



Intel® Quartus® Prime Standard Edition 用户指南

设计优化

针对 Intel® Quartus® Prime 设计套件的更新: **18.1**

本翻译版本仅供参考，如果本翻译版本与其英文版本存在差异，则以英文版本为准。某些翻译版本尚未更新对应到最新的英文版本，请参考[英文版本](#)以获取最新信息。



在线版本



发送反馈

UG-20177

ID: **683230**

版本: **2018.11.12**

内容

| | |
|--|-----------|
| 1. 设计优化概述..... | 7 |
| 1.1. 器件考量..... | 7 |
| 1.1.1. 器件迁移考量..... | 7 |
| 1.2. 初始编译的必需设置..... | 7 |
| 1.2.1. I/O 约束指导..... | 7 |
| 1.2.2. 时序约束指导..... | 8 |
| 1.2.3. Partitions and Floorplan Assignments for Incremental Compilation..... | 8 |
| 1.3. 权衡和限制..... | 9 |
| 1.3.1. Preserving Results and Enabling Teamwork..... | 9 |
| 1.3.2. 减少面积..... | 10 |
| 1.3.3. 减短关键路径延迟..... | 10 |
| 1.3.4. 降低功耗..... | 10 |
| 1.3.5. 减少运行时间..... | 11 |
| 1.4. Intel Quartus Prime 软件工具用于设计优化..... | 11 |
| 1.4.1. Design Visualization Tool (设计可视化工具) | 11 |
| 1.4.2. 向导 (Advisors) | 11 |
| 1.4.3. 设计管理..... | 12 |
| 1.5. Design Space Explorer II..... | 12 |
| 1.5.1. DSE II 工作原理..... | 13 |
| 1.5.2. 使用 DSE II Utility 执行设计管理..... | 15 |
| 1.6. 设计优化概述修订历史..... | 15 |
| 2. 优化设计网表..... | 17 |
| 2.1. 何时使用 Netlist Viewer: 分析设计问题..... | 17 |
| 2.2. 使用 Netlist Viewers 的 Intel Quartus Prime 设计流程..... | 18 |
| 2.3. RTL Viewer 概述..... | 19 |
| 2.3.1. RTL Viewer 中可读性最大化..... | 20 |
| 2.3.2. 运行 RTL Viewer..... | 20 |
| 2.4. State Machine Viewer Overview..... | 21 |
| 2.5. Technology Map Viewer 概述..... | 21 |
| 2.6. Netlist Viewer 用户接口..... | 21 |
| 2.6.1. Netlist Navigator 窗格..... | 24 |
| 2.6.2. Properties 窗格..... | 24 |
| 2.6.3. Netlist Viewer 查找窗格..... | 26 |
| 2.7. 原理图视图..... | 26 |
| 2.7.1. 以多选项卡视图显示原理图..... | 26 |
| 2.7.2. 原理图符号..... | 26 |
| 2.7.3. 在 Schematic View 中选择项目..... | 31 |
| 2.7.4. Schematic View 中的快捷菜单命令..... | 31 |
| 2.7.5. 原理图中进行过滤..... | 32 |
| 2.7.6. 在 Schematic View 中查看节点内容..... | 32 |
| 2.7.7. 在 Schematic View 中移动节点..... | 34 |
| 2.7.8. 在 Technology Map Viewer 中查看 LUT 表达..... | 34 |
| 2.7.9. 缩放控制..... | 35 |
| 2.7.10. Bird's Eye View 导览..... | 35 |

| | |
|---|-----------|
| 2.7.11. 原理图分页..... | 35 |
| 2.7.12. 关注原理图页面中的网络..... | 36 |
| 2.8. State Machine Viewer..... | 36 |
| 2.8.1. State Diagram View..... | 37 |
| 2.8.2. State Transition Table..... | 37 |
| 2.8.3. State Encoding Table..... | 37 |
| 2.8.4. Switch Between State Machines..... | 38 |
| 2.9. 交叉探查 Source Design File 和其他 Intel Quartus Prime Windows..... | 38 |
| 2.10. 从其他 Intel Quartus Prime 窗口交叉探查 Netlist Viewer..... | 38 |
| 2.11. 查看时序路径..... | 39 |
| 2.12. 优化设计网表修订历史..... | 40 |
| 3. 时序收敛与优化..... | 42 |
| 3.1. 优化 Multi Corner 时序..... | 42 |
| 3.2. 关键路径..... | 42 |
| 3.2.1. 查看关键路径..... | 43 |
| 3.3. 时序收敛的设计评估..... | 43 |
| 3.3.1. 查看编译结果..... | 43 |
| 3.3.2. 查看时序路径详细信息..... | 51 |
| 3.3.3. 调整和重新编译..... | 54 |
| 3.4. 设计分析..... | 55 |
| 3.4.1. 被忽略的时序约束..... | 55 |
| 3.4.2. I/O 时序..... | 56 |
| 3.4.3. Register-to-Register 时序分析..... | 56 |
| 3.5. 时序优化..... | 60 |
| 3.5.1. 显示失败路径的时序收敛建议..... | 60 |
| 3.5.2. 时序优化向导..... | 61 |
| 3.5.3. 可选 Fitter 设置..... | 62 |
| 3.5.4. I/O 时序优化技术..... | 63 |
| 3.5.5. Register-to-Register 时序优化技术..... | 67 |
| 3.5.6. Logic Lock (Standard) Assignments..... | 72 |
| 3.5.7. 位置约束..... | 73 |
| 3.5.8. 亚稳定性分析和优化技术..... | 74 |
| 3.6. 外设到内核寄存器布局和布线优化 (P2C) | 74 |
| 3.6.1. 在 Advanced Fitter Setting 对话框中设置外设到内核优化..... | 75 |
| 3.6.2. 在 Assignment Editor 中设置外设到内核优化..... | 75 |
| 3.6.3. 在 Fitter Report 中查看外设到内核优化..... | 76 |
| 3.7. 脚本支持..... | 77 |
| 3.7.1. 初始编译设置..... | 77 |
| 3.7.2. I/O 时序优化技术..... | 78 |
| 3.7.3. Register-to-Register 时序优化技术..... | 78 |
| 3.8. 时序收敛和优化修订历史..... | 79 |
| 4. 区域优化..... | 82 |
| 4.1. 资源使用情况..... | 82 |
| 4.1.1. Flow Summary 报告..... | 82 |
| 4.1.2. Fitter 报告..... | 82 |
| 4.1.3. Analysis 和 Synthesis 报告..... | 82 |
| 4.1.4. 编译消息..... | 83 |

| | |
|--|-----------|
| 4.2. 优化资源利用率..... | 83 |
| 4.2.1. Using the Resource Optimization Advisor..... | 83 |
| 4.2.2. 资源使用问题概述..... | 83 |
| 4.2.3. I/O 管脚使用情况或布局..... | 84 |
| 4.2.4. 逻辑资源使用或布局..... | 84 |
| 4.2.5. 布线..... | 88 |
| 4.3. 脚本支持..... | 90 |
| 4.3.1. 初始编译设置..... | 90 |
| 4.3.2. 资源使用优化技术..... | 91 |
| 4.4. 区域优化修订历史..... | 91 |
| 5. 分析和优化设计平面布局规划..... | 93 |
| 5.1. Chip Planner 中的设计布局规划分析..... | 93 |
| 5.1.1. 启动 Chip Planner..... | 93 |
| 5.1.2. Chip Planner GUI 组件..... | 94 |
| 5.1.3. 查看特定体系结构设计信息..... | 95 |
| 5.1.4. 查看器件中可用的时钟网络..... | 96 |
| 5.1.5. 查看 I/O Bank..... | 97 |
| 5.1.6. 查看高速串行接口 (HSSI) | 97 |
| 5.1.7. 查看已布局节点的源和目标..... | 98 |
| 5.1.8. 查看已布局资源的扇入和扇出连接..... | 99 |
| 5.1.9. 生成即时扇入和扇出连接..... | 100 |
| 5.1.10. 在 Chip Planner 中管理路径..... | 100 |
| 5.1.11. 在 Chip Planner 中查看约束..... | 102 |
| 5.1.12. 在 Chip Planner 中查看高速和低功耗 Tile..... | 103 |
| 5.1.13. Viewing Design Partition Placement..... | 103 |
| 5.2. Logic Lock (Standard)区域..... | 104 |
| 5.2.1. Logic Lock (Standard)区域的属性..... | 104 |
| 5.2.2. 创建 Logic Lock (Standard)区域..... | 104 |
| 5.2.3. 定制 Logic Lock 区域的形状..... | 107 |
| 5.2.4. Placing Logic Lock (Standard) Regions..... | 108 |
| 5.2.5. 将器件资源放入 Logic Lock (Standard)区域..... | 109 |
| 5.2.6. Hierarchical (Parent and Child) Logic Lock (Standard) Regions..... | 111 |
| 5.2.7. 其他 Intel Quartus Prime Logic Lock (Standard)设计功能..... | 112 |
| 5.2.8. Logic Lock (Standard)区域窗口..... | 112 |
| 5.3. 在 Chip Planner 中使用 Logic Lock (Standard)区域..... | 113 |
| 5.3.1. 在 Chip Planner 中查看 Logic Lock (Standard)区域之间的连接..... | 113 |
| 5.3.2. Using Logic Lock (Standard) Regions with the Design Partition Planner..... | 114 |
| 5.4. 脚本支持..... | 114 |
| 5.4.1. Initializing and Uninitializing a Logic Lock (Standard) Region..... | 114 |
| 5.4.2. Creating or Modifying Logic Lock (Standard) Regions..... | 115 |
| 5.4.3. Obtaining Logic Lock (Standard) Region Properties..... | 115 |
| 5.4.4. Assigning Logic Lock (Standard) Region Content..... | 115 |
| 5.4.5. Save a Node-Level Netlist for the Entire Design into a Persistent Source File.. | 115 |
| 5.4.6. Setting Logic Lock (Standard) Assignment Priority..... | 116 |
| 5.4.7. 通过 Tcl 命令约束虚拟管脚..... | 116 |
| 5.5. 分析和优化设计布局规划修订历史..... | 116 |

| | |
|--|------------|
| 6. 网表优化和物理综合..... | 118 |
| 6.1. 物理综合优化..... | 118 |
| 6.1.1. 使能物理综合优化..... | 118 |
| 6.1.2. 物理综合选项..... | 119 |
| 6.1.3. Perform Register Retiming for Performance..... | 120 |
| 6.1.4. Preventing Register Movement During Retiming..... | 121 |
| 6.2. 应用网表优化..... | 122 |
| 6.2.1. WYSIWYG 原语再综合..... | 123 |
| 6.2.2. Saving a Node-Level Netlist..... | 124 |
| 6.3. Viewing Synthesis and Netlist Optimization Reports..... | 125 |
| 6.4. 脚本支持..... | 125 |
| 6.4.1. 综合网表优化..... | 126 |
| 6.4.2. 物理综合优化..... | 126 |
| 6.4.3. Back-Annotating Assignments..... | 127 |
| 6.5. 网表优化和物理综合修订历史..... | 127 |
| 7. 使用 Chip Planner 的工程变更命令..... | 129 |
| 7.1. 工程变更命令..... | 129 |
| 7.1.1. 性能保留..... | 129 |
| 7.1.2. 编译时间..... | 130 |
| 7.1.3. 验证..... | 130 |
| 7.1.4. 更改修改记录..... | 130 |
| 7.2. ECO 设计流程..... | 130 |
| 7.3. Chip Planner 概述..... | 132 |
| 7.3.1. 打开 Chip Planner..... | 132 |
| 7.3.2. Chip Planner 任务和分层..... | 132 |
| 7.4. 使用 Chip Planner (布局规划视图) 执行 ECO..... | 133 |
| 7.4.1. 创建, 删除和移动原子..... | 133 |
| 7.4.2. 查看并保存网表变化..... | 133 |
| 7.5. 在 Resource Property Editor 中执行 ECO..... | 133 |
| 7.5.1. 逻辑单元..... | 133 |
| 7.5.2. 自适应逻辑模块..... | 135 |
| 7.5.3. FPGA I/O 元件..... | 137 |
| 7.5.4. FPGA RAM 块..... | 141 |
| 7.5.5. FPGA DSP 块..... | 142 |
| 7.6. Change Manager (更改管理器) | 143 |
| 7.6.1. Change Manager 中的复杂更改..... | 144 |
| 7.6.2. 管理 Signal Probe 信号..... | 144 |
| 7.6.3. 导出更改..... | 144 |
| 7.7. 脚本支持..... | 144 |
| 7.8. 常规 ECO 应用程序..... | 144 |
| 7.8.1. 使用 Chip Planner 调整 I/O 的驱动强度..... | 145 |
| 7.8.2. 使用 Chip Planner 修改 PLL 属性..... | 146 |
| 7.8.3. PLL 属性..... | 146 |
| 7.8.4. 修改资源原子之间的连接性..... | 148 |
| 7.9. 发布 ECO 的步骤..... | 149 |
| 7.10. 使用 Chip Planner 的工程变更命令修订历史..... | 149 |

| | |
|--|------------|
| A. Intel Quartus Prime Standard Edition 用户指南..... | 151 |
|--|------------|

1. 设计优化概述

典型设计流程中，开发的早期阶段专注于满足时序，面积和功耗目标。一旦设计达到这些目标后，则集中投入于性能提升。本章介绍 Intel® Quartus® Prime 软件中可用以实现最高设计性能的技术和工具。

FPGA 设计优化需要一种多维方式来实现减小面积，缩短关键路径延迟，降低功耗和减少运行时间的设计目标。Intel Quartus Prime 软件中包含的各种向导可针对解决每个问题。通过实现向导中的建议，可减少设计迭代所花费的时间。

相关链接

[Intel Quartus Prime 设计软件-支持中心](#)

1.1. 器件考量

所有 Intel FPGA 具有独特时序模型，其中包含了器件中物理元件的延迟信息，例如组合式自适应逻辑模块，存储器块，交互连接和寄存器。延迟包括目标 FPGA 操作条件的全部有效组合。此外，器件尺寸和封装决定管脚约束和资源可用性。

相关链接

[利用 FPGA 时序模型保证芯片性能](#)
[Intel FPGA 白皮书 \(PDF\)](#)

1.1.1. 器件迁移考量

如果预计在设计周期的后期更改目标器件，则请从周期开始就进行迁移规划。该策略有助于将后期设计变更最小化。

在 Intel Quartus Prime 软件中选择设计的目标器件时，通过点击 **Device** 对话框中的 **Migration Devices** 按钮可就会看到可兼容器件列表。

相关链接

[移植器件对话框](#)
[Intel Quartus Prime Help 中](#)

1.2. 初始编译的必需设置

根据您的约束和设置，编译结果会有显著差异。Intel Quartus Prime 软件中，各设置和选项的默认值提供了编译时间，资源利用和时序性能之间的最佳权衡。在 Intel Quartus Prime 软件中编译设计之前，请先考虑如下指导。

1.2.1. I/O 约束指导

在 FPGA 设计中，I/O 标准和驱动强度会影响 I/O 时序。

- 指定 I/O 约束时，请确保 Intel Quartus Prime 软件使用精确的 I/O 时序延迟进行时序分析和 Fitter 优化。
- 如果 PCB 布局未指明管脚位置，则保留管脚位置无约束。该技术可支持 Compiler 搜索最佳布局。否则，进行管脚约束以适当约束编译。

相关链接

I/O 规划概述

In *Intel Quartus Prime Standard Edition User Guide: Design Constraints*

1.2.2. 时序约束指导

为获得最佳结果，请使用实时要求。如果应用高于设计需要的时序要求，则可能导致 Compiler 提高资源使用，电源利用率或编译时间以进行协调。

综合型时序要求设置可实现最佳效果，原因如下：

- 正确的时序约束能使软件充分运行从而实现设计中关键时序部分的性能优化并同时协调性能。该优化还可设计中非关键部分的使用面积和电源利用率。
- 使能后，Intel Quartus Prime 软件基于时序要求执行物理综合优化。
- Depending on the **Fitter Effort** setting, the Fitter can reduce runtime if the design meets the timing requirements.

Intel Quartus Prime Timing Analyzer 决定设计实现是否符合时序要求。Compilation Report 显示设计是否满足时序要求，而时序分析报告命令提供有关时序路径的详细信息。

相关链接

- [时序收敛与优化](#) (第 42 页)
- [使用 Intel Quartus Prime Timing Analyzer](#)
Intel Quartus Prime Standard Edition User Guide: Timing Analyzer
- [Intel Quartus Prime Timing Analyzer Cookbook](#)
- [高级设置 \(Fitter\)](#)
Intel Quartus Prime Help 中

1.2.3. Partitions and Floorplan Assignments for Incremental Compilation

The Intel Quartus Prime incremental compilation feature enables hierarchical and team-based design flows in which you can compile parts of your design while other parts of your design remain unchanged. You can also Import parts of your design from separate Intel Quartus Prime projects.

Using incremental compilation for your design with good design partitioning methodology helps to achieve timing closure. Creating design partitions on some of the major blocks in your design and assigning them to Logic Lock (Standard)[™] regions, reduces Fitter time and improves the quality and repeatability of the results. Logic Lock (Standard) regions are flexible, reusable floorplan location constraints that help you place logic on the target device. When you assign entity instances or nodes to a Logic Lock (Standard) region, you direct the Fitter to place those entity instances or nodes inside the region during fitting.

Using incremental compilation helps you achieve timing closure block by block and preserve the timing performance between iterations, which aid in achieving timing closure for the entire design. Incremental compilation may also help reduce compilation times.

注意: If you plan to use incremental compilation, you must create a floorplan for your design. If you are not using incremental compilation, creating a floorplan is optional.

相关链接

- [缩短编译时间](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*
- [增量式编译分区和布局约束的最佳实践](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*

1.3. 权衡和限制

各优化目标之间可能会相互冲突，因此需要协调相冲突的目标。

表 1. 设计优化中的权衡示例

| 权衡 (Trade-off) | 备注 |
|-------------------------|---|
| 资源使用和关键路径时序。 | 某些技术（如逻辑复制）可以增加面积为代价来改善时序性能。 |
| 功率要求会影响面积和时序权衡。 | 例如，减少可用高速 tile 的数量，或尝试缩短高功率网路牺牲关键路径网络。 |
| 系统成本和上市时间考量因素会影响对器件的选择。 | 例如，具有较高速度等级或更多时钟网络的器件可通过更高功耗和系统成本促进时序收敛。 |

最后，过于严格限制设计可行性的约束可能导致所选器件无可解决方案。如果因资源限制，时序约束或功率约束导致 **Fitter** 无法解析设计，则请考虑改写 HDL 代码部分。

1.3.1. Preserving Results and Enabling Teamwork

For some Intel Quartus Prime Fitter algorithms, small changes to the design can have a large impact on the final result. For example, a critical path delay can change by 10% or more because of seemingly insignificant changes. If you are close to meeting your timing objectives, you can use the Fitter algorithm to your advantage by changing the fitter seed, which changes the pseudo-random result of the Fitter.

Conversely, if you cannot meet timing on a portion of your design, you can partition that portion and prevent it from recompiling if an unrelated part of the design is changed. This feature, known as incremental compilation, can reduce the Fitter runtimes by up to 70% if the design is partitioned, such that only small portions require recompilation at any one time.

When you use incremental compilation, you can apply design optimization options to individual design partitions and preserve performance in other partitions by leaving them untouched. Many optimization techniques often result in longer compilation times, but by applying them only on specific partitions, you can reduce this impact and complete iterations more quickly.

In addition, by physically floorplanning your partitions with Logic Lock (Standard) regions, you can enable team-based flows and allow multiple people to work on different portions of the design.

相关链接

针对层次和基于团队设计的 [Intel Quartus Prime 增量式编译](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*

1.3.2. 减少面积

默认情况下，**Intel Quartus Prime Fitter** 可将设计物理性扩展至整个器件，以满足时序约束。如果希望通过优化设计占用最小面积，则可更改此行为。如果需要缩小面积，则可启用特定物理综合选项修改网表以创建更具区域效率的实现，但代价是增加运行时间和降低性能。

相关链接

- [区域优化](#) (第 82 页)
- [网表优化和物理综合](#) (第 118 页)

1.3.3. 减短关键路径延迟

为满足涉及多个时钟，布线资源和面积约束的复杂时序要求，**Intel Quartus Prime** 软件提供介于综合，布局规划编辑，布局布线以及时序分析处理之间的紧密交互。

默认情况下，运行 **Intel Quartus Prime Fitter** 以满足时序要求，并在达到要求后停止。因此，实际约束对时序收敛至关重要。

收敛不足的设计会导致结果欠佳。而对于过度收敛的设计，**Fitter** 可能会牺牲真正关键路径而过度优化非关键路径。此外，面积和编译时间也可能随之增加。

对于资源使用率高的设计，**Intel Quartus Prime Fitter** 可能无法找到合法位置。该情况下，**Fitter** 会自动修改设置以尝试协调面积方面的性能。

Intel Quartus Prime Fitter 提供的高级选项可在您正确设置约束时帮助提高设计性能。请使用 **Timing Optimization Advisor** 确定适合您设计的最佳选项。

If you use incremental compilation, you can help resolve inter-partition timing requirements by locking down results, one partition at a time, or by guiding the placement of the partitions with Logic Lock (Standard) regions. You might improve the timing on such paths by placing the partitions optimally to reduce the length of critical paths. Once the inter-partition timing requirements are met, use incremental compilation to preserve the results and work on partitions that have not met timing requirements.

高密度 **FPGA** 中，布线占据关键路径时序的主要部分。因此，复制或重新定时逻辑可促进 **Fitter** 减少关键路径上的延迟。**Intel Quartus Prime** 软件提供的网表优化按钮和物理综合选项可提高设计性能，但会大幅增加编译时间和面积。仅开启有助于保持合理编译时间和资源使用的选项。或者，可修改 HDL 以手动复制或调整时序逻辑。

相关链接

[关键路径](#) (第 42 页)

1.3.4. 降低功耗

Intel Quartus Prime 软件具有帮助降低功耗的功能。功率优化选项控制用于综合和适配 (**Synthesis and the Fitter**) 的电源驱动编译设置。

[相关链接](#)

[功耗优化](#)

In *Intel Quartus Prime Standard Edition User Guide: Power Analysis and Optimization*

1.3.5. 减少运行时间

许多 Fitter 设置会影响编译时间。Intel Quartus Prime 软件中的大部分默认设置都以减少编译时间为目标。也可基于您工程的要求修改这些设置。

Intel Quartus Prime 软件支持在多处理器计算机中并行编译。这样编译时间可最多减少 15%。

You can also reduce compilation time with your iterations by using incremental compilation. Use incremental compilation when you want to change parts of your design, while keeping most of the remaining logic unchanged.

[相关链接](#)

[缩短编译时间](#)

In *Intel Quartus Prime Standard Edition User Guide: Compiler*

1.4. Intel Quartus Prime 软件工具用于设计优化

Intel Quartus Prime 软件提供各种优化设计的工具。

1.4.1. Design Visualization Tool (设计可视化工具)

Intel Quartus Prime 软件提供可显示设计图形表达的工具。

表 2. Visualization Tool (可视化工具)

| 工具 | 说明 |
|---|---|
| RTL Viewer | 提供综合和布局布线前的设计原理图。 |
| technology map viewer (查看 FPGA 中的实际连线情况) | 提供综合和布局布线后所选器件体系结构中设计实现的原理图。可选择性包含时序信息。 |
| Design Partition Planner | 以分区或实体级显示设计，并可显示实体间的连接。 |
| Design Partition Planner 和 Chip Planner (通过增量式编译) | 允许在更高级对设计进行分区和布局。 |
| Chip Planner | 支持进行布局规划约束，实现工程变更命令 (engineering change orders, ECO)，执行功率分析和可视化关键路径和路由拥塞。 |

[相关链接](#)

- [Chip Planner 中的设计布局规划分析](#) (第 93 页)
- [优化设计网表](#) (第 17 页)
- [RTL Viewer 概述](#) (第 19 页)

1.4.2. 向导 (Advisors)

Intel Quartus Prime 软件包含多种向导以帮助优化设计并缩短编译时间。

各向导根据工程设置和设计约束提供建议。这些建议有助于适配工程，满足时序或功率要求，或提高设计性能。

向导中的建议从宽泛到具体。适用情况下，其以阶段分门别类并按复杂度呈现。

向导分为：

- Resource Optimization Advisor (资源优化)
- Timing Optimization Advisor (时序优化)
- Power Optimization Advisor (功率优化)
- Compilation Time Advisor (编译时间)
- Pin Optimization Advisor (管脚优化)
- Incremental Compilation Advisor (增量编译)

相关链接

- [编译时间向导](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*
- [Intel Quartus Prime 软件中的向导](#) (第 0 页)
Intel Quartus Prime Help 中
- [时序优化向导](#) (第 61 页)
- [功耗优化向导](#)
In *Intel Quartus Prime Standard Edition User Guide: Power Analysis and Optimization*

1.4.3. 设计管理

Design Space Explorer II 工具 (DSE II) 为您运行设计设置实验提供便捷有效的方法。可在 PC 上局部运行单个编译，也可远程使用运算中心 (compute farm) 资源。

相关链接

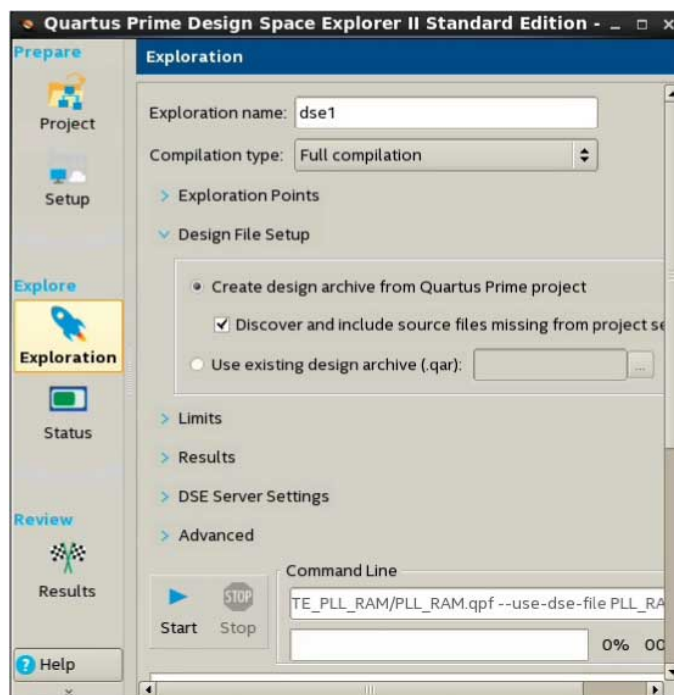
[Design Space Explorer II](#) (第 12 页)

1.5. Design Space Explorer II

Design Space Explorer II 工具 (**Tools > Launch Design Space Explorer II**) 支持针对资源，性能和功率目标进行最佳项目设置。Design Space Explorer II (DSE II) 结合设置和约束对设计进行处理，并报告关于设计的最佳设置。可利用 DSE II 的并行功能在多台计算机上进行编译。

如果设计已接近满足时间或面积要求，也可使用 DSE II 尝试不同 seed 以找出符合时序或面积要求的那个 seed。

图 1. Design Space Explorer II User Interface



可在设计过程的任何阶段运行 DSE II；但由于设计中的较大改动会抵消来自优化设置的获益，因此 Intel FPGA 建议在设计周期的后期才运行 DSE II。

相关链接

- [Design Space Explorer II 工具](#)
Intel Quartus Prime Help 中
- [使用 Design Space Explorer](#)
21 分钟在线课程

1.5.1. DSE II 工作原理

DSE II 中，管理点 (*exploration point*) 为 Analysis & Synthesis, Fitter 和布局设置的集合，而一组 *exploration point* 称为一个设计管理 (*design exploration*)。一个设计管理可包含各种适配器 (*fitter*) seed。

DSE II 使用与每个管理点对应的设置编译设计。编译完成后，DSE II 对照您指定的优化目标评估性能数据。可指示 DSE II 针对时序，面积和功率进行优化。

相关链接

[电源驱动优化的设计空间浏览器 II](#)

In Intel Quartus Prime Standard Edition User Guide: Power Analysis and Optimization

1.5.1.1. 使用计算资源

可配置 DSE II 以利用您的计算资源来运行设计探索。DSE II GUI 中，**Setup** 页面包含工作启动选项，而 **Status** 页面支持监督和控制工作。

DSE II 支持在本地计算机上编译或使用 LSF, SSH 或 Torque 在远程主机运行编译。对于 SSH，还可定义以逗号分隔的远程主机清单。

如果有笔记本电脑或标准计算机，就可使用单一编译功能在具有更高计算性能和内存容量的工作站上编译设计。

在运算中心上运行时，可在提交全部工作后指示 DSE II 安全退出，而编译保持运行直至完成。或者，编译完成后可收到电子邮件。

如果使用 SSH 启动作业，则远程主机必须使能公钥和私钥验证。对于采用短语加密的私钥，远程主机必须运行 ssh 密钥代理解密私钥，从而 quartus_dse 可执行文件能访问该密钥。

注意: Windows 远程主机需要 Cygwin 的 sshd 服务器和 PuTTY。

相关链接

- [建立页 \(Design Space Explorer II\)](#)
Intel Quartus Prime Help 中
- [状态页 \(Design Space Explorer II\)](#)
Intel Quartus Prime Help 中

1.5.1.2. 优化参数

DSE II 针对您需要优化的内容提供一系列预定义管理空间。此外，还可定义一套编译 seed。管理点的数量为管理 seed 数乘以管理模式的数量。

注意: 预定义空降的可用性取决于设计所针对的器件系列。

在 DSE GUI 中，请在 **Exploration** 页面指定这些设置。

相关链接

[探索页 \(Design Space Explorer II\)](#)
Intel Quartus Prime Help 中

1.5.1.3. 结果管理

DSE II 比较编译结果以确定关于设计的 Intel Quartus Prime 软件最佳设置。**Report** 页面显示结果摘要。

探索中，DSE II 从所有探索点的全部时序拐点中选择最差情况下的最佳 slack（时序余量）。如果要优化最差情况下的设置时序余量或保持时序余量，可在 Intel Quartus Prime 软件中指定时序约束。

磁盘空间

默认情况下，DSE II 保存所有编译数据。可限制编译完成后要保存的文件类型来节省磁盘空间。这些设置位于 **Exploration** 页面，**Results** 部分。

报告

DSE II 具有助于快速确定重要设计指标的报告工具，例如所有探索点中，较差情况的时序余量。

DSE II 针对其探索的全部点提供性能数据报告并将信息保存在工程目录下的 `project-name.dse.rpt` 文件中。DSE II 将探索点设置存档到 Intel Quartus Prime Archive Files (`.qar`) 中。

相关链接

报告页 ([Design Space Explorer II](#))
Intel Quartus Prime Help 中

1.5.2. 使用 DSE II Utility 执行设计管理

注意: 运行 DSE II 之前，指定设计的时序约束。

本说明涵盖运行设计管理时需定义的设计类型。有关 GUI 中所有可用选项的详细信息，请参阅 *Intel Quartus Prime Help*。

使用 DSE II 工具执行设计管理：

1. 启动 DSE II 工具。

如果您的 Intel Quartus Prime 软件中打开的工程并启动了 DSE II，则会出现一个对话框询问是否关闭 Intel Quartus Prime 软件。请点击 **Yes**。

2. 在 **Project** 页，指定要管理的工程和版本。
3. **Setup** 页面中，指定执行本地管理或远程管理，以及设置作业启动。
4. 在 **Exploration** 页，指定优化设置和目标。
5. 配置完成后，点击 **Start**。

相关链接

- [Design Space Explorer II 工具](#)
Intel Quartus Prime Help 中
- [使用 Design Space Explorer](#)
21 分钟在线课程

1.6. 设计优化概述修订历史

以下修订历史适用于本章：

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|------------|------------------------|--|
| 2018.09.24 | 18.1.0 | Intel Quartus Prime Standard 版用户指南首次发布 |
| 2018.05.07 | 18.0.0 | <ul style="list-style-type: none">• 常规主题重组。 |
| 2015.11.02 | 15.1.0 | 将 <i>Quartus II</i> 实体更改为 <i>Quartus Prime</i> 。 |
| 2014.12.15 | 14.1.0 | <ul style="list-style-type: none">• 将 Fitter Settings, Analysis & Synthesis Settings 和 Physical Synthesis Optimizations 的位置更新到 Compiler Settings。• 更新了 DSE II 内容。 |
| 继续... | | |

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|---------------|------------------------|--|
| June 2014 | 14.0.0 | 更新了格式。 |
| November 2013 | 13.1.0 | 少许修订 HardCopy。 |
| May 2013 | 13.0.0 | 添加关于初始编译要求的信息。本节已从 Intel Quartus Prime 手册的 Area Optimization （区域优化）章节中删除。少许更新以描述时序和区域优化章节的划分。 |
| June 2012 | 12.0.0 | 删除了调查链接。 |
| November 2011 | 10.0.3 | 模板更新。 |
| December 2010 | 10.0.2 | 更换为新的文档模板。文档内容未更改。 |
| August 2010 | 10.0.1 | 更正了链接 |
| July 2010 | 10.0.0 | 首次发布。章节基于第二卷第 III 部分的主题和文字。 |

相关链接

文档存档

欲了解 *Intel Quartus Prime* 手册以前的版本，请搜索文档存档。

2. 优化设计网表

本章节说明如何使用 Intel Quartus Prime Netlist Viewer 分析和调试您的设计。

As FPGA designs grow in size and complexity, the ability to analyze, debug, optimize, and constrain your design is critical. With today's advanced designs, several design engineers are involved in coding and synthesizing different design blocks, making it difficult to analyze and debug the design. The Intel Quartus Prime RTL Viewer, State Machine Viewer, and Technology Map Viewer provide powerful ways to view your initial and fully mapped synthesis results during the debugging, optimization, and constraint entry processes.

相关链接

- [使用 Netlist Viewers 的 Intel Quartus Prime 设计流程](#) (第 18 页)
- [State Machine Viewer Overview](#) (第 21 页)
- [RTL Viewer 概述](#) (第 19 页)
- [Technology Map Viewer 概述](#) (第 21 页)
- [原理图中进行过滤](#) (第 32 页)
- [查看时序路径](#) (第 39 页)

2.1. 何时使用 Netlist Viewer: 分析设计问题

You can use the Netlist Viewers to analyze and debug your design. The following simple examples show how to use the RTL Viewer, State Machine Viewer, and Technology Map Viewer to analyze problems encountered in the design process.

Using the RTL Viewer is a good way to view your initial synthesis results to determine whether you have created the necessary logic, and that the logic and connections have been interpreted correctly by the software. You can use the RTL Viewer and State Machine Viewer to check your design visually before simulation or other verification processes. Catching design errors at this early stage of the design process can save you valuable time.

If you see unexpected behavior during verification, use the RTL Viewer to trace through the netlist and ensure that the connections and logic in your design are as expected. You can also view state machine transitions and transition equations with the State Machine Viewer. Viewing your design helps you find and analyze the source of design problems. If your design looks correct in the RTL Viewer, you know to focus your analysis on later stages of the design process and investigate potential timing violations or issues in the verification flow itself.

可使用 **Technology Map Viewer** 查看 **Analysis** 和 **Synthesis** 结束时的结果。如果已在 **Fitter** 阶段编译了设计，则可在 **Technology Map Viewer (Post-Mapping)** 中查看“映射后” (post-mapping) 网表，适配后的网表在 **Technology Map Viewer** 中查看。如果仅执行 **Analysis** 和 **Synthesis**，则两个 **Netlist Viewer** 显示相同“映射后” (post-mapping) 网表。

此外，可使用 **RTL Viewer** 或 **Technology Map Viewer** 查找特定信号来源以助于调试您的设计。使用本章中介绍的导航技术可对设计中的内容进行全面搜索。可从您的兴趣点追溯到信号源但前提是确保连接符合预期要求。

Technology Map Viewer 有助于查找网表中的合成后节点，并在优化期间进行约束。该功能对设计中两个寄存器间的多周期时序约束十分有利。从某个 **I/O** 端口开始，在设计中向前后或向后追踪以及通过层次结构层找到兴趣节点或直观检查原理图找到特定寄存器。

贯穿整个 **FPGA** 设计，调试和优化阶段，可通过多种方式使用所有网表查看器以提高分析设计时的工作效率。

相关链接

- [使用 Netlist Viewers 的 Intel Quartus Prime 设计流程](#) (第 18 页)
- [State Machine Viewer Overview](#) (第 21 页)
- [RTL Viewer 概述](#) (第 19 页)
- [Technology Map Viewer 概述](#) (第 21 页)

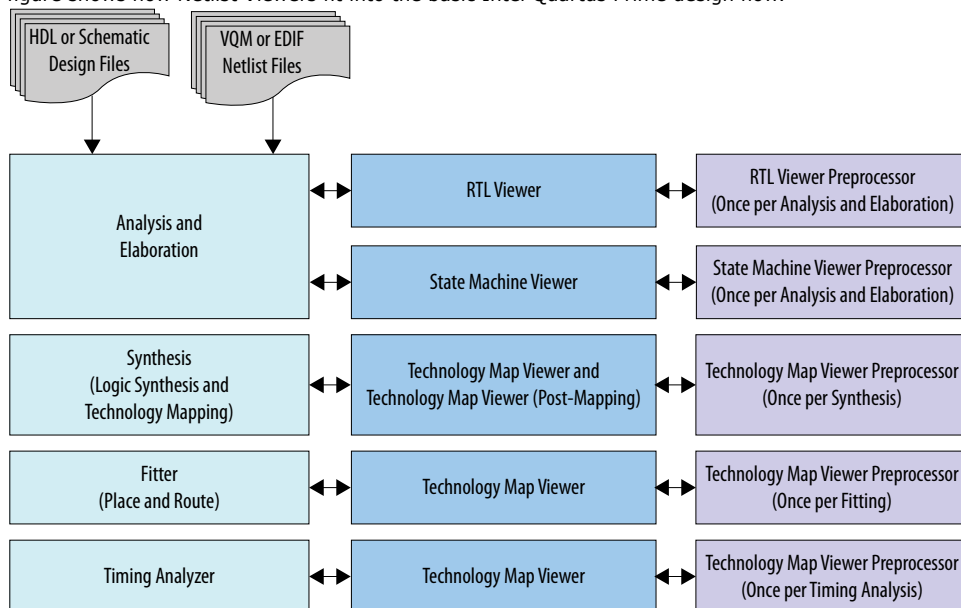
2.2. 使用 Netlist Viewers 的 Intel Quartus Prime 设计流程

编译设计后首次打开其中一个 **Netlist Viewer** 时，会先自动运行预处理器，然后开启 **Netlist Viewer**。

若单击预处理器进程框中的链接转到 **Settings > Compilation Process Settings** 页，在此开启 **Run Netlist Viewers preprocessing during compilation** 选项。如果开启该选项，则预处理将成为完整工程编译流程的一部分，这样即使 **Netlist Viewer** 立即打开而不显示预处理对话框。

图 2. Intel Quartus Prime Design Flow Including the RTL Viewer and Technology Map Viewer

This figure shows how Netlist Viewers fit into the basic Intel Quartus Prime design flow.



Netlist Viewer 可以运行预处理器阶段之前，必须先编译您的设计：

- To open the RTL Viewer or State Machine Viewer, first perform Analysis and Elaboration.
- 要打开 Technology Map Viewer (Post-Fitting)或 Technology Map Viewer (Post-Mapping)，应首先执行 Analysis 和 Synthesis。

Netlist Viewer 显示最近一次成功编译的结果。

- 因此，如果因更改设计而导致 Analysis 和 Elaboration 期间出现错误，则无法查看该新设计文件的网表，但仍可查看最近成功编译的设计文件版本的结果。
- 如果编译期间出现错误，且与工程相应的编译阶段尚未成功运行，则 Netlist Viewer 无法显示；该情况下尝试打开 Netlist Viewer 时，Intel Quartus Prime 软件将发送一条错误消息。

注意：开始新的编译时，如果有开启的 Netlist Viewer，则该 Netlist Viewer 会自动关闭。仅可在编译成功完成后才能重新打开 Netlist Viewer 查看新的设计网表。

2.3. RTL Viewer 概述

RTL Viewer 支持查看 Intel Quartus Prime 软件中 Intel Quartus Prime 集成综合 结果或第三方网表文件的寄存器传输级 (RTL) 图形表示。

可查看对设计执行 Analysis 和 Elaboration 后的结果，只要该设计使用支持的 Intel Quartus Prime 设计实体方式，具体包括 Verilog HDL Design Files (.v)，SystemVerilog Design Files (.sv)，VHDL Design Files (.vhd)，AHDL Text Design Files (.tdf)或原理图 Block Design Files (.bdf)。

还可查看通过综合工具生成 Verilog Quartus Mapping File (.vqm) 或 Electronic Design Interchange Format (.edf) 文件的设计的原子原语层次（例如器件逻辑单元和 I/O 端口）。

RTL Viewer 会在技术映射和综合或适配器优化以前，显示执行 Analysis 和 Elaboration 后，或 Intel Quartus Prime 软件执行网表提取后的设计网表原理图。该视图为初步预优化设计结构图，接近原始源设计。

- 对于通过 Intel Quartus Prime 集成综合 进行综合的设计，该视图显示 Intel Quartus Prime 软件如何转换设计文件。
- 对于利用第三方综合工具进行综合的设计，该视图显示由综合工具生成的网表。

运行 Intel Quartus Prime 工程的 RTL Viewer，首先需分析设计生成 RTL 网表。要分析设计并生成 RTL 网表，请单击 **Processing > Start > Start Analysis & Elaboration**。还可在任何包含 Intel Quartus Prime 编译流程初始 Analysis 和 Elaboration 阶段的处理中执行完整编译。

要打开 RTL Viewer，单击 **Tools > Netlist Viewers > RTL Viewer**。

相关链接

[Netlist Viewer 用户接口](#) (第 21 页)

2.3.1. RTL Viewer 中可读性最大化

显示设计时，RTL Viewer 优化网表以最大化可读性：

- 从现实图中删除无扇出（未连接的输出）和扇入（无连接的输入）的逻辑。
- 隐藏默认连接，例如 V_{CC} 和 GND。
- 将管脚，网络，线路，模块端口和具体逻辑适当分组到总线中。
- 分组恒定总线连接。
- 数值以十六进制格式显示。
- 将原理图中的非门（NOT gate）转换为气泡取反（非）符。
- 将等效组合门控链合并为单个门控；例如，一个馈送 2-input 与门（AND gate）的 2-input 与门（AND gate）转换为单个 3-input 与门（AND gate）。
- 将状态机逻辑转换成状态图，状态跳转表和状态编码表，并可通过 State Machine Viewer 查看。

相关链接

[State Machine Viewer Overview](#) (第 21 页)

2.3.2. 运行 RTL Viewer

对 Intel Quartus Prime 工程运行 RTL Viewer：

1. 单击 **Processing > Start > Start Analysis & Elaboration** 分析设计以生成 RTL 网表。

还可在任何包含 Intel Quartus Prime 编译流程初始 Analysis 和 Elaboration 阶段的处理中执行完整编译。

2. 单击 **Tools > Netlist Viewers > RTL Viewer** 打开 RTL Viewer。

2.4. State Machine Viewer Overview

The State Machine Viewer presents a high-level view of finite state machines in your design. The State Machine Viewer provides a graphical representation of the states and their related transitions, as well as a state transition table that displays the condition equation for each of the state transitions, and encoding information for each state.

To run the State Machine Viewer, on the Tools menu, point to **Netlist Viewers** and click **State Machine Viewer**. To open the State Machine Viewer for a particular state machine, double-click the state machine instance in the RTL Viewer.

2.5. Technology Map Viewer 概述

Intel Quartus Prime Technology Map Viewer 提供 Analysis 和 Synthesis 后或 Fitter 将设计映射到目标器件后，FPGA 设计的技术特定图形表示。

Technology Map Viewer 显示设计中原子图原层次（如，器件逻辑单元和 I/O 端口）。对于支持的器件系列，还可查看内部寄存器和逻辑单元（LCELL）中的查找列表（LUT），以及 I/O 原子原语中的寄存器。

可能的情况下，Intel Quartus Prime 软件在整个综合过程中保留每层次中的端口名称。但软件可能会更改或删除设计中的端口名称。例如，软件在合成期间删除未连接或由 GND 或 V_{CC} 驱动的端口。如果某端口名称改变，则软件会在设计中为其约束一个相关用户逻辑名或常规端口名，如 IN1 或 OUT1。

可查看综合，拟合或时序分析后的 Intel Quartus Prime 技术映射结果。要运行 Intel Quartus Prime 工程的 Technology Map Viewer，在 **Processing** 菜单上，指向 **Start** 并单击 **Start Analysis & Synthesis** 将设计综合并映射到目标技术。在此阶段，Technology Map Viewer 显示的“映射后”网表与 Technology Map Viewer (Post-Mapping)相同。您还可执行完整编译，或任何包含编译流程综合阶段的处理。

对于已完成 Fitter 阶段的设计，Technology Map Viewer 会显示 Fitter 如何通过物理性综合优化更改网表，而 Technology Map Viewer (Post-Mapping)显示“映射后”（post-mapping）网表。如果已完成 Timing Analysis 阶段，就可从 Technology Map Viewer 的 Timing Analyzer 报告中找到时序路径。

要打开 Technology Map Viewer，单击 **Tools > Netlist Viewers > Technology Map Viewer (Post-Fitting)**或 **Technology Map Viewer (Post Mapping)**。

相关链接

- [查看时序路径](#) (第 39 页)
- [在 Schematic View 中查看节点内容](#) (第 32 页)
- [Netlist Viewer 用户接口](#) (第 21 页)

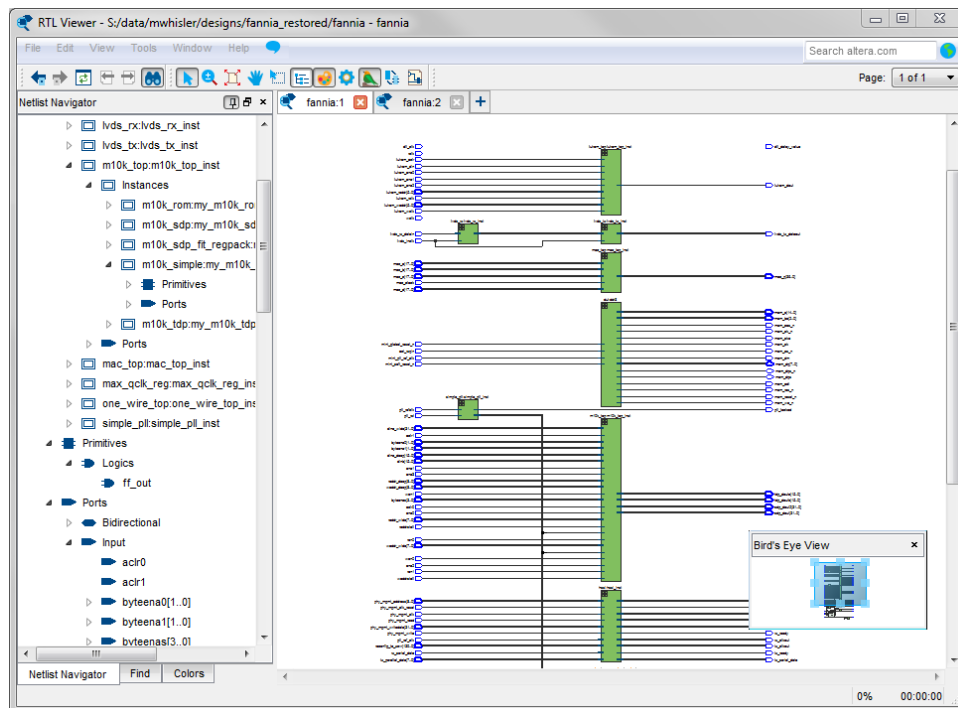
2.6. Netlist Viewer 用户接口

Netlist Viewer 是查看和操作网表中节点和网络的图形用户界面。

RTL Viewer 和 Technology Map Viewer 各由以下主要部分组成：

- **Netlist Navigator** 窗格—显示工程层次。
- **Find** 窗格—支持在原理视图中查找和定位指定设计元件。
- **Properties** 窗格—从快捷菜单选择 **Properties** 后显示所选模块的属性。
- 原理图视图—显示设计内部结构的图形表达。

图 3. RTL Viewer

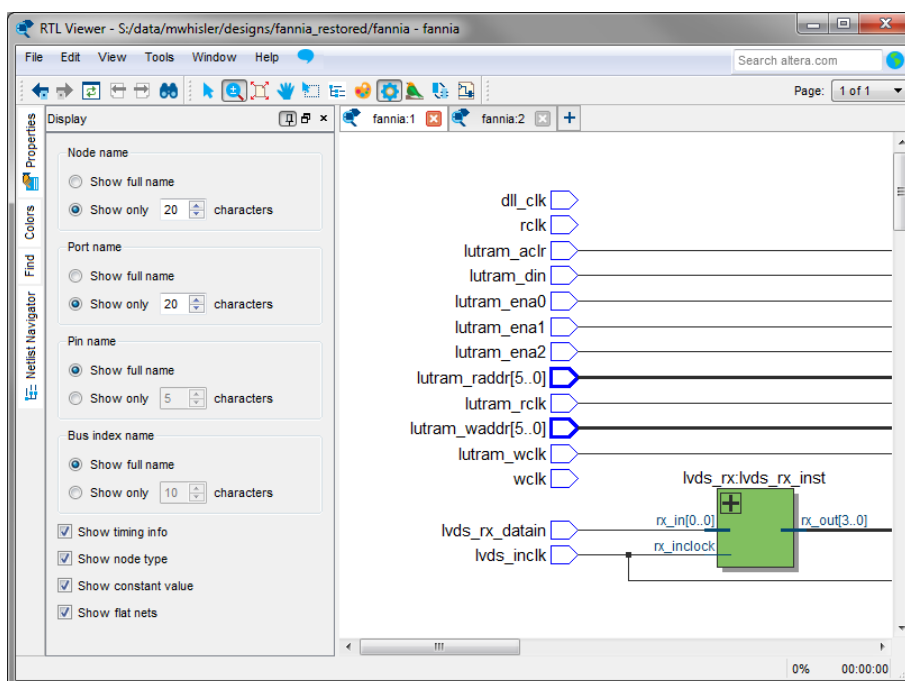


Netlist Viewer 还包含一个为原理图视图提供使用工具的工具栏。

- 使用 **Back** 和 **Forward** 按钮在原理图视图间进行切换。仅在返回时未对视图进行任何更改才可继续前进。这些命令不会撤销操作，例如选择节点。Netlist Viewer 最多存储 10 个操作，包括过滤，层次导航，网表导航和缩放操作。
- **Refresh** 按钮恢复原理图视图并优化布局。如果更改设计并重新编译，而 **Refresh** 不会重新加载数据库。
- 单击 **Find** 按钮打开和关闭 **Find** 窗格。
- 单击 **Selection Tool** 和 **Zoom Tool** 按钮在选择模式和缩放模式间切换。
- 单击 **Fit in Page** 按钮重置原理图视图以包含整个设计。
- 使用 **Hand Tool** 更换查看焦点且不改变视角。
- 单击 **Area Selection Tool** 在某区域内的端口，管脚和节点周围拖动选择框。
- 单击 **Netlist Navigator** 按钮打开或关闭 **Netlist Navigator** 窗格。
- 单击 **Color Settings** 按钮打开 **Colors** 窗格，在此自定义 Netlist Viewer 颜色。

- 单击 **Display Settings** 钮，打开 **Display** 窗格在此指定以下设置：
 - Show full name** 或 **Show only <n> characters**。可单独为 **Node name**, **Port name**, **Pin name** 或 **Bus name** 指定该项。
 - 打开或关闭 **Show timing info**。
 - 打开或关闭 **Show node type**。
 - 打开或关闭 **Show constant value**。
 - 打开或关闭 **Show flat nets**。

图 4. **Display (显示) 设置**



- Bird's Eye View** 按钮打开 **Bird's Eye View** 窗口显示设计的微缩版，可在设计内进行浏览且快速调整原理图视图中的缩放率。
- Show/Hide Instance Pins** 钮交替显示因某些功能而未显示的实例管脚，例如 **Netlist Viewer** 和 **Timing Analyzer** 间的交互探测 (cross-probing)。也可在大量未连接或为使用管脚中过滤节点结果时使用该按钮隐藏未连接的实例管脚。**Netlist Viewer** 默认隐藏实例管脚。
- 如果 **Netlist Viewer** 显示涵盖多个页面，则 **Show Netlist on One Page** 钮会将网表视图调整为单个页面。此操作可使网表追踪更容易。

You can have only one RTL Viewer, one Technology Map Viewer (Post-Fitting), one Technology Map Viewer (Post-Mapping), and one State Machine Viewer window open at the same time, although each window can show multiple pages, each with multiple tabs. For example, you cannot have two RTL Viewer windows open at the same time.

相关链接

- [RTL Viewer 概述](#) (第 19 页)
- [Technology Map Viewer 概述](#) (第 21 页)

- [Netlist Navigator 窗格](#) (第 24 页)
- [Netlist Viewer 查找窗格](#) (第 26 页)
- [Properties 窗格](#) (第 24 页)

2.6.1. Netlist Navigator 窗格

Netlist Navigator (网表导航) 窗格根据设计的层次级别以树形格式显示整个网表。每个级别将类似单元分组为子类别。

Netlist Navigator 窗格支持遍历设计层次查看每个级别的逻辑原理图。还可在 **Netlist Navigator** 中选择要在原理图视图中突出显示的元素。

注意: **Netlist Navigator** 窗格中不罗列原子原语内的节点。

对于设计层次中的每个模块，**Netlist Navigator** 窗格通过以下表格显示可应用的单元。单击“+”图标展开单元。

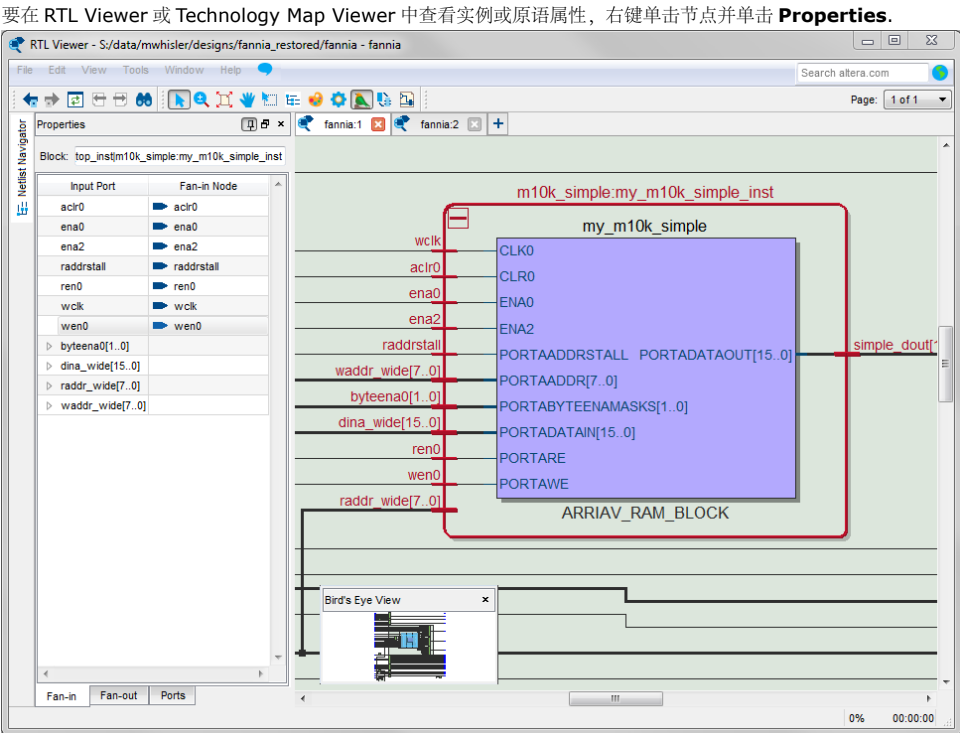
表 3. **Netlist Navigator 窗格单元**

| 单元 | 说明 |
|----------------|--|
| Instances | 设计中的模块或实例，展开后可查看低层次级别。 |
| State Machines | State machine instances in the design that can be viewed in the State Machine Viewer. |
| Primitives | 低级别节点，无法向更低层次级别扩展。这些原语包括： <ul style="list-style-type: none"> • 使用 Intel Quartus Prime 集成综合 时，可在 RTL Viewer 中查看的寄存器和门控。 • 使用第三方综合软件中的 VQM 或 EDIF 时，Technology Map Viewer 或 RTL Viewer 中的逻辑单元原子。 在 Technology Map Viewer 中，可查看具体原子原语的内部实现，但不能遍历层次结构的较低层。 |
| Ports | 层次结构中当前级别的 I/O 端口。 <ul style="list-style-type: none"> • 查看顶层层次级别时，管脚为器件 I/O 管脚，查看底层级别时，管脚是设计的 I/O 端口。 • 当管脚代表总线或管脚阵列时，展开列表视图中的管脚条目查看各个管脚名称。 |

2.6.2. Properties 窗格

可通过 **Properties** 窗格查看实例或原语属性。

图 5. Properties 窗格



Properties 窗格的选项卡中包含所选节点的信息，如下：

- **Fan-in** 选项卡显示 **Input port**（输入端口）和 **Fan-in Node**（扇入节点）。
- **Fan-out** 选项卡显示 **Output port**（输出端口）和 **Fan-out Node**（扇出节点）。
- **Parameters** 选项卡显示示例中的 **Parameter Name**（参数名称）和 **Values**（值）。
- **Ports** 选项卡显示 **Port Name** 和 **Constant** 值，（例如，V_{CC} 或 GND）。下表列出端口的可能值：

表 4. 端口的可能值

| 值 | 说明 |
|-----------------|--|
| V _{CC} | 端口未连接但有具 V _{CC} 值（连接到 V _{CC} ） |
| GND | 端口未连接但具有 GND 值（连接到 GND） |
| -- | 端口已连接具有数值（V _{CC} 或 GND 除外） |
| Unconnected | 端口未连接且无值（挂起） |

如果所选节点为原子原语，则 Properties 窗格显示内部逻辑的原理图。

2.6.3. Netlist Viewer 查找窗格

可通过在 **Find** 窗格中设置以下选项缩小搜索过程的范围：

- 在 **Find** 窗格中单击 **Browse** 以指定搜索层次级别。在 **Select Hierarchy Level** 对话框中，选择要搜索的特定实例。
- 打开 **Include subentities** 选项以在搜索中包含父级实例的子层次。
- 单击 **Options** 打开 **Find Options** 对话框。打开 **Instances**, **Nodes**, **Ports** 或三者的任意组合，以进一步细化搜索参数。

单击 **List** 按钮时，**Find** 对话框下方出现一个进度条。

所有与您设置的条件匹配的结果都罗列在表格中。双击列表中的项目时，相关节点在原理图视图用红色突显。

2.7. 原理图视图

原理图视图显示于 RTL Viewer 和 Technology Map Viewer 的右侧。原理图视图包含表示网表中设计逻辑的原理图。该视图是 RTL Viewer 中查看门级网表和 Technology Map Viewer 中查看技术映射网表的主屏。

RTL Viewer 和 Technology Map Viewer 默认尝试以单页视图显示原理图。如果原理图跨多个页面，则可突出显示某个网络并且使用连接器追踪单个页面中的信号。

2.7.1. 以多选项卡视图显示原理图

RTL Viewer 和 Technology Map Viewer 支持多选项卡式视图。

通过多选项卡式视图，原理图可在不同选型卡中显示。个选项卡式视图中的选择相互独立，但选项卡中的选择集中同步于 Netlist Navigator 窗格。

要创建一个新的选项卡，在选项卡行的末尾单击 **New Tab** 按钮。此时就可从 **Netlist Navigator** 窗格中将节点拖动到原理图视图中。

在选项卡中右键单击查看快捷菜单执行如下操作：

- 使用 **New Tab** 创建空白视图
- 为目标选项卡创建 **Duplicate Tab**
- 选择 **Cascade Tabs**
- 选择 **Tile Tabs**
- 选择 **Close Tab** 关闭目标选项卡
- 选择 **Close Other Tabs** 关闭目标选项卡之外的所有选项卡。

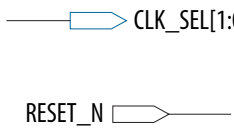
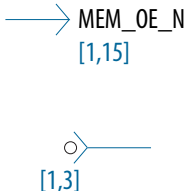
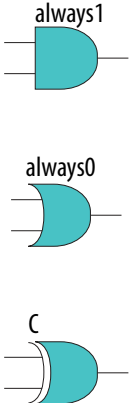
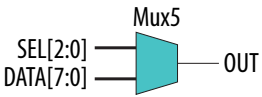
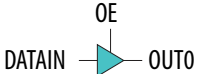
2.7.2. 原理图符号

原理图中节点的符号表示您设计网表中的单元。这些单元包括输入和输出端口，寄存器，逻辑门控，Intel 原语，高级别操作符和层次实例。

注意： 逻辑门控和操作符原语仅在 RTL Viewer 中出现。Technology Map Viewer 中的逻辑以原子原语呈现，例如寄存器和 LCELL。

表 5. 原理图中的符号

本列表罗列并说明可在 RTL Viewer 和 Technology Map Viewer 的原理图中显示的原语和基本符号。

| 符号 | 说明 |
|--|---|
| <p>I/O 端口</p>  | <p>层次结构中当前级别的输入，输出或双向端口。查看顶层层次时，器件输入，输出或双向管脚。该符号还可代表总线。仅显示一条线路连接到双向符号，表示输入和输出路径。</p> <p>输入符出现在原理图的最左侧。输出和双向符出现在原理图的最右侧。</p> |
| <p>I/O 连接器</p>  | <p>输入或输出连接器，表示来自相同层次中另一页面的网络。要转到包含源或目的地的页面，请双击连接器跳转到相应页面。</p> |
| <p>OR, AND, XOR 门</p>  | <p>OR（或），AND（与）或 XOR（非）门原语（端口数目可不相同）。输入或输出端口上的小圆圈（气泡符）表示端口反转。</p> |
| <p>MULTIPLEXER</p>  | <p>具有段选择子的多路复用器原语在端口 0 和端口 1 之间选择。具有两个以上输入的多路复用器显示为操作符。</p> |
| <p>BUFFER</p>  | <p>缓冲器原语。本图显示三态缓冲器，和一个已反转的输出使能端口。其他不具备使能端口的缓冲器包括，LCELL，SOFT，CARRY，和 GLOBAL。NOT 门和 EXP 扩展器缓冲器使用该符号，不具备使能的端口和已反转输出端口。</p> |
| 继续... | |

| 符号 | 说明 |
|--------------|---|
| <p>LATCH</p> | <p>锁存器/DFF（数据触发器原语）。DFF 的端口与锁存器和时钟触发器相同。其他触发器原语与之类似：</p> <ul style="list-style-type: none"> • DFFEAS（具有使能和异步加载的数据触发器）原语和其他 ALOAD 异步加载和 ADATA 数据信号 • DFFEAS（具有使能和同步以及异步加载的数据触发器），其还有 ASDATA 作为下游总线数据端口 |
| <p>原子原语</p> | <p>原子原语。该符号显示原子名称，端口名称和原子类型。蓝色阴影表示可查看内部详细信息的原子原语。</p> |
| <p>其他原语</p> | <p>任何未归入前述类别的原语。该原语是不可扩展为更低层次的低级别节点。此符号显示端口名称，原语或操作符类型及其名称。</p> |
| <p>实例</p> | <p>设计中无对应原语或操作符的实例（用户定义的层次块）。该符号显示端口名称和实例名称。</p> |
| <p>加密实例</p> | <p>设计中用户定义的加密实例。该符号显示实例名称。无法打开较低级别层次的原理图，因为源设计已加密。</p> |
| <p>状态机实例</p> | <p>设计中有限状态机实例。</p> |
| 继续... | |

| 符号 | 说明 |
|----|---|
| | |
| | 同步存储器实例具有已寄存输入和可选已寄存输出。该符号显示器件系列和存储器块类型。该图示显示为 Stratix M-RAM 块中真正的双端口存储器块。 |
| | 常量信号值以灰色突出显示，默认情况下整个原理图中以十六进制格式显示。 |

表 6. 仅 State Machine Viewer 中可用的符号

下表罗列并说明了仅 State Machine Viewer 中开放可用的符号。

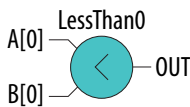
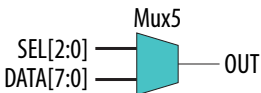
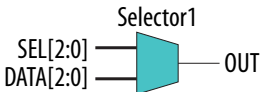
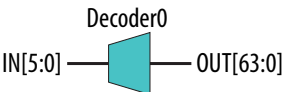
| 符号 | 说明 |
|----|--|
| | 该节点代表有限状态集中的状态。状态节点间的状态转变由弧线表示。双圆边框表示状态连接到状态机外部的逻辑，单个圆边框表示状态节点不馈送外部逻辑。 |

表 7. RTL Viewer Schematic View 中的操作符号

下表罗列并说明 RTL Viewer 原理图中其他更高级别操作符号。

| 符号 | 说明 |
|----|-----------------------------------|
| | 加法器操作符: $OUT = A + B$ |
| | 乘法器操作符: $OUT = A \times B$ |
| | 除法器操作符: $OUT = A / B$ |
| | 等式 |
| | 左移操作符: $OUT = (A \ll COUNT)$ |
| | 右移操作符: $OUT = (A \gg COUNT)$ |
| | 右移操作符: $OUT = (A \% B)$ |
| | 小于比较器: $OUT = (A < B : A > B)$ |

继续...

| 符号 | 说明 |
|---|--|
|  | |
|  | <p>多路复用器:</p> $OUT = DATA [SEL]$ <p>数据范围大小为 $2^{sel\ range\ size}$</p> |
|  | <p>选择器:</p> <p>多路复用器带有一个热选输入和两个以上输入信号</p> |
|  | <p>二进制数解码器:</p> $OUT = (binary_number(IN) == x)$ <p>for $x = 0$ 到 $x = 2^{(n+1)} - 1$</p> |

相关链接

- [原理图分页](#) (第 35 页)
- [关注原理图页面中的网络](#) (第 36 页)

2.7.3. 在 Schematic View 中选择项目

要在原理图视图中选择一个项目，请确保 **Netlist Viewer** 功能条中的 **Selection Tool** 已使能。在原理图中单击一个项目会以红色突出显示。

使用鼠标选择时，可按 **Shift** 键选择多个项目。

Netlist Navigator 窗格中会自动选择在原理图中已选择的项目。如要求显示所选择项，则文件夹自动展开；但取消选择条目后，文件夹不会自动收起。

在原理图中选择层次框，节点或端口时，**Schematic View** 以红色突出显示该项目，但不突出显示连接网络。在原理图中选择网络（线缆或总线）时，**Schematic View** 以红色突出显示已连接网络。

一旦选择某项目后，就可根据右键选择时出现的快捷菜单上的内容对其执行不同操作。

相关链接

[Netlist Navigator 窗格](#) (第 24 页)

2.7.4. Schematic View 中的快捷菜单命令

右键单击原理图中所选实例或原语时，**Netlist Viewer** 显示快捷菜单。

如果已选项目是一个节点，则会看到如下选项：

- 单击 **Expand to Upper Hierarchy** 显示目标节点的父级层次。
- 单击 **Copy ToolTip** 复制所选项目的名称到剪贴板。此命令不适用于网络。
- 单击 **Hide Selection** 从原理图中删除所选项目。该命令不会将项目从设计中删除，仅会在当前视图中将其屏蔽。
- 单击 **Filtering** 显示子菜单和带有过滤选择的选项。

2.7.5. 原理图中进行过滤

通过过滤可将节点和网络从网表中过滤，从而仅查看您感兴趣的逻辑单元。

You can filter a netlist by selecting hierarchy boxes, nodes, ports of a node, or states in a state machine that are part of the path you want to see. The following filter commands are available:

- **Sources**—显示选择的来源。
- **Destinations**—显示选择的目的地。
- **Sources & Destinations**—显示选择的来源和目的地。
- **Selected Nodes**—仅显示已选择节点。
- **Between Selected Nodes**—显示介于所选节点间路径中的节点和连接。
- **Bus Index**—显示输入或输出总线端口一个或多个指数的源和目标。
- **Filtering Options**—显示 **Filtering Options** 对话框：
 - **Stop filtering at register**—打开该选项指示 Netlist Viewer 过滤到最近的寄存器边界。
 - **Filter across hierarchies**—打开该选项指示 Netlist Viewer 跨层次过滤。
 - **Maximum number of hierarchy levels**—设置原理图最多可显示的层次级别。

要过滤网表，选择一个层次框，节点，端口，网络或状态节点，在窗口中单击右键，指向 **Filter** 并单击相应的过滤命令。Netlist Viewer 生成一个新页面显示网表经过过滤后剩下的内容。

When filtering in a state diagram in the State Machine Viewer, sources and destinations refer to the previous and next transition states or paths between transition states in the state diagram. The transition table and encoding table also reflect the filtering.

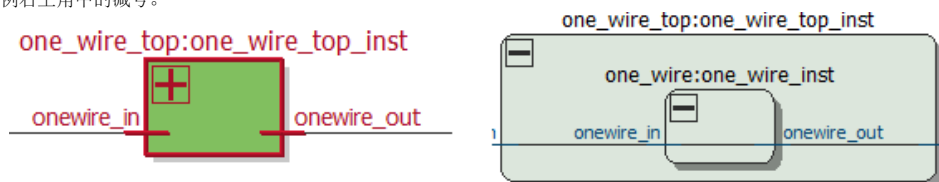
2.7.6. 在 Schematic View 中查看节点内容

在 RTL Viewer 和 Technology Map Viewer 中，可查看节点的内容以了解实际实现详情。

可查看 LUT，寄存器和逻辑门控。还可在 RTL Viewer 或 Technology Map Viewer 中查看具体器件中 RAM 和 DSP 块的实现。在 Technology Map Viewer 中，可查看原语的内容以了解其实际实现详情。

图 6. Wrapping 和 Unwrapping 对象

如果可打开 (unwrap) 实例内容，则原理图中目标对象的右上角出现一个加号。要收起内容（以及恢复压缩模式），单击已打开实例右上角中的减号。



注意:

原理图视图中，原子实例中的内部信息不可作为单个节点进行选择。任何内部细节上的任何鼠标操作都视作在原子实例上的鼠标操作。

图 7. 与层次外连接的节点

某些情况下，所选实例与原理图中层次可视级别以外的某些内容相连接。该情况下，网络显示为虚线。双击虚线展开视图显示连接目标。

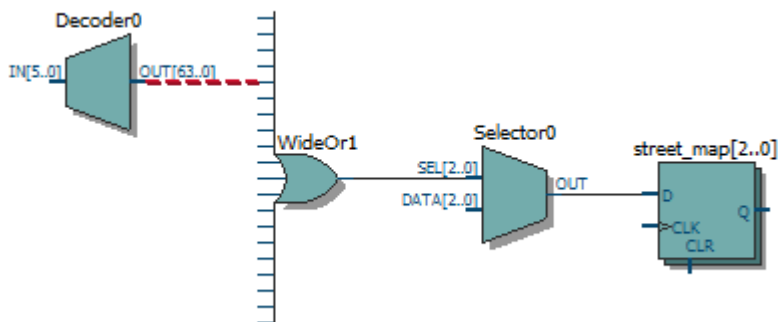


图 8. 显示层次中的网络

在网络连接到层次外实例的情况下。可选择网络并展开节点查看目标端口。

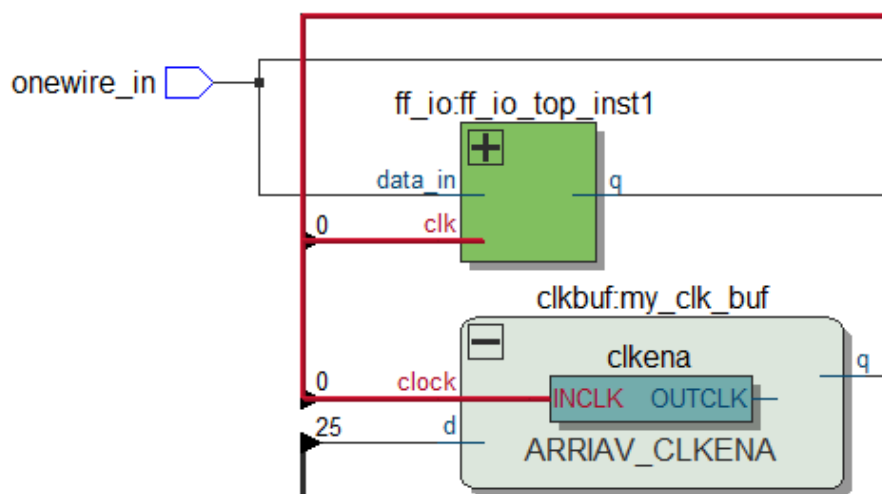
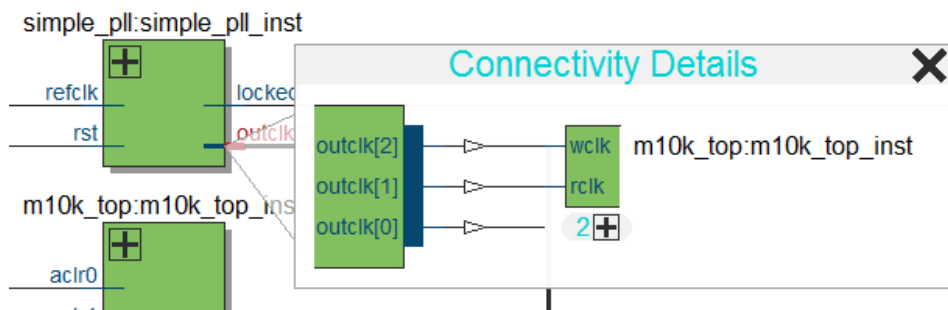


图 9. 显示连接性详细信息

可选择一个总线端口或总线管脚并单击工程相应菜单中的 **Connectivity Details**。



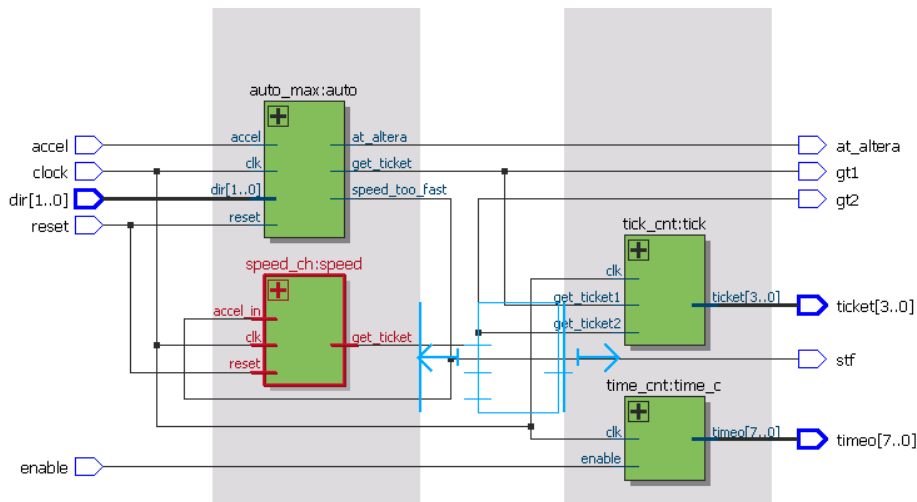
可双击 **Connectivity Details** 窗口中的目标快速导航到对应内容。如果出现加号，则可在视图中进一步展开目标对象。该功能有助于在复杂网表中追踪信号。

2.7.7. 在 Schematic View 中移动节点

通过将项目拖动到目的地重新排列原理图。

将节点从网表中的一个区域移动到另一区域，选择节点并按住 **Shift** 键。层次中的合法布局以阴影区域显示。单击以拖动所选择节点。

图 10. 移动节点时的合法布局



要将原理图恢复到其默认布局，右键单击并点选 **Refresh**。

2.7.8. 在 Technology Map Viewer 中查看 LUT 表达

右键点击所选 LUT 并点击 **Properties** 可查看 LUT 的不同表示。

可在 **Properties** 对话框的以下三个选项卡中查看 LUT 表示：

- **Schematic** 选项卡—表示 LUT 的等效门控。
- **Truth Table** 选项卡—表示真值表。

相关链接

[Properties 窗格](#) (第 24 页)

2.7.9. 缩放控制

使用工具栏中的 **Zoom Tool**，或鼠标手势在 **View** 菜单上控制原理图放大率。

默认情况下，**Netlist Viewer** 显示最符合窗口大小的页面。如果原理图页面过大，则该原理图以最小缩放水平显示，且视图以第一个节点为中心。单击 **Zoom In** 以较大尺寸查看图像；单击 **Zoom Out** 以较小尺寸查看图像（未显示完整图像时）。**Zoom** 命令允许您指定放大百分比（100%被视为原理图符号的正常大小）。

可使用 **Netlist Viewer** 工具栏上的 **Zoom Tool** 控制原理图中的放大率。在工具栏中选择 **Zoom Tool**，单击原理图会放大并以您点击的位置为中心放置视图。右键单击原理图会缩小并以点击处为中心放置视图。选择 **Zoom Tool** 后，还可通过鼠标光标选择矩形框区放大特定部分。放大原理图以显示所选区域。

在原理图视图中，还可使用鼠标手势放大特定部分：

- **zoom in**—从左上角开始围绕区域拖动一个框，然后向右下方拖动放大该区域。
- **Zoom -0.5**—从左下方向右上方沿直线拖动缩小 0.5 级放大率。
- **zoom 0.5**—从右下方往左上方直线拖动放大 0.5 级放大率。
- **zoom fit**—从右上方向左下方直线拖动配合页面调整原理图视图。

相关链接

[原理图中进行过滤](#) (第 32 页)

2.7.10. Bird's Eye View 导航

要打开 **Bird's Eye View**，请在 **View** 菜单，单击 **Bird's Eye View**，或在工具栏中单击 **Bird's Eye View** 图标。

查看整个原理图在调试和追踪大型网表时非常有用。Intel Quartus Prime 软件支持使用鸟瞰图功能快速导航原理图中的指定部分，该功能适用于 **RTL Viewer** 和 **Technology Map Viewer**。

Bird's Eye View 显示当前感兴趣的区域：

- 点击并拖动光标在所选区域周围形成一个矩形框来选择该区域。
- 点击并拖动矩形框在原理图中移动。
- 调整矩形框大小在原理图中进行缩放。

2.7.11. 原理图分页

对于较大设计的层次，**RTL Viewer** 和 **Technology Map Viewer** 可将您的网表分成多个原理图页面。

层次级别被分成多个页面后，原理图窗口的标题栏标示当前显示页页码以及该层次级别的总页数。原理图视图显示为 **Page<当前页码> of <总页数>**。

相关链接

[Netlist Viewer 用户接口](#) (第 21 页)

2.7.12. 关注原理图页面中的网络

输入和输出连接器符号表示相同层次页面中连接的节点。双击连接器通过网络追踪到层次结构的下一层。

注意: 在双击并关注连接器端口后，**Netlist Viewer** 打开一个新页面，而该页面使用的缩放系数与上一页相同，且以指定源或目标网络为中心呈现。要追踪层次中新页面的指定网络，Intel 建议首先选择必要网络，将其以红色突出显示，然后才双击在页面中导航。

相关链接

原理图符号 (第 26 页)

2.8. State Machine Viewer

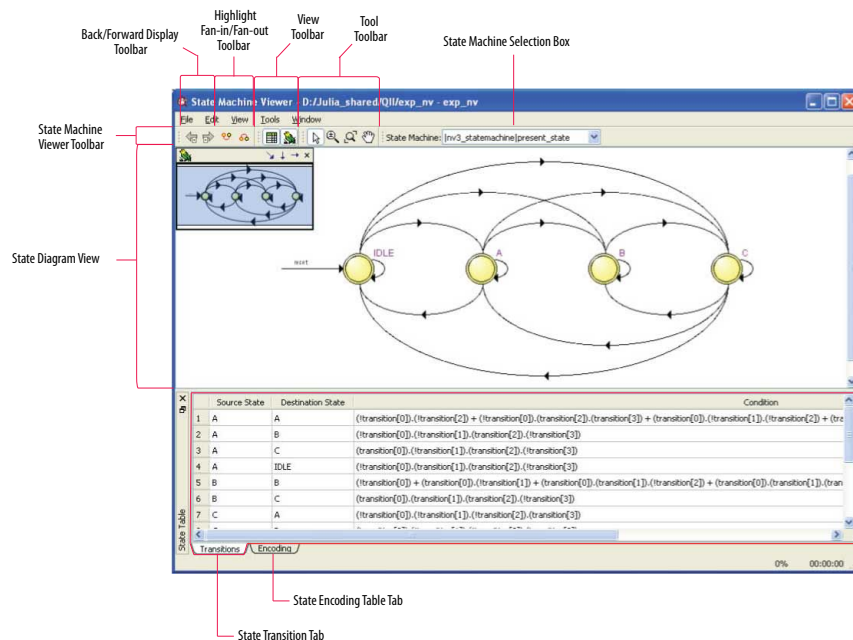
The State Machine Viewer displays a graphical representation of the state machines in your design.

You can open the State Machine Viewer in any of the following ways:

- On the Tools menu, point to **Netlist Viewers** and click **State Machine Viewer**.
- Double-click a state machine instance in the RTL Viewer

图 11. The State Machine Viewer

The following figure shows an example of the State Machine Viewer for a simple state machine and lists the components of the viewer.



2.8.1. State Diagram View

The state diagram view appears at the top of the State Machine Viewer. It contains a diagram of the states and state transitions.

The nodes that represent each state are arranged horizontally in the state diagram view with the initial state (the state node that receives the reset signal) in the left-most position. Nodes that connect to logic outside of the state machine instance are represented by a double circle. The state transition is represented by an arc with an arrow pointing in the direction of the transition.

When you select a node in the state diagram view, and turn on the **Highlight Fan-in** or **Highlight Fan-out** command from the View menu or the State Machine Viewer toolbar, the respective fan-in or fan-out transitions from the node are highlighted in red.

注意:

An encrypted block with a state machine displays encoding information in the state encoding table, but does not display a state transition diagram or table.

2.8.2. State Transition Table

The state transition table on the **Transitions** tab at the bottom of the State Machine Viewer displays the condition equation for each state transition.

Each row in the table represents a transition (each arc in the state diagram view). The table has the following columns:

- **Source State**—the name of the source state for the transition
- **Destination State**—the name of the destination state for the transition
- **Condition**—the condition equation that causes the transition from source state to destination state

To see all of the transitions to and from each state name, click the appropriate column heading to sort on that column.

The text in each column is left-aligned by default; to change the alignment and to make it easier to see the relevant part of the text, right-click the column and click **Align Right**. To revert to left alignment, click **Align Left**.

Click in any cell in the table to select it. To select all cells, right-click in the cell and click **Select All**; or, on the Edit menu, click **Select All**. To copy selected cells to the clipboard, right-click the cells and click **Copy Table**; or, on the Edit menu, point to **Copy** and click **Copy Table**. You can paste the table into any text editor as tab-separated columns.

2.8.3. State Encoding Table

The state encoding table on the **Encoding** tab at the bottom of the State Machine Viewer displays encoding information for each state transition.

To view state encoding information in the State Machine Viewer, you must synthesize your design with the **Start Analysis & Synthesis** command. If you have only elaborated your design with the **Start Analysis & Elaboration** command, the encoding information is not displayed.

2.8.3.1. Select Items in the State Machine Viewer

You can select and highlight each state node and transition in the State Machine Viewer. To select a state transition, click the arc that represents the transition.

When you select a node or transition arc in the state diagram view, the matching state node or equation conditions in the state transition table are highlighted; conversely, when you select a state node or equation condition in the state transition table, the corresponding state node or transition arc is highlighted in the state diagram view.

2.8.4. Switch Between State Machines

A design may contain multiple state machines. To choose which state machine to view, use the **State Machine** selection box located at the top of the State Machine Viewer. Click in the drop-down box and select the necessary state machine.

2.9. 交叉探查 Source Design File 和其他 Intel Quartus Prime Windows

The RTL Viewer, Technology Map Viewer, and State Machine Viewer allow you to cross-probe to the source design file and to various other windows in the Intel Quartus Prime software.

可在 **Netlist Viewer** 中选择一个或多个感兴趣的层次框，节点，状态模式或状态转变弧线，并在其他可用的 **Intel Quartus Prime** 软件窗口中找到相应项。随后可查看并在对应编辑器或平面布局规划中进行更改或约束。

要在其他窗口中找到 **Netlist Viewer** 中的对应项，右键单击原理图或状态图中感兴趣的项，指向 **Locate**，并点击相应命令。可用命令如下：

- **Locate in Assignment Editor**
- **Locate in Pin Planner**
- **Locate in Chip Planner**
- **Locate in Resource Property Editor**
- **Locate in Technology Map Viewer**
- **Locate in RTL Viewer**
- **Locate in Design File**

可用于查找项目的选项取决于节点类型，以及布局布线后其是否存在。如果菜单中某个命令已使能，则其可用于所选节点。**Locate in Assignment Editor** 命令可用于所有节点，但如果将其应运于综合后便不存在的节点，则布局布线期间的约束可能会被忽略。

Netlist Viewer 自动打开相应编辑器或布局规划窗口，并在新窗口中突出显示所选节点和网络。可在 **Window** 菜单选择切换回 **Netlist Viewer**，或关闭，最小化和移动新窗口。

2.10. 从其他 Intel Quartus Prime 窗口交叉探查 Netlist Viewer

在 **Intel Quartus Prime** 软件中可从其他窗口交叉查探 **RTL Viewer** 和 **Technology Map Viewer**。可在其他窗口中选择 1 个或多个节点和网络并在其中一个 **Netlist Viewer** 中进行查看。

You can locate nodes between the RTL Viewer, State Machine Viewer, and Technology Map Viewer, and you can locate nodes in the RTL Viewer and Technology Map Viewer from the following Intel Quartus Prime software windows:

- Project Navigator
- Timing Closure Floorplan
- Chip Planner
- Resource Property Editor
- Node Finder
- Assignment Editor
- Messages Window
- Compilation Report
- Timing Analyzer (仅支持 Technology Map Viewer)

要从另一 Intel Quartus Prime 软件窗口找到 Netlist Viewer 中的组件，请在相应窗口选择选择一个或多个节点；例如，在 Project Navigator 的 **Hierarchy** 选项卡上，从 **Entity** 清单上选择一个条目，或在 Timing Closure Floorplan 中选择节点，或在 Assignment Editor 中的 **From** 或 **To** 列选择节点名称。然后，右键单击所选对象，指向 **Locate**，并点选 **Locate in RTL Viewer** 或 **Locate in Technology Map Viewer**。点选该命令后，Netlist Viewer 打开，或者如果 Netlist Viewer 已打开则被置顶。

注意： 编译后首次打开窗口时，会先运行预处理器阶段然后才打开 Netlist Viewer。

Netlist Viewer 显示已选节点和节点间的连接（如果适用）。该显示与右键单击目标对象，然后点选 **Filter > Selected Nodes** 使用 **Filter across hierarchy** 时相似。如果无法在 Netlist Viewer 中找到节点，则消息框显示此消息：**Can't find requested location**（无法找到所请求位置）。

2.11. 查看时序路径

完成完整设计编译后（包括时序分析阶段），可从时序报告看到时序路径交叉探查的直观表达。关于生成时序报告的详细信息，请参阅 *Intel Quartus Prime Standard Edition User Guide: Timing Analyzer*。

找到从 Timing Analyzer 到 Technology Map Viewer 的时序路径后，与每个节点关联的互连和单元延迟将在原理图视图中置顶显示。所选时序路径的总时序裕量（slack）在原理图 Page Title 部分显示。

1. 要打开 Contents 中 **Compilation Report Table** 的报告，请点击 **Timing Analyzer GUI > Report Timing**，并双击 timing corner。
2. 要打开 **Timing Analyzer** 中的报告，请从 **Report** 窗格打开 **Report Timing** 文件，并双击 timing corner。
3. 在 **Summary of Paths** 选项卡中，右键单击表格中的行，并选择 **Locate Path > Locate in Technology Map Viewer**。Technology Map Viewer 中，原理图页面显示时序路径上的节点以及总延迟的总结。

相关链接

[Report Timing \(Dialog Box\)](#)

Intel Quartus Prime Standard Edition User Guide: Timing Analyzer

2.12. 优化设计网表修订历史

本章修订历史如下：

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|---------------|------------------------|---|
| 2018.09.24 | 18.1.0 | <ul style="list-style-type: none"> 首次发布 Intel Quartus Prime Standard Edition (标准版) 用户指南。 为 <i>Viewing a Timing Path</i> 添加了链接。 |
| 2016.05.03 | 16.0.0 | 删除了 Schematic Viewer 主题。 |
| 2015.11.02 | 15.1.0 | 添加了关于以下新功能和功能更新的信息： <ul style="list-style-type: none"> Nets visible across hierarchies (跨层次的网络可见性) Connection Details (连接详情) Display Settings (显示设置) Hand Tool (手形工具) Area Selection Tool (区域选择工具) New default behavior for Show/Hide Instance Pins (目前为默认关闭) (显示/隐藏实例管脚的新默认行为) |
| 2014.06.30 | 14.0.0 | 在 One Page 和 Show/Hide Instance Pins 命令中添加了 Show Netlist。 |
| November 2013 | 13.1.0 | 删除了 HardCopy 器件信息。 重组并更换为新模板。 添加关于新 Netlist viewer 的支持。 |
| November 2012 | 12.1.0 | 添加了支持 Global Net Routing 功能的部分。 |
| June 2012 | 12.0.0 | 删除了反馈问卷链接。 |
| November 2011 | 10.0.2 | 模板更新。 |
| December 2010 | 10.0.1 | 更换为新文档模板。 |
| July 2010 | 10.0.0 | <ul style="list-style-type: none"> 更新了截图。 更新了 Intel Quartus Prime 软件 10.0 的章节，包括主要用户界面更改。 |
| November 2009 | 9.1.0 | <ul style="list-style-type: none"> 更新器件 少量文本编辑 |
| March 2009 | 9.0.0 | <ul style="list-style-type: none"> 第 13 章原为版本 8.1.0 中的第 12 章 更新了图 13-2、图 13-3、图 13-4、图 13-14 和图 13-30。 添加了“使能或禁用自动层次列表” (第 13 - 15 页) 更新了“查找命令” (第 13 - 44 页) |
| November 2008 | 8.1.0 | 页面尺寸更改为 8.5" × 11" |
| May 2008 | 8.0.0 | <ul style="list-style-type: none"> 添加了 Arria GX 支持 更新了操作符符号 更新了径向菜单功能的信息。 更新了缩放功能 更新了从原理图到 Signal Tap Analyzer 的交叉探查信息。 更新了常量信号的信息。 在支持的图像文件格式清单中添加了 .png 和 .gif。 更新了多个图示和表格。 |
| 继续... | | |

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|------|------------------------|--|
| | | <ul style="list-style-type: none"> • 添加了新的部分：“Enabling and Disabling the Radial Menu”（使能和禁用 Radial Menu），“Changing the Time Interval”（更改时间间隔），“Changing the Constant Signal Value Formatting”（更改常量信号值格式），“Logic Clouds in the RTL Viewer”（RTL Viewer 中的逻辑云），“Logic Clouds in the Technology Map Viewer”（Technology Map Viewer 中的逻辑云），“Manually Group and Ungroup Logic Clouds”（手动组合和取消组合逻辑云），“Customizing the Shortcut Commands”（自定义快捷命令） • 重命名多个部分 • 删除了“Customizing the Radial Menu”（自定义 Radial Menu）部分 • 删除了“Grouping Combinational Logic into Logic Clouds”（将组合逻辑分组为逻辑云） • 基于 Intel Quartus Prime 软件 8.0 更新文档内容 |

相关链接

文档存档

欲了解 *Intel Quartus Prime* 手册以前的版本，请搜索文档存档。

3. 时序收敛与优化

本章介绍在设计 Intel 器件时提高时序性能的技术。具体应用技术因设计而已。应用各个技术后并不一定改善设计结果。

Intel Quartus Prime 软件中的默认设置和选项提供编译时间，资源利用率和时序性能之间的最佳平衡。可调整这些设置确定是否其他设置能为设计取得更佳结果。

3.1. 优化 Multi Corner 时序

工艺变化和操作条件导致路径延迟，但其显著小于 **slow corner timing** 模型中的路径延迟。因而，设计中的这些路径上会出现保持时间违规，并且极少情况下还会出现额外建立时间违规。

此外，针对具有较小工艺几何结构的新器件系列的设计在最高操作温度下并不一定出现最慢电路性能。使电路最慢运行的温度取决于所选器件，设计和编译结果。Intel Quartus Prime 软件为新的器件系列提供三种不同的 **timing corner** 以管理这种新的依存关系—Slow 85°C corner, Slow 0°C corner 和 Fast 0°C corner。对于其他器件系列，有 2 个 **timing corner** 可用—Fast 0°C 和 Slow 85°C corner。

Optimize multi-corner timing 选项指示 Fitter 满足所有工艺极限（process corner）和操作条件下的时序要求。最终的设计实现稳健应对工艺，温度和电压差异。该选项默认开启，并增加大约 10% 的编译时间。

该选项关闭后，Fitter 仅鉴于 **slow-corner** 时序模式（最慢的制造工艺器件用于给定速度等级并在低电压条件操作）的 **slow-corner** 延迟进行设计优化。

3.2. 关键路径

关键路径是设计中具有负时序余量的时序路径。关键路径涉及器件 I/O 到内部寄存器，寄存器到寄存器，或从寄存器到器件 I/O。

路径时序余量决定其关键度；该时序余量呈现在时序分析报告中，而时序报告可通过 **Timing Analyzer** 生成。

针对时序收敛的设计分析是高度复杂设计中最佳性能的基本要求。Chip Planner 的分析能力有助于复杂设计中的时序收敛。

相关链接

- [减短关键路径延迟](#) (第 10 页)
- [使用 Timing Analyzer 显示 Path Report](#) (第 56 页)

3.2.1. 查看关键路径

在 **Chip Planner** 中查看关键路径可了解特定路径失败的原因。可查看布局中的任何修改是否能减少负时序余量。要显示布局规划（**floorplan**）中的路径，请通过 **Timing Analyzer** 执行时序分析并显示结果。

3.3. 时序收敛的设计评估

设计中出现时序失败时，可遵循本节中的指导。该指导演示了如何评估设计的编译结果以及如何应对问题。虽然该指导中未涵盖重组 **RTL** 提高设计速度的具体示范，但本分析技术可帮助评估对 **RTL** 的更改并利于时序收敛。

3.3.1. 查看编译结果

3.3.1.1. 查看消息

编译设计后，请查看编译报告各部分中的消息。

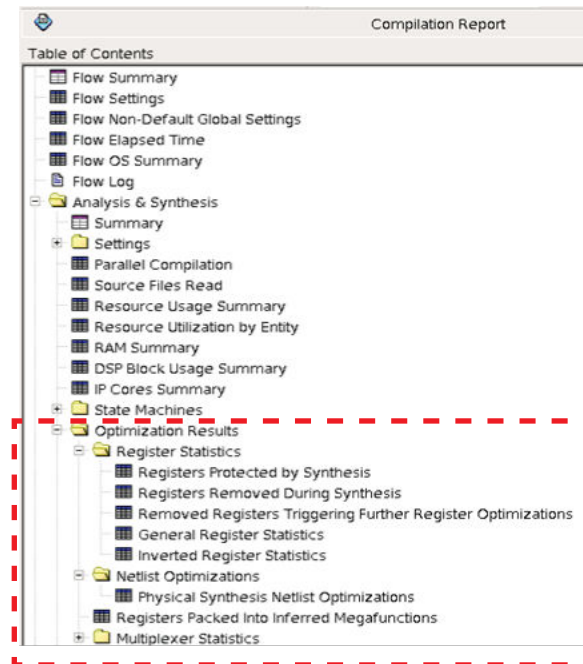
大多数时序失败的设计皆因其他问题而起，编译期间 **Fitter** 将其报告为警告消息。确定导致警告消息的原因，以及是否需要修复或忽略警告。

查看警告消息后，请查看信息消息。标注意外情况，例如，未连接的端口，忽略的约束，丢失的文件和软件做出的假设和优化。

3.3.1.2. Evaluate Physical Synthesis Results

If you enable physical synthesis options, the Compiler can duplicate and retime registers, and modify combinatorial logic during synthesis. After compilation, review the Optimization Results reports in the Analysis & Synthesis section. The reports list the optimizations performed by the physical synthesis optimizations, such as register duplication, retiming, and removal. These reports can be found in the Compilation Report panel.

图 12. Optimization Results Reports



When you enable physical synthesis, the compilation messages include a information about the physical synthesis algorithm performance improvement. The reported improvement is the sum of the largest improvement in each timing-critical clock domain. Although typically similar, the values for the slack improvements vary per compilation due to the random starting point of compilation algorithms.

3.3.1.3. 评估 Fitter 网表优化

Fitter 还可对设计网表执行优化。主要变化包括寄存器封装，复制或删除逻辑单元，反转信号或以常规方式修改节点，例如将输入从一个逻辑单元移动到另一个。可在 Fitter 部分的 Netlist Optimizations 结果中查找并查看这些报告。

3.3.1.4. 评估优化结果

在检查完已完成优化和性能提高情况后，请评估为获得额外性能所花费的运行时间。为减少编译时间，请在几次编译后查看物理综合和网表优化，并编辑 RTL 以反映物理综合执行的更改。如果一组特定的寄存器持续重新定时，则可编辑 RTL 以相同方式重新定时寄存器。如果是为匹配物理综合算法而进行更改，则可关闭物理综合选项以节省编译时间并获得同等类型的性能改善。

3.3.1.5. 评估资源使用情况

要评估设计中各种资源的使用情况，包括全局和非全局信号使用，布线使用情况和集群难度。

3.3.1.5.1. 全局和非全局使用情况

对于包含多个时钟的设计，请评估全局和非全局信号以确定是否有效使用全局资源，如果未有效使用，请考虑更改。可在 **Compilation Report** 的 Fitter 下 Resource 部分找到这些报告。

该图示显示未有效使用全局时钟的示例。突出显示的行具有来自全局时钟的单个扇出。

图 13. 全局时钟的低效使用

| Global & Other Fast Signals | | | |
|-----------------------------|---------|----------------------|------------------|
| Location | Fan-Out | Global Resource Used | Global Line Name |
| FRACTIONALPLL_X98_Y2_N0 | 1 | Global Clock | -- |
| PLLOUTPUTCOUNTER_X98_Y2_N1 | 29044 | Global Clock | GCLK7 |
| PLLOUTPUTCOUNTER_X98_Y13_N1 | 253103 | Global Clock | GCLK6 |
| FF_X185_Y66_N13 | 280349 | Global Clock | GCLK8 |
| PIN_AE17 | 4887 | Global Clock | GCLK4 |
| FRACTIONALPLL_X98_Y11_N0 | 1 | Global Clock | -- |
| PLLOUTPUTCOUNTER_X98_Y3_N1 | 1 | Global Clock | GCLK5 |
| PLLOUTPUTCOUNTER_X98_Y1_N1 | 1691 | Regional Clock | RCLK29 |
| PLLOUTPUTCOUNTER_X98_Y8_N1 | 302 | Regional Clock | RCLK23 |
| PLLOUTPUTCOUNTER_X98_Y11_N1 | 141 | Regional Clock | RCLK25 |
| PLLOUTPUTCOUNTER_X98_Y10_N1 | 17 | Regional Clock | RCLK22 |

如果将这些资源约束给 Regional Clock（区域时钟），则 Global Clock（全局时钟）可用于另一信号。可忽略 **Global Line Name** 栏中的空值信号，因为该信号用于专属布线，并不是时钟缓冲器。

Non-Global High Fan-Out Signal 报告列出未在全局信号上路由的最高扇出节点。

Reset 和 enable 信号显示于列表顶部。

如果设计中存在路由拥塞，且拥塞区域中有高扇出非全局节点，则请考虑使用全局或区域信号扇出节点，或复制高扇出寄存器以使每个复制部分都仅有较少扇出。

使用 Chip Planner 找到高扇出节点从而报告路由拥塞，并确定备选方案是否可行。

3.3.1.5.2. 布线使用

在 **Fitter Resource Usage Summary** 报告中查看布线使用情况。

图 14. Fitter Resource Usage Summary Report

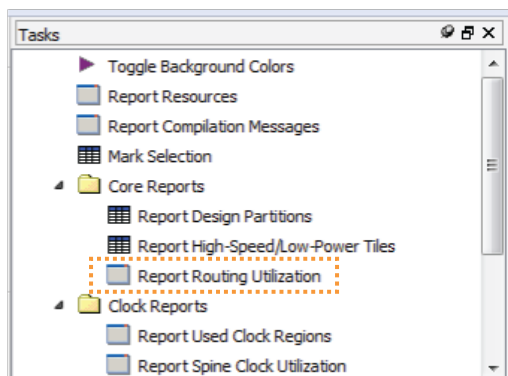
| Table of Contents | | | Fitter Resource Usage Summary | |
|-------------------|--|--|--|------------------|
| | | | Resource | Usage |
| | | | 43 10G TX PCSs | 12 / 36 (33 %) |
| | | | 44 HSSI FMA TX Serializers | 12 / 36 (33 %) |
| | | | 45 CHANNEL PLLs | 12 / 36 (33 %) |
| | | | 46 Impedance control blocks | 1 / 4 (25 %) |
| | | | 47 Average interconnect usage (total/HV) | 95% / 95% / 95% |
| | | | 48 Peak interconnect usage (total/HV) | 88% / 88% / 90% |
| | | | 49 | |

Average interconnect usage 报告器件上可用互连中已使用的互连平均量。**Peak interconnect usage** 报告最拥塞区域中所使用的最大互连量。

平均值低于 50% 的设计通常都无布线问题。平均值介于 50-65% 之间的设计可能出现布线困难。而平均值超过 65% 的设计通常难以满足时序要求，除非 RTL 支持高利用率芯片。峰值达到或超过 90% 的情况下很可能在时序收敛时出现问题；峰值达到 100% 表示器件该区域内的所有布线已被使用，因此时序性能降低的可能性很高。

该图显示为 **Report Routing Utilization** 报告。

图 15. Report Routing Utilization Report

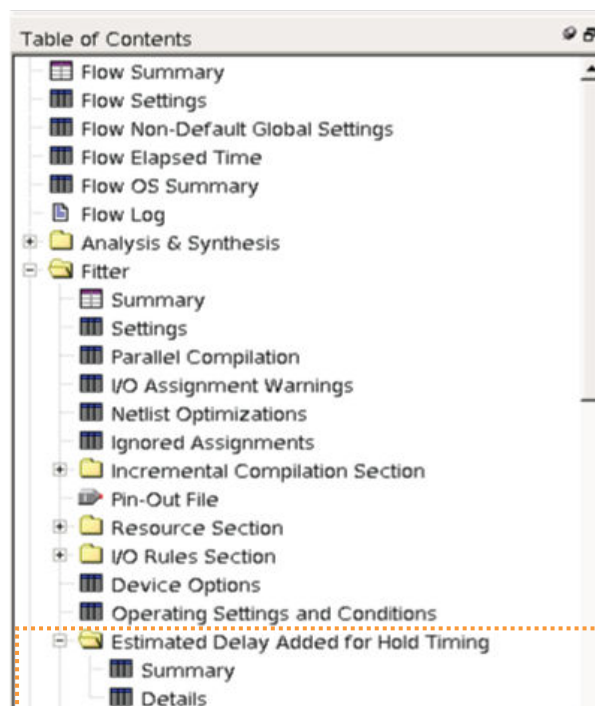


3.3.1.5.3. 添加线缆以“保持”

布线期间 Fitter 可能在寄存器路径之间添加线缆以增加延迟，从而满足保持时间要求。Fitter 在 **Estimated Delay Added for Hold Timing** 中报告已添加的布线延迟的量。过多添加线缆可能出现约束错误。导致该错误的原因通常是多速率时钟之间，以及不同时钟网络之间不正确的多周期传输。

查看 **Estimated Delay Added for Hold Timing** 报告中指定寄存器路径以确定 Fitter 是否添加过多线缆以满足保持时序要求。

图 16. Hold Timing Report 的预估添加延迟

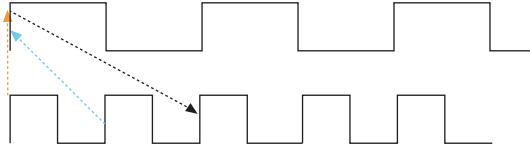


当数据传输从 1x 到 2x 时钟时，导致路由器为满足保持要求而添加线缆的错误约束示例。假设计意图是允许每传输两个周期。而通过添加多周期建立约束，数据可在两个目标时钟周期内的任何时间达到，如范例中所示：

```
set_multicycle_path -from 1x -to 2x -setup -end 2
```

时序要求放松一个 2x 时钟周期，如图示波形中的黑色线条所示。

图 17. 时序要求的松弛波形



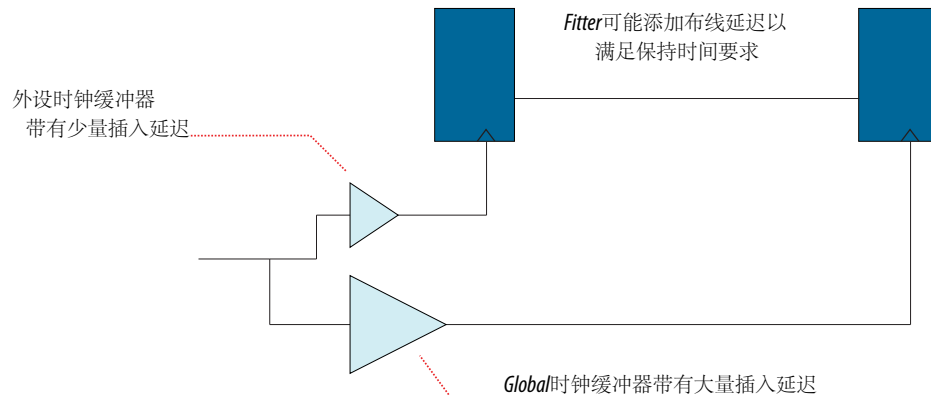
默认保持要求（如蓝色虚线所示），可强制路由器添加线缆以保证数据延迟一个周期。要更正保持要求，通过保持选项添加一个多周期约束。

```
set_multicycle_path -from 1x -to 2x -setup -end 2  
set_multicycle_path -from 1x -to 2x -hold -end 1
```

上图中橙色虚线代表保持关系，且无需额外线缆来延迟数据。

当数据在同一时钟域中传输时，布线程序也可添加线缆以保持时序要求，但各时钟分支之间使用不同缓冲。时钟网络类型间的传送常常发生在外设和内核之间。下图显示数据正进入器件，外设时钟驱动源寄存器，全局时钟驱动目标寄存器。全局时钟缓冲器的插入延迟大于外设时钟缓冲器。所以到目标寄存器的时钟延迟远大于到源寄存器的时钟延迟，因此数据路径上需要额外的延迟以确保满足其保持要求。

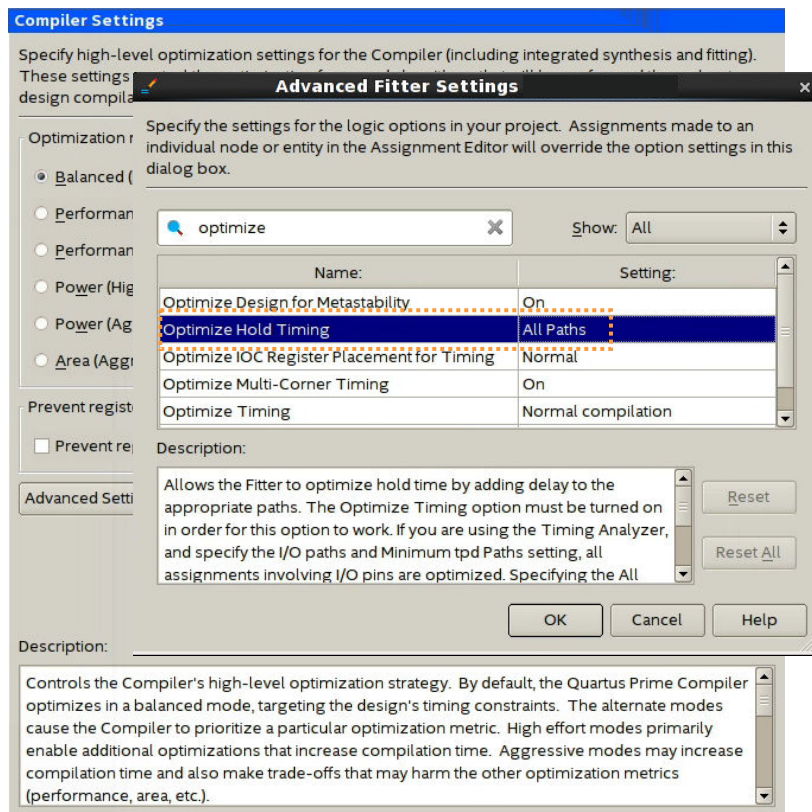
图 18. 时钟延迟



要确认路径中各种时钟网络类型，请在 Timing Analyzer 中查看路径，并沿源时钟路径和目标时钟路径检查节点。此外，检查源时钟频率和目标时钟频率以查看两者是否相同，或成倍数，以及路径上是否存在多周期异常。最后，请确保所有有意的跨域路径错误都有相关联的错误路径异常。

如果您怀疑已添加了修复实际保持问题的路径，则可禁用 **Optimize hold timing** 高级 Fitter 设置 (**Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) > Optimize hold timing**)。在禁用 **Optimize hold timing** 的条件下重新编译设计，然后重新运行分析以确认并更正不符合保持时间要求的路径。

图 19. Optimize Hold Timing Option



注意: 仅在调试设计时才禁用 **Optimize hold timing** 选项。请确保常规编译期间启用该选项（默认状态）。布线期间针对保持时间而添加的线缆是时序优化的常规部分，不一定是存在问题。

3.3.1.6. 评估其他报告并进行相应调整

3.3.1.6.1. 难以封装的设计

Fitter Resource Section 中, 在 **Resource Usage Summary** 下, 查看 **Difficulty Packing Design** 报告。**Difficulty Packing Design** 详细报告 Fitter 在将设计适配到器件, 分区和 Logic Lock (Standard) 区域时的工作级别 (低, 中或高)。

随着 **Difficulty Packing Design** 的难度不断增加, 时序收敛变得更加困难。水平从中到高可能导致性能显著下降或编译时间增加。可考虑减少逻辑以降低封装难度。

3.3.1.6.2. 查看被忽略的约束

Compilation Report 包含被 Fitter 忽略的所有约束详情。如果设计名称更改且未更新约束, 则通常旧的约束会被忽略。请务必确保任何预期约束未被忽略。

3.3.1.6.3. 查看非默认设置

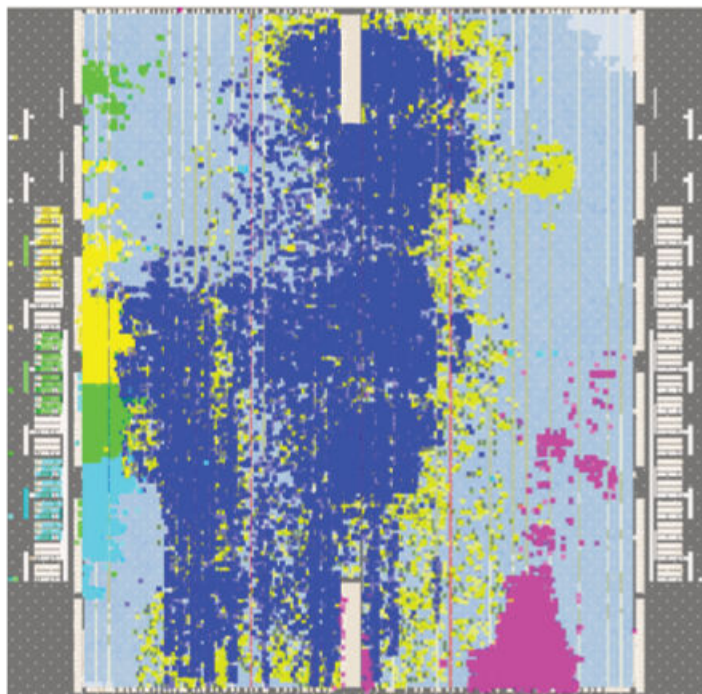
Synthesis 和 Fitter 的报告显示编译中使用的非默认设置。查看非默认设置以确保更改有助于改善设计。

3.3.1.6.4. 查看布局规划

使用 **Chip Planner** 查看布局。可使用 **Chip Planner**，通过布局规划平面图中每个实体的不同颜色来找到层次实体。对于未归位的逻辑，可根据您对其预计的位置进行查找。

例如，与 **I/O** 对接的逻辑应靠近 **I/O**，而与 **IP** 或存储器对接的逻辑应靠近 **IP** 或存储器。

图 20. 以颜色编码的布局规划图



以下注释说明如何使用 *Floorplan with Color-Coded Entities*（以颜色编码的布局规划图）中的可视性检查时序路径：

- 绿色块分散排开。检查这些路径是否时序失败，如果失败则与该模块连接的部分可能影响布局。
- 蓝色和水蓝色散开且混合相间。检查该状态是否因两种模块间的连接而形成。
- 位于底部的粉色逻辑必须与底部边缘的 **I/O** 对接。可通过任务栏上的按钮检查突出显示模块的扇入和扇出。
检查跨芯片的长程信号，并查看是否这些信号导致时序失败。
- 检查影响逻辑布局的信号的全局信号使用情况，并验证经 **Fitter** 置位的逻辑是否靠近其进行馈送的缓冲器，但远离相关逻辑。对非全局资源使用高扇出设置来拉拢逻辑。
- 检查布线拥塞。**Fitter** 中逻辑分散到高拥塞区域，使得设计中难以布线。

3.3.1.6.5. 评估布局布线

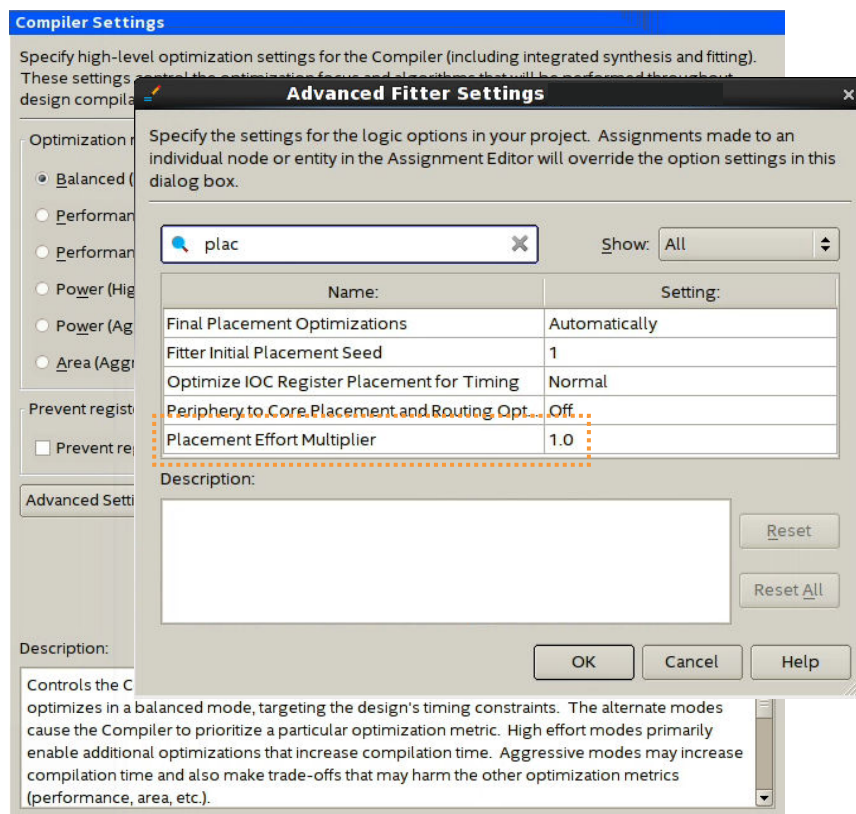
查看 **Fitter** 消息中编译时间的持续时间部分。如果布线比布局花费更多时间，则可能比预计中更难达到时序要求。

3.3.1.6.6. 调整布局工作量

可增加 **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) > Placement Effort Multiplier** 值，以在 Fitter 的 Place 阶段有更多编译时间和工作量。

查看和优化其他设置和 RTL 后，调整乘法器。提高该值（最大到 4），进行尝试。如果性能或编译时间并未提高，则复位到默认设置。

图 21. Placement Effort Multiplier



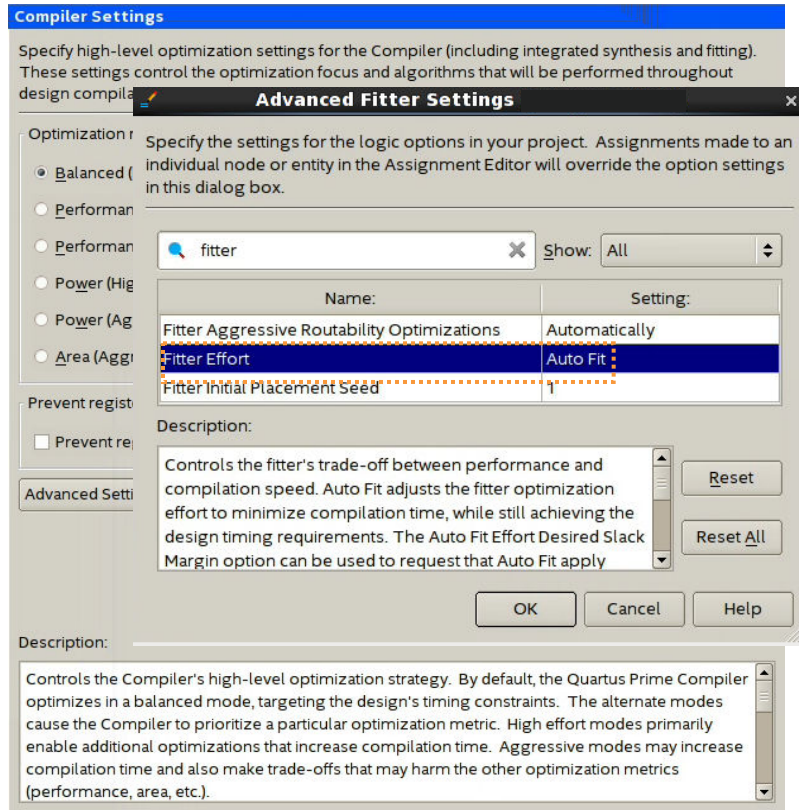
3.3.1.6.7. 调整 Fitter 工作强度

Fitter 中 **Optimization mode** 设置允许指定 Compiler 的优化工作量是否集中于性能，资源利用率，功率或编译时间。

By default, the Fitter **Optimization mode** is set to **Balanced (Normal flow)**, which reduces Fitter effort once timing requirements are met. You can optionally select another **Optimization mode** to target performance, power, or resource usage.

To increase Fitter effort further, you can also enable the **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) > Fitter Effort** option. The default **Auto Fit** setting reduces Fitter effort once timing requirements are met. **Standard Fit (highest effort)** setting uses maximum effort regardless of the design's requirements, leading to higher compilation time and more timing margin.

图 22. Fitter Effort



3.3.1.6.8. 查看时序约束

请确保使用正确频率要求约束时钟。使用 `derive_pll_clocks` 约束可保持更新已生成时钟设置。Timing Analyzer 可用于查看 SDC 约束。例如，Task（任务）窗中的 **Diagnostic** 下，**Report Ignored Constraints** 报告显示设计中的错误名称，而导致该错误的最普遍原因是设计层级结构中的更改。使用 **Report Unconstrained Paths** 报告查找不受约束的路径。按需添加约束，以便设计优化。

3.3.1.7. 评估集群难度

可评估集群难度以助于实现时序收敛。

随时添加逻辑和重编译后都可监控集群困难。使用集群信息判断设计中固有的时序收敛难度：

- 如果设计已满，但集群难度低或中等，则堵塞的原因很可能是设计本身而非集群问题。
- 相反，设计中添加少量逻辑后出现的堵塞，反而可能由集群所导致。如果集群难度大，则无论设计大小如何，都会导致堵塞。

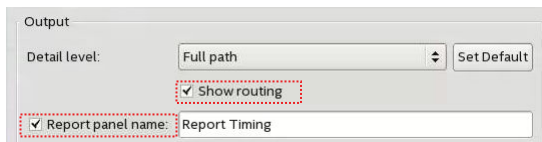
3.3.2. 查看时序路径详细信息

3.3.2.1. 查看时序路径布线

显示路径布线有助于发现异常布线延迟。

Timing Analyzer **Report Timing** 对话框中，开启 **Report panel name** 和 **Show routing** 选项，并点击 **Report Timing**。

图 23. **Report** 窗格和 **Show Routing** 选项



Extra Fitter Information 选项卡显示缩微布局规划图，其中突出显示路径。

您还可在 **Chip Planner** 中查找路径以检查布线拥塞，并查看路径中的节点是否紧密相邻或相距甚远。

相关链接

在 **Chip Planner** 中管理路径 (第 100 页)

3.3.2.2. 全局网络缓冲器

布线缆径支持确认时序失败的全局网络缓冲器。缓冲位置根据其所驱动的网络命名。

- CLK_CTRL_Gn—用于 Global（全局）驱动程序
- CLK_CTRL_Rn—用于 Regional（区域）驱动程序

访问全局网络的缓冲器位于器件每侧的正中。对全局信号网络上内核逻辑信号进行缓冲会导致插入延迟。如果信号降级为局部布线，则请考虑全局和非全局布线间的权衡，包括源位置，插入延迟，扇出，信号传播的距离，以及可能出现的拥塞。

3.3.2.2.1. 源位置

如果馈送全局缓冲的寄存器无法移动至更加靠近，则可考虑更改设计逻辑或布线类型。

3.3.2.2.2. 插入延迟

如果需要全局信号，则可考虑使用负边沿触发的寄存器生成信号（上图）并使用多周期建立约束（下图）从而对时序添加半个周期。

图 24. **Negative-Edge** 触发的寄存器

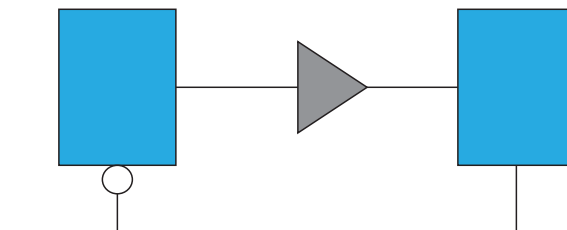
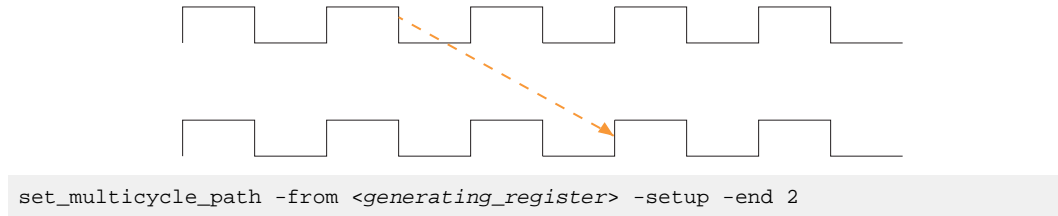


图 25. 多周期建立约束



3.3.2.2.3. 扇出

使用局部布线的高扇出节点倾向于将驱动到源节点附近的逻辑拉高。这样就可能使其他路径时序失败。复制寄存器可帮助减少高扇出路径的影响。考虑手动复制和保留这些寄存器。使用 MAX_FANOUT 约束可得到扇出节点的任意分组，而设计人员可得到更智能的扇出分组。

3.3.2.2.4. 全局网络

可使用 Global Signal 约束基于每个信号控制全局信号。例如，如果信号需要节点布线，将 Global Signal 约束设置为 **OFF**。

图 26. 全局信号约束

| To | Assignment Name | Value | Enabled |
|---------|-----------------|-------|---------|
| reg_clk | Global Signal | Off | Yes |

3.3.2.3. 复位和全局网络

复位信号通常布线于全局网络。有时，会因使用全局网络而导致恢复失败。可考虑查看生成复位和布线路径信号的寄存器布局。

3.3.2.4. Suspicious Setup

Suspicious setup（可疑建立）错误包括需要非常小或特别大的路径。

其中一个典型原因是数学精度误差。例如， $10\text{MHz}/3 = 33.33\text{ ns/周期}$ 。三个周期中，时间为 99.999 ns 对 100.000 ns。设置最大延迟可提供正确建立关系。

失败的另一原因是设计中有意的路径错误，例如：

- 通过 FIFO 处理的异步路径，或
- 依赖数据握手的缓慢异步路径在多个时钟周期保持可用。

为避免 Fitter 必须符合不必要的限制性时序要求，请考虑添加 **false** 或多周期路径声明。

3.3.2.5. 逻辑深度

Timing Analyzer 路径报告中的 **Statistics** 选项卡显示路径中的逻辑级别。如果路径中时序失败且逻辑级别数较高，则考虑在设计的该部分添加流水线。

3.3.2.6. 自动移位寄存器更换

Synthesis（综合）期间，Compiler 可将移位寄存器或寄存器转换为 RAM 以节省面积。然而，转换为 RAM 后通常会降低速度。转换后的寄存器的名称包括 “altshift_taps”。

- 如果路径时序错误出现在移位寄存器的开始或末尾，则可考虑禁用 **Auto Shift Register Replacement** 选项。请不要转换打算用于流水线的寄存器。
- 对于已转换为链的移位寄存器，请评估 RAM 或逻辑单元中实现的面积/速度权衡。
- 如果设计接近于满，则可将寄存器转换移位到 RAM，使非关键时钟域受益。可将设置从默认的 **AUTO** 更改为全局性 **OFF**，或基于寄存器或层次结构性关闭。

3.3.2.7. 时钟架构

为获得更佳时序结果，请将由区域时钟驱动的寄存器放置到芯片的某个象限中。可使用 **Chip Planner** 查看时钟区域边界。

当器件顶部的 I/Q 接口连接到器件某象限中由区域时钟驱动的逻辑时，可能出现时序失败，并且布局限制来自或到 I/O 的长路径跨象限到逻辑。

使用不同类型的时钟源驱动覆盖整个器件的逻辑-全局 (logic - global)，或覆盖器件一半的双区域。或者，可降低 I/O 接口的频率以适应长路径延迟。也可重新设计器件管脚说明，将所有指定 I/O 与区域时钟象限相邻放置。当寄存器位置受限时就会出现该问题，例如使用 **Logic Lock (Standard)** 区域，时钟资源或硬块（存储器，DSP，IP）。

Timing Analyzer 时序报告中的 **Extra Fitter Information** 选项卡会通知路径中节点布局何时受限。

相关链接

[查看器件中可用的时钟网络 \(第 96 页\)](#)

3.3.2.8. 时序收敛建议

Timing Analyzer 中的 **Report Timing Closure Recommendation**（报告时序收敛建议）任务分析路径并基于路径特征提供具体建议。

3.3.3. 调整和重新编译

找出可轻松解决的问题。要确定 **Compiler** 无法满足时序的位置，请通过大约 5 次编译执行 **seed** 扫描。这样就可连续显示失败路径。可考虑重新编写该设计部分的代码或重新设计该部分。

要实现时序收敛，编写良好的 RTL 可能比更改编译设置更有效。对于时序错误非常小，或者已经过性能优化并接近最终发布的设计，**Seed** 扫描也很有用。此外，**seed** 扫描还可用于评估编译设置的更改。编译结果会因为适配器算法的随机性而不同。如果编译设置的改变后产生低于平均水平的性能，则撤销该更改。

有时，一些设置或约束导致的问题甚至多于其修复的问题。对 RTL 或设计体系结构进行重大更改时，请使用默认设置，避开 **Logic Lock (Standard)** 区域进行周期性编译，并重新评估时序失败的路径。

分区处理通常对时序收敛毫无帮助，因而必须在设计进程的初期完成。如果其阻碍了跨边界优化，使时序收敛更加苦难并增加了编译时间，则添加分区可提高逻辑利用率。

3.3.3.1. 使用分区实现时序收敛

有一种实现时序收敛的技术为：将失败路径限制在单个设计分区内，从而分区之间不存在失败路径。然后可使用增量式编译按需进行更改以更正失败路径，并仅重新编译受影响的分区。

为使用该技术：

1. 可在 Design Partition Planner 中，单击 **View > Show Timing Data** 加载时序数据。
失败路径上包含节点的实体在 Design Partition Planner 中显示为红色。
2. 从顶层实体窗口拖动包含失败路径的实体以对其进行提取。
 - 如果已提取的实体和顶层实体中并无失败路径，则右键单击已提取实体，然后单击 **Create Design Partition** 将该实体放置到其所属分区中。
3. 将失败路径保持在分区中，以便分区间无交叉失败路径。
如果无法从已提取的实体中分离失败路径，则表示无跨分区边界的失败路径，因为可将实体还原至其主体中并不创建分区。
4. 找出具有最差时间裕量的分区。对于所有其他分区，保留内容并设置为 **Empty**。
关于保留分区内容的信息，请参阅 *Intel Quartus Prime Pro Edition 用户指南：基于块的设计* 中的 *基于编译流程的增量式模块*。
5. 调整分区中的逻辑并根据需要重新运行 **Fitter**，直到分区满足时序要求。
6. 对具有失败路径的所有其他设计分区重复此过程。

相关链接

- [查看设计连接性和层次结构](#) (第 0 页)
- [使用基于块的编译](#)
Intel Quartus Prime Pro Edition 用户指南：编译器
- [基于块的增量式编译流程](#)
In Intel Quartus Prime Pro Edition 用户指南：基于块的设计

3.4. 设计分析

初始编译确定设计是否实现成功适配并满足指定时序要求。本小节介绍如何在 Intel Quartus Prime 软件中分析设计结果。

3.4.1. 被忽略的时序约束

Intel Quartus Prime 软件忽略非法，过时和冲突的约束。

单击 **Reports > Report Ignored Constraints** 在 Timing Analyzer GUI 中查看被忽略约束清单或键入以下命令生成被忽略时序约束清单：

```
report_sdc -ignored -panel_name "Ignored Constraints"
```

分析任何被 Intel Quartus Prime 软件忽略的约束。如有必要，先更正约束并重新编译设计，然后才执行设计优化。

可在 Fitter 生成的 **Ignored Assignment Report** 中查看被忽略约束的列表。

相关链接

[Creating I/O Requirements](#)

3.4.2. I/O 时序

Timing Analyzer 支持 Synopsys* Design Constraints (SDC) 格式对设计进行约束。使用 Timing Analyzer 时序分析时，通过 `set_input_delay` 约束指定数据根据给定时钟到达出入端口的时间。对于输出端口，请使用 `set_output_delay` 命令指定数据根据给定时钟到达输出端口接收器的时间。可使用 `report_timing` Tcl 命令生成 I/O 时序报告。

未符合所需时序性能的 I/O 路径被报告为具有负时间裕量，并在 Timing Analyzer Report 窗口中以红色突出显示。即使您未对 I/O 管脚应用明确的 I/O 时序约束，而 Intel Quartus Prime 时序分析软件仍会报告 **Actual** 数目，表示器件在系统中运行时，时序参数必须满足的时序数量。

相关链接

[创建 I/O Requirements](#)

In *Intel Quartus Prime Standard Edition Handbook Volume 3*

3.4.3. Register-to-Register 时序分析

当时钟域上的任何 register-to-register（寄存器到寄存器）路径中无负时间裕量时，您的设计就能符合时序要求。未满足时序要求时，可从有关失败路径的报告中发现更多详细信息。

3.4.3.1. 使用 Timing Analyzer 显示 Path Report

Timing Analyzer 生成包含所有有效 register-to-register 路径的信息。为查看全部时序总结，可双击 **Tasks** 窗格中的 **Report All Summaries**。

如果任何时钟域有失败路径（在 **Report** 窗中以红色突出显示），右键点击 **Clocks Summary** 窗中罗列的时钟名称并选择 **Report Timing** 了解详细信息。

在 **Summary of Paths** 选项卡中选择路径时，路径详情窗显示所有路径信息。**Extra Fitter Information** 选项卡提供物理器件上路径位置的直观表示。从而透露时序失败是否与距离相关，是由于源和目标节点太过靠近或相距甚远。

Data Path 选项卡显示 Data Arrival Path 和 Data Required Path。可通过增量信息确定导致时序违规的最主要路径段。**Waveform** 选项卡显示时间域中的信号，并描述达到数据和所需数据间的时间裕量。

The RTL Viewer or Technology Map Viewer provide schematic (gate-level or technology-mapped) representations of the design netlist, and can help you to assess which areas in a design can benefit from reducing the number of logic levels. To locate a timing path in one of the viewers, right-click a path in the timing report, point to **Locate**, and select either **Locate in RTL Viewer** or **Locate in Technology Map Viewer**. You can also investigate the physical layout of a path in detail with the Chip Planner.

相关链接

- [何时使用 Netlist Viewers: 分析设计问题](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*
- [生成时序报告](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 3*

3.4.3.2. 分析失败路径的提示

分析失败路径时，检查报告和波形以确定是否应用正确的约束，并根据需要添加时序异常。多周期约束通过指定的时钟周期数放宽建立或保持关系。错误的路径约束会指定时序分析期间可被忽略的路径。两种约束都支持 **Fitter** 在受影响的路径上充分工作。

- 专注改善显示最差时间裕量的路径。**Fitter** 会针对最差时间裕量的路径充分工作。如果修复了这些路径，**Fitter** 就可能改善设计中其他失败时序路径。
- 检查出现在多个失败路径的节点。这些节点在时序报告窗中被置顶，并附带其最小时间裕量。查找具有公共源寄存器，目标寄存器，或公共中介组合性节点的路径。某些情况下，寄存器各不相同，但却是同一总线的其中一部分。
- 时序分析报告面板中，点击 **From** 或 **To** 栏的页眉以源或目标寄存器排列路径。如果看到公共节点，则这些节点表示设计中已通过更改源代码或 **Intel Quartus Prime** 优化而改善的区域。仅约束其中一个路径的布局可能导致器件中公共节点被移至远处从而使得时序性能下降。

相关链接

- [在 Chip Planner 中管理路径 \(第 100 页\)](#)
- [时序收敛的设计评估 \(第 43 页\)](#)

3.4.3.3. 分析跨时钟域失败时钟路径的提示

分析时钟路径失败时：

- 查看路径是否跨两个时钟域。
对于跨两个时钟域的路径，**From Clock** 和 **To Clock** 在时序报告中并不相同。

图 27. **From Clock** 和 **To Clock** 中的不同值

| Setup Transfers | | | | | | |
|-----------------|------------|----------|------------|----------|----------|----------|
| | From Clock | To Clock | RR Paths | FR Paths | RF Paths | FF Paths |
| 1 | clkin | clkin | 21 | 0 | 0 | 0 |
| 2 | clkin | clkout | false path | 0 | 0 | 0 |
| 3 | clkout | clkout | 31 | 0 | 0 | 0 |

- 查看设计中包含的路径是否存在路径中涉及不同时钟的情况，即使源和目标寄存器相同。
- 查看是否需要同步分析时钟域之间的失败路径。
将不需要同步分析的失败路径设置为错误路径。
- 对设计运行 **report_timing** 时，报告显示每个故障路径的启动时钟和锁存时钟，查看启动时钟和锁存时钟之间的关系是否切合实际，以及对已知设计所期望的内容。
例如，路径可从上升沿开始并在下降沿结束，从而将建立关系时间减少了半个时钟周期。
- 查看 **Timing Report** 中的时钟偏斜：

较大偏斜可能表示设计中存在问题，例如门控时钟，或物理布局中的问题（例如，使用局部布线而非专用时钟布线的时钟）。在已确保路径被同步分析且路径上无较大偏斜，以及约束正确后，就可以分析数据路径。这些步骤有助于微调跨时钟域的路径，以保证获得准确的时序报告。

- 查看 PLL 相移是否降低了建立要求。
可使用 PLL 参数和设置进行调整。
- 忽略跨时钟域的路径以获得受同步逻辑保护的逻辑（例如，FIFO 或双数据同步寄存器），即使时钟相关。
- 在所有不必要路径上设置错误路径约束：
尝试优化不必要路径可防止 Fitter 为满足时序路径上的时序要求而运行，对于设计来说至关重要。

相关链接

[report_clock_transfers](#)

Intel Quartus Prime Help 中

3.4.3.4. 分析来自/到关键路径的源和目标的提示

分析设计中失败路径时，通常有助于更加全面了解围绕路径的交互情况。

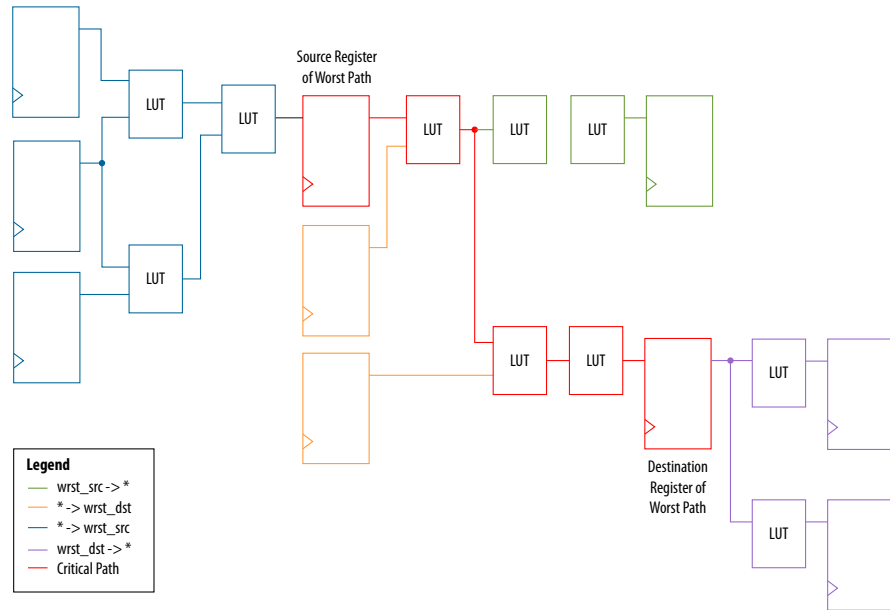
要了解关键路径上受牵制的内容，可使用以下 `report_timing` 命令。

1. 工程目录中，运行 `report_timing` 命令找到关键路径中的节点。
2. 复制 .tcl 文件中的代码（如下），并将首两个变量替换为最差路径的 **From Node** 和 **To Node** 栏中的节点名称。脚本分析最差源和目标寄存器之间的路径。

```
set wrst_src <insert_source_of_worst_path_here>
set wrst_dst <insert_destination_of_worst_path_here>
report_timing -setup -npaths 50 -detail path_only -from $wrst_src \
-panel_name "Worst Path||wrst_src -> *"
report_timing -setup -npaths 50 -detail path_only -to $wrst_dst \
-panel_name "Worst Path||* -> wrst_dst"
report_timing -setup -npaths 50 -detail path_only -to $wrst_src \
-panel_name "Worst Path||* -> wrst_src"
report_timing -setup -npaths 50 -detail path_only -from $wrst_dst \
-panel_name "Worst Path||wrst_dst -> *"
```

3. 从 **Script** 菜单，获取 .tcl 文件。
4. 在生成时序的面板中，找到 **Chip Planner** 中时序失败路径（以红色突出显示），并查看节点和较大扇出之间的距离等相关信息。
下图显示为报告分析的简化示例。

图 28. 时序报告



设计中的关键路径为红色。 .tcl 脚本和图示间的关系为：

- 首两行显示关键路径的两个端点中的全部内容，并将其导向不同方向。
 - 第一个 `report_timing` 命令分析源正驱动的所有路径，以绿色显示。
 - 第二个 `report_timing` 命令分析到目标寄存器的所有路径，包括关键路径，以橙色显示。
- 最后两个 `report_timing` 命令显示端点之外的所有内容，并将其导向其他方向。

如果这些邻近路径中的任何一个具有时间裕量且靠近关键路径，则 **Fitter** 会将这些路径与关键路径进行平衡，以尝试实现最佳时间裕量。

3.4.3.5. 创建.tcl 脚本监控跨编译的关键路径的提示

许多设计在每次编译后都会显示相同的关键路径。而其他设计中，关键路径在不同层次结构间弹动，并随着每次编译而变化。

该行为出现在高速设计中且 **register-to-register** 路径大多具有极少量时序裕量。不同的布局可能导致边缘路径中时序失败。

- 在工程目录中，创建一个脚本名称 `TQ_critical_paths.tcl`。
- 编译后，查看关键路径，然后编写通用 `report_timing` 命令采集这些路径。

例如，如果低级别层级结构中有多个路径失败，请添加命令，如

```
report_timing -setup -npaths 50 -detail path_only \  
  -to "main_system: main_system_inst|app_cpu:cpu|*" \  
  -panel_name "Critical Paths||s: * -> app_cpu"
```

3. 如果存在特定路径，例如状态机的某个位要到其他*count_sync*寄存器，就可添加如下类似命令：

```
report_timing -setup -npaths 50 -detail path_only \
  -from "main_system: main_system_inst|egress_count_sm:egress_inst|
update" \
  -to "*count_sync*" -panel_name "Critical Paths||s: egress_sm|
update -> count_sync"
```

4. 每次编译后在 **Timing Analyzer** 中执行该脚本，并在新的关键路径出现时添加新的 `report_timing` 命令。

这样有助于您监控持续失败的路径以及仅为边际的路径，从而有效确定优先级。

3.4.3.6. 全局布线资源

全局布线资源旨在约束高扇出，低偏斜信号（如，时钟）且不消耗常规布线资源。基于器件的不同，这些资源可遍布整个芯片或较小部分，如象限。**Intel Quartus Prime** 软件尝试自动为全局布线资源约束信号，但也可手动进行更合适的约束。

关于可用全局布线资源的数量和类型，请参阅相关器件手册。

查看设计中全局信号利用率，可确保全局布线资源上放置了正确的信号。**Compilation Report** 中，打开 **Fitter** 报告并单击 **Resource Section**。分析 **Global & Other Fast Signals** 和 **Non-Global High Fan-out Signals** 报告以确定是否需要任何更改。

可通过将高扇出信号放置到全局布线资源上来减少偏斜。反之，可将全局布线资源上的低扇出移除来减少插入延迟。这样就可提高时钟使能时序并控制信号恢复/移除时序，但会增加时钟偏斜。使用 **Assignment Editor** 中的 **Global Signal** 设置控制全局布线资源。

3.5. 时序优化

如果设计未能满足时序要求，则请使用以下指导。

3.5.1. 显示失败路径的时序收敛建议

使用 **Timing Closure Recommendations** 报告获得关于设计失败路径的特定建议以及可能修复失败路径的各种更改。

1. **Timing Analyzer** 的 **Tasks** 窗格中，选择 **Report Timing Closure Recommendations** 任务，打开 **Report Timing Closure Recommendations** 对话框。
2. 基于时钟域选择路径，按路径上的节点进行过滤，然后选择要分析的路径数量。
3. **Timing Analyzer** 中运行完 **Report Timing Closure Recommendations** 任务后，从 **Timing Analyzer GUI** 的 **Report** 窗格检查 **Report Timing Closure Recommendations** 文件夹中的报告。每个建议以星号（*）标注。带有多颗星号的建议更可能帮助您设计中的时序收敛。

报告为您提供每个已分析路径中最有可能的失败原因，并显示有助于修复失败路径的建议。

报告分为几个部分，具体取决于从设计中发现的问题类型，例如加大时钟偏斜，受限优化，不均衡逻辑，跳过优化，寄存器之间的编码样式存在过多逻辑级别，或特定于您的工程的区域或分区约束。

要详细分析关键路径，请在指定路径上运行 `report_timing` 命令。在 **Path** 报告窗格的 **Extra Fitter Information** 选项卡中，可查看与适配相关详细信息，有助于更直观看待问题。

相关链接

- [Fast Forward 时序收敛建议](#) (第 0 页)
- [报告时序收敛建议对话框](#)
Intel Quartus Prime Help 中

3.5.2. 时序优化向导

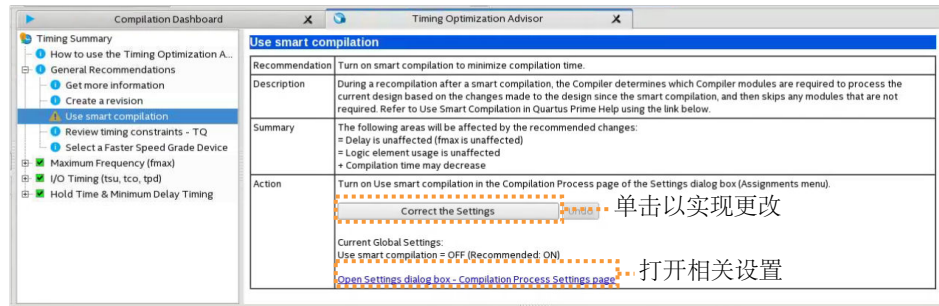
Timing Analyzer **Report Timing Closure Recommendations** 任务提供修复失败路径的特定建议的同时，Timing Optimization Advisor 还提供更多改善设计时序性能的常规建议。

Timing Optimization Advisor 指导针对设计优化的设置以满足时序要求。要运行 Timing Optimization Advisor 可单击 **Tools > Advisors > Timing Optimization Advisor**。该向导说明本节中提出的多个建议。

编译后开启 Timing Optimization Advisor，可找到提高设计中时序性能的建议。如果各指导中的建议相互矛盾，则请评估各选项并选择最适合给定要求的设置。

本实例显示了 Timing Optimization Advisor，在通过编译设计来满足其频率要求后，还需要更改设置提高时序性能。

图 29. 时序优化向导



在 Timing Optimization Advisor 中展开其中一个类别，如 **Maximum Frequency (fmax)** 或 **I/O Timing (tsu, tco, tpd)** 时，建议会分阶段呈现。这些阶段显示应用建议设置的顺序。

首个阶段包含最容易更改的选项，通过最轻度的更改来优化设计，并对编译时间的影响最小。

图标标示当前工程中是否进行了每个推荐设置。图示中，**Stage 1** 的建议清单中的勾选图标表示已实现的建议。警告图标表示本此编译中未采纳的建议。信息图标表示常规建议。对于这些实体，向导中并不报告是否采纳这些建议，但会解释如何实现更佳性能。关于提供每个图标详细信息的图例，请参阅 Timing Optimization Advisor (时序优化向导) 中的 “How to use” 页。

每个建议提供前往 Intel Quartus Prime GUI 中正确位置的链接，可在此更改设置。例如，可考虑 **Settings** 对话框中的 **Synthesis Netlist Optimizations** 页或 **Assignment Editor** 中的 **Global Signals category**。此方法可最大程度控制需要的设置并帮助了解软件中的设置。适时还可使用 **Correct the Settings** 按钮自动更改全局设置。

对于 Timing Optimization Advisor 中的某些条目，可使用该按钮进一步分析设计并查看更多信息。本向导提供包含设计中各时钟的表格，以显示是否已对时钟进行时序约束。

3.5.3. 可选 Fitter 设置

本小节仅关注可选时序优化 Fitter 设置，即 **Optimize Hold Timing**、**Optimize Multi-Corner Timing** 和 **Fitter Aggressive Routability Optimization**。

警告:

不同设计的最佳优化设置各不相同。对某设计最有效的一组设置并不一定对另一设计产生最好结果。

相关链接

高级 Fitter 设置对话框

Intel Quartus Prime Help 中

3.5.3.1. 优化保持时序

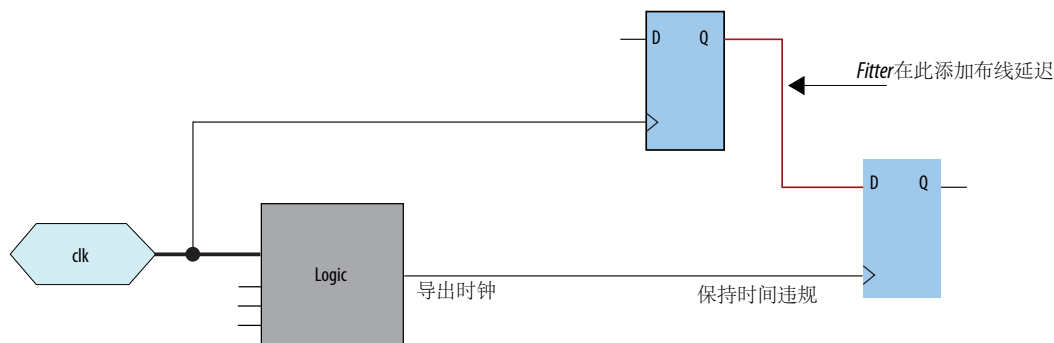
Optimize Hold Timing 选项指示 Intel Quartus Prime 软件优化最小延迟时序约束。Check your device information to determine whether the Intel Quartus Prime software optimizes hold timing for all paths or only for I/O paths and minimum t_{PD} paths.

从 **Advanced Fitter Settings** 对话框中开启 **Optimize Hold Timing** 后，Intel Quartus Prime 软件会增加路径延迟，以确保您的设计满足最小延迟要求。如果选择 **I/O Paths** 和 **Minimum TPD Paths**，则 Fitter 运行以符合如下条件：

- 从器件输入管脚到寄存器的保持时间(t_H)
- 从 I/O 管脚到 I/O 寄存器或从 I/O 寄存器到 I/O 管脚的最小延迟
- 从寄存器到输出管脚的最小时钟输出时间(t_{CO})

如果选择 **All Paths**，Fitter 还可运行至满足从寄存器到寄存器的保持要求（如图中蓝色突出显示），其中由逻辑驱动产生而得的时钟会导致另一寄存器上出现保持时间问题。

图 30. 优化保持时间选项修复内部保持时间违规



然而，如果您设计中寄存器之间仍然存在内部保持时间违规，则可通过例化 LCELL 原语，或更改设计来手动添加延迟，例如使用时钟使能信号而非导出或门控时钟。

相关链接

建议的设计实践

In Intel Quartus Prime Standard Edition User Guide: Design Recommendations

3.5.3.2. Fitter 主动布通性优化

Fitter Aggressive Routability Optimizations (主动布通性优化) 逻辑选项支持可指定 Fitter 主动优化的布通性。执行主动布通性优化可能减慢设计速度，但也会减少布线线缆使用和布线时间。

如果布线资源导致不适配错误，且您希望减少布线线缆的使用，则该选项大有用处。

本表格罗列了 **Fitter Aggressive Routability Optimizations** 逻辑选项的设置内容。

表 8. Fitter 主动布通性优化逻辑选项设置

| 设置 | 说明 |
|---------------|--|
| Always | Fitter 时钟执行主动布通性优化。如果将 Fitter Aggressive Routability Optimizations 逻辑选项设置为 Always ，会降低线缆利用率从而可能影响设计性能。 |
| Never | Fitter 从不执行主动布通性优化。如果盖上时序比减少线缆使用更重要，则将该选项设置为 Automatically 或 Never 。 |
| Automatically | Fitter 基于设计的布通性和时序要求，自动执行主动布通性优化。如果改善时序比降低线缆使用率更重要，则将该选项设置为 Automatically 或 Never 。 |

3.5.4. I/O 时序优化技术

该设计阶段侧重于 I/O 时序，包括建立延迟 (t_{SU})，保持时间 (t_H)，和时钟-输出 (t_{CO}) 参数。

开始 I/O 时序优化之前，请确保：

- 设计中的约束按照 *Design Optimization Overview* (设计优化概述) 章节中 *Initial Compilation: Required Settings* (初始编译：必要设置) 部分的建议。
- 资源利用率符合要求。

可小节中的建议可应运于所有 Intel FPGA 系列和 CPLD 的系列。

注意：需先完成该阶段再开始寄存器-寄存器 (register-to-register) 时序优化阶段前。更改 I/O 路径会影响内部 register-to-register 时序。

改善建立和时钟到输出时间的技术总结

本表格列出应用各建议技术时的顺序以减少 t_{SU} 和 t_{CO} 时间。减少 t_{SU} 时间会增加保持 (t_H) 时间。

注意：验证可用于每个器件系列的选项

表 9. 改善建立和时钟到输出时间

| 顺序 | 技术 | 影响 t_{SU} | 影响 t_{CO} |
|-------|---|-------------|-------------|
| 1 | 验证是否对失败的 I/O 设置了正确的约束 (请参阅 <i>初始编译：必需设置</i>) | Yes | Yes |
| 2 | 对 I/O 使用时间驱动的编译 (请参阅 <i>快速输入，输出和输出使能寄存器</i>) | Yes | Yes |
| 3 | 使用快速输入寄存器 (请参阅 <i>可编程延迟</i>) | Yes | N/A |
| 4 | 使用快速输出寄存器，快速输出使能寄存器和快速 OCT 寄存器 (请参阅 <i>可编程延迟</i>) | N/A | Yes |
| 5 | 减小 Input Delay from Pin to Input Register 的值或设置 Decrease Input Delay to Input Register = ON | Yes | N/A |
| 继续... | | | |

| 顺序 | 技术 | 影响 t_{SU} | 影响 t_{CO} |
|----|--|-------------|-------------|
| 6 | 减小 Input Delay from Pin to Input Register 的值或设置 Decrease Input Delay to Input Register = ON | Yes | N/A |
| 7 | 减小 Delay from Output Register to Output Pin 的值或设置 Increase Delay to Output Pin = OFF (请参阅 快速输入，输出和输出使能寄存器) | N/A | Yes |
| 8 | 增加 Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations 的值 (请参阅 快速输入，输出和输出使能寄存器) | Yes | N/A |
| 9 | Use PLLs to shift clock edges | Yes | Yes |
| 10 | Use Fast Regional Clock (请参阅 <i>Change How Hold Times are Optimized for MAX[®] II Devices</i>) | N/A | Yes |
| 11 | 对于 MAX II 或 MAX V 系列器件，将 Guarantee I/O Paths Have Zero Hold Time at Fast Corner 设置为 OFF ，或 When T_{SU} and T_{PD} Constraints Permit (请参阅 更改 MAX II 器件的保持时间优化方式) | Yes | N/A |
| 12 | 增加 Delay to output enable pin 的值或设置 Increase delay to output enable pin (请参阅 Use PLLs to Shift Clock Edges) | N/A | Yes |

[优化时序逻辑的 IOC 寄存器布局选项 \(第 64 页\)](#)

[快速输入，输出和输出使能寄存器 \(第 65 页\)](#)

[可编程延迟 \(第 65 页\)](#)

[使用 PLL 移位时钟沿 \(第 66 页\)](#)

[使用 Fast Regional Clock Network 和 Regional Clocks Network \(第 66 页\)](#)

[脊柱时钟限制 \(第 66 页\)](#)

[更改器件的保持时间优化方式 \(第 67 页\)](#)

相关链接

[初始编译的必需设置 \(第 7 页\)](#)

3.5.4.1. 优化时序逻辑的 IOC 寄存器布局选项

该选项将寄存器移入 I/O 单元以从而满足 t_{SU} 或 t_{CO} 约束，必要时复制寄存器（如一个寄存器扇出对应多个输出位置的情况）。该选项默认开启，并且是全局设置。

注意: 该选项不适用于系列器件，由于其不具备 I/O 寄存器。

Optimize IOC Register Placement for Timing 逻辑选项仅影响具有 t_{SU} 或 t_{CO} 要求的管脚。可能仅当寄存器直接馈送管脚或由管脚直接馈送时才使用 I/O 寄存器。因此，该逻辑选项不影响如下特性的寄存器：

注意: 要优化具有这些特征的寄存器，请使用其他 Intel Quartus Prime Fitter 优化。

- 寄存器和管脚之间具有组合逻辑
- 作为进位链或级联链的一部分
- 具有覆盖性位置约束
- 使用异步加载端口且值不为 1（在该端口可用的器件系列中）

相关链接

优化时序逻辑的 **IOC** 寄存器布局选项
Intel Quartus Prime Help 中

3.5.4.2. 快速输入，输出和输出使能寄存器

通过 **Assignment Editor** 进行快速 I/O 约束，可手动将单个寄存器放入 I/O 单元中。默认情况下，按照正确的时序约束，**Fitter** 会将寄存器放置到正确的 I/O 单元或内核中，以满足性能要求。

在系列器件中，无 I/O 寄存器，因此如果存在被约束到 I/O 管脚管脚位置则时序约束将与该 I/O 管脚邻近的寄存器锁存为 LAB。

如果快速 I/O 设置为开启，则寄存器始终放置于 I/O 单元中。如果快速 I/O 设置为关闭，寄存器从不被放置到 I/O 单元。即使 **Optimize IOC Register Placement for Timing** 选项开启，也是如此。如果不存在快速 I/O 约束，在开启 **Optimize IOC Register Placement for Timing** 选项时，Intel Quartus Prime 软件决定是否将寄存器放入 I/O 单元中。

还可使用 4 个快速 I/O 选项（**Fast Input Register**，**Fast Output Register**，**Fast Output Enable Register** 和 **Fast OCT Register**）覆盖 Logic Lock (Standard) 区域中寄存器的位置，并强制其进入 I/O 单元。如果对馈送多个管脚的寄存器应用该约束，则 **Fitter** 复制该寄存器并将其放入所有相关 I/O 元件中。

系列器件中，**Fitter** 复制寄存器并将其放入按管脚约束的管脚旁边的每个不同的 LAB 处。

更多关于 **Fast Input Register** 选项，**Fast Output Register** 选项，**Fast Output Enable Register** 选项和 **Fast OCT (on-chip termination) Register** 选项的信息，请参阅 Intel Quartus Prime Help。

相关链接

- 快速输入寄存器逻辑选项
- 快速输出寄存器逻辑选项
- 快速输出使能寄存器逻辑选项
- 快速 OCT 寄存器逻辑选项

3.5.4.3. 可编程延迟

可使用各种可编程延迟选项最小化 t_{SU} 和 t_{CO} 时间。可编程延迟是旨在编译工程后才可使用的高级选项，检查 I/O 时序，并确定是否合格。

对于 Arria®、Cyclone®、MAX II、MAX V 和 Stratix® 系列器件，Intel Quartus Prime 软件自动调整适用的可编程延迟，以帮助满足时序要求。关于这些选项的效果的详细信息，请参阅器件系列手册或数据表。

完成可编程延迟约束并编译设计后，可在 **Compilation Report** 的 **Delay Chain Summary** 部分查看每个延迟链和每个 I/O 管脚的已实现延迟值。

可使用 **Assignment Editor** 为支持的节点约束可编程延迟选项。还可通过 **Chip Planner** 和 **Resource Property Editor** 查看并修改目标器件的延迟链设置。执行完整编译后，使用 **Resource Property Editor** 进行更改时不需要重新编译整个设计；可直接将更改保存到网表。由于是直接对网表进行更改，重新编译时就不再自动进行更改。更改管理功能支持在后续编译中再次应用所更改内容。

尽管新近器件中的可编程延迟为用户可控，但 Intel 建议仅高级用户使用。然而 Intel Quartus Prime 软件可能在 Fitter 阶段内部使用可编程延迟。

关于 Intel 器件可用的可编程延迟逻辑选项，请参阅 Intel Quartus Prime Help 主题：

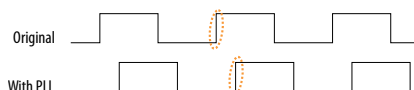
相关链接

- [从管脚到输入寄存器的输入延迟逻辑选项](#)
 - [从管脚到内部单元的输入延迟逻辑选项](#)
 - [输出使能管脚延迟逻辑选项](#)
 - [从输出寄存器到输出管脚逻辑延迟的逻辑选项](#)
 - [从复用时钟管脚到扇出目的地的输入延迟的逻辑选项](#)
- Intel Quartus Prime Help 中*

3.5.4.4. 使用 PLL 移位时钟沿

使用 PLL 通常可自动提高 I/O 时序性能。如果仍未满足时序要求，则大多数器件允许 PLL 输出相移来更改 I/O 时序。向后移位时钟会牺牲 t_{SU} 来提供较好的 t_H ，而时钟向前移位则牺牲 t_H 但提供较好的 t_{SU} 。仅可在提供具有相位移动选项的 PLL 的器件中使用该技术。

图 31. 向前移位时钟沿牺牲 t_H 来提高 t_{SU}



通过使用名为 **Input Delay from Dual Purpose Clock Pin to Fan-Out Destinations** 的可编程延迟，可在某些器件中实现相同类型的效果。

3.5.4.5. 使用 Fast Regional Clock Network 和 Regional Clocks Network

区域时钟（regional clock）为单个象限中的逻辑提供最低时钟延迟和偏斜。一般而言，快速区域时钟（fast regional clock）到 I/O 单元的延迟小于区域和全局时钟，并且用于高扇出控制信号。在这些低偏斜和低延迟时钟网络上放置时钟可提供更好的 t_{CO} 性能。

Intel 器件具有各种层次时钟结构。这些包括专用全局时钟网络，区域时钟网络，快速区域时钟网络和外设时钟网络。不同 Intel 器件系列之间的可用资源各不相同。

有关每个目标器件中可用的时钟资源数量，请参阅相关器件手册。

3.5.4.6. 脊柱时钟限制

在高时钟布线要求的工程中，Intel Quartus Prime 软件中的限制会导致脊柱时钟（spine clock）错误。这些错误通常出现在使用多个存储器接口和高速串行接口（HSSI）通道的设计中，尤其是 PMA Direct 模式。

全局时钟网络，区域时钟网络和外设时钟网络具有其他级别的时钟网络，称为 spine clock。Spine clock 将最后行和列时钟驱动到寄存器；因此，芯片中的每个时钟通过 spine clock 到达每个芯片。Spine clock 为用户非直接可控。

要减少 spine clock 有关的错误，请对设计进行约束以更好使用您的区域时钟资源：

- 如果您的设计中未使用 **Logic Lock (Standard)** 区域，或者如果 **Logic Lock (Standard)** 区域未与您的时钟域边界对齐，则请创建其他 **Logic Lock (Standard)** 区域并进一步约束您的逻辑。
- 如果 **Periphery** 功能忽略 **Logic Lock (Standard)** 区域约束，很可能由于全局时钟资源自动检测过程未正常运转。为确保全局时钟资源自动检测过程使用正确的位置，可为使用这些外设功能的 I/O 约束指定管脚。
- 默认情况下，一些 Intel FPGA IP 功能应用值为“双区域时钟”的全局信号约束。如果将您的逻辑约束到区域性时钟区域并将全局时钟设置为 **Regional** 而非 **Dual-Regional**，则可减少时钟资源争用。

相关链接

- [查看器件中可用的时钟网络](#) (第 96 页)
- [层次设置和编辑模式](#) (第 94 页)
- [报告 Spine Clock 使用率对话框 \(Chip Planner\)](#)
Intel Quartus Prime Help 中

3.5.4.7. 更改器件的保持时间优化方式

对于器件，可使用 **Guarantee I/O Paths Have Zero Hold Time at Fast Corner** 选项控制 Intel Quartus Prime 软件优化保持时间的方式。

3.5.5. Register-to-Register 时序优化技术

设计优化的下一阶段旨在提高 register-to-register (f_{MAX}) 时序。如果编译后设计无法符合时序要求，则以下部分提供可用选项。

编码风格比更改设置更大程度上影响设计性能。始终评估代码并确保使用同步设计实践。

注意：在 Timing Analyzer 的上下文中，register-to-register 时序优化与最大化设计中时钟域上的时间裕量效果相同。本小节中的技术可改善设计中不同时序路径上的时间裕量。

执行设计优化之前，请先了解设计结构以及各种技术在不同逻辑类型中的效果。无法使逻辑结构受益的技术甚至会降低设计性能。

相关链接

建议的设计实践

In Intel Quartus Prime Standard Edition User Guide: Design Recommendations

3.5.5.1. 优化源代码

多数情况下，优化设计源代码能非常显著提高设计性能。事实上，优化您的源代码通常是提高设计结果质量最有效的技术，往往是比使用 **Logic Lock (Standard)** 或位置约束更好的选择。

编码时请留意实现设计中的逻辑所需要的逻辑级别的数量。寄存器之间过多的逻辑级别可能导致关键路径时序失败。请尝试重构设计以使用流水线或更有效的编码技术。另外，尝试限制源代码中的高扇出信号。如有可能，请复制并流水线控制信号。务必确保复制寄存器由保留属性保护，从而避免综合期间进行合并。

如果设计中的关键路径涉及存储器或 DSP 功能，请检查设计中说明存储器或功能的编码块是否未经推断且未布局到专用逻辑中。可修改源代码使得这些功能放入高性能专用存储器或目标器件的资源中。使用 RAM/DSP 块时，使能可选输入和输出寄存器。

请确保您的状态机被识别为状态逻辑且在综合工具中被适当优化。已识别出的状态机通常会被优化，否则被当作一般逻辑处理。在 Intel Quartus Prime 软件中，可从 **Compilation Report** 的 **Analysis & Synthesis** 下查看 **State Machine** 报告。该报告提供包括编译期间识别出的每个状态机的状态编码等各种详细信息。如果无法识别您的状态机，则可能必须更改源代码才能识别。

相关链接

[AN 584: 高级 FPGA 设计的时序收敛方法](#)

3.5.5.2. 改善 Register-to-Register 时序

对于提高时序裕量 (slack) 或改善 register-to-register 时序的选项和设置的选择取决于设计中的失败路径。要实现最接近性能要求的结果，请应用如下技术并在每个步骤后编译设计：

1. 确保完成时序约束及其正确性。请参阅 *设计优化概述* 章节中的 *初始化编译：必要设置* 部分获得详细信息。
2. 查看初始化编译过程中的所有警告消息，并查看被忽略的时序约束。
3. 应用网表综合优化选项。
4. 要优化速度，可应用以下综合选项：
 - Optimize Synthesis for Speed, Not Area (优化关于速度而非面积的综合)
 - Flatten the Hierarchy During Synthesis (综合期间展开层级结构)
 - Set the Synthesis Effort to High (Synthesis Effort 设置为 High)
 - Change State Machine Encoding
 - Prevent Shift Register Inference (防止 Shift Register 推断)
 - Use Other Synthesis Options Available in Your Synthesis Tool (使用 Synthesis Tool 中的其他可用综合选项)
5. To optimize for performance using physical synthesis, apply the following options:
 - Enable physical synthesis
 - Perform automatic asynchronous signal pipelining
 - Perform register duplication
 - Perform register retiming
 - Perform logic to memory mapping
6. 尝试各种 Fitter seed。如果只有少量路径因少许负时间而导致时序失败，则可尝试使用其他 seed 找出能够符合 Fitter seed 噪音要求的约束。
注意： 如果大量路径出现故障，或因长时间裕量造成的路径失败，则忽略此步骤。
7. 为控制布局，需进行 Logic Lock (Standard) 约束。
8. 修改设计源代码以修复设计中因大量时间裕量而无法及时序要求的设计区域。
9. 进行位置约束，或万不得已，通过反向标注设计执行手动布局。

可使用 Design Space Explorer II (DSE) 自动运行不同设置下的各种编译过程。

如果这些技术都无法满足性能要求，则可能需要修改其他设计源代码。

相关链接

初始编译：必要设置

Intel Quartus Prime Standard Edition User Guide: Design Optimization

3.5.5.3. 物理综合优化

Intel Quartus Prime 软件提供有助于提高设计性能的物理性综合优化，且无需考虑综合工具因素。可在综合或适配期间应用物理中和优化。

在 Intel Quartus Prime 编译的综合阶段过程中，既可在其他 EDA 综合工具的输出上运行物理性优化，也可将其作为综合过程中的一个中间步骤进行。这些优化将修改综合网表以提高面积或速度，具体取决于您选择的技术和效力级别。

要查看并修改综合网表优化选项，请点击 **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**。

如果使用第三方 EDA 综合工具并需要确定是否 Intel Quartus Prime 软件可通过重新映射电路来提高性能，可使用 **Perform WYSIWYG Primitive Resynthesis** 选项。该选项指示 Intel Quartus Prime 软件取消原子网表到逻辑门控的 LE 映射，然后将门控映射回 Intel 特定原语。Intel 特定原语使能 Fitter 使用指定体系结构技术重新映射电路。

Intel Quartus Prime 技术映射器根据 **Optimization Technique** 选项的设置优化设计以实现最大速度性能，最小化使用面积，或平衡高性能和最小化逻辑使用。将该选项设置为 **Speed** 或 **Balanced**。

Intel Quartus Prime 编译的 Fitter 阶段中，物理综合优化在网表中进行指定布局更改，以提高特定 Intel 器件的速度性能结果。

注意:

If you want the performance gain from physical synthesis only on parts of your design, you can apply the physical synthesis options on specific instances with the Assignment Editor.

相关链接

- 执行 **WYSIWYG 原语再综合逻辑选项**
- 优化技术逻辑选项
Intel Quartus Prime Help 中

3.5.5.4. 关闭 Extra-Effort Power 优化设置

如果功率优化设置为 **Extra Effort**，则可能影响您的设计性能。如果时序性能比功率更重要，则请将电源优化设置为 **Normal**。

相关链接

- **Power Optimization** (第 0 页)
- 电源优化逻辑选项
Intel Quartus Prime Help 中

3.5.5.5. 优化关于速度而非面积的综合

设计性能因编码样式，综合工具的使用和综合时指定的选项而各不相同。如果出现大量路径失败，或由于较大时间裕量和过多逻辑级别导致特定路径失败，就可尝试更改综合选项。

确定综合工具中默认的优化目标，并相应设置器件和时序约束。例如，如果未指定目标频率，某些综合工具会针对面积进行优化。

可使用 **Assignment Editor** 为设计中的特定模块指定逻辑选项，与此同时，对于特定器件系列中对面积和速度的最佳权衡，可保持默认 **Optimization Technique** 设置为 **Balanced**，或者如果面积是重要考虑因素，则将上述选项设置为 **Area**。还可使用 **Assignment Editor** 中的 **Speed Optimization Technique for Clock Domains** 选项指定针对速度而优化的特定时钟域中或相互之间的全部组合逻辑。

要通过按钮编译获得最佳性能，请按照如下部分中的建议进行其他综合设置。可使用 **DSE II** 通过各个 **Intel Quartus Prime** 综合选项进行设计优化实验以获得最佳性能。

相关链接

[优化技术逻辑选项](#)

Intel Quartus Prime Help 中

3.5.5.6. 综合期间展开层级结构

综合工具通常允许保留层次边界，以利于验证或其他用途。然而，当综合工具跨层次边界进行优化时，通常出现最佳优化结果，由于这样操作往往能允许综合工具执行最大程度的逻辑最小化，进而提高性能。尽可能展开设计层次以实现最佳效果。

注意:

If you are using Intel Quartus Prime incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions. Follow Intel's recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.

3.5.5.7. Synthesis Effort 设置为 High

Synthesis 工具提供不同综合效力水平以权衡编译时间和综合结果。适用时将综合效力设置为 **high** 获得最佳结果。

3.5.5.8. Change State Machine Encoding

State machines can be encoded using various techniques. One-hot encoding, which uses one register for every state bit, usually provides the best performance. If your design contains state machines, changing the state machine encoding to one-hot can improve performance at the cost of area.

相关链接

[State Machine Processing Logic Option online help](#)

3.5.5.9. 复制扇出控制逻辑

通常，时序失败的出现并不是因为高扇出寄存器，而是这些寄存器的位置。复制寄存器，其中源和目标寄存器物理上接近，有助于改善关键路径上的时间裕量。

综合工具支持指定寄存器最大扇出的选项或属性。使用 **Intel Quartus Prime** 集成综合时，可设置 **Assignment Editor** 中的 **Maximum Fan-Out** 逻辑选项控制节点的目标的数量，使得扇出计数不超过指定值。还可在 HDL 代码中使用 `maxFan` 属性。软件根据需要复制节点以实现指定的最大扇出。

使用 **Maximum Fan-Out** 约束的逻辑复制通常会提高资源利用率，并可能潜在提高编译时间，具体取决于所选器件中的布局以及资源使用总体情况。

由 **Maximum Fan-Out** 约束产生的时序性能提高因设计而异。这是由于使用 **Maximum Fan-Out** 约束时，Fitter 复制源逻辑以限制扇出，但并不控制每个复制源驱动的目标。因此，复制的源逻辑可以是位于器件周围的驱动逻辑。为避免这种情况，可使用 **Manual Logic Duplication** 逻辑选项。

如果使用 **Maximum Fan-Out** 约束，测量在使用和不使用这些约束时的性能以评估这些约束是否提供期望的时序性能提高。仅在获得改进结果时才使用该约束。

可在 Intel Quartus Prime 软件中手动复制寄存器，无论使用何种综合工具。要复制寄存器，请通过 Assignment Editor 针对寄存器应用 **Manual Logic Duplication** 逻辑选项。

注意: 某些 Fitter 优化可能引起改善时序的 **Maximum Fan-Out** 约束出现少量违规。

3.5.5.10. 防止 Shift Register 推断

关闭移位寄存器推断可提高性能。该设置强制软件使用逻辑单元以实现移位寄存器，而并非使用 `ALTSHIFT_TAPS` IP 核实现存储器块中的寄存器。如果在逻辑单元而非存储器中实现移位寄存器，则会增加逻辑利用率。

3.5.5.11. 使用 Synthesis Tool 中的其他可用综合选项

使用您的综合工具，尝试以下选项是否可用：

- Turn on register balancing or retiming (开启寄存器权衡或重新定时)
- Turn on register pipelining (开启寄存器流水线)
- Turn off resource sharing (关闭资源共享)

这些选项可提高性能，但通常也会提高设计中的资源使用率。

3.5.5.12. Fitter Seed

Fitter seed 影响设计的初始布局配置。初始条件的任何变化都会改变 Fitter 结果；因此，每个 seed 值导致一些不同的适配结果。可尝试不同的 seed，以图获得更好的适配结果和时序性能。

设计中的更改会影响编译之间的性能。这种随机变化是布局布线算法中固有的，所以不可能尝试所有 seed 以期获得绝对最佳结果。

注意: 任何直接或间接影响 Fitter 的设计变更都会都与更改 seed 值产生的随机影响类型相同。其中包括源文件中，**Compiler Settings** 或 **Timing Analyzer Settings** 中的任何更改。如果使用不同的计算机处理系统类型或不同的操作系统，都会出现类似影响，因为不同的系统会改变 Fitter 中计算浮点数的方式。

如果优化设置中的更改仅少许影响 register-to-register 时序或错误路径的数量，就无法确定是否因为更改而引起提高或降低，又或者是否因为 Fitter 中的随机影响而造成。如果您的设计仍在更改，请运行 seed sweep (通过多个 seed 编译设计) 确定优化更改后平均结果是否得到改善，以及从增加编译时间的设置中得到值得牺牲时间的理想结果，例如物理综合设置。该扫描还会显示设计中预期的随机变化量。

如果设计已定案，就可使用各个 seed 进行编译获得最佳结果。然而，如果随后对设计进行任何更改，就可能需要再次执行 seed 扫描。

点击 **Assignments > Compiler Settings** 通过 seed 控制初始布局。可使用 DSE II 轻松执行 seed 扫描。

使用如下 Tcl 命令指定 Fitter seed:

```
set_global_assignment -name SEED <value>
```

[相关链接](#)

[Design Space Explorer II](#) (第 12 页)

3.5.5.13. 将 Router Timing Optimization 设置为 Maximum

当布线程序不采用最佳布线线缆时，为改善设计中的布线性，可将 **Router Timing Optimization Level** 设置为 **Maximum**。该设置将确定布线程序尝试满足时序要求的积极程度。将该选项设置为 **Maximum** 可略微提高设计速度但会以增加编译时间为代价。该选项设置为 **Minimum** 可减少编译时间但代价是设计速度略有降低。默认值为 **Normal**。

[相关链接](#)

[路由器时序优化水平逻辑选项](#)

Intel Quartus Prime Help 中

3.5.6. Logic Lock (Standard) Assignments

Using Logic Lock (Standard) assignments to improve timing performance is only recommended for older devices, such as the MAX II family. For other device families, especially for larger devices such as Arria and Stratix series devices, do not use Logic Lock (Standard) assignments to improve timing performance. For these devices, use the feature for performance preservation and to floorplan your design.

Logic Lock (Standard) assignments do not always improve the performance of the design. In many cases, you cannot improve upon results from the Fitter by making location assignments. If there are existing Logic Lock (Standard) assignments in your design, remove the assignments if your design methodology permits it. Recompile the design, and then check if the assignments are making the performance worse.

When making Logic Lock (Standard) assignments, it is important to consider how much flexibility to give the Fitter. Logic Lock (Standard) assignments provide more flexibility than hard location assignments. Assignments that are more flexible require higher Fitter effort, but reduce the chance of design overconstraint.

The following types of Logic Lock (Standard) assignments are available, listed in the order of decreasing flexibility:

- Auto size, floating location regions
- Fixed size, floating location regions
- Fixed size, locked location regions

If you are unsure about the best size or location of a Logic Lock (Standard) region, the **Auto/Floating** options are useful for your first pass. After you determine where a Logic Lock (Standard) region must go, modify the Fixed/Locked regions, as Auto/Floating Logic Lock (Standard) regions can hurt your overall performance. To determine what to put into a Logic Lock (Standard) region, refer to the timing analysis

results and analyze the critical paths in the Chip Planner. The register-to-register timing paths in the Timing Analyzer section of the Compilation Report help you recognize patterns.

相关链接

[分析和优化设计平面布局规划 \(第 93 页\)](#)

3.5.6.1. Hierarchy Assignments

For a design with the hierarchy shown in the figure, which has failing paths in the timing analysis results similar to those shown in the table, `mod_A` is probably a problem module. In this case, a good strategy to fix the failing paths is to place the `mod_A` hierarchy block in a Logic Lock (Standard) region so that all the nodes are closer together in the floorplan.

图 32. Design Hierarchy

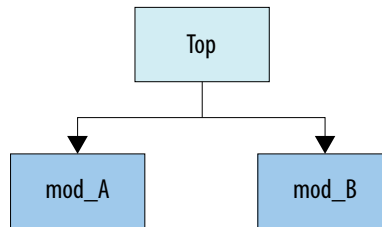


表 10. Failing Paths in a Module Listed in Timing Analysis

| From | To |
|------------|-------------|
| mod_A reg1 | mod_A reg9 |
| mod_A reg3 | mod_A reg5 |
| mod_A reg4 | mod_A reg6 |
| mod_A reg7 | mod_A reg10 |
| mod_A reg0 | mod_A reg2 |

Hierarchical Logic Lock (Standard) regions are also important if you are using an incremental compilation flow. Place each design partition for incremental compilation in a separate Logic Lock (Standard) region to reduce conflicts and ensure good results as the design develops. You can use the auto size and floating location regions to find a good design floorplan, but fix the size and placement to achieve the best results in future compilations.

相关链接

[分析和优化设计平面布局规划 \(第 93 页\)](#)

3.5.7. 位置约束

如果仅少数路径无法满足其时序要求，就可使用硬性位置约束优化布局。

位置约束关于 Intel Quartus Prime Fitter 的灵活性不如 Logic Lock (Standard) 约束好。此外，如果您熟悉自己的设计，就可通过进入位置约束的方式来得到更好的结果。

注意: 改善适配结果，尤其对于较大器件，例如 **Arria** 和 **Stratix** 系列器件，会有困难。位置约束并非总能提高设计性能。很多情况下，无法通过位置约束来改善 **Fitter** 的结果。

3.5.8. 亚稳定性分析和优化技术

当信号在不相关或异步时钟域的电路之间传输时，就会出现亚稳定性问题，因为设计人员无法保证信号满足其建立和保持时间要求。故障间隔时间（**MTBF**）是实例之间平均时间的估计值，也是可能导致设计失败的亚稳定状态。

当设计在综合一部信号以优化设计提高 **MTBF** 时，可使用 **Intel Quartus Prime** 软件分析因亚稳定性导致的平均 **MTBF**。这些亚稳态功能仅支持由 **Timing Analyzer** 约束设计以及特定器件系列。

如果您设计中的 **MTBF** 较低，请参阅 **Timing Optimization Advisor** 中的 **Metastability Optimization** 部分，其中建议了各种设置，以亚稳态为依据帮助优化您的设计。

本章节说明如何使能亚稳态分析并识别设计中的寄存器同步链，提供亚稳态报告的详细信息，以及提供管理亚稳态的其他直到原则。

相关链接

- 了解 [FPGA 中的亚稳定性](#)
- 使用 [Intel Quartus Prime Software](#) 管理亚稳态
在 *Intel Quartus Prime Standard Edition User Guide: Design Recommendations* 中

3.6. 外设到内核寄存器布局和布线优化（P2C）

Periphery to Core Register Placement and Routing Optimization (P2C) 选项指定 **Fitter** 是否对外设逻辑和 **FPGA** 内核中寄存器之间的直接连接上执行集成目标性布局和布线优化。**P2C** 是一个可选的预布线感知布局优化阶段，使您能够更可靠实现时序收敛。

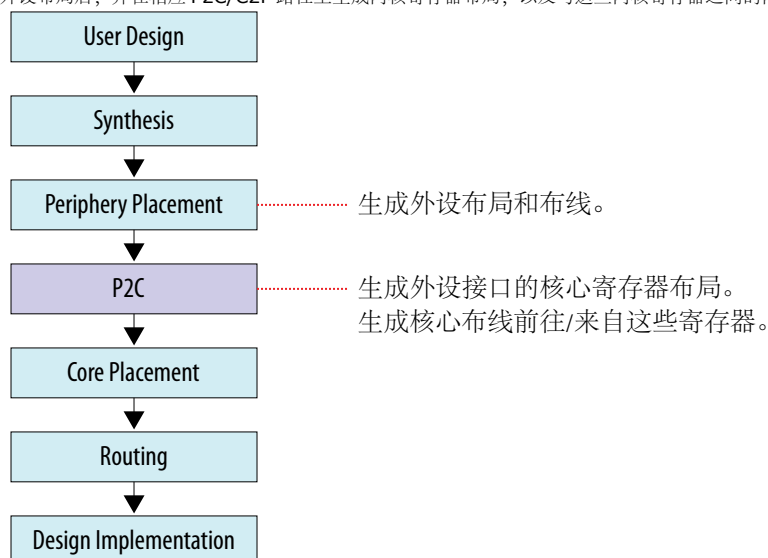
注意: **Periphery to Core Register Placement and Routing Optimization** 选项在两个方向都适用：外设到内核，以及内核到外设。

外部接口（如，高速 **I/O** 或串行接口）和 **FPGA** 之间的传输通常需要布线多个严格设置和保持时间要求的连接。开启该选项时，**Fitter** 先执行 **P2C** 布局和布线决定，然后才是内核布局和布线。这样就能保留必需资源确保设置实现其时序要求，并避免与外部接口进行传输时出现布线拥塞。

该选项可用作全局约束，或可应用于设计中的特定实例。

图 33. 外设到内核寄存器布局和布线优化 (P2C) 流程

P2C 运行于外设布局后，并在相应 P2C/C2P 路径上生成内核寄存器布局，以及与这些内核寄存器之间的内核布线。



在 [Advanced Fitter Setting](#) 对话框中设置外设到内核优化 (第 75 页)

在 [Assignment Editor](#) 中设置外设到内核优化 (第 75 页)

在 [Fitter Report](#) 中查看外设到内核优化 (第 76 页)

3.6.1. 在 [Advanced Fitter Setting](#) 对话框中设置外设到内核优化

Periphery to Core Placement and Routing Optimization 设置指定 Fitter 是否优化 FPGA 内核中外设逻辑和寄存器之间直接连接上的目标布局和布线。

可使用 Assignment Editor 中的设置对实例选择性执行通过外设到内核优化。

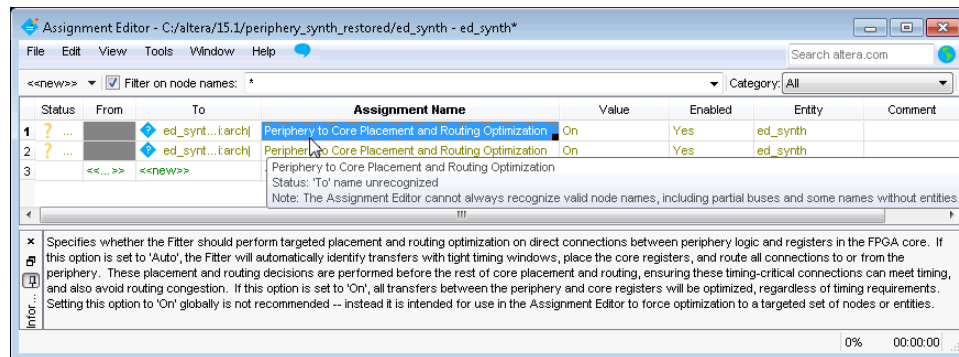
1. 在 Intel Quartus Prime 软件中，点击 **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**。
2. 在 **Advanced Fitter Settings** 对话框中，对于 **Periphery to Core Placement and Routing Optimization** 选项，根据您希望设计中外围对内核优化的方式选择如下选项：
 - a. 选择 **Auto** 可指示软件自动识别具有严格时序窗口的传输，放置内核寄存器，并布线所有与外设间的往来连接。
 - b. 选择 **On** 可指示软件全局优化外设和内核寄存器之间的全部传输，且不考虑时序要求。
注意： 不建议将 **Advanced Fitter Settings** 中的该选项设置为 **On**。该设置旨在通过 Assignment Editor 强制优化目标节点和实例集。
 - c. 选择 **Off** 禁用设计中外设到内核路径优化。

3.6.2. 在 [Assignment Editor](#) 中设置外设到内核优化

在 Assignment Editor 中开启 **Periphery to Core Placement and Routing Optimization (P2C/C2P)** 设置时，Intel Quartus Prime 软件对设计中的选定实例执行外设到内核，或内核到外设优化。

可使用 **Advanced Fitter Settings** 对话框中的设置对实例选择性执行外设到内核优化。

1. 在 Intel Quartus Prime 软件中，点击 **Assignments > Assignment Editor**。
2. 对于已选路径，双击 **Assignment Name** 栏，在下拉菜单中点击 **Periphery to core register placement and routing optimization** 选项。
3. 在 **To** 栏，选择 P2C/C2P 路径上需要优化的外设节点或内核寄存器。将 **From** 栏留空。对于 Assignments Editor 中出现的路径，必须首先在设计上运行 Analysis & Synthesis 。



3.6.3. 在 Fitter Report 中查看外设到内核优化

编译后，Intel Quartus Prime 软件通过 **Fitter (Place & Route)** 报告生成外设到内核优化和布线优化总结。

1. 编译您的 Intel Quartus Prime 工程。
2. 在 **Tasks** 窗格中，选择 **Compilation**。
3. **Fitter (Place & Route)** 下，双击 **View Report**。
4. **Fitter** 文件夹中，展开 **Place Stage** 文件夹。
5. 双击 **Periphery to Core Transfer Optimization Summary**。

表 11. Fitter Report 一外设到内核传输优化 (P2C) 摘要

| 出发路径 | 路径 | 状态 |
|--------|--------|---|
| Node 1 | Node 2 | Placed and Routed —内核寄存器已锁定。外设到内核/内核到外设布线已提交。 |
| Node 3 | Node 4 | Placed but not Routed —内核寄存器已锁定。布线未提交。该情况出现在 P2C 未使能无法优化单个组内的所有目标路径时，例如，相同的延迟/线缆要求，或相同的控制信号。部分 P2C 布线提交可能导致无法解决的布线拥塞。 |
| Node 5 | Node 6 | Not Optimized —改状态出现在 P2C 设置为 Auto 时且由于如下问题路径未得到优化： <ol style="list-style-type: none"> a. 不可能实现延迟要求。 b. 最小延迟要求（保持时间）过大。当需要添加多个线缆以满足保持时间时，P2C 算法不能有效处理。 c. P2C 遇到无法解决的特定路径布线拥塞。 |

| Periphery to Core Transfer Optimization Summary | | | |
|---|---|---|-----------------------|
| | From | To | Status |
| 1 | Periphery to Core Transfer | | |
| 2 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[1]lane_gen[3]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[1]lane_gen[3]lane_inst | Placed but Not Routed |
| 3 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[2]lane_gen[2]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[2]lane_gen[2]lane_inst | Placed but Not Routed |
| 4 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[1]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[1]lane_inst | Placed but Not Routed |
| 5 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | Placed but Not Routed |
| 6 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | Placed but Not Routed |
| 7 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | Placed but Not Routed |
| 8 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | Placed but Not Routed |
| 9 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | Placed but Not Routed |
| 10 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[0]lane_inst | Placed but Not Routed |
| 11 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[2]lane_gen[3]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[2]lane_gen[3]lane_inst | Placed but Not Routed |
| 12 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[2]lane_gen[3]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[2]lane_gen[3]lane_inst | Placed but Not Routed |
| 13 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[1]lane_gen[3]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[1]lane_gen[3]lane_inst | Placed but Not Routed |
| 14 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[3]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[3]lane_inst | Placed but Not Routed |
| 15 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[3]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[3]lane_inst | Placed but Not Routed |
| 16 | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[2]lane_inst | dutlarchlarch_instlto_tiles_wrap_instlto_tiles_instlto_gen[0]lane_gen[2]lane_inst | Placed but Not Routed |

3.7. 脚本支持

可运行本手册中 Tcl 脚本说明的过程并进行设置。还可按照提示命令指示运行。关于脚本选项的详细信息，请参阅 [Intel Quartus Prime 命令行](#) 和 [Tcl API Help 浏览器](#)。运行“Help”浏览器，可在提示命令键入如下命令：

```
quartus_sh --qhelp
```

可在实例或/和全局级中指定本小节介绍的多个选项。

使用以下 Tcl 命令进行全局约束：

```
set_global_assignment -name <.qsf variable name> <value>
```

使用以下 Tcl 命令进行实例约束：

```
set_instance_assignment -name <.qsf variable name> <value> -to <instance name>
```

注意：如果<value>字段包含空格（例如，‘Standard Fit’），则必须以英文双引号附上其值。

相关链接

- [Tcl Scripting](#) (第 0 页)
- [Intel Quartus Prime Standard Edition 设置文件参考手册](#)
关于 Intel Quartus Prime 软件中所有设置和约束的信息。
- [Tcl Scripting](#)
Intel Quartus Prime Standard Edition 用户指南：脚本
- [Command Line Scripting](#)
Intel Quartus Prime Standard Edition 用户指南：脚本

3.7.1. 初始编译设置

在 Tcl 约束中使用 Intel Quartus Prime Settings File (.qsf) 变量名以及正确值进行设置。**Type** 栏标示设置是否用做全局设置，实例设置，或两者兼顾。

上方表格罗列 .qsf 变量名称以及 [设计优化概述](#) 章节的 **初始编译：必要设置** 部分中介绍的可用于设置的值。下方表格列出了各高级编译设置。

表 12. 初始编译设置

| 设置名称 | .qsf File Variable Name | 值 | 类型 |
|-------------------|--|--|----|
| 优化针对时序的 IOC 寄存器布局 | OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING | ON, OFF | 全局 |
| 优化保持时间 | OPTIMIZE_HOLD_TIMING | OFF, IO PATHS AND MINIMUM TPD PATHS, ALL PATHS | 全局 |

表 13. 高级编译设置

| 设置名称 | .qsf 文件变量名称 | 值 | 类型 |
|------------|----------------------------------|--------------------------|----|
| 布线程序时序优化级别 | ROUTER_TIMING_OPTIMIZATION_LEVEL | NORMAL, MINIMUM, MAXIMUM | 全局 |

[相关链接](#)[设计优化概述 \(第 7 页\)](#)

3.7.2. I/O 时序优化技术

本表格列出 .qsf 文件变量名称以及可用于 I/O 时序优化设置的值。

表 14. I/O 时序优化设置

| 设置名称 | .qsf 文件变量名称 | 值 | 类型 |
|---|--|---------|----------|
| Optimize IOC Register Placement For Timing (针对时序优化 IOC 寄存器布局) | OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING | ON, OFF | Global |
| Fast Input Register | FAST_INPUT_REGISTER | ON, OFF | Instance |
| Fast Output Register | FAST_OUTPUT_REGISTER | ON, OFF | Instance |
| Fast Output Enable Register | FAST_OUTPUT_ENABLE_REGISTER | ON, OFF | Instance |
| Fast OCT Register | FAST_OCT_REGISTER | ON, OFF | Instance |

3.7.3. Register-to-Register 时序优化技术

本表格列出 .qsf 文件变量名称以及可用于 *Register-to-Register* 时序优化技术中介绍的设置的值。

表 15. Register-to-Register 时序优化技术

| 设置名称 | .qsf 文件变量名称 | 值 | 类型 |
|-----------------------------------|---|---------|--------|
| 执行 WYSIWYG 原语再综合 | ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP | ON, OFF | 全局, 实例 |
| 执行组合逻辑的物理综合 (无 Intel Arria 10 支持) | PHYSICAL_SYNTHESIS_COMBO_LOGIC | ON, OFF | 全局, 实例 |
| 执行寄存器复制 (无 Intel Arria 10 支持) | PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION | ON, OFF | 全局, 实例 |
| 执行寄存器重定时 (无 Intel Arria 10 支持) | PHYSICAL_SYNTHESIS_REGISTER_RETIMING | ON, OFF | 全局, 实例 |
| 继续... | | | |

| 设置名称 | .qsf 文件变量名称 | 值 | 类型 |
|--------------------------------------|---|-----------------------------|--------|
| 执行自动异步信号流水行 (无 Intel Arria 10 支持) | PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING | ON, OFF | 全局, 实例 |
| 物理综合效力 (无 Intel Arria 10 支持) | PHYSICAL_SYNTHESIS_EFFORT | NORMAL, EXTRA, FAST | 全局 |
| Fitter Seed | SEED | <integer> | 全局 |
| 最大扇出 | MAX_FANOUT | <integer> | 实例 |
| 手动逻辑复 | DUPLICATE_ATOM | <node name> | 实例 |
| 综合器件优化功率 | OPTIMIZE_POWER_DURING_SYNTHESIS | NORMAL, OFF EXTRA_EFFORT | 全局 |
| 适配期间优化功率 | OPTIMIZE_POWER_DURING_FITTING | NORMAL, OFF EXTRA_EFFORT | 全局 |

相关链接

[Register-to-Register 时序优化技术 \(第 67 页\)](#)

3.8. 时序收敛和优化修订历史

本章修订历史如下:

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|------------|------------------------|--|
| 2018.11.12 | 18.1.0 | <ul style="list-style-type: none"> 更新了“调整布局工作量”主题中的“布局工作量乘法器”图示和文字说明。 更新了“调整布局工作量”主题中的“布局工作量乘法器”图示和文字说明。fb 609697 更新了“为保持时间添加的线路”主题中的“优化保持时间选项”的截屏。 |
| 2018.09.24 | 18.1.0 | <ul style="list-style-type: none"> Intel Quartus Prime Standard 版用户指南首次发布。 删除了重复主题: 资源利用率优化技术。该主题现位于 区域优化章节。 |
| 2017.11.06 | 17.1.0 | <ul style="list-style-type: none"> 移动主题位置: 将“时序收敛的设计评估”移动到“初始编译: 可选的 Fitter 设置”之后。 更新了关于资源利用率优化设置的逻辑选项。 |
| 2017.05.08 | 17.0.0 | <ul style="list-style-type: none"> 添加了主题: 关键路径。 更新了 寄存器-寄存器时序并重命名 寄存器-寄存器时序分析。 重命名主题: 将使用 Timing Analyzer 的时序分析重命名为使用 Timing Analyzer 显示路径报告。 删除主题中 (基于 LUT 器件) 的备注。 重命名主题: 将优化时序 (基于 LUT 器件) 重命名为时序优化。 重命名主题: 将在 Timing Analyzer 中调试时序错误重命名为显示错误路径的时序收敛建议。 重命名主题: 将改善 Register-to-Register 时序摘要重命名为改善 Register-to-Register 时序。 |
| 2016.05.02 | 16.0.0 | <ul style="list-style-type: none"> 阐述关于已淘汰的物理综合选项的限制。 添加了关于监控集群困难度的信息。 |
| 2015.11.02 | 15.1.0 | <ul style="list-style-type: none"> 添加了: 外设到内核寄存器布局及布线优化。 将 Quartus II 实例更改为 Quartus Prime。 |
| 继续... | | |

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|---------------|------------------------|---|
| 2014.12.15 | 14.1.0 | <ul style="list-style-type: none"> 将 Fitter Settings、Analysis & Synthesis Settings 和 Physical Synthesis Optimizations 的位置更改为 Compiler Settings 更新了 DSE II 的内容。 |
| June 2014 | 14.0.0 | <ul style="list-style-type: none"> DITA 转换。 删除了 QII 软件 v14.0 中不支持的废弃器件：Arria GX, Arria II, Cyclone III, Stratix II, Stratix III。 以 IP 核内容替代 Megafunction 内容。 |
| November 2013 | 13.1.0 | <ul style="list-style-type: none"> 为时序收敛部分添加设计评估。 删除了优化时序（基于 Macrocell 的 CPLD）部分。 更新了优化 Multi-Corner 时序和 Fitter 主动可布线性优化。 更新了通过 Timing Analyzer 进行时序分析以显示如何访问 Report All Summaries 命令。 更新了忽略时序收敛，包含了关于 <i>Fitter Summary Reports</i> 的帮助链接提供 Ignored Assignment Report 信息。 |
| May 2013 | 13.0.0 | <ul style="list-style-type: none"> 将标题“面积和时序优化”重命名为“时序收敛和优化”。 删除了设计和面积/资源优化信息。 添加了如下部分： Fitter 主动可布线性优化。 关于分析往来关键路径的源和目标路径的提示。 定位 Chip Planner 的多个路径的提示。 创建 .tcl 脚本以监控跨编译的关键路径的提示。 |
| November 2012 | 12.1.0 | <ul style="list-style-type: none"> 更新了“初始编译：可选的 Fitter 设置”，第 13 – 2 页；“I/O 约束”，第 13 – 2 页；“初始编译：可选的 Fitter 设置”，第 13 – 2 页；“资源利用率”，第 13 – 9 页；“布线”，第 13 – 21 页，以及“解决资源利用率问题”，第 13 – 43 页。 |
| June 2012 | 12.0.0 | <ul style="list-style-type: none"> 更新了“优化 Multi-Corner 时序”，第 13 – 6 页；“资源利用率”，第 13 – 10 页；“使用 Timing Analyzer 进行时序分析”，第 13 – 12 页；“使用 Resource Optimization Advisor”，第 13 – 15 页；“增加 Placement Effort Multiplier”，第 13 – 22 页；“增加 Router Effort Multiplier”，第 13 – 22 页；和“在 Timing Analyzer 中调试时序失败”，第 13 – 24 页。 本章中的少量文本编辑。 |
| November 2011 | 11.1.0 | <ul style="list-style-type: none"> 更新了“时序要求设置”，“标准 Fit”，“快速 Fit”，“优化 Multi-Corner 时序”，“使用 Timing Analyzer 进行时序分许”，“Timing Analyzer 中调试时序错误”，“LogicLock 约束”，“跨时钟域分析错误时钟的提示”，“综合期间展开层次”，“快速输入，输出和输出使能寄存器”和“层次约束”部分 更新了表 13-6。 添加了“脊柱时钟限制”部分 从第 19 页删除了更改状态机编码部分 删除了图 13-5 本章中的少量文本编辑。 |
| May 2011 | 11.0.0 | <ul style="list-style-type: none"> 重新组织“初始编译：可选 Fitter 设置”部分的内容 在“资源利用率”部分添加了新信息。 在“复制扇出控制逻辑”部分添加了新信息。 添加了 Help 链接 本章中的其他编辑和更新 |

继续...

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|---------------|------------------------|---|
| December 2010 | 10.1.0 | <ul style="list-style-type: none"> • 添加了 Help 链接 • 更新了器件支持 • 添加了“Timing Analyzer 中调试时序错误”部分 • 删除了 Classic Timing Analyzer 参考 • 本章中起其他更新 |
| August 2010 | 10.0.1 | 更正了链接 |
| July 2010 | 10.0.0 | <ul style="list-style-type: none"> • 将编译时间优化技术部分移动到新的 <i>缩短编译时间</i> 章节 • 删除了关于时序收敛布局规划的参考 • 将智能编译设置和早期时序估算部分移动到新的 <i>缩短编译时间</i> 章节 • 添加了其他优化资源部分 • 删除了过时信息。 • 将 DSE 章节的参考更改为 Help 链接 • 在适当的地方添加了 Help 链接 • 删除“参考文档”部分。 |

相关链接

文档存档

欲了解 *Intel Quartus Prime* 手册以前的版本，请搜索文档存档。

4. 区域优化

本章说明设计 Intel 器件时减少资源利用率的技术。

4.1. 资源使用情况

无论设计是否完成适配，确定资源利用率都可提供有用信息。如果编译结果中无适配错误，则资源利用率信息有利于分析设计中的适配问题。如果适配成功，该信息允许您确定设计更改会引入适配困难。此外，还可确定资源利用率对时序性能的影响。

Compilation Report 提供资源使用情况的信息。

4.1.1. Flow Summary 报告

编译报告中的 **Flow Summary** 部分指示设计是否超出可用器件资源，并报告资源使用率，包括管脚，存储器位，数字信号处理（DSP）块和锁相环（PLL）。

Fitter 可将逻辑扩展至整个器件，但可能会提高整体资源使用率。

器件填满时，Fitter 自动搜索逻辑功能并通过公共输入放置到 ALM 中。被打包的寄存器数量也会增加。因此，如果逻辑和寄存器可更紧密封装在一起，则具有较高整体利用的设计可能仍然具有用于额外逻辑的空间。这样，具有更多详细信息的报告就更有意义。

4.1.2. Fitter 报告

在编译报告的 **Fitter** 部分，**Resource Section** 下的报告提供详细的资源信息。

Fitter Resource Usage Summary 报告分解逻辑资源使用信息并提供其他资源信息，包括每个存储器块类型中的位数。该窗格还包含全局时钟，PLL，DSP 块和其他特定器件资源的使用情况摘要。

[相关链接](#)

[Fitter 资源报告](#)

4.1.3. Analysis 和 Synthesis 报告

对于由 Intel Quartus Prime 综合引擎进行综合的设计，可在编译期间查看说明优化的报告。

例如，在 **Analysis & Synthesis** 部分，**Optimization Results** 文件夹中，可找到综合期间被删除的寄存器列表。通过该报告可估算设计部分中的资源使用率，从而确保不因丢失与设计其中其他部分中的连接而将寄存器删除。

[相关链接](#)

[综合优化结果报告](#)

4.1.4. 编译消息

如果报告显示布线资源使用低于 100%，但设计仍未适配，则要么布线资源不足，或者设计包含无效约束。任何一种情况下，Compiler 在 **Messages** 窗口的 **Processing** 选项卡下生成说明问题的消息。

如果 Fitter 未成功完成并且运行速度比类似设计上快得多，则资源可能被过度使用或可能存在非法约束。

如果与类似设计相比，Intel Quartus Prime 软件运行时间过长，则很可能 Compiler 无法找到有效布局或布线。Compilation Report 中，查看标示这些问题类型的错误和警告。

Chip Planner 有助于查找器件中特定类型布线资源中布线堵塞的区域。如果发现具有非常高堵塞的区域，则请分析导致堵塞的原因。诸如不使用全局资源的高扇出网络，错误选择优化目标（速度与面积），非常严格的布局规划约束，或编码风格都会导致布线堵塞。确定原因后，修改源或设置以降低布线堵塞。

相关链接

[查看消息](#)

4.2. 优化资源利用率

以下列出设计分析后的阶段：

1. 优化资源使用—确保已经设置基本约束
2. I/O 时序优化—优化资源使用和所需目标器件中的设计适配后，再优化 I/O 时序。
3. Register-to-register 时序优化

相关链接

- [设计优化概述](#) (第 7 页)
- [时序收敛与优化](#) (第 42 页)

4.2.1. Using the Resource Optimization Advisor

The Resource Optimization Advisor provides guidance in determining settings that optimize resource usage. To run the Resource Optimization Advisor click **Tools** > **Advisors** > **Resource Optimization Advisor**.

The Resource Optimization Advisor provides step-by-step advice about how to optimize resource usage (logic element, memory block, DSP block, I/O, and routing) of your design. Some of the recommendations in these categories might conflict with each other. Intel recommends evaluating the options and choosing the settings that best suit your requirements.

相关链接

[Resource Optimization Advisor 命令工具菜单](#)

4.2.2. 资源使用问题概述

资源使用问题可分为三大类：

- 与 I/O 管脚使用率或布局有关的问题，包括专用 I/O 块，例如 PLL 或 LVDS 收发器。
- 与逻辑使用率或布局有关的问题，包括具有寄存器的逻辑单元和 LUT 以及专用逻辑，如存储器块和 DSP 块。
- 与布线有关的问题。

4.2.3. I/O 管脚使用情况或布局

按照指导解决 I/O 资源问题。

4.2.3.1. Guideline: Use I/O Assignment Analysis

To help with pin placement, click **Processing > Start > Start I/O Assignment Analysis**. The **Start I/O Assignment Analysis** command allows you to check your I/O assignments early in the design process. You can use this command to check the legality of pin assignments before, during, or after compilation of your design. If design files are available, you can use this command to accomplish more thorough legality checks on your design's I/O pins and surrounding logic. These checks include proper reference voltage pin usage, valid pin location assignments, and acceptable mixed I/O standards.

Common issues with I/O placement relate to the fact that differential standards have specific pin pairings and certain I/O standards might be supported only on certain I/O banks.

If your compilation or I/O assignment analysis results in specific errors relating to I/O pins, follow the recommendations in the error message. Right-click the message in the **Messages** window and click **Help** to open the Intel Quartus Prime Help topic for this message.

4.2.3.2. 指导：修改管脚约束或选择较大的封装

如果设计中因管脚约束失败而无法适配，则请在无管脚约束的情况下编译设计，以确定该设计是否可能用于指定器件和封装。如果有 Intel Quartus Prime 错误消息显示因管脚约束的适配问题就可使用这种方法。

如果忽略所有管脚约束，或忽略和删除多个管脚约束后，设计得以适配，则可能必须修改设计管脚约束或选择较大的封装。

如果由于 I/O 管脚不足而导致设计无法适配，则具有较多可用用户 I/O 管脚的较大器件封装（可以是相同器件密度）能实现成功适配。

相关链接

管理器件 I/O 管脚

>In *Intel Quartus Prime Standard Edition User Guide: Design Constraints*

4.2.4. 逻辑资源使用或布局

解决逻辑资源使用问题，包括具有寄存器的逻辑单元和 LUT 以及专用逻辑，如存储器块和 DSP 块。

4.2.4.1. 指导：优化源代码

如果设计因逻辑使用率而无法适配，则对源进行评估并修改设计。对源代码进行特定于设计的更改，通常可显著改善逻辑。也是最有效改善结果质量的常用办法。

如果您的设计无法适配到可用逻辑单元（LE）或 ALM，但仍有未使用的存储器或 DSP 块，请检查设计中是否存在说明存储器或 DSP 功能的代码块未被推断且未置于专用逻辑中。可修改源代码，以允许将专用存储器或 DSP 资源置于目标器件中。

请确保您的状态机被识别为状态逻辑且在综合工具中被适当优化。已识别出的状态机通常会被优化，否则被当作一般逻辑处理。在 Intel Quartus Prime 软件中，可从 **Compilation Report** 的 **Analysis & Synthesis** 下查看 **State Machine** 报告。该报告提供包括编译期间识别出的每个状态机的状态编码等各种详细信息。如果无法识别您的状态机，则可能必须更改源代码才能识别。

相关链接

- [AN 584: 高级 FPGA 设计的时序收敛方法](#)
- [建议的 HDL 编码样式](#)
在 *Intel Quartus Prime Standard Edition User Guide: Design Recommendations* 中

4.2.4.2. 指导：优化针对区域而非速度的综合

如果由于逻辑资源限制，导致 Fitter 无法解析设计，则重新综合设计以提高区域利用率。

首先，确保在综合工具中正确设置器件和时序约束。尤其是在考虑设计中区域利用率时，请务必不要过度约束设计的时序要求。综合工具在满足指定要求时，如果约束过于主动，则可能导致更高的器件资源使用率。

如果资源利用率需重点考虑，则可优化区域而非速度。

- 如果使用 Intel Quartus Prime integrated 综合，单击 **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)** 并将 **Optimization Technique** 选择为 **Balanced** 或 **Area**。
- 如果要使用 **Area** 或 **Speed** 设置减少设计中指定模块的面积，同时保留默认 **Optimization Technique** 设置为 **Balanced**，则请使用 Assignment Editor。
- 还可打开 **Speed Optimization Technique for Clock Domains** 逻辑选项优化指定时钟域之内或之间的所有组合逻辑的速度。
- 某些综合工具中，不指定 f_{MAX} 要求可导致低资源利用率。

优化面积或速度会影响 register-to-register 时序性能。

注意:

Intel Quartus Prime 软件中，**Balanced** 设置通常会与 **Area** 设置非常相似的资源利用率结果。**Area** 设置在某些情况下可提供更好的结果。

Intel Quartus Prime 软件提供其他属性和选项，有助于提高综合结果的质量。

相关链接

[Intel Quartus Prime 集成综合](#)

4.2.4.3. 指导：重组多路复用器

多路复用器在许多 FPGA 设计占逻辑利用率的较大部分。通过优化多路复用逻辑，可在 Intel 器件中达成更有效实现。

相关链接

- [重组多路复用器逻辑选项](#)
了解关于 Restructure Multiplexers 选项的更多信息。
- [建议的 HDL 编码样式](#)
实现多路复用器设计最佳资源使用的设计指导

4.2.4.4. 指导：通过 **Balanced** 或 **Area** 设置执行 **WYSIWYG** 原语重新综合

Perform WYSIWYG Primitive Resynthesis 逻辑选项指定综合期间是否执行 WYSIWYG 原语重新综合。该选项使用 **Optimization Technique** 逻辑选项中的指定设置。**Perform WYSIWYG Primitive Resynthesis** 逻辑选项有助于重新综合设计中的一些或全部 WYSIWYG 原语，以获得更适合的面积或性能。然而，仅在使用第三方综合工具时才可进行 WYSIWYG 原语再综合。

注意： **Balanced** 设置通常生成与 **Area** 设置非常相似的利用率结果，以获得更好的性能结果。**Area** 设置在某些情况下可带来更好结果。这种方式执行 WYSIWYG 再综合通常会降低 register-to-register 时序性能。

相关链接

[执行 WYSIWYG 原语再综合逻辑选项](#)
关于该逻辑选项的信息

4.2.4.5. 指导：使用寄存器封装

Auto Packed Registers 选项通过将仅使用寄存器和仅使用 LUT 的单元中的寄存器和 LUT 合并，进而以一个逻辑单元实现两个单元的功能。

相关链接

[自动封装寄存器逻辑选项](#)
了解关于 Auto Packed Register 逻辑选项的更多信息。

4.2.4.6. 指导：删除 **Fitter** 约束

设计中存在约束冲突或难以满足甚至不适合目标器件的约束。例如，如果位置或 Logic Lock (Standard) 约束过于严格且器件中无足够的布线资源可用，则设计可能适配失败。

要解决限制性位置约束或 Logic Lock (Standard) 区域约束导致的布线拥塞，请使用 Chip Planner 中的 **Routing Congestion** 任务解决布局规划中的布线问题，然后从该区域中删除内部位置或 Logic Lock (Standard) 区域约束。如果设计仍无法适配，则设计被过度约束。为解决该问题，可删除所有位置和 Logic Lock (Standard) 约束并运行连续编译，每次编译之前逐步约束设计。可从 Assignment Editor 或 Chip Planner 中删除特定位置约束。要从 Chip Planner, Logic Lock (Standard) Regions Window 或 Assignments 菜单中删除 Logic Lock (Standard) 约束，请点击 **Remove Assignments**。在 **Available assignment categories** 列表中打开要从设计中删除的约束类别。

相关链接

[分析和优化设计平面布局规划 \(第 93 页\)](#)

4.2.4.7. 指导：综合期间展开层级结构

综合工具通常提供保留层级结构边界的选项，有助于验证或其他目的。然而，Intel Quartus Prime 软件跨层级边界优化，以便执行最大程度的逻辑最小化，从而减少设计中无设计分区的区域。

If you are using Intel Quartus Prime incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions, and the synthesis does not flatten it across partitions. Follow Intel's recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.

4.2.4.8. 指导：更换目标存储器块

如果由于存储器资源现在导致 Fitter 无法解析设计，则设计可能需要器件中没有的存储器类型。

对于使用 Parameter Editor 创建的存储器块，编辑 RAM 块类型以针对另一尺寸的存储器块。

Compiler 还可从 HDL 代码中推断 ROM 和 RAM 存储器块，且综合引擎可通过推断 Shift 寄存器（基于 RAM）IP 核来讲大型移位寄存器放入存储器块中。关闭综合工具中的此推断功能时，综合引擎将存储器或移位寄存器放入逻辑而非存储器块。此外，关闭高推断可防止寄存器被移动到 RAM 中，从而提高时序性能。

根据综合工具，也可为已推断的存储器块设置 RAM 块类型。Intel Quartus Prime 综合中，设置 **ramstyle** 属性以说明已推断 RAM 块的存储器类型。或者，将该选项设置为 **logic** 以标准逻辑实现存储器块。

参考报告文件中 Entity 报告的 Resource Utilization 内容以决定是否任何模块中存在异常高的寄存器计数。由于块的体系结构实现，一些编码样式会阻止 Intel Quartus Prime 软件从源代码推断 RAM 块，并强制软件在触发器中实现逻辑。例如，寄存器 bank 上的异步同步可能使寄存器组与器件体系结构中的 RAM 块不兼容，因此 Compiler 在触发器中实现寄存器 bank。稍许修改相关逻辑，就可将大型寄存器 bank 移动到 RAM 中。

相关链接

- [以 HDL 代码推断移位寄存器](#)
- [通过 Entity Report 报告的 Fitter 资源利用率](#)

4.2.4.9. 指导：使用物理综合选项减少区域

Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) 中可用的物理综合选项可帮助减少资源使用。使能物理综合时，Intel Quartus Prime 软件对网表进行特定布局更改，以降低指定 Intel 器件中的资源使用率。

注意：物理综合会增加编译时间。要减少对编译时间的影响，可对指定实例应用物理综合选项。

相关链接

[高级 Fitter 设置对话框](#)

4.2.4.10. 指导：平衡或更换目标 DSP 块

如果设计需要过多 DSP 块，则可能无法成功适配。可使用逻辑单元实现所有 DSP 块功能，因而可更换逻辑的目标 DSP 块以完成适配。

如果通过参数编辑器创建 DSP 功能，请打开参数编辑器并编辑该功能，使其以逻辑单元为目标对象而非 DSP 块。Intel Quartus Prime 软件使用 DEDICATED_MULTIPLIER_CIRCUITRY IP 核参数控制该实现。

还可通过乘法器，乘法加法器和乘法累加器的 HDL 代码推断 DSP 块。可在综合工具中关闭此推断功能。使用 Intel Quartus Prime integrated 综合时，可关闭 **Auto DSP Block Replacement** 逻辑选项在整个工程中禁用推断功能。点击 **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)**。关闭 **Auto DSP Block Replacement**。或者，通过 Assignment Editor 针对指定块禁用该功能。

Intel Quartus Prime 软件还提供 **DSP Block Balancing** 逻辑选项，以在逻辑单元或不同 DSP 块模式中实现 DSP 块元件。默认 **Auto** 设置允许 DSP 块平衡设置以适当自动转换 DSP 块切片从而最小化面积并最大化设计速度。可对指定节点或实体使用其他设置，或基于工程内容，控制 Intel Quartus Prime 软件如何将 DSP 功能转换成逻辑单元和 DSP 块。使用 **Auto** 或 **Off** 以外的值覆盖 IP 核类型中使用的 DEDICATED_MULTIPLIER_CIRCUITRY 参数。

4.2.4.11. 指导：使用大型器件

如果由于缺乏布线资源而导致适配无法成功完成，则可能需要使用较大的器件。

4.2.5. 布线

按照指导解决布线资源问题。

4.2.5.1. 指导：将 **Auto Packed Register** 设置为 **Sparse** 或 **Sparse Auto**

Auto Packed Registers 选项可减少设计中 LE 或 ALM 的数量。可点击 **Assignment > Settings > Compiler Settings > Advanced Settings (Fitter)** 设置该选项。

相关链接

[自动打包寄存器逻辑选项](#)

4.2.5.2. 指导：将 **Fitter Aggressive Routability Optimizations** 设置为 **Always**

如果由于过多布线线缆使用率而导致设计无法适配，则 **Fitter Aggressive Routability Optimization** 选项可帮助解决。

如果布局布线时间之间存在显著不平衡（首次适配尝试期间），则可能是线缆利用率高造成。开启 **Fitter Aggressive Routability Optimizations** 选项可缩短编译时间。

平均而言，该选项可节省线缆使用率最高达 6%，但也会将性能降低最多达 4%，具体取决于器件。

相关链接

[Fitter 主动可布线性优化逻辑选项](#)

4.2.5.3. 指导：增加 **Router Effort Multiplier**

Router Effort Multiplier（布线程序效力倍增器）控制布线程序查找有效解决方案的速度。默认值为 1.0 且合法值必须大于 0。

- 高于 1 的数字有利于难以通过提高布线效力进行布线的设计。
- 接近 0 的数字（例如，0.1）可缩短布线程序运行时间，但通常也会略微降低布线质量。

实验证据表明，3.0 倍的倍增器可降低整体线缆使用率大约 2%。使用大于默认值的 Router Effort Multiplier 有益于超过五级逻辑且具有复杂数据路径的设计。然而，设计中的拥塞主要因为布局，但增加 Router Effort Multiplier 不一定减少拥塞。

注意: 任何大于 4 的 Router Effort Multiplier 值，每增加 1 都能提高 10% 使用率。例如，值 10 实际上是 4.6。

4.2.5.4. 指导：删除 Fitter 约束

设计中存在约束冲突或难以满足甚至不适合目标器件的约束。例如，如果位置或 Logic Lock (Standard) 约束过于严格且器件中无足够的布线资源可用，则设计可能适配失败。

要解决限制性位置约束或 Logic Lock (Standard) 区域约束导致的布线拥塞，请使用 Chip Planner 中的 **Routing Congestion** 任务解决布局规划中的布线问题，然后从该区域中删除内部位置或 Logic Lock (Standard) 区域约束。如果设计仍无法适配，则设计被过度约束。为解决该问题，可删除所有位置和 Logic Lock (Standard) 约束并运行连续编译，每次编译之前逐步约束设计。可从 Assignment Editor 或 Chip Planner 中删除特定位置约束。要从 Chip Planner, Logic Lock (Standard) Regions Window 或 Assignments 菜单中删除 Logic Lock (Standard) 约束，请点击 **Remove Assignments**。在 **Available assignment categories** 列表中打开要从设计中删除的约束类别。

相关链接

[分析和优化设计平面布局规划 \(第 93 页\)](#)

4.2.5.5. 指导：优化针对区域而非速度的综合

某些情况下，重新综合设计提高区域使用率还可改善设计的布通性。首先，确保综合工具中已正确设置器件和时序约束。请勿过度约束设计的时序要求，尤其是需考量设计中区域使用率时。通常综合工具会尝试满足指定要求，如果约束过激，则会导致较高器件资源使用率。

如果资源使用率是重要考量，则可优化区域而非速度。

- 如果使用 Intel Quartus Prime integrated 综合，单击 **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)** 并将 **Optimization Technique** 选择为 **Balanced** 或 **Area**。
- 如果要使用 **Area** 或 **Speed** 设置减少设计中指定模块的面积，同时保留默认 **Optimization Technique** 设置为 **Balanced**，则请使用 Assignment Editor。
- 还可使用 **Speed Optimization Technique for Clock Domains** 逻辑选项指定对特定时钟域内或之间的所有组合逻辑进行速度优化。
- 某些综合工具中，不指定 f_{MAX} 要求可导致较低资源利用率。

优化面积或速度会影响 register-to-register 时序性能。

注意: Intel Quartus Prime 软件中，**Balanced** 设置通常会产生与 **Area** 设置非常相似的资源利用率结果。**Area** 设置在某些情况下可提供更好的结果。

Intel Quartus Prime 软件提供其他属性和选项，以助于提高综合结果的质量。

相关链接

[Intel Quartus Prime 集成综合](#)

4.2.5.6. 指导：优化源代码

如果因为布线问题以及前述部分中介绍的方法未显著提高设计布通性而导致设计无法适配，请从源处修改设计以达到所需结果。通常对源代码进行设计相关的修改，可显著改善结果，例如复制逻辑或更改需要大量布线资源的块之间的连接。

4.2.5.7. 指导：使用大型器件

如果由于缺乏布线资源而导致适配无法成功完成，则可能需要使用较大的器件。

4.3. 脚本支持

可运行本章中 Tcl 脚本说明的过程并进行约束设置。还可按照提示命令指示运行。关于脚本选项的详细信息，请参阅 **Intel Quartus Prime 命令行和 Tcl API Help 浏览器**。运行“Help”浏览器，可在提示命令键入如下命令：

1. 运行 **Help** 浏览器，在命令提示中键入如下命令：

```
quartus_sh --qhhelp
```

可在实例中或/和全局级中指定本小节中介绍的多个选项。

2. 使用以下 Tcl 命令进行全局约束：

```
set_global_assignment -name <QSF variable name> <value>
```

3. 使用以下 Tcl 命令进行实例约束：

```
set_instance_assignment -name <QSF variable name> <value> \ -to <instance name>
```

注意：如果<value>字段包含空格（例如，‘Standard Fit’），则必须以英文双引号附上其值。

相关链接

- [Tcl Scripting](#)
Intel Quartus Prime Standard Edition 用户指南：脚本
- [Command Line Scripting](#)
Intel Quartus Prime Standard Edition 用户指南：脚本

4.3.1. 初始编译设置

使用 Tcl 约束中的 **Intel Quartus Prime Settings File (.qsf)** 变量名以及正确值进行设置。**Type** 栏标示设置是否用做全局设置，实例设置，或两者兼顾。

表 16. 高级编译设置

| 设置名称 | .qsf 文件类型名称 | 值 | 类型 |
|----------------------------------|----------------------------------|------------------------------|--------|
| Placement Effort Multiplier | PLACEMENT_EFFORT_MULTIPLIER | 任何正，非零值 | Global |
| Router Effort Multiplier | ROUTER_EFFORT_MULTIPLIER | 任何正，非零值 | Global |
| Router Timing Optimization level | ROUTER_TIMING_OPTIMIZATION_LEVEL | NORMAL, MINIMUM, MAXIMUM | Global |
| Final Placement Optimization | FINAL_PLACEMENT_OPTIMIZATION | ALWAYS, AUTOMATICALLY, NEVER | Global |

4.3.2. 资源使用优化技术

本表格列出 Resource Utilization Optimization 设置的 QSF 约束和适用值：

表 17. 资源使用优化设置

| 设置名称 | .qsf 文件类型名称 | 值 | 类型 |
|--|---|--|----------------------------|
| Auto Packed Registers ⁽¹⁾ | QII_AUTO_PACKED_REGISTERS | AUTO, OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, SPARSE, SPARSE AUTO | Global (全局), Instance (实例) |
| Perform WYSIWYG Primitive Resynthesis | ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP | ON, OFF | Global, Instance |
| Perform Physical Synthesis for Combinational Logic for Reducing Area (no Intel Arria 10 support) | PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA | ON, OFF | Global, Instance |
| Perform Physical Synthesis for Mapping Logic to Memory (no Intel Arria 10 support) | PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA | ON, OFF | Global, Instance |
| Optimization Technique | <device family name>_OPTIMIZATION_TECHNIQUE | AREA, SPEED, BALANCED | Global, Instance |
| Speed Optimization Technique for Clock Domains | SYNTH_CRITICAL_CLOCK | ON, OFF | Instance |
| State Machine Encoding | STATE_MACHINE_PROCESSING | AUTO, ONE-HOT, GRAY, JOHNSON, MINIMAL BITS, ONE-HOT, SEQUENTIAL, USER-ENCODE | Global, Instance |
| Auto RAM Replacement | AUTO_RAM_RECOGNITION | ON, OFF | Global, Instance |
| Auto ROM Replacement | AUTO_ROM_RECOGNITION | ON, OFF | Global, Instance |
| Auto Shift Register Replacement | AUTO_SHIFT_REGISTER_RECOGNITION | ON, OFF | Global, Instance |
| Auto Block Replacement | AUTO_DSP_RECOGNITION | ON, OFF | Global, Instance |
| Number of Processors for Parallel Compilation | NUM_PARALLEL_PROCESSORS | 包括 1 和 16 之间的整数, 或 ALL | Global |

4.4. 区域优化修订历史

以下修订历史适用于本章：

(1) 该设置的允许值取决于您选择的器件系列。

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|------------|------------------------|--|
| 2018.09.24 | 18.1.0 | <ul style="list-style-type: none"> Initial release in Intel Quartus Prime Standard Edition User Guide. 主题划分：资源使用分为：资源使用情况信息，流程摘要报告，Fitter 报告，分析和综合报告，编译消息。 |
| 2018.07.03 | 18.00 | 修复笔误并在主题中添加了链接指导：更换目标存储块。 |
| 2017.05.08 | 17.0.0 | <ul style="list-style-type: none"> 修订主题：解决资源使用率问题，指导：优化针对区域而非速度的综合 |
| 2016.05.02 | 16.0.0 | <ul style="list-style-type: none"> Stated limitations about deprecated physical synthesis options. |
| 2015.11.02 | 15.1.0 | Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> . |
| 2014.12.15 | 14.1.0 | 将 Fitter Setting, Analysis & Synthesis 和 Physical Synthesis Optimizations Setting 的位置更新到 Compiler Setting。 |
| 2014 年 6 月 | 14.0.0 | <ul style="list-style-type: none"> 删除了 Cyclone III 和 Stratix III 器件参考。 删除了基于 Macrocell 的 CPLD 的相关信息。 更新了模板。 |
| 2013 年 5 月 | 13.0.0 | 首次发布。 |

相关链接

文档存档

关于之前版本的 *Intel Quartus Prime* 手册，请搜索文档存档。

5. 分析和优化设计平面布局规划

随着 FPGA 设计密度的增加，分析设计性能，布线拥塞和逻辑布局的能力对于满足设计要求至关重要。本章讨论 Chip Planner 和 Logic Lock (Standard) 区域如何帮助改善设计布局规划。

设计布局规划分析有助于收敛时序并确保高度复杂设计中实现最佳性能。通过 Intel Quartus Prime Chip Planner 的分析能力，可帮助快速完成设计的时序收敛。可将 Chip Planner 连同 Logic Lock (Standard) 区域一起使用以分层编译设计并协助布局规划。此外，使用分区保留单次编译运行后的布局和布线结果。

可执行设计分析，以及通过 Chip Planner 创建并优化设计布局规划。要进行 I/O 约束，请使用 Pin Planner。

相关链接

[管理器件 I/O 管脚 \(第 0 页\)](#)

5.1. Chip Planner 中的设计布局规划分析

Chip Planner 直观显示芯片资源，简化了布局规划分析。通过 Chip Planner，可查看编译后的布局，连接和布线路径。

Chip Planner 允许：

- 进行约束更改，例如创建并删除资源约束。
- 执行编译后更改，例如创建，移动并删除逻辑单元和 I/O 原子。
- 执行功率和设计分析。
- ECO 实现。
- 更改资源之间的连接并对逻辑单元，I/O 元件，PLL，RAM 和数字信号处理（DSP）块的属性进行编译后更改。

Chip Planner 实例展示：

- Logic Lock (Standard) 区域
- 相关资源使用情况
- 布线详情
- 节点间扇入和删除连接
- 寄存器之间的实现路径
- 路径的延迟估算
- 布线拥塞信息

5.1.1. 启动 Chip Planner

要启动 Chip Planner，请选择 **Tools > Chip Planner**。还可通过以下方法启动 Chip Planner：

- 在 Intel Quartus Prime 软件工具栏上点击 Chip Planner 图标 .
- 在以下工具中，右键点击任何芯片资源并选择 **Locate > Locate in Chip Planner**:
 - Design Partition Planner
 - Compilation Report
 - **Logic Lock (Standard) Regions Window**
 - Technology Map Viewer
 - **Project Navigator** window
 - RTL source code
 - Node Finder
 - Simulation Report
 - RTL Viewer
 - Report Timing panel of the Timing Analyzer

5.1.2. Chip Planner GUI 组件

5.1.2.1. Chip Planner Toolbar

Chip Planner 工具栏为直观设计分析提供强大工具。可从 **View** 或 **Shortcut** 菜单，或点击工具栏中的图标访问 Chip Planner 命令。

5.1.2.2. 层次设置和编辑模式

Chip Planner 允许控制资源的显示。要确定可执行的操作，请使用 **Editing Mode**。

Layers Setting (层次设置) 窗格

通过 **Layers Settings** 窗格，可管理 Chip Planner 显示的图形元素。

单击 **View > Layers Settings** 开启 **Layers Settings** 窗格。**Layers Settings** 窗格提供图层预设，以将经常一同使用的资源进行分组。**Basic**、**Detailed** 和 **Floorplan Editing** 默认预设有助于常规约束相关的活动。还可根据需要创建自定义预设。**Design Partition Planner** 预设被优化用于特定活动。

Editing Mode

The Chip Planner's **Editing Mode** determines the operations that you can perform. The **Assignment** editing mode allows you to make assignment changes that are applied by the Fitter during the next place and route operation. The **ECO** editing mode allows you to make post-compilation changes, commonly referred to as engineering change orders (ECOs).

Select the editing mode appropriate for the work that you want to perform, and a preset that displays the resources that you want to view, in a level of detail appropriate for your design.

相关链接

- [查看特定体系结构设计信息 \(第 95 页\)](#)

- [层次设置对话框](#)
Intel Quartus Prime Help 中

5.1.2.3. 查找历史窗口

优化设计布局规划时，可能需要多次在 Chip Planner 查找路径或节点。**Locate History** 窗口列出您使用 **Locate in Chip Planner** 命令显示的所有节点和路径，可轻松访问你感兴趣的节点和路径。

如果从 **Timing Analyzer Report Timing** 窗格中查找所需路径，**Locate History** 窗会显示所需时钟路径。如果从 **Timing Analyzer Report Timing** 窗格查找到达路径，**Locate History** 窗口会显示从到达时钟到到达数据的路径。**Locate History** 窗口中双击一个节点或路径，Chip Planner 中会显示所选节点或路径。

5.1.2.4. Chip Planner 布局规划视图

Chip Planner 使用分层缩放查看器显示各抽象级下的目标 Intel 器件。放大时，抽象水平下降，展现更多设计相关的详细信息。

鸟瞰视图

Bird's Eye View (鸟瞰视图) 显示整个芯片资源使用情况的高级图像并提供一种快速有效的方式在 Chip Planner 中对感兴趣区域进行浏览。

当需要查看的设计部分位于芯片的两端时，以及在不丢失参考框架的情况下，进行资源元件之间快速浏览时，尤为有用。

属性窗口

Properties 窗口显示 Chip Planner 中当前选择对象的详细属性（例如原子，路径，Logic Lock (Standard) 区域或布线元素）。要显示 **Properties** 窗口，右键单击对象并选择 **View > Properties**。

相关链接

鸟瞰图视图

Intel Quartus Prime Help 中

5.1.3. 查看特定体系结构设计信息

Chip Planner 允许查看设计相关的特定体系结构信息。在 **Layers Settings** 窗格中使能选项，可以查看：

- **Device routing resources used by your design**—查看块的连接方式，以及连接块的信号路由。
- **LE configuration**—查看设计中的逻辑元件 (LE) 配置。例如，可查看已使用哪些 LE 输入；LE 是否使用寄存器，look-up table (LUT，查找表) 或两者兼顾；还有通过 LE 的信号流。
- **ALM configuration**—查看设计中的 ALM 配置。例如，查看已使用哪些 ALM 输入；ALM 是否占用寄存器，上层 LUT，底层 LUT，或两者兼顾。还可查看通过 ALM 的信号流。

- **I/O configuration**—查看器件 I/O 资源使用情况。例如，可查看已使用 I/O 资源则哪些组件，延迟链设置是否使能，设置为哪个 I/O 标准，和通过 I/O 的信号流。
- **PLL configuration**—查看设计中的锁相环（phase-locked loop, PLL）配置。例如，可查看已使用的 PLL 控制信号中哪些按照您的 PLL 设置。
- **Timing**—查看 FPGA 元件的输入和输出之间的延迟。例如，可分析 DATAB 输入到 COMBOUT 输出的时序。

In addition, you can modify the following device properties with the Chip Planner:

- LEs and ALMs
- I/O cells
- PLLs
- Registers in RAM and DSP blocks
- Connections between elements
- Placement of elements

For more information about LEs, ALMs, and other resources of an FPGA device, refer to the relevant device handbook.

相关链接

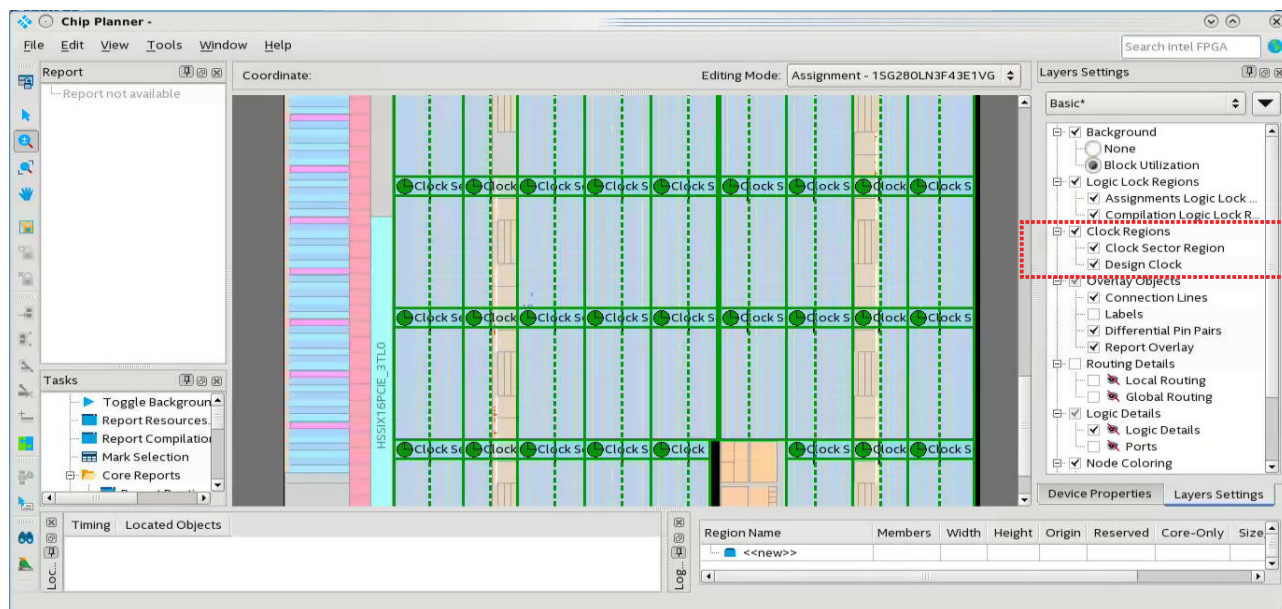
- [层次设置和编辑模式](#) (第 94 页)
- [层次设置对话框](#)
Intel Quartus Prime Help 中

5.1.4. 查看器件中可用的时钟网络

在 **Layers Settings** 窗格中使能一个时钟区域层时，将显示由全局和区域时钟网络驱动的芯片区域。所选器件不包含给定时钟区域时，对话框中无该类别的选项。

This global clock display feature is available for Arria V, Intel Arria 10, Cyclone V, Stratix IV, and Stratix V device families.

图 34. 时钟区域



- 根据您在 **Layers Settings** 窗格中激活的时钟层，Chip Planner 显示器件中的区域和全局时钟域，以及时钟区域，管脚和 PLL 之间的连接。
- 时钟区域显示为矩形覆盖框，以标签指示时钟类型和索引。点击时钟区域选择一个时钟网络区域。左上角钟形图标表示该区域代表时钟网络区域。
- Spine/sector 时钟区域中间有垂直虚线。本虚线表示行时钟的两列在扇区时钟里相遇的位置。
- 要更改 Chip Planner 中显示时钟区域的颜色，选择 **Tools > Options > Colors > Clock Regions**。

相关链接

- [脊柱时钟限制](#) (第 66 页)
- [层次设置和编辑模式](#) (第 94 页)
- [报告 Spine Clock 使用率对话框 \(Chip Planner\)](#)
Intel Quartus Prime Help 中

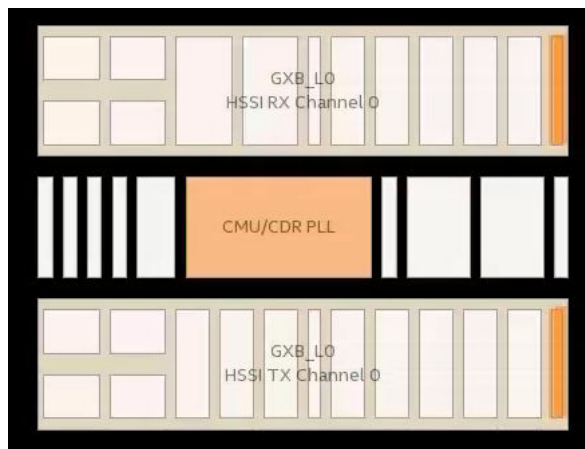
5.1.5. 查看 I/O Bank

要在 Chip Planner 中查看器件的 I/O bank 映射，请在 **Tasks** 窗格中双击 **Report All I/O Banks**。

5.1.6. 查看高速串行接口 (HSSI)

对于所选器件系列，该 Chip Planner 显示高速串行接口接收器和发送器的块视图详情。要显示 iHSSI 块视图，在 **Tasks** 窗格中双击 **Report HSSI Block Connectivity**。

图 35. Intel Arria 10 HSSI Channel 块



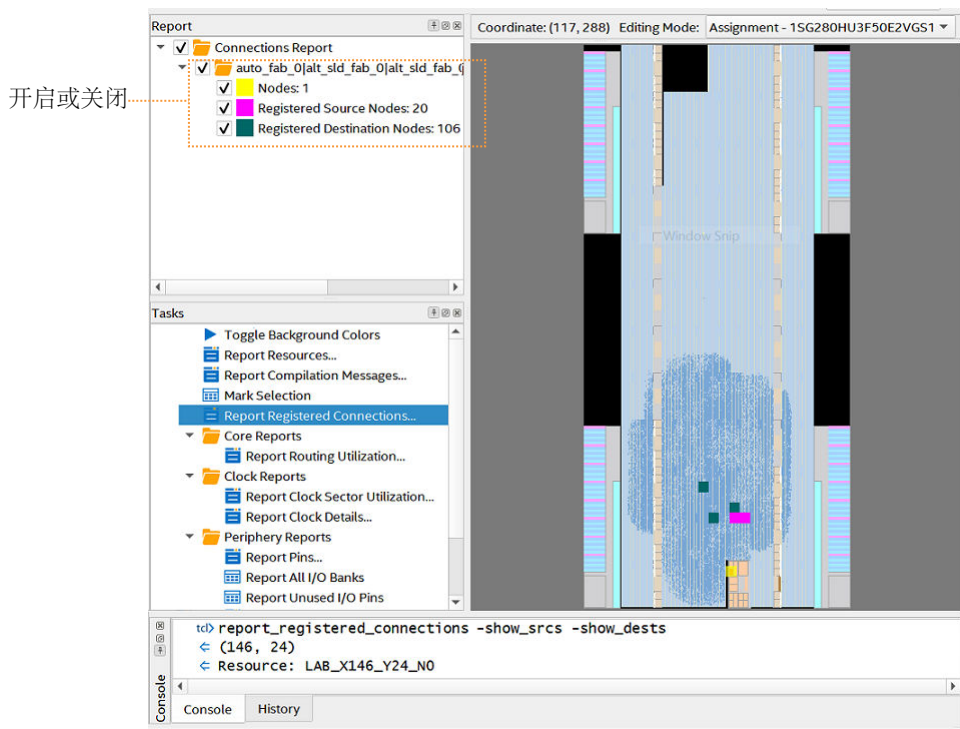
5.1.7. 查看已布局节点的源和目标

Chip Planner 允许使用 **Report Registered Connections** 任务查看已编译设计中节点的已寄存扇入和扇出。该报告与 **Generate Fanin/Fanout connections** 报告的不同在于，显示无连接线的源和目标，可能导致视图模糊不清。

1. Chip Planner 中，选择 1 个或多个节点。
2. **Task** 窗格中，双击 **Report Registered Connections**。
3. 从对话框选择选项，并单击 **OK**。

Reports 窗格显示已寄存的源和目标节点。可开启或关闭图形视图的可见性。

图 36. 报告已寄存的连接






相关链接

- 查看已布局资源的扇入和扇出连接 (第 99 页)
- 展开连接命令 (查看 [Menu](#))
Intel Quartus Prime Help 中

5.1.8. 查看已布局资源的扇入和扇出连接

显示扇入到或扇出至某资源的原子，包括连接线。

要显示所选资源的扇入或扇出连接，

1. 在 Chip Planner 工具栏中，单击 **Generate Fan-In Connections**  图标或 **Generate Fan-Out Connections**  图标。
2. 删除“Chip Planner”视图上显示的其他连接，请单击 **Clear Unselected Connections**  图标。

也可从 Chip Planner 的 **View** 菜单执行该操作。


相关链接

- 查看已布局节点的源和目标 (第 98 页)
- 展开连接命令 (查看 [Menu](#))
Intel Quartus Prime Help 中

5.1.9. 生成即时扇入和扇出连接

显示所选原子的即时扇入或扇出连接。


例如，通过查看即时扇入获取逻辑资源时，可看到驱动逻辑资源的布线资源。可为所有逻辑资源和布线资源生成即时扇入或扇出。

- 显示即时扇入或扇出连接，单击 **View > Generate Immediate Fan-In Connections** 或 **View > Generate Immediate Fan-Out Connections**。
- 删除显示中的连接，请使用 Chip Planner 工具栏中的 **Clear Unselected Connections** 图标 。

5.1.10. 在 Chip Planner 中管理路径

使用 Chip Planner 管理逻辑单元之间的路径。以下实例使用 Chip Planner 从 Timing Analysis 报告详细研究路径。

5.1.10.1. 分析路径的连接

在 Chip Planner 中确定所选路径或连接的组成单元，请在 Chip Planner 工具栏单击 **Expand Connections** 图标 。

相关链接

展开连接命令 (查看 Menu)

Intel Quartus Prime Help 中

5.1.10.2. 找到从 Timing Analysis Report 到 Chip Planner 的路径

要周到从 Timing Analysis 报告到 Chip Planner 的路径，请执行以下步骤：

- 在“Timing Analysis 报告”中选择要查找的路径。
- 右键点击路径并指向 **Locate Path > Locate in Chip Planner**。
该路径将出现在 Chip Planner 的 **Locate History** 窗口中。


图 37. Locate History Window 中的路径列表

| Timing | Located Objects |
|--|---|
| Located 10 paths | |
|  -0.790 | ram1~port_b_address1FITTER_CREATED_FF -> ram1 |
|  -0.758 | ram1~port_b_address2FITTER_CREATED_FF -> ram1 |
|  -0.753 | ram1~port_b_address1FITTER_CREATED_FF -> ram1 |
|  -0.725 | ram1~port_b_address1FITTER_CREATED_FF -> ram1 |
|  -0.723 | ram1~port_b_address4FITTER_CREATED_FF -> ram1 |
|  -0.710 | ram1~port_b_address4FITTER_CREATED_FF -> ram1 |
|  -0.707 | ram1~port_b_address0FITTER_CREATED_FF -> ram1 |
|  -0.678 | ram1~port_b_address0FITTER_CREATED_FF -> ram1 |
|  -0.677 | ram1~port_b_address0FITTER_CREATED_FF -> ram1 |
|  -0.670 | ram1~port_b_address3FITTER_CREATED_FF -> ram1 |

相关链接

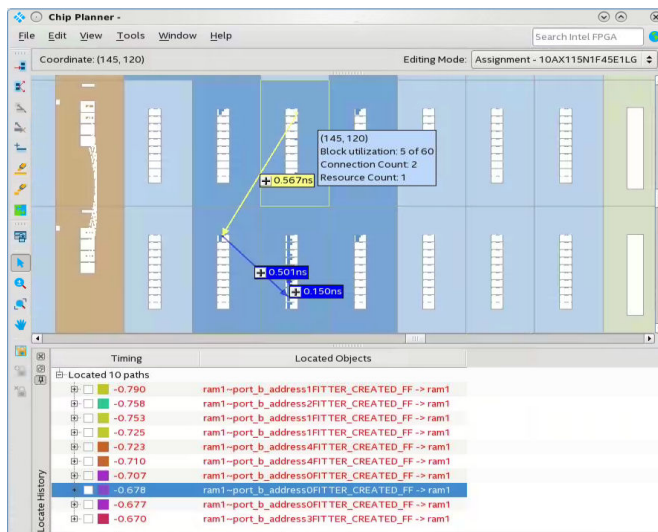
使用 Timing Analyzer 显示 Path Report (第 56 页)

5.1.10.3. 显示延迟

通过 **Show Delays** 功能，可查看 Timing Analyzer 报告中出现的路径的时序延迟。要访问该功能，在主菜单中单击 **View > Show Delays**。或 Chip Planner 工具栏中单击“Show Delays”图标。要查看所选路径上的局部延迟，单击 **Locate History** 窗口中已显示路径旁的“+”号。

例如，可查看两个逻辑资源间的延迟或逻辑资源和布线资源之间的延迟。

图 38. 显示与 Timing Analyzer Path 相关的延迟



5.1.10.4. 查看布线资源

通过 Chip Planner 和 **Locate History** 窗口，可查看路径或连接使用的布线资源。还可选择显示“Arrival Data”路径和“Arrival Clock”路径。

图 39. 显示物理布线

Locate History 窗口中，右键单击路径选择 **Show Physical Routing** 显示物理路径。要调整显示，单击右键并选择 **Zoom to Selection**。

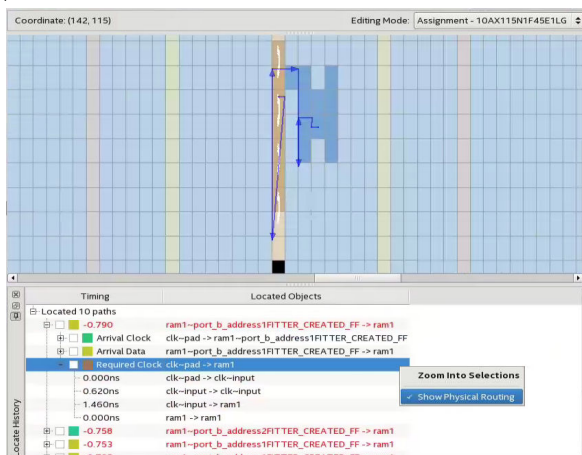
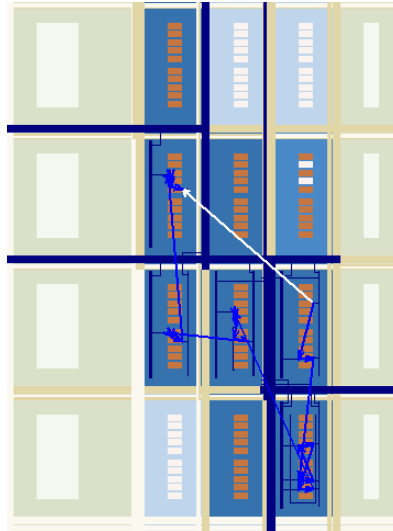


图 40. 突出显示布线

要查看 Fitter 布线路径的行和列，右键单击路径并选择 **Highlight Routing**（突出显示布线）。



相关链接

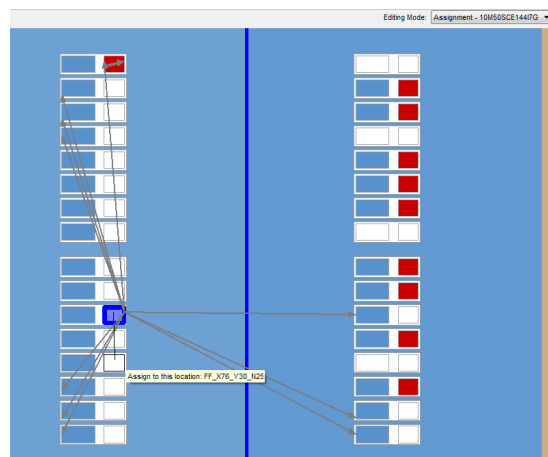
[查看布线拥塞](#) (第 0 页)

5.1.11. 在 Chip Planner 中查看约束

可使用约束编辑模式和 **Layers Settings** 窗格中 **Floorplan Editing** 预设 在 Chip Planner 中查看位置约束。

Chip Planner 以预定义颜色显示已约束资源（默认为灰色）。

图 41. 在 Chip Planner 中查看约束



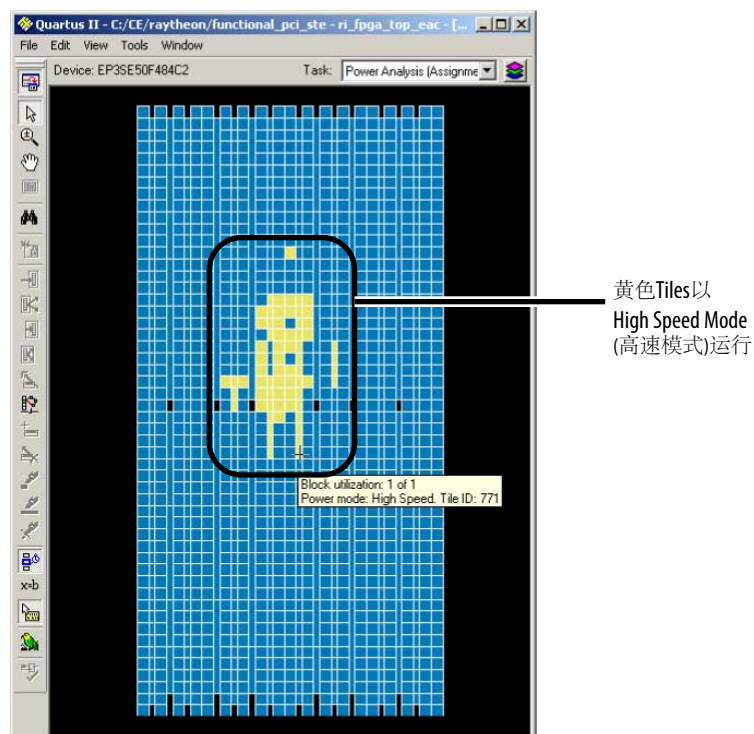
创建或移动约束，或对 **Logic Lock (Standard)** 区域进行节点和管脚位置约束，可将已选资源拖动到新的位置。Fitter 会应用您在下个布局布线操作期间创建的约束。

5.1.12. 在 Chip Planner 中查看高速和低功耗 Tile

一些 Intel 器件具有可在高速模式和低功耗模式中运行的 ALM。适配处理期间可在 Intel Quartus Prime 软件中设置节能模式。将 ALM 组合在一起形成较大的块，称为“tiles”。

查看功耗映射，请在运行 Fitter 后双击 **Tasks > Core Reports > Report High-Speed/Low-Power Tiles**。Chip Planner 以对比色显示低功耗和高速 tiles；黄色 tiles 以高速模式运行，而蓝色 tiles 以低功耗模式运行。

图 42. Viewing High-Speed and Low Power Tiles in a Stratix Device



相关链接

AN 514: Stratix IV FPGA 中的功率优化

5.1.13. Viewing Design Partition Placement

With the **Report Design Partitions** command, you can view the physical placement of design partitions using the same color map as the Design Partition Planner.

The **Report Design Partitions Advanced** command opens the **Report Design Partitions Advanced** dialog box that allows you to select a partition and generate a report of the pins belonging to the partition. It highlights the selected partition's boundary ports and pins in the Chip Planner, and optionally reports the routing utilization and routing element details.

5.2. Logic Lock (Standard)区域

Logic Lock (Standard)区域是布局规划位置约束。When you assign instances or nodes to a 为 Logic Lock (Standard)区域进行实例和节点约束时，指示 Fitter 将这些实例和节点放置在该区域中。一个布局规划可包含多个 Logic Lock (Standard)区域。

You can use the Design Partition Planner in conjunction with Logic Lock (Standard) regions to create a floorplan for your design.

相关链接

[创建 Logic Lock \(Standard\)区域 \(第 104 页\)](#)

5.2.1. Logic Lock (Standard)区域的属性

以下表格列出 Logic Lock (Standard)区域的属性。Intel Quartus Prime 软件中，Logic Lock (Standard) Regions 窗口显示设计中所有 Logic Lock (Standard)区域的属性。

表 18. Logic Lock (Standard)区域属性

| 名称 | 值 | 行为 |
|----------|-------------------------|--|
| Size | Auto Fixed | Auto allows the Intel Quartus Prime software to determine the appropriate size of a region given its contents. Fixed regions have a shape and size that you define. |
| Width | 列数 | 指定 Logic Lock (Standard)区域的宽度。 |
| Height | 行数 | 指定 Logic Lock (Standard)区域的宽度。 |
| State | Floating Locked | Floating allows the Intel Quartus Prime software to determine the location of the region on the device. Floating regions appear with a dashed boundary in the floorplan. Locked allows you to specify the location of the region. Locked regions appear with a solid boundary in the floorplan. A Locked region must have a Fixed size. |
| Origin | 任何布局规划位置 Undetermined | 指定 Logic Lock (Standard)区域在布局规划中的位置。原本位于 Logic Lock (Standard)区域的左下角。 |
| Reserved | Off On | 防止 Fitter 在该区域中放置其他逻辑。 |

相关链接

[Logic Lock \(Standard\)区域窗口 \(第 112 页\)](#)

5.2.2. 创建 Logic Lock (Standard)区域

You can define a Logic Lock (Standard) region by its height, width, and location; Alternatively, you can specify the size or location of a region, or both, or the Intel Quartus Prime software can generate these properties automatically. The Intel Quartus Prime software bases the size and location of a region on the contents of the region and the timing requirements of the module.

The Intel Quartus Prime software displays Logic Lock (Standard) regions with colors indicating the percentage of resources available in the region. An orange Logic Lock (Standard) region indicates a nearly full Logic Lock (Standard) region.

Intel Quartus Prime software cannot automatically define the size of a region if the location is **Locked**. Therefore, if you want to specify the exact location of the region, you must also specify the size.

5.2.2.1. 使用 **Chip Planner** 创建 **Logic Lock (Standard)** 区域

1. 点击 **View > Logic Lock (Standard) Regions > Create Logic Lock (Standard) Region**
2. 单击并在 **Chip Planner** 布局规划图上拖动以创建首选位置和大小区域。

创建区域后，可定义区域的形状，随后为区域约束单个实体。约束实体或定义形状无先后顺序。

5.2.2.2. 使用 **Project Navigator** 创建 **Logic Lock (Standard)** 区域

1. 对设计进行完整编译或分析和详细说明。
2. 如果 **Project Navigator** 未开启，则单击 **View > Utility Windows > Project Navigator**。Project Navigator 显示设计的层次。
3. 随着设计层次完全展开，右键单击任意设计实例，并单击 **Create New Logic Lock (Standard) 区域**。
4. 对新区域约束实例。

新区域的名称与实体名称相同。

5.2.2.3. 使用 **Logic Lock (Standard) Regions Window** 创建 **Logic Lock (Standard)** 区域

1. 点击 **Assignments > Logic Lock (Standard) Regions Window**。
2. **Logic Lock (Standard) Regions** 窗口中，点击 **<<new>>**。

创建区域后，可定义区域的形状，随后为区域约束单个实体。约束实体或定义形状无先后顺序。

相关链接

[Logic Lock \(Standard\) 区域窗口](#) (第 112 页)

5.2.2.4. 定义布线区域

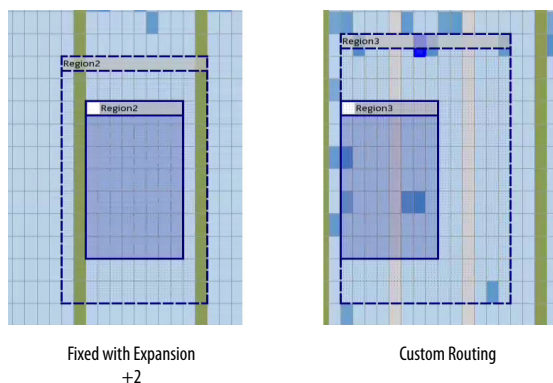
布线区域是 **Logic Lock** 区域中指定布线区域的元素。不限区域必须包含现有 **Logic Lock** 布局区域。布线区域不可设置为“reserved”（保留）。要定义布线区域，在 **Logic Lock (Standard) Regions** 窗口双击 **Routing Region** 单元，并从下拉菜单选择一个选项。

有效的布线区域选项为：

表 19. 布线区域选项

| 选项 | 说明 |
|----------------------|---|
| Unconstrained (默认) | 允许适配器使用器件中任何可用路线。 |
| Whole Chip | 与 Unconstrained 相同，但会在 Intel Quartus Prime 设置文件 (.qsf) 中写入约束。 |
| Fixed with Expansion | 按照布局区域的轮廓。布线区域按大于布局区域的行数/列数进行缩放。 |
| Custom | 允许在 Logic Lock 区域周围自定义布线区域形状。选择 Custom 选项时，布局 and 布线区域在 Chip Planner 中独立移动。这中情况下，使用 Shift 键选择移动布局 and 布线区域。 |

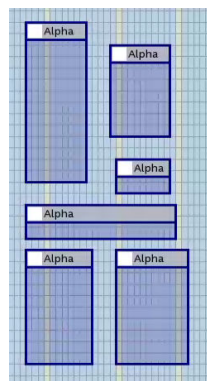
图 43. Routing Regions



5.2.2.5. 非连续 Logic Lock (Standard)区域

使用 Logic Lock (Standard)区域操控工具创建不相交的区域。非连续区域作为所有 Logic Lock (Standard)区域属性中的单个 Logic Lock (Standard)区域运行。

图 44. 非连续 Logic Lock (Standard)区域



相关链接

[合并 Logic Lock \(Standard\)区域](#) (第 107 页)

5.2.2.6. 使用 Auto Sized Region 的考量

如果使用 **Auto** Sized Logic Lock (Standard)区域，请考虑：

- **Auto/Floating** 区域无法保留
- 验证您的 **Logic Lock (Standard)** 区域不为空。如果对区域约束任何实例，则 **Fitter** 会将尺寸减小为 0 x 0，使得该区域无效。
- 区域可能与分区相关联，也可能无关联。当分区和 **Auto Sized Logic Lock (Standard)** 区域组合时，可灵活解决特定适配挑战。然而，每添加一个约束都会减少可用的解决方案，过多约束会导致 **Fitter** 无法找到解决方案。具体示例如下：
 - 如果分区在综合期间被保留或未被保留，则 **Logic Lock (Standard)** 区域将逻辑限制与特定区域，同时允许 **Fitter** 优化分区内的逻辑，以及优化 **Logic Lock (Standard)** 分区内的布局。
 - 如果在布局，布线或最终阶段保留分区；**Logic Lock (Standard)** 不是有效布局边界，因为分区逻辑的位置已固定。
 - 然而，如果 **Logic Lock (Standard)** 区域被保留，**Fitter** 避免将其他逻辑置于本区域中，有助于降低资源拥塞。
- **Logic Lock (Standard)** 区域的设置结果满足规格后，可进行如下操作：
 - 将 **Logic Lock (Standard)** 区域转换成 **Fixed Size**。
 - 保留 **Logic Lock (Standard)** 区域属性为 **Auto Sized** 并将区域用作“keep together”类型的功能。
 - 如果 **Logic Lock (Standard)** 区域也是一个分区，则可通过分区保留布局和布线并完整删除 **Logic Lock (Standard)** 区域。

5.2.3. 定制 **Logic Lock** 区域的形状

创建定制形状的 **Logic Lock** 区域。可执行逻辑操作。非矩形 **Logic Lock (Standard)** 区域可帮助排除某些资源，或将设计的某些部分放置于指定器件资源周围以提高性能。

注意: 17.1 版本中无 **Logic Lock (Standard)** 形状撤销功能。

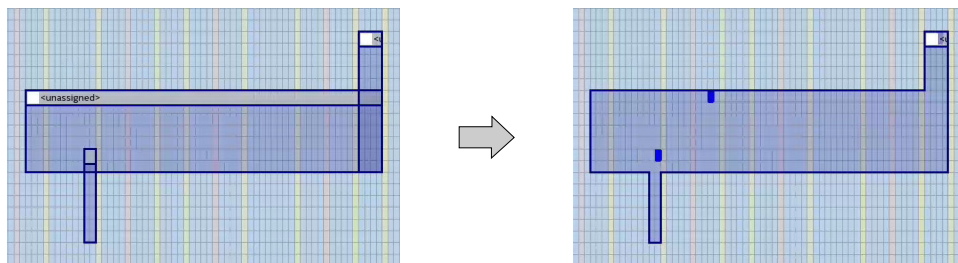
5.2.3.1. 合并 **Logic Lock (Standard)** 区域

合并 2 个或多个 **Logic Lock (Standard)** 区域，可执行如下步骤：

1. 确保您要合并的区域中，具有逻辑约束的区域不能多于一个。
2. 将区域排列到合成区域应在的位置。
3. 按住 **Shift** 键，并单击选择所有需要合并的单个区域。
4. 右键单击任何已选择的 **Logic Lock (Standard)** 区域并选择 **Logic Lock (Standard) Regions > Merge Logic Lock (Standard) Region**。所选的单个区域合并后创建为新的单个区域。

注意: By default, the new **Logic Lock (Standard)** region has the same name as the component region containing the greatest number of resources; however, you can rename the new region. In the **Logic Lock (Standard) Regions Window**, the new region is shown as having a **Custom Shape**.

图 45. 使用合并 **Logic Lock (Standard)** 区域命令



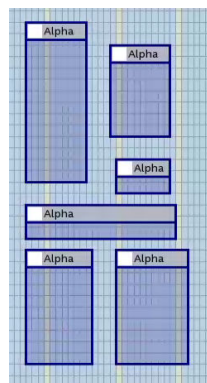
相关链接

创建 **Logic Lock (Standard)** 区域 (第 104 页)

5.2.3.2. 非连续 **Logic Lock (Standard)** 区域

使用 **Logic Lock (Standard)** 区域操控工具创建不相交的区域。非连续区域作为所有 **Logic Lock (Standard)** 区域属性中的单个 **Logic Lock (Standard)** 区域运行。

图 46. 非连续 **Logic Lock (Standard)** 区域



相关链接

合并 **Logic Lock (Standard)** 区域 (第 107 页)

5.2.4. Placing Logic Lock (Standard) Regions

A fixed region must contain all resources required by the design block assigned to the region. Although the Intel Quartus Prime software can automatically place and size Logic Lock (Standard) regions to meet resource and timing requirements, you can manually place and size regions to meet your design requirements.

If you manually place or size a Logic Lock (Standard) region:

- Logic Lock (Standard) regions with pin assignments must be placed on the periphery of the device, adjacent to the pins. You must also include the I/O block within the Logic Lock (Standard) Region.
- Floating Logic Lock (Standard) regions can overlap with their ancestors or descendants, but not with other floating Logic Lock (Standard) regions.

5.2.5. 将器件资源放入 Logic Lock (Standard) 区域

仅可将实体约束给设计中的一个 Logic Lock (Standard) 区域，但该实体可按层次集成区域。该层及结构允许保留区域具有子区域且不保留子区域中的资源。

如果一个 Logic Lock (Standard) 区域边界包含部分器件资源，Intel Quartus Prime 软件将整个资源约束给该 Logic Lock (Standard) 区域。当 Intel Quartus Prime 软件在放置一个浮动且自动调整大小的区域时，会将该区域放置在满足 Logic Lock (Standard) 区域内容要求的范围内。

使用 **Logic Lock Region** 窗口添加实例，右键点击该区域并选择 **Logic Lock Properties > Add**，或者，在 Intel Quartus Prime 软件中，将实体从 Hierarchy 查看器拖动到 Logic Lock (Standard) Regions Window 的 Logic Lock (Standard) 区域名字段中。

5.2.5.1. 空白 Logic Lock 区域

Intel Quartus Prime 允许 Logic Lock 区域中无内容。“Empty regions”是管理 FPGA 空间用于日后所需逻辑的工具。该技术仅在将区域设置为 **Reserved** 时有效。

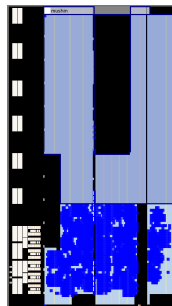
使用空白 Logic Lock 区域的原因：

- 初步平面布局规划。
- 复杂增量构建。
- 基于团队的设计和互连逻辑。
- 限制逻辑布局。

由于 Logic Lock 区域未保留任何布线资源，Fitter 可能将该区域用于布线目的。

将空白 Logic Lock 区域设置为 **Core Only** 属性。当空白区域中包含外设资源时，可限制外设组件布局，从而可能导致设计无法适配。命名空白区域后，可执行与任何已填充 Logic Lock 区域相同的操控。

图 47. 放置于空白区域外的逻辑



该图显示为其周围的 empty (空白) Logic Lock 区域和逻辑。但某些 IO, HSSIO 和 PLL 位于 empty 区域中。出现这样的布局是因为输出端口连接到 IO，且此 IO 始终为 root_partition (top-level partition) 的一部分。

5.2.5.2. 管脚约束

A Logic Lock (Standard) region incorporates all device resources within its boundaries, including memory and pins. The Intel Quartus Prime Standard Edition software automatically includes pins when you assign an instance to a region. You can manually exclude pins with a **Core Only** assignment.

注意: Pin assignments to Logic Lock (Standard) regions are effective only in fixed and locked regions. Pin assignments to floating regions do not influence the placement of the region.

You can assign an entity in the design to only one Logic Lock (Standard) region, but the entity can inherit regions by hierarchy. This hierarchy allows a reserved region to have a subregion without reserving the resources in the subregion.

When the Intel Quartus Prime software places a floating auto-sized region, it places the region in an area that meets the requirements of the contents of the Logic Lock (Standard) region.

5.2.5.3. 保留 Logic Lock (Standard) 区域

Reserved 属性指示 Fitter 仅对您指定约束到 Logic Lock (Standard) 区域 (Logic Lock (Standard) region) 中的实体和节点进行布局。

Intel Quartus Prime 软件实现对 Logic Lock (Standard) 区域的所有实体和节点约束。有时实体和节点并不占用整个区域, 从而使得某些区域资源保持空闲。

要提高区域资源利用率和性能, Intel Quartus Prime 软件默认将仍未约束给另一区域的节点和实体填充为占用的空闲资源。要防止出现该行为, 可在 **Logic Lock (Standard) Region Properties > General** 选项卡中开启 **Reserved**。

5.2.5.4. Excluded Resources

The Excluded Resources feature allows you to easily exclude specific device resources such as DSP blocks or M4K memory blocks from a Logic Lock (Standard) region.

For example, you can assign a specific entity to a Logic Lock (Standard) region but allow the DSP blocks of that entity to be placed anywhere on the device. Use the Excluded Resources feature on a per- Logic Lock (Standard) region member basis.

To exclude certain device resources from an entity, in the **Logic Lock (Standard) Region Properties** dialog box, highlight the entity in the **Design Element** column, and click **Edit**. In the **Edit Node** dialog box, under **Excluded Element Types**, click the **Browse** button. In the **Excluded Resources Element Types** dialog box, you can select the device resources you want to exclude from the entity. When you have selected the resources to exclude, the **Excluded Resources** column is updated in the **Logic Lock (Standard) Region Properties** dialog box to reflect the excluded resources.

注意: The Excluded Resources feature prevents certain resource types from being included in a region, but it does not prevent the resources from being placed inside the region unless you set the region's **Reserved** property to **On**. To indicate to the Fitter that certain resources are not required inside a Logic Lock (Standard) region, define a resource filter.

5.2.5.5. Logic Lock (Standard) Assignment Precedence

You can encounter conflicts during the assignment of entities and nodes to Logic Lock (Standard) regions. For example, an entire top-level entity might be assigned to one region and a node within this top-level entity assigned to another region.

To resolve conflicting assignments, the Intel Quartus Prime software maintains an order of precedence for Logic Lock (Standard) assignments. The following order of precedence, from highest to lowest, applies:

1. Exact node-level assignments
2. Path-based and wildcard assignments
3. Hierarchical assignments

注意:

To open the **Priority** dialog box, select **Logic Lock (Standard) Regions Properties > General > Priority**. You can change the priority of path-based and wildcard assignments with the **Up** and **Down** buttons in the **Priority** dialog box. To prioritize assignments between regions, you must select multiple Logic Lock (Standard) regions and then open the **Priority** dialog box from the **Logic Lock (Standard) Regions Properties** dialog box.

5.2.5.6. 虚拟管脚

虚拟管脚是编译期间 **Compiler** 临时映射到逻辑单元而非管脚的 I/O 元件。软件将虚拟管脚实现为 LUT。要约束 **Virtual Pin**，可使用 **Assignment Editor**。通过将 **Virtual Pin** 约束到 I/O 元件来创建虚拟管脚。

将 **Virtual Pin** 约束应用到输入管脚时，该管脚不再显示为 FPGA 管脚；**Compiler** 将虚拟管脚固定到设计的 **GND** 中。虚拟管脚不是浮动节点。

仅将虚拟管脚用于底层设计实体中的 I/O 元件，从而在实体导入设计后成为节点；例如，编译部分设计时。

注意:

Virtual Pin 逻辑选项必须约束到输入管脚或输出管脚。如果将该选项约束到双向管脚，三态管脚或已寄存的 I/O 元件，则 **Analysis & Synthesis** 会忽略该约束。如果将该选项约束到三态管脚，则 **Fitter** 插入一个 I/O 缓冲作为三态逻辑；因此，该管脚不能是虚拟管脚。如果要继续将所约束管脚用作虚拟管脚，则可使用多路复用器代替三态管脚。除了直接连接器件 I/O 管脚的信号，请勿使用三态逻辑。

在顶层设计中，将这些虚拟管脚连接到另一模块的内部节点。通过对虚拟管脚的约束，可按照顶层模块中对应内部节点位置，将这些管脚放置在器件中相同的地方和区域。编译带有多于目标器件所允许管脚数的 **Logic Lock (Standard)** 模块时，则可使用 **Virtual Pin** 选项。将 **Virtual Pin** 选项集成到顶层设计后，该选项可使命设计模块的时序分析并更加紧密匹配模块性能。

要通过 **Node Finder** 显示设计中所有已约束的虚拟管脚，可将 **Filter Type** 设置为 **Pins: Virtual**。要从 **Assignment Editor** 访问 **Node Finder**，双击 **To** 字段；当箭头出现在字段右侧时，点选 **Node Finder**。

相关链接

- [通过 Tcl 命令约束虚拟管脚](#) (第 116 页)
 - [管理器件 I/O 管脚](#) (第 0 页)
 - [Node Finder Command](#) (查看)
- Intel Quartus Prime Help 中*

5.2.6. Hierarchical (Parent and Child) Logic Lock (Standard) Regions

To further constrain module locations, you can define a hierarchy for a group of regions by declaring parent and child regions.

The Intel Quartus Prime software places a child region completely within the boundaries of its parent region; a child region must be placed entirely within the boundary of its parent. Additionally, parent and child regions allow you to further improve the performance of a module by constraining nodes in the critical path of a module.

To make one Logic Lock (Standard) region a child of another Logic Lock (Standard) region, in the Logic Lock (Standard) Regions window, select the new child region and dragging the new child region into its new parent region.

注意:

The Logic Lock (Standard) region hierarchy does not have to be the same as the design hierarchy.

You can create both auto-sized and fixed-sized Logic Lock (Standard) regions within a parent Logic Lock (Standard) region; however, the parent of a fixed-sized child region must also be fixed-sized. The location of a locked parent region is locked relative to the device; the location of a locked child region is locked relative to its parent region. If you change the parent's location, the locked child's origin changes, but maintains the same placement relative to the origin of its parent. The location of a floating child region can float within its parent. Complex region hierarchies might result in some LABs not being used, effectively increasing the resource utilization in the device. Do not create more levels of hierarchy than you need.

5.2.7. 其他 Intel Quartus Prime Logic Lock (Standard) 设计功能

为补充 **Logic Lock (Standard) Regions Window**，Intel Quartus Prime 软件具有其他功能以帮助设计 Logic Lock (Standard) 区域。

5.2.7.1. Analysis and Synthesis Resource Utilization by Entity

The Compilation Report contains an **Analysis and Synthesis Resource Utilization by Entity** section, which reports resource usage statistics, including entity-level information. You can use this feature to verify that any Logic Lock (Standard) region you manually create contains enough resources to accommodate all the entities you assign to it.

5.2.7.2. Intel Quartus Prime 修订功能

评估设计中的各 Logic Lock (Standard) 区域时，可能希望尝试各种配置以达到所需结果。Intel Quartus Prime Revisions (修订) 功能允许通过不同设置组织同一工程直到找到最佳配置。

要使用 Revisions 功能，请选择 **Project > Revisions**。可根据当前设计或任何之前创建的版本创建修订。每个修订都有相关说明。可使用修订内容组织针对 Logic Lock (Standard) 区域创建的布局约束。

5.2.8. Logic Lock (Standard) 区域窗口

Logic Lock (Standard) Regions Window (区域窗口) 提供关于设计中定义的所有 Logic Lock (Standard) 区域的摘要。可使用 Logic Lock (Standard) Regions Window 创建，约束并修改 Logic Lock (Standard) 区域的属性。

在 Chip Planner 中单击 **View > Logic Lock (Standard) Window** 打开 Logic Lock (Standard) Regions Window，而在 Intel Quartus Prime 中，请单击 **Assignments > Logic Lock (Standard) Window**。

图 48. Logic Lock (Standard) Regions Window

| Region Name | Size | Width | Height | State | Origin | Reserved |
|----------------------|-------|--------------|--------------|----------|--------------|----------|
| LogicLock Regions | | | | Locked | | Off |
| Root Region | Fixed | | | Locked | | Off |
| <<new>> | Auto | Undetermined | Undetermined | Floating | Undetermined | Off |
| path:high_speed_path | Auto | Undetermined | Undetermined | Floating | Undetermined | Off |
| path:low_power_path | Auto | Undetermined | Undetermined | Floating | Undetermined | Off |

The Logic Lock (Standard) Regions Window also has a recommendations toolbar; select a Logic Lock (Standard) region from the drop-down list in the recommendations toolbar to display the relevant suggestions to optimize that Logic Lock (Standard) region.

The Intel Quartus Prime software automatically creates a Logic Lock (Standard) region that encompasses the entire device. This default region is labeled `Root_Region`, and is locked and fixed.

可拖放“列”以更改列顺序来定制 Logic Lock (Standard) Regions Window；也可右键点击任何“列”标题，然后在快捷菜单中适当选择要显示或隐藏的“列”。

Logic Lock (Standard) Regions 属性对话框

使用 **Logic Lock (Standard) Regions Properties** 对话框查看并修改 Logic Lock (Standard)区域详细信息，例如约束到区域的实体和节点，以及需要的资源。

要打开 **Logic Lock (Standard) Regions Properties** 对话框，右键点击区域并选择 **Logic Lock (Standard) Regions Properties...**

相关链接

- [Logic Lock \(Standard\)区域的属性](#) (第 104 页)
- [使用 Logic Lock \(Standard\) Regions Window 创建 Logic Lock \(Standard\)区域](#) (第 105 页)
- [Logic Lock \(Standard\)分区窗口](#)
Intel Quartus Prime Help 中

5.3. 在 Chip Planner 中使用 Logic Lock (Standard)区域

可在 Chip Planner 中轻松创建 Logic Lock (Standard)区域并为其约束资源。

5.3.1. 在 Chip Planner 中查看 Logic Lock (Standard)区域之间的连接

可使用 Chip Planner 查看并编辑 Logic Lock (Standard)区域。要查看并编辑 Logic Lock (Standard)区域，请使用 **Layers Settings** 窗口中的 **Floorplan Editing**，或任何已启用 **User-assigned Logic Lock (Standard) regions** 设置的层次设置模式。

Chip Planner 显示 Logic Lock (Standard) 区域之间的连接。默认情况下, 可将每个连接作为单个线路查看。可选择将两个 Logic Lock (Standard) 区域之间的连接显示为单个绑定连接而非单条连接线路。要使用该选项, 请打开 Chip Planner 并在 View 菜单上, 单击 **Inter-region Bundles**。

相关链接

[Inter-region Bundles 对话框](#)

关于 Inter-region Bundles 对话框的更多信息, 请参阅 Intel Quartus Prime Help。

5.3.2. Using Logic Lock (Standard) Regions with the Design Partition Planner

You can optimize timing in a design by placing entities that share significant logical connectivity close to each other on the device.

By default, the Fitter usually places closely connected entities in the same area of the device; however, you can use Logic Lock (Standard) regions, together with the Design Partition Planner and the Chip Planner, to help ensure that logically connected entities retain optimal placement from one compilation to the next.

You can view the logical connectivity between entities with the Design Partition Planner, and the physical placement of those entities with the Chip Planner. In the Design Partition Planner, you can identify entities that are highly interconnected, and place those entities in a partition. In the Chip Planner, you can create Logic Lock (Standard) regions and assign each partition to a Logic Lock (Standard) region, thereby preserving the placement of the entities.

5.4. 脚本支持

可在 Tcl 脚本中运行本章介绍的处理过程并指定设置。还可在命令提示下运行某些处理过程。

相关链接

- [Tcl Scripting](#)
Intel Quartus Prime Standard Edition 用户指南: 脚本
- [Command Line Scripting](#)
Intel Quartus Prime Standard Edition 用户指南: 脚本

5.4.1. Initializing and Uninitializing a Logic Lock (Standard) Region

You must initialize the Logic Lock (Standard) data structures before creating or modifying any Logic Lock (Standard) regions and before executing any of the Tcl commands listed below.

Use the following Tcl command to initialize the Logic Lock (Standard) data structures:

```
initialize_logiclock
```

Use the following Tcl command to uninitialize the Logic Lock (Standard) data structures before closing your project:

```
uninitialize_logiclock
```

5.4.2. Creating or Modifying Logic Lock (Standard) Regions

Use the following Tcl command to create or modify a Logic Lock (Standard) region:

```
set_logiclock -auto_size true -floating true -region <my_region-name>
```

注意:

The command in the above example sets the size of the region to auto and the state to floating.

If you specify a region name that does not exist in the design, the command creates the region with the specified properties. If you specify the name of an existing region, the command changes all properties you specify and leaves unspecified properties unchanged.

[相关链接](#)

创建 [Logic Lock \(Standard\) 区域](#) (第 104 页)

5.4.3. Obtaining Logic Lock (Standard) Region Properties

Use the following Tcl command to obtain Logic Lock (Standard) region properties. This example returns the height of the region named `my_region`:

```
get_logiclock -region my_region -height
```

5.4.4. Assigning Logic Lock (Standard) Region Content

Use the following Tcl commands to assign or change nodes and entities in a Logic Lock (Standard) region. This example assigns all nodes with names matching `fifo*` to the region named `my_region`.

```
set_logiclock_contents -region my_region -to fifo*
```

You can also make path-based assignments with the following Tcl command:

```
set_logiclock_contents -region my_region -from fifo -to ram*
```

5.4.5. Save a Node-Level Netlist for the Entire Design into a Persistent Source File

Make the following assignments to cause the Intel Quartus Prime Fitter to save a node-level netlist for the entire design into a `.vqm` file:

```
set_global_assignment-name LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT ON  
set_global_assignment-name LOGICLOCK_INCREMENTAL_COMPILE_FILE <file name>
```

Any path specified in the file name is relative to the project directory. For example, specifying `atom_netlists/top.vqm` places `top.vqm` in the **atom_netlists** subdirectory of your project directory.

A `.vqm` file is saved in the directory specified at the completion of a full compilation.

注意: The saving of a node-level netlist to a persistent source file is not supported for designs targeting newer devices such as MAX V , Stratix IV, or Stratix V.

5.4.6. Setting Logic Lock (Standard) Assignment Priority

Use the following Tcl code to set the priority for a Logic Lock (Standard) region' s members. This example reverses the priorities of the Logic Lock (Standard) region in your design.

```
set reverse [list]
for each member [get_logiclock_member_priority] {
set reverse [insert $reverse 0 $member]
}
set_logiclock_member_priority $reverse
```

5.4.7. 通过 Tcl 命令约束虚拟管脚

使用以下 Tcl 命令开启名为 my_pin 的管脚的虚拟管脚设置:

```
set_instance_assignment -name VIRTUAL_PIN ON -to my_pin
```

相关链接

- [虚拟管脚](#) (第 111 页)
- [管理器件 I/O 管脚](#) (第 0 页)
- [Node Finder Command](#) ([查看](#))
Intel Quartus Prime Help 中

5.5. 分析和优化设计布局规划修订历史

本章修订历史如下:

表 20. 文档修订历史

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|----------------|------------------------|--|
| 2018.09.24 | 18.1.0 | <ul style="list-style-type: none"> Intel Quartus Prime Standard 版用户指南首次发布。 重命名主题: 将生成扇入和扇出连接更改为查看已布局资源的扇入和扇出连接。 |
| 2018 年 5 月 7 日 | 18.0.0 | <ul style="list-style-type: none"> 添加了关于使用迭代法进行布局规划的建议。 |
| 2017.11.06 | 17.1.0 | <ul style="list-style-type: none"> 将 <i>LogicLock</i> 实例更改为 <i>Logic Lock (Standard)</i>。 |
| 2017.05.08 | 17.0.0 | <ul style="list-style-type: none"> 章节结构重组和内容更新。 添加了图示: 时钟区域, Creating a Hole in a LogicLock Region, Noncontiguous LogicLock Region, Routing Regions, Logic Placed Outside of an Empty Region。 移动主题: 将查看关键路径移动到时序收敛和优化章节并重命名为关键路径。 重命名主题: 将创建 <i>Non-Rectangular LogicLock Plus</i> 区域重命名为合并 <i>LogicLock Plus</i> 区域。 重命名主题: 将 <i>Chip Planner</i> 概述重命名为 <i>Chip Planner</i> 中的设计布局规划。 重命名章节, 从使用 <i>Chip Planner</i> 分析和优化设计布局规划重命名为分析和优化设计布局规划。 |
| 2015.11.02 | 15.1.0 | <ul style="list-style-type: none"> 将 <i>Quartus II</i> 更改为 <i>Quartus Prime</i>。 |
| 继续... | | |

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|-------------|------------------------|--|
| 2015.05.04 | 15.0.0 | 添加关于 LogicLock 区域颜色代码的信息。 |
| 2014.12.15 | 14.1.0 | 更新了 Virtual Pins 约束的说明，以阐明约束的输入不可用。 |
| 2014 年 6 月 | 14.0.0 | 更新格式 |
| 2013 年 11 月 | 13.1.0 | 删除了 HardCopy 器件信息。 |
| 2013 年 5 月 | 13.0.0 | 更新了“查看布线拥塞”部分 更新了对 Chip Planner Quartus UI 空间的参考文献 |
| 2012 年 6 月 | 12.0.0 | 删除了反馈问卷链接。 |
| 2011 年 11 月 | 11.0.1 | 文档模板更新。 |
| 2011 年 5 月 | 11.0.0 | <ul style="list-style-type: none"> 更新了 11.0 发布。 编辑了“LogicLock 区域” 更新了“查看布线拥塞” 更新了“查找历史记录” 更新了图示 15-4、15-9、15-10 和 15-13 添加图示 15-6 |
| 2010 年 12 月 | 10.1.0 | <ul style="list-style-type: none"> 更新了 10.1 发布。 |
| 2010 年 7 月 | 10.0.0 | <ul style="list-style-type: none"> 更新了器件支持信息。 删除了关于 Timing Closure Floorplan 的参考文献；删除了“使用 Timing Closure Floorplan 设计分析”部分。 添加了在线 Help 主题的链接 添加了“通过 Design Partition Planner 使用 LogicLock 区域”部分 更新了“查看关键路径”部分 更新了多个图形 更新了文档修订历史表格的格式 |
| 2009 年 11 月 | 9.1.0 | <ul style="list-style-type: none"> 全篇更新支持器件的信息。 删除了关于旧器件系列的 Timing Closure Floorplan 弃用部分。（关于使用就器件系列的 Timing Closure Floorplan 的信息，请参阅文本存档中的早前版本 Quartus Prime 手册。） 更新了“创建非矩形 LogicLock 区域”部分 添加了“已选元件窗口”部分 更新了表格 12-1 |
| 2008 年 5 月 | 8.0.0 | <ul style="list-style-type: none"> 更新了以下部分： <ul style="list-style-type: none"> “Chip Planner 任务和层次” “LogicLock 区域” “反向标注 LogicLock 区域” “Timing Closure Floorplan 中的 LogicLock Regions 区域” 添加以下区域： <ul style="list-style-type: none"> “保留 LogicLock 区域” “创建非矩形 LogicLock 区域” “查看器件中可用的时钟网络” 更新了表格 10-1 删除了以下部分： <ul style="list-style-type: none"> 使用 Timing Closure Floorplan 保留 LogicLock 区域设计分析 |

相关链接

文档存档

了解 *Intel Quartus Prime* 手册以前的版本，请搜索文档存档。

6. 网表优化和物理综合

Intel Quartus Prime 软件提供的网表和物理综合优化可提高设计性能。适配期间单击使能物理综合选项。本章也提供关于应用网表和物理综合选项，以及通过“反向标注”（back-annotation）保留编译结果的指导。

表 21. 网表优化和物理综合选项

| 选项 | 位置/说明 |
|--------------------------------------|--|
| Enable physical synthesis options. | Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) 。物理综合优化应用于编译流程的不同阶段，如综合，适配期间或两者兼顾。 |
| Enable netlist optimization options. | Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis) 。通过以特定原语说明设计的设计原子网表进行网表优化操作。原子网表文件可以是 Electronic Design Interchange Format (.edf) 文件或由第三方综合工具生成的 Verilog Quartus Mapping (.vqm) 文件。Intel Quartus Prime 综合生成并在内部使用原子网表 |

注意： 由于设计中的原语节点名称在使用物理综合优化时可能发生改变，因而需评估您的设计是否依赖固定的节点名称。如果您使用的验证流程要求固定节点名称，如 **Signal Tap Logic Analyzer**，正是验证或基于 **Logic Lock (Standard)** 的优化流程（对于 legacy 器件），则禁用物理综合选项。

6.1. 物理综合优化

Intel Quartus Prime Fitter 对逻辑单元进行布局和布线，以确保逻辑的关键部分相互靠近且使用尽可能快的布线资源。然而布线延迟通常是常规关键路径延迟的主要部分。物理综合优化通过考量布局信息，布线延迟和时序信息以确定最佳布局。随后 Fitter 着重针对设计关键部分进行时序驱动优化。综合与适配过程的紧密整合被称为物理综合。

Some physical synthesis options affect only registered logic, while others affect only combinational logic. Select options based on whether you want to keep the registers intact. For example, if your verification flow involves formal verification, you might want to keep the registers intact.

以下部分说明 Intel Quartus Prime 软件中可用的物理综合优化，以及其如何帮助提高所选器件的性能和进行适配。

相关链接

[Compiler Settings Page \(Settings Dialog Box\)](#)

Intel Quartus Prime Help 中

6.1.1. 使能物理综合优化

物理综合优化可通过执行组合和顺序优化以及寄存器复制来改善电路性能。

使能物理综合选项：

1. 单击 **Assignments > Settings > Compiler Settings**。
2. To enable physical synthesis, click **Advanced Settings (Fitter)**, and then enable **Perform Physical Synthesis for Combinational Logic for Performance** and **Perform Physical Synthesis for Combinational Logic for Fitting**.
3. 在 **Netlist Optimizations** 报告中查看物理综合结果。

6.1.2. 物理综合选项

Intel Quartus Prime 软件提供物理综合优化选项来提高适配结果。要访问这些选项，单击 **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**。

注意: 针对设计中特定组件禁用全局物理综合优化，请将特定节点或实体的 **Netlist Optimizations** 逻辑选项约束为 **Never Allow**。

表 22. 物理综合选项

| 选项 | 说明 |
|--|---|
| Perform asynchronous signal pipelining (no Intel Arria 10 support) | Automatically inserts pipeline stages for asynchronous clear and asynchronous load signals during fitting to increase circuit performance. This option is useful for asynchronous signals that are failing recovery and removal timing because they feed registers using a high-speed clock. You can use this option if asynchronous control signal recovery and removal times are not achieving requirements. This option adds registers and potential latency to nets driving the asynchronous clear or asynchronous load ports of registers. The additional register delays can change the behavior of the signal in the design; therefore, you should use this option only if additional latency on the reset signals does not violate any design requirements. This option also prevents the promotion of signals to global routing resources. |
| Perform Register Duplication for Performance (no Intel Arria 10 support) | Duplicates registers based on Fitter placement information to reduce the delay of one path without degrading the delay of another. You can also duplicate combinational logic when you enable this option. The Fitter can place the new logic cell closer to critical logic without affecting the other fan-out paths of the original logic cell. This setting does not apply to logic cells that are part of a chain, drive global signals, are constrained to a single LAB, or the Netlist Optimizations option set to Never Allow . |
| Perform Register Retiming for Performance (no Arria 10 support) | Enables the movement of registers across combinational logic, allowing the Quartus Prime software to trade off the delay between timing-critical paths and non-critical paths. |
| Perform Physical synthesis for combinational logic for Performance (no Intel Arria 10 support) | Performs physical synthesis optimizations on combinational logic during synthesis and fitting to increase circuit performance. Swaps the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. Also allows the duplication of LUTs to enable further optimizations on the critical path. |
| Physical Synthesis for Combinational Logic for Fitting (no Intel Arria 10 support) | Reduces delay along critical paths. This option swaps the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. The option also allows the duplication of LUTs to enable further optimizations on the critical path. The option causes registers that do not have a Power-Up Level logic option setting to power up with a don't care logic level (x). When the Power-Up Don't Care option is turned on, the Compiler determines when it is beneficial to change the power-up level of a register to minimize the area of the design. A power-up state of zero is maintained unless there is an immediate area advantage. The registers contained in the affected logic cells are not modified. Inputs into memory blocks, DSP blocks, and I/O elements (IOEs) are not swapped. This setting does not apply to logic cells that are part of a chain, drive global signals, are constrained to a single LAB, or the Netlist Optimizations option set to Never Allow . |
| Perform WYSIWYG Primitive Resynthesis | Specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the Optimization Technique logic option. |

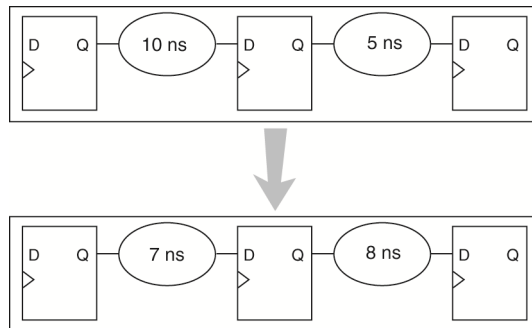
继续...

| 选项 | 说明 |
|--|--|
| Physical Synthesis Effort Level (no Intel Arria 10 support) | Specifies the amount of effort, in terms of compile time, physical synthesis should use. Compared to the Default setting, a setting of Extra uses extra compile time to try to gain extra circuit performance. Conversely, a setting of Fast uses less compile time but may reduce the performance gain that physical synthesis is able to achieve. |
| Netlist Optimizations (网表优化) | 可使用 Assignment Editor 应用 Netlist Optimizations 逻辑选项。使用该选项禁用部分设计的物理综合优化。 |
| Allow Register Duplication (允许寄存器复制) | 允许 Compiler 复制寄存器以提高设计性能。使能该选项时, Compiler 复制该寄存器并将其中一些扇出移动到对应的新节点。该优化可改善布通性, 并减少具有多个扇出的网络中的布线线缆总数。如果禁用该选项, 则会禁用重定时寄存器的优化。本设置会影响 Analysis & Synthesis 和 Fitter。 |
| Allow Register Duplication (允许寄存器合并) | 允许 Compiler 删除与设计其他寄存器相同的寄存器。使能该选项后, 如果两个寄存器生成相同逻辑, Compiler 删除一个寄存器并将寄存器扇出保留在被删除寄存器的目的地。该选项有助于防止 Compiler 删除特意使用的复制寄存器。如果禁用寄存器合并, 则 Compiler 禁用重定时寄存器的优化。本设置会影响 Analysis & Synthesis 和 Fitter。 |

6.1.3. Perform Register Retiming for Performance

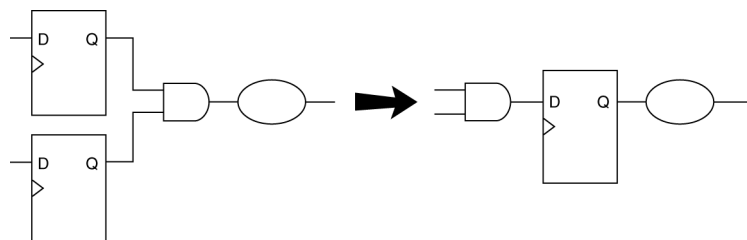
The **Perform Register Retiming for Performance** option enables the movement of registers across combinational logic, allowing the Intel Quartus Prime software to trade off the delay between timing-critical paths and non-critical paths. Register retiming can be done during Intel Quartus Prime integrated synthesis or during the Fitter stages of design compilation.

图 49. Reducing Critical Delay by Moving the Register Relative to Combinational Logic



Retiming can create multiple registers at the input of a combinational block from a register at the output of a combinational block. In this case, the new registers have the same clock and clock enable. The asynchronous control signals and power-up level are derived from previous registers to provide equivalent functionality. Retiming can also combine multiple registers at the input of a combinational block to a single register.

图 50. Combining Registers with Register Retiming



To move registers across combinational logic to balance timing, click **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**. Specify your preferred option under **Optimize for performance (physical synthesis)** and **Effort level**.

6.1.4. Preventing Register Movement During Retiming

If you want to prevent register movement during register retiming, you can set the **Netlist Optimizations** logic option to **Never Allow**. You can apply this option to either individual registers or entities in the design using the Assignment Editor.

In digital circuits, synchronization registers are instantiated on cross clock domain paths to reduce the possibility of metastability. The Intel Quartus Prime software detects such synchronization registers and does not move them, even if register retiming is turned on.

The following sets of registers are not moved during register retiming:

- Both registers in a direct connection from input pin-to-register-to-register if both registers have the same clock and the first register does not fan-out to anywhere else. These registers are considered synchronization registers.
- Both registers in a direct connection from register-to-register if both registers have the same clock, the first register does not fan out to anywhere else, and the first register is fed by another register in a different clock domain (directly or through combinational logic). These registers are considered synchronization registers.

The Intel Quartus Prime software does not perform register retiming on logic cells that have the following properties:

- Are part of a cascade chain
- Contain registers that drive asynchronous control signals on another register
- Contain registers that drive the clock of another register
- Contain registers that drive a register in another clock domain
- Contain registers that are driven by a register in another clock domain

注意:

The Intel Quartus Prime software does not usually retime registers across different clock domains; however, if you use the Classic Timing Analyzer and specify a global f_{MAX} requirement, the Intel Quartus Prime software interprets all clocks as related. Consequently, the Intel Quartus Prime software might try to retime register-to-register paths associated with different clocks.

To avoid this circumstance, provide individual f_{MAX} requirements to each clock when using Classic Timing Analysis. When you constrain each clock individually, the Intel Quartus Prime software assumes no relationship between different clock domains and considers each clock domain to be asynchronous to other clock domains; hence no register-to-register paths crossing clock domains are retimed.

When you use the Timing Analyzer, register-to-register paths across clock domains are never retimed, because the Timing Analyzer treats all clock domains as asynchronous to each other unless they are intentionally grouped.

- Contain registers that are constrained to a single LAB location
- Contain registers that are connected to SERDES
- Are considered virtual I/O pins
- Registers that have the **Netlist Optimizations** logic option set to **Never Allow**

The Intel Quartus Prime software assumes that a synchronization register chain consists of two registers. If your design has synchronization register chains with more than two registers, you must indicate the number of registers in your synchronization chains so that they are not affected by register retiming. To do this, perform the following steps:

1. Click **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)**.
2. Modify the **Synchronization Register Chain Length** setting to match the synchronization register length used in your design. If you set a value of 1 for the **Synchronization Register Chain Length**, it means that any registers connected to the first register in a register-to-register connection can be moved during retiming. A value of $n > 1$ means that any registers in a sequence of length 1, 2, ..., n are not moved during register retiming.

If you want to consider logic cells that meet any of these conditions for physical synthesis, you can override these rules by setting the **Netlist Optimizations** logic option to **Always Allow** on a given set of registers.

相关链接

[分析和优化设计平面布局规划 \(第 93 页\)](#)

6.2. 应用网表优化

使用网表优化时的性能提高情况取决于具体设计。如果以重新组织设计结构来平衡关键路径延迟, 则可能通过网表优化的性能提高度最低。

您可能需要尝试可用选项来查看最适合特定设计的设置组合。请参阅编译报告中的消息查看每个选项的改进程度, 并帮助您确定是否应开启给定选项或特定工作级别。

开启更多网表优化选项可能会导致设计中的节点名称发生更多变化; 因而如果您正在使用验证流程则需谨记, 如 **Signal Tap Logic Analyzer** 或正式验证就需要固定或已知的节点名称。

Applying all the physical synthesis options at the **Extra** effort level generally produces the best results for those options, but adds significantly to the compilation time. You can also use the **Physical synthesis effort level** options to decrease the compilation time. The WYSIWYG primitive resynthesis option does not add much compilation time relative to the overall design compilation time.

为获得最佳效果，可使用 Intel Quartus Prime Design Space Explorer II (DSE)应用各种网表优化选项集。

相关链接

[Design Space Explorer II](#) (第 12 页)

6.2.1. WYSIWYG 原语再综合

对于使用第三方工具进行综合的设计，**Perform WYSIWYG primitive resynthesis** 选项允许将优化应用于已综合的网表。

Perform WYSIWYG primitive resynthesis 选项指示 Intel Quartus Prime 软件取消将原子网表中的逻辑元件 (LE) 映射到逻辑门，然后将门控重新映射回 Intel 指定原语。第三方综合工具生成使用 Intel 指定原语的 .edf 或 .vqm 原子网表文件。开启 **Perform WYSIWYG primitive resynthesis** 选项后，Intel Quartus Prime 软件在重新映射进程中使用特定于器件的技术。该功能使用针对工程 (**Speed**, **Area** 或 **Balanced**) 的 **Optimization Technique** 对设计重新映射。

Perform WYSIWYG primitive resynthesis 选项仅对逻辑单元（也称为 LCELL 或 LE 原语）和常规 I/O 原语（可能包含寄存器）取消映射或重新映射。双数据率 (DDR) I/O 原语，存储器原语，数字信号处理 (DSP) 原语和进位/级联链中的逻辑单元不被重新映射。该进程不处理加密 .vqm 文件或 .edf 文件中的特定逻辑，如第三方知识产权 (IP)。

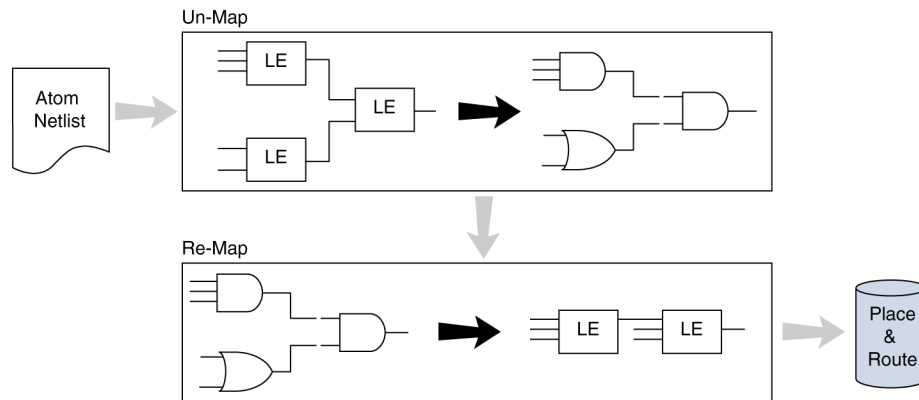
Perform WYSIWYG primitive resynthesis 选项可从第三方综合工具更改 .vqm 文件或 .edf 文件中的节点名称，因为原子网表中的原语是拆分状态随后由 Intel Quartus Prime 软件对其重新映射。重映射处理会删除重复的寄存器。未被删除的寄存器在重映射后保留相同名称。

任何将 **Netlist Optimizations** 逻辑选项设置为 **Never Allow** 的节点或实体在 WYSIWYG 原语重新综合期间都不受影响。可使用 Assignment Editor 应用 **Netlist Optimizations** 逻辑选项。该选项对设计某些部分禁用 WYSIWYG 再综合。

注意: 综合期间指定原语节点名称。应用网表优化时，节点名称可能会因为原语的创建和删除而更改。无法维护用于保留第三方综合工具中逻辑的 HDL 属性，因为这些属性并未写入 Intel Quartus Prime 软件要读取的原子网表。

如果使用 Intel Quartus Prime 软件综合设计，则可使用 **Preserve Register (preserve)** 和 **Keep Combinational Logic (keep)** 属性维持设计中的某些节点。

图 51. WYSIWYG 原语再综合的 Intel Quartus Prime 流程



6.2.2. Saving a Node-Level Netlist

For non- Intel Arria 10 designs, you can preserve a node-level netlist in Verilog Quartus Mapping File (.vqm) format. You might need to preserve nodes if you use the Logic Lock (Standard) flow to back-annotate placement, import one design into another, or both. For all device families that support incremental compilation, you can use this feature to preserve compilation results.

注意: This feature does not support Intel Arria 10 devices.

Use the **Export version-compatible database** option to save synthesis results as an atom-based netlist in .vqm file format. By default, the Intel Quartus Prime software places the .vqm in the **atom_netlists** directory under the current project directory.

If you use the physical synthesis optimizations and want to lock down the location of all LEs and other device resources in the design with the **Back-Annotate Assignments** command, a .vqm file netlist is required. The .vqm file preserves the changes that you made to your original netlist. Because the physical synthesis optimizations depend on the placement of the nodes in the design, back-annotating the placement changes the results from physical synthesis. Changing the results means that node names are different, and your back-annotated locations are no longer valid.

You should not use an Intel Quartus Prime-generated .vqm file or back-annotated location assignments with physical synthesis optimizations unless you have finalized the design. Making any changes to the design invalidates your physical synthesis results and back-annotated location assignments. If you require changes later, use the new source HDL code as your input files, and remove the back-annotated assignments corresponding to the Intel Quartus Prime-generated .vqm file.

To back-annotate logic locations for a design that was compiled with physical synthesis optimizations, first create a .vqm file. When recompiling the design with the hard logic location assignments, use the new .vqm file as the input source file and turn off the physical synthesis optimizations for the new compilation.

If you are importing a .vqm file and back-annotated locations into another project that has any **Netlist Optimizations** turned on, you must apply the **Never Allow** constraint to make sure node names don't change; otherwise, the back-annotated location or Logic Lock (Standard) assignments are invalid.

To preserve the nodes from Intel Quartus Prime physical synthesis optimization options for devices that do not support incremental compilation, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Compilation Process Settings**. The **Compilation Process Settings** page appears.
3. Turn on **Export version-compatible database**. This setting is not available for some devices.
4. Click **OK**.

6.3. Viewing Synthesis and Netlist Optimization Reports

Physical synthesis optimizations performed during synthesis write results to the synthesis report. To access this report, perform the following steps:

1. On the Processing menu, click **Compilation Report**.
2. In the **Compilation Report** list, open the **Analysis & Synthesis** folder to view synthesis results.
3. In the **Compilation Report** list, open the **Fitter** folder to view the **Netlist Optimizations** table.

6.4. 脚本支持

可运行本章节 Tcl 脚本中介绍的过程及设置。还可按照命令提示运行一些处理过程。关于脚本命令选项的详细信息，请参阅 [Intel Quartus Prime Command-Line](#) 和 [Tcl API Help](#) 浏览器。要运行 Help 浏览器，请在提示命令键入如下命令：

```
quartus_sh --qhelp
```

可在实例中或/和全局级中指定本小节中介绍的多个选项。

使用以下 Tcl 命令进行全局约束：

```
set_global_assignment -name <QSF variable name> <value>
```

使用以下 Tcl 命令进行实例约束：

```
set_instance_assignment -name <QSF variable name> <value> \  
-to <instance name>
```

相关链接

- [Command Line Scripting](#) (第 0 页)
- [Tcl Scripting](#) (第 0 页)
- [API Functions for Tcl](#)
Intel Quartus Prime Help 中

- Intel Quartus Prime Standard Edition 设置文件参考手册
关于 Intel Quartus Prime 软件中所有设置和约束的信息。

6.4.1. 综合网表优化

工程 .qsf 文件保留您在 GUI 中指定的设置。或者，可直接编辑 .qsf。 .qsf 文件支持如下综合网表优化命令。**Type** 栏标示是否支持设置作为全局设置，实例设置，或两者兼而有之。

表 23. 综合网表优化和相关设置

| 设置名称 | Intel Quartus Prime 设置文件变量名称 | 值 | 类型 |
|---|--|---|------------------|
| Perform WYSIWYG Primitive Resynthesis | ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP | ON, OFF | Global, Instance |
| Optimization Mode | OPTIMIZATION_MODE | BALANCED HIGH PERFORMANCE EFFOR AGGRESSIVE PERFORMANCE | Global, Instance |
| Power-Up Don't Care | ALLOW_POWER_UP_DONT_CARE | ON, OFF | Global |
| Save a node-level netlist into a persistent source file | LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT | ON, OFF | Global |
| | LOGICLOCK_INCREMENTAL_COMPILE_FILE | <file name> | |
| Allow Netlist Optimizations | ADV_NETLIST_OPT_ALLOWED | "ALWAYS ALLOW", DEFAULT, "NEVER ALLOW" | Instance |

6.4.2. 物理综合优化

工程 .qsf 文件保留您在 GUI 中指定的设置。或者，可直接编辑 .qsf。 .qsf 文件支持如下综合网表优化命令。**Type** 栏标示是否支持设置作为全局设置，实例设置，或两者兼而有之。

表 24. 物理综合优化和相关设置

| 设置名称 | Intel Quartus Prime 设置文件变量名称 | 值 | 类型 |
|--|---|---------|--------|
| Perform Physical Synthesis for Combinational Logic for Performance (no Arria 10 support) | PHYSICAL_SYNTHESIS_COMBO_LOGIC | ON, OFF | Global |
| Perform Physical Synthesis for Combinational Logic for Fitting (no Intel Arria 10 support) | PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA | ON, OFF | Global |
| Advanced Physical Synthesis | ADVANCED_PHYSICAL_SYNTHESIS | ON, OFF | Global |
| Automatic Asynchronous Signal Pipelining | PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING | ON, OFF | Global |
| 继续... | | | |

| 设置名称 | Intel Quartus Prime 设置文件变量名称 | 值 | 类型 |
|---|--|--|------------------|
| Perform Register Duplication for Performance (no Intel Arria 10 support) | PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION | ON, OFF | Global |
| Perform Register Retiming for Performance (no Intel Arria 10 support) | PHYSICAL_SYNTHESIS_REGISTER_RETIMING | ON, OFF | Global |
| Power-Up Don't Care | ALLOW_POWER_UP_DONT_CARE | ON, OFF | Global, Instance |
| Power-Up Level | POWER_UP_LEVEL | HIGH, LOW | Instance |
| Allow Netlist Optimizations | ADV_NETLIST_OPT_ALLOWED | "ALWAYS ALLOW", DEFAULT, "NEVER ALLOW" | Instance |
| Save a node-level netlist into a persistent source file (no Intel Arria 10 support) | LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT | ON, OFF | Global |
| | LOGICLOCK_INCREMENTAL_COMPILE_FILE | < file name > | |

6.4.3. Back-Annotating Assignments

You can use the `logiclock_back_annotate` Tcl command to back-annotate resources in your design. This command can back-annotate resources in Logic Lock (Standard) regions, and resources in designs without Logic Lock (Standard) regions.

The following Tcl command back-annotates all registers in your design:

```
logiclock_back_annotate -resource_filter "REGISTER"
```

The `logiclock_back_annotate` command is in the `backannotate` package.

6.5. 网表优化和物理综合修订历史

以下修订历史适用于本章：

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|------------|------------------------|--|
| 2018.09.24 | 18.1.0 | Intel Quartus Prime Standard 版用户指南首次发布。 |
| 2018.05.07 | 18.0.0 | 删除主题： <i>Isolating a Partition Netlist</i> 。 |
| 2017.1106 | 17.1.0 | <ul style="list-style-type: none"> 添加主题： <i>Isolating a Partition Netlist</i>。 |
| 2016.10.31 | 16.1.0 | <ul style="list-style-type: none"> 更新了物理综合选项和处理过程。 |
| 2016.05.02 | 16.0.0 | <ul style="list-style-type: none"> 阐述关于已淘汰的物理综合选项的限制。 |
| 2015.11.02 | 15.1.0 | <ul style="list-style-type: none"> 将 <i>Quartus II</i> 更改成 <i>Intel Quartus Prime</i>。 |
| 2014.12.15 | 14.1.0 | <ul style="list-style-type: none"> 将 Fitter Setting, Analysis & Synthesis Settings 和 Physical Synthesis Optimizations Setting 的位置更新到 Compiler Setting。 更新了 DSE II 的内容。 |
| 继续... | | |

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|-------------|------------------------|---|
| 2014 年 6 月 | 14.0.0 | 更新了格式。 |
| 2013 年 11 月 | 13.1.0 | 删除了 HardCopy 器件信息。 |
| 2012 年 6 月 | 12.0.0 | 删除了反馈问卷链接。 |
| 2011 年 11 月 | 10.0.2 | 文档模板更新。 |
| 2010 年 12 月 | 10.0.1 | 文档模板更新。 |
| 2010 年 7 月 | 10.0.0 | <ul style="list-style-type: none"> 在多个部分添加 Intel Quartus Prime Help 的链接。 删除“参考文档”部分。 重新编排文档修订历史 |
| 2009 年 11 月 | 9.1.0 | <ul style="list-style-type: none"> 对“寄存器物理综合-寄存器重新定时”添加信息。 对“应用网表优化选项”添加信息。 少量编辑更新。 |
| 2009 年 3 月 | 9.0.0 | <ul style="list-style-type: none"> 第 11 章于 8.1.0 发布。 更新了“寄存器的物理综合 — 寄存器重新定时”和“用于 Fitting 的物理综合选项” 更新了“执行物理综合优化” 删除了“门级寄存器重定时”部分。 更新了引用文档 |
| 2008 年 11 月 | 8.1.0 | 更改为“8½ × 11”页面尺寸。文档内容无变化。 |
| 2008 年 5 月 | 8.0.0 | <ul style="list-style-type: none"> 更新了第 11-9 页的“针对性能的物理综合优化” 添加了第 11-16 页的“针对 Fitting 的物理综合选项” |

相关链接

文档存档

欲了解 *Intel Quartus Prime* 手册以前的版本，请搜索文档存档。

7. 使用 Chip Planner 的工程变更命令

可编程逻辑可适应设计周期后期出现的系统规范变化。在典型工程项目开发周期中，可编程逻辑部分的规范可能在工程项目开始后或集成全部系统元件期间有所改变。最后关头的设计变更，通常称为工程变更命令（ECO，engineering change orders），是完全编译设计后对设计功能性的小规模针对性变更。

Chip Planner 支持 ECO，允许在支持器件的设计后期快速有效更改逻辑。Chip Planner 提供在所选 FPGA 器件体系结构中映射出的后布局布线设计的可视化显示，从而允许您创建，移动和删除逻辑单元和 I/O 原子。

注意: Intel Quartus Prime Standard Edition ECO 功能不支持 Intel Arria 10 器件。

除进行 ECO 外，Chip Planner 还允许对路由拥塞，相对资源使用，逻辑布局，Logic Lock (Standard) 区域，扇入和扇出，寄存器之间的路径和路径延迟估算执行详细分析。

ECO 直接应用于支持目标器件的原子。因此，执行一个 ECO 依赖于您对目标器件的器件体系结构的理解。

相关链接

- [分析和优化设计平面布局规划 \(第 93 页\)](#)
关于使用 Chip Planner 进行设计分析的更多信息。
- [文献](#)
关于器件体系结构的更多信息

7.1. 工程变更命令

FPGA 设计环境下，可将 ECO 直接应用于器件上的物力资源以修改其行为。通常在设计周期的验证阶段制定 ECO。当设计中需要小规模变更时（如，将 PLL 修改为另一时钟频率或将信号路由到进行分析的管脚）重新编译整个设计非常耗时，尤其是对于较大的设计。

由于验证周期中会发生几次小规模的设计变更迭代，因此重新编译时间快速增加。此外，由于小规模设计变更而造成的重新编译可能导致之前设计优化的丢失。制定 ECO，以取代对设计执行完整重新编译，可将改变限制于受影响的逻辑部分。

7.1.1. 性能保留

更改现有设计时，保留先前设计优化结果的方法如下：

- 增量式编译
- 快速重新编译
- ECOs

根据变更范围选择修改您设计的方法。上述方法从对被编译设计产生较大变更到最小目标变更排列。

增量式编译功能允许保留 RTL 组件或模块级编译结果。设计的初始编译后，可将设计层次中的模块约束到分区。后续编译时，增量编译根据所选的保留级别重新编译已变更的分区。

快速重编译功能利用最新 post-fit（适配后）网表的结果来确定在兑现对源代码的修改时所需的变更。如果运行快速重新编译，Compiler 仅重新适配网表中已变更的部分。

与增量编译和快速重新编译功能相比，ECO 提供更精细的控制粒度。所有修改都直接在器件的体系结构上执行。应使用 ECO 对适配后的网表进行针对性更改。

注意: Intel Quartus Prime 软件 10.0 及更新版本中，开启增量编译功能重新编译设计时，软件不保留对网表的 ECO 修改。可通过 Change Manager 重新应用之前编译期间制定的 ECO 变更。

相关链接

[Intel Quartus Prime 规划分层和基于团队设计的增量式编译](#)

7.1.2. 编译时间

传统可编程逻辑设计流程中，设计中的小型变化都需要彻底重新编译设计。完整的设计重新编译包括综合和布局布线。对设计进行小规模更改以实现板上最终实现也可能是一个漫长的过程。由于 Chip Planner 仅适用于布局布线后数据库，因此可在几分钟内实现设计更改，而无需执行完整编译。

7.1.3. 验证

设计更改后，可验证对设计的影响。要验证您的更改是否违法时序要求，请在 Chip Planner 中检查和保存您的网表更改后，通过 Intel Quartus Prime Timing Analyzer 执行静态时序分析。

此外，还可使用由 Intel Quartus Prime 软件生成的布局布线后网表执行 ECO 修改设计的门级或时序仿真。

相关链接

[Intel Quartus Prime 时序分析器](#)

7.1.4. 更改修改记录

所有通过 Chip Planner 制定的 ECO 都记录在 Change Manager 中以追踪所有更改。使用 Change Manager，可轻松恢复到原始适配后网表或选择要应用的 ECO。

此外，Intel Quartus Prime 软件为同一工程提供多个编译修订支持。可结合使用修订支持和 Chip Planner 中制定的 ECO 对比几种不同 ECO 更改，并在需要时恢复到之前的工程版本。

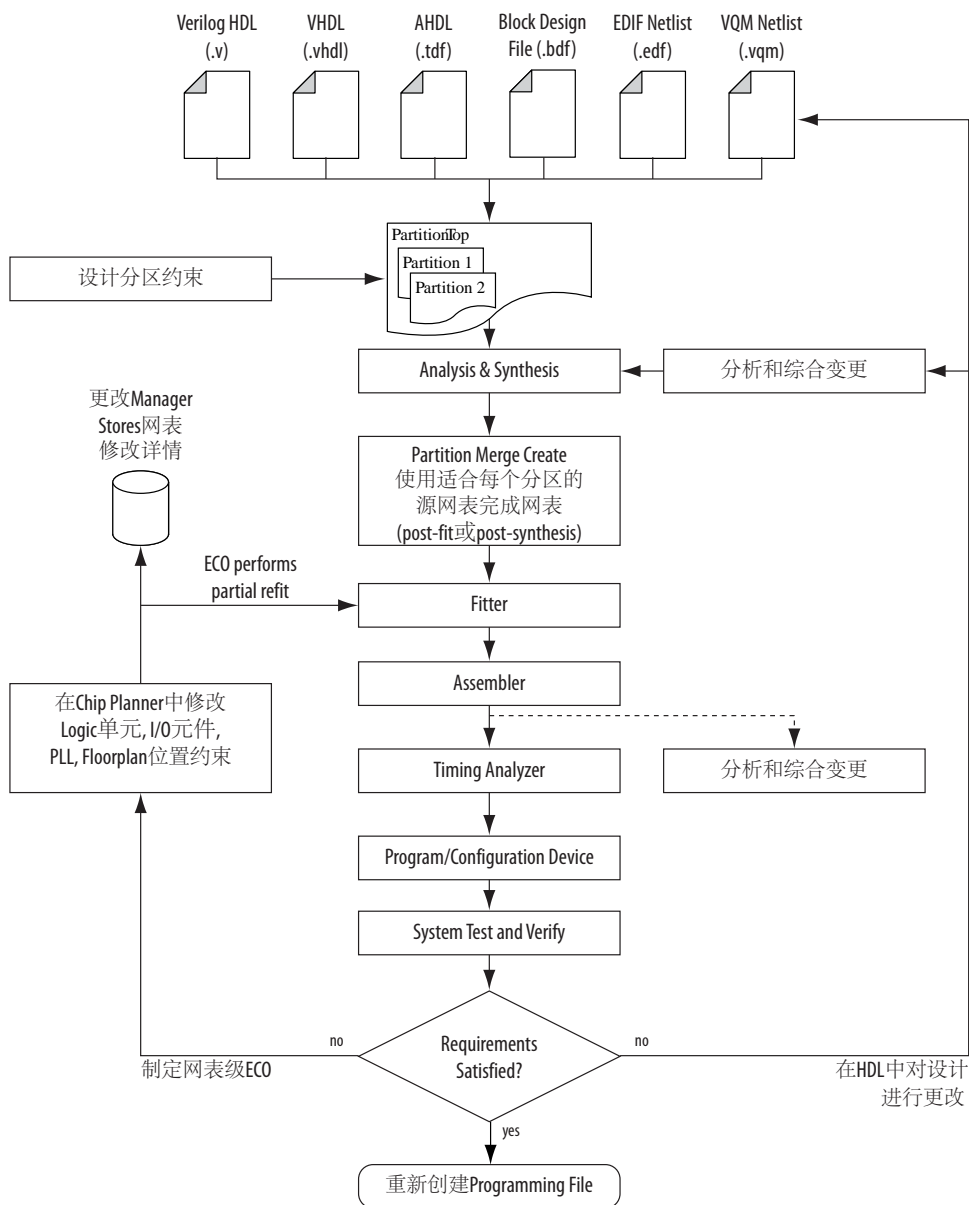
7.2. ECO 设计流程

对于迭代验证周期，实现小规模网表级设计变更比更改 RTL 代码更快。因此，在硅圆上调试设计并需要快速周转时间来生成用于调试设计的编程文件时，ECO 更改特别有帮助。

注意: Intel Quartus Prime Standard Edition ECO 功能不支持 Intel Arria 10 器件。

该图显示制定 ECO 的设计流程。

图 52. 支持 ECO 的设计流程



当发现电路板上的问题并将其隔离于器件上的相应节点或 I/O 单元时，就出现典型 ECO 应用。必须能够快速纠正功能性并生成新的编程文件。使用 **Chip Planner** 进行小规模更改，可直接修改布局布线后网表且无需执行综合和逻辑映射，从而缩短了验证周期内生成编程文件的周转时间。如果此改变纠正了问题，则无需修改 HDL 源代码。可使用 **Chip Planner** 对设计执行以下与 ECO 相关的更改：

- 使用 **Change Manager** 记录更改
- 轻松重新创建用于生成设计更改的步骤
- 生成 **EDA 仿真网表** 用于设计验证

注意: 对于需要修改 HDL 源代码的更复杂设计，增量编译功能可帮助缩短重新编译时间。

7.3. Chip Planner 概述

Chip Planner 提供器件资源的可视化显示。其显示您目标器件体系结构中资源原子的排列和使用。资源原子是器件的构建块，例如 ALMs、LE、PLL、DSP 块、存储块或 I/O 元件。

Chip Planner 还提供一个集成平台，用于分析设计，并在布局布线后为您的设计制定 ECO。该工具包由 Chip Planner（提供被映射设计的器件规划平面视图）和两个集成子工具—Resource Property Editor 和 Change Manager。

为进行分析，Chip Planner 可显示逻辑布局，Logic Lock (Standard) 区域，相对资源使用，详细布线信息，路线拥塞，扇入和扇出，寄存器之间的路径以及关于路径延迟的估计。此外，Chip Planner 允许创建位置约束或资源约束更改。例如根据器件布局规划移动或删除逻辑单元或 I/O 原子。对于 ECO 变更，Chip Planner 支持在布局布线后网表中创建，移动或删除逻辑元件删除编程文件，以快速生成编程文件。此外，还可从 Chip Planner 开启 Property Editor 以编辑资源原子属性或编辑资源原子之间的连接。对资源原子和连接的所有变更都被 Change Manager 自动记录。

7.3.1. 打开 Chip Planner

要打开 Chip Planner，Tools 菜单上，单击 **Chip Planner**。或者，在 Intel Quartus Prime 软件工具栏上单击 **Chip Planner** 图标。

或者，通过跨以下工具中的快捷菜单进行交叉探测从而开启 Chip Planner：

- Design Partition Planner
- Compilation Report
- Logic Lock (Standard) Regions window
- Technology Map Viewer
- Project Navigator window
- RTL source code
- Node Finder
- Simulation Report
- RTL Viewer
- Report Timing panel of the Timing Analyzer

7.3.2. Chip Planner 任务和分层

Chip Planner 允许设置任务快速实现 ECO 或操作器件布局规划分配。每个任务包含一个编辑模式和一组定制分层设置。

相关链接

- 在 [Resource Property Editor](#) 中执行 ECO (第 133 页)
- [分析和优化设计平面布局规划](#) (第 93 页)

7.4. 使用 Chip Planner (布局规划视图) 执行 ECO

选择 ECO 编辑模式时，可在 Chip Planner 中操作资源原子。

可使用 Chip Planner Floorplan 视图进行如下 ECO 变更：

- 创建原子
- 删除原子
- 移动现有原子

注意：为配置原子的属性，例如管理不同 LE/ALM 之间的连接，请使用 Resource Property Editor。

要在 Chip Planner 中选择 ECO 编辑模式，请在 Chip Planner 顶部的 **Editing Mode** 列表中选择 ECO 编辑模式。

相关链接

在 Resource Property Editor 中执行 ECO (第 133 页)

7.4.1. 创建，删除和移动原子

可使用 Chip Planner 在编译后的设计中创建，删除和移动原子。

7.4.2. 查看并保存网表变化

完成制定所有 ECO 后，可点击 Chip Planner 工具栏中的 **Check and Save Netlist Changes** 图标运行 Fitter 以合并各更改。Fitter 编译 ECO 更改，对设计执行设计规则检查，并生成编程文件。

7.5. 在 Resource Property Editor 中执行 ECO

可通过 Resource Property Editor 查看并编辑如下资源。

7.5.1. 逻辑单元

Altera® LE 包含一个用作功能生成器的 4 输入 LUT，可实现四个变量中的任何其中之一。此外，每个 LE 包含一个由 LUT 输出或由另一 LE 中生成的独立功能馈给的寄存器。

可使用 Resource Property Editor 查看并编辑 FPGA 中的任何 LE。要打开一个 LE 的 Resource Property Editor，可在 Project 菜单上，指向“**Locate**”并从如下查看器中点击 **Locate in Resource Property Editor**：

- RTL Viewer
- Technology Map Viewer
- Node Finder
- Chip Planner

关于特定器件系列的 LE 体系结构的更多信息，请参阅该器件系列手册或数据表。

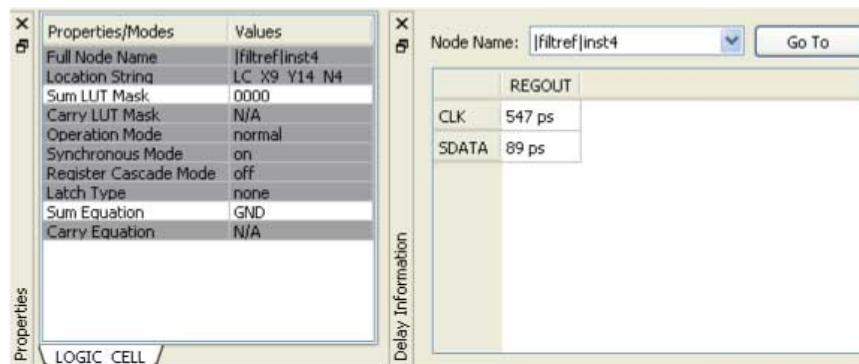
可使用 Resource Property Editor 更改以下 LE 属性：

- Data input to the LUT（数据输入到 LUT）
- LUT mask or LUT（LUT 掩码或 LUT）

7.5.1.1. 逻辑单元属性

查看逻辑单元属性，可在 View 菜单上，点击 **View Properties**。

图 53. Resource Property Editor 中的 LE 属性



7.5.1.2. 运行模式

LE 中的 LUT 可在普通模式或运算模式下运行。

在普通模式中配置 LE 后，LE 中的 LUT 可实现 4 输入功能。

在运算模式中配置 LE 后，LE 中的 LUT 被分成两个 3 输入 LUT。第一个 LUT 生成驱动 LUT 输入的信号，与此同时第二个 LUT 生成进位输出信号。进位输出信号仅可驱动其他 LE 的进位输入信号。

关于 LE 操作模式的更多信息，请参阅相应器件手册的第一卷。

7.5.1.3. 求和进位方程

可通过更改 sum 和 carry 方程来更改由 LUT 实现的逻辑功能。在普通模式中配置 LE 后，仅可更改求和方程。在运算模式中配置 LE 后，既可更改求和方程也可更改运算方程。

LUT 掩码是 LUT 方程输出的十六进制表示。更改 LUT 方程时，Intel Quartus Prime 软件自动更改 LUT 掩码。反之，更改 LUT 掩码，Intel Quartus Prime 软件自动计算 LUT 方程。

7.5.1.4. 同步加载和同步清除信号

每个 LE 寄存器包含一个同步加载 (sload) 信号和一个同步清除 (sclr) 信号。可将 sload 或 sclr 信号反转反馈给到 LE 中。

如果设计在 LE 中使用 sload 信号，则该信号与其反转状态必须与同一 LAB 中的所有其他 LE 相同。例如，如果 LAB 中的 2 个 LE 连接了 sload 信号，则 2 个 LE 的 sload 信号必须设置为相同值。对于 sclr 信号亦如此。

7.5.1.5. 寄存器级联模式

使能寄存器级联模式后，级联输入端口将输入馈给到寄存器。设计中实现移位寄存器后，最常使用寄存器级联模式。

可连接（或断开）端口中的级联来更改寄存器级联模式。但如果创建此端口，就必须确保源端口 LE 直接位于目标 LE 上方。

7.5.1.6. 单元延迟表格

单元延迟表格说明所选 LE 的所有输入到所有输出的传播延迟。

7.5.1.7. 逻辑单元连接

要查看馈给 LE 输入和输出的连接，请在 View 菜单上，单击 **View Port Connections**。

图 54. 在 Connectivity Window 中查看 LE 连接

| Input Port Name | Signal Name | Inverted | Output Port Name | Signal Name |
|-----------------|----------------------------------|----------|------------------|----------------|
| DATAA | <Disconnected> | False | COUT | <Disconnected> |
| DATAB | <Disconnected> | False | COMBOUT | <Disconnected> |
| SDATA | filtref state_minst1 filter_tap4 | False | REGOUT | filtref inst4 |
| DATAD | <Disconnected> | False | | |
| CIN | <Disconnected> | False | | |
| INVERTA | <Disconnected> | False | | |
| REGCASCIN | <Disconnected> | False | | |
| SLOAD | VCC | False | | |
| SCLR | <Disconnected> | False | | |
| IACLR | VCC | False | | |
| ALOAD | <Disconnected> | False | | |
| CLK | filtref clk | False | | |
| ENA | <Disconnected> | False | | |

7.5.1.8. 删除逻辑单元

请按照以下步骤删除 LE：

- 1. 在 Chip Planner 中右键点击需要的 LE，指向 **Locate**，并单击 **Locate in Resource Property Editor**。
- 2. 必须先删除 LE 上的所有扇出连接才可实现删除操作。要删除扇出连接，右键点击每个已连接的输出信号，指向 **Remove**，并单击 **Fanouts**。在 **Remove Fan-outs** 对话框中选中所有扇出信号并单击 **OK**。
- 3. 在所有扇出连接删除后删除原子，可在 Chip Planner 中右键单击该原子并单击 **Delete Atom**。

7.5.2. 自适应逻辑模块

每个 ALM 均包含基于 LUT 的资源，并可以 2 个自适应 LUT（ALUT）对其进行划分。

由于 ALUT 最多有 8 个输入，从而每个 ALM 可实现两个功能的各种组合。该适应性允许 ALM 完全向后兼容 4 输入 LUT 体系结构。一个 ALM 可实现最多 6 输入的任何功能以及某些 7 输入功能。此外，对于基于 ALUT 的资源，每个 ALM 包含 2 个可编程寄存器，2 个专用全加器，一个进位链，一个共享运算链和寄存器链。ALM 可利用这些专用资源有效实现各种运算功能和移位寄存器。

可在单个 ALM 中实现以下类型的功能：

- 2 个独立 4 输入功能
- 1 个独立 5 输入功能和一个独立 3 输入功能
- 1 个 5 输入功能和一个 4 输入功能（如果资源共享一个输入）
- 2 个 5 输入功能（如果资源共享 2 个输入）
- 1 个独立 6 输入功能
- 2 个 6 输入功能（如果资源共享 4 个输入并共享相同功能）
- 特定 7 输入功能

可使用 Resource Property Editor 更改如下 ALM 属性：

- Data input to the LUT（数据输入到 LUT）
- LUT mask or LUT equation（LUT 掩码或 LUT 方程）

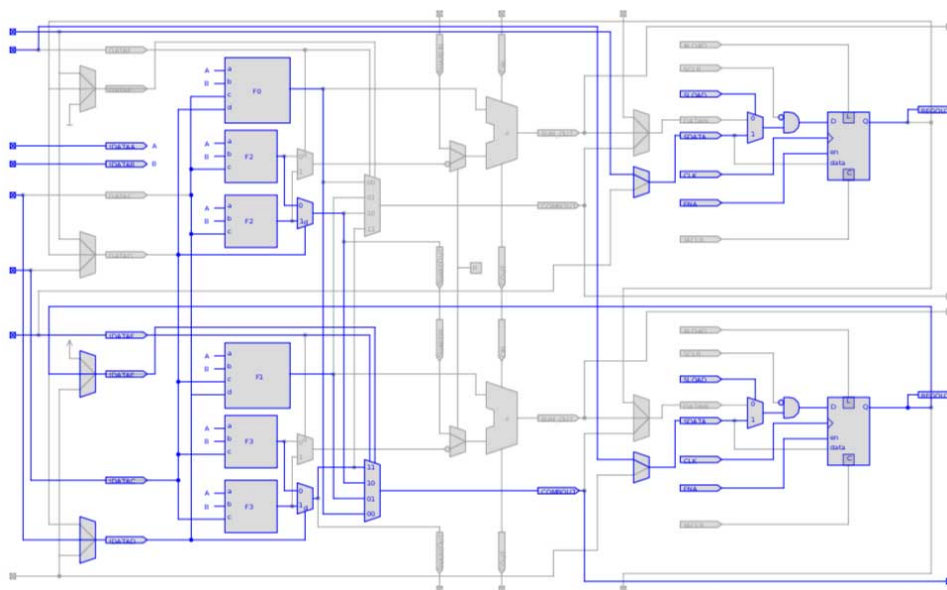
7.5.2.1. 自适应逻辑模块原理图

可使用 Resource Property Editor，在 RTL Viewer，Node Finder 或 Chip Planner 中右键单击 ALM 并点选 **Locate in Resource Property Editor** 查看并编辑 ALM 原子。

关于 ALM 的详细说明，请参阅基于 ALM 体系结构的器件的器件手册。

默认情况下，Intel Quartus Prime 软件以蓝色显示已使用的资源，以灰色显示未使用的资源。如下视图中，已使用资源着蓝色，未使用资源着灰色。

图 55. 自适应逻辑模块



7.5.2.2. 自适应逻辑模块属性

可显示的 ALM 属性包括：一个方程表说明 ALM 中 2 个组合节点和 2 个寄存器节点各自的名称和位置；每个组合节点的单个 LUT 方程；以及每个组合节点的 combout、sumout、carryout 和 shareout 方程。

7.5.2.3. 自适应逻辑模块连接

点击 **View > View Connectivity** 查看 ALM 的输入和输出连接。

7.5.3. FPGA I/O 元件

具有高性能 I/O 元件的 Altera FPGA 最多包含 6 个寄存器，具有多个 I/O 标准支持，因此支持您以最高速运行设计。使用 Resource Property Editor 查看，更改连接性并编辑 I/O 元件属性。使用 Chip Planner (Floorplan 视图) 更改布局，删除和创建新的 I/O 元件。

关于器件 I/O 元件的详细说明，请参阅适用的器件手册。

可更改如下 I/O 属性：

- Delay chain (延迟链)
- Bus hold (总线保持)
- Weak pull up (弱上拉)
- Slow slew rate (慢摆率)
- I/O standard (I/O 标准)
- Current strength (当前强度)
- Extend OE disable (扩展 OE 禁用)
- PCI I/O
- Register reset mode (寄存器复位模式)
- Register synchronous reset mode (寄存器同步复位模式)
- Register power up (寄存器上电)
- Register mode (寄存器模式)

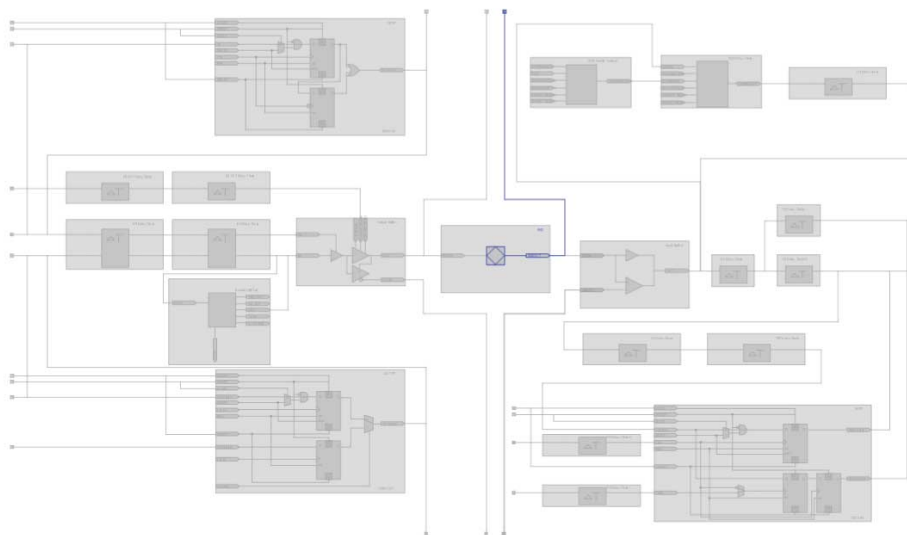
7.5.3.1. Stratix V 的 I/O 元件

Stratix® V 器件中的 I/O 元件包含一个双向 I/O 缓冲器和 I/O 寄存器可支持全嵌入双向信号数据率 (SDR) 或双数据率 (DDR) 传送。

I/O 寄存器由：处理从管脚到内核数据的输入路径，处理内核到管脚数据的输出路径，和处理输出使能信号到输出缓冲器的输出使能路径组成。这些寄存器允许更快的源同步“寄存器-寄存器”传送和再同步。输入路径由 DDR 输入寄存器，对齐和同步寄存器以及半数据速率块组成；可旁路输入路径中的每个块。输入路径使用去偏斜延迟 (deskew delay) 在调整处理，电压和温度 (PVT) 变化之中的输入寄存器时钟延迟。

默认情况下，Intel Quartus Prime 软件以蓝色显示已使用资源并以灰色显示未使用资源。

图 56. Stratix V 器件 I/O 元件结构



相关链接

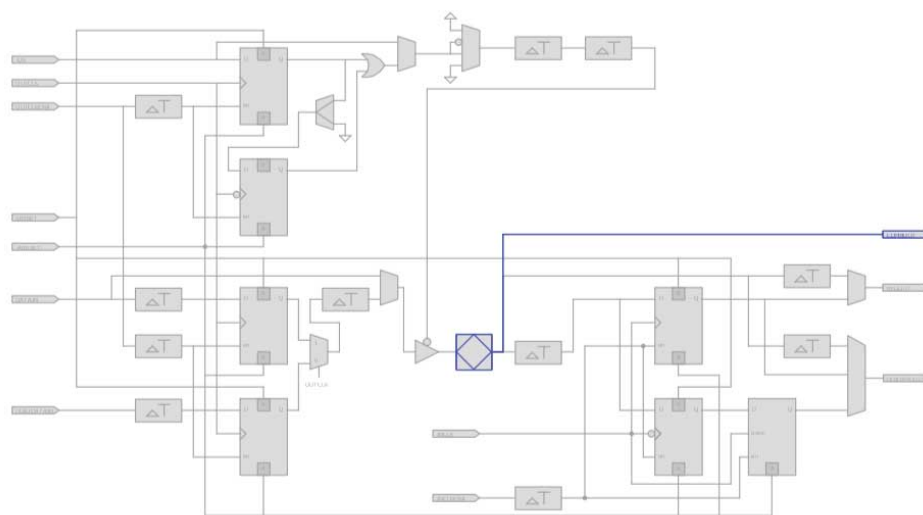
[Stratix V 器件手册](#)

7.5.3.2. Stratix IV 的 I/O 元件

Stratix IV 器件中的 I/O 元件包含一个双向 I/O 缓冲器和 I/O 寄存器以支持全嵌入双向 SDR 或 DDR 传送。

I/O 寄存器由：用于处理管脚到内核数据的输入路径，处理内核到管脚数据的输出路径，和处理输出缓冲器的输出使能信号的路径组成。每个路径包含一组延迟元件，支持通过微调每个路径时序特征的偏斜管理。默认情况下，Intel Quartus Prime 软件以蓝色显示已使用资源并以灰色显示未使用资源。

图 57. Stratix IV 的 I/O 元件和结构



相关链接

文献

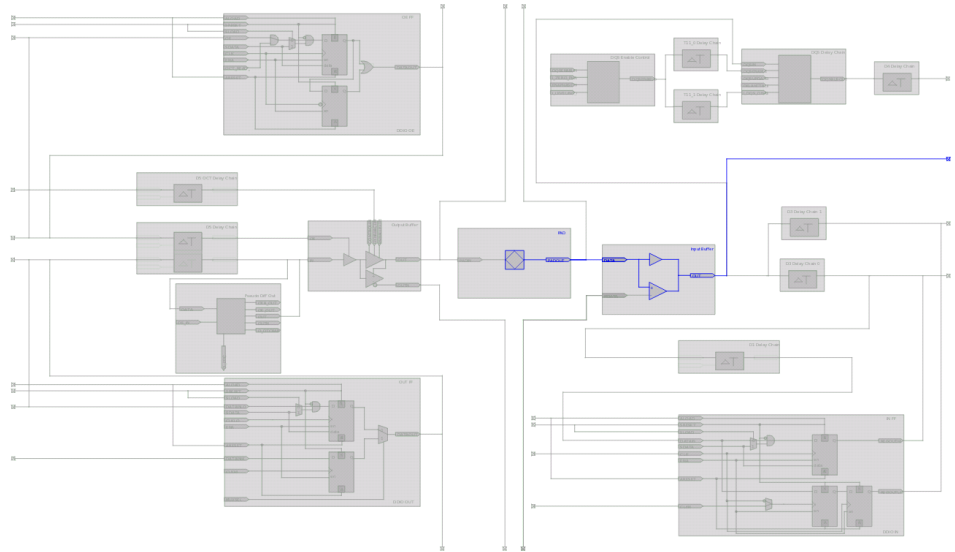
了解关于 Stratix IV 器件中 I/O 元件的信息

7.5.3.3. Arria V 的 I/O 元件

Arria® V 器件中的 I/O 元件包含一个双向 I/O 缓冲器和 I/O 寄存器以支持嵌入双向 SDR 或 DDR 传送。

I/O 寄存器由：用于处理管脚到内核数据的输入路径，处理内核到管脚数据的输出路径，和处理输出缓冲器的输出使能信号的路径组成。每个路径包含一组延迟元件，支持通过微调每个路径时序特征的偏斜管理。默认情况下，Intel Quartus Prime 软件以蓝色显示已使用资源并以灰色显示未使用资源。

图 58. Stratix IV 的 I/O 元件和结构

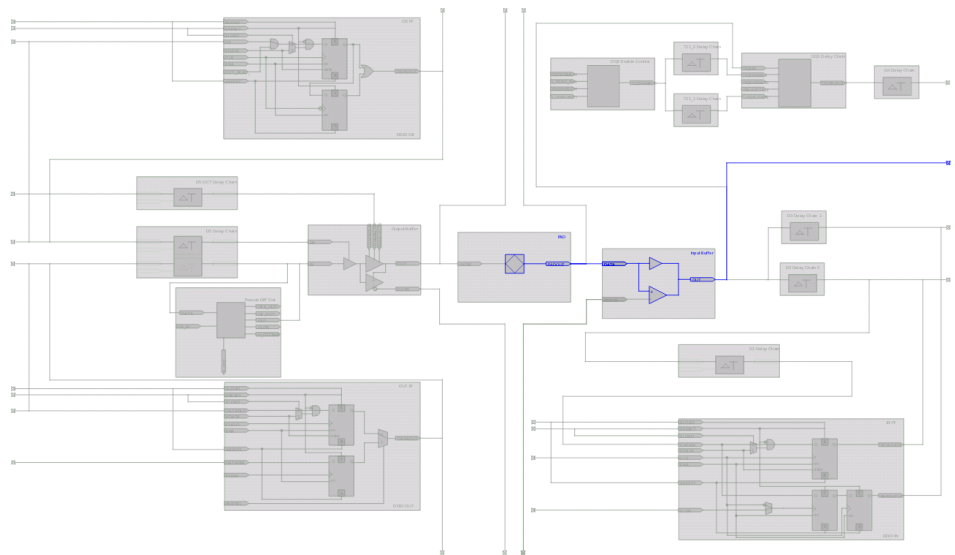


7.5.3.4. Cyclone IV 的 I/O 单元

Cyclone V 器件中的 I/O 元件含有用于全嵌入双向信号数据率传送的双向缓冲器和寄存器。I/O 元件包含 3 个输入寄存器，2 个输出寄存器和 2 个输出使能寄存器。2 个输出寄存器和 2 个输出使能寄存器用于双数据率（DDR）应用。

可使用输入寄存器实现快速设置时间，使用输出寄存器实现快速时钟-输出时间。此外，可使用输出使能（OE）寄存器实现快速时钟-输出使能时序。还可使用 I/O 元件进行输入，输出或双向数据路径。默认情况下，Intel Quartus Prime 软件以蓝色显示已使用资源并以灰色显示未使用资源。

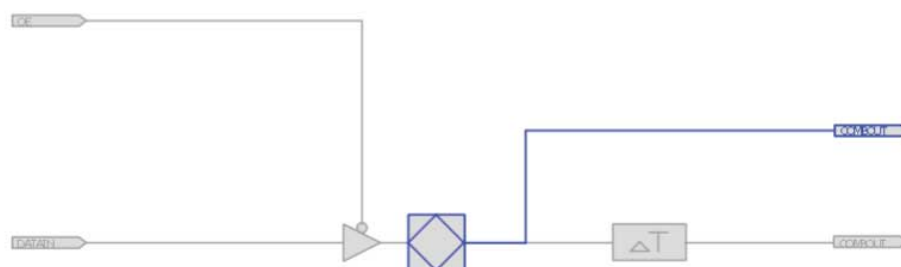
图 59. Cyclone V 器件的 I/O 元件和结构



7.5.3.5. MAX V 的 I/O 元件

MAX[®] V 器件中的 I/O 元件含有一个双向 I/O 缓冲器。可将相邻 LAB 的寄存器驱动到 I/O 元件的双向 I/O 缓冲器或从 I/O 元件的双向 I/O 缓冲器驱动相邻 LAB 的寄存器。默认情况下，Intel Quartus Prime 软件已蓝色显示已使用资源并以灰色显示未使用资源。

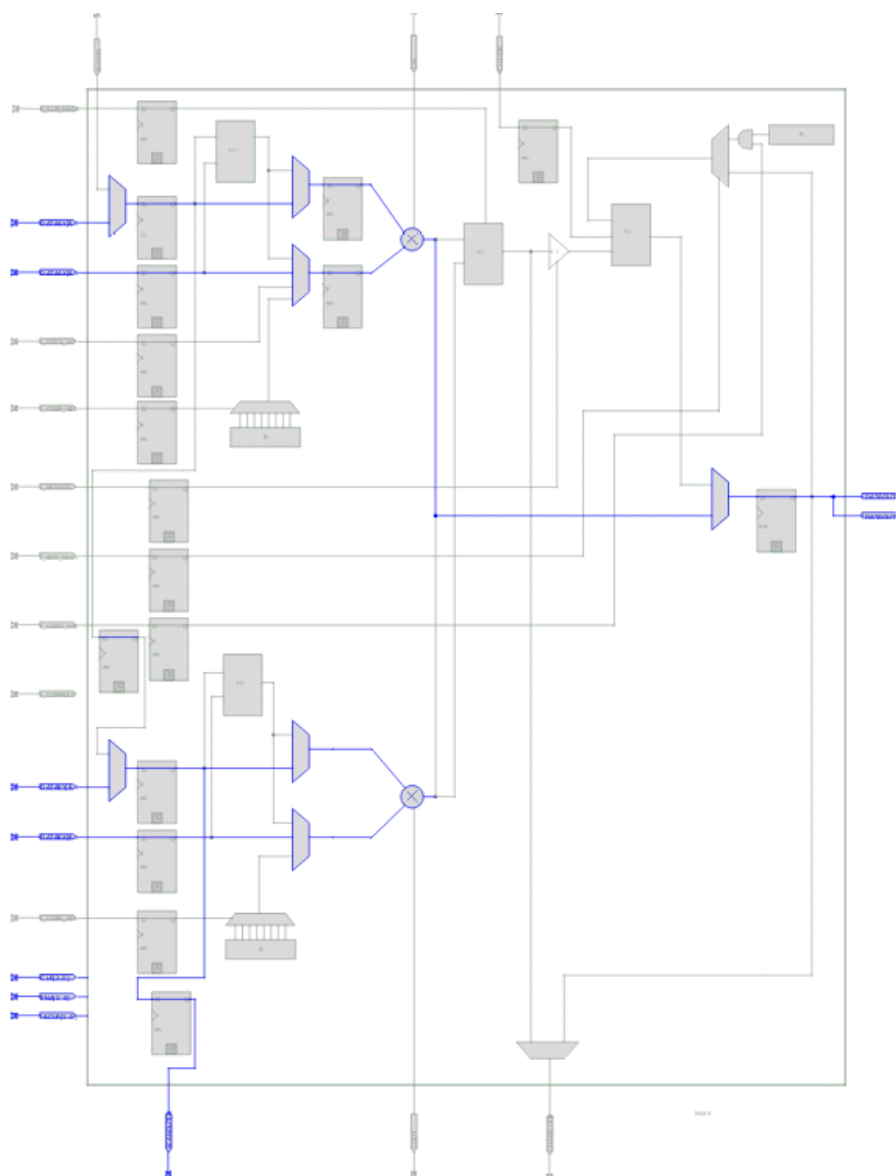
图 60. MAX V 器件的 I/O 元件和结构



7.5.4. FPGA RAM 块

通过 Resource Property Editor，可查看器件中不同 RAM 块的体系结构，修改输入和输出端口连接性。默认情况下，Intel Quartus Prime 软件以蓝色显示已使用资源并以灰色显示未使用资源。

图 61. Stratix V 器件中的 M9K RAM 视图



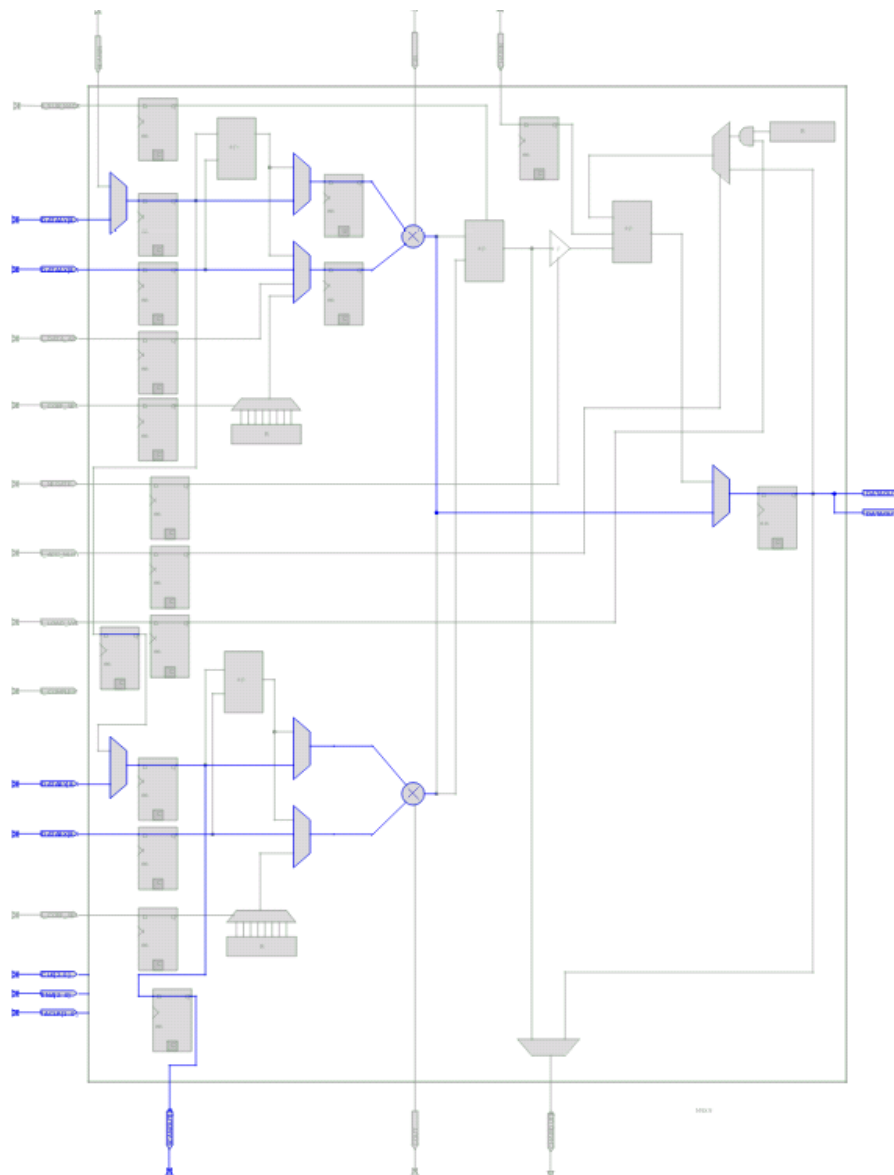
7.5.5. FPGA DSP 块

Altera 器件中的专用硬件 DSP 电路块为您设计中的关键 DSP 功能提供性能优势。

Resource Property Editor 允许您查看 Cyclone 和 Stratix 系列器件的 DSP 块体系结构。

Resource Property Editor 还支持修改与 DSP 块输出输入的信号连接性，并修改与 DSP 块的寄存器输入和输出。默认情况下，Intel Quartus Prime 软件以蓝色显示已使用资源，并以灰色显示未使用资源。

图 62. Stratix V 器件中的 DSP Block 视图



7.6. Change Manager (更改管理器)

Change Manager 维护您使用 Chip Planner, Resource Property Editor, Signal Probe 功能或 Tcl 脚本执行的每个更改的记录。Change Manager 中的每行数据代表一个 ECO。

Change Manager 允许您应用各种更改, 回退 (roll back) 更改, 删除更改并将更改记录导出成 Text File (.txt), Comma-Separated Value File (.csv), 或 Tcl Script File (.tcl)。

Change Manager 追踪更改之间的依赖关系, 因此在应用, 回退或删除一个更改时, 还会应用, 回退或删除任何先决条件或相关更改。

7.6.1. Change Manager 中的复杂更改

Change Manager 中的有些更改（包括创建或删除原子并更改连接性）可能看似独立的，但实际上由多个操作组成。Change Manager 在 **Index** 列中以“+”图标标记此类复杂更改。

可单击加号图标展开更改记录并显示复杂更改中执行的全部组件操作。

相关链接

[使用 Change Manager 管理更改的示例](#)

7.6.2. 管理 Signal Probe 信号

Change Manager 中记录了从 **Signal Probe Pins** 对话框创建的 Signal Probe 管脚。Signal Probe 约束后，可使用 Change Manager，在 Change Manager 的快捷菜单上选择 **Revert to Last Saved Netlist** 快速禁用 Signal Probe 约束。

相关链接

[使用 Signal Probe 快速设计调试](#)

7.6.3. 导出更改

可以 .txt、.csv 或 .tcl 文件导出更改。Tcl 脚本允许重新应用编译期间被删除的更改。

相关链接

[Intel Quartus Prime 规划层次和基于团队设计的增量式编译](#)

7.7. 脚本支持

可在 Tcl 脚本中运行本章介绍的处理过程并进行设置。也可运行命令提示中的某些处理过程。控制 Chip Planner 的 Tcl 命令位于可执行文件 quartus_cdb 的 chip_planner 包中。

相关链接

- [关于 Intel Quartus Prime 脚本](#)
- [Tcl Scripting](#) (第 0 页)
- [Intel Quartus Prime 设置文件手册](#)
- [命令行脚本](#) (第 0 页)

7.8. 常规 ECO 应用程序

可使用 ECO 对设计进行编译后变更。

有助于快速构建您的系统，可使用 Chip Planner 功能执行如下活动：

- 通过 Chip Planner 调整 I/O 的驱动强度
- 使用 Resource Property Editor 修改 PLL 属性，请参阅 [“Modify the PLL Properties With the Chip Planner”](#)
- 使用 Chip Planner 和 Resource Property Editor 修改新的资源原子间的连接性

相关链接

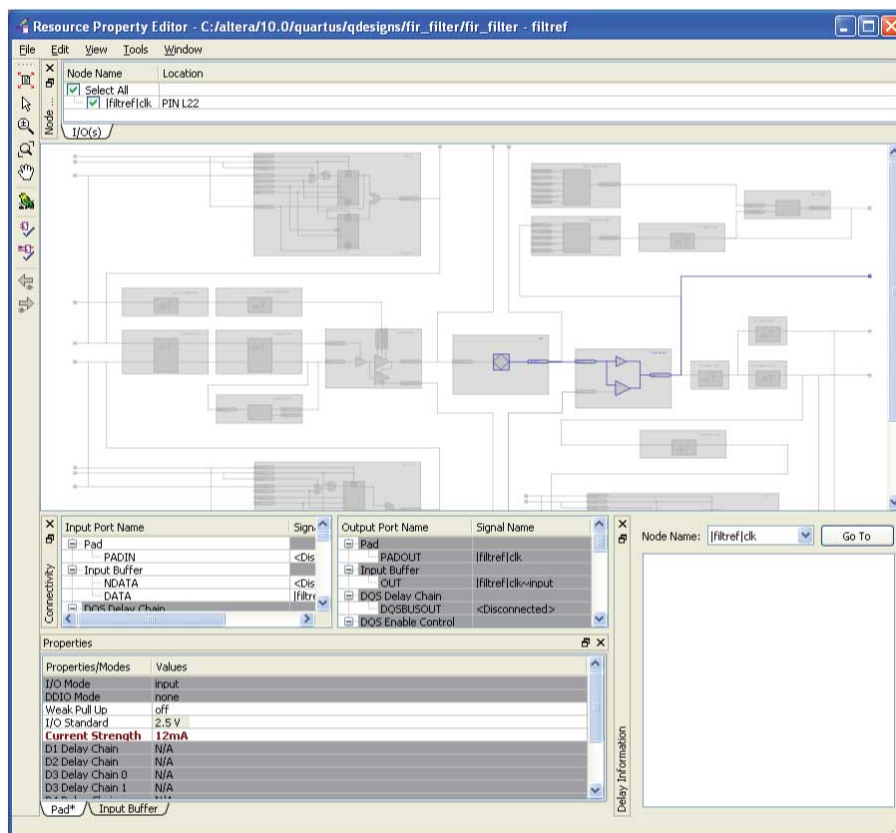
使用 Chip Planner 修改 PLL 属性 (第 146 页)

7.8.1. 使用 Chip Planner 调整 I/O 的驱动强度

调整 I/O 的驱动强度，请按照以下步骤将 ECO 包含于设计网表中。

1. 在 Chip Planner 顶部的 **Editing Mode** 列表中选择 ECO 编辑模式。
2. 在 **Resource Property Editor** 中找到 I/O。
3. 在 **Resource Property Editor** 中，指向 **Properties** 窗格中的 **Current Strength** 选项并双击该值启动下拉列表。
4. 更改 **Current Strength** 选项的值。
5. 在 Change Manager 中右键点击 ECO 变更并单击 **Check & Save All Netlist Changes** 以应用 ECO 变更。

图 63. 在 Resource Property Editor 中找到 I/O。



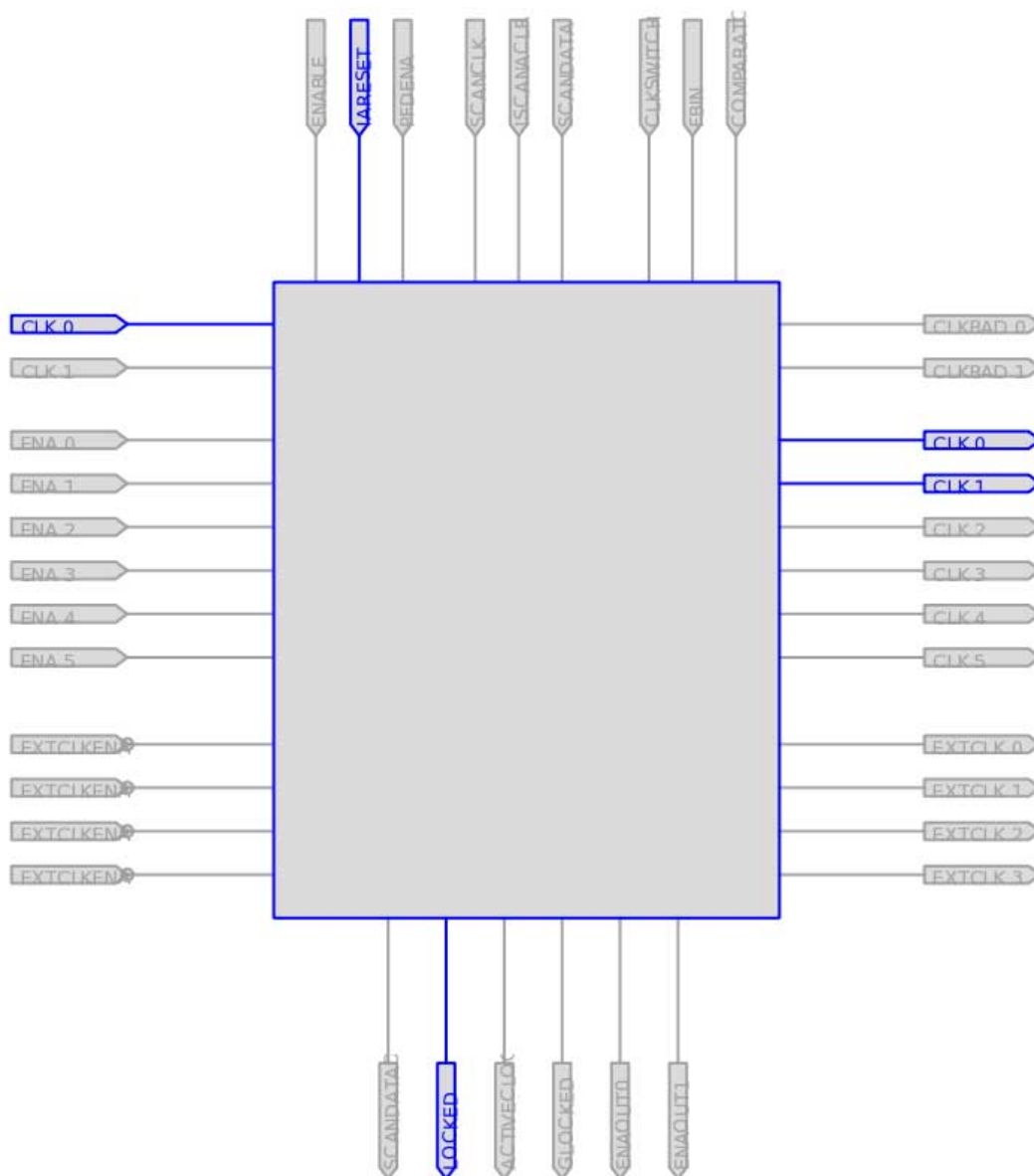
注意: 可通过 ECO 流程更改输入或输出端口的管脚位置。在 Chip Planner 的 Post Compilation Editing (ECO) 任务中，可将信号从现有管脚位置拖动或移动到新位置。然后单击 **Check & Save All Netlist Changes** 以编译 ECO。

7.8.2. 使用 Chip Planner 修改 PLL 属性

使用 PLL 修改并生成时钟信号以满足设计要求。此外，可使用 PLL 将时钟信号分配给设计中各器件，从而减少器件之间的时钟偏斜，同时提高 I/O 时序，并生成内部时钟信号。

Resource Property Editor 支持查看并修改 PLL 属性以满足设计要求。

图 64. Stratix 器件的 Resource Property Editor 中的 PLL 视图



7.8.3. PLL 属性

Resource Property Editor 允许修改 PLL 选项，例如相移，输出时钟频率和占空比。

还可使用 Resource Property Editor 更改以下 PLL 属性:

- Input frequency (输入频率)
- M V_{CO} tap (三通)
- M initial (M 初始值)
- M value (M 值)
- N value (N 值)
- M counter delay (M 计数器延迟)
- N counter delay (N 计数器延迟)
- M2 value (M2 值)
- N2 value (N2 值)
- SS counter (SS 计数器)
- Charge pump current (电荷泵电流)
- Loop filter resistance (环路滤波器电阻)
- Loop filter capacitance (环路滤波器电容)
- Counter delay (计数器延迟)
- Counter high (计数器高电平)
- Counter low (计数器低电平)
- Counter mode (计数器模式)
- Counter initial (计数器初始值)
- V_{CO} tap (三通)

还可在 Compilation Report 中查看“编译后”的 PLL 属性。为此, 在 Compilation Report 中, 选择 **Fitter**, 随后选择 **Resource Section**。

7.8.3.1. 调整占空比

使用如下方程调整各个输出时钟的占空比。

$$\text{High \%} = \frac{\text{Counter High}}{(\text{Counter High} + \text{Counter Low})}$$

7.8.3.2. 调整相移

使用此方程调整 PLL 输出时钟的相移。

$$\text{Phase Shift} = (\text{Period } V_{CO} \times 0.125 \times \text{Tap } V_{CO}) + (\text{Initial } V_{CO} \times \text{Period } V_{CO})$$

普通模式下, Tap V_{CO}、Initial V_{CO} 和 Period V_{CO} 通过以下设置管理:

$$\text{Tap } V_{CO} = \text{Counter Delay} - M \text{ Tap } V_{CO}$$

$$\text{Initial } V_{CO} = \text{Counter Delay} - M \text{ Initial}$$

$$\text{Period } V_{CO} = \text{In Clock Period} \times N \div M$$

外部反馈模式下，Tap V_{CO}、Initial V_{CO} 和 Period V_{CO} 通过以下设置管理：

Tap V_{CO} = Counter Delay - M Tap V_{CO}

Initial V_{CO} = Counter Delay - M Initial

Period V_{CO} = In Clock Period x N

(M + Counter High + Counter Low)

[相关链接](#)

[Stratix 器件手册](#)

7.8.3.3. 调整输出时钟频率

使用该方程调整普通模式下的 PLL 输出时钟。

$$\text{Output Clock Frequency} = \text{Input Frequency} \cdot \frac{M \text{ Value}}{N \text{ Value} + \text{Counter High} + \text{Counter Low}}$$

使用该方程调整外部反馈模式下的 PLL 输出时钟。

$$\text{OUTCLK} = \frac{M \text{ Value} + \text{External Feedback Counter High} + \text{External Feedback Counter Low}}{N \text{ Value} + \text{Counter High} + \text{Counter Low}}$$

7.8.3.4. 调整扩展频谱 (Spread Spectrum)

使用该方程调整您的 PLL 扩频。

$$\% \text{ Spread} = \frac{M_2 N_1}{M_1 N_2}$$

7.8.4. 修改资源原子之间的连接性

Chip Planner 和 Resource Property Editor 支持创建新的资源原子并操作“适配后”网表中资源原子间的现有连接。这些功能有助于调试设计时的小规模变更，例如将流水线手动插入时许失败的组合路径中，或将信号路由到用于分析的备用 I/O 管脚进。

使用以下处理过程在 Cyclone V 器件中创建一个新的寄存器，并将寄存器输出路由到备用 I/O 管脚。该实例说明如何创建新的西元原子并修改资源原子间的连接。

创建新的资源原子并操作“适配后”网表中资源原子之间的现有连接，请按以下步骤进行操作：

1. 在 Chip Planner 中创建新的寄存器。
2. 在 Resource Property Editor 中找到原子。
3. 将时钟信号约束到寄存器，右键点击寄存器的时钟输入端口，指向 **Edit connection**，并单击 **Other**。使用 Node Finder 约束设计中的时钟信号。
4. 将 SLOAD 输入端口连接到 V_{CC}，右键点击寄存器的时钟输入端口，指向 **Edit connection**，并单击 **VCC**。
5. 将设计中的数字信号约束到 SDATA 端口。
6. Connectivity 窗口中，在输出端口名称下，复制寄存器的端口名称。

7. 在 Chip Planner 中，找到一个空闲的 I/O 资源并创建一个输出缓冲器。
8. 在 Resource Property Editor 中找到新的 I/O 原子。
9. 在输出缓冲区的输入端口上，右键单击，指向 **Edit connection**，并单击 **Other**。
10. 在 **Edit Connection** 对话框中，键入已创建的寄存器输出端口的名称。
11. 运行 ECO Fitter 应用更改，请点击 **Check and Save Netlist Changes**。

注意： 成功的 ECO 连接受可用布线资源的影响。可通过 Chip Planner 中的 **Layers Settings** 对话框，选择 **Routing Utilization** 作为 Background Color Map（背景色示意图）查看相关布线使用情况。此外，将鼠标指针放置在相应资源上时，可通过已创建的工具提示查看每个本地，行和列互连中的布线通道使用情况。请参阅器件数据手册获得关于器件中布线互连体系结构的更多信息。

7.9. 发布 ECO 的步骤

通过 Chip Planner 指定 ECO 变更后，必须使用 Timing Analyzer 对设计执行静态时序分析，以确保您的更改不会对设计的实现性能产生消极影响。

例如，开启特定管脚的某个延迟链设置时，会更改 I/O 时序。因此，为确保设计仍满足全部时序要求，就应执行静态时序分析。

相关链接

[Intel Quartus Prime Timing Analyzer](#)

关于对设计执行静态时序分析的更多信息。

7.10. 使用 Chip Planner 的工程变更命令修订历史

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|-------------|------------------------|--|
| 2018.09.24 | 18.1.0 | Intel Quartus Prime Standard 版用户指南首次发布。 |
| 2018.05.07 | 18.0.0 | 添加了声明以表明 Intel Arria 10 器件无 Chip Planner ECO 支持。 |
| 2015.11.02 | 15.1.0 | 将 <i>Quartus II</i> 更改为 <i>Quartus Prime</i> 。 |
| 2014 年 6 月 | 14.0.0 | <ul style="list-style-type: none"> 更新了格式。 删除了关于 Stratix、Stratix II、Stratix III、Arria GX、Arria II GX、Cyclone、Cyclone II、Cyclone III 和 MAX II 器件的参考文献内容。 添加了 MAX V、Cyclone V、Arria V I/O 元件 |
| 2012 年 6 月 | 12.0.0 | 删除了反馈问卷链接。 |
| 2011 年 11 月 | 10.1.1 | 文档模板更新。 |
| 2010 年 12 月 | 10.1.0 | <ul style="list-style-type: none"> 章节更新为新的模板 删除了“Chip Planner 平面规划视图”部分 合并了“创建原子”，“删除原子”和“移动原子”部分，并链接到 Help。 在“FPGA I/O 元件”中添加 Stratix V I/O 元件。 |
| 2010 年 7 月 | 10.0.0 | <ul style="list-style-type: none"> 为第 17 - 1 页添加信息。 为第 17 - 2 页上的“工程变更命令”添加信息。 将第 7 - 2 页上的标题“性能”更改为“性能保留”。 对第 17 - 2 页上的“性能保留”中的信息更新。 将第 17 - 3 页上的标题“文档”更改为“更改及修订记录”。 |

继续...

| 文档版本 | Intel Quartus Prime 版本 | 修订内容 |
|-------------|------------------------|--|
| | | <ul style="list-style-type: none"> 将第 17 – 15 页上的标题“Resource Property Editor”更改为“在 Resource Property Editor 中执行 ECO”。 删除了“在 ECO 流程中使用增量式编译”部分。Preservation support for ECOs with the incremental compilation flow has been removed in the Quartus II 软件 10.0 中删除了“保留增量式编译流程中的 ECO 支持”。 删除了“参考文档”部分。 |
| 2009 年 11 月 | 9.1.0 | <ul style="list-style-type: none"> 更新了器件支持清单。 少量编辑修改。 |
| 2009 年 3 月 | 9.0.0 | <ul style="list-style-type: none"> 更新了图 17-17。 少量编辑修改。 第 15 章原为版本 8.1.0 中的第 13 章 |
| 2008 年 11 月 | 8.1.0 | <ul style="list-style-type: none"> 在第 15-31 页的“在 ECO 流程中使用增量式编译”部分中更正了 ECO 的保留属性。 少量编辑修改。 更改为 8½” x 11” 页面规格。 |
| 2008 年 5 月 | 8.0.0 | <ul style="list-style-type: none"> 更新了器件支持清单。 修改了关于 block RAM 和 DSP block 的 ECO 支持说明 更正了 Stratix PLL ECO 实例 添加了应用实例以演示修改资源原子间的连接性 |

相关链接

文档存档

了解 *Intel Quartus Prime 手册* 以前的版本，请搜索文档存档。

A. Intel Quartus Prime Standard Edition 用户指南

关于 Intel Quartus Prime Standard Edition FPGA 设计流程的所有阶段的详细信息，请参考以下用户指南。

相关链接

- [Intel Quartus Prime Standard Edition 用户指南：入门](#)
介绍了 Intel Quartus Prime Standard Edition 软件的基本功能，文件和设计流程，包括管理 Intel Quartus Prime Standard Edition 工程和 IP，初始设计规划注意事项以及从之前软件版本的工程迁移。
- [Intel Quartus Prime Standard Edition 用户指南：平台设计程序](#)
说明使用 Platform Designer (Standard) 创建和优化系统，该系统集成工具可简化工程中自定义 IP 核聚合。Platform Designer (Standard) 自动生成互连逻辑以连接知识产权 (IP) 功能和子系统。
- [Intel Quartus Prime Standard Edition 用户指南：设计建议](#)
介绍使用 Intel Quartus Prime Standard Edition 软件设计 FPGA 的最佳设计实践。HDL 代码样式和同步设计实践可显著影响设计性能。以下建议的 HDL 编码样式确保了 Intel Quartus Prime Standard Edition 综合将设计在硬件中最佳实现。
- [Intel Quartus Prime Standard Edition 用户指南：设计编译](#)
说明了 Intel Quartus Prime Standard Edition Compiler 从建立，运行到优化的全部阶段。生成器件编程文件之前，Compiler 对设计进行综合，布局和布线。
- [Intel Quartus Prime Standard Edition 用户指南：设计优化](#)
说明在 Intel FPGA 中实现最高设计性能可使用的 Intel Quartus Prime Standard Edition 设置，工具和技术。可使用的技术包括优化设计网表，解决限制重定时和时序收敛的关键链以及器件资源使用优化。
- [Intel Quartus Prime Standard Edition 用户指南：Programmer](#)
说明 Intel Quartus Prime Standard Edition Programmer (下载配置工具) 的运行，并通过连接 Intel FPGA 下载电缆配置 Intel FPGA 器件，编程 CPLD 和配置器件。
- [Intel Quartus Prime Standard Edition 用户指南：局部重新配置](#)
介绍 Partial Reconfiguration 高级设计流程，其支持动态重配置 FPGA 某一部分的同时其余 FPGA 设计继续运行。将部分设计区域定义为多重角色时，而不影响其他区域中的操作。
- [Intel Quartus Prime Standard Edition 用户指南：第三方仿真](#)
说明通过 Aldec*，Cadence*，Mentor Graphics* 和 Synopsys 为第三方仿真工具提供 RTL-和门级设计仿真支持，从而允许在器件编程之前验证设计行为。包括仿真器支持，仿真流程和仿真 Intel FPGA IP。
- [Intel Quartus Prime Standard Edition 用户指南：第三方综合](#)
说明通过 Mentor Graphics* 和 Synopsys，第三方综合工具为设计中选择性综合部分提供支持。包括设计流程步骤，生成的文件说明和综合指导。

- [Intel Quartus Prime Standard Edition 用户指南：调试工具](#)
介绍为设计进行实时验证的 Intel Quartus Prime Standard Edition 在系统设计调试工具文件夹。这些工具通过将设计中的信号路由选择（或“分接”）到调试逻辑来提供可视性。具体工具包括，System Console，Signal Tap logic analyzer，Transceiver Toolkit，In-System Memory Content Editor 和 In-System Sources and Probes Editor。
- [Intel Quartus Prime Standard Edition 用户指南：Timing Analyzer](#)
解释基本静态时序分析原则和 Intel Quartus Prime Standard Edition Timing Analyzer 的使用。其作为功能强大的 ASIC 式时序分析工具，通过使用行业标准的约束，分析和报告方法验证设计中所有逻辑的时序性能。
- [Intel Quartus Prime Standard Edition 用户指南：功耗分析和优化](#)
说明 Intel Quartus Prime Standard Edition Power Analysis 工具支持准确估算器件功耗。估算器件功耗以开发功率预算和设计电源，稳压器，散热器和冷却系统。
- [Intel Quartus Prime Standard Edition 用户指南：设计优化](#)
说明影响 Compiler 如何实现设计的时序和逻辑约束，例如管脚约束，器件选项，逻辑选项和时序约束。使用 Pin Planner 可以在目标器件的图形表示中可视化，修改和验证所有 I/O 约束。
- [Intel Quartus Prime Standard Edition 用户指南：PCB 设计工具](#)
说明通过 Mentor Graphics* 和 Cadence* 实现对可选第三方 PCB 设计工具的支持。还包括信号集成分析和使用 HSPICE 和 IBIS 模型进行仿真的信息。
- [Intel Quartus Prime Standard Edition 用户指南：脚本](#)
说明使用 Tcl 和命令行编制脚本进行 Intel Quartus Prime Standard Edition 软件控制并广泛执行各种功能，例如管理工程，指定约束，运行编译或时序分析，以及生成报告。