目录

[Clocks and Generated Clocks (时钟和衍生时钟) 2](#_Toc415063348)

[Basic Non-50/50 Duty Cycle Clock 2](#_Toc415063349)

[Offset Clocks(时钟偏移) 2](#_Toc415063350)

[Basic Clock Divider Using -divide\_by 3](#_Toc415063351)

[Toggle Register Generated Clock 5](#_Toc415063352)

[PLL Clocks 锁相环时钟 5](#_Toc415063353)

[Method 1 – Create Base Clocks and PLL Output Clocks Automatically 自动创建基准时钟和PLL输出时钟 6](#_Toc415063354)

[Method 2 – Create Base Clocks Manually and PLL Output Clocks Automatically 手动创建基准时钟，自动创建PLL输出时钟 6](#_Toc415063355)

[Method 3 – Create Base Clocks and PLL Output Clocks Manually 手动创建基准时钟和PLL输出时钟 6](#_Toc415063356)

[Multi-Frequency Analysis 多频率分析 7](#_Toc415063357)

[Clock Multiplexing 时钟多路传输 7](#_Toc415063358)

[Externally Switched Clock 外部切换时钟 8](#_Toc415063359)

[PLL Clock Switchover PLL时钟替换 8](#_Toc415063360)

[I/O Constraints I/O约束 9](#_Toc415063361)

[Input and Output Delays with Virtual Clocks 9](#_Toc415063362)

[Tri-State Outputs 12](#_Toc415063363)

[System Synchronous Input 系统同步输入 13](#_Toc415063364)

[System Synchronous Output 系统同步输出 15](#_Toc415063365)

[I/O Timing Requirements tSU, tH, and tCO 16](#_Toc415063366)

[Exceptions 17](#_Toc415063367)

[Multicycle Exceptions 17](#_Toc415063368)

[False Paths 18](#_Toc415063369)

[Miscellaneous 19](#_Toc415063370)

[JTAG Signals 19](#_Toc415063371)

[Input and Output Delays with Multiple Clocks 20](#_Toc415063372)

[Clock Enable Multicycle 25](#_Toc415063373)

这个手册由 design scenarios（设计脚本），约束指导，建议组成。为了用好这个手册，你应当对TimeQuest Timing Analyzer（时序约束分析）以及基本的集成电路设计约束（SDC）熟悉。

For information about the TimeQuest analyzer and SDC, refer to [www.altera.com/timequest](http://www.altera.com/timequest)

# Clocks and Generated Clocks (时钟和衍生时钟)

这章节展示了多种时钟结构以及约束它们的方法

■ “Basic Non-50/50 Duty Cycle Clock”                     方波时钟

■ “Offset Clocks” on page 1–2                             有偏移的时钟

■ “Basic Clock Divider Using -divide\_by” on page 1–2 分频时钟

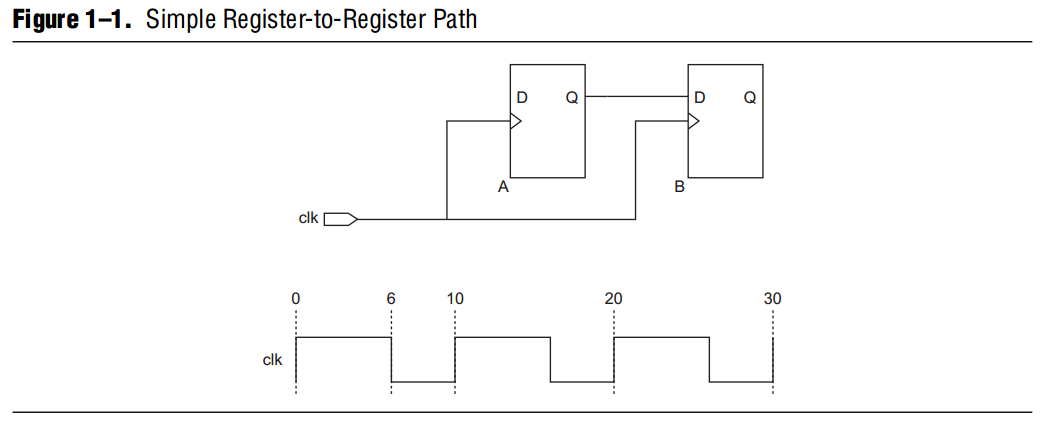
■ “Toggle Register Generated Clock” on page 1–4          触发寄存器衍生时钟

■ “Multi-Frequency Analysis” on page 1–7             多频分析

## Basic Non-50/50 Duty Cycle Clock

这部分介绍约束命令 ：*create\_clock*(创建时钟)。

时钟的占空比在不同的设计中是迥异的。默认条件下，在Timequest Analyser中创建的时钟都是占空比50%。当然，你能使用 *-waveform* 选项来改变占空比，Figure 1–1展现了一个简单的寄存器到寄存器路径，该路径由一个60%占空比的时钟控制。



Example 1–1 展示了一个占空比6 0%时钟的约束

**Example 1–1.** Non-50/50 Duty Cycle Clock Constraints

#60/40 duty cycle clock

create\_clock **\**

**-**period 10**.**000 **\**

**-**waveform **{**0**.** 000 6 **.**000**}** **\**

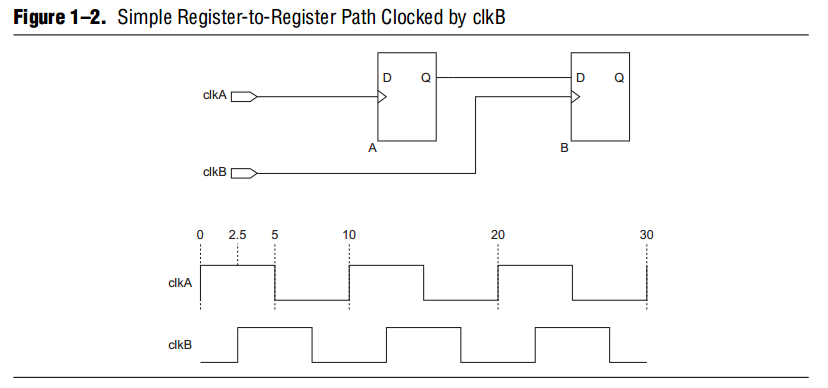
**-**name clk6040 **[**get\_ports **{**clk**}]**

## Offset Clocks(时钟偏移)

这部分介绍的约束命令是 *create\_clock*

当在Timequest Analyser中对clock进行约束的时候，第一个上升沿（fising）或者下降沿（falling edge）发生在绝对时间0。你能为这个时钟创建一个偏移（offset）通过 *-waveform* 选项.

Figure 1–2展现了一个由时钟clkB控制的寄存器-寄存器路径，该时钟向后偏移了2.5ns



Example 1–2是一个设置offset命令的示例

**Example 1–2.** Offset Clock Constraints

# -waveform defaults to 50/50 duty cycle

create\_clock **-**period 10**.**000 **\**

**-**name clkA **\**

**[**get\_ports **{** clkA**}]**

#create a clock with a 2.5 ns offset

create\_clock **-**period 10**.**000 **\**

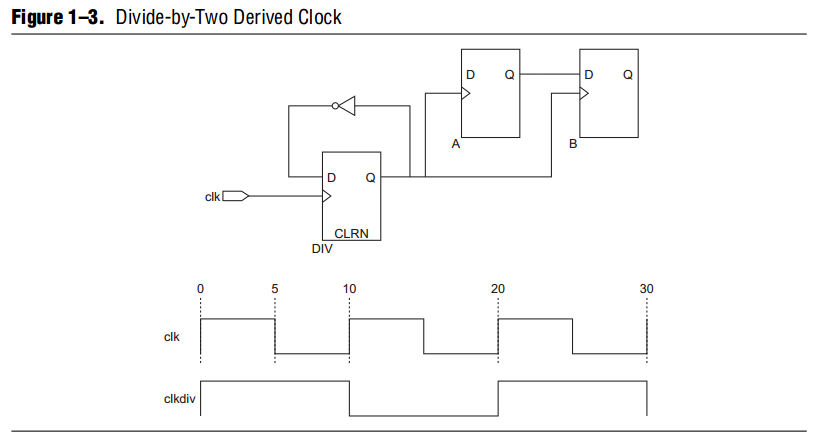
**-**waveform **{**2**.** 500 7 **.**500**}** **\**

**-**name clkB **[** get\_ports **{**clkB**}]**

## Basic Clock Divider Using -divide\_by

这部分描述的命令是 *create\_clock* 和 *create\_generated\_clock*

你可以通过一个源时钟衍生出一个时钟（衍生时钟derived clock），当这个时钟比源时钟慢的时候（You can derive clocks in a design from a clock source where the derived clock is slower than the source clock）。当约束一个比源时钟慢的衍生时钟，使用*-divide\_by* 选项。Figure 1–3显示了一个2分频衍生时钟。



Example 1–3 shows the constraints for a *divide-by* with -*waveform* clock  
是一个分频时钟约束的例子，通过-*waveform*命令

**Example 1–3.** Divide-by with -waveform Clock Constraints

create\_clock **-**period 10**.**000 **-**name clk **[**get\_ports **{** clk**}]**

#使用 -divide\_by 命令

create\_generated\_clock**\**

**-**divide\_by 2**\**

**-**source **[** get\_ports**{**clk**}]\**

**-**name clkdiv **\**

**[**get\_pins **{** DIV**|** q**}]**

#选择性的使用引脚来约束分频，在不知道主时钟的情况下

create\_generated\_clock**\**

**-**divide\_by 2**\**

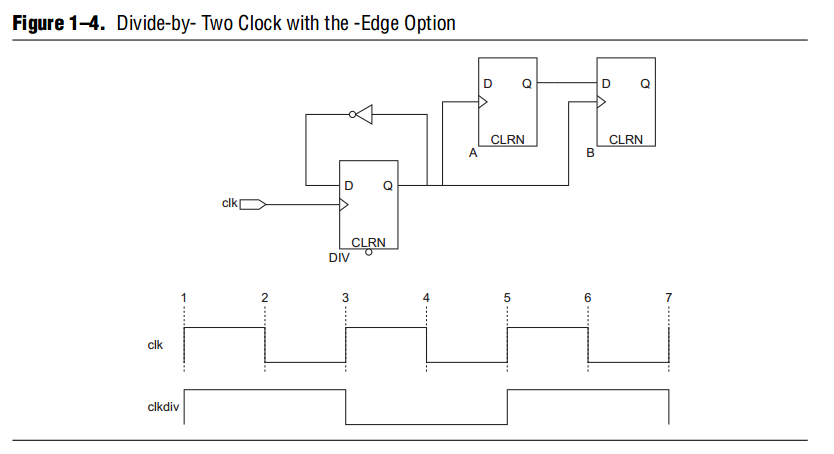
**-**source **[** get\_pins**{**DIV**|**clk **}]\**

**-**name clkdiv **\**

**[**get\_pins **{** DIV**|** q**}]**

#第二个命令工作，当寄存器 DIV的时钟引脚连接到由时钟 clk驱动的网络

你也可以使用 -*edges* 选项来创建分频时钟。Figure 1–4展示了用-edges 选项创建二分频时钟。



Example 1–4 shows constraints for the divide-by with -waveform clock.

**Example 1–4.** Divide-by with -waveform Clock Constraints

# Edge numbers are based on the master clock

create\_generated\_clock **\**

**-**edges **{**1 3 5**}** **\**

**-**source **[** get\_pins **{**DIV**|**clk **}]** **\**

**-**name clkdiv **\**

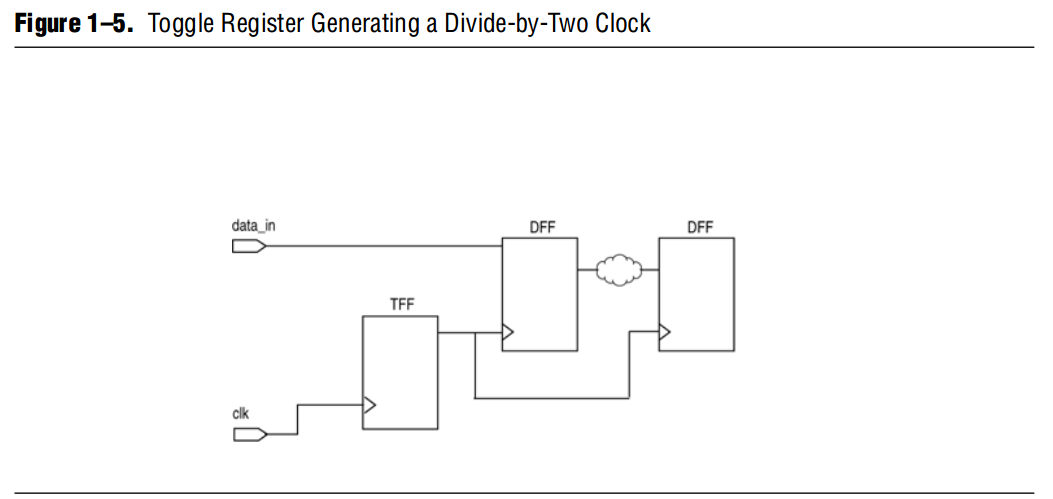
**[**get\_pins **{** DIV**|** q**}]**

## Toggle Register Generated Clock

使用toggle register 来创建一个二分频时钟。如果输入给toggle register的信号电平为1并由周期为10ns的时钟来驱动，那么该寄存器的输出则是一个周期20ns的时钟信号。

对toggle register 的约束和前面的例子十分相似。

Figure 1–5展示了一个toggle register 生成二分频时钟信号



Example 1–5示例了对toggle register的约束

**Example 1–5.** Toggle Register Constraints

#Create a base clock 创建基准时钟

create\_clock **\**

**-**period 10**.**000 **\**

**-**name clk **\**

**[**get\_ports **{** clk**}]**

#Create the generated clock on the output of the toggle register

#在toggle register 的输出端创建一个 generated时钟

create\_generated\_clock **\**

**-**name tff\_clk **\**

**-**source **[** get\_ports**{**clk**}]** **\**

**-**divide\_by 2 **\**

**[**get\_pins **{** tff**|** q**}]**

## PLL Clocks 锁相环时钟

该章表述的约束为 ***derive\_pll\_clocks***,**create\_clock**,和***create\_generated\_clock***

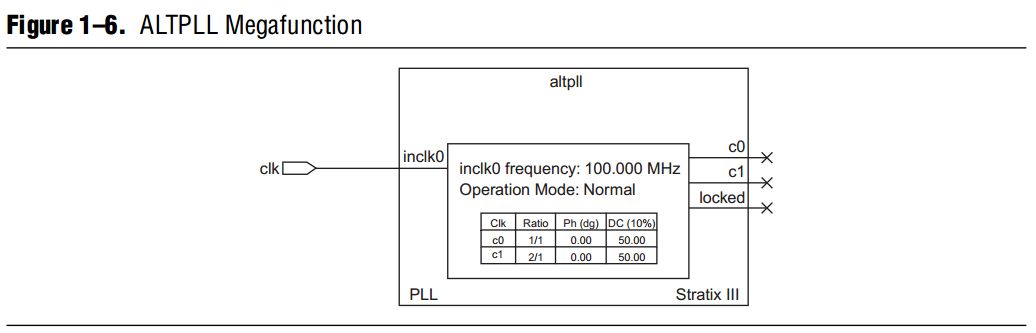
锁相环（PLL）用来在Altera FPGA中perform clock synthesis。所有的输出时钟必须被约束，为了进行合理的分析。有三种方法能够constrain 一个PLL。

■ Create base clocks and PLL output clocks automatically 自动创建基准时钟和PLL输出时钟  
■ Create base clocks manually and PLL output clocks automatically 手动创建基准时钟，自动创建PLL输出时钟  
■ Create base clocks manually and PLL output clocks manually 手动创建基准时钟和PLL输出时钟

这部分展示了每种方法的优点。

Altera中的PLL电路用ALTPLL模块集成

Figure 1–6 shows an example of the ALTPLL megafunction.



### Method 1 – Create Base Clocks and PLL Output Clocks Automatically 自动创建基准时钟和PLL输出时钟

这种方式让你能够自动约束PLL的输入和输出时钟。所有在ALTPLL中规定的参数都被用来约束PLL的输入和输出时钟。对ALTPLL模块的修正也是自动更新的。你不需要去track PLL参数的变化或者是指定 创建PLL输入、输出时钟 具体的参数。

为了自动约束所有的输入和输出，使用 ***derive\_pll\_clocks***命令以及 ***-create\_base\_clocks***选项.  Timequest Analyser决定了正确的输入 基于MegaWizard Plug-In Manager 的PLL实例。Example 1–6展示了这个命令：

**Example 1–6.** Constraining PLL Base Clocks Automatically

derive\_pll\_clocks -create\_base\_clocks

### Method 2 – Create Base Clocks Manually and PLL Output Clocks Automatically 手动创建基准时钟，自动创建PLL输出时钟

使用这种方法，你能够手动约束PLL的输入时钟，并且允许Timequest Analyser自动的去约束输出PLL时钟。除此之外，你还能够指定（specify）一个与ALTPLL模块不同的输入时钟频率。PLL输出时钟是自动创建的，使用ALTPLL模块中的参数。你能尝试不同的输入频率，但却保持相同的PLL输出参数。（**Note：可能想表达的意思是ALTPLL可以使用不同的输入频率产生相同的输出PLL时钟？**）

1 确保任何输入时钟频率与当前配置的PLL匹配。（**Note:神马意思？**）

你能够使用这个方法通过 ***derive\_pll\_clocks*** 命令，来手动创建PLL的输入时钟。Example 1–7展示了这一命令：

**Example 1–7.** Constraining PLL Base Clocks Manually

create\_clock -period 10.000 -name clk [get\_ports {clk}]  
derive\_pll\_clocks

### Method 3 – Create Base Clocks and PLL Output Clocks Manually 手动创建基准时钟和PLL输出时钟

通过该方法，你能够同时手动约束输入时钟和输出PLL时钟。所有的PLL参数 被指定，同时参数值也可以与ALTPLL模块中的不同。除此之外，你还能实验各种PLL输入和输出频率 以及各种参数。

你能够使用这种方法通过 ***create\_clock*** 和 ***create\_generate\_clock*** 命令的结合。Example 1–8展示了这种命令。

**Example 1–8.** Constraining PLL Output and Base Clocks Manually

create\_clock **-**period 10**.**000 **-**name clk **[**get\_ports **{** clk**}]**

create\_generated\_clock **\**

**-** name PLL\_C0 **\**

**-** source **[**get\_pins **{**PLL **|** altpll\_component**|**pll**|**inclk **[**0**]}]** **\**

**[** get\_pins **{**PLL**|**altpll\_component **|** pll**|** clk**[**0 **]}]**

create\_generated\_clock **\**

**-** name PLL\_C1 **\**

**-** multiply\_by 2 **\**

**-** source **[**get\_pins **{**PLL **|** altpll\_component**|**pll**|**inclk **[**0**]}]** **\**

**[** get\_pins **{**PLL**|**altpll\_component **|** pll**|** clk**[**1 **]}]**

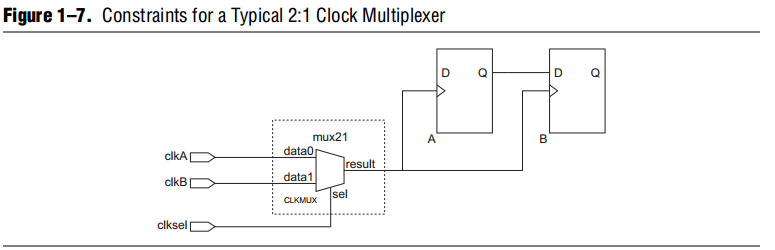
## Multi-Frequency Analysis 多频率分析

一些FPGA的设计当中会有多个时钟，而各个时钟的频率也会不同。

### Clock Multiplexing 时钟多路传输

这部分介绍的约束命令是：***create\_clock*** 和 ***set\_clock\_groups***

在时钟多路传输中，你可以从多个时钟里选择一个。Figure 1–7展示了一个典型的2:1时钟多路复用器（clock multiplexer） 约束



Example 1–9展示了约束命令 针对时钟多路复用器

**Example 1–9.** Clock Multiplexer Constraints

#Create the first input clock clkA to the mux

create\_clock **-**period 10**.**000 **-**name clkA **[**get\_ports **{**clkA**}]**

#Create the second input clock clkB to the mux

create\_clock **-**period 20**.**000 **-**name clkB **[**get\_ports **{**clkB**}]**

#Cut paths between clkA and clkB

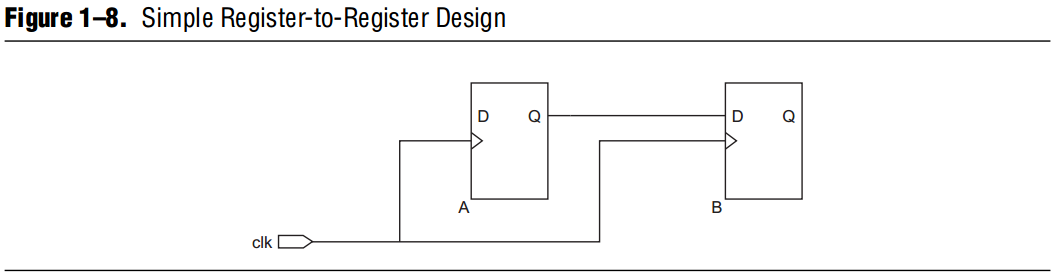
set\_clock\_groups **-**exclusive **-**group **{** clkA**}** **-**group **{** clkB**}**

### Externally Switched Clock 外部切换时钟

这部分介绍的约束命令是：***create\_clock*** 和 ***set\_clock\_groups***

通过一个外部时钟复用器或跳钱设置（jumper setting），数字系统能对同一个时钟端口提供不同频率的时钟。Timequest analyzer 能够建模这种实例，通过使用***create\_clock***约束命令和 -***add***选项。

Figure 1–8展示了一个简单的寄存器-寄存器路径，你可以使用100MHz或者50MHz的时钟来驱动其时钟端口



Example 1–10展示了对可切换外部时钟的约束

**Example 1–10.** Externally Switched Clock Constraints

# The clk port can be driven at 100MHz (10ns) or

# 50MHz (20ns)

# clkA is 10ns

create\_clock **\**

**-**period 10**.**000 **\**

**-**name clkA **\**

**[**get\_ports **{**clk**}]**

# clkB is 20ns assigned to the same port

# Requires -add option

create\_clock **\**

**-**period 20**.**000 **\**

**-**name clkB **\**

**[**get\_ports **{**clk**}]** **\**

**-**add

set\_clock\_groups **\**

**-**exclusive **\**

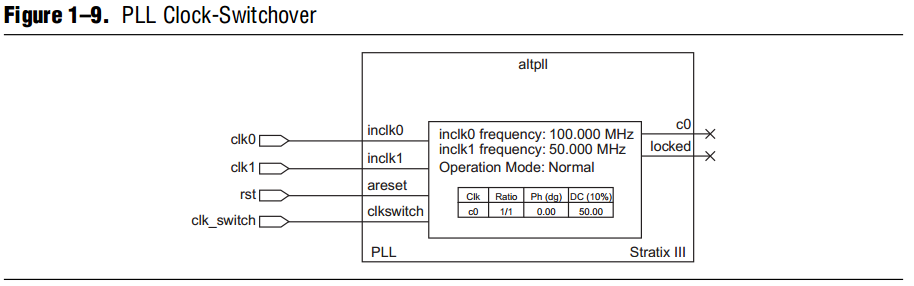
**-**group **{**clkA**}** **\**

**-**group **{**clkB**}**

### PLL Clock Switchover PLL时钟替换

这里描述了 derive\_pll\_clocks命令

PLL可以从两个输入时钟里选取一个 通过Altera FPGA里面的 PLL clock-switchover 输出(Figure 1–9).



Example 1–11 展示了对PLL clocks switchover 的约束

**Example 1–11.** PLL Clock Switchover Constraints

#create a 10ns clock for clock port clk0

create\_clock **\**

**-**period 10**.**000 **\**

**-**name clk0 **\**

**[**get\_ports **{**clk0**}]**

#create a 20ns clock for clock port clk1

create\_clock **\**

**-**period 20**.**000 **\**

**-**name clk1 **\**

**[**get\_ports **{**clk1**}]**

#automatically create clocks for the PLL output clocks

#derive\_pll\_clocks automatically makes the proper

#clock assignments for clock-switchover

derive\_pll\_clocks

# I/O Constraints I/O约束

This section contains the following topics:

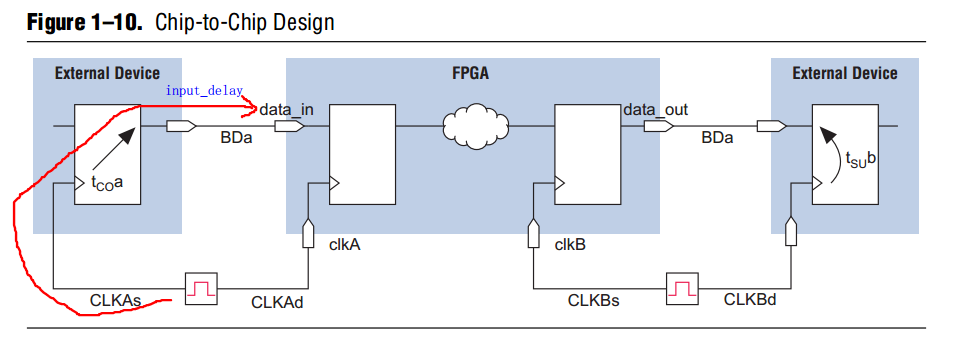
■ “Input and Output Delays with Virtual Clocks” on page 1–9 输入输出延时在虚拟时钟下  
■ “Tri-State Outputs” on page 1–13三态输出  
■ “System Synchronous Input” on page 1–14 系统同步输入

## Input and Output Delays with Virtual Clocks

 所有的输入输出延时都应当有一个参照的虚拟时钟，这样就能够在***derive\_clock\_uncertainty***命令被使用时,Timequest Analyse得到正确的时钟uncertainty value。如果输入输出延时参照基准时钟或者PLL时钟而不是irtual clock，那么 intra-clock 和inter\_clock的传递时钟uncertainty（由命令derive\_clock\_uncertainty决定）将被错误的应用在I/O口上。同时，如果使用virtual clock，额外的外部时钟不确定性也能够独立的添加到clock uncertainties里面，通过使用 ***derive\_clock\_uncertainty***命令。

虚拟时钟**virtual clock的属性**应当**和**用于输入（input delay）或输出（output delay）的**原始时钟original clock一致。**

Figure 1–10 展示了一个简单的chip-to-chip设计，这里virtual clock被用作输入和输出端口。



为了约束上图展示的输入输出延时，使用如 Example 1–12.的约束指令

**Example 1–12.** Input and Output Delays Referencing a Virtual Clock

#specify the maximum external clock delay from the external device

#指定从外部设备进来的最大外部时钟延时

**set** CLKAs\_max 0 **.**200

#specify the minimum external clock delay from the external device

#指定从外部设备进来的最小外部时钟延时

**set** CLKAs\_min 0 **.**100

#specify the maximum external clock delay to the FPGA

#指定信号传递到FPGA的最大外部时钟延时

**set** CLKAd\_max 0 **.**200

#specify the minimum external clock delay to the FPGA

#指定信号传递到FPGA的最小外部时钟延时

**set** CLKAd\_min 0 **.**100

#specify the maximum clock-to-out of the external device

#指定外部设备最大Tco延时

**set** tCOa\_max 0 **.**525

#specify the minimum clock-to-out of the external device

#指定外部设备最小Tco延时

**set** tCOa\_min 0 **.**415

#specify the maximum board delay

#定义最大的board延时（ PCB走线延时？）

**set** BDa\_max 0 **.**180

#specify the minimum board delay

#定义最小的board延时

**set** BDa\_min 0 **.**120

#create the input maximum delay for the data input to the

#FPGA that accounts for all delays specified

#指定最大的输入延时（Note:即从 clkA处时钟上升沿开始计时<从 clk到

#clkA端口也有一段时延CLKAd，所以需要减去该值 >，数据进入FPGA的寄

#存器耗费的时间）

set\_input\_delay **-**clock clk **\**

**-**max **[** expr **$CLKAs\_max** **+** **$tCOa\_max** **+** **$BDa\_max** **-** **$CLKAd\_min]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

#create the input minimum delay for the data input to the #FPGA that

#accounts for all delays specified

#指定最小的输入延时

set\_input\_delay **-**clock clk **\**

**-**min **[** expr **$CLKAs\_min** **+** **$tCOa\_min** **+** **$BDa\_min** **-** **$CLKAd\_max]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

#create the input clock

#创建输入时钟

create\_clock **-**name clkA **-**period 10 **[**get\_ports clkA**]**

#create the associated virtual input clock

#创建与输入时钟关联的virtual clock

create\_clock **-**name clkA\_virt **-**period 10 **（Note:虚拟时钟是没有指定实际的端口的）**

#specify any uncertainty from the external clock to the virtual clock

#指定从外部时钟到virtual clock的不确定时间

set\_clock\_uncertainty **-**from **{** clkA\_virt **}** **-**setup 0**.**25

#create the output clock

#创建输出时钟

create\_clock **-**name clkB **-**period 5 **[**get\_ports clkB**]**

#create the associated virtual input clock

#创建相关联的virtual input clock

create\_clock **-**name clkB\_virt **-**period 5

#specify any uncertainty from the external clock to the virtual clock

#指定从外部时钟到virtual clock的不确定时间

set\_clock\_uncertainty **-**from **{** clkB\_virt **}** **-**setup 0**.**25

#determine internal clock uncertainties

#决定内部时钟不确定性

derive\_clock\_uncertainty

#create the input delay referencing the virtual clock

#引用vritual clock 来创建输入延时

#specify the maximum external clock delay from the external

#device

**set** CLKAs\_max 0 **.**200

#specify the minimum external clock delay from the external

#device

**set** CLKAs\_min 0 **.**100

#specify the maximum external clock delay to the FPGA

**set** CLKAd\_max 0 **.**200

#specify the minimum external clock delay to the FPGA

**set** CLKAd\_min 0 **.**100

#specify the maximum clock-to-out of the external device

**set** tCOa\_max 0 **.**525

#specify the minimum clock-to-out of the external device

**set** tCOa\_min 0 **.**415

#specify the maximum board delay

**set** BDa\_max 0 **.**180

#specify the minimum board delay

**set** BDa\_min 0 **.**120

#create the input maximum delay for the data input to the

#FPGA that accounts for all delays specified

set\_input\_delay **-**clock clkA\_virt **\**

**-**max **[** expr **$CLKAs\_max** **+** **$tCOa\_max** **+** **$BDa\_max** **-** **$CLKAd\_min]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

#create the input minimum delay for the data input to the

#FPGA that accounts for all delays specified

set\_input\_delay **-**clock clkA\_virt **\**

**-**min **[** expr **$CLKAs\_min** **+** **$tCOa\_min** **+** **$BDa\_min** **-** **$CLKAd\_max]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

#creating the output delay referencing the virtual clock

#specify the maximum external clock delay from the external

#device

**set** CLKBs\_max 0 **.**100

#specify the minimum external clock delay from the external

#device

**set** CLKBs\_min 0 **.**050

#specify the maximum external clock delay to the FPGA

**set** CLKBd\_max 0 **.**100

#specify the minimum external clock delay to the FPGA

**set** CLKBd\_min 0 **.**050

#specify the maximum clock-to-out of the external device

**set** tSUb\_max 0 **.**500

#specify the hold time of the external device

**set** tHb 0 **.**400

#specify the maximum board delay

**set** BDb\_max 0 **.**100

#specify the minimum board delay

**set** BDb\_min 0 **.**080

#create the output maximum delay for the data output from the

#FPGA that accounts for all delays specified

set\_output\_delay **-**clock clkB\_virt **\**

**-**max **[** expr **$CLKBs\_max** **+** **$tSUb\_max** **+** **$BDb\_max** **-** **$CLKBd\_min]** **\**

**[**get\_ports **{**data\_out**}]**

#create the output minimum delay for the data output from the

#FPGA that accounts for all delays specified

set\_output\_delay **-**clock clkB\_virt **\**

**-**min **[** expr **$CLKBs\_min** **-** **$tHb** **+** **$BDb\_min** **-** **$CLKBd\_max]** **\**

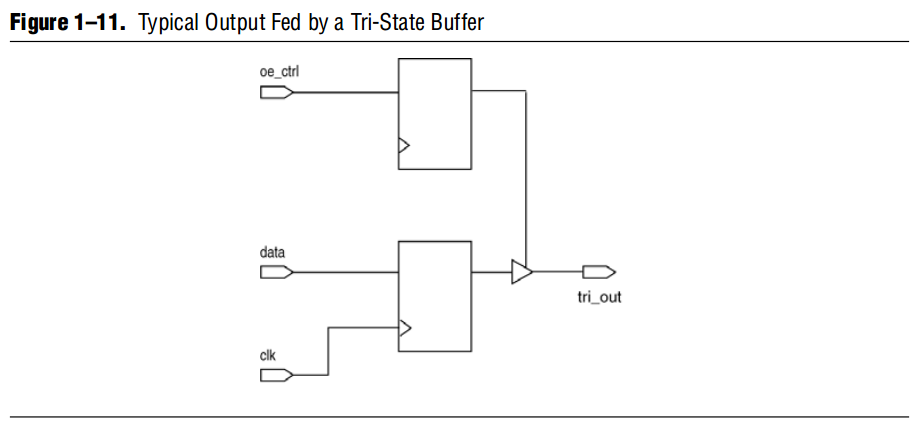
**[**get\_ports **{**data\_out**}]**

## Tri-State Outputs

Tri-state outputs allow either a valid data signal or a high impedance signal to be driven out of an input port. The timing of either signal is important in the overall system timing of the design.

三态输出允许一个有效的数据信号或者一个高阻抗的信号，被当作一个输入端口。任意信号的时序对整个系统来说都是很重要的。对三态输出的约束和一般的输出端口约束是一致的。

Figure 1–11 展示了一个典型的三态缓存器驱动的输出



Example 1–13 展示了对三态输出端口的约束

**Example 1–13.** Tri-State Output Port Constraints

# Base clock

create\_clock **[**get\_ports **{** clk**}]** **\**

**-** name **{** clk**}** **\**

**-** period 10 **.**0 **\**

**-** waveform **{**0**.**0 5 **.**0**}**

# Virtual clock for the output port

create\_clock **\**

**-** name **{** clk\_virt**}** **\**

**-** period 10 **.**0 **\**

**-** waveform **{**0**.**0 5 **.**0**}**

# Output constraints

set\_output\_delay 2**.** 0 **\**

**-** max **\**

**-** clock **[** get\_clocks **{**clk\_virt **}]** **\**

**[** get\_ports **{**tri\_out**}]**

set\_output\_delay 1**.** 0 **\**

**-** min **\**

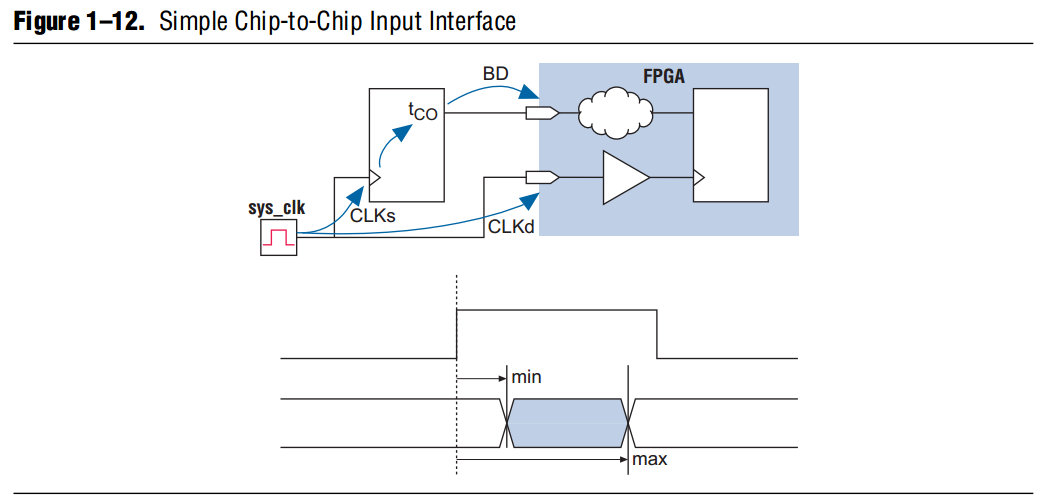
**-** clock **[** get\_clocks **{**clk\_virt **}]** **\**

**[** get\_ports **{**tri\_out**}]**

## **System Synchronous Input 系统同步输入**

这部分描述的约束命令是***create\_clock*** 和 ***set\_input\_delay***

Figure 1–12 展示了一个典型的chip-to-chip 输入接口，以及为了指定该接口输入延时所需要的参数。



Example 1–14展示了对一个同步系统输入的约束

**Example 1–14.** System Synchronous Input Constraints

#specify the maximum external clock delay from the external device

**set** CLKs\_max 0 **.**200

#specify the minimum external clock delay from the external device

**set** CLKs\_min 0 **.**100

#specify the maximum external clock delay to the FPGA

**set** CLKd\_max 0 **.**200

#specify the minimum external clock delay to the FPGA

**set** CLKd\_min 0 **.**100

#specify the maximum clock-to-out of the external device

**set** tCO\_max 0 **.**525

#specify the minimum clock-to-out of the external device

**set** tCO\_min 0 **.**415

#specify the maximum board delay

**set** BD\_max 0 **.**180

#specify the minimum board delay

**set** BD\_min 0 **.**120

#create a clock 10ns

create\_clock **-**period 10 **-**name sys\_clk **[**get\_ports sys\_clk**]**

#create the **associated virtual input clock**

create\_clock **-**period 10 **-**name virt\_sys\_clk

#create the input maximum delay for the data input to the FPGA that

#accounts for all delays specified

set\_input\_delay **-**clock virt\_sys\_clk **\**

**-**max **[** expr **$CLKs\_max** **+** **$tCO\_max** **+** **$BD\_max** **-** **$CLKd\_min]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

#create the input minimum delay for the data input to the FPGA that

#accounts for all delays specified

set\_input\_delay **-**clock virt\_sys\_clk **\**

**-**min **[** expr **$CLKs\_min** **+** **$tCO\_min** **+** **$BD\_min** **-** **$CLKd\_max]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

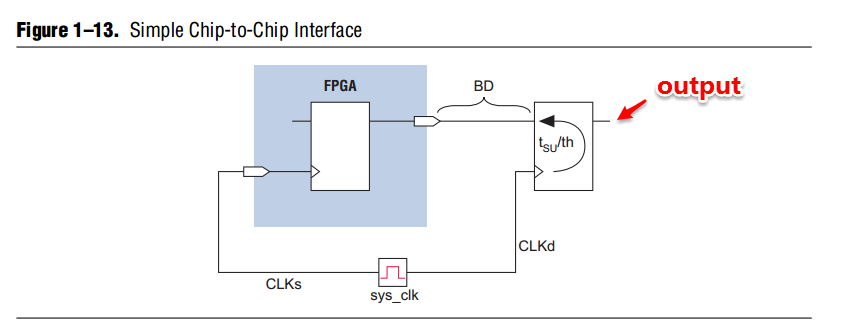
如果想要了解更多关于约束 源同步输入输出（source synchronous input and output）接口，请查询

[*AN 433: Constraining and Analyzing Source-Synchronous Interfaces*.](https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/an/an433.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=Constraining%20and%20Analyzing%20Source-Synchronous%20Interfaces.)

# **System Synchronous Output 系统同步输出**

这部分描述的约束命令是 ***create\_clock*** 和 ***set\_output\_delay***

Figure 1–13展示了一个典型的chip-to-chip输出接口，以及为了指定一个输出延时必要的各种参数。



Example 1–15 shows the constraints for system synchronous output  
**Example 1–15.** System Synchronous Output Constraints

#specify the maximum external clock delay to the FPGA

**set** CLKs\_max 0 **.**200

#specify the minimum external clock delay to the FPGA

**set** CLKs\_min 0 **.**100

#specify the maximum external clock delay to the external device

**set** CLKd\_max 0 **.**200

#specify the minimum external clock delay to the external device

**set** CLKd\_min 0 **.**100

#specify the maximum setup time of the external device

**set** tSU 0 **.**125

#specify the minimum setup time of the external device

**set** tH 0 **.**100

#specify the maximum board delay

**set** BD\_max 0 **.**180

#specify the minimum board delay

**set** BD\_min 0 **.**120

#create a clock 10ns

create\_clock **-**period 10 **-**name sys\_clk **[**get\_ports sys\_clk**]**

#create the associated virtual input clock

create\_clock **-**period 10 **-**name virt\_sys\_clk

#create the output maximum delay for the data output from the FPGA that

#accounts for all delays specified

set\_output\_delay **-**clock virt\_sys\_clk **\**

**-**max **[**expr **$CLKs\_max** **+** **$BD\_max** **+** **$tSU** **-** **$CLKd\_min]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

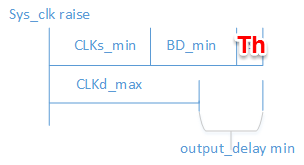
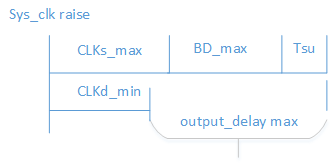
#create the output minimum delay for the data output from the FPGA that

#accounts for all delays specified

set\_output\_delay **-**clock virt\_sys\_clk **\**

**-**min **[**expr **$CLKs\_min** **+** **$BD\_min** **-** **$tH** **-** **$CLKd\_max]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

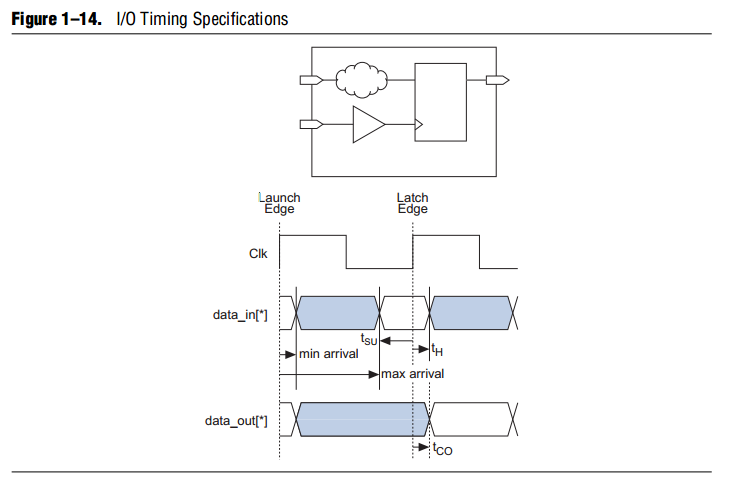


如果想要了解更多关于约束 源同步输入输出（source synchronous input and output）接口，请查询[*AN 433: Constraining and Analyzing Source-Synchronous Interfaces*.](https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/an/an433.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=Constraining%20and%20Analyzing%20Source-Synchronous%20Interfaces.)

## I/O Timing Requirements tSU, tH, and tCO

这部分描述的约束命令是 ***set\_input\_delay*** 和 ***set\_output\_delay***

Example 1–16展示了如何指定tsu和tH，通过用***set\_input\_delay ，***以及如何指定Tco，通过使用***set\_output\_delay。***Figure 1–14展示了一个在指定时序约束下的FPGA diagram



Example 1–16 shows the constraints for tSU, tH, and tCO.  
**Example 1–16.** tSU, tH, and tCO Constraints

#Specify the clock period

**set** period 10 **.**000

#Specify the required tSU

**set** tSU 1 **.**250

#Specify the required tH

**set** tH 0 **.**750

#Specify the required tCO

**set** tCO 0 **.**4

#create a clock 10ns

create\_clock **-**period **$period** **-** name clk **[**get\_ports sys\_clk **]**

#create the associated virtual input clock

create\_clock **-**period **$period** **-** name virt\_clk

set\_input\_delay **-**clock virt\_clk **\**

**-**max **[** expr **$period** **-** **$tSU]** **\**

**[**get\_ports **{**data\_in**[\*]}]**

set\_input\_delay **-**clock virt\_clk **\**

**-**min **$tH** **\**

**[**get\_ports **{**data\_in**[\*]}]**

set\_output\_delay **-**clock virt\_clk **\**

**-**max **[** expr **$period** **-** **$tCO]** **\（Note:这里不是很明白为什么这样计算）**

**[**get\_ports **{**data\_out**[\*]}]**

# Exceptions

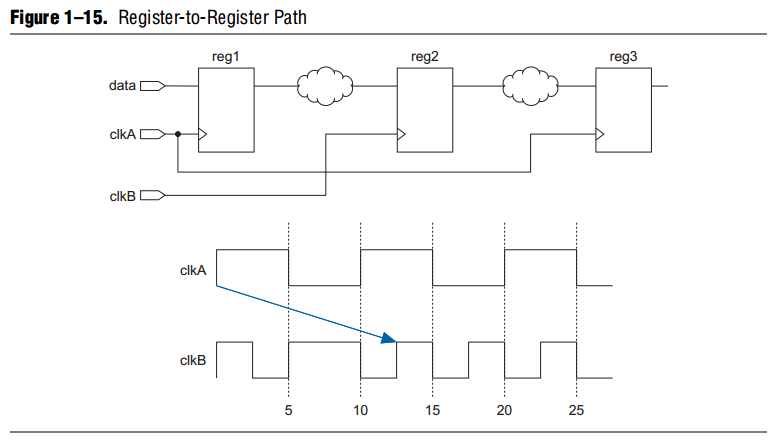
This section contains the following topics:  
     ■ “Multicycle Exceptions”           多周期选择  
     ■ “False Paths” on page 1–20     伪路径

## Multicycle Exceptions

这部分描述的约束命令是***create\_clock*** 和***set\_multicycle\_path。***

默认情况下，Timequest Analyser使用单周期分析寄存器之间的 setup relationship**（Note:即一个Launch沿对应的latch延间隔时长）** 和 hold relationship（**即hold check得到的时长**）。这就造成了最严格的setup 和hold 时序要求。

然而，多周期则可用来削弱这种严格的约束关系。Figure 1–15展示了一个简单的寄存器路径约束。（Note：这部分参看之前的一篇文章《[**不同时钟域之间的多周期路径**](http://bbs.ednchina.com/BLOG_ARTICLE_3028144.HTM)》）



多周期可以应用于所有的clock-to-clock transfer或者单独的寄存器individual register。在clock-to-clock transfer应用Multicycles，会影响所有源端到目的端的setup relationship 和hold relationship。Example 1–17展示了一个多周期约束。

**Example 1–17.** Multicycle Clock-to-Clock

create\_clock **-**period 10 **[** get\_ports clkA**]**

create\_clock **-**period 5 **[** get\_ports clkB**]**

set\_multicycle\_path **-**from **[** get\_clocks **{**clkA**}]** **-** to **[** get\_clocks **{**clkB **}]** **-** setup **-** end 2**（Note：CLKA和CLKB之间没有offset，所以设置Multicycle=2，即CLKB往后数第二个上升沿，即蓝色箭头指向的位置）**

在Example 1–17中，setup relationship 的约束条件更加宽松，因为多了一个时钟周期。这样reg1 和reg2 就有了一个长达12.5ns的setup relationship而不是默认单周期时候的5ns。而reg2和reg3之间的setup relationship没有受影响（**Note:为什么会没有影响呢**）。

将多周期multicycles 应用于单独的寄存器，那么仅仅会影响指定寄存器的setup hold relationship。Example 1–18 展示了对单独寄存器的多周期约束。

**Example 1–18.** Multicycle Register-to-Register

create\_clock **-**period 10 **[** get\_ports clkA**]**

create\_clock **-**period 5 **[** get\_ports clkB**]**

set\_multicycle\_path **-**from **[get\_pins {reg1 |q** **}]** **-** to **[get\_pins { reg2| d}]** **-** setup **-** end 2

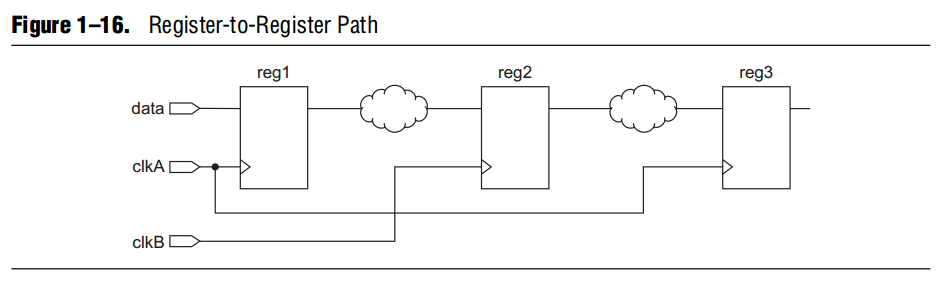
在 Example 1–18中，从reg1到reg2的setup relationship 通过增加了额外的一个时钟，变得更加宽松。这就使得reg1 、reg2的setup relationship长达12.5ns而不是5ns。而reg2和reg3之间的setup relationship没有受影响 。

如果想得到更多关于多周期约束的分析，请查阅 *Best Practices for*  *the TimeQuest Timing Analyzer*章节， in volume 3 of the *Quartus II Handbook*.

## False Paths

这部分描述的约束命令是create\_clock 和set\_false\_path.

我们并不需要对每一条路径都进行时序分析。非关键的同步路径可以从时序分析中移除。当我们声明非关键路径时，Quartus II的Fitter（布局布线工具）可以关注于关键路径的优化，减少编译时间。 Figure 1–16展示了一个简单的寄存器-寄存器的设计，在这个例子中，减去了对reg1到reg2的路径时序分析。



Palse Path 可以应用于时钟路径（clock-to-clock）或是独立的寄存器（individual register）。将Palse Path应用于时钟路径可以在时序分析中，排除目的时钟之间的所有路径。Example 1–19 展示了对false path的约束应用。

**Example 1–19.** False Path Clock-to-Clock

create\_clock **-**period 10 **[** get\_ports clkA**]**

create\_clock **-**period 5 **[** get\_ports clkB**]**

set\_false\_path **-**from **[** get\_clocks **{**clkA**}]** **-** to **[** get\_clocks **{**clkB **}]**

在Example 1–19中，所有源时钟是clka，目的时钟是clkb的寄存器之间的路径都被时序分析排除在外。同时，这样不会排除对源时钟是clkb，目的时钟是clka的路径时序分析。

对于Example 1–19来说，**set\_false\_path**命令除去了clka到clkb路径的时序分析。但是并没有出去clkb到clka的，为了出去clkb到clka的路径，必须再添加一个***set\_false\_path***命令。(for example,set\_false\_path -from clkB -to clkA)。另外，我们也以通过***set\_clock\_groups***这一个命令来同时除去这两条路径。

■ For more information about the set\_clock\_groups command, refer to *Set Clock* *Groups Dialog Box (set\_clock\_groups)* in Quartus II Help.

对单独寄存器使用false path命令，仅仅除去指定的路径。Example 1–20展示了这个约束。

**Example 1–20.** False Path Register-to-Register

create\_clock **-**period 10 **[** get\_ports clkA**]**

create\_clock **-**period 5 **[** get\_ports clkB**]**

set\_false\_path **-**from **[** get\_pins **{**reg1**|**q **}]** **-** to **[**get\_pins **{** reg2**|** d**}]**

# Miscellaneous

This section contains the following topics:

■ “JTAG Signals”                                                         JTAG信号

■ “Input and Output Delays with Multiple Clocks” on page 1–22 多时钟下的输入输出延时

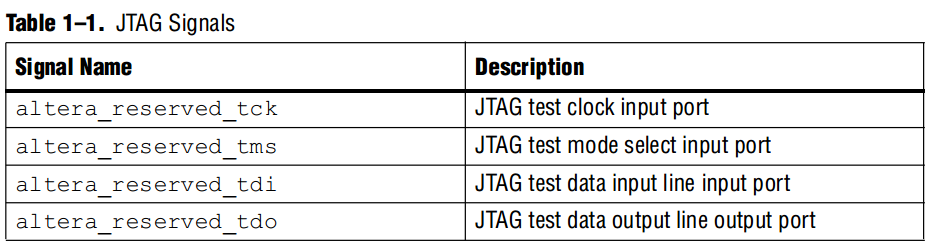
■ “Clock Enable Multicycle” on page 1–26 使能多时钟周期的时钟

## JTAG Signals

这里描述的约束命令是***create\_clock***, ***set\_input\_delay***, ***set\_output\_delay***

许多系统调试工具使用JTAG接口。当调试时，JTAG的TCK，TMS，TDI，和TDO信号将成为设计中的一部分。因此，Timequest Analyser标记这些信号为unconstrained，当unconstrained path report生成的时候。

Table 1–1 shows the JTAG signals that might appear as unconstrained.



你可以通过SDC命令对JTAG信号进行时序约束。

**Example 1–21.** JTAG Signal Constraints

#JTAG Signal Constraints

#constrain the TCK port

create\_clock **\**

**-** name tck **\**

**-** period "10MHz" **\**

**[** get\_ports altera\_reserved\_tck**]**

#cut all paths to and from tck

set\_clock\_groups **-**exclusive **-** group **[** get\_clocks tck**]**

#constrain the TDI port

set\_input\_delay **\**

**-** clock tck **\**

     20 **\**

**[** get\_ports altera\_reserved\_tdi**]**

#constrain the TMS port

set\_input\_delay **\**

**-** clock tck **\**

     20 **\**

**[** get\_ports altera\_reserved\_tms**]**

#constrain the TDO port

set\_output\_delay **\**

**-** clock tck **\**

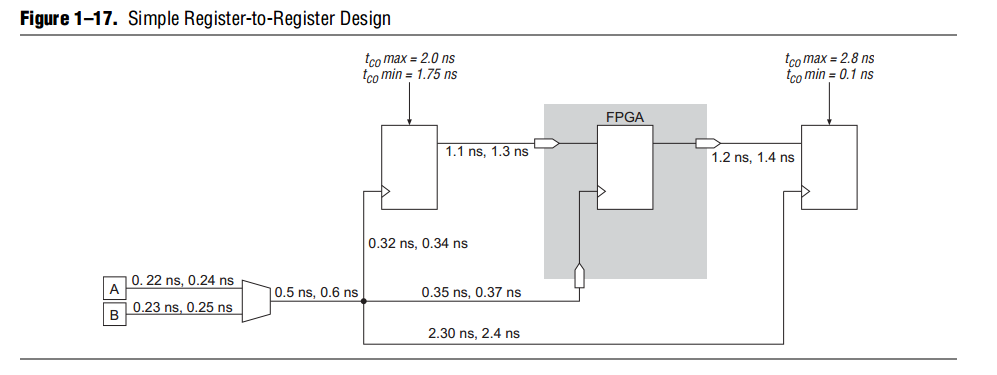
     20 **\**

**[** get\_ports altera\_reserved\_tdo**]**

## Input and Output Delays with Multiple Clocks

这部分描述的约束命令是***create\_clock, create\_generated\_clock, set\_clock\_groups, set\_clock\_latency, set\_input\_delay, and set\_output\_delay.***

These constraints provide both a primary and secondary clock.  The primary clock acts as the main clock and the secondary clock may act as a redundant clock at a slower speed. Figure 1–17 shows an example of this setup。



Example 1–22 shows the command for input delay with multiple clocks.

**Example 1–22.** Input Delay with Multiple Clocks (Part 1 of 4)

#########################

# Create all the clocks #

#########################

# Create variables for the clock periods.

**set** PERIOD\_CLK\_A 10 **.**000

**set** PERIOD\_CLK\_B 7 **.**000

# Create the clk\_a clock which will represent the clock  that routes to the FPGA.

# 创建clk\_a时钟，该时钟代表了布线到FPGA的时钟

create\_clock **\**

**-** name **{** clk\_a**}** **\**

**-** period  **$PERIOD\_CLK\_A** **\**

**[** get\_ports **{**clk**}]**

# Create the clk\_b clock which will represent the clock that routes to the FPGA.

# Note the -add is needed because this is the second clock that has the same 'clk' port as a target.

# 请注意-add命令是必须的，因为这是clk端口的第二个时钟

create\_clock **\**

**-** name **{** clk\_b**}** **\**

**-** period **$PERIOD\_CLK\_B** **\**

**[** get\_ports **{**clk**}]** **\**

**-** add

# Create a virtual clock which will represent the clock  that routes to the external source device when clk\_a is selected a the external mux.

# 创建一个虚拟时钟，该时钟将代表，clka被选作时钟时，连接到外部源时钟设备的时钟

create\_clock **\**

**-** name virtual\_source\_clk\_a **\**

**-** period **$PERIOD\_CLK\_A**

# Create a virtual clock which will represent the clock that routes to the external source device when clk\_b is selected a the external mux.

# 创建一个虚拟时钟，该时钟将代表，连接到外部源时钟设备

create\_clock **\**

**-** name virtual\_source\_clk\_b **\**

**-** period **$PERIOD\_CLK\_B**

# Create a virtual clock which will represent the clock that routes to the external destination device when clk\_a is selected a the external mux.

create\_clock **\**

**-** name virtual\_dest\_clk\_a **\**

**-** period **$PERIOD\_CLK\_A**

# Create a virtual clock which will represent the clock that routes to the external destination device when clk\_b is selected a the external mux.

create\_clock **\**

**-** name virtual\_dest\_clk\_b **\**

**-** period **$PERIOD\_CLK\_B**

**Example 1–22.** Input Delay with Multiple Clocks (Part 2 of 4)

##########################################

# Cut clock transfers that are not valid #

##########################################

# Cut this because virtual\_source\_clk\_b can not be clocking

# the external source device at the same time that clk\_a is

# clocking the FPGA.

set\_clock\_groups **-**exclusive **\**

**-**group **{**clk\_a**}** **\**

**-**group **{**virtual\_source\_clk\_b**}**

# Cut this because virtual\_source\_clk\_a can not be clocking

# the external source device at the same time that clk\_b is

# clocking the FPGA.

set\_clock\_groups **-**exclusive **\**

**-**group **{**clk\_b**}** **\**

**-**group **{**virtual\_source\_clk\_a**}**

# Cut this because virtual\_dest\_clk\_b can not be clocking

# the external destination device at the same time that

# clk\_a is clocking the FPGA.

set\_clock\_groups **-**exclusive **\**

**-**group **{**clk\_a**}** **\**

**-**group **{**virtual\_dest\_clk\_b**}**

# Cut this because virtual\_dest\_clk\_a can not be clocking

# the external destination device at the same time that

# clk\_b is clocking the FPGA

set\_clock\_groups **-**exclusive **\**

**-**group **{**clk\_b**}** **\**

**-**group **{**virtual\_dest\_clk\_a**}**

########################################

# Define the latency of all the clocks #

########################################

# Since TimeQuest does not know what part of the clock

# latency is common we must simply remove the common part

# from the latency calculation. For example when

# calculating the latency for virtual\_source\_clk\_a we must

# ignore the 220ps,240ps route and the 500ps/600ps mux

# delay if we want to remove the common clock path

# pessimism.

#

# Define fastest and slowest virtual\_source\_clk\_a path to

# the external source device.

set\_clock\_latency **-**source **\**

**-**early **.**320 **\**

**[**get\_clocks virtual\_source\_clk\_a **]**

set\_clock\_latency **-**source **\**

**-**late **.**340 **\**

**[**get\_clocks virtual\_source\_clk\_a **]**

# Define fastest and slowest virtual\_source\_clk\_b path to

# the external source device.

set\_clock\_latency **-**source **\**

**-**early **.**320 **\**

**[**get\_clocks virtual\_source\_clk\_b **]**

set\_clock\_latency **-**source **\**

**-**late **.**340 **\**

**[**get\_clocks virtual\_source\_clk\_b **]**

# Define fastest and slowest clk\_a path to the FPGA.

set\_clock\_latency **-**source **\**

**-**early **.**350 **\**

**[**get\_clocks clk\_a **]**

set\_clock\_latency **-**source **\**

**-**late **.**370 **\**

**[**get\_clocks clk\_a **]**

**Example 1–22.** Input Delay with Multiple Clocks (Part 3 of 4)

# Define fastest and slowest clk\_b path to the FPGA.

set\_clock\_latency **-**source **\**

**-**early **.**350 **\**

**[**get\_clocks clk\_b **]**

set\_clock\_latency **-**source **\**

**-**late **.**370 **\**

**[**get\_clocks clk\_b **]**

# Define fastest and slowest virtual\_dest\_clk\_a path to

# the external destination device.

set\_clock\_latency **-**source **\**

**-**early 2**.**3 **\**

**[**get\_clocks virtual\_dest\_clk\_a **]**

set\_clock\_latency **-**source **\**

**-**late 2**.**4 **\**

**[**get\_clocks virtual\_dest\_clk\_a **]**

# Define fastest and slowest virtual\_dest\_clk\_b path to

# the external destination device.

set\_clock\_latency **-**source **\**

**-**early 2**.**3 **\**

**[**get\_clocks virtual\_dest\_clk\_b **]**

set\_clock\_latency **-**source **\**

**-**late 2**.**4 **\**

**[**get\_clocks virtual\_dest\_clk\_b **]**

#####################################

# Constrain the input port 'datain' #

#####################################

# This Tco is the min/max value of the Tco for the

# external module.

**set** Tco\_max 2 **.**0

**set** Tco\_min 1 **.**75

# Td is the min/max trace delay of datain from the

# external device

**set** Td\_min 1 **.**1

**set** Td\_max 1 **.**3

# Calculate the input delay numbers

**set** input\_max **[**expr **$Td\_max** **+** **$Tco\_max]**

**set** input\_min **[**expr **$Td\_min** **+** **$Tco\_min]**

# Create the input delay constraints when clk\_a is selected

set\_input\_delay **\**

**-**clock virtual\_source\_clk\_a **\**

**-**max **$input\_max** **\**

**[**get\_ports datain **]**

set\_input\_delay **\**

**-**clock virtual\_source\_clk\_a **\**

**-**min **$input\_min** **\**

**[**get\_ports datain **]**

# Create the input delay constraints when clk\_b is selected

set\_input\_delay **\**

**-**clock virtual\_source\_clk\_b **\**

**-**max **$input\_max** **\**

**[**get\_ports datain **]** **\**

**-**add\_delay

set\_input\_delay **\**

**-**clock virtual\_source\_clk\_b **\**

**-**min **$input\_min** **\**

**[**get\_ports datain **]** **\**

**-**add\_delay

**Example 1–22.** Input Delay with Multiple Clocks (Part 4 of 4)

#######################################

# Constrain the output port 'dataout' #

#######################################

# This Tsu/Th is the value of the Tsu/Th for the external

# device.

**set** Tsu 2 **.**8

**set** Th 0 **.**1

# This is the min/max trace delay of dataout to the

# external device.

**set** Td\_min 1 **.**2

**set** Td\_max 1 **.**4

# Calculate the output delay numbers

**set** output\_max **[**expr **$Td\_max** **+** **$Tsu]**

**set** output\_min **[**expr **$Td\_min** **-** **$Th]**

# Create the output delay constraints when clk\_a is

# selected.

set\_output\_delay **\**

**-**clock virtual\_dest\_clk\_a **\**

**-**max **$output\_max** **\**

**[**get\_ports dataout **]**

set\_output\_delay **\**

**-**clock virtual\_dest\_clk\_a **\**

**-**min **$output\_min** **\**

**[**get\_ports dataout **]**

# Create the output delay constraints when clk\_b is

# selected.

set\_output\_delay **\**

**-**clock virtual\_dest\_clk\_b **\**

**-**max **$output\_max** **\**

**[**get\_ports dataout **]** **\**

**-**add\_delay

set\_output\_delay **\**

**-**clock virtual\_dest\_clk\_b **\**

**-**min **$output\_min** **\**

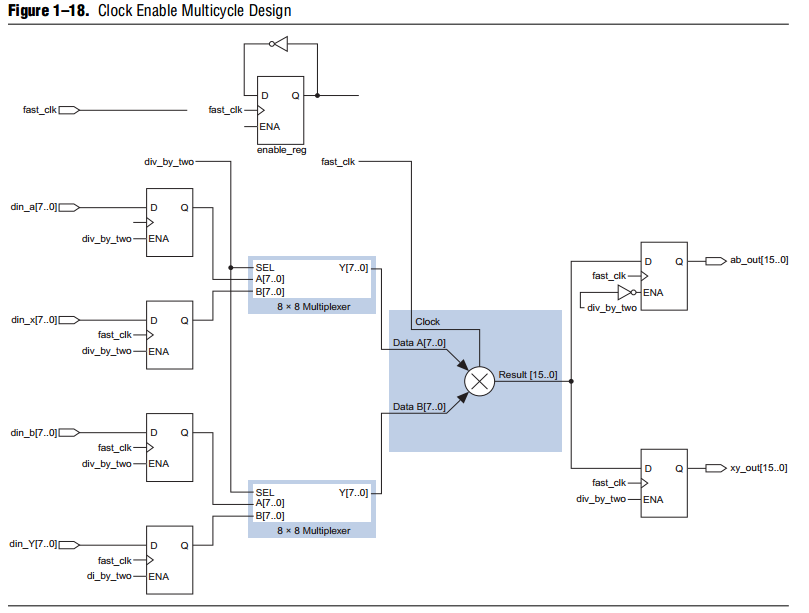
**[**get\_ports dataout **]** **\**

**-**add\_delay

## Clock Enable Multicycle

这部分描述的约束命令是 ***create\_clock***, ***set\_multicycle\_path***, ***get\_fanouts***

你可以为寄存器指定多周期。For example,Figure 1–18



展示了一个简单的电路，该电路中，寄存器enable\_reg的输出，被用来当作其他寄存器的 使能信号。

enable\_reg生成一个使能脉冲，该脉冲周期是寄存器周期的两倍（即二分频）；因此，需要设置Multicycle exception以进行正确的时序分析。我们必须对由enable\_reg寄存器驱动的register设置Multicycle=2，Multicycle hold=1。

该Multicycle exception只被应用于目的寄存器由enable\_reg控制的reg-to-reg路径。我们可以使用set\_multicycle\_path 命令，但命令却会变得冗长，因为所有的寄存器都必须要被指定。另一个替代的方式是使用set\_multicycle\_path 和get\_fanouts命令。见例Example 1–23.

**Example 1–23.** Clock Enable Multicycle Constraints

#Setup multicycle of 2 to enabled driven destination registers

set\_multicycle\_path 2 **-**to **[**get\_fanouts **[**get\_pins enable\_reg**|**q **\*]** **-** through **[**get\_pins **-**hierarchical **\*|\***ena **\*]]**

**-**end

**-**setup

#Hold multicycle of 1 to enabled driven destination registers

set\_multicycle\_path 1 **-**to **[**get\_fanouts **[**get\_pins enable\_reg**|**q **\*]** **-** through **[**get\_pins **-**hierarchical **\*|\***ena **\*]]**

**-**end

**-**hold

set\_multicycle\_path命令被作用于enable\_reg寄存器所有扇出端口，通过调用下面的这个命令：

**[**get\_fanouts **[**get\_pins enable\_reg**|**q **\*]** **-** through **[**get\_pins **-**hierarchical **\*|\***ena**\*]]**

Table 1–2展示了新的setup relationship 和hold relationship。

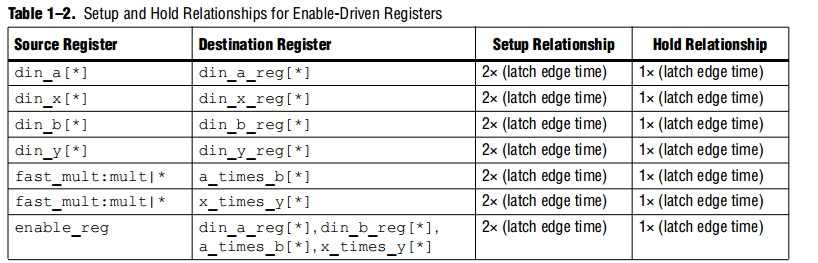


Table 1–2展示了从enable\_reg寄存器开始，结束于被驱动的寄存器的setup hold relationship。如果这些路径不要求setup hold relationship被修改，可以使用如下命令

set\_multicycle\_path 1 **-**from **[**get\_pins enable\_reg**|**q**\*]** **-**end **-** setup

set\_multicycle\_path 0 **-**from **[**get\_pins enable\_reg**|**q**\*]** **-**end **-** hold

如果想得到更多关于多周期约束的分析，请查阅 *Best Practices for*  *the TimeQuest Timing Analyzer*章节， in volume 3 of the *Quartus II Handbook*.