FPGA 静态时序分析与时序约束

–ALIENTEK 开拓者/新起点 FPGA开发板教程

广州市星翼电子科技有限公司

淘宝店铺 1：http://eboard.taobao.com

淘宝店铺 2：http://openedv.taobao.com

技术支持论坛 (开源电子网)：www.openedv.com

官方网站：www.alientek.com

最新资料下载链接：http://www.openedv.com/posts/list/13912.htm

E-mail: 389063473@qq.com **QQ:** 389063473

咨询电话：020-38271790

传真号码：020-36773971

团队：正点原子团队

正点原子，做最全面、最优秀的嵌入式开发平台软硬件供应商。

友 情 提 示

如果您想及时免费获取“正点原子”最新资料，敬请关注正点原子微信公众平台，我们将及时给您发布最新消息和重要资料。

关注方法：

(1)微信“扫一扫”，扫描右侧二维码，添加关注

(2)微信→添加朋友→公众号→输入“正点原子”→关注

(3)微信→添加朋友→输入“alientek_stm32”→关注

第一章 静态时序分析与时序约束

静态时序分析是检查芯片时序特性的一种方法，可以用来检查信号在芯片中的传播是否符合时序约束的要求。相比于动态时序分析，静态时序分析不需要测试矢量，而是直接对芯片的时序进行约束，然后通过时序分析工具给出时序分析结果，并根据设计者的修复使设计完全满足时序约束的要求。

本章包括以下几个部分：

1.1 静态时序分析简介

1.2 FPGA 设计流程

1.3 TimeQuest 的使用

1.4 常用时序约束

1.5 时序分析的基本概念

1.1 静态时序分析简介

静态时序分析（Static Timing Analasis，STA）用来验证电路的性能，找到时序违规路径，并指导EDA工具对设计进行布局布线，以满足时序要求。静态时序分析的速度很快，但是它并不对电路的功能进行验证。

时序约束（Timing Constraints）用来描述设计人员对时序的要求，比如时钟频率，输入输出的延时等。比如，对时钟频率的约束最简单的理解就是，设计者需要告诉EDA工具设计中所使用的时钟的频率是多少；然后工具才能按照所要求的时钟频率去优化布局布线，使设计能够在要求的时钟频率下正常工作。

Intel Quartus软件中的时序分析工具TimeQuest Timing Analyzer使用工业标准的时序约束和分析方法，通过检查信号的到达时间是否符合约束所要求的时间，从而决定使设计正常工作所需要满足的时序关系。

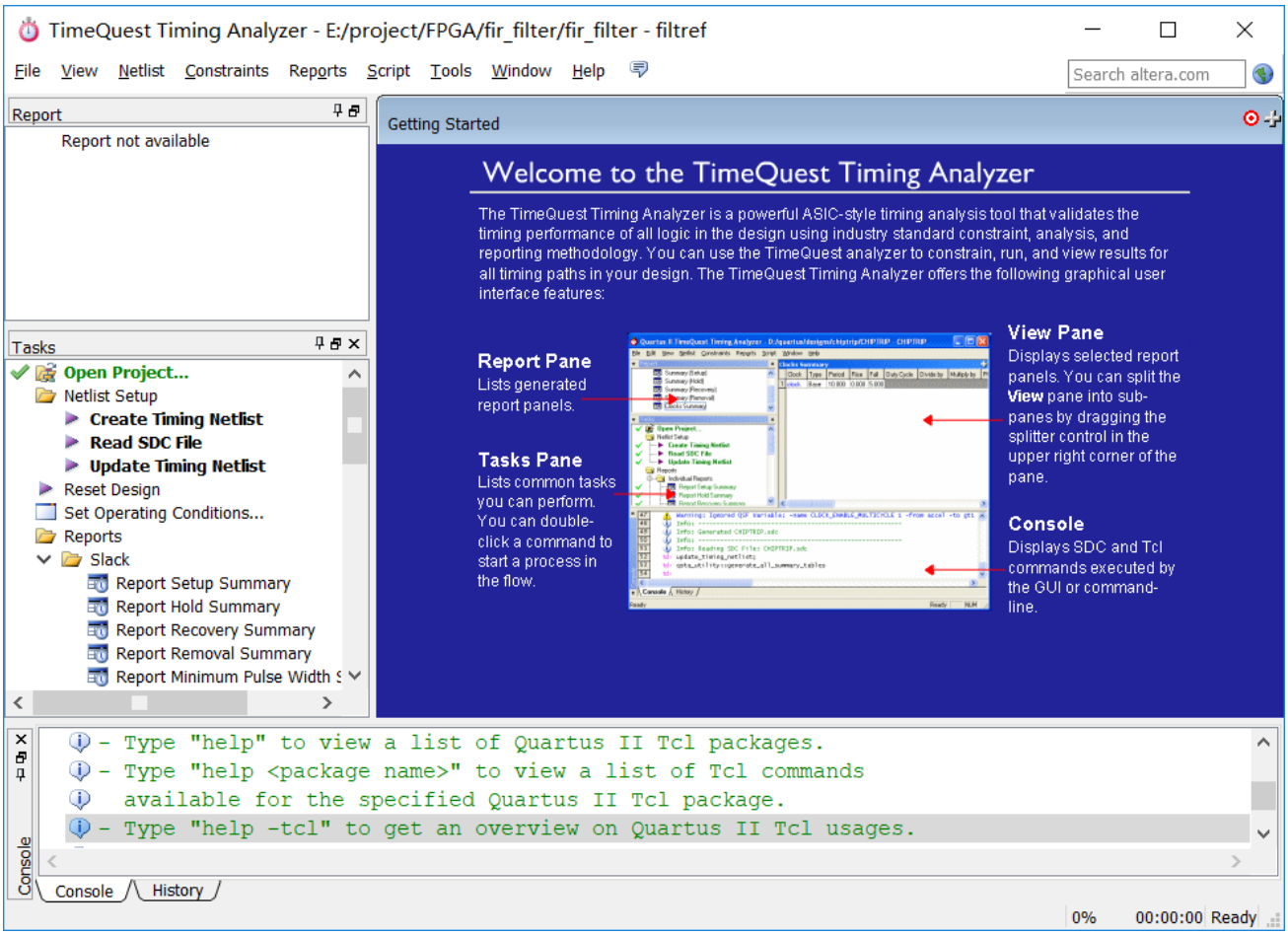


图 1.1.1 时序分析工具TimeQuest Timing Analyzer

在正点原子《开拓者FPGA开发指南》里的许多例程中，无论设计的复杂程序如何，大都

用到了时序约束以实现功能。如果没有给设计添加时序约束，则有可能会影响到最终的功能，而且问题不容易排查。例如，在第四十八章《基于以太网的板对板音频互传实验》，未添加时序约束之前，通过以太网传输的音频有噪声，添加时序约束后噪声消失。

在上面的例子中，由于未添加时序约束，设计所实现的功能受到了一些影响。而更为严重的情况是，在一些设计中由于没有进行时序约束，而导致设计的功能根本无法实现。比如第五十二章《FIR滤波器实验》中，在没进行时序约束之前，设计的低通滤波器根本无法工作。

下面分别给出了在时序约束前后，红外摄像头所拍摄的两幅图像，在图 1.1.2中可以明显看到由于未进行时序约束，对图像质量造成了严重的影响。

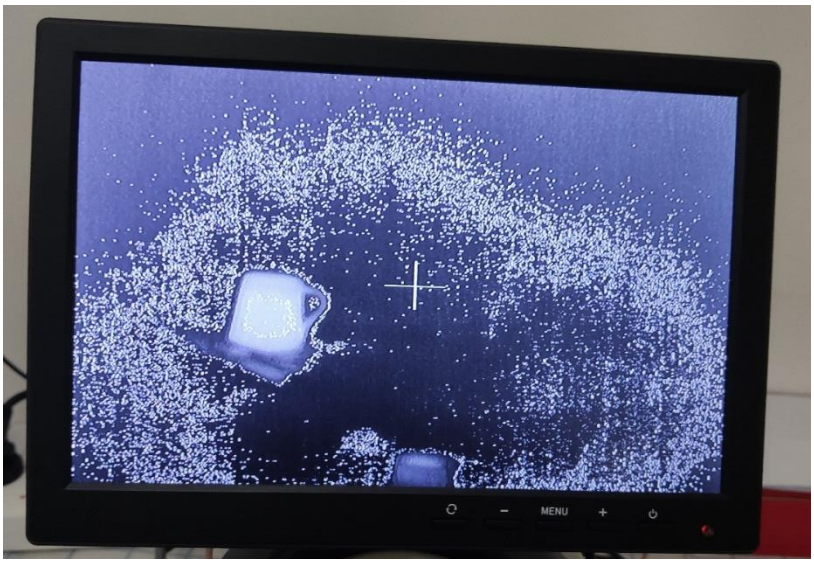


图 1.1.2 时序约束前的红外图像

图 1.1.3 时序约束后的红外图像

既然在FPGA的设计过程中时序约束如此重要，那么为什么在前面的例程中很少提到呢？一方面是因为我们的设计功能比较简单，像流水灯、数码管这样简单的外设，即使不进行时序约束，也不影响最终的功能；另一方面是由于设计所工作的时钟频率比较低，大多数时候只要求设计能够在50MHz的时钟频率下正常工作。

当设计变得复杂起来，或者系统的时钟频率比较高的时候，如果不添加时序约束，那么就有可能在验证设计结果的时候出现一些意料之外的情况。最常见的一个问题是，在调试的过程中，设计上一次还正常工作，在重新编译一次之后再次下载验证结果就不对了；或者说在SignalTap中添加或删除了某些信号，实验结果就不一样了。

这些现象通常会使初学者百思不得其解，因为设计本身的逻辑功能是没有改动的，但是实验结果却有可能不同。这实际上很可能就是时序的问题，因为工具在多次编译时布局布线的结果是很有可能不一样的，由于未添加时序约束，某些布局布线方式没能满足时序要求，最终导致实验结果出错。

在学习时序约束以及静态时序分析之前，我们有必要先了解一下FPGA设计流程。

1.2 FPGA 设计流程

我们先来看一下使用TimeQuest Timing Analyzer进行时序约束和分析的设计流程。

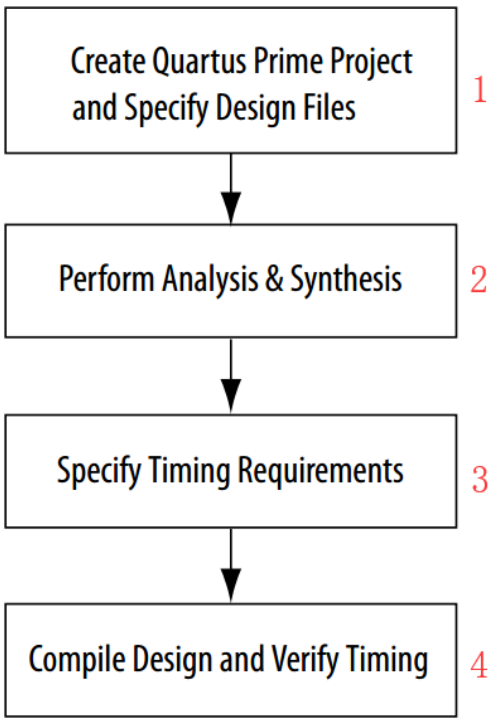


图 1.2.1 FPGA设计流程

首先我们在Quartus II中创建工程，并指定或输入设计文件（如.v格式的文件）。然后我们需要执行分析和综合（Analysis & Synthesis）过程，这一步可以在Quartus II中Tasks栏的导航面板中双击“Analysis & Synthesis”来实现，也可以直接点击工具栏中的“Start Analysis & Synthesis”按钮，如图 1.2.2所示。

上面这两个步骤是我们在使用Quartus II进行设计开发常用的操作，相信大家也比较熟练了。问题是工具在执行分析和综合过程的时候，做了什么事呢？

在分析（Analysis）阶段，工具会检查我们的设计有没有错误，比如源文件中的语法错误等；然后在综合（Synthesis）阶段，工具会把设计中的源文件转换成门级电路网表（netlist）；最后把门级网表中的各个元素与FPGA里的基本元件逐一对应起来，这就是映射（Map）。综合过程的示意图如图 1.2.3所示。

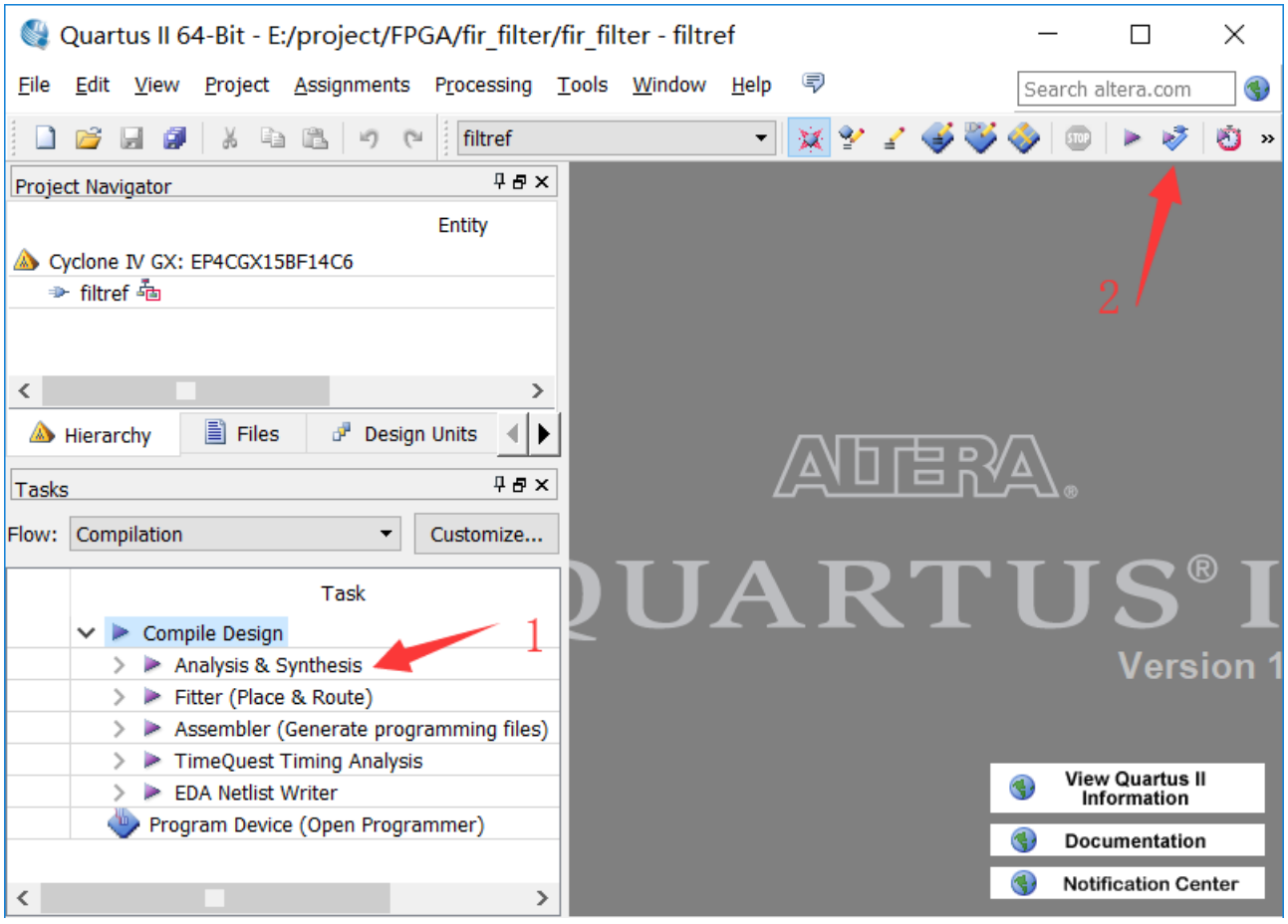


图 1.2.2 分析和综合

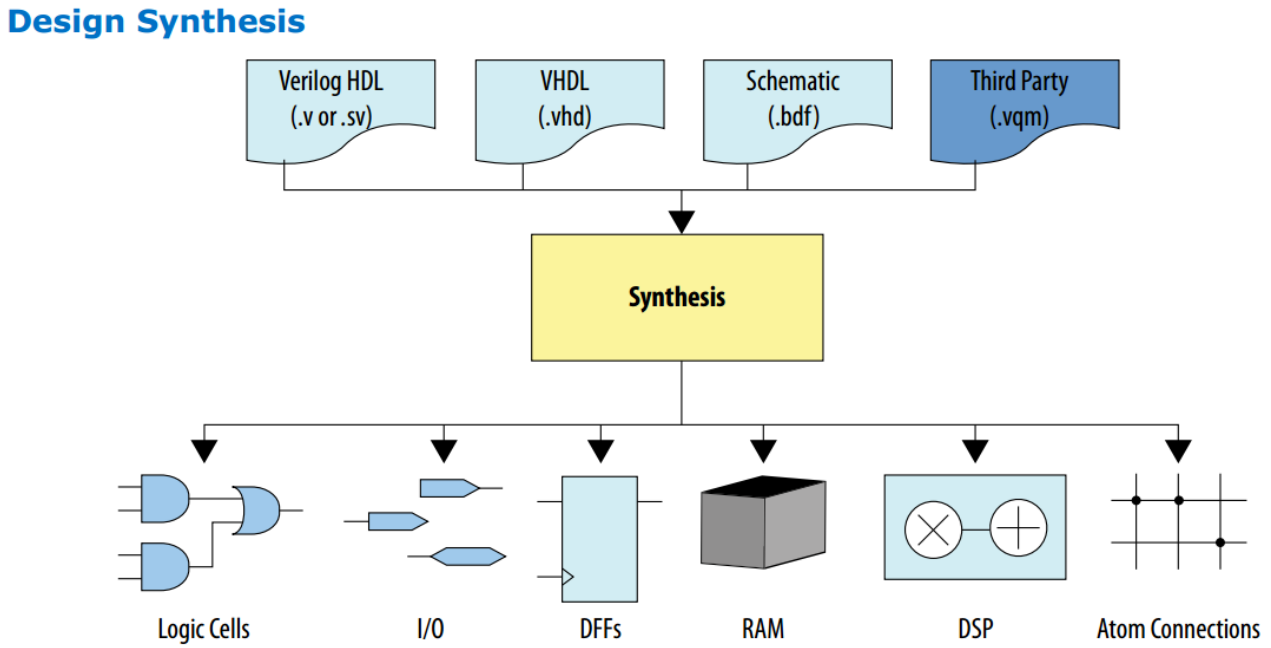


图 1.2.3 综合过程示意图

在综合之后，我们就进入了图1.4 中FPGA设计流程的第3个阶段：指定时序要求（Specify Timing Requirement）。所谓的指定时序要求，就是我们常说的时序约束。在这一过程中，Quartus II使用ASIC设计中常用的Synopsys Design Constraints（SDC）格式的文件来描述对时序的要求。我们可以直接创建并编辑SDC文件，当然这要求我们对SDC约束的语法及格式比较熟悉；对于初学者来说，更直观的方法是在TimeQuest Timing Analyzer工具中使用GUI界面来创建约束。

在指定时序要求之后，我们对设计进行一次全编译（Compile Design），在此过程中，Quartus II中的适配器（Fitter）会对综合后的结果进行布局和布线（Place & Route）。所谓的布局布线（Place & Route）是指把综合过程中映射到FPGA中的各种硬件资源（如逻辑单元、IO、RAM等）放到FPGA芯片上的合适位置，并用可编程互连线把它们连接起来。

布局布线的结果影响到设计最终的性能（如最大工作频率Fmax），该过程虽然是由EDA工具来完成，但是工具并不知道设计需要工作在什么样的环境下（比如输入时钟频率是多少、输入输出信号的延迟等），也就不清楚应该把布局布线结果优化到何种程度。因此工具需要一个说明文件来告诉它这些信息，从而来指导布局布线的过程，以满足设计者的需求。而这个说明文件的内容，就是我们前面所指定的时序要求，也就是时序约束。

工具会努力按照设计者所提供的时序约束去优化布局布线结果，在布局布线结束之后，我们需要验证最终结果是否满足时序要求（Verify Timing），这个过程就是静态时序分析（STA）。如果不满足时序要求，我们就需要利用静态时序分析工具所提供的各种报告，去

查找原因，进而修改我们的设计，或者完善我们的时序约束，然后再次进行编译及时序分析流程。

在初学者刚接触FPGA设计时，一般只注重逻辑功能的设计，代码仿真没问题之后，就直接交给工具对工程进行全编译，然后开始下载验证、在线调试等过程。这其实还是软件开发的流程，然而FPGA开发属于硬件设计的范畴，如果想要成为高手，就需要了解FPGA底层的资源，了解从设计到实现过程中工具帮我们做了哪些事，并学习如何指导工具去帮助我们完成设计。

接下来我们就来学习一下，如何在Quartus II中使用TimeQuest Timing Analyzer对设计进行时序约束以及静态时序分析。

1.3 TimeQuest 的使用

首先我们需要打开一个安装Quartus II时所提供的一个示例工程，将Quartus II安装目录下的quartus/qdesigns文件夹中的fir_filter文件夹拷贝到一个英文目录中，该目录只能由字母、数字和下划线组成。fir_filter文件夹的位置如下图所示：

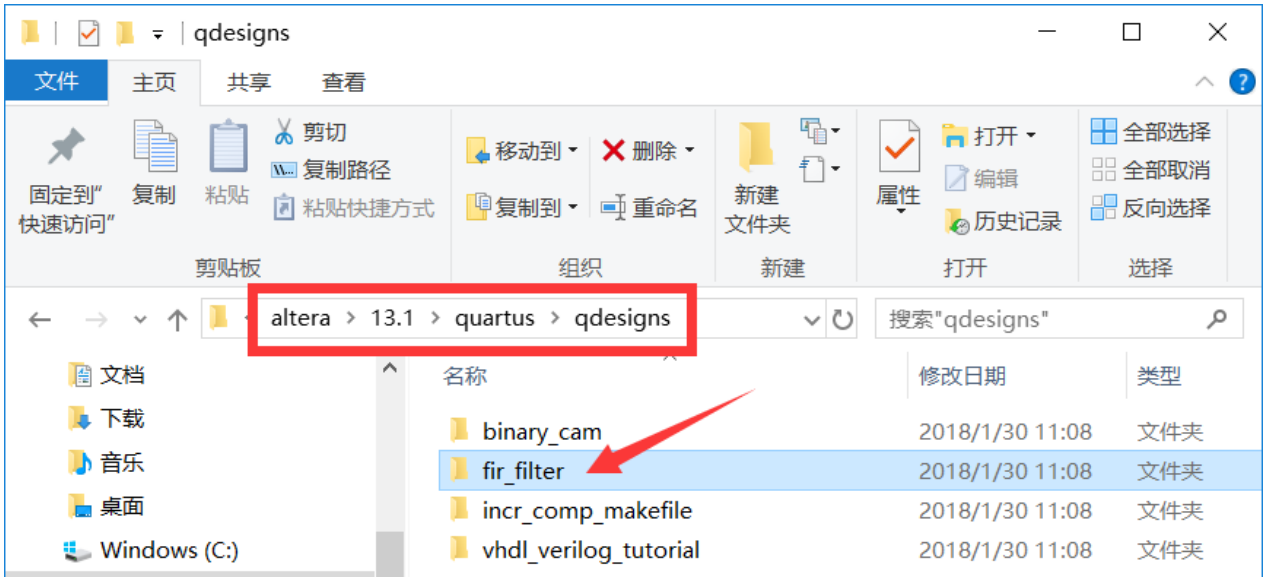


图 1.3.1 示例工程fir_filter

在拷贝出来的fir_filter文件夹中双击“fir_filter.qpf”文件在Quartus II中打开工程，然后在工具栏中点击“Start Analysis & Synthesis”（下图按钮1）执行综合过程。综合完成后点击TimeQuest Timing Analyser”（下图按钮2）来运行TimeQuest Timing

Analyzer。

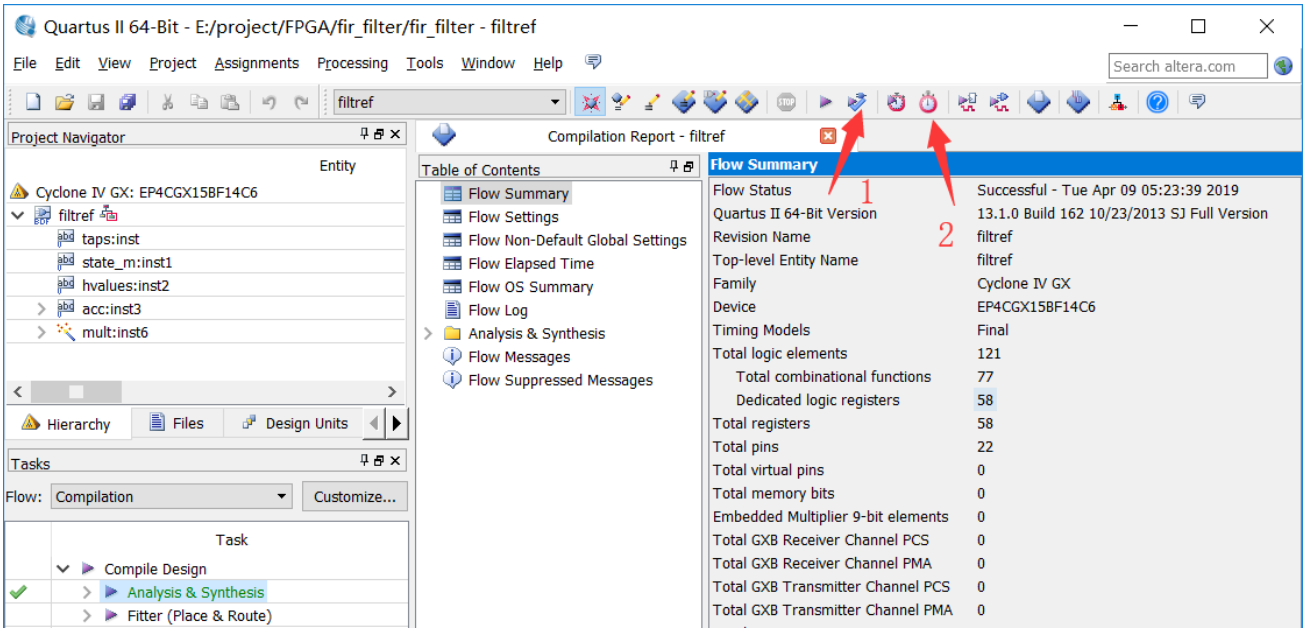


图 1.3.2 综合并运行TimeQuest Timing Analyzer

TimeQuest Timing Analyzer打开后界面如下图所示：

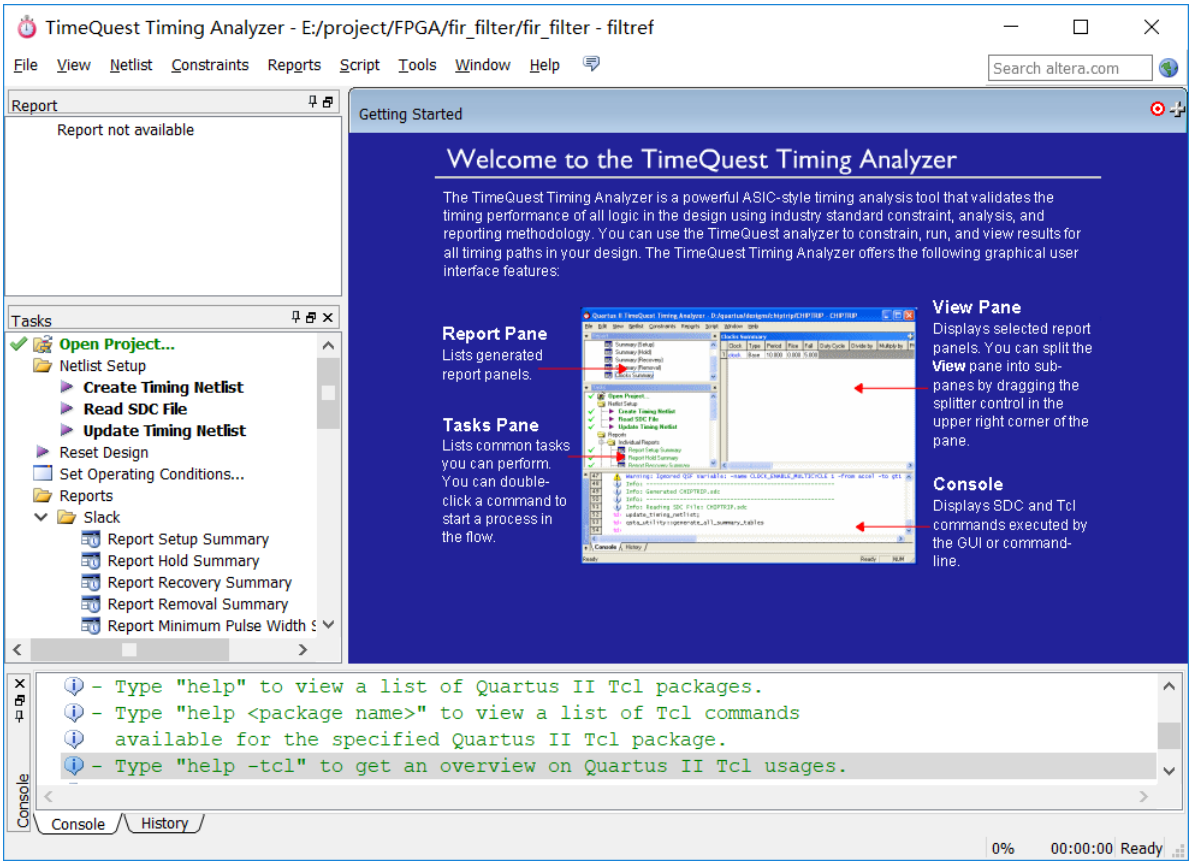


图 1.3.3 TimeQuest Timing Analyzer界面

接下来，创建一个 Post-Map 时序网表。 在Netlist菜单上，点击Create Timing Netlist。出现Create Timing Netlist对话框。在 Input netlist 中，选择 Post-Map，点击 OK。

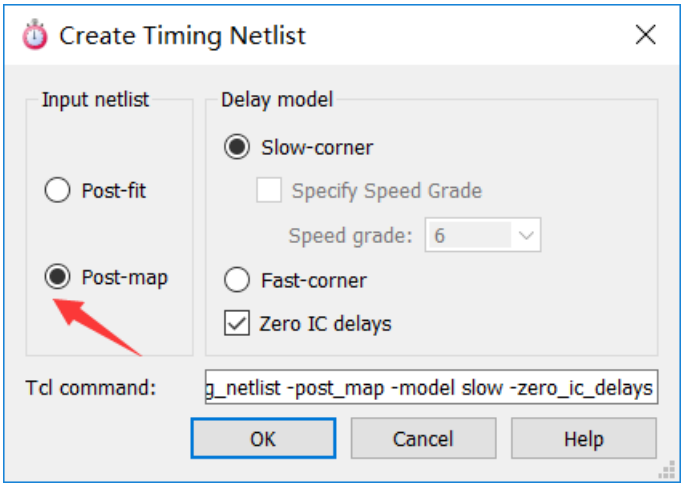


图 1.3.4 创建时序网表

注意，不能在Tasks面板中使用Create Timing Netlist命令来创建一个post-map时序网表。默认情况下， Create Timing Netlist 需要一个 post-fit 数据库。

接下来我们就可以指定时序要求了，首先我们来看一下这个例程顶层模块的原理图：

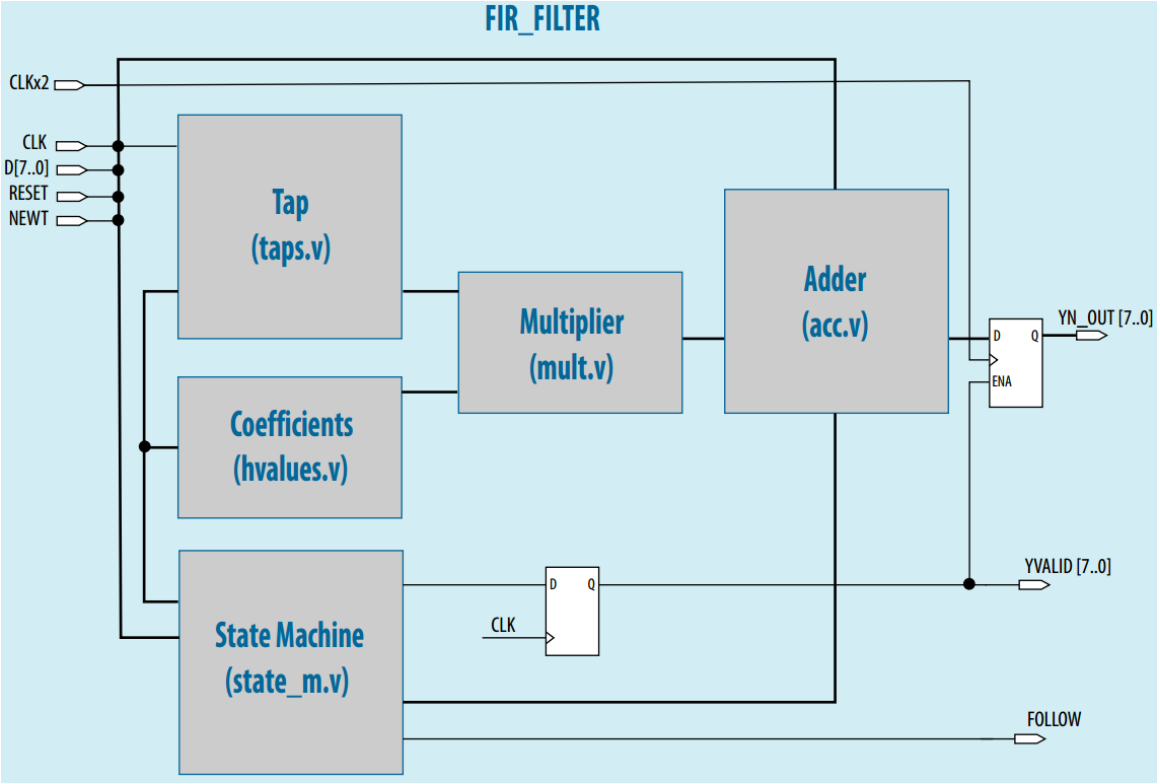


图 1.3.5 FIR滤波器原理图

从原理图中可以看到，该例程顶层模块有两个输入时钟clk和clkx2。它们的属性如下图所示：

时钟端口名称	要求
clk	50/50 占空比的 50 MHz
clkx2	60/40 占空比的 100 MHz

图 1.3.6 输入时钟属性

我们需要输入的第一个时序要求就是告诉工具每一个时钟的频率及占空比信息，这个过程需要通过创建时钟并分配时钟端口来实现。

在 Constraints 菜单中，点击Create Clock。出现 Create Clock 对话框。在该对话框中设置时钟名称，并按照上面列出的时钟属性计算时钟周期以及上升沿和下降沿的时刻，然后在“Target”一栏选择对应的端口。

在我们设置该对话框时，对话框中SDC command一栏列出了等效的SDC约束命令，我们在手动创建SDC约束文件的时候，可以通过输入该命令来创建时钟约束。默认情况下，如果未使用 -waveform 选项，那么 create_clock 命令假设 50/50 的占空比。

因为有两个时钟，所以该过程需要执行两次，下图列出了创建时钟clkx2时的设置：

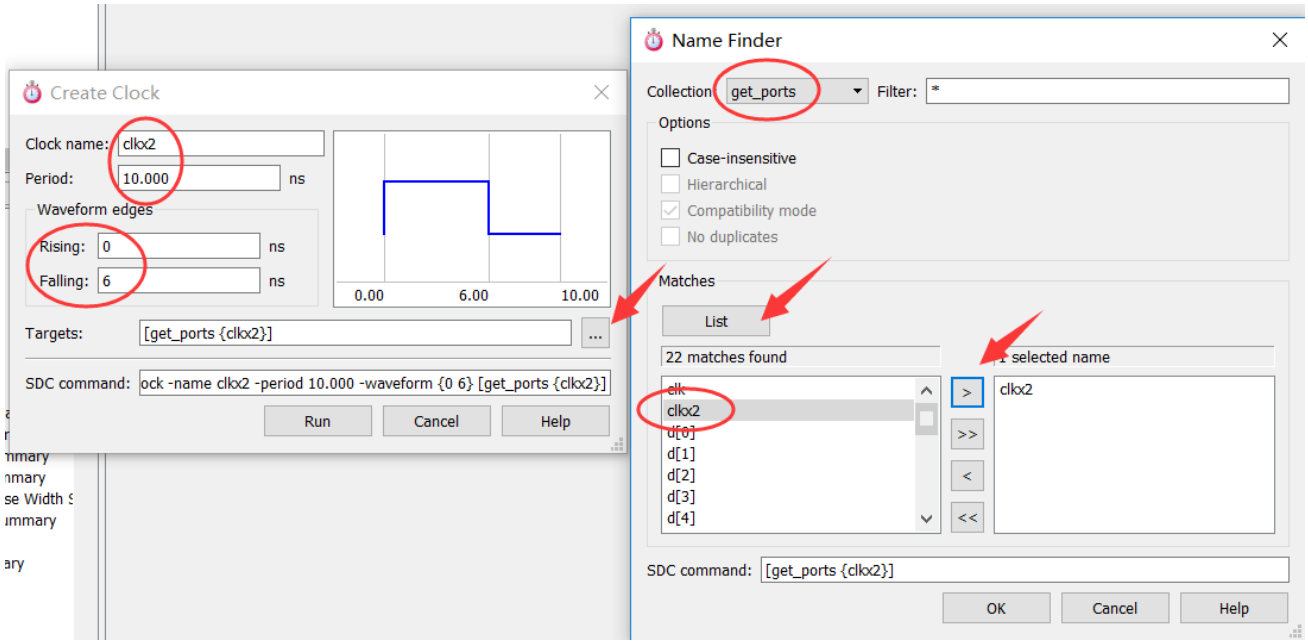


图 1.3.7 创建时钟clkx2

创建时序约束后，需要对时序网表进行更新，将所有时序要求应用到时序网表 （新的

clk 和 clkx2 时钟约束）。只要应用了新的时序约束，就必须对时序网表进行更新，方法是在 Tasks 面板中，双击 Update Timing Netlist 命令。

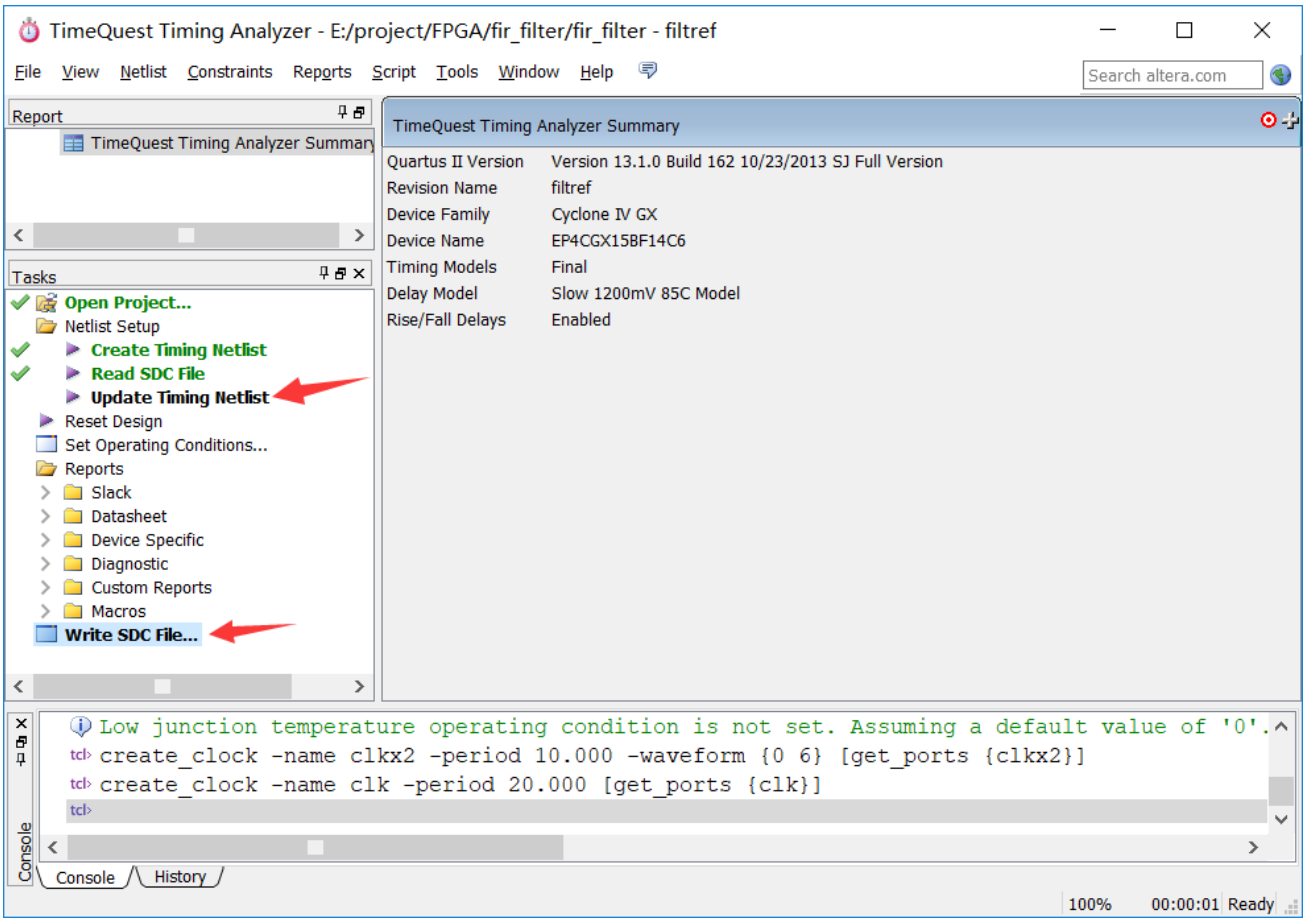


图 1.3.8 更新时序网表

在为设计指定时钟约束并更新时序网表后，需要创建 SDC 文件并将上面的约束保存到SDC文件中。通过 TimeQuest Timing Analyzer GUI 或者在控制台（console）中指定的约束不会自动保存。

在 Tasks 面板中，双击 Write SDC File 命令。出现 Write SDC File 对话框。在 File Name 栏输入 filtref.sdc。

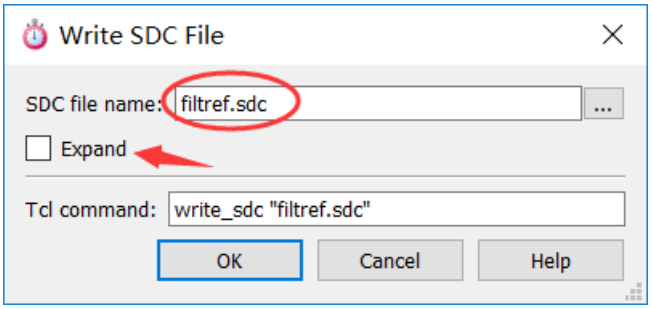


图 1.3.9 创建SDC文件

通过以上步骤我们定义了两个时钟，在指定时序约束并更新时序网表后，接下来我们可以生成时序报告，以验证所有时钟被正确地定义并应用到正确的节点。

首先生成 **SDC 约束报告**(SDC Assignments Report)，SDC Assignments 报告了在指定设计中包含的所有时序约束。在 Tasks 面板中，双击 **Report SDC** 命令。

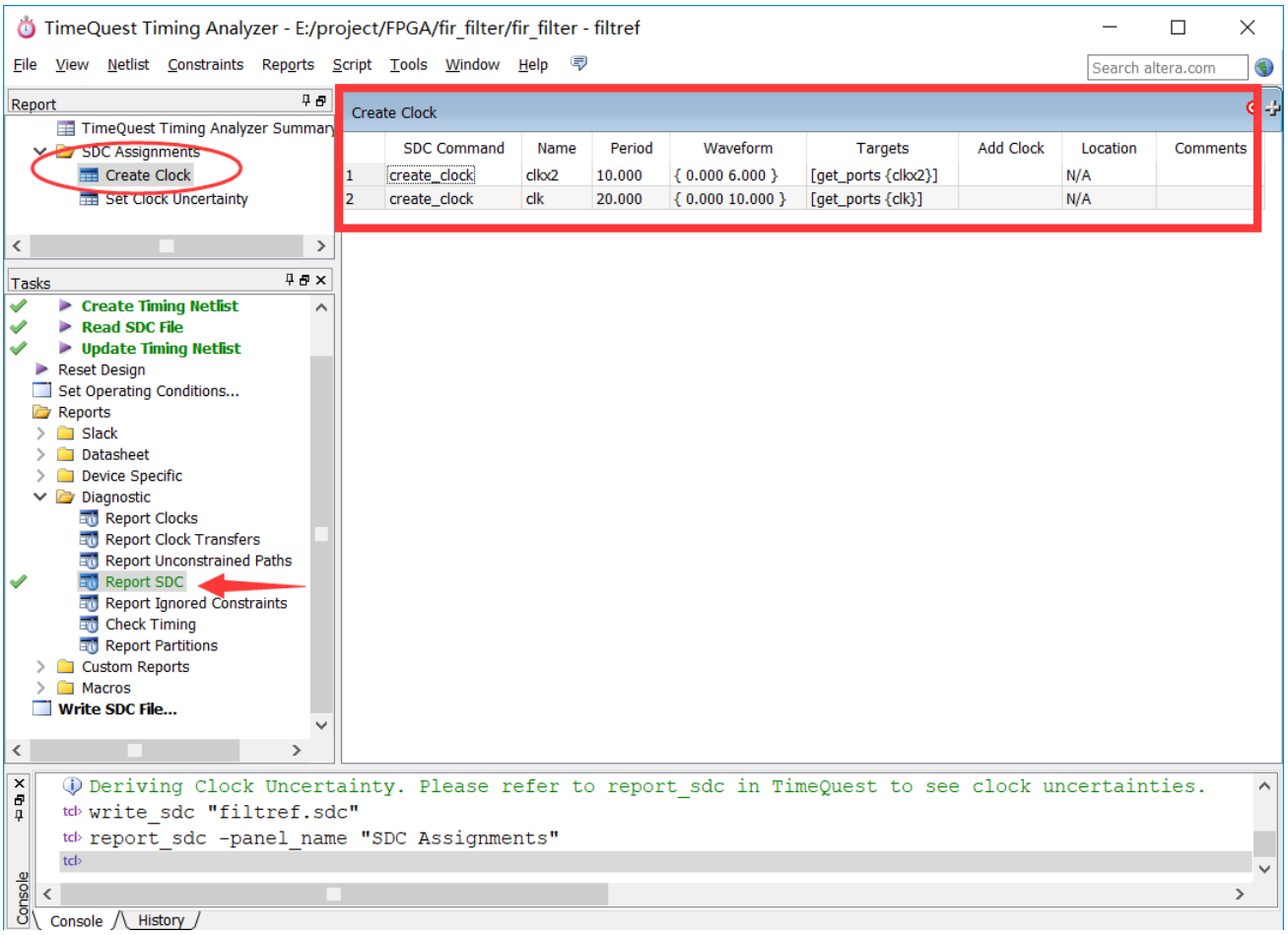


图 1.3.10 SDC约束报告

生成**时钟总结报告**（Clocks Summary Report)，来总结设计中所有的时钟。在 Tasks 面板中，双击 **Report Clocks** 命令。

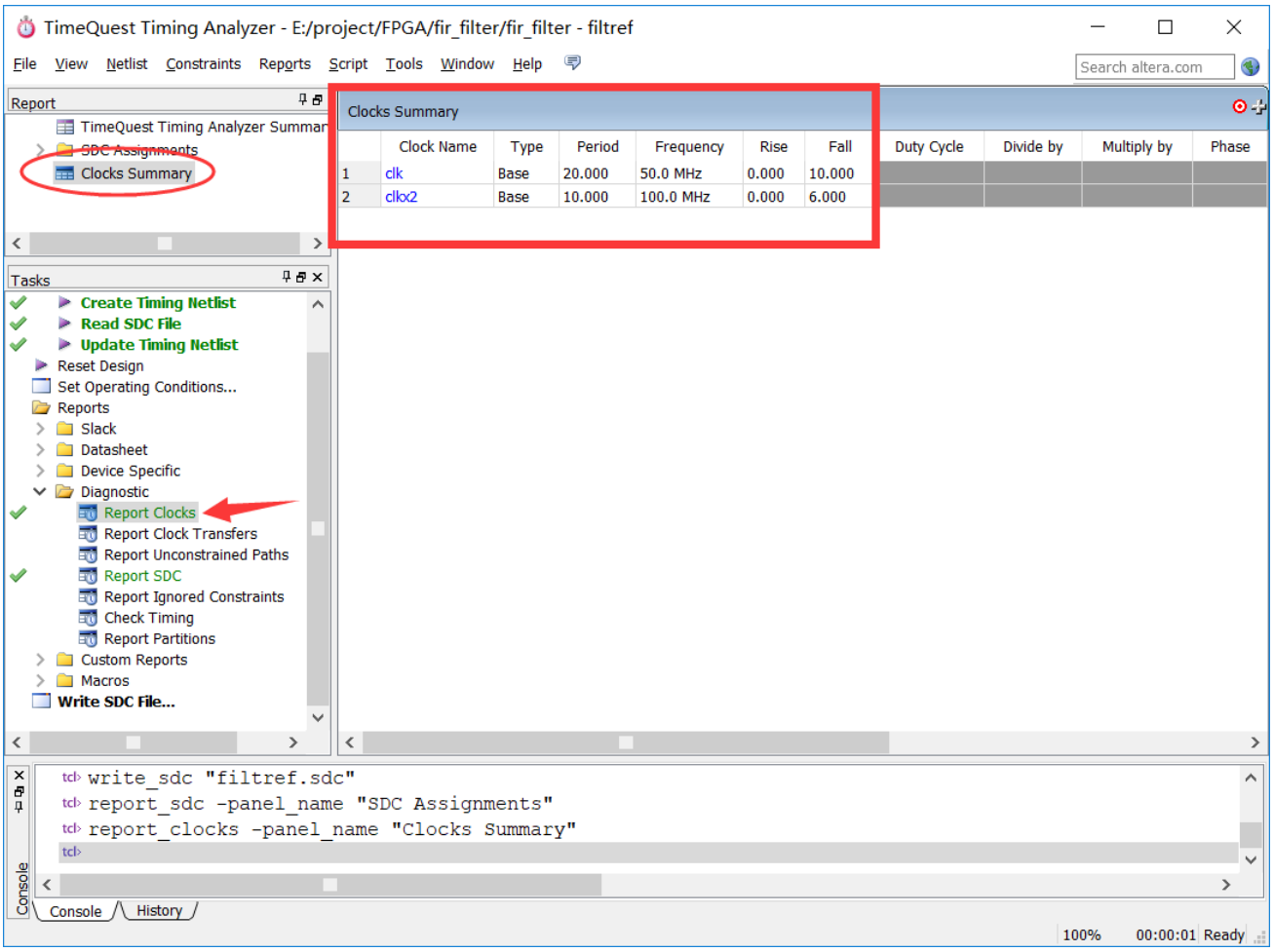


图 1.3.11 时钟总结报告

生成**时钟传输报告**（Clock Transfers Report），使用 Report Clock Transfers 命令生成一个报告来验证所有的时钟到时钟传输都是有效的。这种报告包含设计中所有的时钟到时钟传输。在 Tasks 面板中，双击 **Report Clock Transfers** 命令。

如下图所示，Clock Transfers 报告表明在 clk （源时钟）和 clkx2 （目的时钟）之间存在跨时钟域路径，共有16条路径，其中 clk 为源节点提供时钟， clkx2 为目的节点提供时钟。

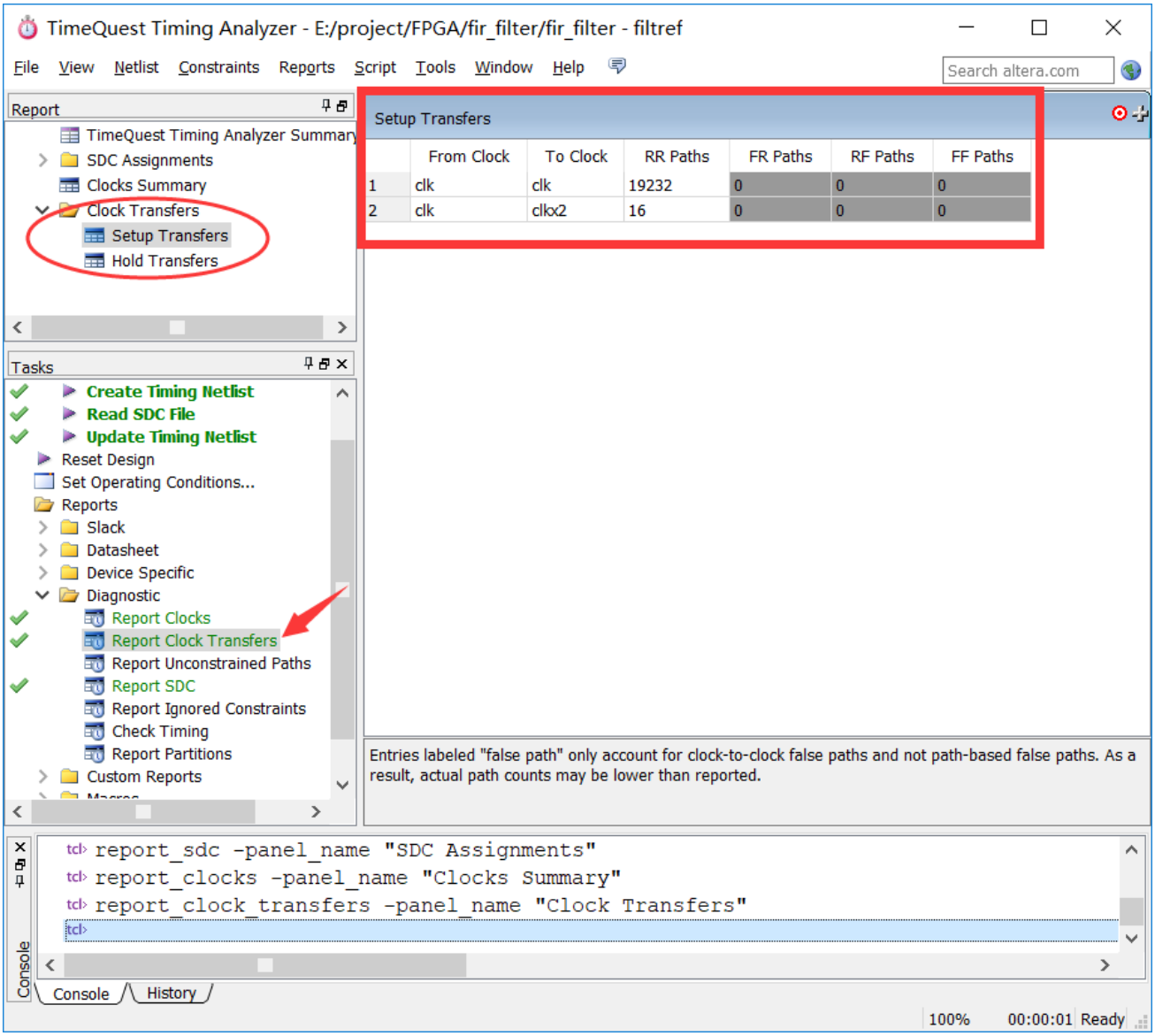


图 1.3.12 时钟传输报告

在 fir_filter 设计中，不必分析 clk 至 clkx2 的时钟传输，因为它们是忽略路径。通过以下方式声明 clk 至 clkx2 的路径为伪路径：

在 Clock Transfers 报告中，在 From Clock 列选择 clk，右击并选择 “Set False Paths”这个命令表明将所有由 clk 驱动的源寄存器到由 clkx2 驱动的目的寄存器之间的路径设为伪路径。

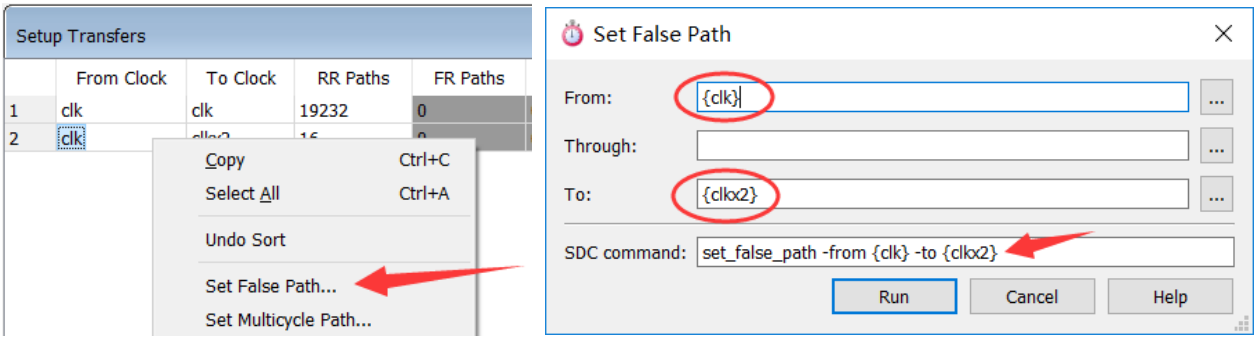


图 1.3.13 设置伪路径

另外，也可以使用 set_clock_groups 命令来声明两个时钟域之间的路径为伪路径。例如， set_clock_groups -asynchronous -group [get_clocks clk] -group [get_clocks clkx2]。该命令表明clk 到clkx2 以及 clkx2 到clk 的所有路径为伪路径。这是一种更加推荐的方法。

当完成该程序后，TimeQuest Timing Analyzer 变成黄色表明当前 Clock Transfers 报告是过时的。

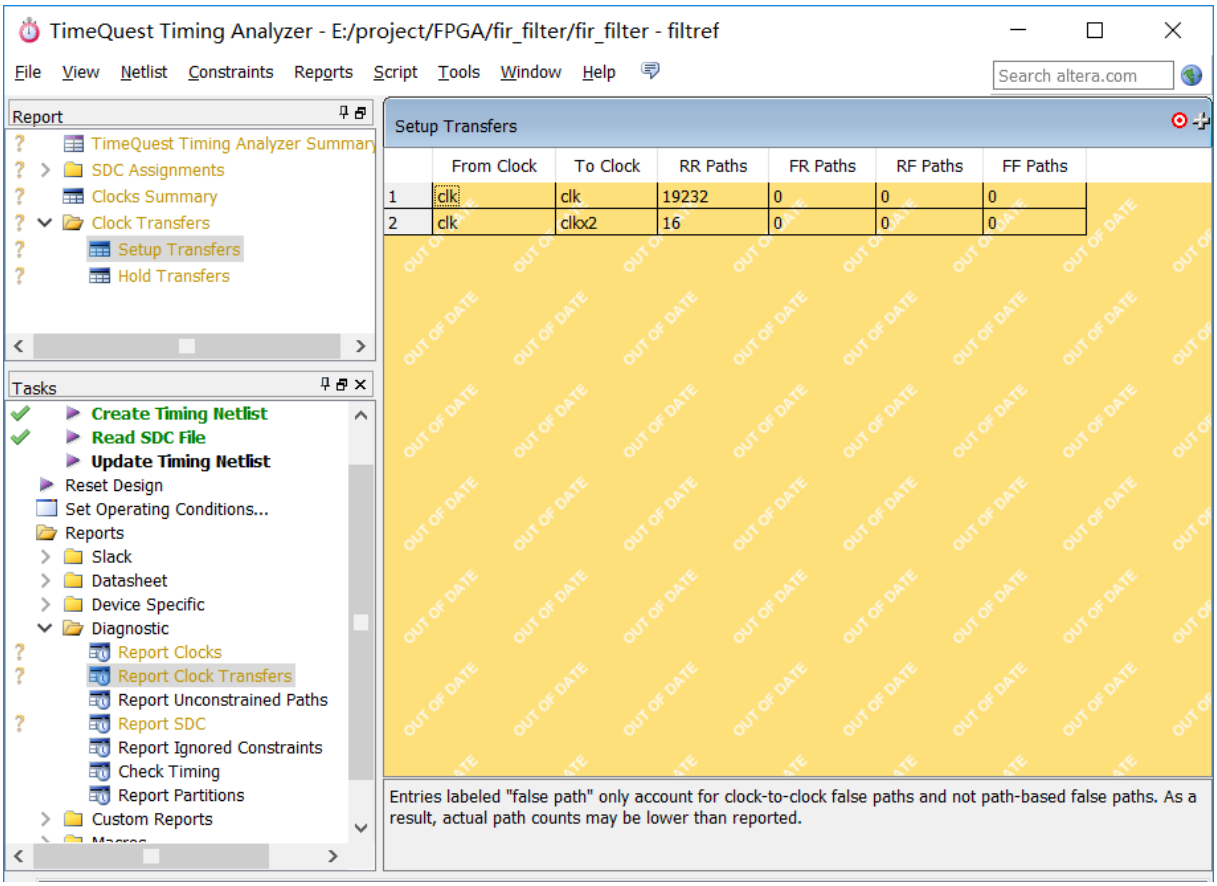


图 1.3.14 过时的报告

由于在此处添加了一个新的时序约束（设置伪路径），通过在 Tasks 面板中，双击 Update Timing Netlist 命令，更新时序网表 。

在 GUI 中输入 set_false_path 后，所有生成的报告面板上都标有 “Out of Date”，这表明报告面板不包含反映 TimeQuest Timing Analyzer 中当前状态的约束的结果。要更新报告面板，必须重新生成所有的报告：右击报告面板列表中任何过时(out-of-date)的报告，并选择Regenerate all。

重新生成报告后，可以看到在SDC Assignments报告栏多了一个“Set False Path”约束。同时打开Clock Transfers报告可以看到由clk到clkx2的路径被声明为伪路径，意味着时序分析工具将不再分析这些路径，像这种约束我们称之为例外（exceptions）。

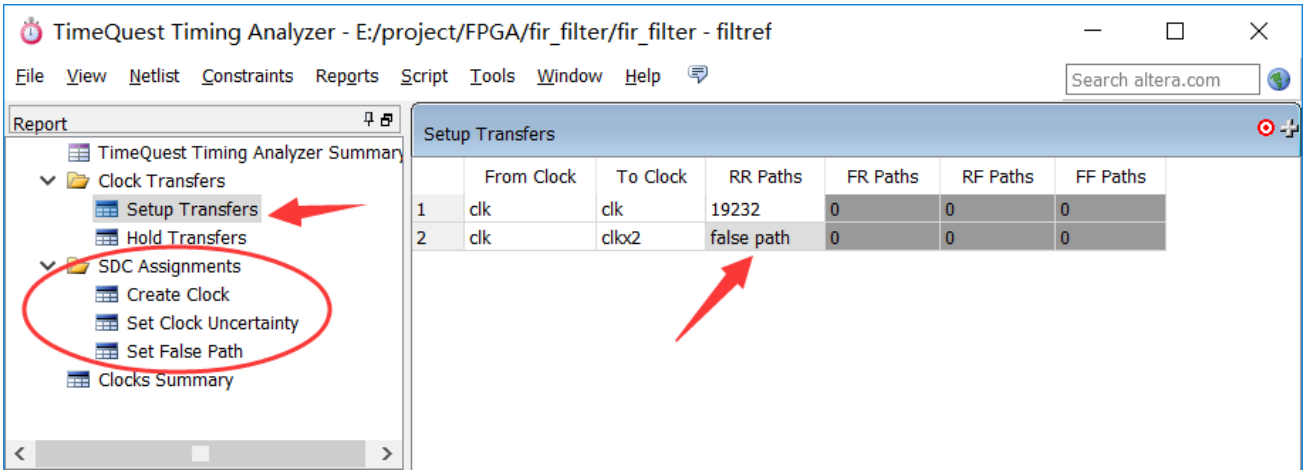


图 1.3.15 伪路径

在指定设计中所有的时钟约束和伪路径后，在 Tasks 面板中，双击 Write SDC File。出现 Write SDC File 对话框，在 File name 栏，输入 filtref.sdc，将时序约束和例外保存到 SDC 文件。

这一过程覆盖之前所创建的filtref.sdc文件，新的约束文件包含两个时钟约束和一个伪路径例外。保存约束到 SDC 文件后，在设计上运行一个全编译以优化布线，从而符合约束。不过，在开始全编译之前，需要将 SDC 文件添加到工程中。

在Quartus II软件的Assignments菜单栏中选择“Settings”，然后在左侧Category一栏选中“Files”，在右侧通过浏览来选择 filtref.sdc，点击“Add”将其添加到工程中，然后先后点击下方的“Apply”、“OK”。

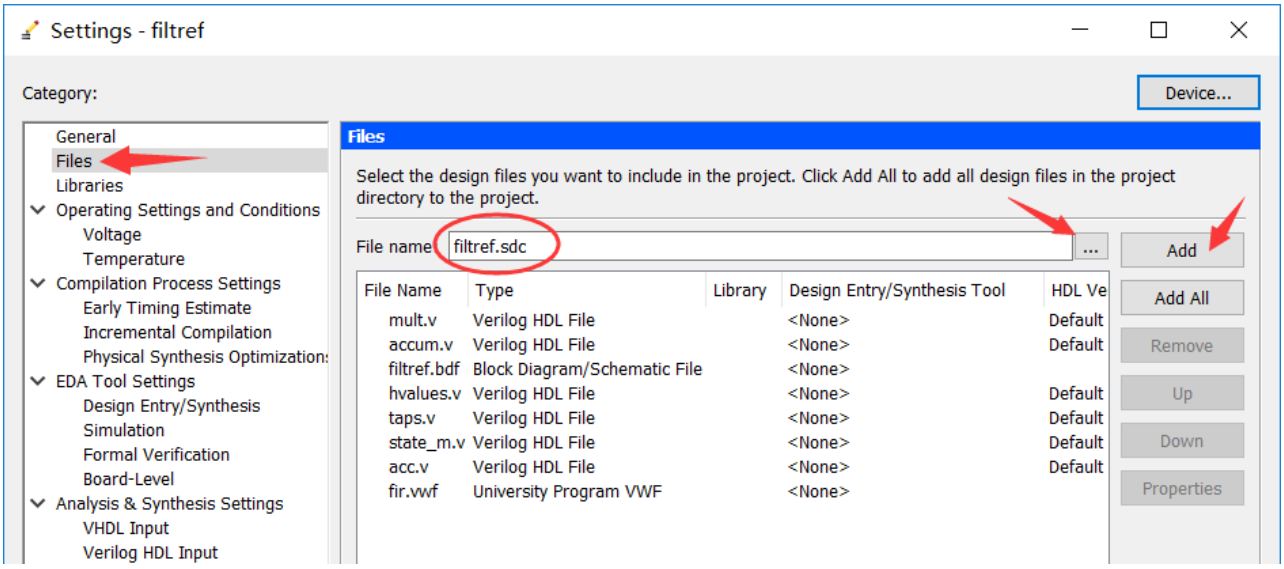


图 1.3.16 添加SDC文件

将 SDC 添加到工程后，在Quartus II软件的Processing 菜单中，点击 Start Compilation，对设计运行一个全编译。

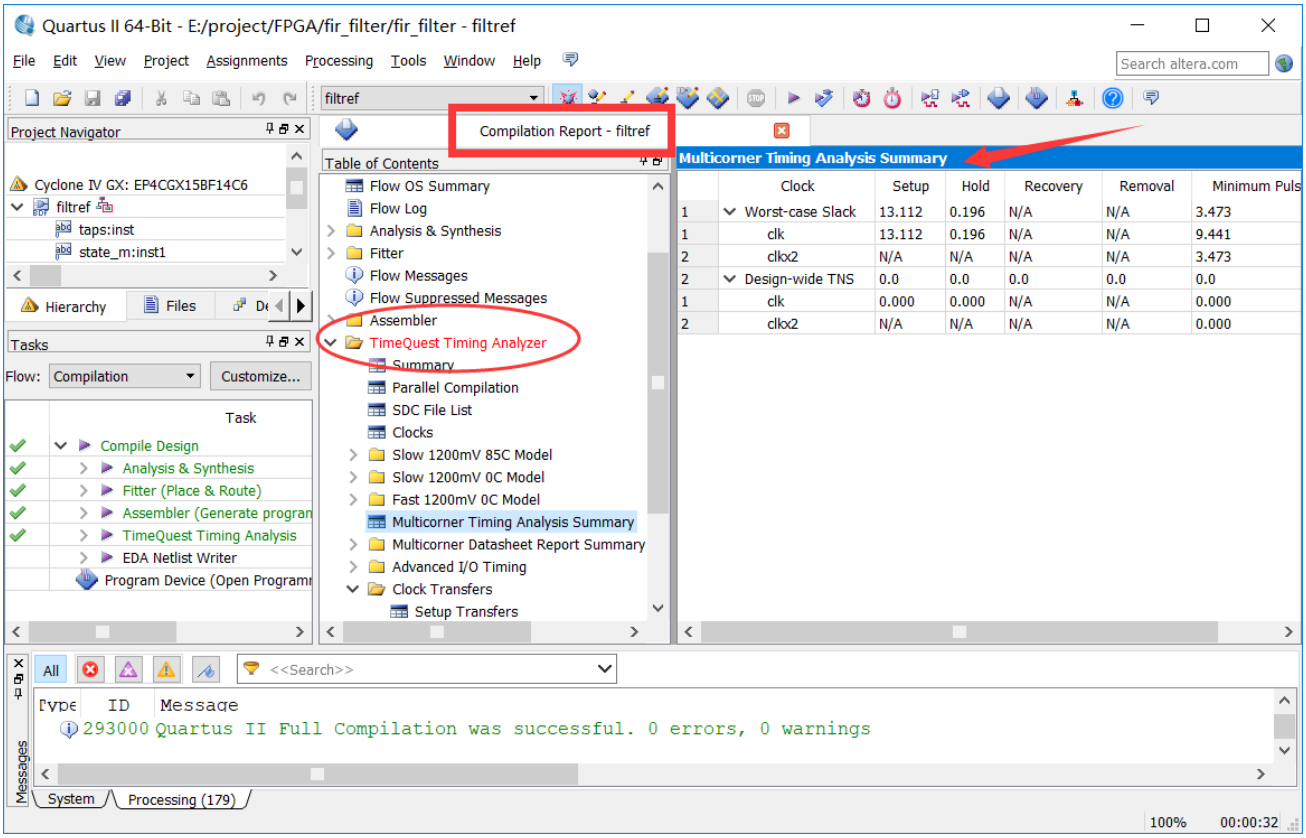


图 1.3.17 全编译后的编译报告

完成编译后，TimeQuest Timing Analyzer 在 Compilation Report 中生成时钟建立和时钟保持的检查总结报告。

完全执行布线布局功能（place-and-route）后，运行TimeQuest Timing Analyzer。在Tasks 面板中，双击所需报告的命令来生成关于最新编译的报告。当双击其中一个报告命令时， Create Timing Netlist、 Read SDC 和 Update Timing Netlist 命令依次在 Tasks 面板中执行，自动生成时序网表，读取 SDC 文件并更新时序网表。

生成**建立总结报告**（Summary Setup）。时钟建立检查确保每个寄存器至寄存器的传输不违反 SDC 指定的时序约束。通过在 Tasks 面板中，双击 Report Setup Summary，生成一个时钟建立总结，对设计中的所有时钟进行检查，来验证没有出现违规。

clkx2时钟没有出现在 Summary (Setup) 报告中，这是因为 clk和 clkx2之间所有的时钟路径已经声明是伪路径。此外，fir_filter 设计不包含任何由 clkx2 来驱动的寄存器到寄存器的路径。

Summary (Setup) 报告中的 Slack 列表明 clk 能满足约束，并有 13.112 ns 的余量。End Point TNS 列是指定时钟域中所有的总负裕量（TNS,Total Negative Slack）的总和。使用这个值来测量指定时钟域中失败路径的总数。

生成 Summary (Setup) 报告后，在 Tasks 面板中，双击 Report Hold Summary，在设计中生成一个时钟保持检查总结。Summary (Hold) 报告表明 clk 时钟节点符合时序约束，并有 0.372 ns 的余量。

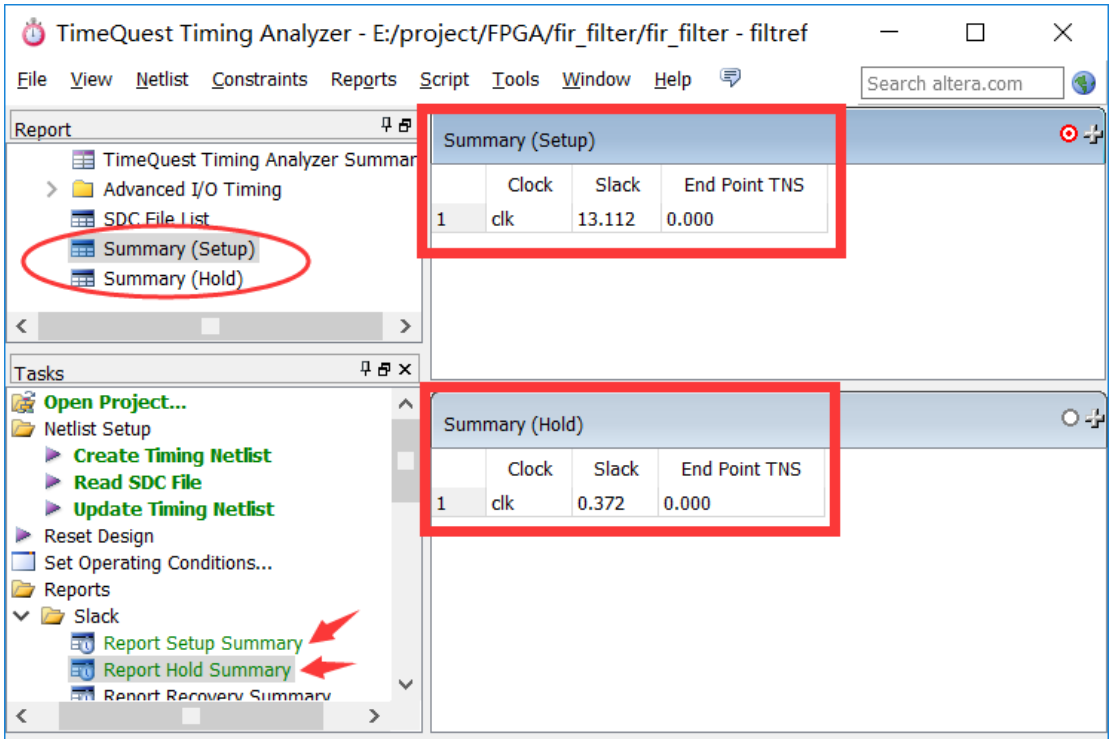


图 1.3.18 建立/保持总结报告

在执行全编译之前，指定所有的时序约束和例外，这样可以确保 Fitter 优化设计中的关键路径。可以使用 **Report Unconstrained Paths** 命令来验证已经约束 fir_filter 设计中的所有路径：在 Tasks 面板中，双击 Report Unconstrained Paths。

如下图所示，Unconstrained Paths 总结报告表明有大量的未约束路径，并详细介绍了这些路径的类型。要充分约束此设计，需要利用由 TimeQuest Timing Analyzer 所提供的整套 SDC 约束。

下图总结报告中的未约束路径集中在Input/Output Ports上，因此我们需要约束所有的输入和输出端口。可以使用 Set Input Delay 和 Set Output Delay 对话框，或者 set_input_delay 和 set_output_delay 约束命令来指定输入和输出延迟值，从而约束端口。

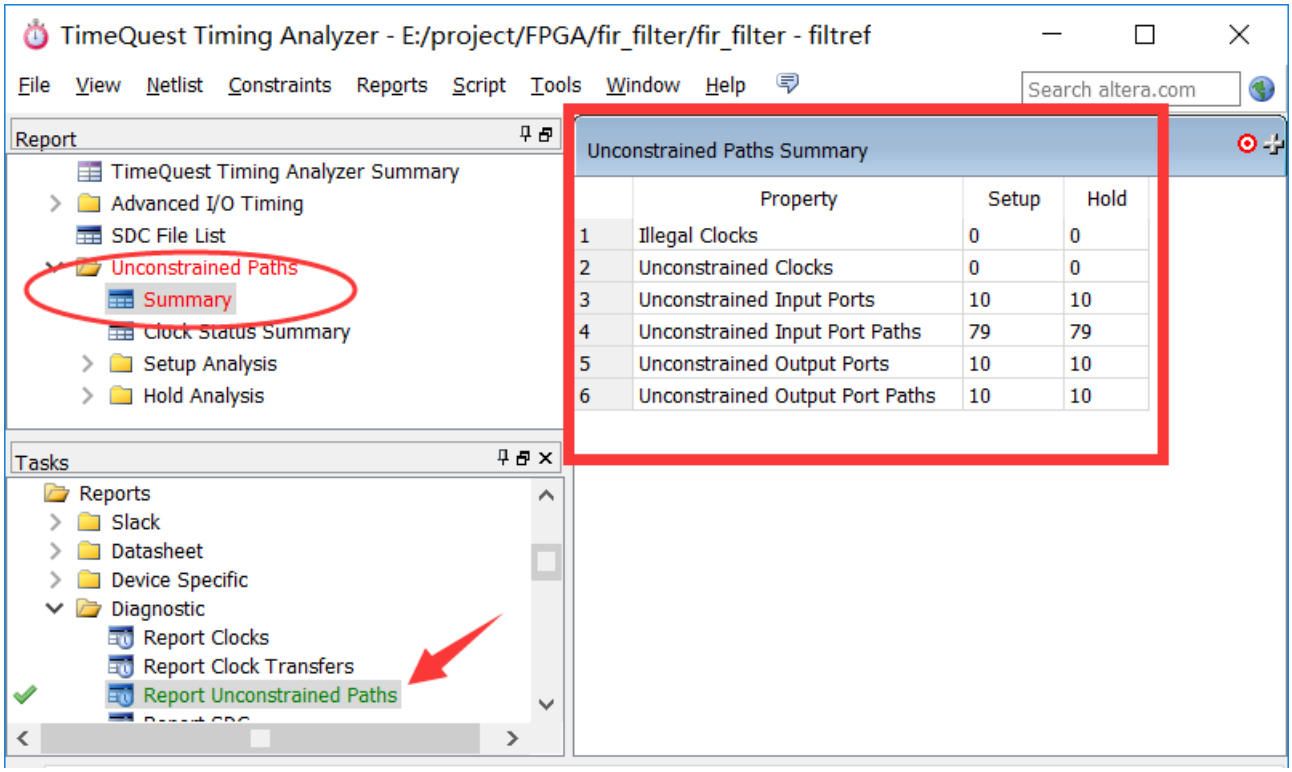


图 1.3.19 未约束路径

在 Constraints 菜单中，点击 Set Input Delay，出现 Set Input Delay 对话框，在对话框中输入以下内容：

```
Clock name: clk

Delay value: 2

Targets: [get_ports {d[0] d[1] d[2] d[3] d[4] d[5] d[6] d[7] newt reset}]
```

在 Constraints 菜单中，点击 Set Output Delay。出现 Set Output Delay 对话框，在对话框中输入以下内容：

```
Clock name: clk
Delay value: 1.5
Targets: [get_ports {yn_out[0] yn_out[1] yn_out[2] yn_out[3] yn_out[4] yn_out[5] \
yn_out[6] yn_out[7] yvalid follow}]
```

Set Output Delay对话框如下图所示：

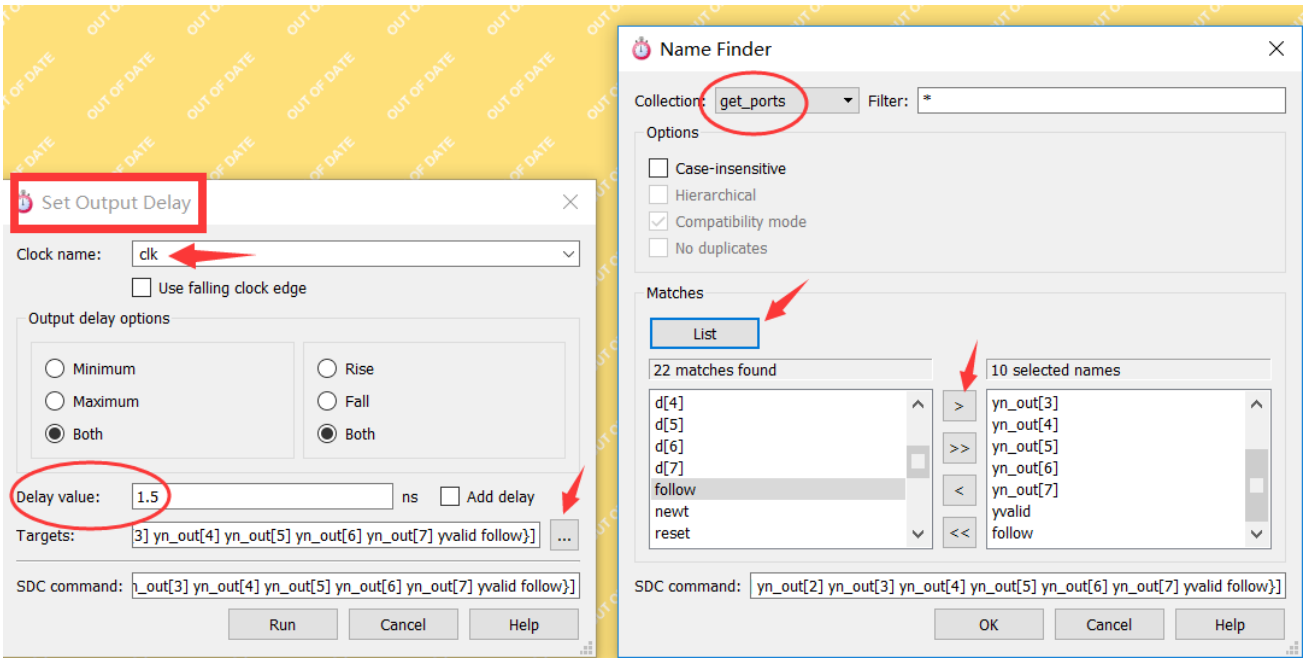


图 1.3.20 设置输出延迟

以上是采用GUI对话框的方式设置约束，也可以在控制台中输入以下等效命令：

要约束输入端口，输入：

```
set_input_delay -clock clk 2 [get_ports {d* newt reset}]
```

要约束输出端口，输入：

```
set_output_delay -clock clk 1.5 [get_ports {yn_out* yvalid follow}]
```

通过Write SDC File将输入输出延迟约束保存到SDC文件中，更新时序网表，重新生成Unconstrained Paths Summary，验证所有设计中的端口都已经加上了约束，如下图所示：

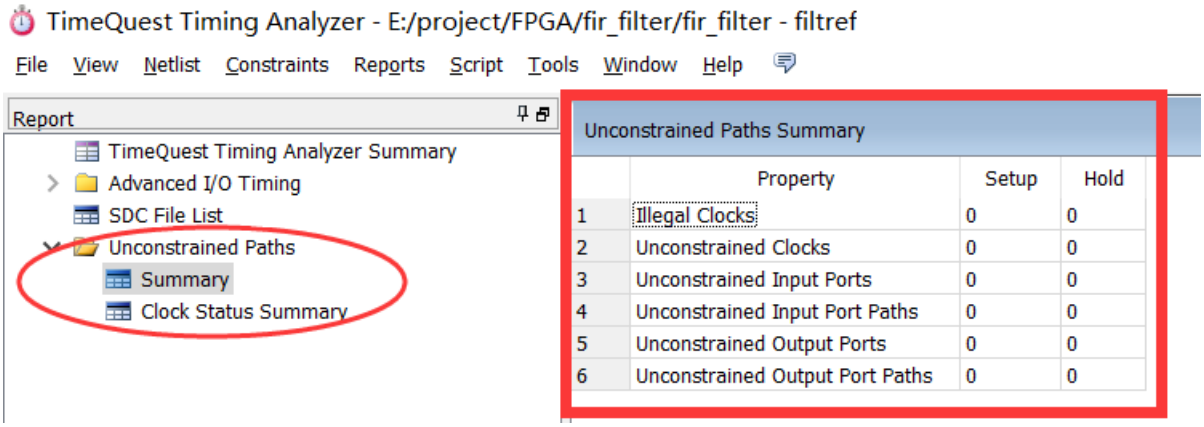


图 1.3.21 未约束的路径数目为零

TimeQuest还可以对设计的时钟或节点生成特定的时序检查报告。在 Tasks 面板中，双击 **Report Timing**，出现 Report Timing对话框。在对话框中的设置如下图所示：

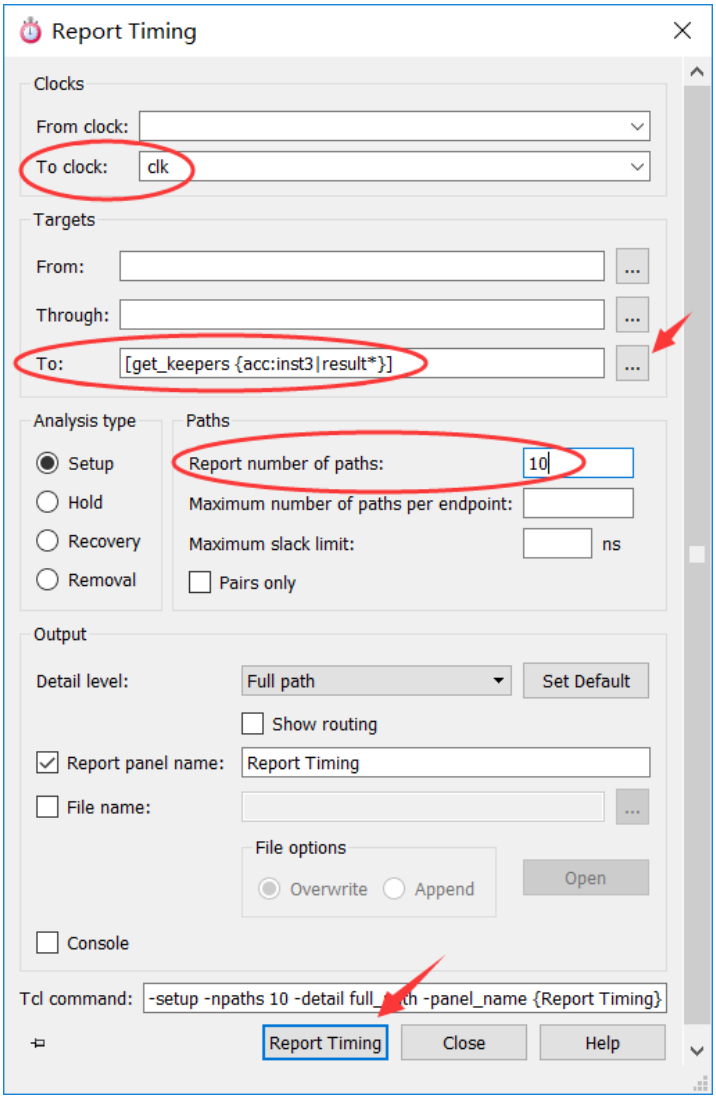


图 1.3.22 Report Timing

对话框设置完成后点击“Report Timing”，会生成一个报告。报告会列出10条最差路径，这些路径均由 clk 驱动目的寄存器，并且目的寄存器为acc:inst3|result，如图1.19所示。

在“Summary of Paths”一栏列出了这些路径的起点和终点，以及启动时钟和锁存时钟等信息，其中第一列“Slack”指示了这些路径是否满足时序要求，以及满足或违反时序要求的程度。当Slack为正时，表示满足时序要求；Slack为负时表明该路径不满足时序要求，那么我们就需要针对该路径进行修复（fix）。

点击Summary of Paths中的任意一个时序路径可以在下方看到它的详细信息，比如在“Data Path”中列出了该路径中信号到达各个节点的延迟信息。时序分析工具通过比较数据到达时间（Data Arrival Times）和数据要求时间（Data Require Times）来验证电路性能，并检测可能出现的时序违规。“Waveform”一列给出了时序路径的波形图，从图中可以更直观的比较各个时序参数。

Report Timing默认分析的是建立时间，在图1.28中同样可以选择分析其他类型（Analysis Tape），比如保持时间、恢复/移除时间等。

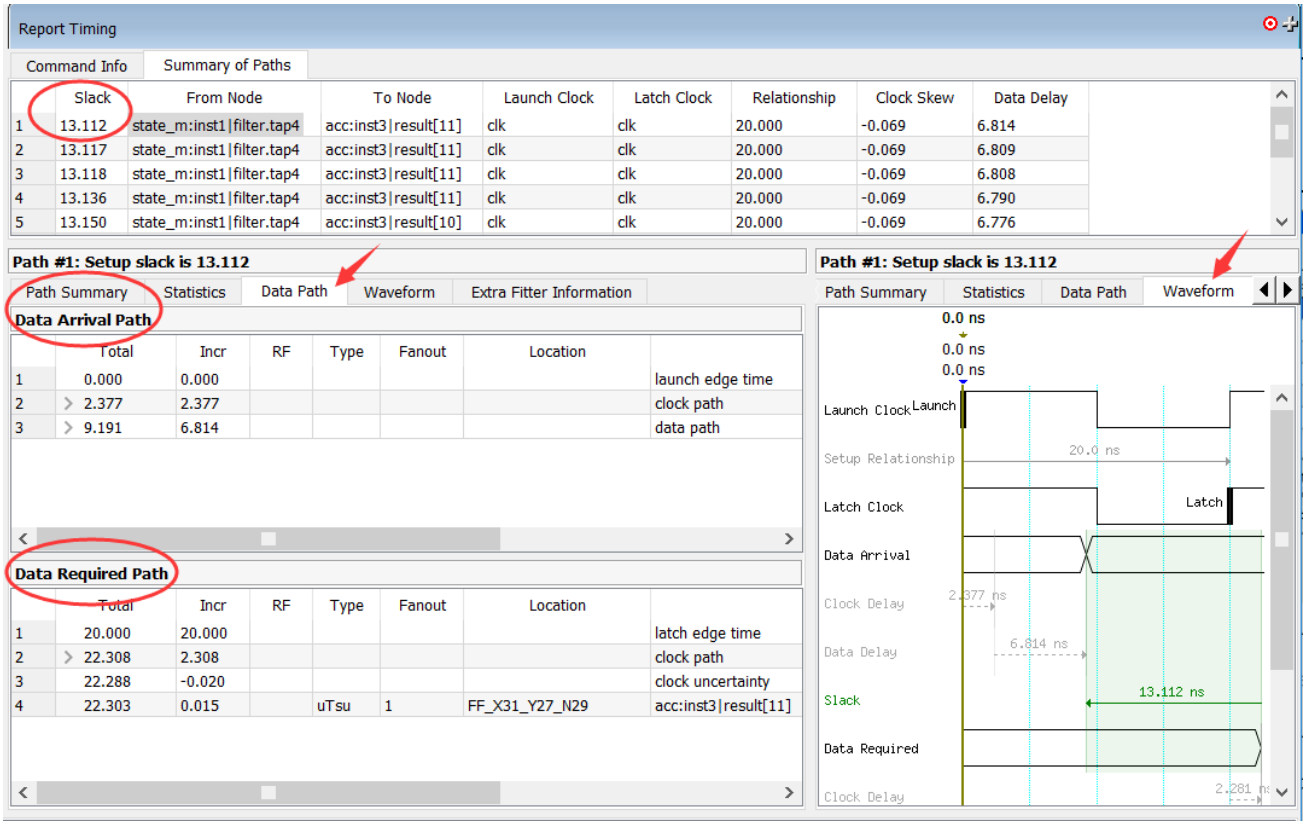


图 1.3.23 时序报告

1.4 常用时序约束

我们在上一节学习了TimeQuest的使用，包括如何创建约束以及如何查看时序报告。其中我们用到了以下约束命令：

```
create_clock
set_input_delay
set_output_delay
set_false_path
```

这些命令包含了对时钟的约束（create_clock），和对I/O的约束（set_input_delay / set_output_delay），以及一些时序例外（set_false_path）。这些都是我们常用的约束，但是并不完整。

约束PLL输出时钟

对时钟的约束分为两种，一种是上面我们用到的被称之为**基准时钟**（base clock）的约束，还有一种是对**生成时钟**（generated clock）的约束。最常见的生成时钟是使用PLL IP核生成的时钟，我们同样需要通过时序约束来告诉工具PLL输出时钟与输入时钟的频率、以及相位上的关系。不过在TimeQuest中对生成时钟的约束很简单，我们只需要一条指令就可以让工具自动识别PLL所输出的生成时钟：

```
derive_pll_clocks
```

我们可以直接打开并编辑SDC文件，在SDC文件中输入该命令，也可以像前面一样在TimeQuest中使用GUI对话框输入此约束。在Constraints菜单栏中选择“Derive PLL Clocks”，会弹出下面的对话框，其中SDC command一栏显示了等效的约束命令，点击Run即可。

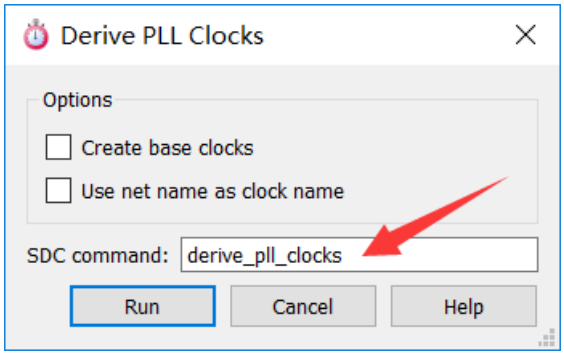


图 1.4.1 约束PLL输出时钟

另外我们还要在SDC文件中添加这样一条指令：

derive_clock_uncertainty

clock uncertainty是指时钟信号的不确定因数。这个命令用来告诉TimeQuest有关时钟的不确定性，这个不确定性是由PLL或者时钟树的抖动（jitter）所造成的。这个命令同样可以直接输入到SDC文件中，或者通过Constraints菜单栏中选择“Derive PLL Clocks”来打开对话框，在对话框中点击Run即可添加约束。

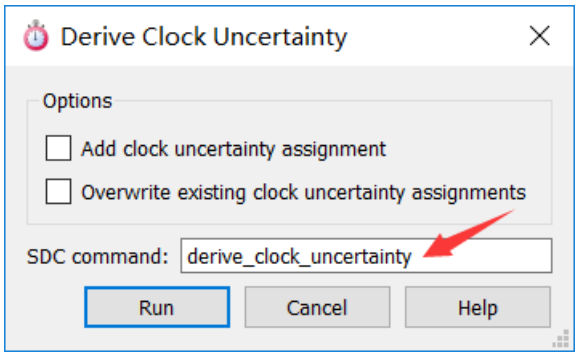


图 1.4.2 约束时钟不确定性

需要注意的是，在TimeQuest中通过GUI方式添加的约束不会自动保存到SDC文件中，需要通过执行Write SDC File命令将其写入SDC文件。

约束组合逻辑I/O接口

对于I/O接口还有一种类型的约束，用来约束输入端口直接通过组合逻辑连接到输出端口的情形，如图1.32所示。我们通过以下指令来约束指定输入到输出端口之间组合逻辑的最大/最小延时：

set_max_delay

set_min_delay

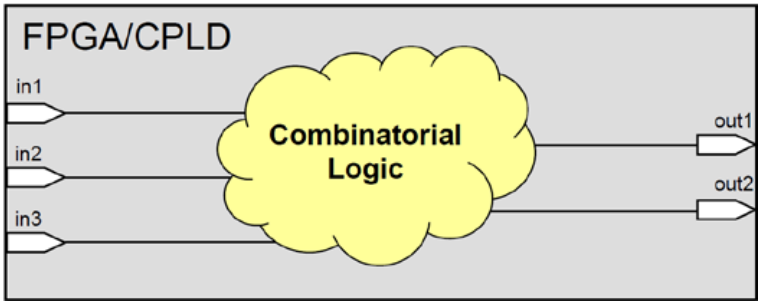


图 1.4.3 组合逻辑连接的I/O接口

图中，所有输入到输出的路径都需约束，对于图1.32来说就需要约束6条路径，在下面我们给出了一个针对上图的SDC约束示例：

```
set_max_delay -from [get_ports in1] -to [get_ports out*] 5.0
set_max_delay -from [get_ports in2] -to [get_ports out*] 7.5
set_max_delay -from [get_ports in3] -to [get_ports out*] 9.0
set_min_delay -from [get_ports in1] -to [get_ports out*] 1.0
set_min_delay -from [get_ports in2] -to [get_ports out*] 2.0
set_min_delay -from [get_ports in3] -to [get_ports out*] 3.0
```

我们同样可以使用GUI对话框来添加此类型的约束，在Constraints菜单栏中选择“Set Maximum Delay”或者“Set Minimum Delay”来打开对话框，如下图所示：

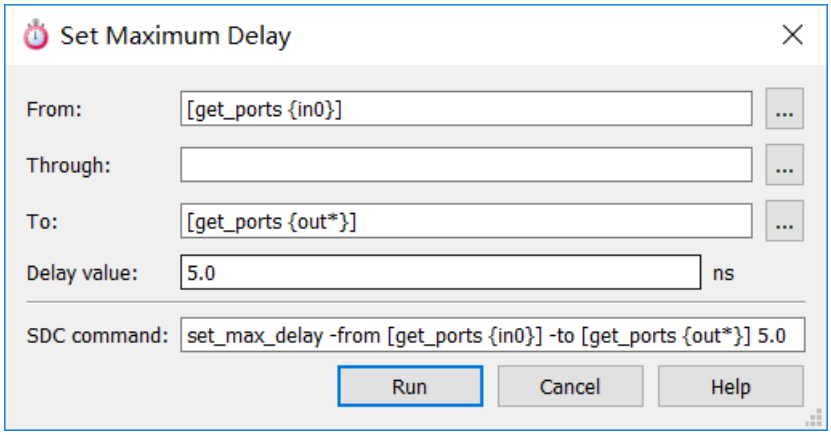


图 1.4.4 设置最大延迟

除了我们上面所介绍的这些约束之外，还有一些其他的类型的约束，比如虚拟时钟（Virtual Clock）约束、多周期路径（Multicycle Paths）约束等。由于这些约束不经常用，在这里不作过多介绍。所有的这些约束都在TimeQuest Timing Analyzer的用户手册中有详细介绍，下面是该用户手册的链接：

https://www.intel.com/content/www/us/en/programmable/documentation/psq1513989797346.html

如果大家想更全面的了解SDC约束或者Quartus II中时序分析器的使用可以参考上面给出的链接。

本文从FPGA的设计流程入手，一步步地讲解如何使用Quartus II软件的时序分析工具TimeQuest Timing Analyzer，来对一个完整的设计进行时序约束。这样更方便初学者在

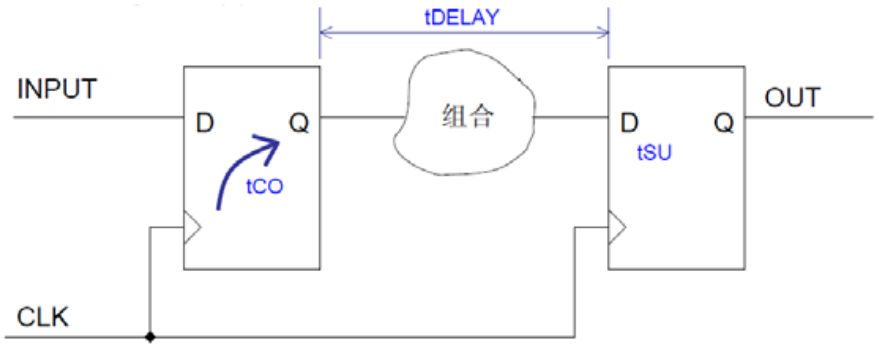
Intel FPGA设计过程中掌握时序约束和分析的方法，但是如果想要更深一步理解时序分析的原理，需要学习时序分析过程中的基本概念。

1.5 时序分析的基本概念

静态时序分析（Static Timing Analysis---STA)的前提是同步逻辑设计：通过路径计算延迟的总和，并比较相对于预定义时钟的延迟。因为异步逻辑时钟没有任何相位和频率关系，所以没有办法进行时序分析。

静态时序分析仅关注时序间的相对关系而不是评估逻辑功能。

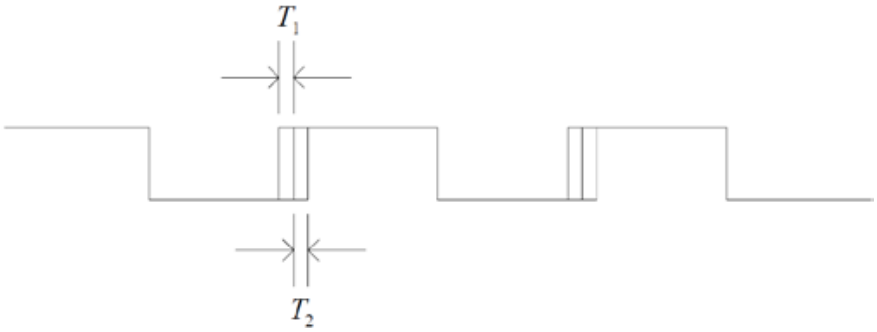
同步逻辑时延模型



如上图所示，T = tCO+tDELAY+tSU。时钟周期大于T，触发器正常工作；时钟周期小于T，不满足建立时间，触发器可能经历亚稳态。即最高时钟频率f = 1/T。分析的对时序象主要是以上图所示的“寄存器对”，即register-to - register为分析目标。

下文有详细说明。以下就模型所涉及到的各个概念进行具体讲解。

时钟抖动与偏斜

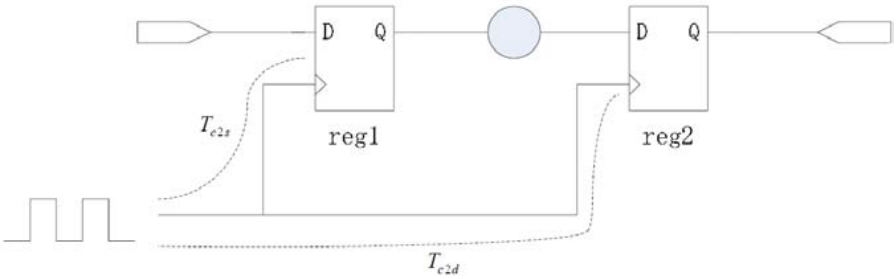


（1）时钟抖动

时钟信号边沿变化的不确定时间称之为时钟抖动。严格说，建立时间应该是Tsu + T1，而保持时间应该是Th + T2。

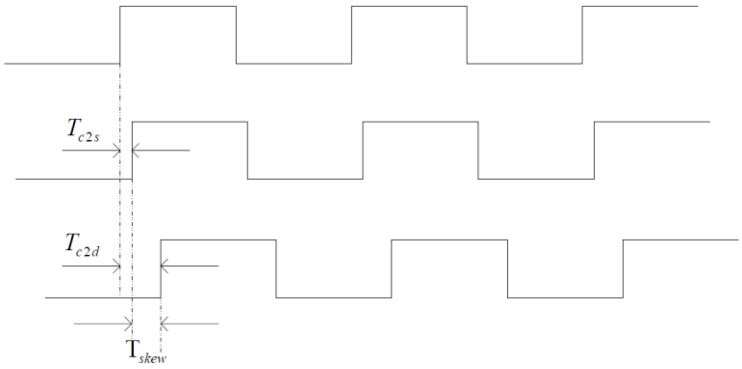
（2）时钟偏斜

时序分析的起点是源寄存器（reg1），终点是目标寄存器（reg2）。时钟和其它信号的传输一样会有延时的。下图中，时钟信号从时钟源传输到源寄存器的延时定义为Tc2s，传输到目标寄存器的延时定义为Tc2d。



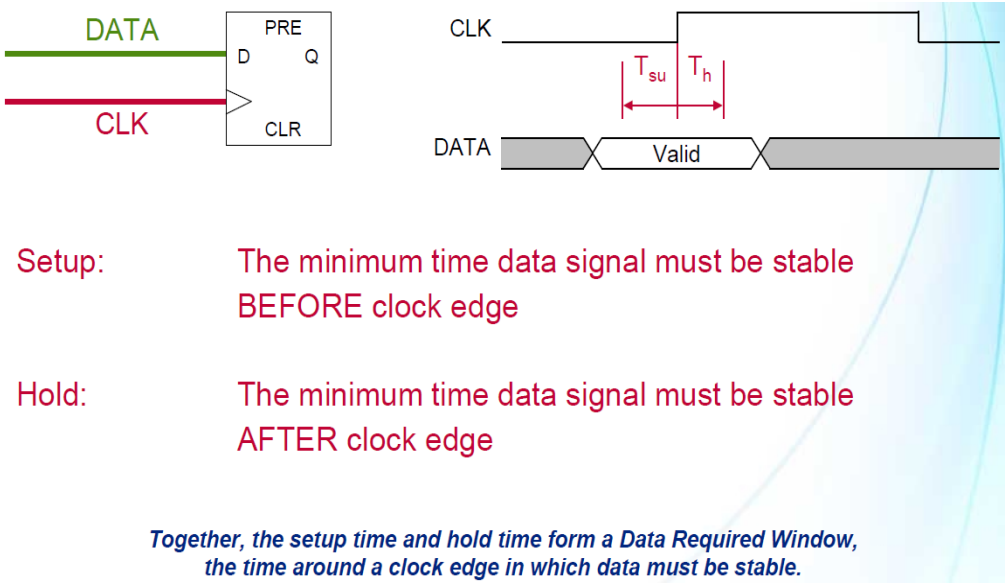
时钟网络延时就定义为Tc2d与Tc2s之差，即Tskew=Tc2d-Tc2s。

如下图所示：



如果考虑时钟偏斜的话，上文所说公式应改为：T+Tskew（Tc2d - Tc2s）=tCO + tDELAY + tSU。

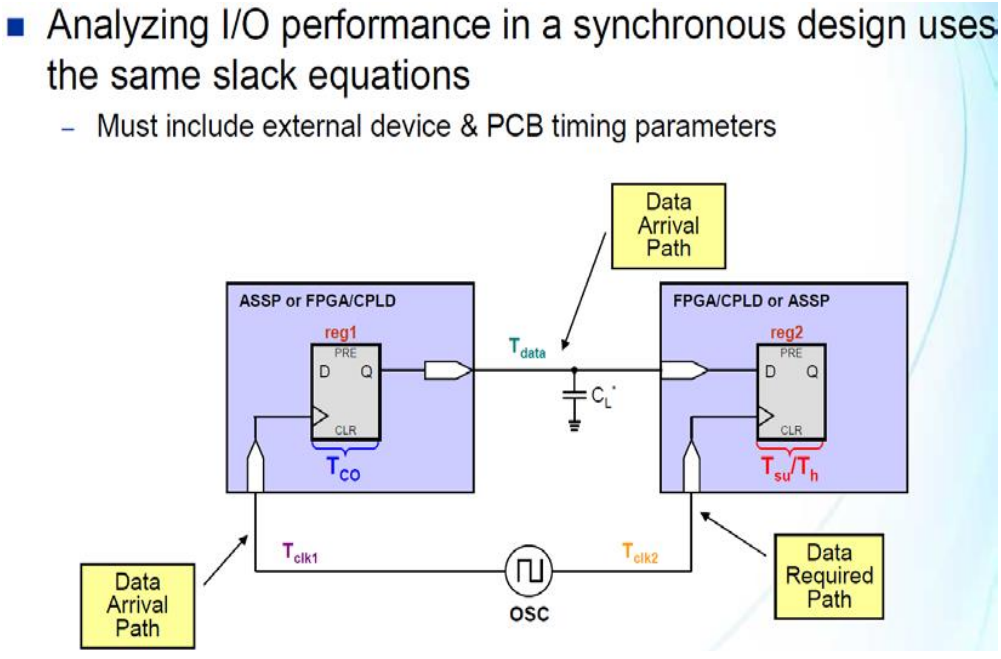
建立时间/保持时间

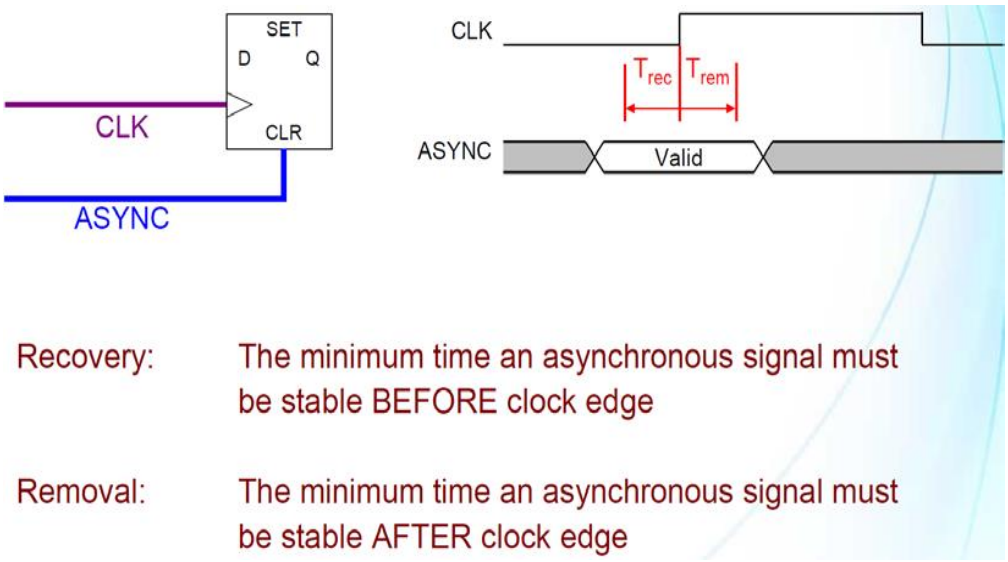


Setup Time：即建立时间，在时钟上升沿之前数据必须稳定的最短时间。若不满足setup time，数据无法进入寄存器：Tsu ＜ Tclk + Tskew － Tco。

Hold Time：即保持时间，在时钟上升沿之后数据必须稳定的最短时间。若不满足hold time，数据无法进入寄存器：Th ＜ Tco。Th 限制了数据传输的速度。如果Tco延时太短导致上一级寄存器锁存的数据侵占了下一级寄存器正在锁存数据的保持时间，那么下一级寄存器就无法有效的锁存数据，系统时序也就无法达到要求。

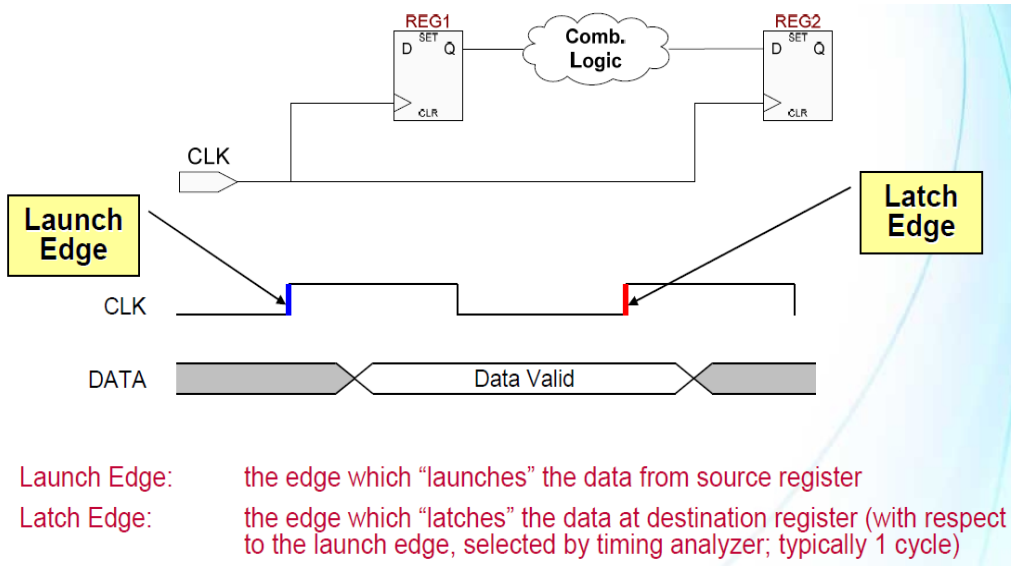
恢复时间/移除时间





此处大家先记住复位信号相对时钟也有“建立”时间和“保持”时间即可，先不用深究。

Launch Edge & Latch Edge



Launch Edge：启动沿，前级寄存器发送数据对应的时钟沿，是时序分析的起点。

Latch Edge：锁存沿，后级寄存器捕获数据对应的时钟沿，是时序分析的终点。相对于launch edge通常为一个时钟周期，但不绝对，如多周期约束。

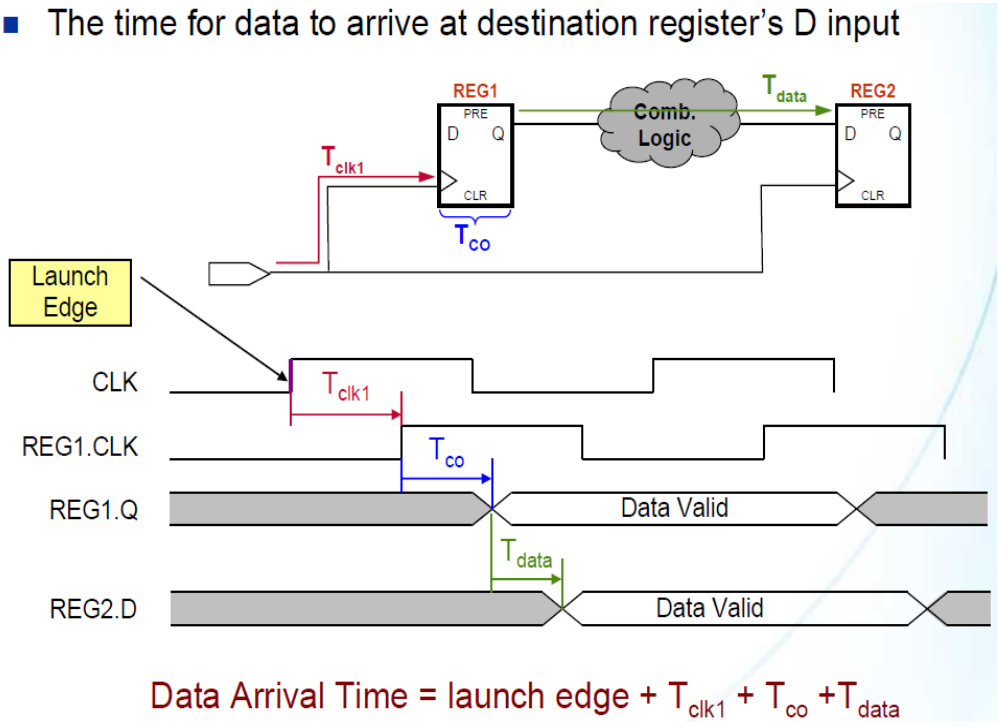
“信号跳变抵达窗口”：对launch寄存器来说，从previous时钟对应的Hold Time开始，到current时钟对应的Setup Time结束。

“信号电平采样窗口”： 对latch寄存器来说，从current时钟对应的Setup Time 开始，到current时钟对应的Hold Time结束。

launch寄存器REG1必须保证驱动的信号跳变到达latch寄存器REG2的时刻处于“信号跳变抵达窗口”内，才能保证不破坏latch寄存器的“信号电平采样窗口”。

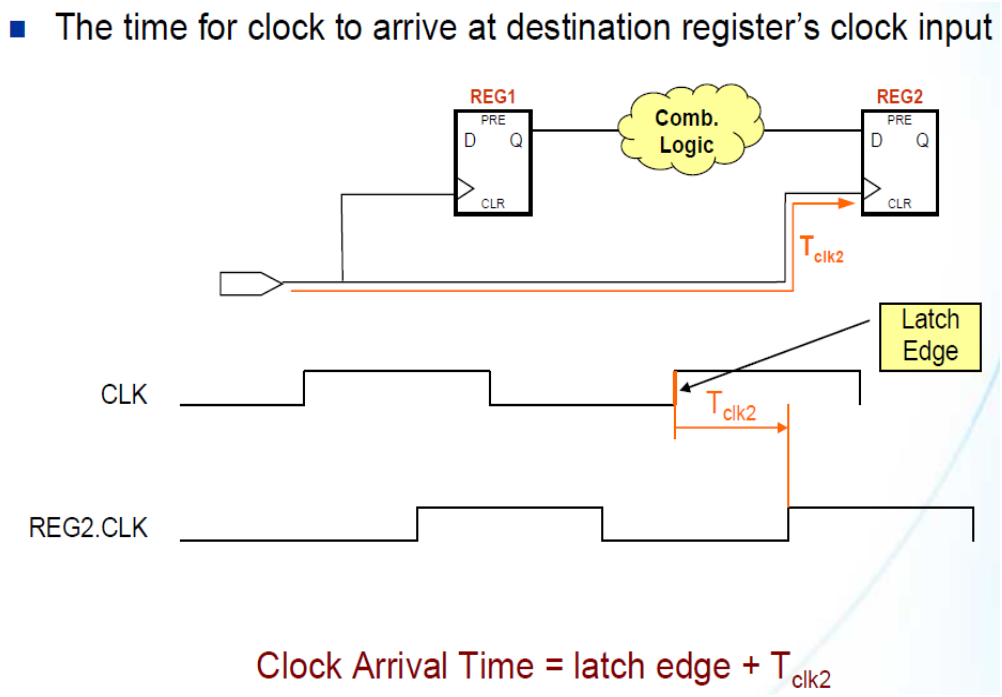
Data & Clock Time

（1）Data Arrival Time

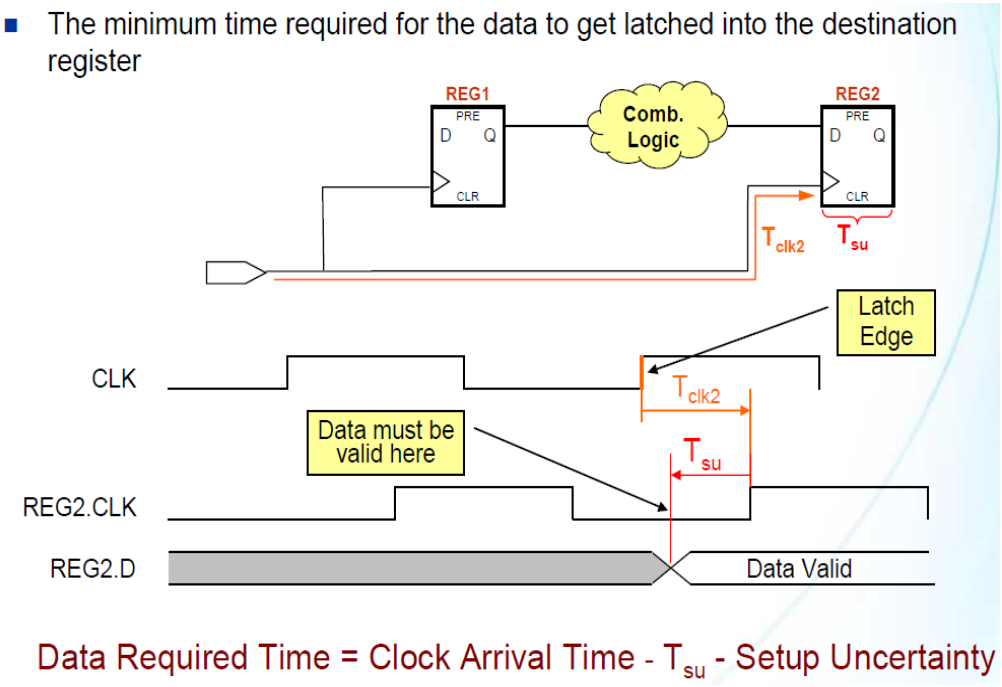


Tclk1是时钟从时钟源点到达REG1时钟端的延迟。

（2） Clock Arrival Time

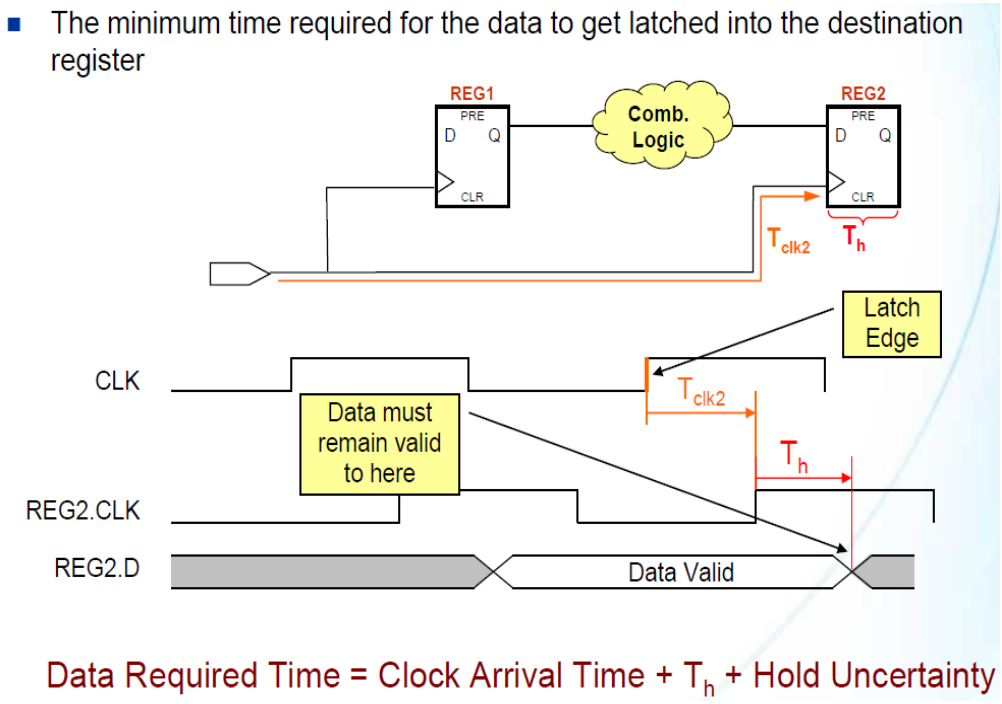


（3）Data Required Time (Setup)



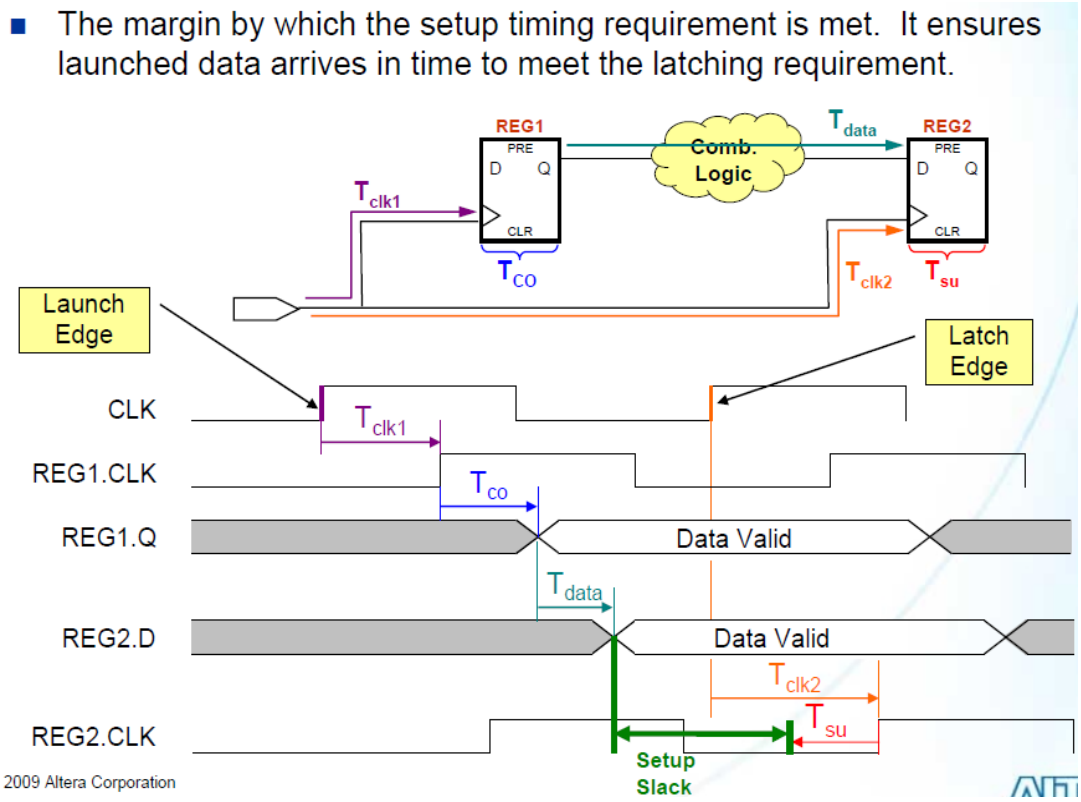
上图公式减去了setup uncertainty，即考虑最坏的情况仍需满足data required time (setup)。

（4）Data Required Time (Hold)



同理，上图公式加上了hold uncertainty，即考虑最坏的情况仍要满足的data required time (hold)。

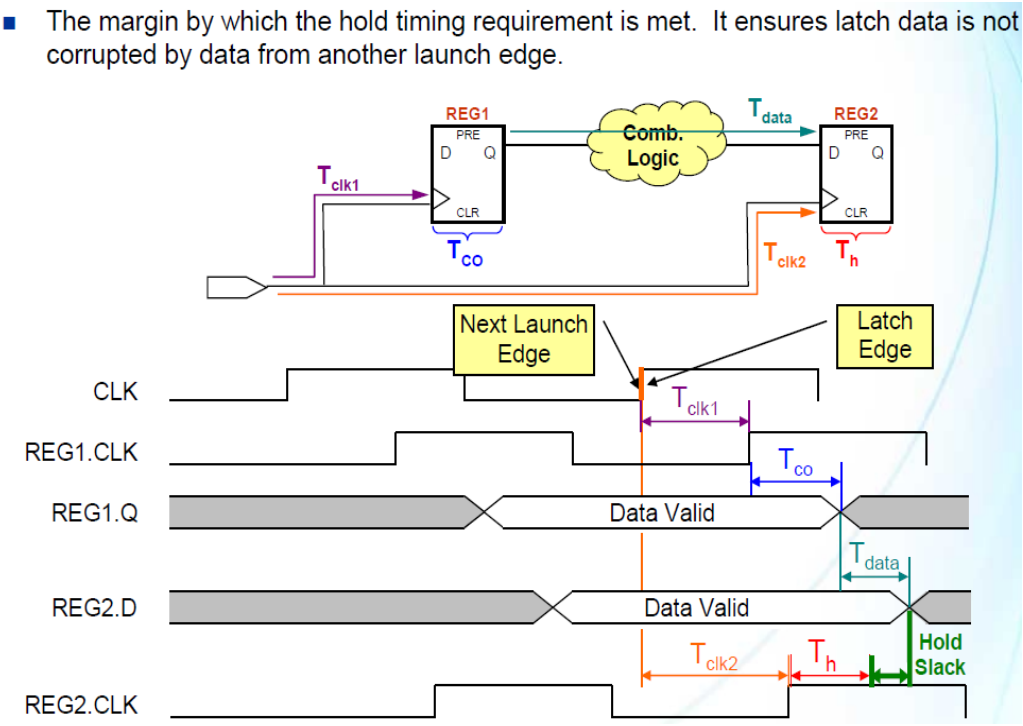
(5) Setup Slack



由上图可知，Setup Slack = Data Required Time - Data Arrival Time。

若Setup Slack为正，表示Data Required Time在Data Arrival Time之后，满足Setup Time；反之若Setup Slack为负，则表示Data Arrival Time在Data Required Time之后，无法满足Setup Time。

（6）Hold Slack



由上图可知，Hold Slack = Data Arrival Time - Data Required Time。若Hold Slack为正，表示Data Arrival Time在Data Required Time之后，满足Hold Time；反之若Hold Slack为负，则表示Data Required Time在Data Arrival Time之后，无法满足Hold Time。

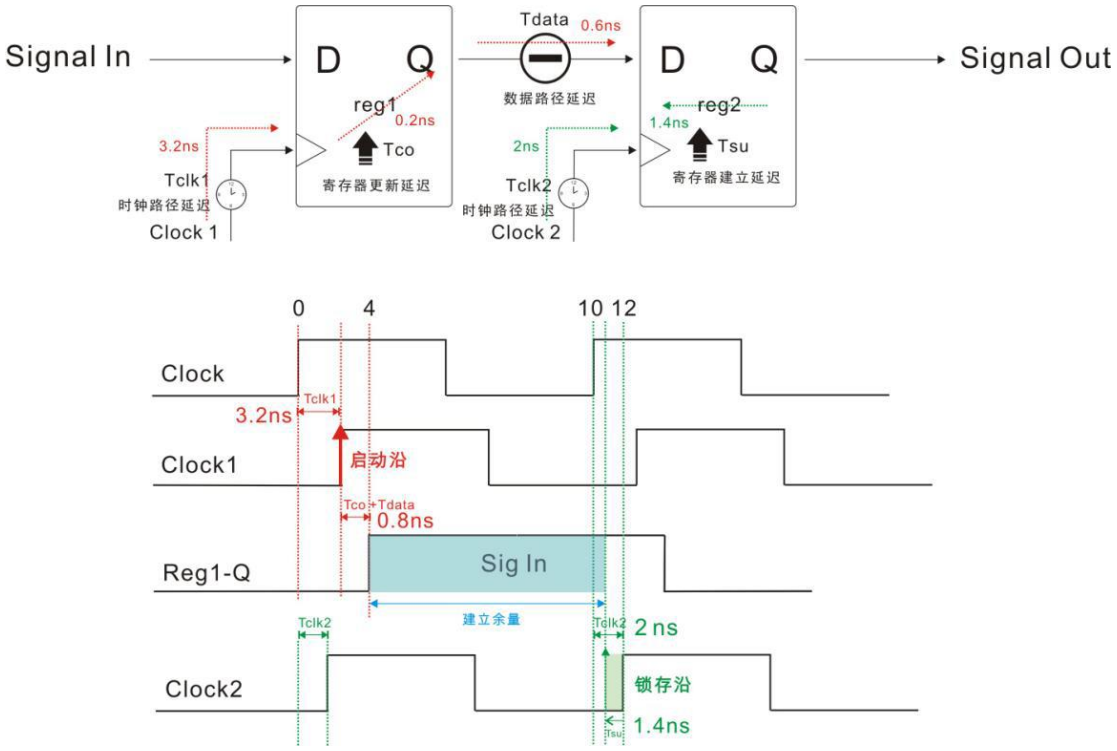
时序分析基本公式

TimeQuest 静态时序分析的对象包括：寄存器和寄存器之间的路径、I/O 之间、I/O 和寄存器之间的路径、异步复位和寄存器之间的路径。TimeQuest 根据Data Arrival Time 和Data Required Time 计算出时序余量（Slack）。当时序余量为负值时，就发生了时序违规（Timing Violation）。需要特别指出的一点是：由于时序分析是针对时钟驱动的电路进行的，所以分析的对象一定是“寄存器-寄存器”对。在分析涉及到I/O 的时序关系对时，看似缺少一个寄存器分析对象，构不成“寄存器-寄存器” 对，其实是穿过FPGA 的I/O 引脚，在FPGA 外部虚拟了一个寄存器作为分析对象。

建立时间（Setup Time）检查

建立余量英文名字是setup slack，它是TimeQuest 模型中重要的结果之一，正值的建立余量说明两个寄存器有合格的建立关系。

通过下图来说明如何获得建立余量。



上图是物理的建立关系过程，寄存器 1 和寄存器2 也使用拥有相同属性的时钟源。

首先要获得 Clock1 的启动沿受延迟的影响，而TimeQuest 模型用了一条公式评估启动沿受Tclk1 + Tco + Tdata 的影响：

数据到达时间 Time Arrival Time = 启动沿时间+ Tclk1 + Tco + Tdata

= 0ns + 3.2ns + 0.2ns + 0.6ns

= 4ns

下面我们要获得 Clock2 的锁存沿受延迟的影响。同样TimeQuest 模型也使用了一条公式评估锁存沿受 Tclk2 + Tsu 的影响：

数据锁存（读取）时间Time Required Time = 锁存沿时间+ Tclk2 - Tsu= 10ns + 2ns - 1.4ns

= 10.6ns

最后我们要用“数据抵达时间”和“数据锁存时间”取得建立余量：

建立余量 = 数据锁存时间 - 数据抵达(获取|读取)时间

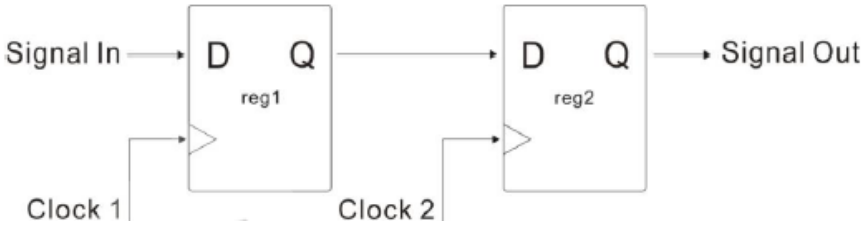
```
Setup Slack = Time Required Time - Time Arrival Time

= 10.6ns - 4ns

= 6.6ns
```

其中建立余量 6.6ns 是在上图的“蓝色框图的部分”。在这里，建立余量是“正值”，亦即“reg1 和reg2 之间的建立关系是合格”。

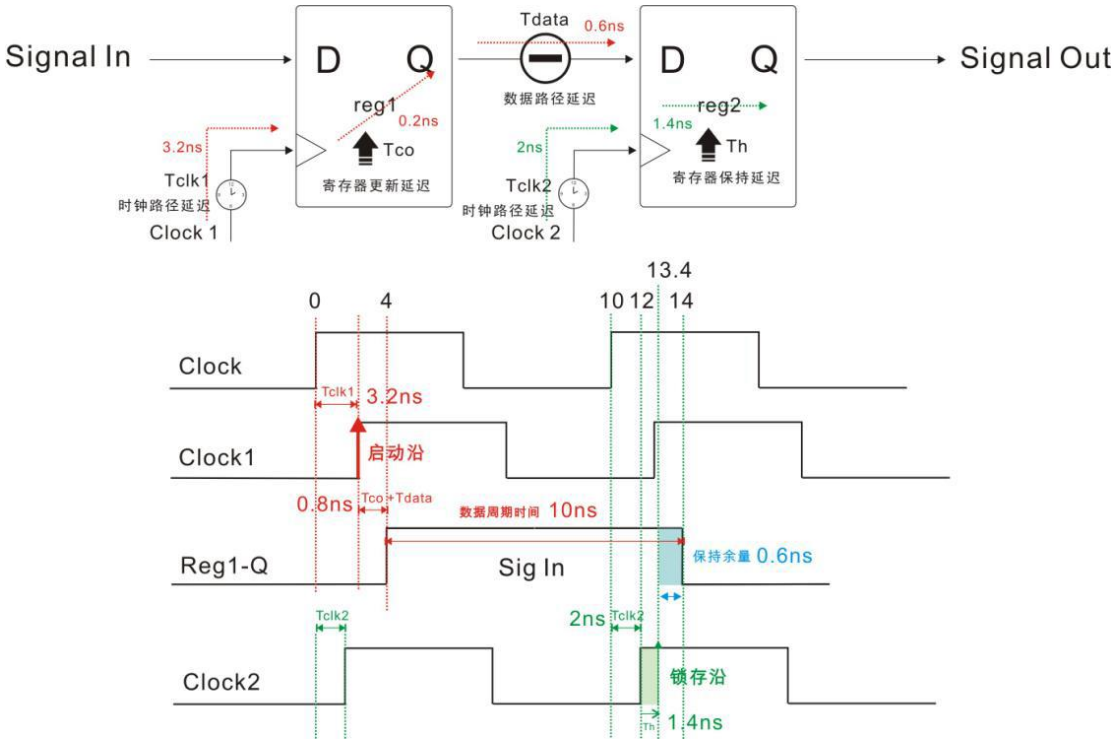
保持时间（Hold Time）检查

“保持时间”在寄存器的概念中是表示：寄存器在读取某个数据以后，需用一段最小时间来“确保数据锁存”的稳定。

保持余量在英文中Hold Slack，保持关系Hold Relationship 是求出两个寄存器（节点）之间的保持余量的过程。保持关系是指寄存器1 在还没有更新输出之前，寄存器2 用最快速的速度读取从寄存器1 发送过来的数据。



通过下图来说明如何获得保持余量。



数据保持时间 Data Hold Time = 启动沿+ Tclk1 + Tco+ Tdata + 数据周期时间

= 0ns + 3.2ns + 0.2ns + 0.6ns + 10ns

= 14ns

数据锁存（读取）时间Data Required Time = 锁存沿+ Tclk2 + Th

= 10ns + 2ns + 1.4ns

= 13.4ns

保持余量 Hold Slack = 数据保持时间- 数据锁存（获取|读取）时间

= 14ns - 13.4ns

= 0.6ns

最后取得的保持余量是 0.6ns，这也意味着保持余量是正值，所以寄存器 1和寄存器2之间的保持关系是合格的。

多周期路径（Multicycle Paths）检查

在同步逻辑设计中，通常都是按照单周期关系考虑数据路径的。但是往往存在这样的情况：一些数据不需要在下一个时钟周期就稳定下来，可能在数据发送后几个时钟周期之后才起作用；一些数据经过的路径太复杂，延时太大，不可能在下一个时钟周期稳定下来，必须要在数据发送后数个时钟周期之后才能被采用。

不设置多周期路径约束的后果有两种：一是按照单周期路径检查的结果，虚报时序违规；二是导致布局布线工具按照单周期路径的方式执行，虽然满足了时序规范，但是过分优化了本应该多个周期完成的操作，造成过约束。过约束会侵占本应该让位于其他逻辑的布局布线资源，有可能造成其他关键路径的时序违规或时序余量变小。

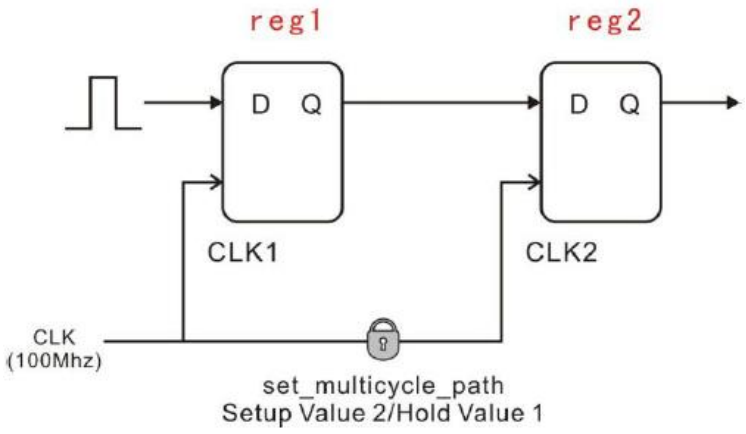
在多周期路径的建立时间（Setup Time）检查中，Timequest会按照用户指定的周期数延长Data Required Time，放松对相应数据路径的时序约束，从而得到正确的时序余量计算结果；

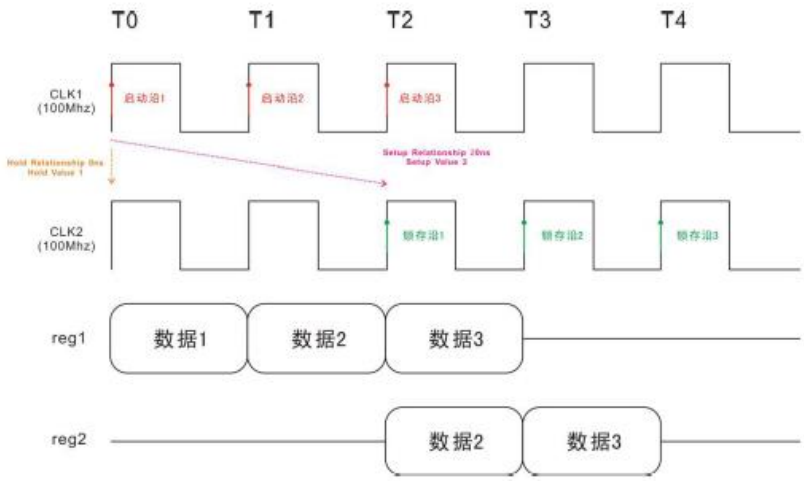
在保持时间（Hold Time）检查中，Timequest也会相应地延长Data Required Time，不再按照单周期路径的分析方式执行（不再采用launch edge最近的时钟沿，而是采用latch edge最近的时钟沿），这就需要用户指定保持时间对应的多周期个数。

Timequest缺省的Hold Time检查公式是需要用户修改的——针对Setup Time多周期路径的设置也会影响到Hold Time的检查。

究其原因，多周期路径是为了解决信号传播太慢的问题，慢到一个周期都不够，所以要把Setup Time的检查往后推几个周期——扩大Setup Time检查的时间窗口。而Hold Time检查信号是否传播得太快，如果把检查时刻往后推，就缩小了Hold Time检查的时间窗口。

一个易于理解的例子：





Altera 器件时序模型

TimeQuest 作为数学工具必须根据各种CASE 作出分析，CASE 简称点就是环境或者状况，然而TimeQuest 却有两种CASE，其一是Worst Case，其二是Best Case ，前者对应slow model，后者对应fast model。

Quartus提供两种PVT(芯片工艺、电压、温度)条件下的器件时序模型（注：65nm及以下的高级器件还有其他模型）：

Case	模型	环境	温度	性能限制
Worst Case	slow	极端	低温/高温	高
Best Case	fast	最佳	室温	低

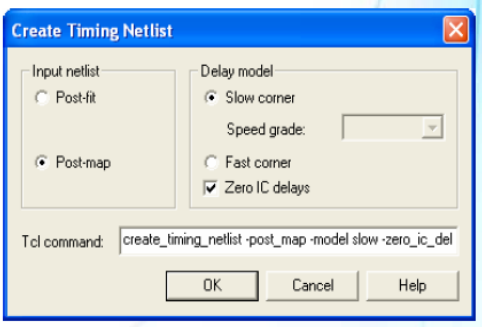
（1） Slow Corner模型通过假设最大的环境温度（operating temperature）和VCCmin来模拟一条信号路径可能的最慢的情况。

（2） Fast Corner模型通过假设最小的环境温度（operating temperature）和VCCmin来模拟一条信号路径可能的最快的情况。这两个模型的意义在于，我们可以通过slow corner模型来保证建立时间的时序，

通过fast corner来保证保持时间的时序（对于源同步来说必须使用）。由于一般情况下设计以建立时间违规为主，所以Timequest默认使用slow corner。

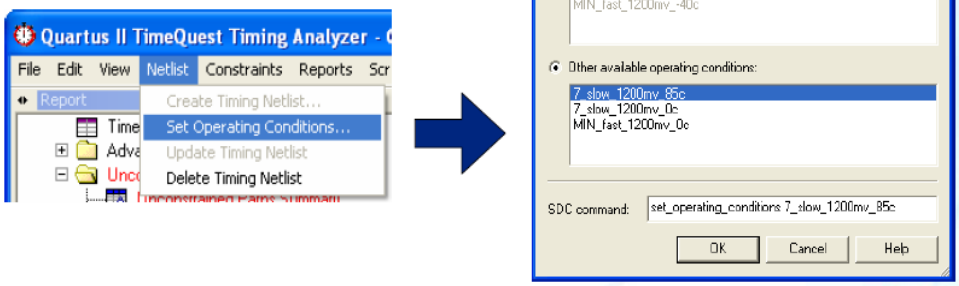
Generating Fast/Slow Netlist

- Specify one of the default timing models to be used when creating your netlist
- Default is the slow timing netlist
- To specify fast timing netlist
  - Use `-fast_model` option with `create_timing_netlist` command
  - Choose **Fast corner** in GUI when executing **Create Timing Netlist** from **Netlist** menu
  - CANNOT select fast corner from Tasks Pane



Specifying Operating Conditions

- Perform timing analysis for different delay models without recreating the existing timing netlist
- Takes precedence over already generated netlist
- Required for selecting slow, min. temp. model and other models (industrial, military, etc.) depending on device
- Use `get_available_operating_conditions` to see available conditions for target device



基本单元与 paths

Timequest需要读入布局布线后的网表才能进行时序分析。读入的网表是由以下一系列的基本单元构成的：

基本单位	说明
Cell	SDC 网表中最普遍的个体。FPGA 中可以找到的寄存器呀，片上 RAM 呀，PLL 资源呀，硬件乘法器等逻辑资源。
Pin	Cell 进出口。
Net	Pins 之间的连线。
Port	顶层模块的顶层输入输出口。

(1) Cells ：Altera器件中的基本结构单元(例如，查找表、寄存器、IO单元、PLL 、存储器块等) 。LE可以看作是Cell 。

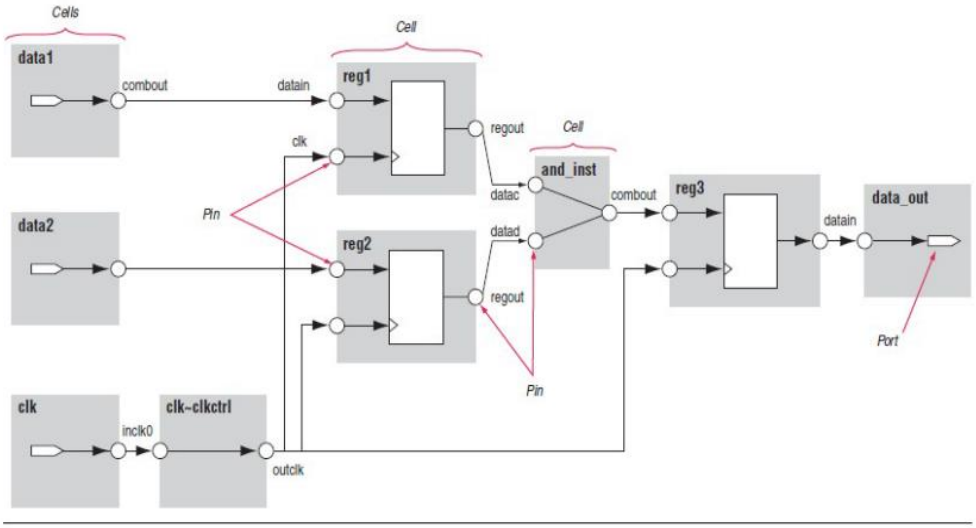
(2) Pins：Cell 的输入输出端口。可以认为是LE的输入输出端口。注意：这里的Pins不包括器件的输入输出引脚，代之以输入引脚对应LE的输出端口和输出引脚对应LE的输入端口。

(3) Nets：同一个Cell中，从输入pin 到输出pin 经过的逻辑。注意，网表中连接两个相邻Cell 的连线不被看作Net，而被看作同一个点，等价于Cell 的pin 。虽然连接两个相邻Cell 的连线不被看作Net，但这个连线还是有其物理意义的，即等价于Altera器件中的一段布线逻辑，会引入一定的延迟。

(4) Ports ：顶层逻辑的输入输出端口。对应已经分配的器件引脚。

下面这幅图给出了一个时序网表的示例，展示了基本单元中的一部分。

Figure 7–4. The TimeQuest Timing Analyzer Timing Netlist



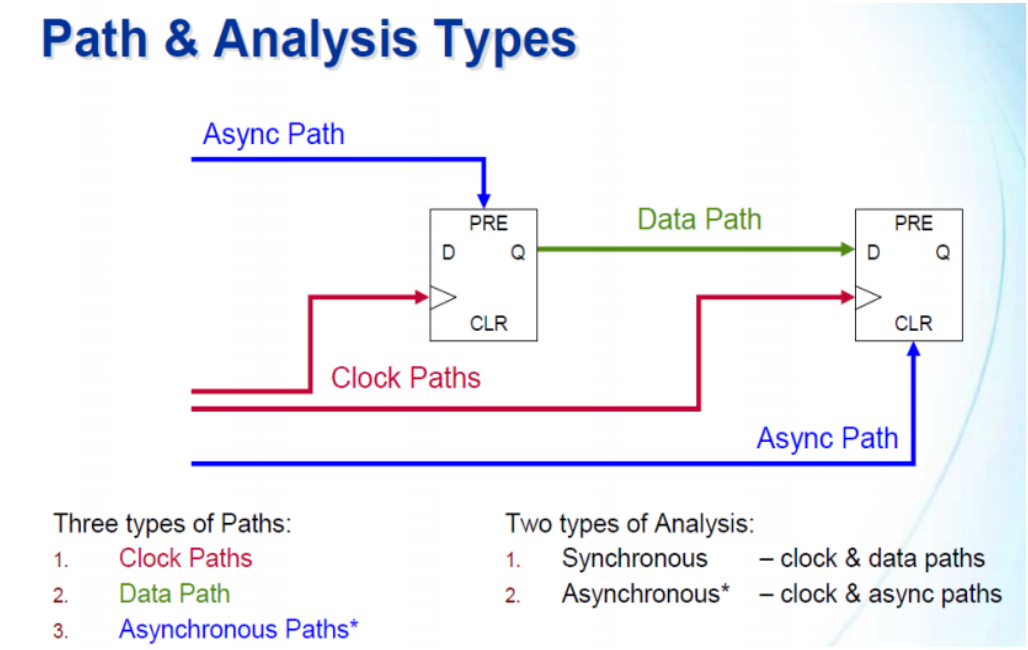
Path 通常分为三类：

(1) Clock paths：从Clock Port或内部生成的clock pin 到寄存器Cell 的时钟输入Pin。

(2) Data paths：从输入Port 到寄存器Cell 的数据输入pin ，或从寄存器Cell 的数据输出pin 到另一个寄存器Cell 的数据输入pin 。

(3) Asynchronous paths：从输入Port 到寄存器Cell 的异步输入pin ，或从寄存器Cell 的数据输出pin 到另一个寄存器Cell 的异步输入pin 。

三种path 如下图所示。



FPGA 时序约束的几种方法

对自己的设计的实现方式越了解，对自己的设计的时序要求越了解，对目标器件的资源分布和结构越了解，对EDA工具执行约束的效果越了解，那么对设计的时序约束目标就会越清晰，相应地，设计的时序收敛过程就会更可控。 按从易到难的顺序提供如下几种进行时序约束的方法。

（0）核心频率约束

这是最基本的，所以标号为0，因为Fmax 是针对“最差劲节点”... 换言之，如果该“最差劲节点”得到好成绩，那些不是最差劲的节点的成绩当然比“最差劲节点”好。 best case 的Fmax 评估比起worst case 有更好的表现（也更接近实体的Fmax评估）。

（1）核心频率约束+ 时序例外约束 时序例外约束包括FalsePath 、MulticyclePath、MaxDelay、MinDelay。但这不是完整的时序约束。如果仅有这些，说明思路还局限在FPGA芯

片内部。

（2）核心频率约束+ 时序例外约束+I/O 约束IO约束包括引脚分配位置、空闲引脚驱动方式、外部走线延时(InputDelay 、OutputDelay)、上下拉电阻、驱动电流强度等。加入I/O约束后的时序约束，才是完整的时序约束。FPGA作为PCB 上的一个器件，是整个PCB 系统时序收敛的一部分。FPGA作为PCB 设计的一部分，是需要PCB 设计工程师像对待其他器件一样，阅读并分析其I/O时序图的。FPGA不同于一般器件之处在于，其I/O Timing在设计后期可以于一定范围内调整。虽然如此，最好还是在PCB 设计前期给与充分的考虑并归入设计文档。正因为FPGA的I/O Timing 会在设计期间发生变化，所以准确地对其进行约束是保证设计稳定可控的重要因素。许多在FPGA重新编译后，FPGA对外部器件的操作出现不稳定的问题都有可能是由此引起的。

好的时序是设计出来的，不是约束出来的 。好的约束必须以好的设计为前提。没有好的设计，在约束上下再大的功夫也是没有意义的。不过，通过正确的约束也可以检查设计的优劣，通过时序分析报告可以检查出设计上时序考虑不周的地方，从而加以修改。通过几次“ 分析—修改—分析” 的迭代也可以达到完善设计的目标。应该说，设计是约束的根本，约束是设计的保证，二者是相辅相成的关系。

时序分析官方资料

网络上介绍时序约束和时序分析的文章和教程繁如星海，但是内容良莠不齐。在这里我们推荐大家通过Intel官方提供的教程学习，虽然文档大多数是英文版的，但是内容全面，描述精练准确，是学习时序约束和时序分析最完美的材料。在这里我们给出Intel时序分析器资源中心的链接：

https://www.intel.com/content/www/us/en/programmable/support/support-resources/design-examples/design-software/timinganalyzer/sof-qts-timinganalyzer.html

上面的链接中还有Timing Analyzer中文版的视频培训教程，共四讲。在每讲视频的最后都详细地介绍了有关时序分析各方面的知识应该去哪个文档中查找。为方便大家学习，专门把该视频的链接列在下面：

https://www.intel.com/content/www/us/en/programmable/support/training/course/ocdsw1115.html