的布线都穿过横向 NoC。

图 245：**初始 NoC 解决方案**



下图所示“QoS Report”（QoS 报告）显示已满足带宽要求，报告显示每条连接的结构时延均为 26 个 NoC 时钟周期。

图 246：**初始 NoC QoS 报告**

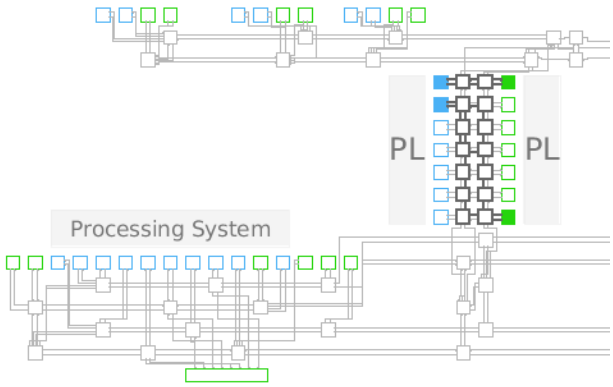
Tcl Console | Messages | Log | Reports | Design Runs | NoC QoS

All QoS requirements met

Name	Traffic Class Read	Read Bandwidth Required (MB/s)	Read Bandwidth Estimate (MB/s)	Read Latency Estimate (cycles)	Traffic Class Write	Write Bandwidth Required (MB/s)	Write Bandwidth Estimate (MB/s)	Write Latency Estimate (cycles)
axi_noc_0/inst/S00_AXI_nmu								
axi_noc_0/inst/M00_AXI_nsu	Best Effort	1000	1000 (0)	26	Best Effort	1000	1000 (0)	26
axi_noc_0/inst/S01_AXI_nmu								
axi_noc_0/inst/M01_AXI_nsu	Best Effort	1000	1000 (0)	26	Best Effort	1000	1000 (0)	26

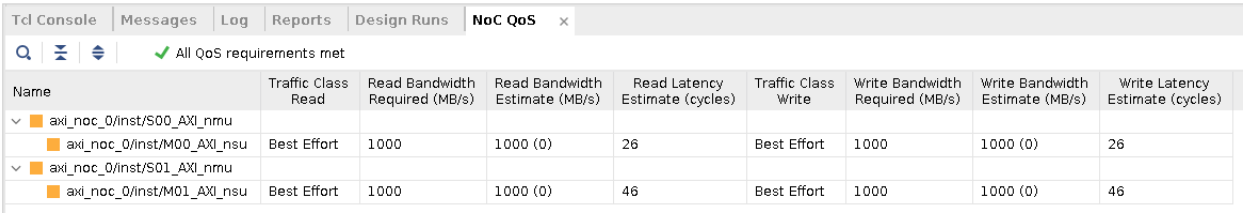
IP Integrator 中的“NoC View”（NoC 视图）支持 NMU/NSU 分配，可在“QoS Report”（QoS 报告）中观察 NoC 解决方案更新后的更改。在 NoC 视图中，`axi_noc_0/inst/M01_AXI_nsu` 已从 `axi_noc_0/inst/S01_AXI_nmu` 移至更远处，以便创建 1 条穿越更多 NoC 交换机的更长的路径。生成的 NoC 视图如下图所示。

图 247：**含 NSU 手动分配的 NoC 解决方案**



对于生成的 NoC QoS，带宽予以保留，但路径的结构时延已从 26 个 NoC 时钟周期增加至 46 个 NoC 时钟周期，如下图所示。

图 248：**含 NSU 手动分配的 NoC QoS**



Name	Traffic Class Read	Read Bandwidth Required (MB/s)	Read Bandwidth Estimate (MB/s)	Read Latency Estimate (cycles)	Traffic Class Write	Write Bandwidth Required (MB/s)	Write Bandwidth Estimate (MB/s)	Write Latency Estimate (cycles)
axi_noc_0/inst/S00_AXI_nmu								
axi_noc_0/inst/M00_AXI_nsu	Best Effort	1000	1000 (0)	26	Best Effort	1000	1000 (0)	26
axi_noc_0/inst/S01_AXI_nmu								
axi_noc_0/inst/M01_AXI_nsu	Best Effort	1000	1000 (0)	46	Best Effort	1000	1000 (0)	46

AMD

附录 A

时序方法检查

本附录详述了本工具执行的每一项方法论检查，解释了触发每一项检查的原因以及解决办法。

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>。

描述

Vivado Design Suite 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 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。

解决方案

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。

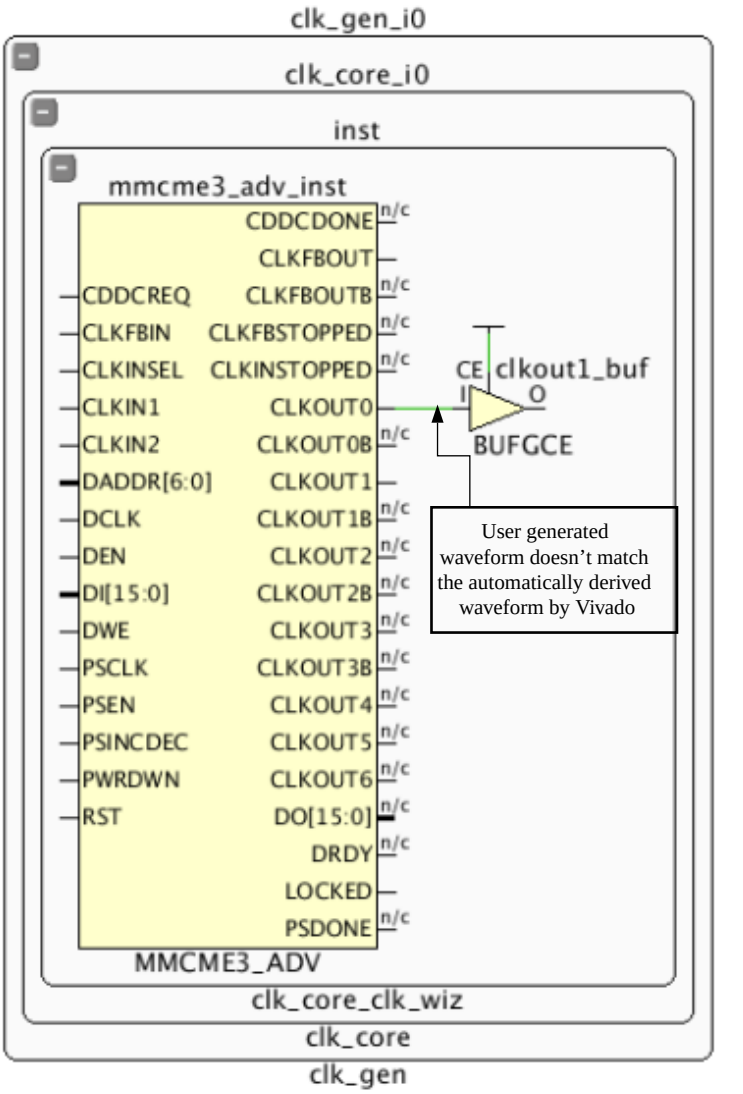
示例

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/
mmcme3_adv_inst/CLKOUT0]
```

图 249：**时钟修改块上的无效时钟波形**



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（不含时序弧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。如果在逻辑路径中间定义基准时钟，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程序管脚上创建基准时钟。否则可能导致硬件故障。

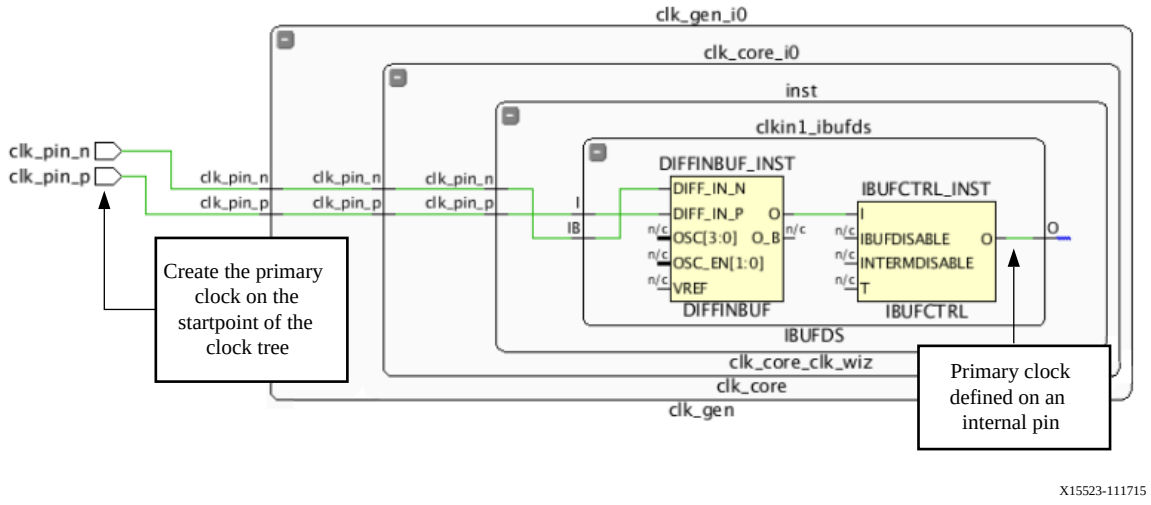
解决方案

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。差分输入缓冲器的基准时钟定义应布局在顶层端口 `clk_pin_p` 上。

图 250：**内部管脚上的基准时钟无效**



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

如果违例与用户时钟有关，请修改约束以移除 CMB 输出上的 `create_clock` 约束。如需强制设置自动生成时钟的名称，AMD 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他信息。

如果时钟是由 Clocking Wizard 生成的，请验证“Input Clock Information” → “Primary Input Clock”（输入时钟信息 > 基准输入时钟）。如果该向导的输入时钟直接连接到顶层 I/O 端口，请使用单端时钟使能管脚，如果输入时钟来自另一个 Clocking Wizard 实例，那么请使用全局缓冲器。

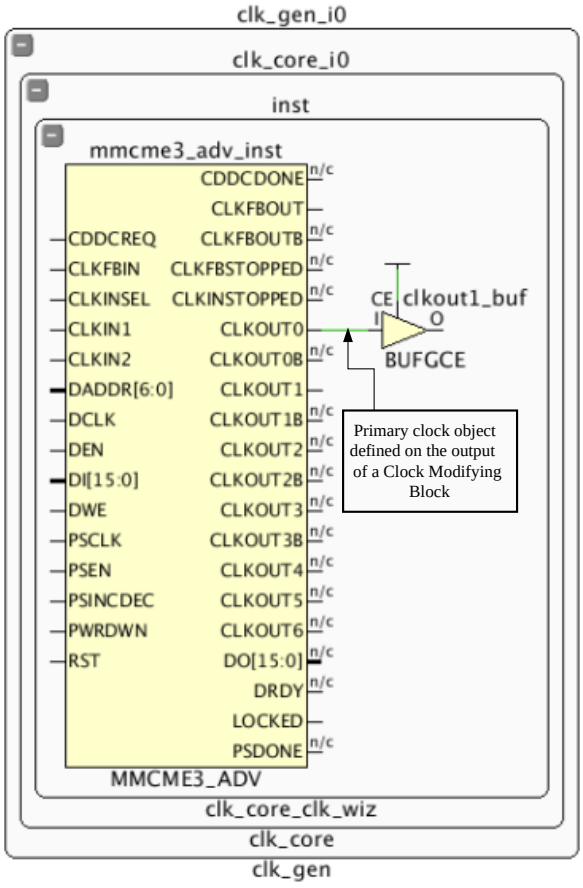
示例

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 251：**时钟修改块上的基准时钟无效**



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。如果在覆盖传入时钟定义的下游定义基准时钟，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。

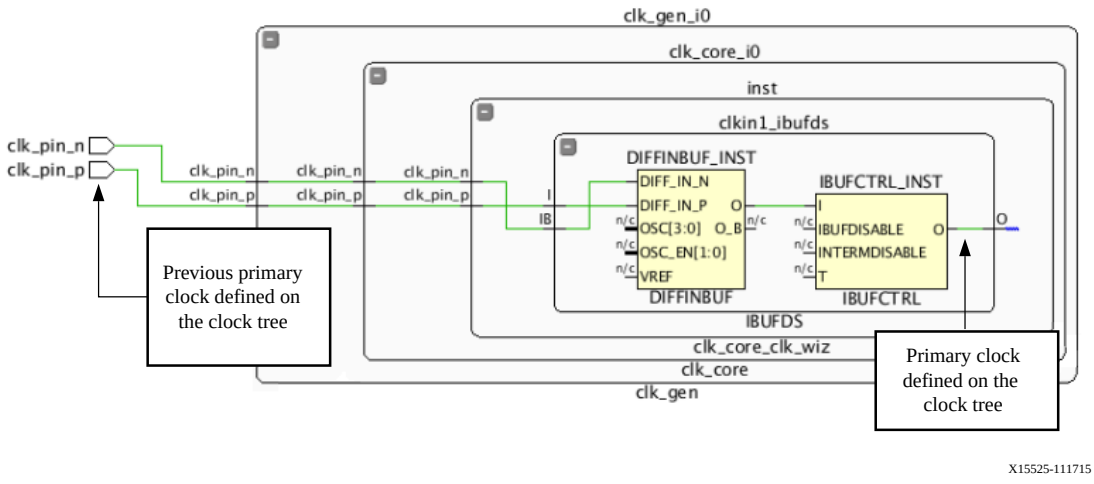
解决方案

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 252：**时钟树上的基准时钟重新定义无效**



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不同的周期、相移或反转。

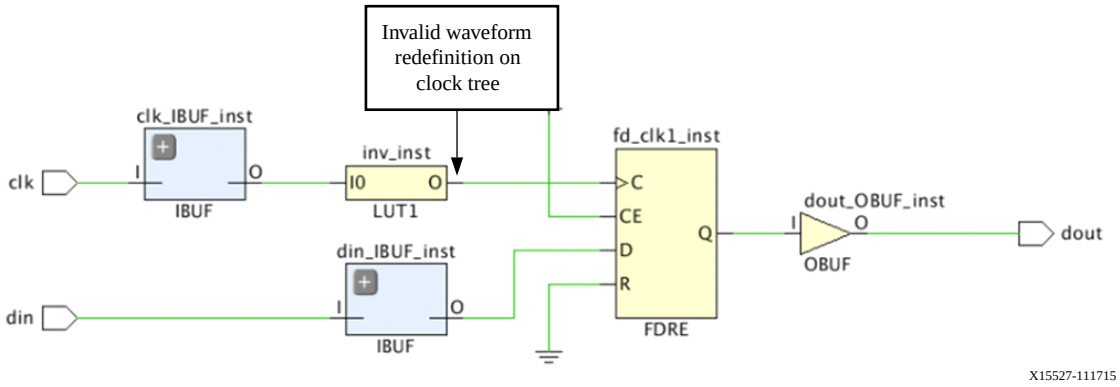
解决办法

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更多详情，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 253：**时钟树上的波形重定义无效**



TIMING-6：相关时钟间无公共基准时钟

时钟 `<clock_name>` 与 `<clock_name>` 之间相互关联（一起定时），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，这 2 个报告的时钟被视为相关联时钟，并以同步方式进行定时，即使这 2 个时钟并非衍生自公共的基准时钟，且不含有已知相位关系，也同样如此。DRC 警告报告称时序引擎无法保证这些时钟处于同步状态。

解决方案

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 (generated clock)，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_name>` 与 `<clock_name>` 之间相互关联（一起定时），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同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。

解决方案

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。

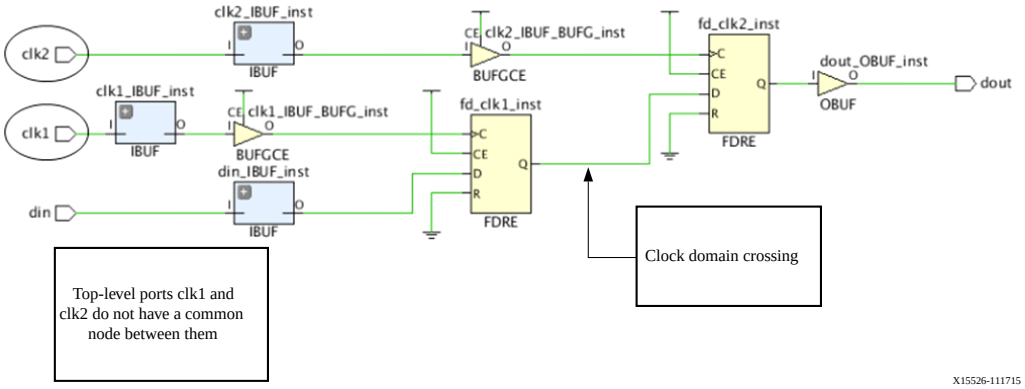
示例

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 254：**关联时钟间无公共节点**



TIMING-8：相关时钟间无公共周期

发现时钟 `<clock_name>` 与 `<clock_name>` 之间存在关联（一起定时），但两者间无公共（可扩展）周期。

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。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。

解决方案

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` 约束的两个时钟域之间检测到一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“Critical”（严重）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 Report Clock Domain Crossings。

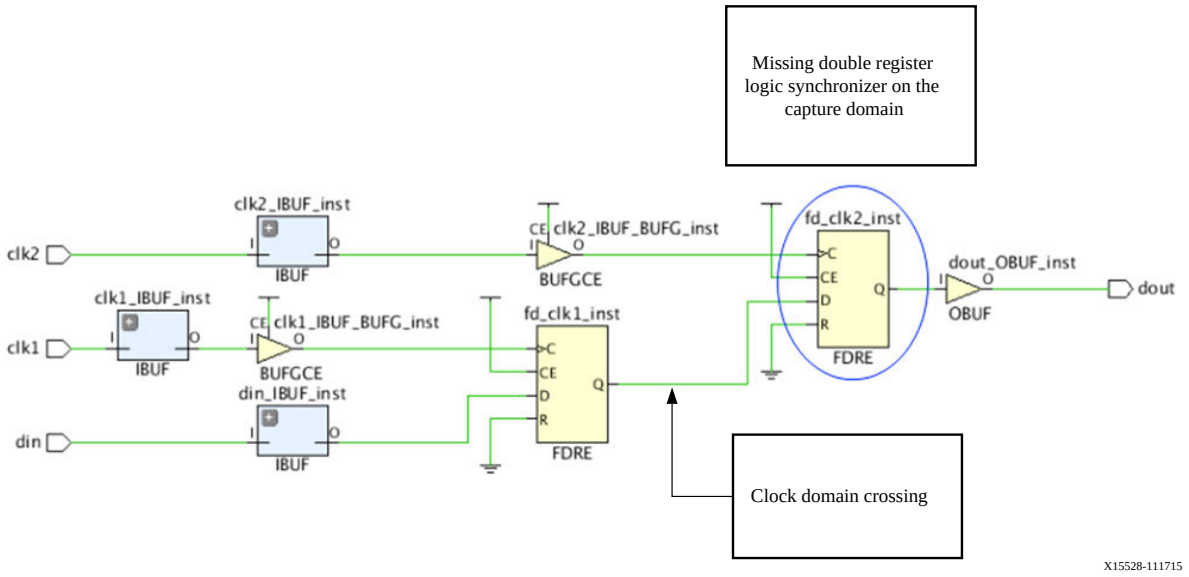
解决方案

建议在相应的设计中使时钟间路径达成正确同步。为此，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 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 255：**缺少同步器**



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。

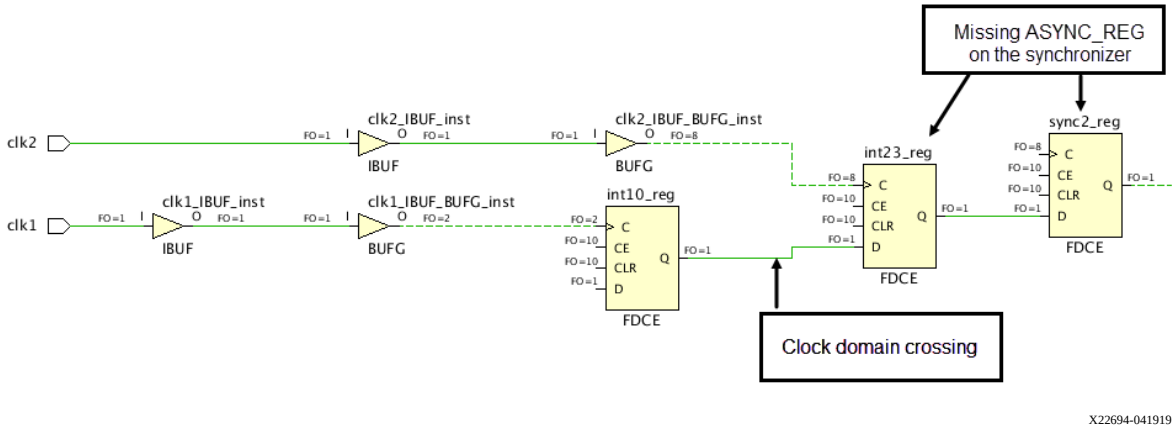
解决办法

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。但对此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 256：**同步器上缺失属性**



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。

描述

`set_max_delay` 搭配 `-datapath_only` 选项可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。

解决方案

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”

描述

“Clock Reconvergence Pessimism Removal (CRPR)”（时钟再收敛消极因素移除）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。

解决方案

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弧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互连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。

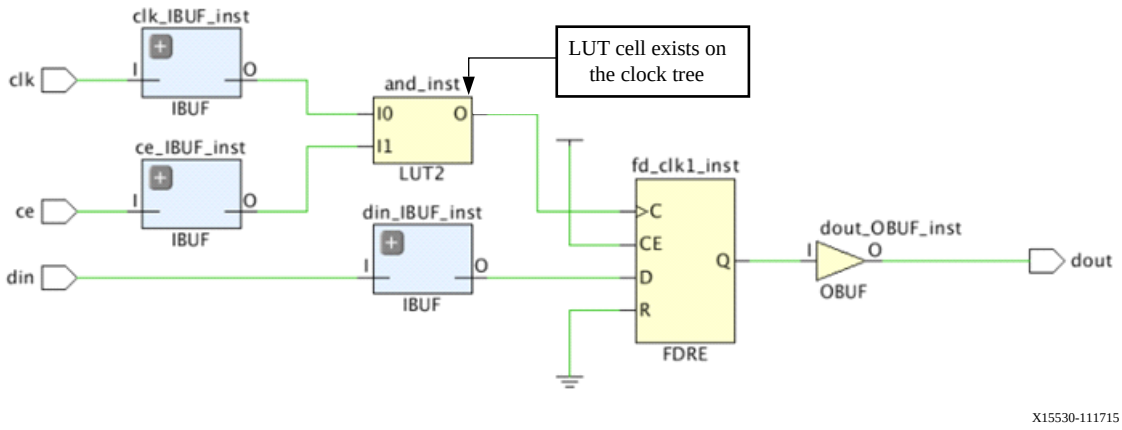
解决方案

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相等许多情况下出现此状况。对于反相 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 257：**时钟树上的 LUT**



TIMING-15：时钟间路径上的严重保持时间违例

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 的严重保持时间违例。布线期间修复严重的保持时间违例可能会影响建立时间裕量，导致时序收敛难度提升。

描述

DRC 警告报告称由于时钟间偏差导致严重的保持时间违例，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。

解决方案

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。如果由于与时钟树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。

描述

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。

解决方案

调查建立时间严重违例是否源于应定时的时序路径，或者违例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>。

描述

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。

解决方案

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。

注释：针对任何缺失的时钟沿，报告 TIMING-18 违例。如果在时钟网络传输期间时钟反相（包括在原语时钟管脚上），并且预期有输入或输出延迟，但缺少下降时钟沿，则即使在消息中未指定该缺失的时钟沿，也会生成 TIMING-18 违例。

解决方案

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相

生成时钟 <clock_name> 的波形与传入时钟 <clock_name> 的波形相比呈反相。

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更多详情，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>。

描述

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。

解决方案

解决办法是在时钟树的源端创建基准时钟或生成时钟，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。

描述

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。

解决方案

建议在设计中，将 MMCM 的 COMPENSATION 属性的默认值 AUTO 保留不变。Vivado 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他信息，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。建议在连接到具有外部反馈回路的 `FBOUT` 管脚和 `FBIN` 管脚的 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 I/O 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>` 之间已禁用时序弧，以中断时序循环。

描述

组合逻辑的输出回馈到其输入，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弧 (timing arc)。

解决方案

如果您不想创建组合反馈回路，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，因此请使用 `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，而不是交由 Vivado 时序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_name>` 和 `<clock_name>` 之间的 `set_clock_groups` 或 `set_false_path` 覆盖了 `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` 约束覆盖 `set_max_delay -datapath_only` 约束，那么将不会生成此 DRC。

解决方案

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>` 的波形与收发器设置不一致，或者缺少参考时钟定义。自动衍生时钟的周期为 <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 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，由于输入端口缺少基准时钟，Vivado 无法自动衍生 GT 的输出时钟。由此导致无法对连接到时钟相关 GT 的下游逻辑进行定时。

解决方案

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。

描述

如果某个时钟遍历驱动程序，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程序管脚扇出上的单元的时序分析。如果在驱动程序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。

解决方案

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，并改为将驱动程序时钟指定为主时钟。

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。除非自动衍生时钟已重命名，否则应不鼓励按名称引用该时钟，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 的“Timing Constraint”（时序约束）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：生成时钟所选主源管脚欠佳

生成时钟 `<clock_name>` 所选的主源管脚欠佳，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，但根据定义，该主时钟在上游比传入主时钟更远处的位置。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。

解决方案

修改 `create_generated_clock` 约束以引用主时钟源管脚，在设计中生成时钟是从该管脚直接衍生的。

TIMING-31：相移时钟上存在多周期路径

MMCM（或 PLL）生成时钟 <clock_name> 属于相移时钟，在一个或多个多周期路径约束内仅用于设置。由于 MMCM（或 PLL）属性 PHASESHIFT_MODE 设为 LATENCY，因此，可能不再需要原有的多周期路径约束。

描述

在 7 系列和 UltraScale 中，默认 MMCM 或 PLL 的相移输出作为时钟波形中的更改 (PHASESHIFT_MODE=WAVEFORM) 来进行建模。这表示为了应对管脚相移，对时钟波形边沿进行相移。

在 UltraScale+ 中，默认管脚相移作为通过时钟修改块的时延传输延迟 (PHASESHIFT_MODE=LATENCY) 来进行建模，时钟波形则无更改。

在 WAVEFORM 模式下，由于在源和目标时钟波形之间引入的相移，可能需要借助多周期约束来调整时序路径要求。在 LATENCY 模式下，则不再需要这些多周期路径约束。

如需了解有关相移模式的更多信息，请参阅 MMCM/PLL 相移模式。

解决方案

评估是否需要多周期约束。如果添加了多周期路径约束用于在 WAVEFORM 模式下调整路径要求，那么在 LATENCY 模式下很可能需要移除此约束。在时钟修改块上设置 PHASESHIFT_MODE=WAVEFORM 也可以反转管脚相移的行为。

TIMING-32：总线偏差约束已应用于过多信号

在过多信号（针对 UltraScale/UltraScale+ 超过 2500 条路径，针对 7 系列超过 1000 路径）上设置了总线偏差约束。请参阅 Vivado IDE 中的时序约束窗口中的约束位置 <position>。此约束所涵盖的第一个端点是 <object>。

描述

Vivado 会尝试通过某些布线绕行来给个别信号添加额外延迟，以满足总线偏差约束。如果相同总线偏差约束中的信号数量过多，那么可能需要添加大量布线绕行。这些绕行可能导致布线拥塞或者无法满足总线偏差约束。它也可能会给 route_design 运行时造成严重影响。

解决方案

复查总线偏差约束，验证所有信号都包含在其中。确认后，考虑重新设计此 CDC 接口以减少信号数量。

TIMING-33：安全定时的路径上的总线偏差约束无效

在已安全定时的路径上设置了总线偏差约束（请参阅 Vivado IDE 的时序约束窗口中的约束位置 <position>）。建议在异步路径上或者在同步时钟域之间的路径上使用 set_bus_skew 约束，但 set_false_path 例外。此约束所涵盖的第一个端点是 <object>。

描述

总线偏差约束是在安全定时的交汇路径上设置的，此路径属于同步时钟域交汇路径或时钟域内部路径。`set_bus_skew` 约束仅限在异步路径上使用。虽然通常建议将 `set_bus_skew` 与 `set_max_delay -datapath_only` 搭配使用，以控制源和目标寄存器之间的相对布局，但在伪路径例外所涵盖的同步路径上，也可以使用约束。

解决方案

复查总线偏差约束，确认约束已应用于异步时钟域交汇路径。如果路径应为异步路径，但 Vivado 报告路径为同步，请复查时序约束中是否缺少时钟组、伪路径或仅最大延迟数据路径。

TIMING-34：总线偏差约束含有不现实的值

某个总线偏差约束具有不现实的值（小于最小值，即，不到源和目标时钟周期的一半），或者该值 <1 ns 要求（请参阅 Vivado IDE 的时序约束窗口中的约束位置 `<position>`）。设置过于激进的值会导致运行时间增加。此约束所涵盖的第一个端点是 `<object>`。

描述

总线偏差约束设为比源时钟或目标时钟周期的一半更低的值，或者小于 1 ns。总线偏差时序可通过在 `route_design` 期间添加布线延迟来满足。如果尝试满足较为激进的值，则可能需要耗费大量运行时，并且可能导致不必要的布线拥塞。也可能导致 `route_design` 无法满足此类目标。

解决方案

复查设置的值。将其设置为至少高于源时钟或目标时钟的最小时钟周期的一半，或者设置为 1 ns。

TIMING-35：在具有相同时钟的路径中不存在公共节点

时钟 `<clock_name>` 所含路径不具有公共节点。在 `<startpoint>` 与 `<endpoint>` 之间可找到第一条路径。请复查时钟约束。

描述

如果在多个对象（端口/管脚/信号线）上定义了时钟，那么在源时钟管脚与目标时钟管脚之间的时钟内部路径可不含任何公共节点。在此类情况下，此路径可能无法安全定时，并且应被视为异步路径，因为源时钟管脚和目标时钟管脚是由不同时钟源管脚所驱动的。

解决方案

复查 `create_clock` 约束和时钟源管脚定义。如果要在多个源对象上定义该时钟，那么部分异步约束（例如，伪路径或仅最大延迟数据路径）应设置为覆盖由不同时钟源对象进行时钟设置的路径。

TIMING-36：由于无时钟沿传输，生成时钟无效

主时钟 <clock_name> 到生成时钟 <clock_name> 之间没有上升沿或下降沿传输。

描述

没有任何来自主时钟源的上升沿或下降沿可传输至生成时钟的源管脚。因此，在主时钟与生成时钟之间不存在已知的时钟关系，并且验收时序可能不准确。

解决方案

复查用于定义生成时钟的主时钟。在主时钟与生成时钟之间应存在物理路径。如有物理路径，请确保沿此路径的所有时序弧 (timing arc) 均已启用，并且上升和下降时钟沿均可传输。

TIMING-37：总线偏差约束已应用于含扇出的信号

在具有扇出的信号上，不建议应用总线偏差约束（请参阅 Vivado IDE 的“Timing Constraint”中的约束位置 <position>）。此约束所涵盖的第一个端点是 <object>。

描述

为避免因目标时钟域内潜在的逻辑再收敛导致任何硬件故障，有必要避免时钟域交汇路径上存在任何扇出。同步逻辑后的信号应仅在目标时钟域内部扇出。

解决方案

复查设计和时钟域交汇路径，移除异步时钟域交汇路径中的扇出。

TIMING-38：已在多个时钟上应用总线偏差约束

在总线偏差约束的源或目标上涉及多个时钟（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <position>）。建议每个总线偏差约束仅限一个源时钟和一个目标时钟。此约束所涵盖的第一个端点是 <object>。

描述

为了正确完成总线偏差时序分析，建议仅限一个源时钟和一个目标时钟可到达 set_bus_skew 约束中指定的时序元件。

解决方案

复查 CDC 交汇，确保仅有一个时钟可到达所有源元素和所有目标元素。如有多个时钟到达源寄存器和/或目标寄存器，那么这些时钟应定义为逻辑或物理互斥时钟。

TIMING-39：路径上所含逻辑层次过多，总线偏差约束无效

总线偏差约束已应用于具有多个逻辑层次的路径（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <MESSAGE_STRING>）。此约束所涵盖的第一个端点是 <NETLIST_ELEMENT>。

描述

为避免因目标时钟域内的逻辑再收敛导致任何硬件故障，有必要避免时钟域交汇路径上存在任何逻辑。来自源时钟域的每个寄存器都只能驱动目标时钟域内的单个寄存器，并且源和目标时钟域之间不应存在任何逻辑。

解决方案

复查设计，移除异步 CDC 交汇中的组合逻辑。

TIMING-40：在跨 SLR 的 OSERDESE3 CLK 与 CLKDIV 管脚之间存在最大偏差违例

在 OSERDESE3 实例 <NETLIST_ELEMENT> 上的 CLK 管脚与 CLKDIV 管脚之间存在最大偏差违例，并且时钟网络跨 SLR 才能到达时钟管脚。不建议从布局在另一个 SLR 内的时钟源来驱动 OSERDESE3CLK 和 CLKDIV。请检查您的设计。

描述

OSERDESE3 的 CLK 和 CLKDIV 管脚具有特定的偏差要求，不得超出此要求。如果 CLK 和 CLKDIV 与 OSERDESE3 位于不同 SLR 中，则可能导致时钟路径延迟出现最大偏差违例。

解决方案

复查 OSERDESE3 的时钟拓扑结构，确保 CLK 管脚和 CLKDIV 管脚的时钟源与 OSERDESE3 都布局在相同的 SLR 内。

TIMING-41：内部管脚上定义的前向时钟无效

前向时钟 <clock_group> 是在管脚 <netlist_element> 上定义的，而不是在端口 <netlist_element> 上定义的。

描述

前向时钟是在连接到输出端口的叶节点管脚上定义的，而不是在输出端口本身上定义的。为了正确执行 I/O 时序计算，前向时钟应在输出端口上定义。

解决方案

复查生成时钟约束，将前向时钟定义移至输出端口而不是内部叶节点管脚。

TIMING-42：在时钟树中检测到路径分段

<message_string> 延迟约束位置 <message_string> 当前正在管脚 <netlist_element> 上阻塞 <clock_group> 的传输。

描述

由于最小/最大延迟时序约束中的起点或端点无效，导致时钟树上检测到路径分段。发生此类情况时，本工具会对此无效管脚强制禁用时序弧 (timing arc) 以阻止时钟传输。这将导致时序验收不准确，并且设计可能在硬件中失败。

解决办法

复查应用于时钟树的最小/最大延迟约束，仅指定有效的起点和端点。欲知详情，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的“路径分段”。

TIMING-43：千兆位收发器 (GT) 上存在最小周期或最小脉冲宽度违例

GT 管脚 <instance/pin> 存在 MIN_PERIOD 或 MIN_PULSE_WIDTH 违例。对于 GT 实例，“Power Analysis Report”（功耗分析报告）并不准确。

描述

GT 时钟管脚上的最短周期检查可确保驱动 GT 实例的时钟的运行频率不高于原语内部硬件所能承受的频率。如果发生违例，设计在硬件中可能发生时序收敛失败，功耗分析报告将变得不准确。

解决方案

要解决此违例，请检查相关器件系列的 AC 和 DC 特性数据手册，查找此原语管脚允许的最大频率。

由于这是硅片层面的限制，您需要调低频率才能解决此违例。请参阅这篇博文以获取更多信息。

TIMING-44：不合理的用户时钟内部不确定性

在 <clock_name> 时钟上定义了 <delay> ns 用户时钟不确定性（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <position>）。用户时钟不确定性过高可能对时序收敛造成负面影响。请复查所需的用户时钟不确定性的量。

描述

不建议在 2 个时钟之间定义过高的用户时钟不确定性，因为它会影响时序收敛、编译时间和 QoR。它还可能影响功耗，导致无法收敛时序。

解决办法

复查用户时钟间不确定性，将其降低至所需的最小值。

建议：过约束不得超过 0.5 ns。过度约束设计会导致实现功耗以及运行时间增加。如需了解更多信息，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“设计过约束”。

TIMING-45：不合理的用户时钟间不确定性

在 <clock_name> 时钟与 <clock_name> 时钟之间定义了 <delay> ns 用户时钟不确定性（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <position>）。用户时钟不确定性过高可能对时序收敛造成负面影响。请复查所需的用户时钟不确定性的量。

描述

不建议定义过高的用户时钟内部不确定性，因为它会影响时序收敛、编译时间和 QoR。它还可能影响功耗，导致无法收敛时序。

解决办法

复查用户时钟内部不确定性，将其降低至所需的最小值。建议不要过约束至超过 0.5 ns。如需了解更多信息，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“设计过约束”。

TIMING-46：多周期路径含绑定 CE 管脚

在具有直接连接的寄存器 <cell_name1> 与 <cell_name2> 之间定义了一条或多条多周期路径，并且 CE 管脚连接到 VCC（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <position>）。这可能导致路径要求不准确。

描述

源寄存器和目标寄存器的 CE 管脚不受动态信号控制（CE 管脚绑定到 VCC）。由于在源寄存器与目标寄存器之间存在直接数据路径连接，因此基于单周期路径要求来捕获此路径上的数据。路径上定义的多周期路径并不表示硬件中的行为，并且可能导致硬件中设计失败。

解决方案

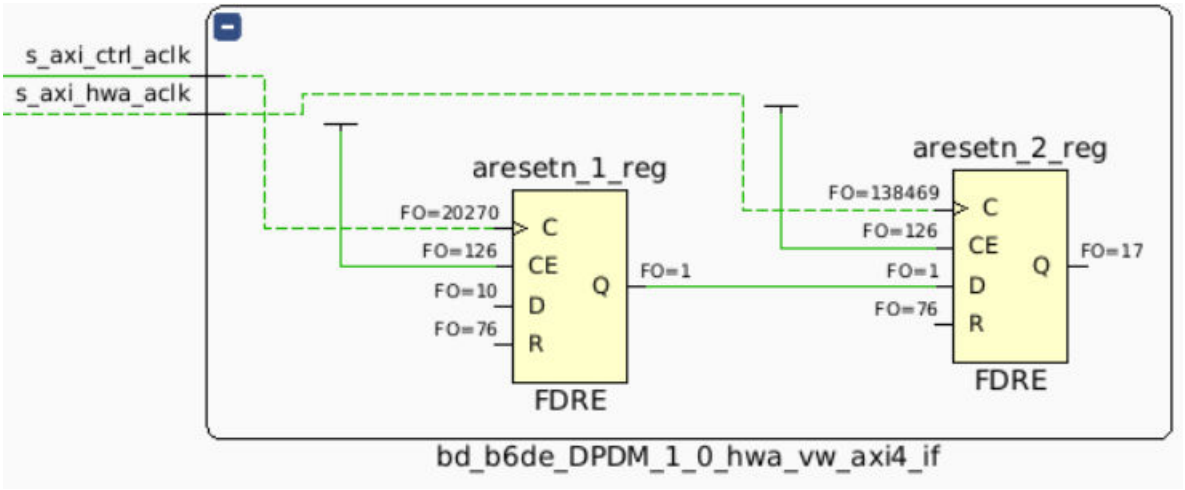
应复查路径和时序约束。如果预计将绑定 CE 管脚，则需移除多周期路径。如果预计路径将成为多周期路径，那么源寄存器和/或目标寄存器的 CE 管脚应由动态信号驱动，此信号根据提供的多周期约束来进行切换。

示例

以下约束是由用户编写的，此约束应用于 2 个寄存器之间的数据路径，如以下板级原理图所示，其中目标寄存器的 CE 管脚连接到 VCC：

```
set_multicycle_path -setup -end -from [get_clocks -of [get_ports -
scoped_to_current_instance s_axi_ctrl_aclk]] -to [get_clocks -of [get_ports
-scoped_to_current_instance s_axi_hwa_aclk]] 2
```

图 258：**含绑定 CE 管脚的多周期路径示例**



在超出多周期路径作用域的每个时钟沿上检查是否发生数据更改。如有更改，请确保添加所需的逻辑（绑定到 CE 管脚）。

下图显示了触发器到触发器路径，其中正确的时钟使能逻辑绑定到 CE 管脚，这样即可按交替时钟周期使能触发器：

```
set_multicycle_path 2 -setup -from [get_pins data0_reg/C] -to [get_pins
data1_reg/D]
```

图 259：**触发器到触发器路径**

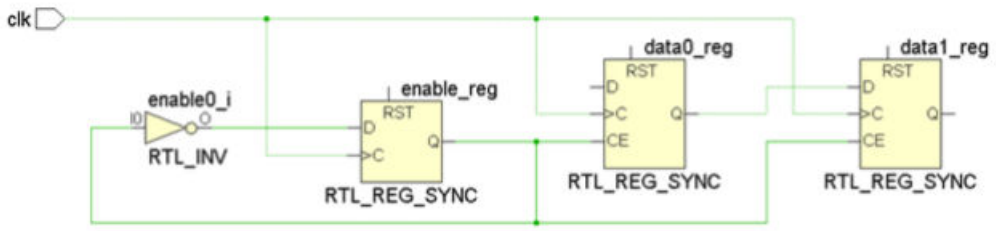
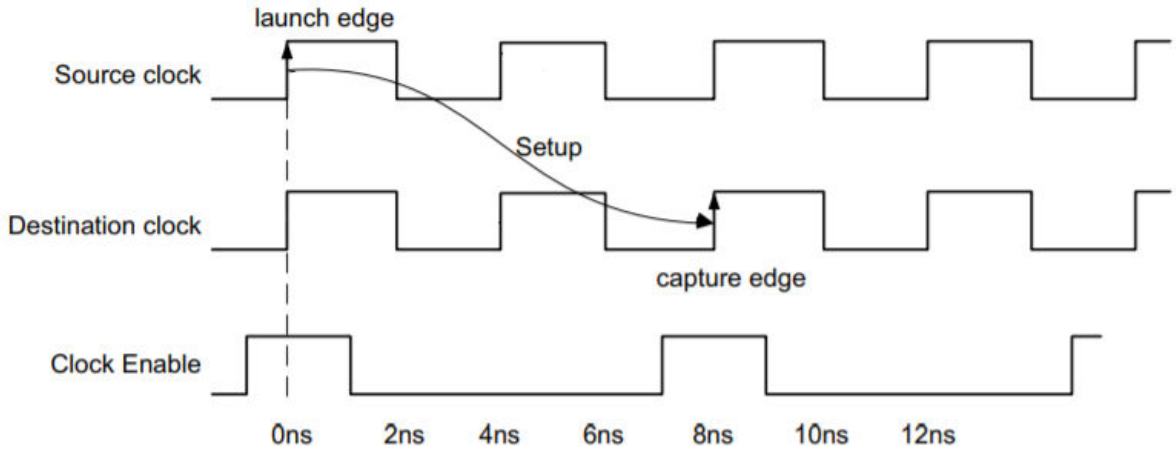


图 260：**时序图**



TIMING-47：同步时钟之间的伪路径、异步时钟组或仅最大延迟数据路径约束

在 <clock_group> 与 <clock_group> 这两个时钟之间设置了 <message_string> 时序约束（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <message_string>）。使用 set_false_path、set_clock_groups 或 set_max_delay -datapath_only 屏蔽整个同步时钟域可能导致硬件故障。

描述

在同步时钟域交汇上不应存在 false_path、set_clock_group -asynchronous 或 set_max_delay -datapath_only 约束。如果存在此类约束，将无法正确完成路径定时，验收时间将不准确，并且设计在硬件中可能失败。

解决方案

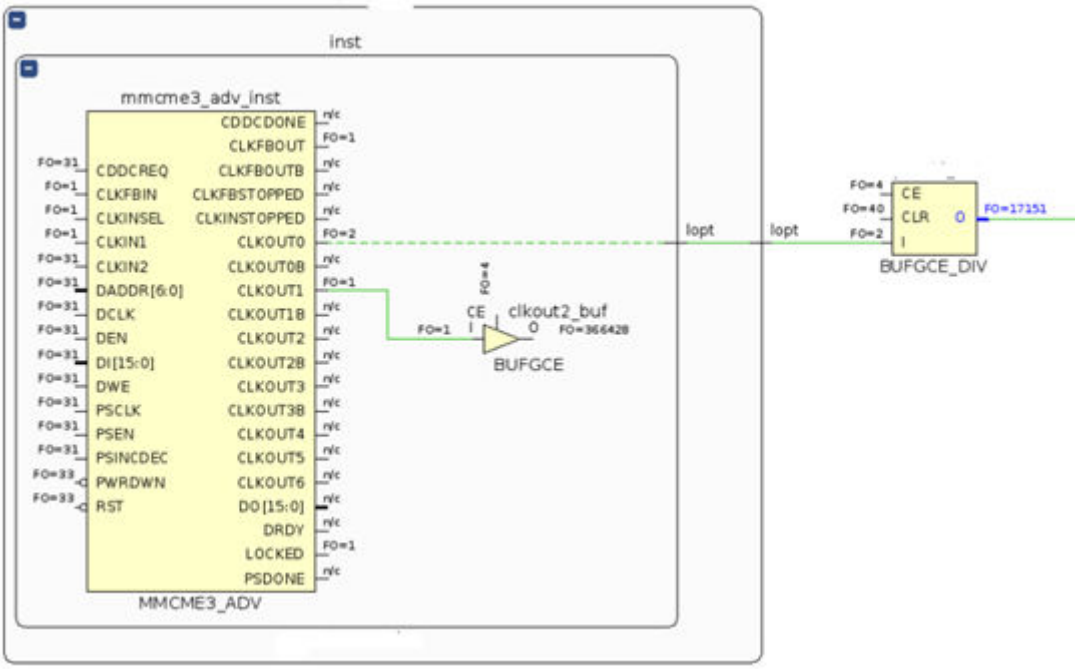
移除同步时钟之间应用的 `false_path`、`set_clock_group -asynchronous` 或 `set_max_delay -datapath_only` 约束。如果时钟应为异步时钟，请添加具有适当同步电路的异步时钟约束，用于异步时钟域交汇。

要了解哪个时钟域交汇属于同步或异步，请参阅时钟交互报告。

示例

在以下示例中，使用 2 项 MMCM 输出来生成时钟。BUFGCE_DIV 输出具有可分频时钟 `clk1`，它具有来自 `CLKOUT0` 的主时钟。`CLKOUT1` 具有生成时钟 `clk2`。如果添加诸如 `set_clock_groups -asynchronous -group [get_clocks clk1] -group [get_clocks clk2]` 之类的约束，那么其中将包含 TIMING-47 警告。

图 261：**同步时钟之间的伪路径、异步时钟组或仅最大延迟数据路径约束**



由于 `clk1` 和 `clk2` 都来自相同 MMCM，因此被视为彼此同步。跨这些域的数据也被视为位于同步 CDC 下。因此，无需添加 `set_clock_groups -asynchronous`。

TIMING-48：在锁存器输入上存在“仅最大延迟数据路径”约束

在锁存器 `<pin_name>` 的输入上检测到“仅最大延迟数据路径”约束（请参阅 Vivado IDE 的“Timing Constraint”（时序约束）窗口中的约束位置 `<position>`）。此约束通常用于异步时钟域交汇，可能触发不现实的锁存器时间借用，从而影响下游时序路径的 QoR。

描述

最大延迟约束不应应用于锁存器的输入，因为约束会更改路径的要求，可能导致不现实的锁存器时间借用，与硬件中的行为不符。这将导致的硬件故障。

解决方案

对照硬件行为验证时序约束，如果锁存器输入上应用的仅最大延迟数据路径与硬件行为不匹配，则将其移除。

TIMING-49：来自并行 BUFGCE_DIV 的使能或复位拓扑结构不安全

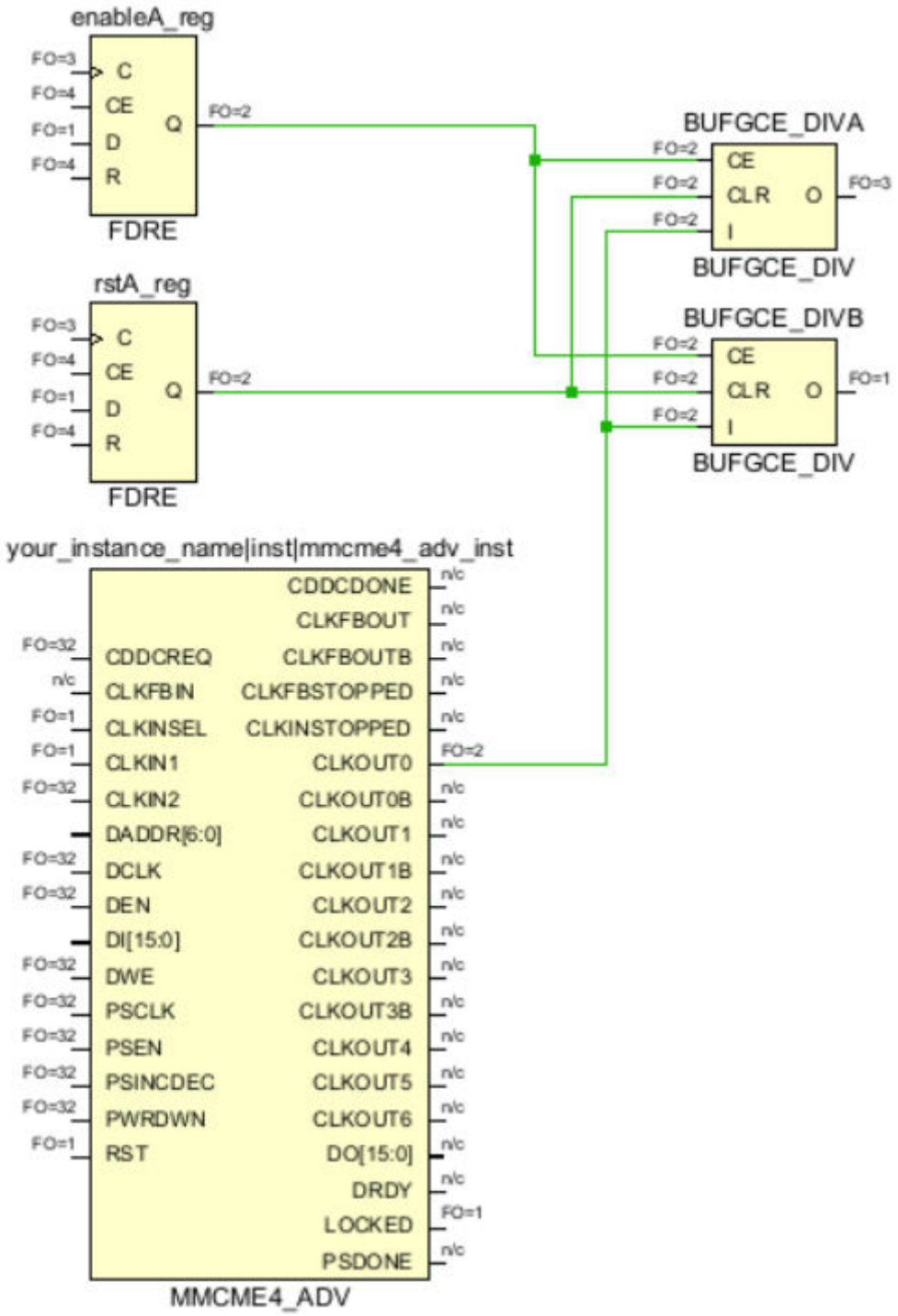
如果并行 BUFGCE_DIV 单元的 BUFGCE_DIVIDE 属性均设为大于 1 的值，并且这两个单元共同驱动时钟，那么为了确保此类时钟路径上的时序安全，缓冲器 <buffer> 和 <buffer> 必须使用相同使能信号 (CE) 和相同清除信号 (CLR)。清除信号不得连接到电源，也不得接地。否则，硬件中的分频时钟之间可能出现相移。建议使用安全时钟启动复位电路来同时复位这两个 BUFGCE_DIV 缓冲器。

描述

如果同步时钟拓扑结构包含 BUFGCE_DIV 缓冲器，并且这些缓冲器的 BUFGCE_DIVIDE>1，那么这些时钟拓扑结构必须采用公用的控制信号来防止 BUFGCE_DIV 内部状态中发生相位不明确。此类情况可能导致在不同时钟周期内发生时钟缓冲器复位，且 BUFGCE_DIV 输出存在未知的时钟关系。此设置在硬件中可能失败。下图显示了电路的正确实现：

注释：TIMING-49 并不局限于由相同 CMB 驱动的并行 BUFGCE_DIV。检查会涵盖已安全定时的所有同步拓扑结构。

图 262：**BUFGCE_DIV 缓冲器的正确实现**



解决办法

各并行 BUFGCE_DIV 缓冲器的 CLR 管脚都应绑定到相同信号或者由相同信号驱动。使用安全的时钟启动复位电路在硬件上执行无缝操作。在 Clocking Wizard IP 中可启用该操作。如需了解更多信息，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“同步 CDC”。

TIMING-50：同级锁存器之间的路径要求不现实

在源管脚 <pin> 与目标管脚 <pin> 之间存在时序路径。这两个锁存器均为 <positive|negative> 电平敏感型锁存器，路径要求为 0 ns。0 ns 路径要求源自于保守的锁存器分析，由于采用时间借用计算，故而可能严重影响运行时。除非存在多周期路径约束用于将路径要求调整为现实的值（至少为时钟周期的一半），否则不建议采用此类拓扑。

描述

在具有相同敏感级别的 2 个锁存器之间，Vivado 时序引擎会计算其保守要求为 0 ns。由于在第二个锁存器上会发生时间借用，因此，此类拓扑结构可能对时序分析和实现造成重大影响。

解决方案

此拓扑结构不常用，因为级联锁存器通常涉及对相反电平敏感的锁存器，以防止数据在相同时钟周期内跨多个锁存器。复查锁存器的拓扑结构。如果其连接方式符合预期，那么请确认硬件中的行为，并在级联锁存器之间使用多周期路径来将路径要求提升至较为现实的值。

TIMING-51：来自并行 CMB 的相关时钟之间无公共相位

时钟 <clock_name> 与 <clock_name> 一起定时，但两者间无相位关系。此设置在硬件中可能失败。时钟源自两个并行时钟修改块，至少有一个 MMCM、PLL 或 XPLL 输入时钟分配器未设为 1。为了安全定时，并行时钟设置中所涉及的所有 MMCM、PLL 或 XPLL 都必须将其时钟分频器设为 1。

描述

如果时钟拓扑结构涉及多个衍生自并行 MMCM 或 PLL 的时钟，那么仅当每个 CMB 的时钟分频器 (DIVCLK_DIVIDE) 都设置为 1 时，才能安全定时。如果 DIVCLK_DIVIDE 设为任何其他值，则都会导致丢失并行 CMB 之间的时钟关系，并且时钟域交汇变为异步状态。

解决办法

如果预计并行 PLL 之间的时钟域交汇处于同步状态，请将时钟分频器 (DIVCLK_DIVIDE) 更改为 1。如果预计时钟域交汇处于异步状态，请更改时钟约束，将时钟域交汇设为异步，并在目标时钟域中添加适当的同步电路。如需了解更多信息，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“同步 CDC”。

TIMING-52：来自扩展频谱 MMCM 的相关时钟之间无公共相位

针对 MMCM <cell> 启用扩展频谱。在此模式下，由于 VCO 频率变动，在 <clock_name> 与 <clock_name> 之间无法安全定时。

描述

在扩展频谱模式下，由于 VCO 频率随时间变化，在相同 MMCM 的 2 个不同输出之间无法安全定时。

解决方案

确认此 MMCM 需采用扩展频谱模式。如需此模式，请将时钟拓扑结构更改为使用单一 MMCM 输出，并使用时钟分频器在 MMCM 外部进行时钟分频。如果无法更改时钟拓扑结构，那么 MMCM 的多个时钟之间的交汇必须视为异步时钟交汇，并且必须添加适当的电路才能在这些时钟域之间传输数据。

TIMING-53：来自 DPLL 的相关时钟之间无公用相位

时钟 <clock_name> 与 <clock_name> 一起定时，但两者间无相位关系。此设置在硬件中可能失败。某个时钟源自当前不使用检相器的 DPLL <cell>，或者其传入时钟未连接到 CLKIN_DESKEW 管脚。在这些条件下，在 DPLL 与其自动衍生的时钟（或下游生成时钟）之间无法安全定时，因为时钟间的关系未知。

描述

此项检查仅适用于 Versal 自适应 SoC。

在 Versal 自适应 SoC 器件中，如果使用检相器 (CLKOUTx_PHASE_CTRL=01) 并且 CLKIN_DESKEW 管脚连接到 CLKIN 管脚，那么仅限在 DPLL 输入时钟与其任一输出时钟之间进行定时才安全。在任何其他情况下，主时钟与自动衍生的时钟之间的相位关系是未知的，这些时钟之间的时钟域交汇应视为异步关系：交汇中的任意路径都应在目标时钟域中具有适当的同步电路。

解决方案

如果主时钟与自动衍生时钟之间的路径应为同步关系，请复查 DPLL 设置和连接方式，以使用正确的去歪斜配置。如果不为同步关系，那么时钟域交汇应被视为异步关系，且设计的目标时钟域中应包含适当的逻辑同步。

TIMING-54：时钟间存在如下约束：限定作用域的伪路径、时钟组，或仅最大延迟数据路径约束

在 <clock_group> 与 <clock_group> 这两个时钟之间设置了限定作用域的 <message_string> 时序约束（请参阅 Vivado IDE 的“Timing Constraint”窗口中的约束位置 <message_string>）。不建议在时钟之间定义此类限定作用域的约束，因为此类约束会影响超出作用域的时序路径。

描述

时序时钟全局传输，无法局限于某个层级模块。不建议在时钟之间定义限定作用域的时序约束（例如，时钟组、伪路径和仅最大延迟数据路径），因为此约束会影响整个设计的全局层面的验收时序。为避免发生混淆以及意外发生的覆盖范围超出层级模块作用域的问题，应在顶层定义此类约束。

解决办法

复查时序约束，如需将约束应用于整个设计，请将其移至顶层。如果只需将约束应用于实例下的作用域内，则可能需要修改设计的时序约束和拓扑结构。如需了解更多信息，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903) 中的“同步 CDC”。

TIMING-55：多个时钟到达同一个 CMB 去歪斜管脚

有多个时钟传输至去歪斜管脚 <pin_name>，导致时序分析不准确且可能导致硬件故障。时序约束应更新为仅将单个时钟传输至该去歪斜管脚。到达管脚的时钟列表为 <clock_names>。

描述

此项检查仅适用于 Versal 自适应 SoC。

为了正常完成时序分析，不支持将多个时钟传输到 MMCM/XPLL/DPLL 的去歪斜管脚。如不满足此要求，则可能导致验收时序不准确，并存在硬件故障风险。

解决方案

复查时钟拓扑结构和时序约束，确保仅将一个时钟传输到 MMCM/XPLL/DPLL 去歪斜管脚。

TIMING-56：缺少按逻辑或物理方式排除的时钟组约束

多个时钟均为在源管脚 <pin_names> 上由用户生成或自动衍生的时钟，但彼此间并未按逻辑或物理方式互斥。为了使硬件中的静态时序分析行为匹配，在相同管脚上不得生成多个时钟。在此类情况下，这些时钟应定义为物理或逻辑互斥。源管脚上生成的时钟列表为 <clock_names>。

描述

如果时钟设置原语的输出管脚中存在多个生成时钟或自动衍生时钟，那么在硬件上不得同时使用这些时钟。要使静态时序分析与硬件行为相匹配，请在这些时钟之间提供以物理或逻辑方式互斥的时钟组约束。否则，某些时钟对即使在硬件中不存在，也仍会予以定时。

解决办法

确保基于时钟树拓扑结构，在源管脚上生成的时钟之间提供适当的物理或逻辑互斥时钟组约束。如需了解更多信息，请参阅《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的“约束专属时钟组”。

TIMING-57：不受支持的配置，其中包含 PHASESHIFT_MODE 和数字去歪斜

时钟修改块 `<netlist_element>` 配置了数字去歪斜和 `PHASESHIFT_MODE=WAVEFORM`。此组合不受支持，时序分析会继续运行并将其作为 `PHASESHIFT_MODE = LATENCY` 来处理。请将指定的单元配置更改为 `PHASESHIFT_MODE= LATENCY`，并确保未写入任何与期望的 `PHASESHIFT_MODE=WAVEFORM` 不符的时序约束。

描述

此项检查仅适用于 Versal 自适应 SoC。

如果 MMCM/XPLL/DPLL 配置有数字去歪斜，那么管脚相移只能通过 CMB 作为时延延迟来建模，而不能作为时钟波形内的更改来建模。此时钟设置元素是通过 `PHASESHIFT_MODE=WAVEFORM` 来配置的，定时器无法遵循此配置。由于本工具会强制执行时延建模，因此该违例不会影响验收时序准确性。

解决方案

要阻止此违例，请将管脚相移建模更改为 `PHASESHIFT_MODE=LATENCY`。复查任何现有时序例外，以支持 `WAVEFORM` 建模的原始意图。

QoR 建议报告 RTL 代码更改示例

本节详述了需进行 RTL 编辑的部分 QoR 建议触发器。其中对每个触发器提供了简要描述，并提供了所需修改的示例。

RQS_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，使其能够实现优化的 RAM 布局，并提供将寄存器布局到互连结构内而不是布局到 RAM 内的选项，从而实现关键路径优化。

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 263：**修改前**

```
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] ADDRB,
    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[ADDRA] <= DINA;

  always @(posedge CLKB)
    if (ENB)
      ram_data <= ram_i[ADDRB];

  // 1 clock cycle read latency at the cost of a longer clock-to-out timing
   assign DOUTB = ram_data;

endmodule
```

图 264：**修改后**

```
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] ADDRB,
    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[ADDRA] <= DINA;

  always @(posedge CLKB)
    if (ENB)
      ram_data <= ram_i[ADDRB];

  // 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 265：**修改前**

```
-- 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 266：**修改后**

```
-- 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;
```

RQS_TIMING-202：添加额外流水打拍以拓宽倍频器

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 的最大工作频率。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 267：**修改前**

```
module wide_mulitplier #(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 268：**修改后**

```
module wide_mulitplier #(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 269：**修改前**

```
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

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);

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 270：**修改后**

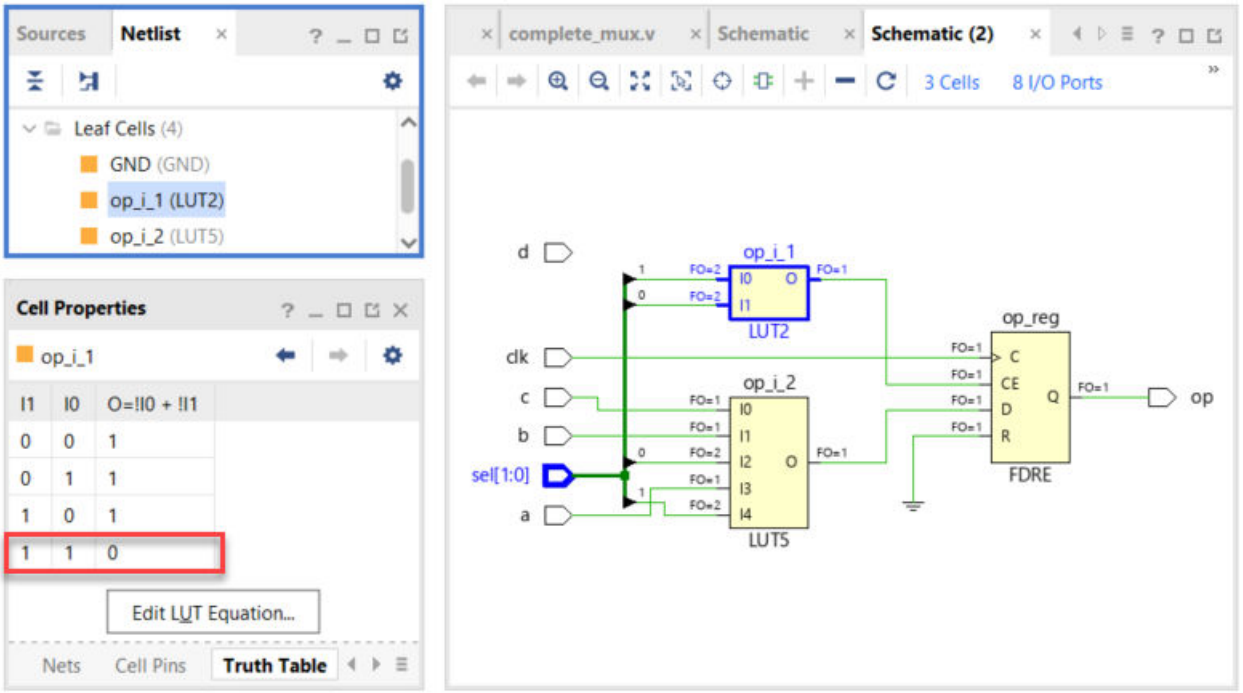
```
entity wide_multiplier is
    Generic ( DATA_WIDTH : integer := 30);
    Port ( clk : in STD_LOGIC;
           a : in STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
           b : in STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
           p : out STD_LOGIC_VECTOR (2*DATA_WIDTH-1 downto 0));
end wide_multiplier;

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;
```

RQS_UTIL-10：Case 语句不完整导致控制集增加

当 case 语句不完整时，会强制输出记住用于推断时钟使能的先前状态。建议定义所有状态。对于其他状态，请将输出定义为先前使用的信号或者常量 1 或常量 0，以免增加逻辑。

下图显示了当状态“11”未定义时，CE LUT 的推断。

图 271：**CE 推断不完整的 Case**



Verilog 代码示例

修改前：

```
module incomplete_mux(
input clk,
input a,b,c,d,
input [1:0] sel,
output reg op
);

always @(posedge clk)
  begin
  case (sel)
    2'b00 : op <= a;
    2'b01 : op <= b;
    2'b10 : op <= c;
    default: ;
  endcase
end
endmodule
```

修改后：

```
module incomplete_mux(
input clk,
input a,b,c,d,
input [1:0] sel,
output reg op
);
```

```
always @(posedge clk)
  begin
  case (sel)
    2'b00 : op <= a;
    2'b01 : op <= b;
    2'b10 : op <= c;
    2'b11 : op <= a; // ADDED
    default: ;
  endcase
end
endmodule
```

VHDL 代码示例

修改前：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity incomplete_mux is
Port ( clk : in STD_LOGIC;
       a,b,c,d : in STD_LOGIC;
       sel : in STD_LOGIC_VECTOR (1 downto 0);
       op : out STD_LOGIC);
end incomplete_mux;

architecture Behavioral of incomplete_mux is

begin

process (clk)
begin
  if rising_edge(clk) then
    case sel is
      when "00" => op <= a;
      when "01" => op <= b;
      when "10" => op <= c;
      when others => null;
    end case;
  end if;
end process;

end Behavioral;
```

修改后：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity incomplete_mux is
Port ( clk : in STD_LOGIC;
       a,b,c,d : in STD_LOGIC;
       sel : in STD_LOGIC_VECTOR (1 downto 0);
       op : out STD_LOGIC);
end incomplete_mux;

architecture Behavioral of incomplete_mux is

begin
```

```
process (clk)
begin
  if rising_edge(clk) then
    case sel is
      when "00" => op <= a;
      when "01" => op <= b;
      when "10" => op <= c;
      when "11" => op <= a;
    end case;
  end if;
end process;

end Behavioral;
```

RQS_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 使用率、时序以及拥塞（如果适用）。

Verilog 代码示例

图 272：**修改前**

```
module sp_rom (clk, en, addr, dout);
input clk;
input en;
input [5:0] addr;
output  [83:0] dout;

reg [83:0] data;

// Combinatoral 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 273：**修改后**

```
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 274：**修改前**

```
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 275：**修改后**

```
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;
```

参考设计文件

请从 AMD 网站下载与本附录关联的参考设计文件。

AMD

附录 C

附加资源与法律声明

查找其他文档

技术信息门户网站

AMD 技术信息门户网站是旨在使用您的网页浏览器提供健全的文档搜索和导航的在线工具。要访问该技术信息门户网站，请转至 https://docs.amd.com。

注释：单击链接将打开英语版本，但您可从下拉列表中选择简体中文版本（如可用）。请注意，简体中文版本可能比英语版本旧。

Documentation Navigator

Documentation Navigator (DocNav) 是预安装的工具，支持访问 AMD 自适应计算文档、视频和支持资源，您可在其中通过筛选和搜索来查找信息。要打开 DocNav，请执行以下操作：

- 在 AMD Vivado™ IDE 中，单击“Help” → “Documentation and Tutorials”。
- 在 Windows 上，单击“Start”（开始）按钮并选中“Xilinx Design Tools” → “DocNav”。
- 在 Linux 命令提示中输入 `docnav`。

注释：如需了解有关 DocNav 的更多信息，请参阅《Documentation Navigator 用户指南》(UG968)。

注释：您无法从 DocNav 访问简体中文版本。请使用设计中心网页。

设计中心

AMD 设计中心提供了根据设计任务和其他主题整理的文档链接，可供您用于了解关键概念以及常见问题解答。要访问设计中心，请执行以下操作：

- 在 DocNav 中，单击“Design Hubs View”选项卡。
- 转至设计中心网页。

支持资源

如需获取答复记录、技术文档、下载以及论坛等支持资源，请访问技术支持。

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：设计流程概述》(UG892)
2. 《Vivado Design Suite 用户指南：使用 Vivado IDE》(UG893)
3. 《Vivado Design Suite 用户指南：使用 Tcl 脚本》(UG894)
4. 《Vivado Design Suite Tcl 命令参考指南》(UG835)
5. 《Vivado Design Suite 用户指南：系统级设计输入》(UG895)
6. 《Vivado Design Suite 用户指南：综合》(UG901)
7. 《Vivado Design Suite 用户指南：使用约束》(UG903)
8. 《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949)
9. 《Vivado Design Suite 用户指南：实现》(UG904)
10. 《Vivado Design Suite 用户指南：功耗分析与优化》(UG907)
11. 7 系列 FPGA 时钟资源用户指南(UG472)
12. 《Vivado Design Suite 属性参考指南》(UG912)
13. 《UltraFast 设计方法时序收敛快捷参考指南》(UG1292)
14. 所有 Vivado Design Suite 文档

培训资料

AMD 提供多种多样的培训课程和 QuickTake 视频，可帮助您进一步了解本文档中提出的概念。使用此链接下的这些培训，获取相关培训资料：

1. 使用 Vivado Design Suite 设计 FPGA 1 (FPGA-VDES1)
2. 使用 Vivado Design Suite 设计 FPGA 2 (FPGA-VDES2)
3. 使用 Vivado Design Suite 设计 FPGA 3 (FPGA-VDES3)
4. 使用 Vivado Design Suite 设计 FPGA 4 (FPGA-VDES4)

修订历史

下表列出了本文档的修订历史。

章节	修订综述
2024 年 12 月 19 日 2024.2 版	
数据流路径	更新内容和截屏

章节	修订综述
策略建议	添加注释
非工程模式	更新内容
自动 QoR 建议和增量编译	添加受支持的器件
自动 QoR 建议和 ML 策略	添加受支持的器件
受支持的家族和设计流程	更新受支持的器件
智能设计运行	添加链接和受支持的器件
管理建议	添加主题
2024 年 6 月 5 日 2024.1 版	
数据流分析	新增章节。
智能设计运行	添加步骤。

请阅读：重要法律声明

本文档所示信息仅做参考，其中可能包含不准确的技术信息、疏漏和印刷错误。受诸多原因影响，此处所含信息可能发生更改，也可能无法准确呈现，这些原因包括但不限于产品和路线图变更、组件和主板版本更改、新增模型和/或产品发布、不同制造商之间存在的产品差异、软件更改、BIOS 刷新、固件升级等。任何计算机系统均存在安全性漏洞风险，无法彻底阻止也无法缓解这类风险。AMD 没有任何义务来更新或者以任何其他方式纠正或修改这些信息。但 AMD 保留随时修改这些信息和更改文档内容的权利，AMD 没有任何义务将此类修改或更改通知任何人。此处信息“按原样”提供。AMD 对于本文档内容不作任何陈述或保证，并且对于这些信息中可能出现的任何不准确、错误或疏漏问题不承担任何责任。对于有关任何暗含的非侵权、适销性及适合特定用途的保证，AMD 特此声明不承担任何责任。无论在任何情况下，对于任何人因使用此处包含的任何信息而形成的依赖或者引发的任何直接、间接、特殊或其他后果性损害，AMD 概不负责，即使 AMD 已明确获悉存在发生此类损害的可能性也是如此。

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