

引言

本快捷参考指南提供了以下分步骤流程，用于根据《适用于 FPGA 和 SoC 的 UltraFast 设计方法指南》(UG949) 中的建议快速完成时序收敛：

- **初始设计检查：** 在实现设计前审核资源利用率、逻辑层次和时序约束。
- **时序基线设定：** 在每个实现步骤后检查并解决时序违例，从而帮助布线后收敛时序。
- **时序违例解决：** 识别建立时间违例或保持时间违例的根源，并解决时序违例。

QoR 评估报告

您可使用结果质量 (QoR) 评估报告来快速复查设计。此报告会将关键设计指标和约束指标与准则中所述限制进行比对。与准则不符的指标都会被标记为 REVIEW。此报告包括下列部分：

- 设计特性
- 方法检查
- 根据目标 Fmax 进行保守的逻辑层次评估

在 AMD Vivado® 工具中，您可按如下所述方式运行此报告：

report\_qor\_assessment

请参阅第 10 页的 **QoR 评估报告概述** 和《Vivado Design Suite 用户指南：设计分析与收敛技巧》(UG906)。

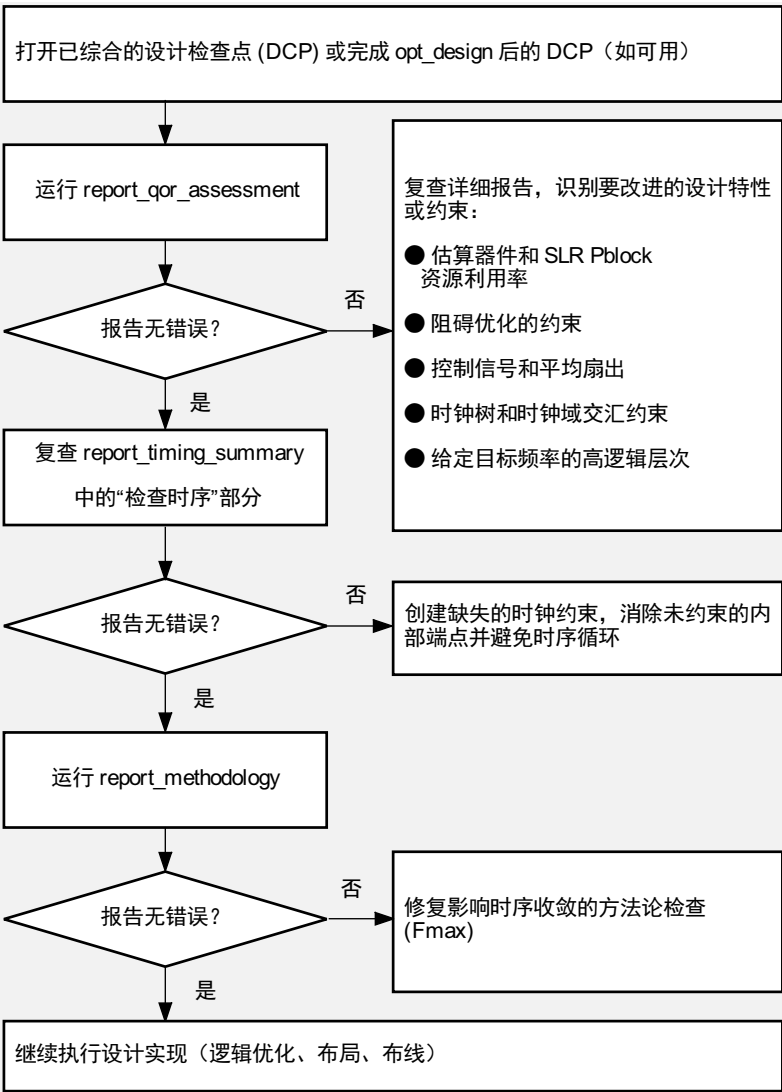
QoR 建议报告

在 Vivado 工具中，在实现阶段会调用 report\_qor\_suggestions。此报告用于分析设计、提供建议，在某些情况下会自动应用建议。

Vitis 环境中的报告

在 AMD Vitis™ 环境中，在编译流程期间使用 v++ - R 1 或 v++ - R 2 来调用 report\_qor\_assessment

初始设计检查流程



**提示：** 通过使用 Intelligent Design Run (IDR) 即可在实现期间自动解决大部分时序收敛难题。IDR 属于特殊类型的实现运行，能够有效利用 report\_qor\_suggestions、基于 ML 的策略预测以及增量编译。欲知详情，请参阅 UG949 中的“使用智能设计运行”。

初始设计检查详细介绍

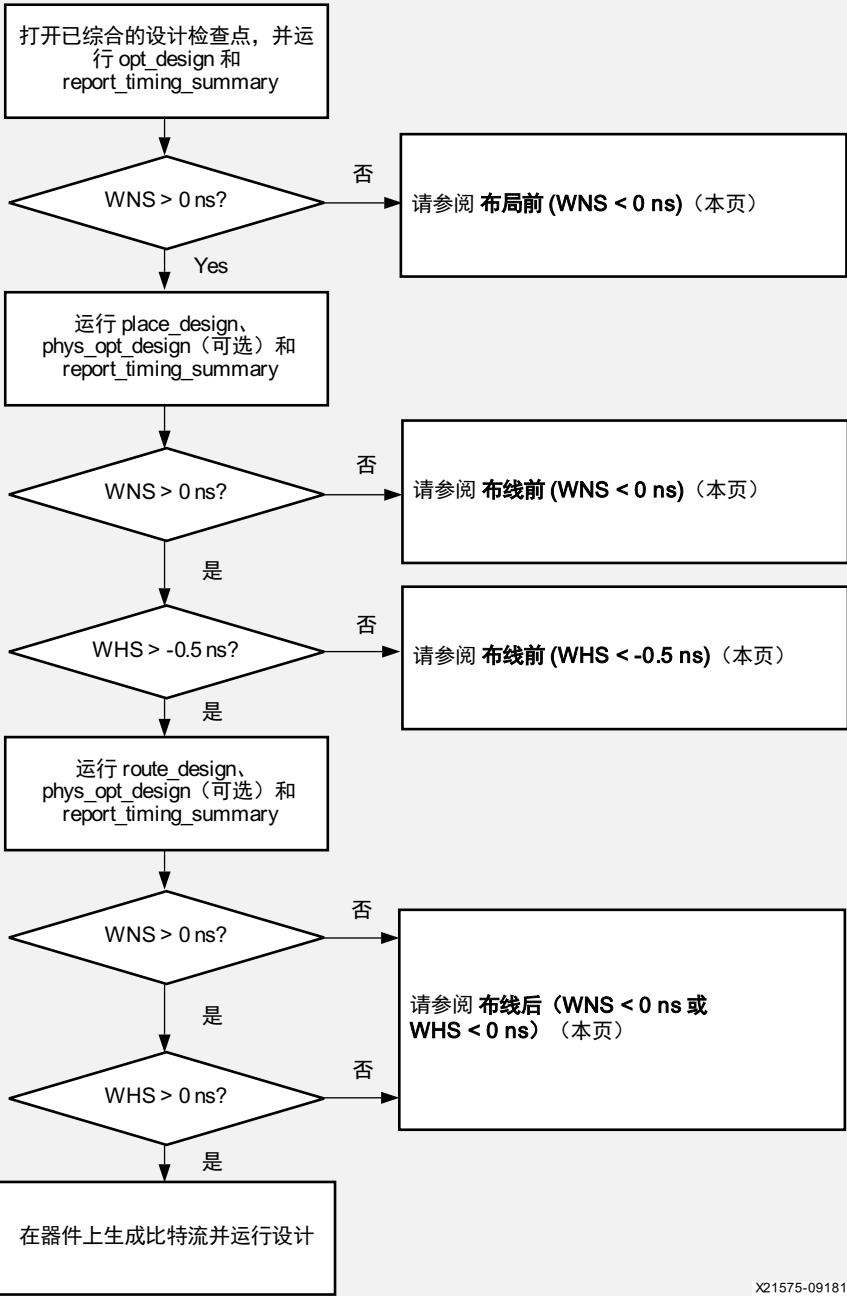
虽然在 AMD 器件上实现设计是一个自动化程度相当高的任务，但要实现更高的性能并解决因时序或布线违例所带来的编译问题，则是一项复杂且耗时的工作。仅根据简单的日志消息或由工具生成的实现后时序报告可能难以明确失败原因。因此有必要采用按步骤进行设计开发和编译的方法，包括复查中间结果以确保设计能继续执行下一个实现步骤。

第一步是确保所有的初始设计检查都已经完成。在下列层次复查检查结果：

- 由定制 RTL 构成或者由 Vivado HLS 生成的每个内核  
**注意：** 检查目标时钟频率约束是否现实。
- 与子系统逐一对应的每个主要层级，例如有多个内核、IP 块和连接逻辑的 Vivado IP integrator 模块框图
- 包括所有主要功能和层级、I/O 接口、完整时钟电路、物理约束和时序约束的完整设计

如果设计使用布局规划约束，如超级逻辑区域 (SLR) 分配或分配给 Pblock 的逻辑，请复查每项物理约束的估算的资源利用率，确保符合资源利用率准则。运行 report\_qor\_assessment 时，会自动检查 SLR 和 Pblock 违例。如未报告任何违例，则表示设计在可接受的限制范围内。

时序基线设定流程



X21575-091811

时序基线设定示例

时序基线设定的目的是通过在每个实现步骤后分析和解决时序问题，确保设计满足时序要求。在编译流程中尽早纠正设计问题和约束问题，能带来更广泛的影响并实现更高的性能。通过创建如下中间报告，在进入下一步前复查并解决时序违例：

Vivado 工程模式报告	Vivado 非工程模式报告	Vitis 软件平台报告
使用 AMD UltraFast™ 设计方法论或时序收敛报告策略	在每个实现步骤后添加下列报告命令： <ul style="list-style-type: none"><li>report_timing_summary</li><li>report_methodology</li><li>report_qor_assessment</li></ul>	使用 v++ -R 1 或 v++ -R 2 选项在下列目录中生成中间时序报告和 DCP： <runDir>/_x/link/vivado/prj/prj.runs/impl_1

布局前 (WNS < 0ns)

执行 place\_design 前，时序报告反映的是以下情况下的设计性能：假定每条逻辑路径均达到最理想逻辑布局。必须通过采用“初始检查”建议来解决建立时间违例。

布线前 (WNS < 0ns)

执行 route\_design 前，时序报告反映的是以下情况下的设计性能：假定每条单独信号线均达到最理想布线延迟，存在一些扇出惩罚且不考虑保持时间修复的影响（信号线布线绕行）或拥塞。建立时间违例往往是下列原因导致的次优布局造成的：(1) 器件或 SLR 利用率过高；(2) 因复杂逻辑连接造成的布局拥塞；(3) 大量路径逻辑级数过多；和 (4) 不平衡时钟间的时钟偏差或时钟不确定性过高。在 Explore 模式或 AggressiveExplore 模式下运行 phys\_opt\_design，尝试提升执行 place\_design 后的 QoR。如果不成功，先重点提升布局 QoR。

布线前 (WHS < -0.5ns)

如果布线后未达成性能目标且布线前最差负时序裕量 (WNS) 为正值，请尽可能减少估算的最差保持时序裕量 (WHS) 严重违例。减少并减轻布线前保持时间违例有助于 route\_design 集中处理 Fmax 而非修复保持时间违例。

布线后 (WNS < 0ns 或 WHS < 0ns)

执行 route\_design 后，首先复查日志文件或在布线后设计检查点 (DCP) 上运行 report\_route\_status，验证设计是否完全布线。布线违例和严重的建立时间 (WNS) 或保持时间 (WHS) 违例是由拥塞过高导致的结果。使用分析建立时间违例（第 3 页）、解决保持时间违例（第 4 页）以及拥塞减少方法（第 6 页），识别和实现解决步骤。执行 route\_design 后，尝试运行 phys\_opt\_design，解决轻微的建立时间违例 (> -0.200 ns)。

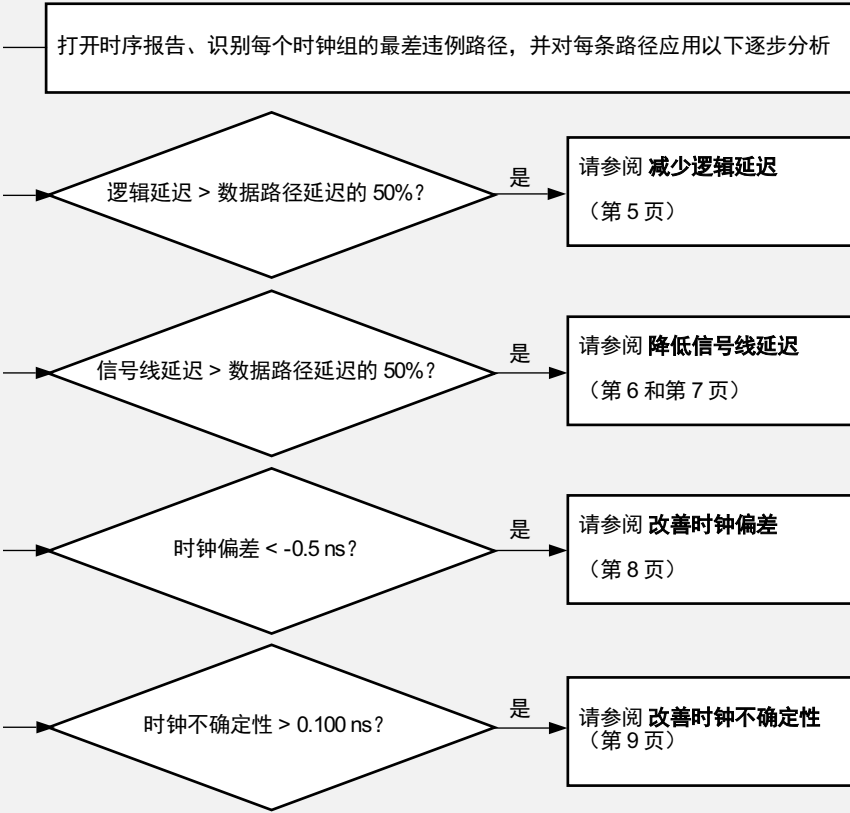
在迭代设计、约束和编译策略时，在每个步骤后持续记录 QoR，包括拥塞信息。使用 QoR 表对运行参数进行比较，并确定在解决剩余时序违例时优先关注的对象。

	opt_design		place_design			phys_opt_design				route_design				
Impl. Run	Directive	WNS	Directive	WNS	Congestion	Directive	WNS	WHS	THS	Directive	WNS	TNS	WHS	Congestion
Run1	ExploreWithRemap	0.034	WLDriVenBlockPlacement	-0.07	5-4-5-5	Explore	0.001	-0.409	-851.052	NoTimingRelaxation	-0.02	-1.68	0.006	5-5-4-5
Run2	Explore	0.054	AltSpreadLogic_medium	-0.368	6-5-5-5	Explore	-0.068	-0.364	-852.889	Default	-1.50	-3680.32	0.003	5-6-5-6
Run3	Default	0.054	AltSpreadLogic_high	-0.393	5-4-5-5	Explore	0.035	-0.364	-906.036	Explore	-1.37	-1495.19	0.006	4-5-5-6
Run4	Default	0.054	ExtraTimingOpt	-0.41	5-5-5-5	Explore	0.075	-0.407	-902.348	Explore	-1.23	-2896.42	0.001	5-5-5-6

提示：在 place\_design 和 route\_design 后使用 report\_qor\_suggestions 可自动识别哪些设计、约束和工具选项更改有助于为新编译提升 QoR。

分析建立时间违例流程

- 设计性能由如下因素决定：
- 时钟偏差与时钟不确定性：时钟如何高效实现
  - 逻辑延迟：每个时钟周期内遍历的逻辑量
  - 信号线延迟或布线延迟：Vivado 如何高效实现设计的布局布线
- 使用时序路径或设计分析报告内的信息来：
- 识别时序违例主要是由哪些因素引发的
  - 确定如何迭代提升 QoR
- 提示：如果需要，可在每个步骤后打开 DCP 以生成更多报告。



X21576-091811

在报告中找出建立时序路径特性

- 在 Vivado 工程模式中，按如下步骤找到建立时序路径特性：
- 在“Design Runs”窗口中，选择要分析的实现运行。
  - 在“Implementation Run Properties”窗口中，选择“**Reports**”选项卡。
  - 打开选定实现步骤的时序汇总报告或设计分析报告：
    - 时序汇总报告：<runName>\_<flowStep>\_report\_timing\_summary（针对文本采用 .rpt 或针对 Vivado IDE 采用 .rpx）
    - 设计分析报告：<runName>\_<flowStep>\_report\_design\_analysis
- 在 Vivado 非工程模式或 Vitis 软件平台中，采取下列操作之一：
- 打开实现运行目录中的报告。
  - 打开 Vivado IDE 中的实现 DCP，然后打开报告的 RPX 版本。
- 注意：使用 Vivado IDE 可便于您在报告、板级原理图和“Device”窗口间进行交叉探测。

对于每条时序路径，逻辑延迟、布线延迟、时钟偏差和时钟不确定性特性都位于路径的报头内：

Summary	
Name	Path 61
Slack	-2.321ns
Source	ingressFifoWrEn_reg/C (rising edge-triggered cell FDRE clocked by wbClk {rise@15.000ns - arrival time})
Destination	ingressLoop[3].ingressFifo/buffer_fifo/infer_fifo.block_ram (rising edge-triggered cell RAMB36E2 clocked by bftClk {rise@14.000ns - arrival time})
Path Group	bftClk
Path Type	Setup (Max at Slow Process Corner)
Requirement	1.000ns (bftClk rise@15.000ns - wbClk rise@14.000ns)
Data Path Delay	2.214ns (Logic 0.318ns (14.363%) route 1.896ns (85.637%))
Logic Levels	1 (LUT3=1)
Clock Path Skew	-0.610ns
Clock Uncertainty	0.035ns

Slack (VIOLATED) : -2.321ns (required time - arrival time)

Source: ingressFifoWrEn\_reg/C (rising edge-triggered cell FDRE clocked by wbClk {rise@15.000ns - arrival time})

Destination: ingressLoop[3].ingressFifo/buffer\_fifo/infer\_fifo.block\_ram (rising edge-triggered cell RAMB36E2 clocked by bftClk {rise@14.000ns - arrival time})

Path Group: bftClk

Path Type: Setup (Max at Slow Process Corner)

Requirement: 1.000ns (bftClk rise@15.000ns - wbClk rise@14.000ns)

Data Path Delay: 2.214ns (Logic 0.318ns (14.363%) route 1.896ns (85.637%))

Logic Levels: 1 (LUT3=1)

Clock Path Skew: -0.610ns (DCD - SCD + CPR)

Destination Clock Delay (DCD): 3.236ns = ( 18.236 - 15.000 )

Source Clock Delay (SCD): 3.846ns = ( 17.846 - 14.000 )

Clock Pessimism Removal (CPR): 0.000ns

Clock Uncertainty: 0.035ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE

Total System Jitter (TSJ): 0.071ns

Total Input Jitter (TIJ): 0.000ns

Discrete Jitter (DJ): 0.000ns

Phase Error (PE): 0.000ns

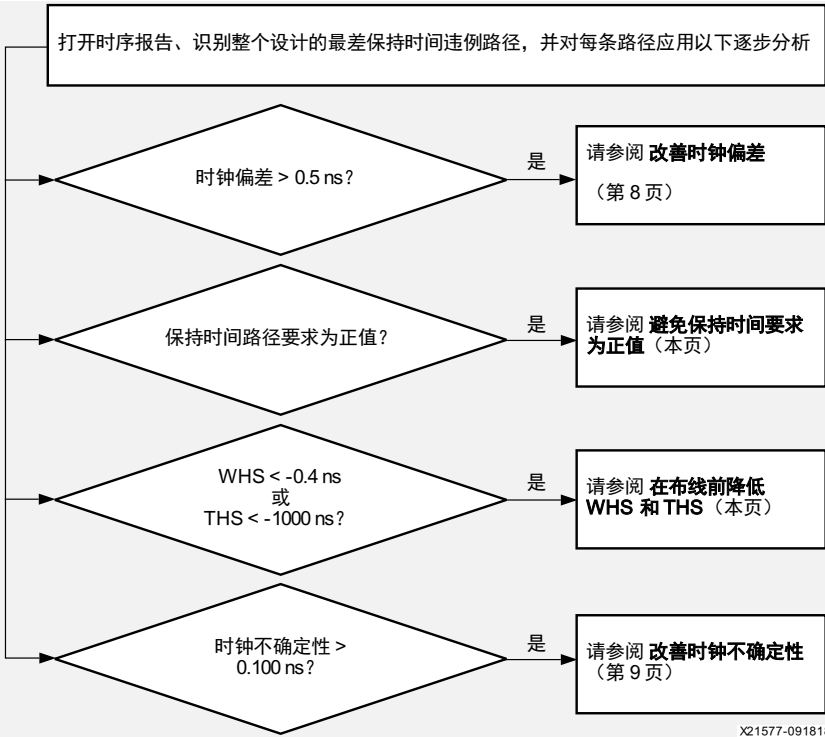
在设计分析报告的“Setup Path Characteristics”中同样包含这些时序路径特性以及其它信息，如逻辑级数和布线：

Name	Slack	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Logic Levels	Routes	Logical Path
Path 1	-1.438	1.592	3.244	6%	94%	0.665	1	2	FDRE LUT4 FDRE
Path 2	-0.708	3.184	3.508	43%	57%	-0.362	5	5	RAMB18E2 LUT6 LUT6 LUT6 LUT6 LUT6 FDRE
Path 3	-0.683	3.184	3.483	42%	58%	-0.362	5	5	RAMB18E2 LUT6 LUT6 LUT6 LUT6 LUT6 FDRE
Path 4	-0.675	3.184	3.505	37%	63%	-0.333	10	8	FDRE LUT6 LUT6 LUT6 LUT5 LUT6 LUT2 CARRY8 CARRY8 LUT4 LUT6 FDRE

提示：在文本模式下，“Setup Path Characteristics”列的全部列都会显示，导致该表非常宽。在 Vivado IDE 中，该表所示列数较少，以便于查看。右键单击表头，根据需要启用或禁用列。例如，默认情况下不显示 DONT\_TOUCH 或 MARK\_DEBUG 列。启用这些列可以查看逻辑优化分析中跳过的重要信息，这些信息较难以通过其它方式来查看。



解决保持时间违例流程



以下是含高时钟偏差的保持时序路径示例：

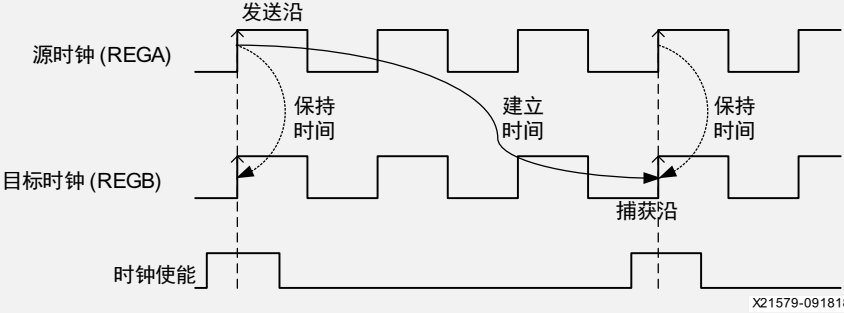
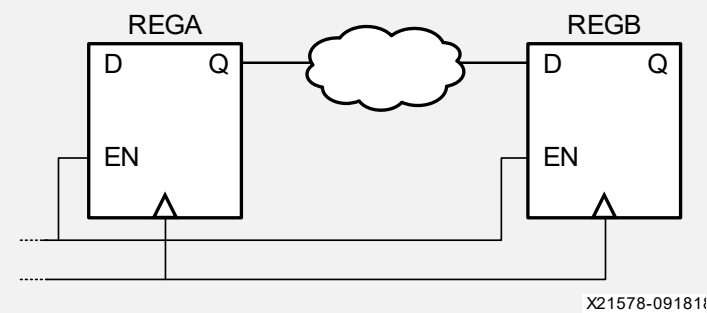
Summary	
Name	Path 259
Slack (Hold)	-1.129ns
Source	inst_209033/inst_381/inst_285879/inst_285870/inst_285584/i
Destination	inst_209033/inst_381/inst_285879/inst_285870/inst_285584/i
Path Group	app_clk
Path Type	Hold (Min at Slow Process Corner)
Requirement	0.000ns (app_clk rise@0.000ns - txoutclk_out[3]_3 rise@0.000ns)
Data Path Delay	0.180ns (logic 0.059ns (32.778%) route 0.121ns (67.222%))
Logic Levels	0
Clock Path Skew	1.247ns

解决保持时间违例的方法

避免保持时间要求为正值

在使用多周期路径约束放宽建立时间检查时，您必须：

- 调整同一路径上的保持时间检查，以便在保持时间分析中使用相同的发送沿和捕获沿。否则会导致保持时间要求（一个或多个时钟周期）为正值，无法实现时序收敛。
- 指定端点管脚，而不仅仅是单元或时钟。例如，端点单元 REGB 有三个输入管脚：C、EN 和 D。只有 REGB/D 管脚需交由多周期路径例外来约束（时钟使能 (EN) 管脚不用），因为 EN 管脚在每个时钟周期都会发生改变。如果将约束连接至单元而不是管脚，那么所有有效的端点管脚（包括 EN 管脚）都在约束的考虑范围内。



AMD 建议您始终使用如下语法：

```
set_multicycle_path -from [get_pins REGA/C] -to [get_pins REGB/D] -setup 3
```

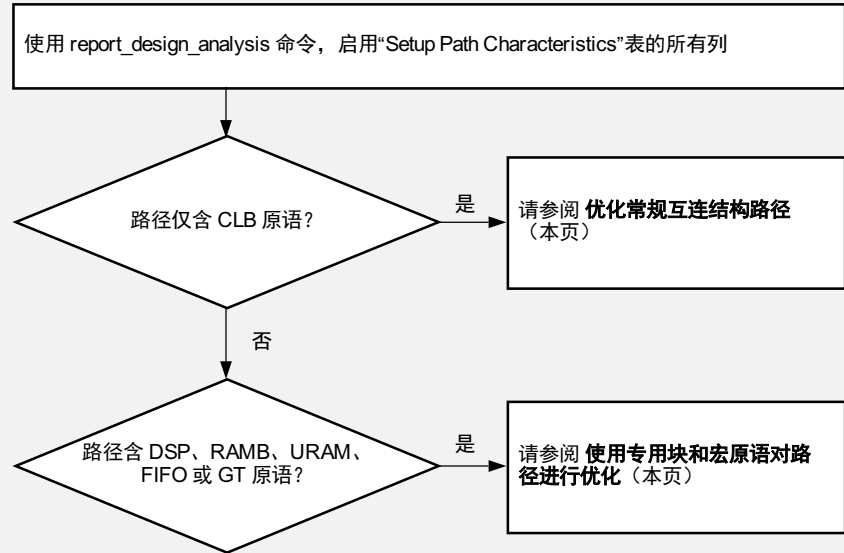
```
set_multicycle_path -from [get_pins REGA/C] -to [get_pins REGB/D] -hold 2
```

在布线前降低 WHS 和 THS

严重的估算保持时间违例会增大布线难度，且并非总能通过 route\_design 解决。布局后 phys\_opt\_design 命令提供多种保持时间修复选项：

- 在时序元件之间插入反向边缘触发寄存器会将时序路径分为两条半周期路径，从而显著减少保持时间违例。这种优化只能在不劣化建立时序的情况下执行。使用下列命令：  
phys\_opt\_design -insert\_negative\_edge\_ffs
- 插入 LUT1 缓冲器会导致数据路径延迟，这样即可减少保持时间违例，而不会造成建立时间违例。使用下列命令：
  - phys\_opt\_design -hold\_fix：仅在含最严重的 WHS 违例的路径上插入 LUT1。
  - phys\_opt\_design -aggressive\_hold\_fix：以 LUT 利用率显著增加和编译时间延长为代价，在更多路径上插入 LUT1 以大幅降低总体保持时序裕量 (THS)。该选项可以与任何 phys\_opt\_design 指令结合使用。
  - phys\_opt\_design -directive ExploreWithAggressiveHoldFix：在完成旨在提升 Fmax 的所有其它物理优化后，执行 LUT1 插入以修复保持时间。

减少逻辑延迟流程



X21580-091811

Vivado 实现首先关注最关键的路径。这意味着收敛难度较低的路径在布局之后或布线之后经常会变为关键路径。AMD 建议在综合之后或 opt\_design 之后识别和改进最长路径，因为它对 QoR 产生的影响最大，并且通常会显著减少实现时序收敛所需的布局布线迭代次数。使用 report\_design\_analysis “逻辑层次分布”表，通过根据要求权衡逻辑层次分布，找出需要改进设计的时钟域。要求越低，允许的逻辑层次就越少。例如，在以下布局前逻辑层次分布报告中：

- 复查针对 txoutclk\_out[0]\_4 存在不少于 8 个逻辑层次的所有路径。
- 复查针对 app\_clk 存在不少于 11 个逻辑层次的所有路径。

Q ⓘ Logic Level Distribution																	
End Point Clock	Requirement	0	1	2	4	5	6	7	8	9	10	11	12	13	14	15	16
app_clk	3.184ns	0	0	0	1	0	0	135	16	37	30	16	16	16	16	15	7
txoutclk_out[0]_4	2.388ns	2	0	0	64	784	1677	0	9	3	0	3	4	0	0	0	0
txoutclk_out[3]_3	1.592ns	2100	5029	20	0	0	0	0	0	0	0	0	0	0	0	0	0

**注意：**级联 CARRY 或 MUXF 单元能够人为增加逻辑层数，且对延迟影响小。  
**提示：**在 Vivado IDE 报告中，单击逻辑层次编号以选择路径，然后按 F4 生成板级原理图并复查逻辑。

减少逻辑延迟的方法

优化常规互连结构路径

常规互连结构路径是寄存器 (FD\*) 或移位寄存器 (SRL\*) 之间的路径，并且遍历 LUT、MUXF 和 CARRY。如果您遇到与常规互连结构路径有关的问题，AMD 建议如下。如需了解更多信息，请参阅《Vivado Design Suite 用户指南：综合》(UG901) 和《Vivado Design Suite 用户指南：实现》(UG904)。

- 对于较高的逻辑层次，可使用 report\_qor\_assessment 通过 LUT/信号线预算检查来识别路径。在设计周期早期可通过对 RTL 进行重新编码或者使用重定时来对较高的逻辑层次进行寻址。<sup>1</sup>  
**建议：**全局使用综合 -retiming。使用块综合策略 BLOCK\_SYNTH.RETIMING 1 设定模块目标，或者使用 RETIMING\_FORWARD/BACKWARD 属性设定特定单元目标。
- 小型级联 LUT (LUT1-LUT4) 可通过合并来减少 LUT 数量，但是设计层级、存在扇出（10 和更高）的中间信号线、或者使用 KEEP、KEEP\_HIERARCHY、DONT\_TOUCH 或 MARK\_DEBUG 属性都会阻止 LUT 合并。<sup>1</sup>  
**建议：**移除属性，从综合步骤或从 opt\_design -remap 开始重新运行。
- 单一 CARRY（非级联）单元会限制 LUT 优化，可能导致布局无法达成最优状态。<sup>1</sup>  
**建议：**使用 FewerCarryChains 综合指令，或者针对将由 opt\_design 移除的单元设置 CARRY\_REMAP 属性。
- 移位寄存器 SRL\* 延迟高于寄存器 FD\* 延迟，且 SRL 布局优化质量可能不及 FD 布局。<sup>1</sup>  
**建议：**在综合后使用 RTL 中的 SRL\_STYLE 属性，或是单元上的 SRL\_STAGES\_TO\_INPUT 或 SLR\_STAGES\_TO\_OUTPUT 属性从 SRL 输入或输出拉出寄存器。在 RTL 中必须修改动态 SRL。
- 当逻辑路径终止时，如由 LUT 来驱动互连结构寄存器 (FD\*) 的时钟使能 (CE) 管脚、同步置位 (S) 管脚或同步复位 (R) 管脚，则布线延迟高于寄存器数据管脚 (D)，当路径最后一个信号线的扇出大于 1 时尤其如此。<sup>1</sup>  
**建议：**如果终止于数据管脚 (D) 的路径裕量更高且逻辑层数更少，则在 RTL 内把信号上的 EXTRACT\_ENABLE 或 EXTRACT\_RESET 属性设置为 no。此外，元上的 CONTROL\_SET\_REMAP 属性也可设置为在 opt\_design 过程中触发相同优化。

使用专用块和宏原语对路径进行优化

往来于专用块与宏原语之间的逻辑路径（例如，DSP、RAMB、URAM、FIFO 或 GT\_CHANNEL）布局难度更高，且单元和布线延迟更高。因此，围绕宏原语添加额外的流水打拍或减少宏原语路径上的逻辑层数，对提升总体设计性能至关重要。

在修改 RTL 之前，通过启用所有可选的 DSP、RAMB 和 URAM 寄存器并重新运行实现，确认添加流水打拍给 QoR 带来的好处。在采用这一评估方法时请勿生成比特流。例如：set\_property -dict {DOA\_REG 1 DOB\_REG 1} [get\_cells xx/ramb18\_inst]

以下是 RAMB18 路径示例，此路径需增加流水线寄存器或减少逻辑层数（执行 route\_design 后报告）：

Name	Slack	Requirement	Path Delay	Logic Delay <sup>1</sup>	Net Delay	Logic Levels	Routes	Logical Path	BRAM
Path 5	-0.663	3.184	3.472	48%	52%	5	5	RAMB18E2 LUT6 LUT6 LUT6 LUT6 LUT6 FDRE	No DO_REG

1. 表示使用 report\_qor\_suggestions 自动解决

降低信号线延迟流程 1

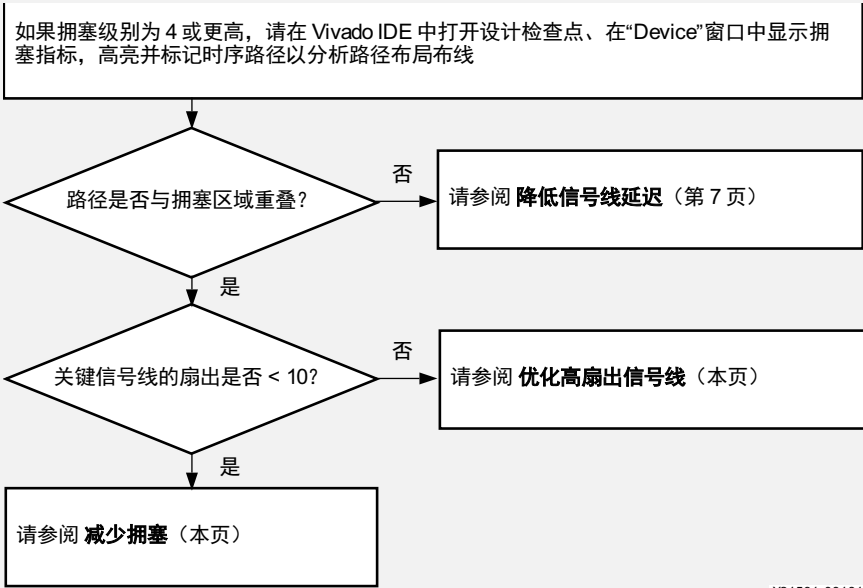
全局拥塞给设计性能造成如下影响：

- 4 级 (16x16)：在 route\_design 过程中 QoR 变化较小
- 5 级 (32x32)：次优布局，QoR 变化明显
- 6 级 (64x64)：布局布线难度大，编译时间长。除非性能目标较低，否则时序 QoR 严重劣化。
- 7 级 (128x128) 及以上：无法布局或布线。

针对 4 级及以上拥塞，route\_design 命令会在日志文件里输出初始估算拥塞表。

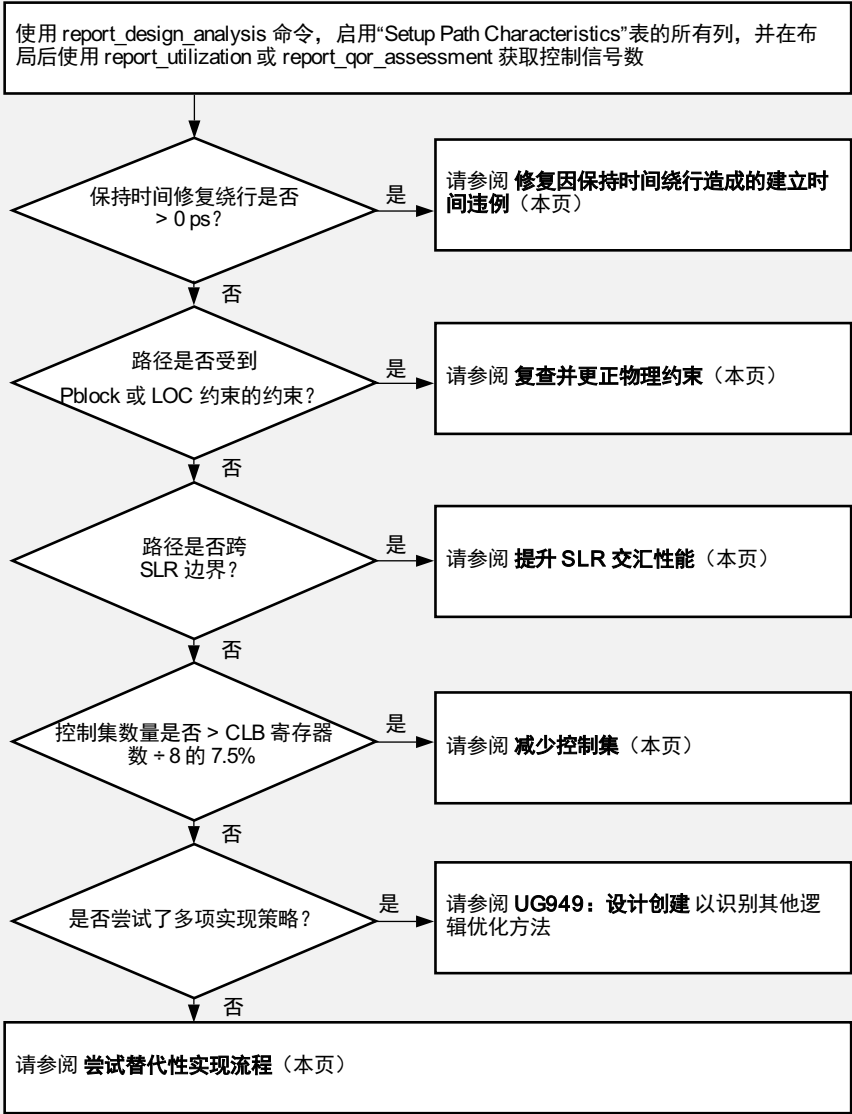
要同时报告布局器和布线器拥塞信息，请使用 report\_design\_analysis - congestion。

**提示：**在 Vivado IDE 内打开布局后或布线后 DCP 即可创建交互式“report\_design\_analysis”窗口。通过交叉探测，即可高亮显示拥塞区域并显示拥塞对每条逻辑路径的布局布线的影响。请参阅 **UG949：降低因拥塞导致的信号线延迟**。





降低信号线延迟流程 2



X21582-091811

降低信号线延迟的方法

修复因保持时间绕行造成的建立时间违例

为了确保设计硬件功能正常，修复保持时间违例优先于修复建立时间违例（或 Fmax）。以下示例显示了 2 个同步时钟之间的路径，此路径偏差较高且建立时间要求较为严格：

Name	Slack	Requirement	Path Delay	Clock Skew	Hold Fix Detour	Logical Path	Start Point Clock	End Point Clock	SLR Crossings
Path 1	-1.438	1.592	3.244	0.665	1181	FDRE LUT4 FDRE	txoutclk_out[3]_3	app_clk	1

说明：“保持时间修复绕行”以皮秒为单位。要解决保持时间绕行对 Fmax 的影响，请参阅第 4 页的“解决保持时间违例的方法”。

复查并更正物理约束

所有设计都包含物理约束。虽然一般不能更改 I/O 位置，但在执行设计更改前必须仔细确认和复查 Pblock 和位置约束。设计更改可能拉开逻辑间距，并导致信号线延迟较长。复查含多个 Pblock（Pblock 列）且含位置约束（固定位置列）的路径。

提升 SLR 交汇性能

在以堆叠硅片互联 (SSI) 技术器件为目标进行设计时，尽早考量下列设计因素有助于提升性能：

- 在主要设计层级或内核边界添加流水线寄存器有助于长距离布线和 SLR 交汇布线。
- 验证每个 SLR 的利用率均符合准则要求（使用 report\_qor\_assessment）。
- 使用 USER\_SLR\_ASSIGNMENT 约束引导实现工具。请参阅 UG949：使用软核 SLR 布局规划约束。
- 如果软核约束无效，请使用 SLR Pblock 布局约束。
- 在布局或布线后，使用 phys\_opt\_design -slr\_crossing\_opt。

减少控制集

在控制集数量超过准则要求 (7.5%) 时，不管对整个器件还是对每个 SLR，都请尝试减少控制集数量：

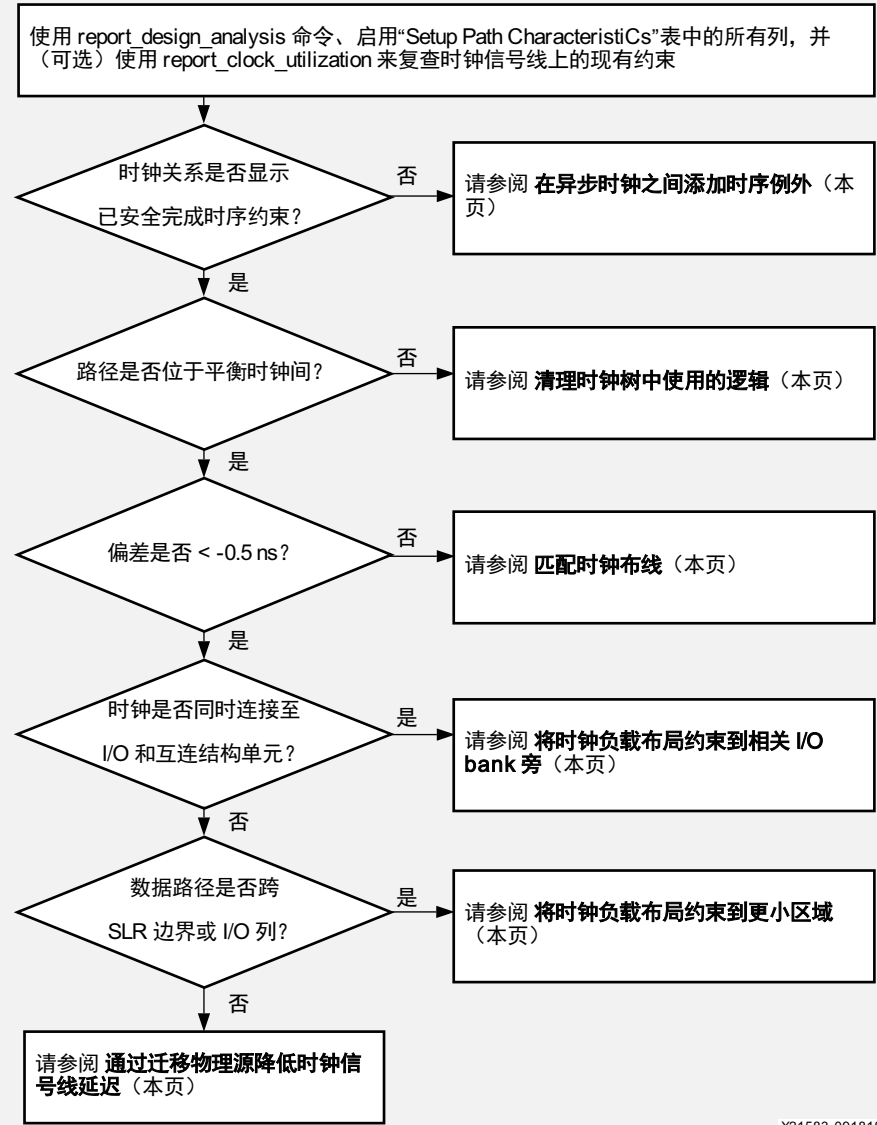
- 在 RTL 内移除时钟使能、置位或复位信号上的 MAX\_FANOUT 属性。<sup>1</sup>
- 增加最小综合控制信号扇出（如 synth\_design -control\_set\_opt\_threshold 16）。<sup>1</sup>
- 使用 opt\_design -control\_set\_merge or -merge\_equivalent\_drivers 合并复制的控制信号。
- 通过在 CLB 寄存器单元上设置 CONTROL\_SET\_REMAP 属性，将低扇出控制信号重新映射到 LUT。<sup>1</sup>

尝试替代性实现流程

可通过默认编译流程快速获得设计基线 (baseline)，并在未满足时序要求的情况下分析设计。如果初始实现后未能满足时序要求，请尝试其它建议的流程：

- 尝试多种 place\_design 指令（最多 10 个）和多次 phys\_opt\_design 迭代（Aggressive\*、Alternate\* 指令）。
- 使用 set\_clock\_uncertainty，在 place\_design/phys\_opt\_design 过程中对最关键的时钟（最大 0.500 ns）进行过约束。
- 使用 group\_path -weight，在必须满足时序要求的时序时钟上提升时序 QoR 的优先级。
- 完成小幅设计修改后使用增量编译流程，保留 QoR 并缩短运行时间。
- 运行由以下命令专为您的设计生成的前 3 项 ML 策略：report\_qor\_suggestions。
  - 表示使用 report\_qor\_suggestions 自动解决

改善时钟偏差流程



改善时钟偏差的方法

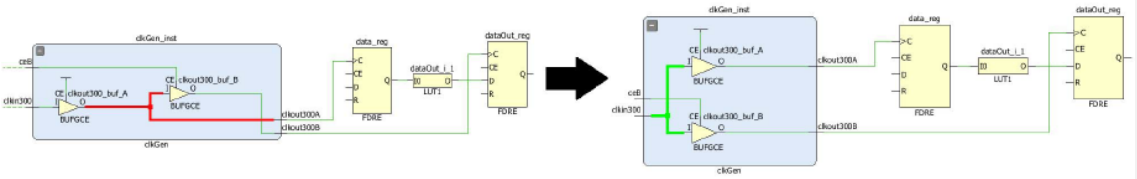
在异步时钟之间添加时序例外

如果时序路径中所含源时钟和目标时钟源自不同基准时钟或没有公共节点，那么这些时序路径必须作为异步时钟来处理。在此情况下，偏差可能极大，导致无法达成时序收敛。请根据需要添加 set\_clock\_groups、set\_false\_path 和 set\_max\_delay -datapath\_only 约束。欲知详情，请参阅 **UG949：在异步时钟之间添加时序例外**。

清理时钟树中使用的逻辑

除非在时钟逻辑中使用 DONT\_TOUCH 约束，否则 opt\_design 命令会自动清理时钟树。选择时序路径，启用 **“Clock Path Visualization”** 工具栏按钮 ，打开原理图 (F4) 以复查时钟逻辑。



- 通过删除不必要的缓冲器或将其并行连接可以避免在级联时钟缓冲器之间出现时序路径。例如：



- 在时钟等效的情况下，把并行时钟缓冲器合并为单个时钟缓冲器。
- 在时钟路径中移除 LUT 或任何组合逻辑，因为它们可使时钟延迟和时钟偏差不可预测。

匹配时钟布线

使用 CLOCK\_DELAY\_GROUP 提升关键同步时钟间的时钟布线延迟，即使两条时钟信号线已拥有相同的 CLOCK\_ROOT 也是如此。以下示例显示了 2 个不含 CLOCK\_DELAY\_GROUP 的同步时钟：<sup>1</sup>

Global Clock Resources										
Global Id	Source Id	Driver Type/Pin	Constraint	Site	Clock Region	Root	Load Clock Region	Clock Loads	Clock Period	Clock
 g0	src0	BUFG_GT/O	None	BUFG_GT_X1Y212	X5Y8	CLOCK_REGION_X5Y6	30	110934	3.184	app_clk
 g1	src0	BUFG_GT/O	None	BUFG_GT_X1Y215	X5Y8	CLOCK_REGION_X5Y6	2	5202	1.592	txoutclk_out[3]_3

将时钟负载布局约束到相关 I/O bank 旁

对于不足 2,000 负载的 I/O 逻辑与互连结构单元之间的时钟，请将时钟信号线上的 CLOCK\_LOW\_FANOUT 属性设置为将所有负载自动布局到时钟缓冲器 (BUFG\*) 所在的时钟区域内，并使插入延迟和偏差保持低位。<sup>1</sup>

将时钟负载布局约束到更小区域

您可以使用 Pblock，强制在更小区域内（例如 1 个 SLR）布局时钟信号线负载，以降低插入延迟和偏差，或避免跨特殊列发生，例如引入偏差损失的 I/O 列。

通过迁移物理源降低时钟信号线延迟

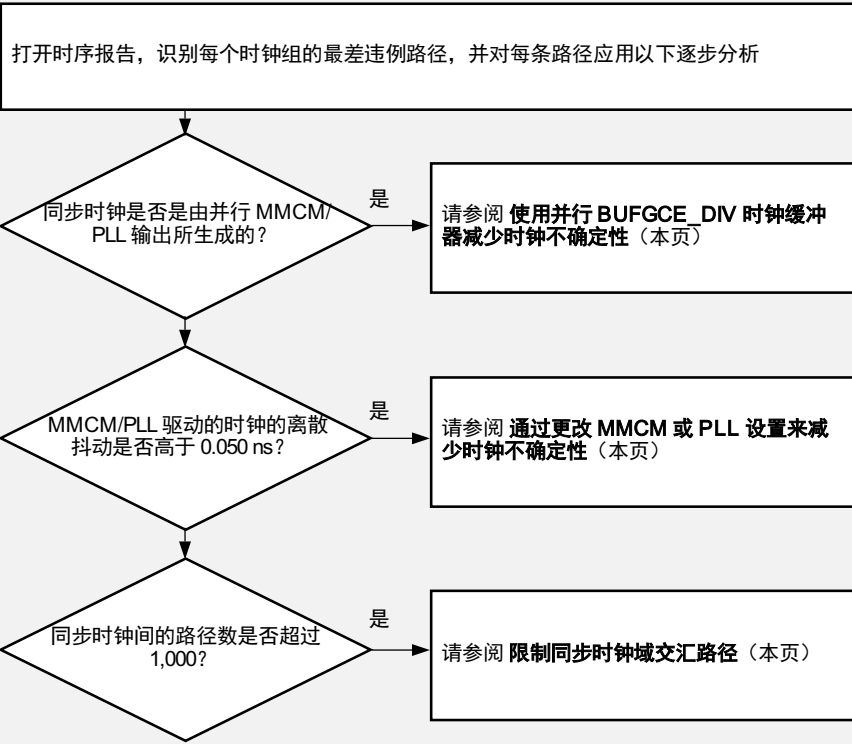
使用位置约束将源混合模式时钟管理器 (MMCM) 或锁相环 (PLL) 迁移到时钟负载中心，以降低最大时钟插入延迟，从而降低时钟消极因素以及偏差。欲知详情，请参阅 **(UG949)：降低 UltraScale 和 UltraScale+ 器件的偏差**。

-  表示使用 report\_qor\_suggestions 自动解决



改善时钟不确定性流程

时钟不确定性指输入抖动、系统抖动、离散抖动、相位误差或用户添加的不确定性的量。用户在理想时钟沿基础上添加不确定性的目的是为了对硬件工作条件进行准确建模。时钟不确定性同时影响建立时序路径和保持时序路径，且根据时钟树中使用的资源而变。

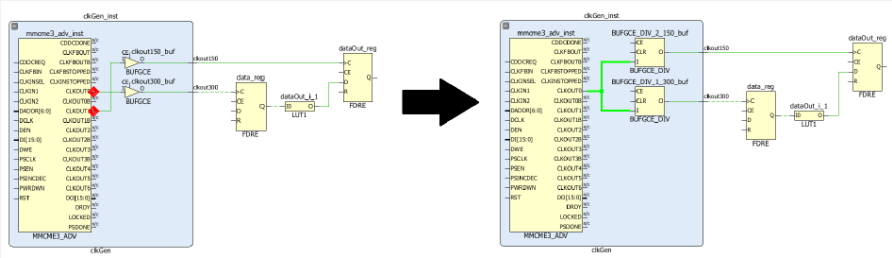


X21581-091811

改善时钟不确定性的方法

使用并行 BUFGCE\_DIV 时钟缓冲器减少时钟不确定性

对于由相同 MMCM 或 PLL 生成并受多个时钟输出驱动的周期比为 2、4 或 8 的同步时钟，仅使用 1 个 MMCM 或 PLL 输出并将其连接到并行 BUFGCE\_DIV 时钟缓冲器（限 AMD UltraScale™ 和 AMD UltraScale+™ 器件）。这种时钟拓扑能消除 MMCM 或 PLL 相位误差，大多数情况能实现 0.120 ns 的时钟不确定性。



以下显示了 150 MHz 时钟与 300 MHz 时钟间时钟域交汇 (CDC) 路径的时钟不确定性降低示例：

- 降低前的时钟不确定性：0.188 ns（建立时间）、0.188 ns（保持时间）
- 降低后的时钟不确定性：0.068 ns（建立时间）、0.000 ns（保持时间）

使用 Clocking Wizard 生成使用并行 BUFGCE\_DIV 缓冲器的时钟拓扑，同时设置时钟上的 CLOCK\_DELAY\_GROUP 属性。

通过更改 MMCM 或 PLL 设置来减小时钟不确定性

诸如 MMCM 和 PLL 等时钟修改块能以离散抖动和相位误差的形式来影响时钟不确定性。<sup>1</sup>

- 在 Clocking Wizard 内或使用 set\_property 命令，通过修改 M（乘法器）或 D（除法器）的值提高压控振荡器 (VCO) 频率。例如，MMCM (VCO=1 GHz) 会引入 167 ps 抖动和 384 ps 相位误差，而 MMCM (VCO=1.43 GHz) 对应值是 128 ps 和 123 ps。

限制同步时钟域交汇路径

由单独的时钟缓冲器驱动的同步时钟之间的时序路径表现出较高的偏差，因为公共时钟树节点位于时钟缓冲器之前，从而造成时序分析中的消极因素较高。因此，在这些路径上更难同时满足建立时间和保持时间要求，对于高频时钟（高于 500 MHz）尤其如此。要想确定两个时钟间的路径数量，请使用 report\_timing\_summary（时钟间路径部分）或 report\_clock\_interaction。以下示例显示了两个高速时钟间存在大量路径的设计（要求 = 1.592 ns）。其中 30% 的路径时序约束失败，说明其实现难度极大。

Source Clock	Destination Clock	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Path Req (WNS)	Inter-Clock Constraints
app_clk	txoutclk_out[3]_3	-0.348	-162.119	1668	5623	1.592	Timed
rxoutclk_out[0]_4	rxoutclk_out[0]_4	0.262	0.000	0	2998	2.388	Partial False Path
pcie_refclk	pcie_refclk	5.054	0.000	0	1508	7.960	Timed
txoutclk_out[3]_3	app_clk	-0.153	-0.196	2	1198	1.592	Partial False Path

请复查时钟域交汇涉及的逻辑，移除不必要的逻辑路径或尝试下列修改：

- 在受时钟使能控制的路径上添加多周期路径约束，因为并非在每个周期内都会传输新数据。
- 以增加时延为代价，使用异步交汇电路和合适的时序例外替代交汇逻辑。例如，使用异步 FIFO 或 XPM\_CDC 参数化宏。欲知详情，请参阅《UltraScale 架构库指南》(UG974)。
  - 表示使用 report\_qor\_suggestions 自动解决

QoR 评估报告概览

QoR 评估报告由以下几个部分组成：

- 1. **Overall Assessment Summary:** 即总体评估汇总，提供 QoR 评估得分和提升 QoR 的建议。
- 2. **QoR Assessment Details:** 即QoR 评估详情，显示每一项的 QoR 信息。对于得分低于 5 的所有项，都会在“Status”列中显示“REVIEW”。
- 提示：**要对标记为 REVIEW 状态的项的风险进行评估，请对阈值“Thresh.”列和实际值“Actual”列中的数据进行比较。阈值随设计和目标器件自动调整。要查看检查合格的项，请使用 -full\_assessment\_details 选项。
- 3. **Methodology Check Details:** 即方法检查详情，显示不合格的项，这些项与方法论有关并影响 QoR。
- 4. **ML Strategy Availability:** 即机器学习策略可用性，列出训练轮次中生成 ML 策略所需的指令（此处未显示详细信息）。

Overall Assessment Summary					
QoR Assessment Score		3 - Design runs have a small chance of success			
Flow Guidance		Run report_methodology and fix or waive critical warnings			
2. QoR Assessment Details					
Name	Thresh.	Actual	Used	Available	Status
Utilization					OK
Clocking					OK
Congestion					OK
Timing					
WNS	-0.100	-0.330	-	-	REVIEW
TNS	-0.100	-29.398	-	-	REVIEW
Paths above Net/LUT Budgeting	0	132	-	-	REVIEW
3. Methodology Check Details					
ID	Description			Criticality	No. Vio.
TIMING-1	Invalid clock waveform on Clock Modifying Block			Critical Warning	6
TIMING-9	Unknown CDC Logic			Warning	1
4. ML Strategy Availability					
Conditions for ML Strategy Availability		Value	Status		
opt_design directive		Explore	OK		
place_design directive			-		
phys_opt_design directive			-		
route_design directive			-		

QoR 评估报告详情

QoR 评估得分

“QoR Assessment Score” 用于估算设计满足时序目标的可能性，如下所示：

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

不同设计阶段下的 QoR 评估得分准确性

AMD 建议在流程早期使用 QoR Assessment Score，因为在此阶段使用能节省大量编译时间。但在流程中不同阶段，分析结果的准确性不尽相同。该 report\_qor\_assessment 命令会根据设计的当前状态自动调整分析，并将流程后期可能执行的优化一并纳入考量。以下是每个设计状态下执行的分析：

- Unplaced:** 即未布局状态，用于分析单元利用率、时钟偏差阈值更高，并执行 LUT/信号线预算检查。不执行拥塞分析。
- Placed:** 即已布局状态，执行拥塞分析，并按更严格的时间偏差阈值执行分析。不执行 LUT/信号线预算检查。
- Routed:** 即已布线状态，分析和阈值都完全精确。

注释：得分为基于约 +/-1 准确性范围内可用的最佳指标所得。

提升 QoR 评估得分

在运行 report\_qor\_assessment 之后，运行 report\_qor\_suggestions 即可获取有关如何修复或降低编译失败风险的建议。如有建议可供提供，那么 QoR 建议报告将自动按评估得分从低到高顺序列出各项（在 QoR 评估报告中标记为 REVIEW 状态的项）。

SLR 和 Pblock 分析

SLR 和 Pblock 分析是自动执行的。报告仅包含得分低于 5 的项。

运行进一步分析

使用 -csv\_output\_dir 选项输出以下 CSV 文件，用于进一步分析：

- qor\_timing\_<design\_stage>.csv: 提供未通过信号线和 LUT 预算检查的时序路径详细信息。
- qor\_dont\_touch\_<design\_stage>.csv: 列出叶节点单元/分层单元以及含有 DONT\_TOUCH 属性的信号线。

提示：在工程模式下，请使用以下 Tcl 命令添加评估报告：

set\_property STEPS.OPT\_DESIGN.TCL.POST <path>/postopt.tcl [get\_runs impl\_\*]

以下提供了 postopt.tcl 的示例：

report\_qor\_assessment -file postopt\_rqa.rpt -csv\_output\_dir ./rqa

本文档为英语文档的翻译版本，若译文与英语原文存在歧义、差异、不一致或冲突，概以英语文档为准。  
译文可能并未反映最新英语版本的内容，故仅供参考，请参阅最新版本的英语文档获取最新信息。

