第5章

时序例外约束

在基本时序路径、时钟约束和输入/输出延时约束章节，案例的源时钟和目标时钟都是同频同相或同频相位差 90°；源时钟和目标时钟在每一个周期的相位都是固定的，分析一个时钟周期的建立时间和保持时间关系，即可代表这条时序路径的时序特性。异步时钟/跨时钟域的时序路径分析时，源时钟和目标时钟的频率与相位不固定，时序分析时需要分析它们最糟糕的频率相位关系，时序过紧难收敛，则不能满足两级寄存器的建立时间和保持时间要求。

时序分析工具采用前文介绍的时序分析方法时往往会出现时序违例，设计者应结合逻辑功能对跨时钟域的路径额外添加一些约束指令，以放宽时序要求且保证逻辑功能正常。有些设计需要对时序路径施加更紧的时序约束，例如跨时钟域-打拍消除亚稳态，额外添加一些约束指令，以施加更紧的时序要求且保证逻辑功能正常。这些额外的约束称为时序例外约束，时序例外约束主要包括伪路径约束/时钟组约束、最大/最小延时约束、多周期路径约束，如图 5-1 所示。

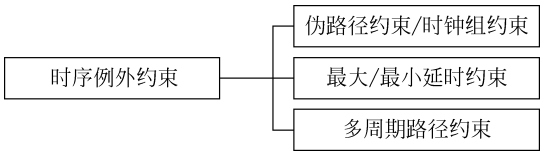


图 5-1　时序例外约束

5.1　时序例外约束的意义

当源时钟和目标时钟为同频同相且仅有少量偏斜 Skew 时，寄存器到寄存器时序路径一般不需要特殊约束，时序工具进行时序检查时一般不会出现违例；当寄存器到寄存器时序路径异步跨时钟域时，极端时刻时序过紧会出现时序违例，建立时间和保持时间要求不能满足，因此需要添加时序例外约束。同步时钟与异步跨时钟域如图 5-2 所示，当 clk1 和 clk2 同频同相时，建立时间关系和保持时间关系时序宽松，一般不添加时序约束也不会出

现时序违例；当 clk1 和 clk2 异步跨时钟域时，极端情况的建立时间关系 k 时刻启动沿到 i 时刻锁存沿，时序紧张且很难满足目标寄存器的建立时间要求，这时就会出现时序违例。

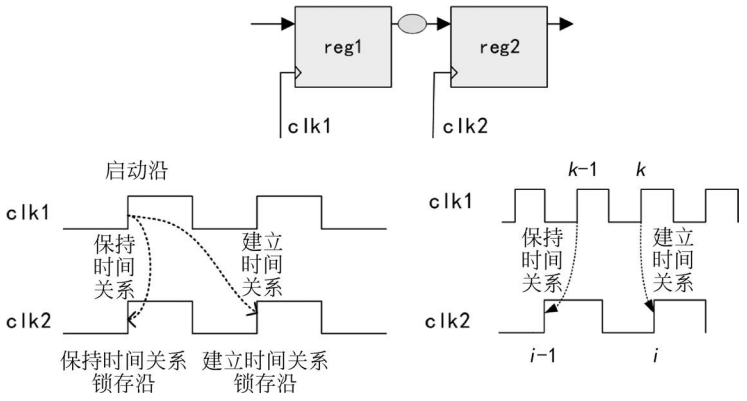


图 5-2　同步时钟与异步跨时钟域

接下来各个击破，看看时序例外约束的意义是什么。

伪路径(set_false_path)约束/时钟组(set_clock_groups)约束：源寄存器和目标寄存器无时序要求，数据到达时间不约束，任意时间均可，可以使用伪路径约束/时钟组约束忽略这些路径，伪路径约束/时钟组约束如图 5-3 所示。注意：伪路径并非路径不存在，只是该路径没有时序要求不进行时序分析。

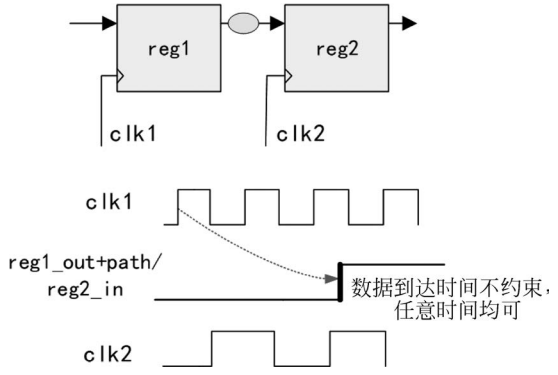


图 5-3　伪路径约束/时钟组约束

在寄存器 reg1 和 reg2 之间添加伪路径/时钟组约束，reg1 和 reg2 之间的组合逻辑可以随意布线，数据到达时间也不约束；寄存器 reg2 在 clk2 上升沿采到什么值就锁存什么值。

最大/最小延时(set_max_delay/set_min_delay)约束：最大/最小延时约束主要约束数据到达时间的延时，最小延时＜数据到达时间＜最大延时，以获得更大的时序余量，提高逻辑稳定性，最大/最小延时约束如图 5-4 所示。

图 5-4 中，跨时钟域-打拍消除亚稳态，寄存器 reg0(clk0)向寄存器 reg1(clk1)传递信号跨时钟域，为了消除寄存器 reg1 采样亚稳态，寄存器 reg2 再次采样打拍操作；在满足建立保持时间要求的前提下，在寄存器 reg1 和 reg2 之间添加了最大/最小延时约束，数据到达时间被限制在很小的区间内，数据到达时间越早，寄存器 reg2 就能采样到越稳定的数据。

当然也可以利用最大/最小延时约束放宽时序要求，请对比 3.3.3 节时钟不确定性约束

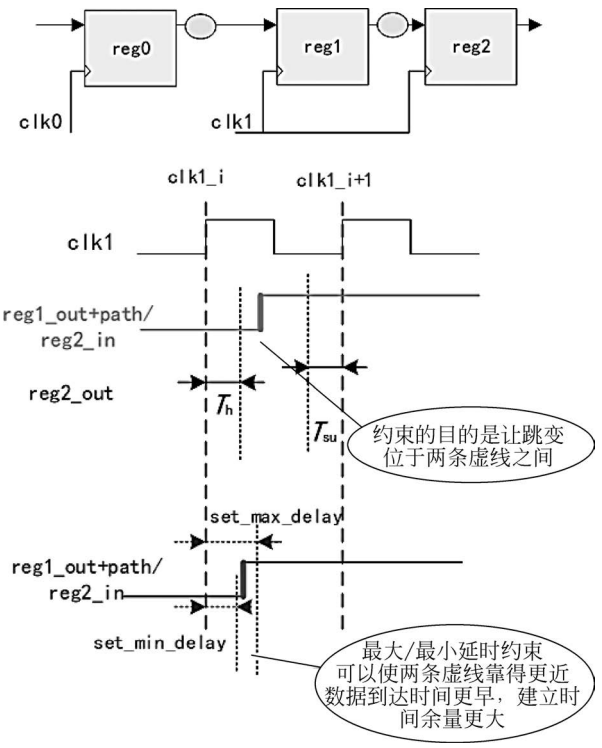


图 5-4　最大/最小延时约束

的妙用，通过 set_max_delay/set_min_delay、set_clock_uncertainty、增加时钟频率提高时序余量的方法有异曲同工之妙，也说明了时序约束真的是条条大路通罗马。

多周期路径约束(set_multicycle_path)：如果数据不是每个周期都更新，数据到达时间就可以多跑几个周期，放宽这些路径约束，多周期路径约束如图 5-5 所示。

图 5-5 中设计了一条多周期路径，其中 clk1 的时钟频率是 clk2 的 3 倍，也就是说，寄存器 reg1 输出 3 次数据，寄存器 reg2 只锁存一次；reg1 输出数据到达时间(跳变)就没必要“赶”在一个周期内到达，即便一个周期内到达，reg2 也不着急锁存；添加多周期路径约束，放宽该路径的时序要求，reg1 输出的数据到达时间(跳变)在 3 个周期后 T_{su} 之前到达就行。

伪路径/时钟组约束后，编译工具不做任何要求和时序分析；多周期路径约束后，编译工具需要在放宽的时序范围内，布局布线并时序分析，数据到达时间不能超出多周期；最大/最小延时约束后，编译工具需要在布局布线时将数据延时(数据到达时间)“卡在”最大/最小延时之间。

FPGA 内部的时钟默认都是相关的，时序分析时所有的时序路径都会进行时序检查。一般地，同时钟源的时序路径时序分析时，不会出现时序违例(超高频除外)；异步跨时钟域数据传递往往时序过紧会存在时序违例，就必须添加时序例外约束，告诉 Vivado 编译工具，这部分时序需要放宽，可以使用多周期约束、伪路径约束、时钟组约束；有些时序路径需要过紧的约束，以提高数据传递的稳定性，必须添加时序例外约束，告诉 Vivado 编译工具，这部分时序需要加紧，可以使用最大/最小延时约束；使用最大/最小延时约束解决异步跨时钟域时序过紧，放宽约束，当然也是可以的；时序例外约束的意义就在这一紧一松之间。

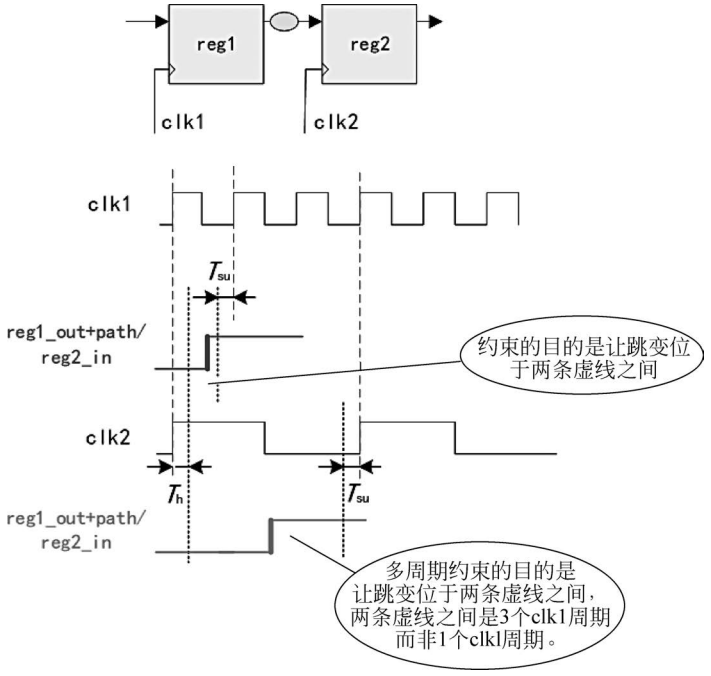


图 5-5　多周期路径约束

FPGA 有限的资源在布局布线时，既要把逻辑放进去，又要满足时序要求。这与整理行李箱的道理是一样的，需要挤在一起的逻辑找地方单独放（时序加紧），不需要挤在一起的逻辑找个缝塞进去（时序宽松/无时序要求）。时序例外约束可以使 FPGA 布局布线合理分配资源，又能保证系统的时序收敛。

在 Vivado 软件中，选中违例的时序路径右键，会有 Set False Path、Set Multicycle Path 和 Set Maximum Delay 选项，直接对该违例的路径添加时序例外约束，如图 5-6 所示。单击会自动跳转到 GUI 配置界面，使用 XDC 指令约束效果也是一样的。

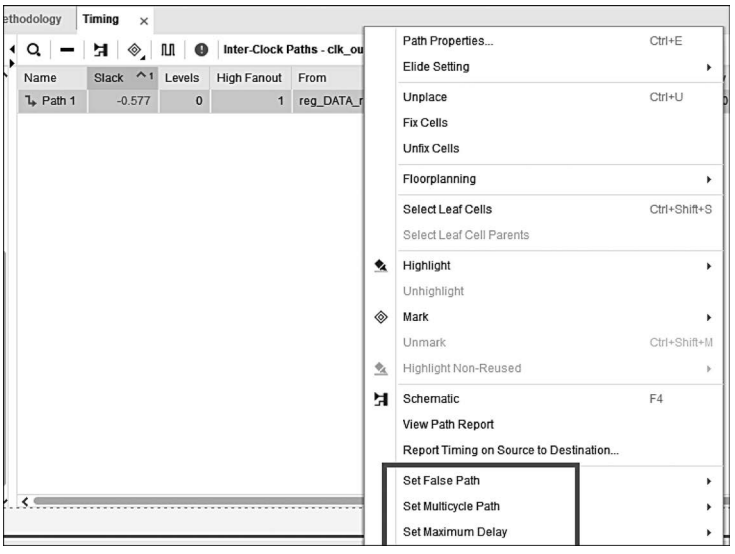


图 5-6　对违例路径进行时序例外约束

5.2 伪路径约束/时钟组约束

Vivado 默认对所有时钟之间的路径进行时序分析，除非在指定路径中使用伪路径约束和时钟组约束，这两种约束可以达到一样的效果。

时钟组约束 set_clock_groups 指令用于禁用时钟组之间的时序分析，而时钟组内的所有时钟之间仍然进行时序分析；伪路径约束 set_false_path 指令用于忽略两个时钟之间的时序分析，建立时间和保持时间可分开约束。

伪路径约束和时钟组约束并不能消除亚稳态，逻辑电路中需要有同步电路或异步传输协议消除亚稳态。

5.2.1 伪路径约束语法

伪路径约束使编译工具不再为指定的路径进行布局布线努力，也不再分析指定路径的时序。以 Vivado 为例，使用 set_false_path 指令定义伪路径约束。set_false_path 约束指令的语法结构如下：

```
set_false_path -setup / -hold
               -from       <node list>
                           -to          <node list>
                           -through     <node list>
```

在 Vivado 中，set_false_path 约束指令的参数定义如表 5-1 所示。

表 5-1 set_false_path 约束指令的参数定义

参　数	说　明
-setup/-hold	-setup 是指将时序路径约束为建立时间分析伪路径(仅分析保持时间)；-hold 是指将时序路径约束为保持时间分析伪路径(仅分析建立时间)；不指定该选项，将时序路径约束为伪路径，建立时间和保持时间都不分析
-from	指定伪路径的起点，可以是一个列表，对象可以是时钟、端口、引脚和单元
-to	指定伪路径的终点，可以是一个列表，对象可以是时钟(驱动寄存器的数据输入引脚)、端口、引脚和单元
-through	指定途径点，可以是一个列表，对象可以是端口、引脚。多个-through 选项时，节点之间有先后顺序，例如以下两条指令并非同一条伪路径： set_false_path -through pin1 -through pin2 set_false_path -through pin2 -through pin1

值得注意的是，选项中，-from、-to 和-through 只指定其中一项，所有经过指定节点的路径都会被设置为伪路径，这样的做法很危险。set_false_path 设定的伪路径最好是一条/一类非常明确的路径。

Case1：将 reset 端口到其连接的所有寄存器设置为伪路径。

```
set_false_path  -from [get_port reset]
```

Case2：将源时钟 clk1 到目标时钟 clk2 的所有时序路径设置为伪路径。

```
set_false_path   -from [get_clocks clk1]   -to [get_clocks clk2]
```

Case3：将源时钟 clk1 到目标时钟 clk2 的所有时序路径，以及源时钟 clk2 到目标时钟 clk1 的所有时序路径，都设置为伪路径。

```
set_false_path   -from [get_clocks clk1]   -to [get_clocks clk2]
set_false_path   -from [get_clocks clk2]   -to [get_clocks clk1]
```

Case1 中，当仅有-from 选项时，代表该起点连接的所有路径设置为伪路径；Case2 中，伪路径约束仅有一个方向，由 from 节点到 to 节点；Case3 中，需要约束两个方向的伪路径，则需要两条约束指令分别约束。

5.2.2　伪路径约束实例

创建一个时序逻辑工程 project14，其工程代码如 project14.v 所示。

project14.v

```
module project1
(
input              I_ad_data,
input              I_clk,
output             O_D
 );

wire    S_clk_160m;
wire    S_clk_200m;
reg     reg_DATA;
reg     reg_DATA1;

ip_pll_20m  inst_ip_pll_160_200m
     (
    .clk_in1              (I_clk),
    .clk_out1             (S_clk_160m),
    .clk_out2             (S_clk_200m),
    .reset                (1'b0),
    .locked               ( )
      );

always@(posedge S_clk_160m) begin
    reg_DATA    <=     I_ad_data;
end

always@(posedge S_clk_200m) begin
    reg_DATA1   <=     reg_DATA;
end

assign  O_D   = reg_DATA1;

endmodule
```

工程 project14 的时序逻辑结构如图 5-7 所示，主时钟为 20MHz，生成衍生时钟 160MHz 和 200MHz。

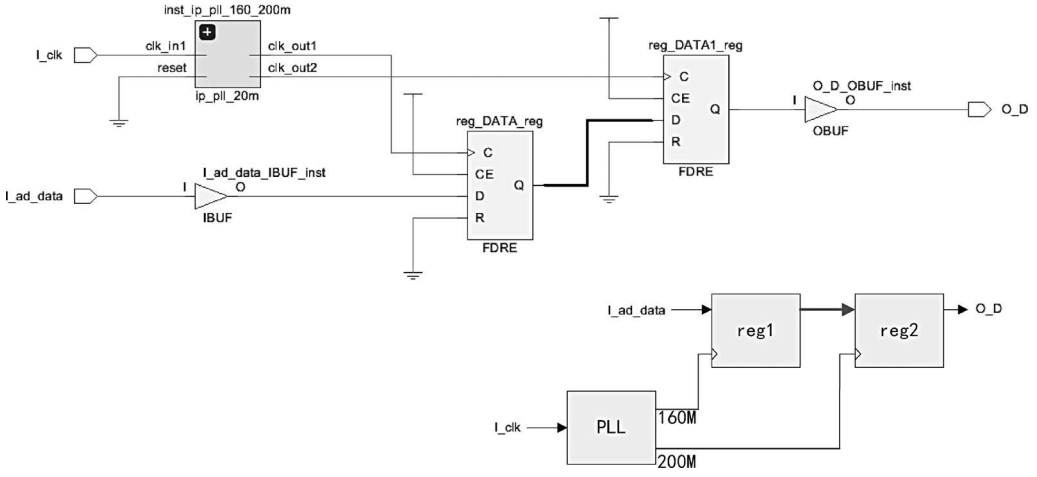


图 5-7　工程 project14 的时序逻辑结构图

为该工程进行物理引脚约束和主时钟约束，约束指令如下：

```
set_property PACKAGE_PIN G13 [get_ports I_clk]
create_clock -period 50 -name I_clk  [get_ports I_clk]
```

对工程编译、实现，生成. bit 文件，单击 IMPLEMENTATION-Open implemented Design 选项，再单击 Report Timing Summary 选项，出现 clk_out1_ip_pll_20m 到 clk_out2_ip_pll_20m(160MHz 到 200MHz)的建立时间时序违例，工程 project14 的时序违例如图 5-8 所示。

Setup 时序路径的时序分析报告如图 5-9 所示。

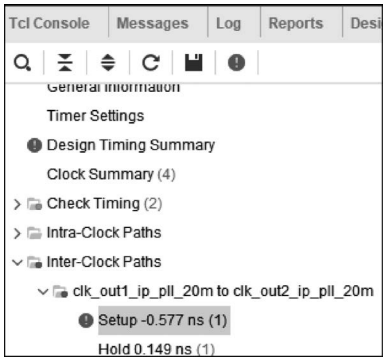


图 5-8　工程 project14 的时序违例

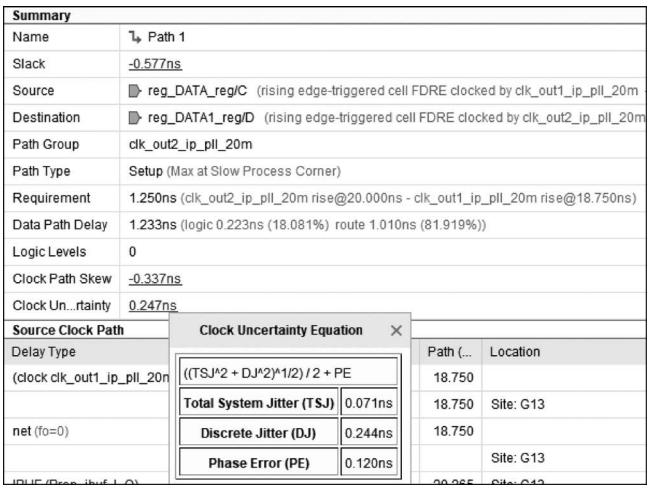
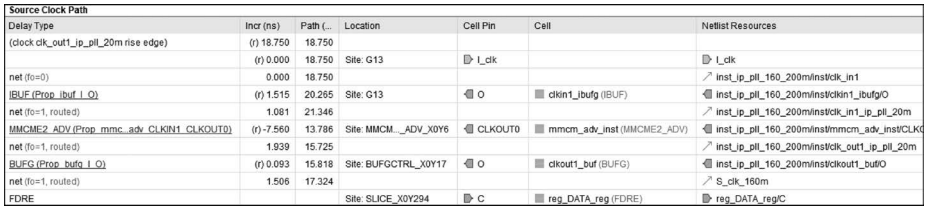


图 5-9　Setup 时序路径的时序分析报告

Setup 时序路径的源时钟路径如图 5-10 所示。

Setup 时序路径的数据路径如图 5-11 所示。

Setup 时序路径的目标时钟路径如图 5-12 所示。



Source Clock Path

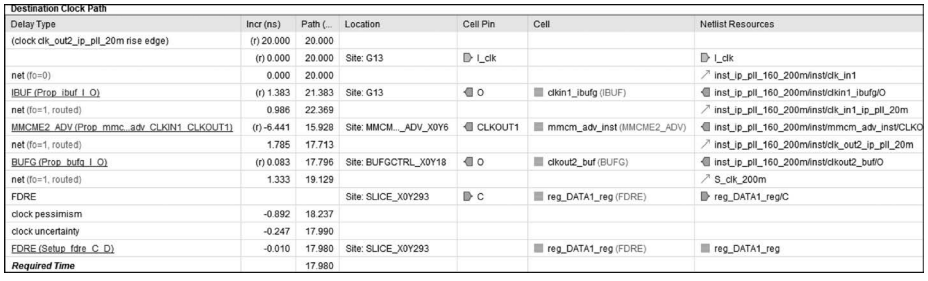
Delay Type	Incr (ns)	Path (...	Location	Cell Pin	Cell	Netlist Resources
(clock clk_out1_ip_pll_20m rise edge)	(r) 18.750	18.750				
	(r) 0.000	18.750	Site: G13	I_clk		I_clk
net (fo=0)	0.000	18.750				inst_ip_pll_160_200m/inst/clk_in1
IBUF (Prop_ibuf_I_O)	(r) 1.515	20.265	Site: G13	O	clkin1_ibufg (IBUF)	inst_ip_pll_160_200m/inst/clkin1_ibufg/O
net (fo=1, routed)	1.081	21.346				inst_ip_pll_160_200m/inst/clk_in1_ip_pll_20m
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT0)	(r) -7.560	13.786	Site: MMCM..._ADV_X0Y6	CLKOUT0	mmcm_adv_inst (MMCME2_ADV)	inst_ip_pll_160_200m/inst/mmcm_adv_inst/CLKO
net (fo=1, routed)	1.939	15.725				inst_ip_pll_160_200m/inst/clk_out1_ip_pll_20m
BUFG (Prop_bufg_I_O)	(r) 0.093	15.818	Site: BUFGCTRL_X0Y17	O	clkout1_buf (BUFG)	inst_ip_pll_160_200m/inst/clkout1_buf/O
net (fo=1, routed)	1.506	17.324				S_clk_160m
FDRE			Site: SLICE_X0Y294	C	reg_DATA_reg (FDRE)	reg_DATA_reg/C

图 5-10　Setup 时序路径的源时钟路径

Data Path

Delay Type	Incr (ns)	Path (...	Location	Cell...	Cell	Netlist Resources
FDRE (Prop_fdre_C_Q)	(r) 0.223	17.547	Site: SLICE_X0Y294	Q	reg_DATA_reg (FDRE)	reg_DATA_reg/Q
net (fo=1, routed)	1.010	18.558				reg_DATA
FDRE			Site: SLICE_X0Y293	D	reg_DATA1_reg (FDRE)	reg_DATA1_reg/D
Arrival Time		18.558				

图 5-11　Setup 时序路径的数据路径



Destination Clock Path

Delay Type	Incr (ns)	Path (...	Location	Cell Pin	Cell	Netlist Resources
(clock clk_out2_ip_pll_20m rise edge)	(r) 20.000	20.000				
	(r) 0.000	20.000	Site: G13	I_clk		I_clk
net (fo=0)	0.000	20.000				inst_ip_pll_160_200m/inst/clk_in1
IBUF (Prop_ibuf_I_O)	(r) 1.383	21.383	Site: G13	O	clkin1_ibufg (IBUF)	inst_ip_pll_160_200m/inst/clkin1_ibufg/O
net (fo=1, routed)	0.986	22.369				inst_ip_pll_160_200m/inst/clk_in1_ip_pll_20m
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT1)	(r) -6.441	15.928	Site: MMCM..._ADV_X0Y6	CLKOUT1	mmcm_adv_inst (MMCME2_ADV)	inst_ip_pll_160_200m/inst/mmcm_adv_inst/CLKO
net (fo=1, routed)	1.785	17.713				inst_ip_pll_160_200m/inst/clk_out2_ip_pll_20m
BUFG (Prop_bufg_I_O)	(r) 0.083	17.796	Site: BUFGCTRL_X0Y18	O	clkout2_buf (BUFG)	inst_ip_pll_160_200m/inst/clkout2_buf/O
net (fo=1, routed)	1.333	19.129				S_clk_200m
FDRE			Site: SLICE_X0Y293	C	reg_DATA1_reg (FDRE)	reg_DATA1_reg/C
clock pessimism	-0.892	18.237				
clock uncertainty	-0.247	17.990				
FDRE (Setup_fdre_C_D)	-0.010	17.980	Site: SLICE_X0Y293		reg_DATA1_reg (FDRE)	reg_DATA1_reg
Required Time		17.980				

图 5-12　Setup 时序路径的目标时钟路径

将时序报告中的延时数据绘制为时序图，建立时间分析时序路径和时序图，如图 5-13 所示。

建立时间时序路径报告 Summary 窗口中时序余量为负，出现时序违例。时钟路径为寄存器 reg_DATA 的时钟端口到 reg_DATA1 的数据输入端口。最糟糕的时序关系是启动沿为 160M@18.750ns，锁存沿为 200M@20.000ns，数据需求时间仅有 1.250ns，启动沿和锁存沿时序非常紧张。

时钟偏斜＝目标时钟延时－源时钟延时＝(18.237ns－20ns)－(17.324ns－18.750ns)

＝－0.337ns

由于引入了 PLL，时钟不确定性新增了离散时钟抖动 DJ 和相位差 PE。

在时序图 5-13 中，相同路径源时钟延时＞目标时钟延时，但是源时钟在 PLL 上有一个更大的“负”延时值，为了补偿这些延时，目标时钟补偿 CPR 为“负”。或者说，源时钟延时减多了，为了对齐补偿，让目标时钟再减去一些延时。

复制 project14，并命名为 project15，为该时序路径添加伪路径约束：

```
set_false_path   -from [get_clocks clk_out1_ip_pll_20m]
-to [get_clocks clk_out2_ip_pll_20m]
```

编译后的路径时序报告如图 5-14 所示，建立时间分析和保持时间分析均无信息。

配置时钟伪路径时序报告如图 5-15 所示，打开时序例外报告，可以看到 clk_out1_ip_pll_20m 到 clk_out2_ip_pll_20m 的伪路径出现在报告中，建立时间 Setup 和保持时间 Hold

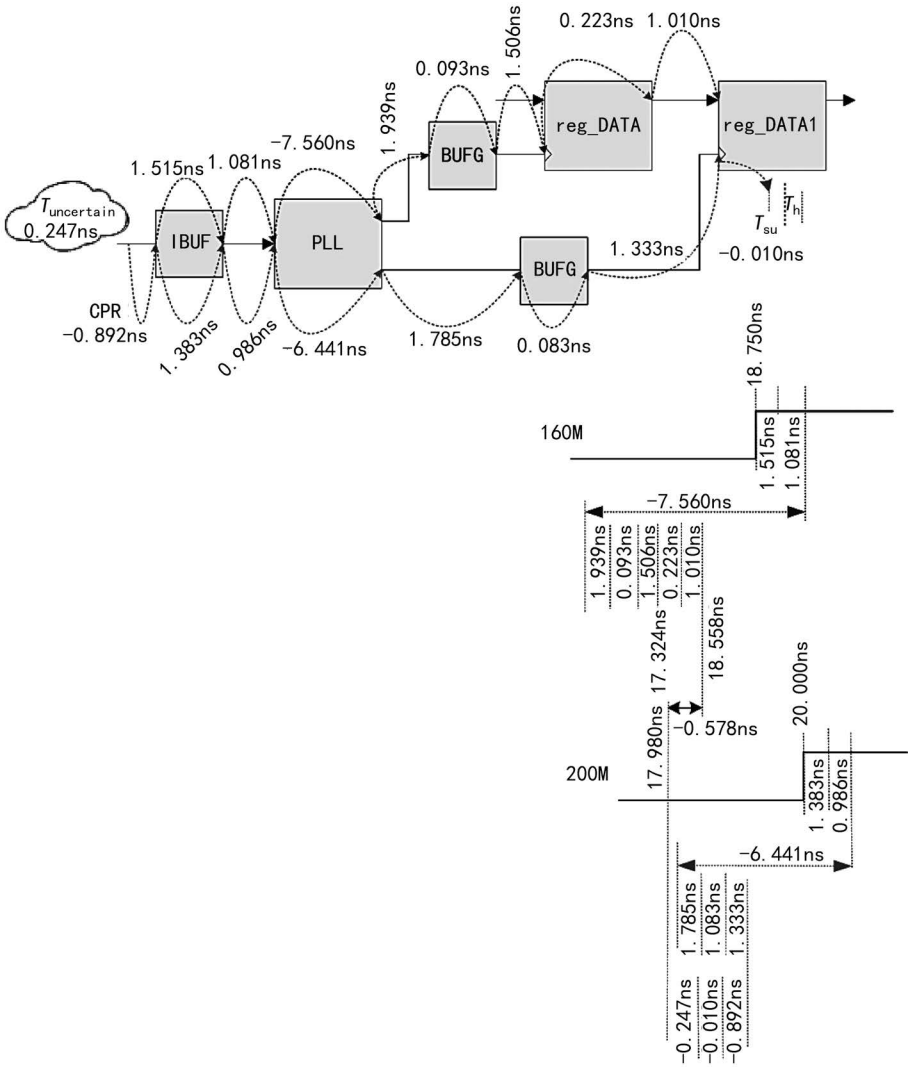


图 5-13　建立时间分析时序路径和时序图

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	NA	Worst Hold Slack (WHS):	NA	Worst Pulse Width Slack (WPWS):	2.150 ns
Total Negative Slack (TNS):	NA	Total Hold Slack (THS):	NA	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	NA	Number of Failing Endpoints:	NA	Number of Failing Endpoints:	0
Total Number of Endpoints:	NA	Total Number of Endpoints:	NA	Total Number of Endpoints:	10

All user specified timing constraints are met.

图 5-14　路径时序报告

均设置为 false。

上述约束是设置两个时钟之间的伪路径，也可以利用 pin 节点设置伪路径：

```
set_false_path   -from [get_pins reg_DATA_reg/C]
-to [get_pins reg_DATA1_reg/D]
```

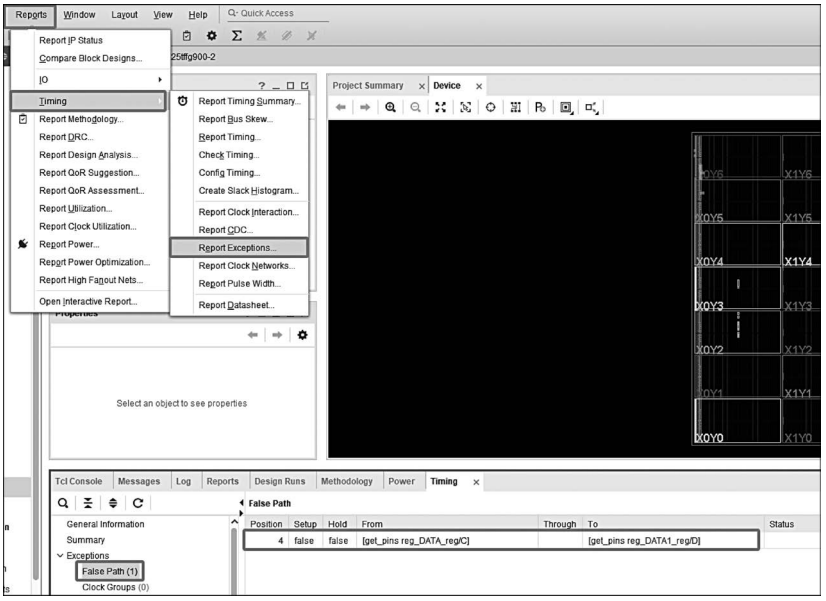


图 5-15　配置时钟伪路径时序报告

编译后的时序报告与图 5-14 一致，配置节点伪路径时序报告如图 5-16 所示。可以看到 reg_DATA_reg/C 到 reg_DATA1_reg/D 的伪路径出现在报告中，建立时间 Setup 和保持时间 Hold 均设置为 false。

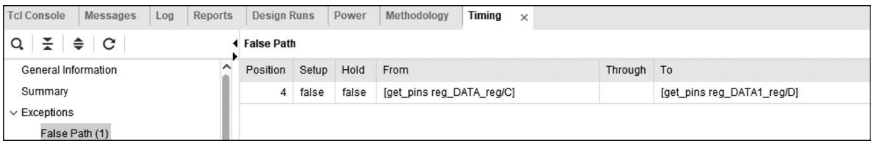


图 5-16　配置节点伪路径时序报告

5.2.3　时钟组约束语法

Vivado 默认分析所有时钟间的时序路径，通过 set_clock_groups 指令约束不同的时钟组，当源时钟和目标时钟属于同一个时钟组时，才会分析此时序路径；当源时钟和目标时钟属于不同时钟组时，则会略过此时序路径的分析。set_clock_groups 指令的语法结构如下：

```
set_clock_groups -asynchronous   -logically_exclusive   -physically_exclusive
-group {ClkA} -group {ClkB}
```

在 Vivado 中，set_clock_groups 约束指令的参数定义如表 5-2 所示。

表 5-2　set_clock_groups 约束指令的参数定义

参　数	说　明
-asynchronous	约束为异步时钟组
-logically_exclusive	约束为逻辑互斥的时钟组
-physically_exclusiv	约束为物理线路互斥的时钟组，设计中不能同时存在
-group	定义时钟组，时钟组内的时钟之间才会进行时序分析

Case1：假设有 ClkA、ClkB、ClkC 和 ClkD 四个时钟设置约束。

```
set_clock_groups -group {ClkA ClkC} -group {ClkB ClkD}
```

该约束中，ClkA 和 ClkC 之间的路径需要时序分析，ClkB 和 ClkD 之间的路径需要时序分析，ClkA- ClkB、ClkA- ClkD、ClkC- ClkB 和 ClkC- ClkD 之间的路径不需要分析。

Case2：假设有 ClkA、ClkB、ClkC、ClkD、ClkE 和 ClkF 六个时钟设置约束。

```
set_clock_groups -group {ClkA ClkC}
```

该约束中，ClkA 和 ClkC 之间的路径需要时序分析，ClkB、ClkD、ClkE 和 ClkF 彼此之间的路径需要时序分析，ClkA 到 ClkB、ClkD、ClkE 和 ClkF 的路径不需要分析，ClkC 到 ClkB、ClkD、ClkE 和 ClkF 的路径不需要分析。换句话说：

```
set_clock_groups -group {ClkA ClkC} =
set_clock_groups -group {ClkA ClkC} -group {ClkB ClkD ClkE ClkF}
```

5.2.4 时钟组约束实例

复制 project14，并命名为 project16，为该时序工程添加时钟组约束：

```
set_clock_groups -asynchronous -group [get_clocks -of_objects [get_pins inst_ip_pll_160_
200m/inst/mmcm_adv_inst/CLKOUT0]] -group [get_clocks -of_objects [get_pins inst_ip_pll_
160_200m/inst/mmcm_adv_inst/CLKOUT1]]
```

编译后的时序报告与图 5-14 一致，时钟组约束时序报告如图 5-17 所示。报告给出了 clk_out1_ip_pll_20m 到 clk_out2_ip_pll_20m 的时钟组，也给出了 clk_out2_ip_pll_20m 到 clk_out1_ip_pll_20m 的时钟组，只是该路径无实际的物理路径。

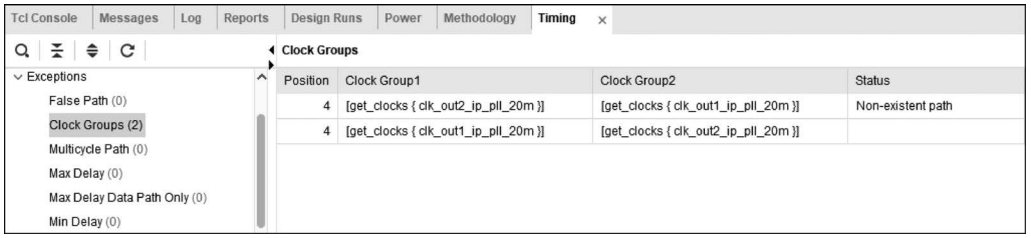


图 5-17 时钟组约束时序报告

5.3 最大/最小延时约束

最大/最小延时约束主要是对异步信号之间的时序路径进行时序约束。最大延时约束(set_max_delay)将默认覆盖建立时间分析中的最大路径延时；最小延时约束(set_min_delay)将默认覆盖保持时间分析中的最小路径延时。

覆盖是什么意思？

最大/最小延时约束如图 5-18 所示，异步跨时钟域传递，使用最大/最小延时约束后，就不考虑源时钟与目标时钟的建立/保持时间关系，而是考虑源时钟与最大/最小延时的建

立/保持时间关系。数据到达时间卡在最大/最小延时之间，相当于最大/最小延时成为建立/保持时间的锁存沿。

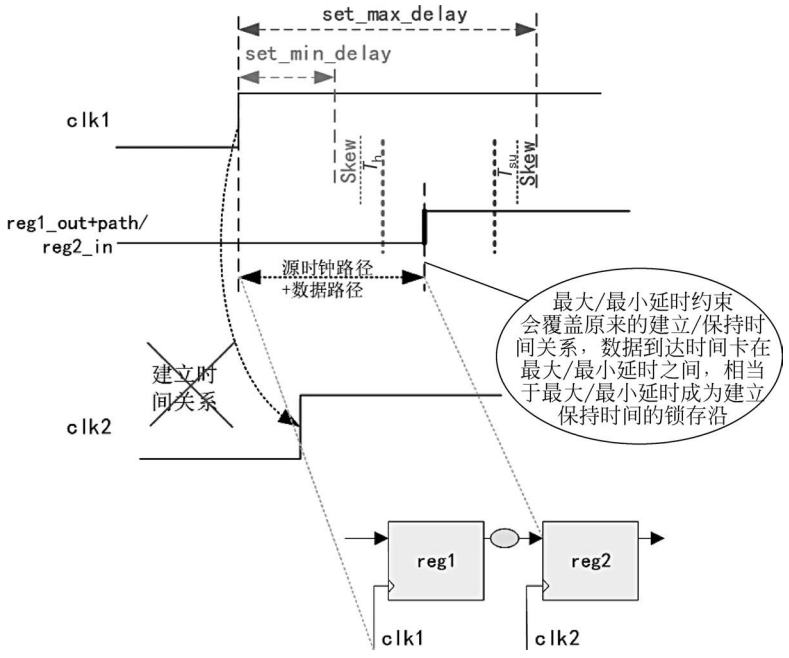


图 5-18　最大/最小延时约束示意图

图 5-18 中默认引入了源时钟和目标时钟的时钟偏斜。请关注指令中的-datapath_only选项，使用-datapath_only 选项可以将时钟偏斜移除，配置 datapath_only 后最大/最小延时约束如图 5-19 所示。

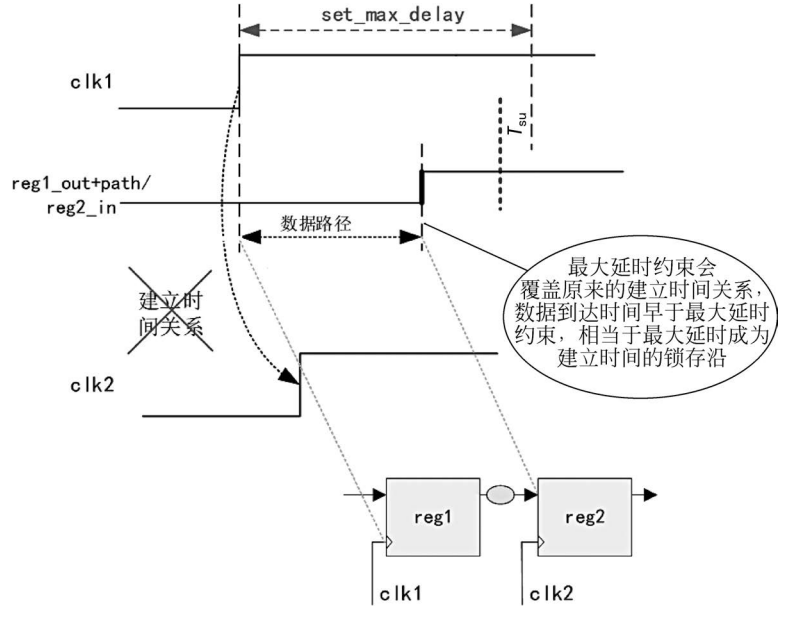


图 5-19　配置 datapath_only 后最大/最小延时约束示意图

5.3.1 最大/最小延时约束语法

以 Vivado 为例，set_max_delay/set_min_delay 指令的语法结构如下：

```
set_max_delay   -datapath_only
                -from     <node list>
                -to       <node list>
                -through  <node list>
                Delay_Value
set_min_delay
                -from     <node list>
                -to       <node list>
                -through  <node list>
                Delay_Value
```

在 Vivado 中，set_max_delay/set_min_delay 约束指令的参数定义如表 5-3 所示。

表 5-3 set_max_delay/set_min_delay 约束指令的参数定义

参　数	说　明
-datapath_only	忽略时序路径中的时钟偏斜，只能用在包含 from 选项的 set_max_delay 指令中
-from	指定约束路径的起点，可以是一个列表
-to	指定约束路径的终点，可以是一个列表
-through	指定途经点，可以是一个列表
Delay_Value	延时值，单位 ns

图 5-18 中，时序分析其实就是“set_min_delay＋目标时钟偏斜＋T_h＜源时钟偏斜＋数据路径延时＝数据到达时间＜set_max_delay－目标时钟偏斜－T_{su}”；图 5-19 中，使用 -datapath_only 后忽略时钟偏斜，变为“set_min_delay＋T_h＜数据路径延时＜set_max_delay－T_{su}”；跨时钟域且不考虑时钟偏斜时，数据路径延时小于最大延时即可，set_min_delay＋T_h＜数据路径延时会被忽略；使用 set_max_delay -datapath_only -from 指令会自动忽略该路径的 Hold 分析，相当于 set_false_path -hold，哪怕有 set_min_delay 约束指令，也会被忽略。

5.3.2 最大/最小延时约束实例

复制 project14，并命名为 project17，为该时序工程添加最大/最小延时约束：

```
set_max_delay -from [get_pins reg_DATA_reg/C] -to [get_pins reg_DATA1_reg/D] 10
set_min_delay -from [get_pins reg_DATA_reg/C] -to [get_pins reg_DATA1_reg/D] 1
```

编译后，最大/最小延时约束时序报告如图 5-20 所示，该跨时钟域路径上建立时间分析和保持时间分析均未违例。这里并未使用-datapath_only 选项，因此保持时间分析时对应最小延时约束值。

最大/最小延时约束 Setup 时序报告如图 5-21 所示。

最大/最小延时约束 Setup 时序报告的源时钟路径如图 5-22 所示。

最大/最小延时约束 Setup 时序报告的数据路径如图 5-23 所示。

最大/最小延时约束 Setup 时序报告的目标时钟路径如图 5-24 所示。

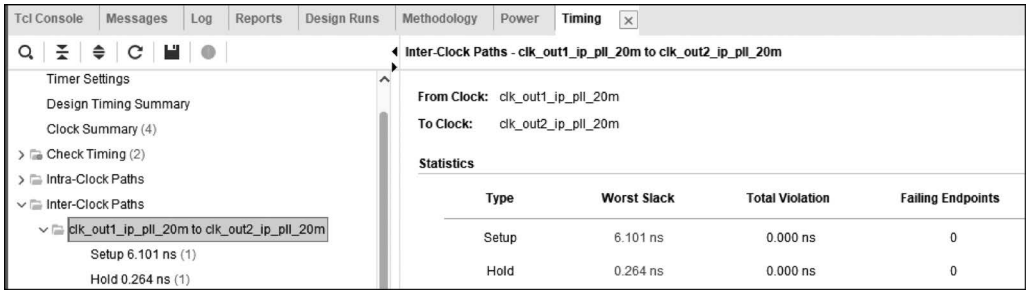


图 5-20　最大/最小延时约束时序报告

Summary	
Name	Path 1
Slack	6.101ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {r
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m
Path Group	clk_out2_ip_pll_20m
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (MaxDelay Path 10.000ns)
Data Path Delay	3.293ns (logic 0.223ns (6.773%) route 3.070ns (93.227%))
Logic Levels	0
Clock Path Skew	-0.337ns
Clock Uncertainty	0.247ns
Timing Exception	MaxDelay Path 10.000ns

图 5-21　最大/最小延时约束 Setup 时序报告

Source Clock Path			
Delay Type	Incr (ns)	Path (...	Location
(clock clk_out1_ip_pll_20m rise edge)	(r) 0.000	0.000	
	(r) 0.000	0.000	Site: G13
net (fo=0)	0.000	0.000	
			Site: G13
IBUF (Prop_ibuf_I_O)	(r) 1.515	1.515	Site: G13
net (fo=1, routed)	1.081	2.596	
			Site: MMCM..._ADV_X0Y6
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT0)	(r) -7.560	-4.964	Site: MMCM..._ADV_X0Y6
net (fo=1, routed)	1.939	-3.025	
			Site: BUFGCTRL_X0Y17
BUFG (Prop_bufg_I_O)	(r) 0.093	-2.932	Site: BUFGCTRL_X0Y17
net (fo=1, routed)	1.506	-1.426	
FDRE			Site: SLICE_X0Y294

图 5-22　最大/最小延时约束 Setup 时序报告的源时钟路径

Data Path				
Delay Type	Incr (ns)	Path (...	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(r) 0.223	-1.203	Site: SLICE_X0Y294	reg_DATA_reg/Q
net (fo=1, routed)	3.070	1.867		reg_DATA
FDRE			Site: SLICE_X0Y292	reg_DATA1_reg/D
Arrival Time		1.867		

图 5-23　最大/最小延时约束 Setup 时序报告的数据路径

将时序报告中的延时数据绘制到时序图中，最大/最小延时约束 Setup 时序分析如图 5-25 所示。

Destination Clock Path			
Delay Type	Incr (ns)	Path (...	Location
max delay	10.000	10.000	
	(r) 0.000	10.000	Site: G13
net (fo=0)	0.000	10.000	
			Site: G13
IBUF (Prop_ibuf_I_O)	(r) 1.383	11.383	Site: G13
net (fo=1, routed)	0.986	12.369	
			Site: MMCM..._ADV_X0Y6
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT1)	(r) -6.441	5.928	Site: MMCM..._ADV_X0Y6
net (fo=1, routed)	1.785	7.713	
			Site: BUFGCTRL_X0Y18
BUFG (Prop_bufg_I_O)	(r) 0.083	7.796	Site: BUFGCTRL_X0Y18
net (fo=1, routed)	1.333	9.129	
FDRE			Site: SLICE_X0Y292
clock pessimism	-0.892	8.237	
clock uncertainty	-0.247	7.990	
FDRE (Setup_fdre_C_D)	-0.022	7.968	Site: SLICE_X0Y292
Required Time		7.968	

图 5-24　最大/最小延时约束 Setup 时序报告的目标时钟路径

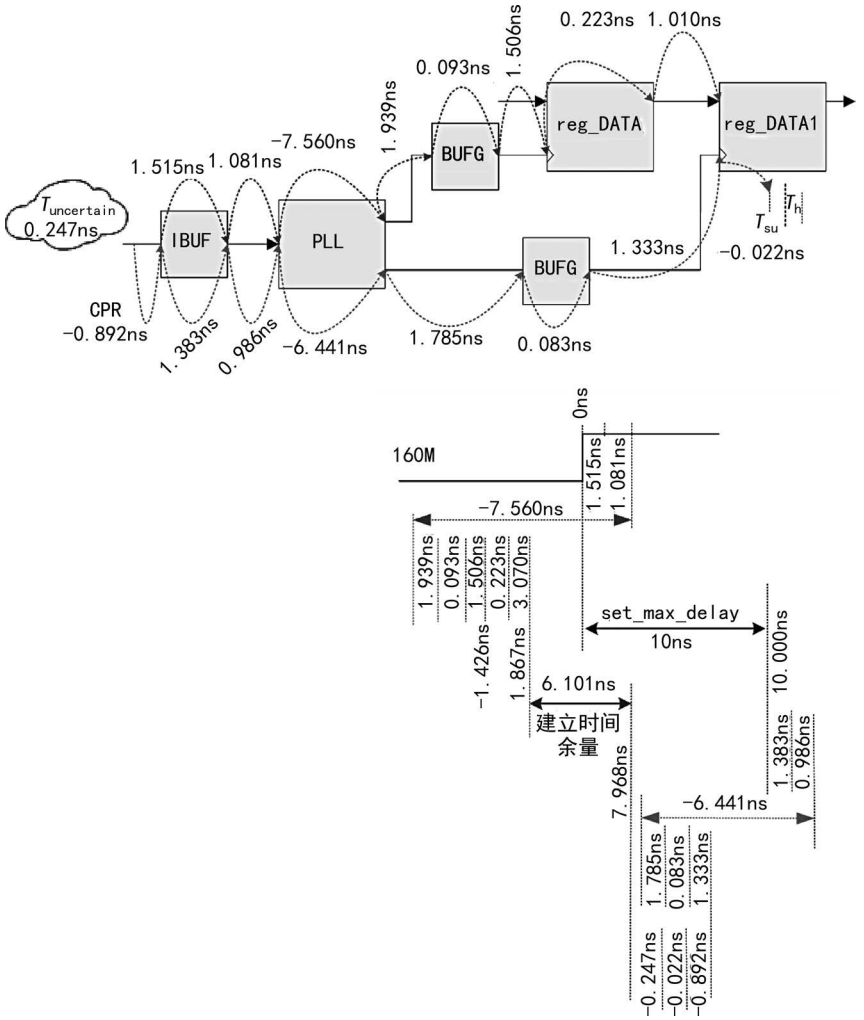


图 5-25　最大/最小延时约束 Setup 时序分析示意图

图 5-21 中，最大/最小延时约束后该路径的建立时间余量＞0，满足时序要求；建立时间分析基于最大的 Slow Process Corner；Requirement 时间为 set_max_delay 约束的 10ns，“覆盖”目标寄存器时钟的锁存沿；数据路径延时如图 5-23 所示，0.223ns＋3.070ns＝3.293ns；时钟偏斜＝(8.237ns－10ns)－(－1.426ns－0ns)＝－0.337ns；时序报告也标明了该路径为时序例外最大延时路径 10.000ns。

图 5-22 中，最大/最小延时约束启动沿为源时钟的 0ns，图 5-24 中，目标时钟路径锁存沿为 max_delay 10ns，而不是目标寄存器时钟。图 5-25 为该路径最大/最小延时约束后的建立时间分析示意图，与图 5-18 的描述一致。

最大/最小延时约束 Hold 时序报告如图 5-26 所示。

Summary

Name	Path 2
Slack (Hold)	0.264ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m
Path Group	clk_out2_ip_pll_20m
Path Type	Hold (Min at Fast Process Corner)
Requirement	1.000ns (MinDelay Path 1.000ns)
Data Path Delay	1.848ns (logic 0.100ns (5.412%) route 1.748ns (94.588%))
Logic Levels	0
Clock Path Skew	0.297ns
Clock Uncertainty	0.247ns
Timing Exception	MinDelay Path 1.000ns

图 5-26　最大/最小延时约束 Hold 时序报告

最大/最小延时约束 Hold 时序报告的源时钟路径如图 5-27 所示。

Source Clock Path

Delay Type	Incr (ns)	Path (...	Location
(clock clk_out1_ip_pll_20m rise edge)	(r) 0.000	0.000	
	(r) 0.000	0.000	Site: G13
net (fo=0)	0.000	0.000	
			Site: G13
IBUF (Prop_ibuf_I_O)	(r) 0.440	0.440	Site: G13
net (fo=1, routed)	0.503	0.943	
			Site: MMCM..._ADV_X0Y6
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT0)	(r) -3.108	-2.165	Site: MMCM..._ADV_X0Y6
net (fo=1, routed)	0.968	-1.197	
			Site: BUFGCTRL_X0Y17
BUFG (Prop_bufg_I_O)	(r) 0.026	-1.171	Site: BUFGCTRL_X0Y17
net (fo=1, routed)	0.688	-0.483	
FDRE			Site: SLICE_X0Y294

图 5-27　最大/最小延时约束 Hold 时序报告的源时钟路径

最大/最小延时约束 Hold 时序报告的数据路径如图 5-28 所示。

最大/最小延时约束 Hold 时序报告的目标时钟路径如图 5-29 所示。

将时序报告中的延时数据绘制到时序图中，最大/最小延时约束 Hold 时序分析如图 5-30 所示。

图 5-26 中，最大/最小延时约束后该路径的保持时间余量＞0，满足时序要求；保持时间分析基于最小的 Fast Process Corner；Requirement 时间为 set_min_delay 约束的 1ns，

Data Path				
Delay Type	Incr (ns)	Path (...	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(r) 0.100	-0.383	Site: SLICE_X0Y294	reg_DATA_reg/Q
net (fo=1, routed)	1.748	1.365		reg_DATA
FDRE			Site: SLICE_X0Y292	reg_DATA1_reg/D
Arrival Time		1.365		

图 5-28　最大/最小延时约束 Hold 时序报告的数据路径

Destination Clock Path			
Delay Type	Incr (ns)	Path (...	Location
min delay	1.000	1.000	
	(r) 0.000	1.000	Site: G13
net (fo=0)	0.000	1.000	
			Site: G13
IBUF (Prop_ibuf_I_O)	(r) 0.636	1.636	Site: G13
net (fo=1, routed)	0.553	2.189	
			Site: MMCM..._ADV_X0Y6
MMCME2_ADV (Prop_mmc...adv_CLKIN1_CLKOUT1)	(r) -3.568	-1.379	Site: MMCM..._ADV_X0Y6
net (fo=1, routed)	1.037	-0.342	
			Site: BUFGCTRL_X0Y18
BUFG (Prop_bufg_I_O)	(r) 0.030	-0.312	Site: BUFGCTRL_X0Y18
net (fo=1, routed)	0.912	0.600	
FDRE			Site: SLICE_X0Y292
clock pessimism	0.214	0.814	
clock uncertainty	0.247	1.061	
FDRE (Hold_fdre_C_D)	0.040	1.101	Site: SLICE_X0Y292
Required Time		1.101	

图 5-29　最大/最小延时约束 Hold 时序报告的目标时钟路径

“覆盖”目标寄存器时钟的锁存沿；数据路径延时如图 5-28 所示，0.100ns＋1.748ns＝1.848ns；时钟偏斜＝(0.814ns－1ns)－(－0.483ns－0ns)＝0.297ns；时序报告也标明了该路径为时序例外最小延时路径 1.000ns。

图 5-27 中，最大/最小延时约束启动沿为源时钟的 0ns，图 5-29 中，目标时钟路径锁存沿为 min_delay 1ns，而不是目标寄存器时钟。图 5-30 为该路径最大/最小延时约束后的保持时间分析示意图，与图 5-18 的描述一致。

在 project17 中，为该时序路径添加最大/最小延时约束，同时添加-datapath_only 选项：

```
set_max_delay -datapath_only  -from [get_pins reg_DATA_reg/C] -to [get_pins reg_DATA1_
reg/D] 10
set_min_delay -from [get_pins reg_DATA_reg/C] -to [get_pins reg_DATA1_reg/D] 1
```

配置 datapath_only 后最大/最小延时约束时序报告如图 5-31 所示。由于使用了-datapath_only 选项，因此即便使用了 set_min_delay 约束也会被忽略，保持时间关系相当于伪路径，因此时序报告中只有建立时间分析报告，时序余量大于 0，最大/最小延时约束后无时序违例。

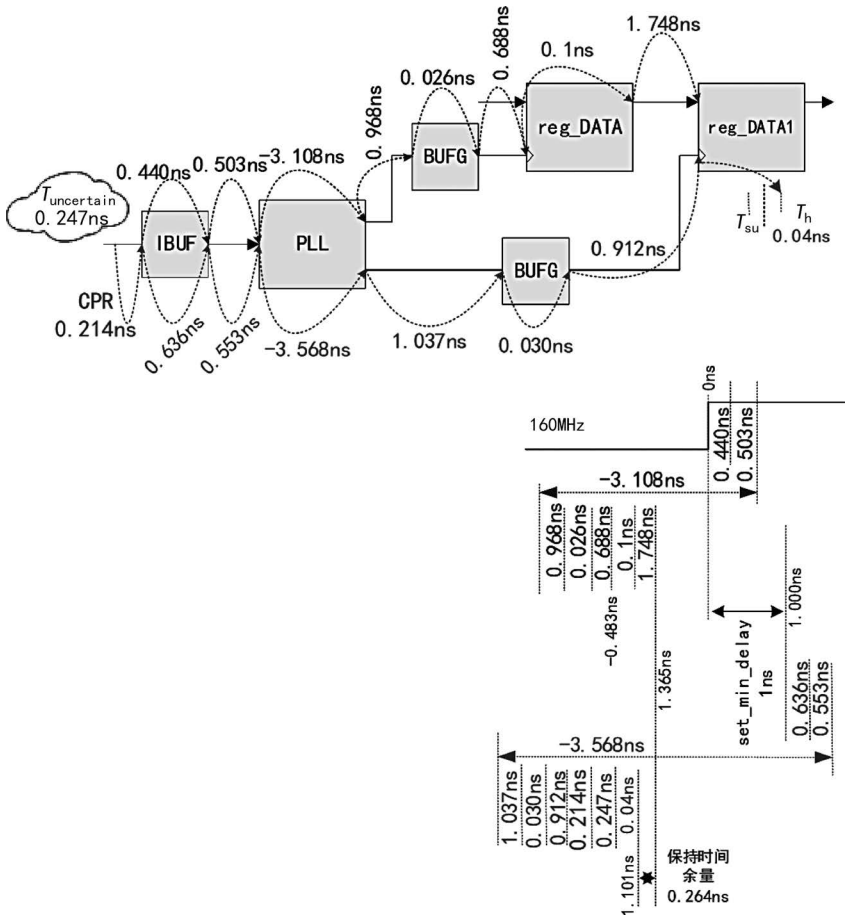


图 5-30　最大/最小延时约束 Hold 时序分析示意图

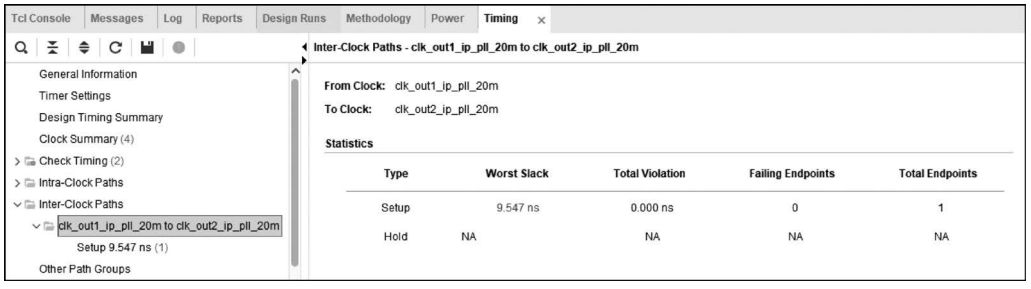


图 5-31　配置 datapath_only 后最大/最小延时约束时序报告

配置 datapath_only 后最大/最小延时约束 Setup 时序报告如图 5-32 所示。

配置 datapath_only 后最大/最小延时约束 Setup 时序报告的数据路径如图 5-33 所示。

配置 datapath_only 后最大/最小延时约束 Setup 时序报告的目标时钟路径如图 5-34 所示。

将时序报告中的延时数据绘制到时序图中，配置 datapath_only 后最大/最小延时约束 Setup 时序分析如图 5-35 所示。使用-datapath_only 选项后忽略时钟偏斜，时序报告中无源时钟路径，目标时钟路径中也仅有 T_{su}。

Summary	
Name	Path 1
Slack	9.547ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {r
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m
Path Group	clk_out2_ip_pll_20m
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (MaxDelay Path 10.000ns)
Data Path Delay	0.431ns (logic 0.223ns (51.771%) route 0.208ns (48.229%))
Logic Levels	0
Timing Exception	MaxDelay Path 10.000ns -datapath_only

图 5-32　配置 datapath_only 后最大/最小延时约束 Setup 时序报告

Data Path						
Delay Type	Incr (ns)	Path ...	Location	Cell...	Cell	Netlist Resources
	(r) 0.000	0.000	Site: S..._X0Y294	C	reg_DATA_reg (FDRE)	reg_DATA_reg/C
FDRE (Prop_fdre_C_Q)	(r) 0.223	0.223	Site: S..._X0Y294	Q	reg_DATA_reg (FDRE)	reg_DATA_reg/Q
net (fo=1, routed)	0.208	0.431				reg_DATA
FDRE			Site: S..._X0Y292	D	reg_DATA1_reg (FDRE)	reg_DATA1_reg/D
Arrival Time		0.431				

图 5-33　配置 datapath_only 后最大/最小延时约束 Setup 时序报告的数据路径

Destination Clock Path						
Delay Type	Incr (n...	Path (...	Location	Cell...	Cell	Netlist Resources
max delay	10.000	10.000				
FDRE (Set...dre_C_D)	-0.022	9.978	Site: S..._X0Y292		reg_DATA1_reg (FDRE)	reg_DATA1_reg
Required Time		9.978				

图 5-34　配置 datapath_only 后最大/最小延时约束 Setup 时序报告的目标时钟路径

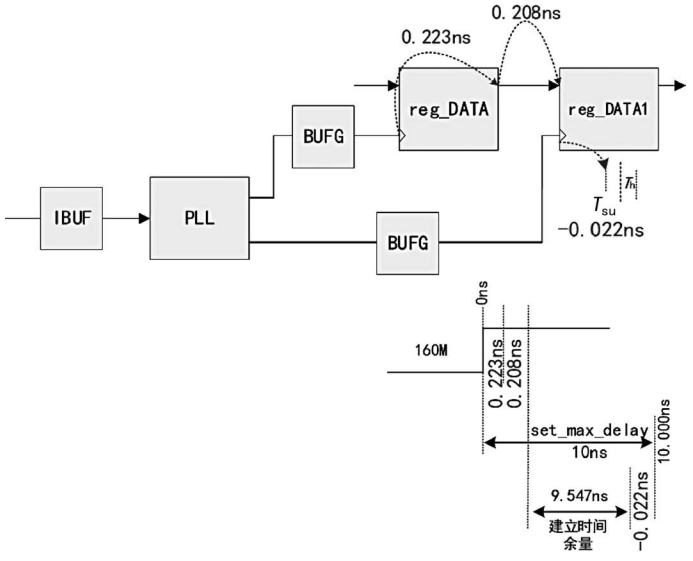


图 5-35　配置 datapath_only 后最大/最小延时约束 Setup 时序分析示意图

图 5-32 中，最大/最小延时约束后该路径的建立时间余量 9.547ns＞0，满足时序要求；建立时间分析基于最大的 Slow Process Corner；Requirement 时间为 set_max_delay 约束

的10ns，“覆盖”目标寄存器时钟的锁存沿；数据路径延时如图5-33所示，0.223ns＋0.208ns＝0.431ns；使用datapath_only后无时钟偏斜；时序报告也标明了该路径为时序例外最大延时路径10.000ns -datapath_only。

图5-33中，最大/最小延时约束启动沿为源时钟的0ns，图5-24中，目标时钟路径锁存沿为max_delay 10ns，而不是目标寄存器时钟。图5-35为该路径最大/最小延时约束后的建立时间分析示意图，未包含任何时钟偏斜，与图5-19的描述一致。

5.4　多周期路径约束

仍然以航班为例。

当航班每1日1班，N日的乘客启动$N+1$日登机，到达机场的时间为(N日＋保持时间)～($N+1$日－建立时间)。

当航班每3日1班，N日的乘客启动$N+3$日登机，到达机场的时间为(N日＋保持时间)～($N+3$日－建立时间)，早到了飞机也不飞。

当航班每1日3班，N日的乘客启动N日第三班登机，到达机场的时间为(N日＋保持时间)～(N日第3班－建立时间)，没必要非得赶前两班，第三班也是当天到。

乘客到达机场的时序要求/建立时间要求变宽松，没必要非得赶在最小周期内到达。

将这个例子映射到FPGA中，寄存器之间传递数据，源寄存器数据更新周期与目标寄存器更新周期不一致，可以使用多周期路径约束放宽该路径的建立时间要求，保持时间约束不变；把有限的布局布线资源分配给频率更高、时序更紧的路径，优化整个系统的时序。当然，多周期路径约束也需要与逻辑设计对应，才能保证数据按设计期望传递。

5.4.1　多周期路径约束语法

以Vivado为例，使用set_multicycle_path指令定义多周期路径约束，该指令用于更改建立时间和保持时间分析时源时钟启动沿与目标时钟锁存沿的相对位置关系，或者说改变建立时间和保持时间分析的时钟数。set_multicycle_path指令的语法结构如下：

```
set_multicycle_path <path_multiplier>
                    [-setup|-hold]
                    [-start|-end]
                    [-from ] [-to ] [-through <pins|cells|nets>]
```

在Vivado中，set_multicycle_path约束指令的参数定义如表5-4所示。

表5-4　set_multicycle_path约束指令的参数定义

参　　数	说　　明
path_multiplier	必须指定该值，表示多周期路径时序分析的时钟周期数，也就是数据到达时间应在指定值的周期之前
-setup/-hold	-setup是指该多周期路径指定建立时间分析；-hold是指该多周期路径指定保持时间分析
-start/-end	-start是指该多周期路径基于源时钟作为参考时钟；-end是指该多周期路径基于目标时钟作为参考时钟

续表

参　　数	说　　明
-from	指定多周期路径的起始节点
-to	指定多周期路径的终止节点
-through	指定多周期路径的途经节点，可选项

注：-from 和-to 可以同时指定，也可以指定其中一个，它会覆盖所有该节点开始(form)/该节点终止(end)的所有路径。

path_multiplier 是指数据从一个时钟域传输到另一个时钟域所需的时间周期数。单周期路径启动沿和锁存沿的关系如图 5-36 所示，建立时间分析时默认 path_multiplier＝1，保持时间分析时默认 path_multiplier＝0。这里有个公式：

保持时间时钟周期数＝建立时间 path_multiplier－1－保持时间 path_multiplier

因此，单周期路径中保持时间的时钟周期数＝1－1－0＝0。

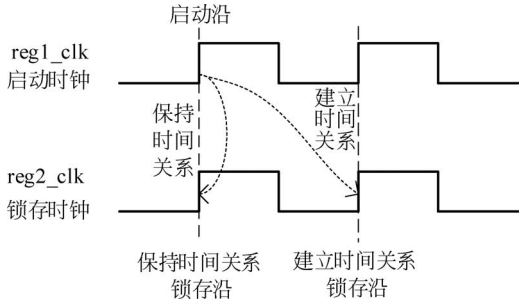


图 5-36　单周期路径启动沿和锁存沿的关系

使用 set_multicycle_path 指令定义多周期路径约束，改变的就是建立时间 path_multiplier 和保持时间 path_multiplier。换句话说，该指令用于更改建立时间和保持时间分析时源时钟启动沿与目标时钟锁存沿的相对位置关系。

set_multicycle_path 指令中，-start|-end 选项对更改源时钟启动沿和目标时钟锁存沿的影响如表 5-5 所示。

表 5-5　-start|-end 选项对更改源时钟启动沿和目标时钟锁存沿的影响

多周期路径	源时钟(-start)启动沿移动方向	目标时钟(-end)锁存沿移动方向
建立时间	左移	右移(默认)
保持时间	右移(默认)	左移

多周期路径启动沿和锁存沿移动方向如图 5-37 所示。

对比图 5-36 和图 5-37，将路径约束为多周期路径，启动沿和锁存沿需要向多周期方向扩展。

在建立时间分析时，基于源时钟(-start)作为参考时钟，需要将启动沿左移才能扩展多周期；基于目标时钟(-end)作为参考时钟，将锁存沿右移也可扩展多周期。

在保持时间分析时，基于源时钟(-start)作为参考时钟，需要将启动沿右移才能保持多个周期；基于目标时钟(-end)作为参考时钟，将锁存沿左移也可保持多周期。

建立时间分析时默认-end，保持时间分析时默认-start，源时钟和目标时钟同频同相，-start|-end 选项无差异；时钟和目标时钟非同频同相，-start|-end 选项需要指定。set_

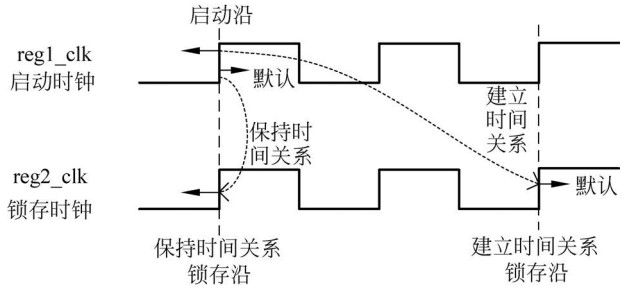


图 5-37　多周期路径启动沿和锁存沿移动方向

multicycle_path -setup 约束多周期路径时，会同时改变建立时间关系和保持时间关系，建立时间关系移动，保持时间关系也移动。一般地，会添加额外的约束-hold 维持保持时间关系。

多周期路径约束被应用到不同类别的跨时钟域路径中，多周期路径分类如图 5-38 所示。异步时钟主要传递复位、初始化、Done 信号、初始参数配置等电平/数据信号，约束相对简单。多周期路径约束需要重点分析同步时钟跨时钟，应结合具体的逻辑设计，主要包括同频同相、同频异相、慢时钟域到快时钟域、快时钟域到慢时钟域。

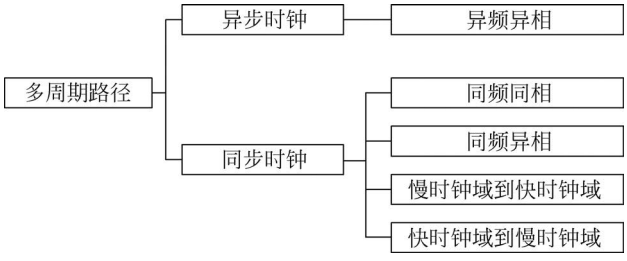


图 5-38　多周期路径分类

5.4.2　同频同相多周期路径约束

创建一个同频同相的多周期路径数据传递的工程 project18，其工程代码如 project18.v 所示。

```
project18.v

    module project1
    (
    input              I_ad_data,
    input              I_clk,
    output             O_D
     );

    //使能 en 计数
    reg [2:0] en_cnt = 3'b000;
    always@(posedge I_clk) begin
        if(en_cnt == 3'b111)
            en_cnt <= 3'b000;
        else
            en_cnt <= en_cnt + 1'b1;
    end
```

```
//寄存器 1
reg     reg_DATA;
always@(posedge I_clk) begin
    if(en_cnt == 3'b111)
        reg_DATA    <=     I_ad_data;
    else
        reg_DATA    <=     reg_DATA;
end

//寄存器 2
reg     reg_DATA1;
always@(posedge I_clk) begin
    if(en_cnt == 3'b111)
        reg_DATA1   <=     reg_DATA;
    else
        reg_DATA1   <=     reg_DATA1;
end

assign  O_D   = reg_DATA1;

endmodule
```

主时钟为 20MHz,同频同相的多周期路径时序如图 5-39 所示。

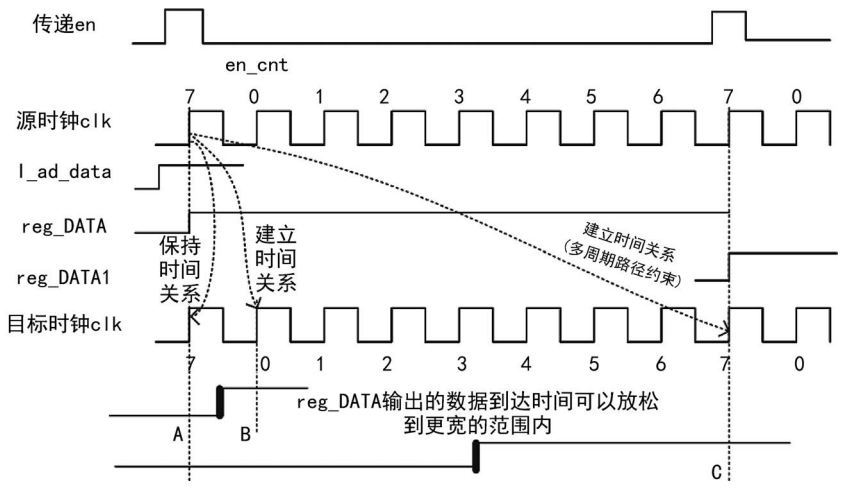


图 5-39　同频同相的多周期路径时序图

图 5-39 中,由于逻辑设计 8 个时钟周期更新一次数据,所以 reg_DATA 寄存器输出的值会锁存 8 个周期不变,之后再传递给 reg_DATA1 寄存器。特殊的应用场景中,这样的设计是没有问题的,时序图也符合设计期望,仅有使能 en(en_cnt=7)时,源寄存器和目标寄存器才更新数据。默认布局布线时,建立时间分析的需求时间仅有一个周期,也就是数据到达时间需要卡在两条虚线 A、B 之间。事实上,reg_DATA 输出的数据早到了也没用,下次使能 en(en_cnt=7)时,reg_DATA1 才锁存该输出值,也就是说 reg_DATA 输出数据到达时间早于下一次使能 en(en_cnt=7)即可,也就是两条虚线 A、C 之间。因此可以设置多周期路径约束。

在工程中添加多周期路径约束:

```
set_multicycle_path 4 -setup -from [get_pins reg_DATA_reg/C] -to [get_pins reg_DATA1_reg/
D]
set_multicycle_path 3 -hold -from [get_pins reg_DATA_reg/C] -to [get_pins reg_DATA1_reg/
D]
```

当然，可以把这里的周期数 4 或 3 设置为 8 或 7，只要适当放宽时序约束即可，实际布局布线也不会有 8 或 7 这么宽松。源时钟和目标时钟同频同相，-start|-end 选项无差异，这里未指定。

编译后的同频同相的多周期路径建立时间分析报告如图 5-40 所示。

Summary	
Name	Path 6
Slack	199.384ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by I_clk {rise@0.000ns fall@25.000ns period=50.000ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by I_clk {rise@0.000ns fall@25.000ns period=50.000ns})
Path Group	I_clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	200.000ns (I_clk rise@200.000ns - I_clk rise@0.000ns)
Data Path Delay	0.559ns (logic 0.223ns (39.900%) route 0.336ns (60.100%))
Logic Levels	0
Clock Path Skew	0.000ns
Clock Uncertainty	0.035ns
Timing Exception	MultiCycle Path Setup -end 4

图 5-40　同频同相的多周期路径建立时间分析报告

同频同相的多周期路径建立时间分析如图 5-41 所示，这里只分析多周期路径，不再分析具体的延时信息。

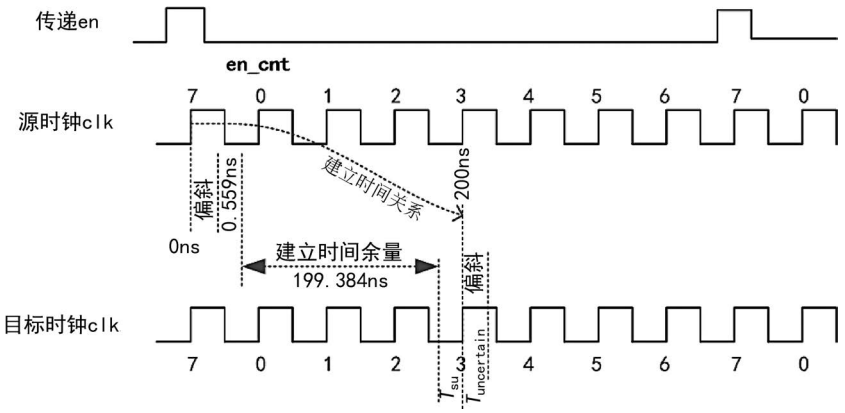


图 5-41　同频同相的多周期路径建立时间分析图

图 5-40 和图 5-41 中，建立时间余量为 199.384ns，满足时序要求；数据需求时间为 200ns，启动沿为 0ns，锁存沿为 200ns，也就是 4 个时钟周期；在时序例外窗口标明该路径为多周期路径 Setup 时序分析，周期数为 4，默认目标时钟(-end)为参考时钟。

同频同相的多周期路径保持时间分析报告如图 5-42 所示。

同频同相的多周期路径保持时间分析如图 5-43 所示。

图 5-42 和图 5-43 中，保持时间余量为 0.224ns，满足时序要求；数据需求时间为 0ns，启动沿为 150ns，锁存沿为 150ns。由于保持时间分析也受到建立时间多周期约束影响，与

Summary	
Name	Path 9
Slack (Hold)	0.224ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by I_clk {rise@0.000ns fall@25.000ns period=50.000ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by I_clk {rise@0.000ns fall@25.000ns period=50.000ns})
Path Group	I_clk
Path Type	Hold (Min at Fast Process Corner)
Requirement	0.000ns (I_clk rise@150.000ns - I_clk rise@150.000ns)
Data Path Delay	0.264ns (logic 0.100ns (37.894%) route 0.164ns (62.106%))
Logic Levels	0
Clock Path Skew	0.000ns
Timing Exception	MultiCycle Path Setup -end 4 Hold -start 3

图 5-42 同频同相的多周期路径保持时间分析报告

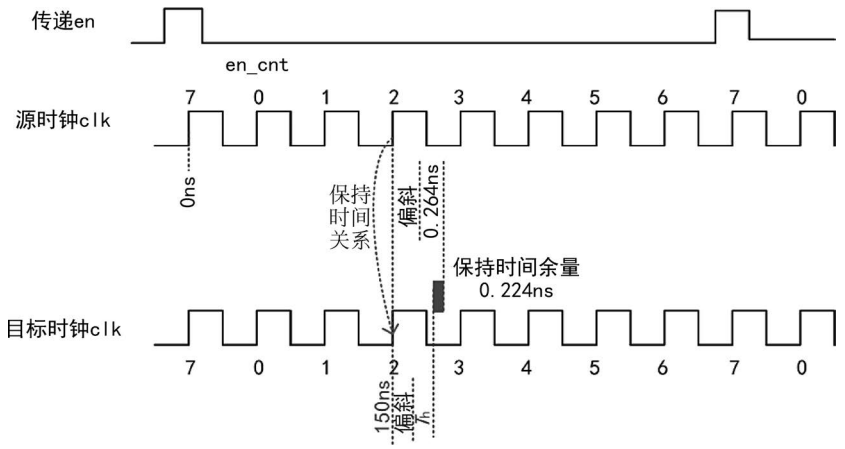


图 5-43 同频同相的多周期路径保持时间分析图

保持时间多周期约束也有关,这里标明建立时间周期数为 4,默认目标时钟(-end)为参考时钟,保持时间周期数为 3,默认源时钟(-start)为参考时钟。

以该工程为例,介绍多周期路径的约束指令是如何扩展路径的多周期的。多周期路径约束扩展多周期示意如图 5-44 所示。

图 5-44 中,未使用多周期路径约束时,建立时间需求时间为 1 个时钟周期,保持时间需求时间为 0 个时钟周期。首先,set_multicycle_path 4 -setup(默认目标时钟 end 右移)指令将建立时间和保持时间的锁存沿右移 3 个周期(建立时间本来有 1 个时钟周期),建立时间多周期约束也会影响保持时间;其次,set_multicycle_path 3 -hold(默认源时钟 start 右移)指令将保持时间的启动沿右移 3 个周期。由于同频同相时-start|-end 选项无差异,且时钟波形具有对称性,将保持时间关系左移与建立时间启动沿对齐,该波形就是多周期路径的设计期望。

5.4.3 同频异相多周期路径约束

创建一个同频异相的多周期路径数据传递的工程 project19,其工程代码如 project19.v 所示。

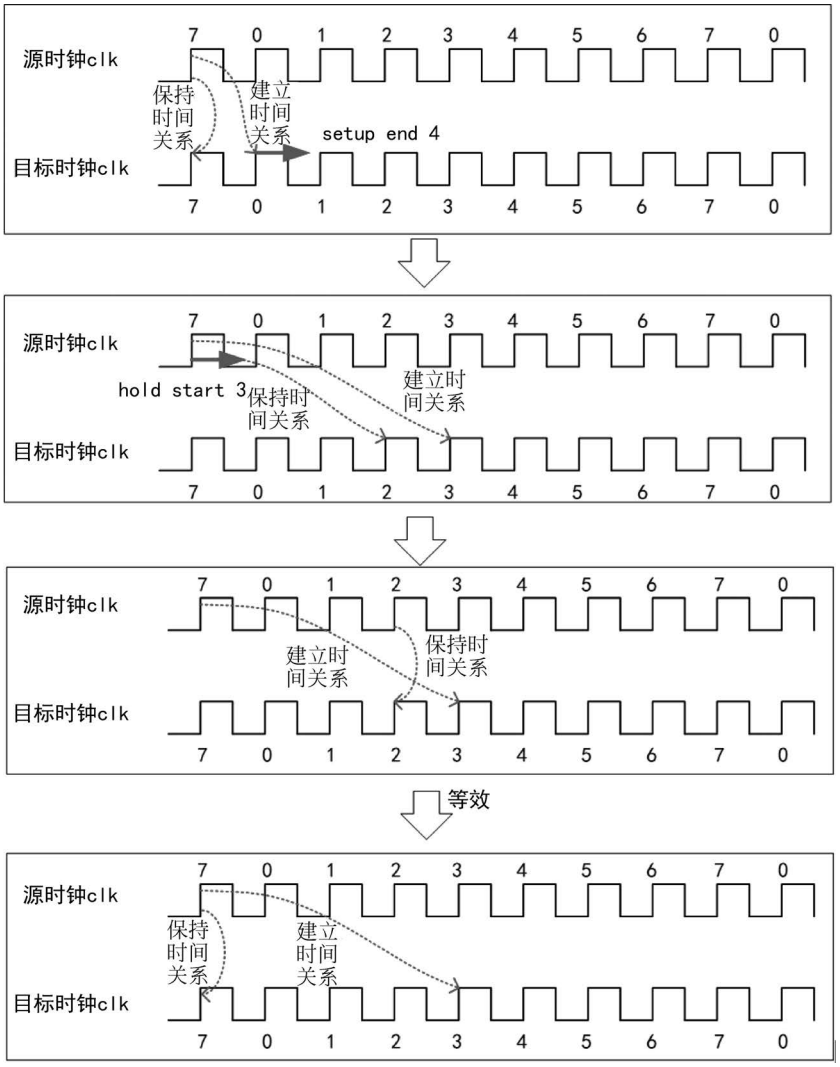


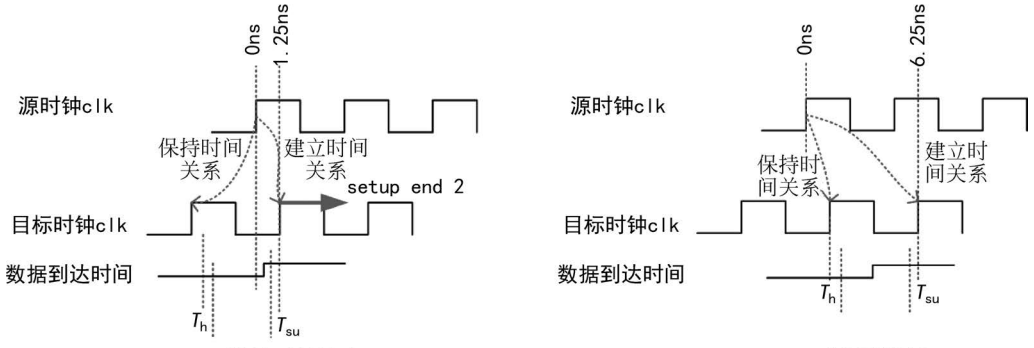
图 5-44　多周期路径约束扩展多周期示意图

project19.v

```
    module project1
    (
    input              I_ad_data,
    input              I_clk,
    output             O_D
     );

    wire     S_clk_200m;
    wire     S_clk_200m_90;
    reg      reg_DATA;
    reg      reg_DATA1;

    ip_pll_20m  inst_ip_pll_200m
         (
```

```
        .clk_in1              (I_clk),
        .clk_out1             (S_clk_200m),
        .clk_out2             (S_clk_200m_90),
        .reset                (1'b0),
        .locked               ( )
          );

    always@(posedge S_clk_200m) begin
         reg_DATA    <=     I_ad_data;
    end

    always@(posedge S_clk_200m_90) begin
         reg_DATA1   <=     reg_DATA;
    end

    assign  O_D   = reg_DATA1;

    endmodule
```

主时钟为 20MHz,衍生时钟分别为 200MHz 和 200MHz 右移 90°,两个时钟为同频异相,同频异相的多周期路径示意如图 5-45 所示。



(a) 默认时间关系　　(b) 多周期路径

图 5-45　同频异相的多周期路径示意图

图 5-45(a)中,当目标时钟的相位比源时钟晚 90°,源时钟启动沿采样的数据马上被目标时钟锁存,建立时间需求时间仅有 1.25ns,时序非常紧张;保持时间余量无时序压力,这时布局布线或许也可以成功。图 5-45(b)中,设计者可以通过多周期路径约束,将建立时间和保持时间的锁存沿向右移动一个周期,这样就可以缓解建立时间时序紧张的状态。该同频异相案例每个周期都会传递数据,数据传递晚一个周期,但不影响逻辑功能。

在工程中添加多周期路径约束:

```
set_multicycle_path 2 -setup -from [get_pins reg_DATA_reg/C] -to [get_pins reg_DATA1_reg/
D]
```

这里的周期数 2 合适且不宜继续增大,建立时间多周期路径约束默认目标时钟(end)向右,与设计期望一致。另外,建立时间多周期路径约束会同时使建立时间和保持时间的锁存沿向右移动,因此不再需要多周期路径约束调整保持时间周期。

编译后的同频异相的多周期路径建立时间分析报告如图 5-46 所示。

Summary	
Name	Path 1
Slack	2.030ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {rise@0.000ns fall@2.500ns period=5.000ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m {rise@1.250ns fall@3.750ns period=5.000ns})
Path Group	clk_out2_ip_pll_20m
Path Type	Setup (Max at Slow Process Corner)
Requirement	6.250ns (clk_out2_ip_pll_20m rise@6.250ns - clk_out1_ip_pll_20m rise@0.000ns)
Data Path Delay	3.617ns (logic 0.223ns (6.165%) route 3.394ns (93.835%))
Logic Levels	0
Clock Path Skew	-0.337ns
Clock Un...rtainty	0.244ns
Timing ...ception	MultiCycle Path Setup -end 2

图 5-46　同频异相的多周期路径建立时间分析报告

同频异相的多周期路径建立时间分析如图 5-47 所示。这里只分析多周期路径，不再分析具体的延时信息。

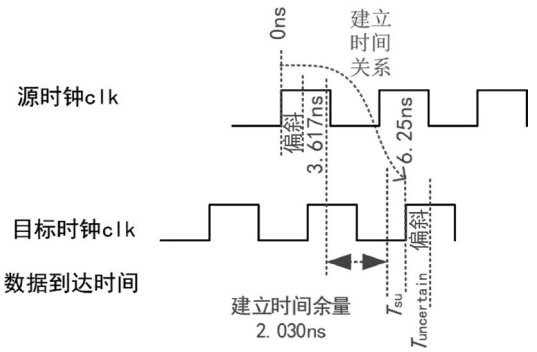


图 5-47　同频异相的多周期路径建立时间分析图

图 5-46 和图 5-47 中，建立时间余量为 2.030ns，满足时序要求；数据需求时间为 6.250ns，启动沿为 0ns，锁存沿为 6.250ns，也就是 1.25(2)个时钟周期；在时序例外窗口标明该路径为多周期路径 Setup 时序分析，周期数为 2，默认目标时钟(-end)为参考时钟。

同频异相的多周期路径保持时间分析报告如图 5-48 所示。

Summary	
Name	Path 2
Slack (Hold)	0.204ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {rise@0.000ns fall@2.500ns period=5.000ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m {rise@1.250ns fall@3.750ns period=5.000ns})
Path Group	clk_out2_ip_pll_20m
Path Type	Hold (Min at Fast Process Corner)
Requirement	1.250ns (clk_out2_ip_pll_20m rise@6.250ns - clk_out1_ip_pll_20m rise@5.000ns)
Data Path Delay	2.035ns (logic 0.100ns (4.914%) route 1.935ns (95.086%))
Logic Levels	0
Clock Path Skew	0.297ns
Clock Un...rtainty	0.244ns
Timing ...ception	MultiCycle Path Setup -end 2

图 5-48　同频异相的多周期路径保持时间分析报告

同频异相的多周期路径保持时间分析如图 5-49 所示。

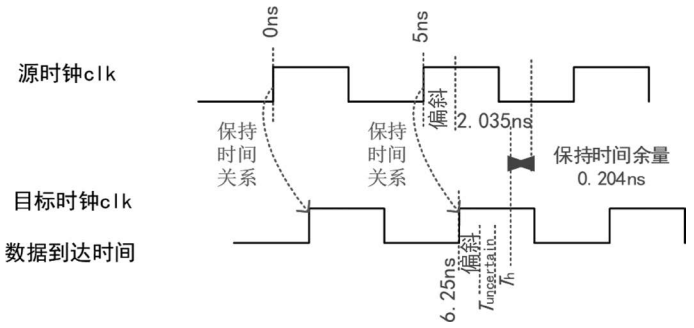


图 5-49　同频异相的多周期路径保持时间分析图

图 5-48 和图 5-49 中，保持时间余量为 0.204ns，满足时序要求；数据需求时间为 1.25ns，启动沿为 5ns，锁存沿为 6.25ns，也就是 0.25(1)个时钟周期。保持时间分析由于受建立时间多周期约束影响，这里标明建立时间周期数为 2，默认目标时钟(-end)为参考时钟。

5.4.4　慢时钟域到快时钟域多周期路径约束

创建一个慢时钟域到快时钟域的多周期路径数据传递工程 project20，其工程代码如 project20.v 所示，主时钟为 20MHz，衍生时钟分别为 200MHz 和 400MHz，数据由 200MHz 时钟域传递到 400MHz 时钟域。

```
project20.v

    module project1
    (
    input             I_ad_data,
    input             I_clk,
    output            O_D
     );

    wire    S_clk_200m;
    wire    S_clk_400m;


    ip_pll_20m  inst_ip_pll_200m
        (
       .clk_in1              (I_clk),
       .clk_out1             (S_clk_200m),
       .clk_out2             (S_clk_400m),
       .reset                (1'b0),
       .locked               ( )
         );


    reg     vld;
    always@(posedge S_clk_200m) begin
        vld         <=     1'b1;
    end

    reg  [1:0]   en_cnt;
```

```
always@(posedge S_clk_400m) begin
    if(vld)
         begin
         if(en_cnt == 2'b01)  // 2 3 4
              en_cnt <= 2'b00;
         else
              en_cnt <= en_cnt + 1'b1;
         end
    else
         en_cnt  <=  2'b00;
end


reg     reg_DATA;
always@(posedge S_clk_200m) begin
    reg_DATA    <=     I_ad_data;
end

reg     reg_DATA1;
always@(posedge S_clk_400m) begin
     if(en_cnt   ==   2'b01)
          reg_DATA1   <=     reg_DATA;
    else
          reg_DATA1   <=     reg_DATA1;
end

assign  O_D   = reg_DATA1;

endmodule
```

慢时钟域到快时钟域的多周期路径时序如图 5-50 所示。

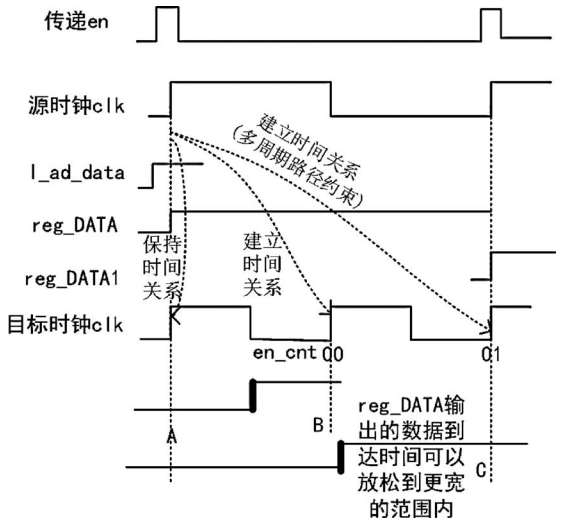


图 5-50　慢时钟域到快时钟域的多周期路径时序图

图 5-50 中，依据逻辑设计，慢时钟一个周期(5ns)更新一次数据，快时钟 2 个时钟周期(2×2.5ns)更新一次数据(en_cnt=01)，所以 reg_DATA 寄存器输出的值会锁存 5ns 不变，之后传递给 reg_DATA1 寄存器，时序图与逻辑设计的期望一致。仅有使能 en 时，也就是 2

个目标时钟周期，目标寄存器才更新数据。默认布局布线时，建立时间分析的需求时间仅有 1 个目标时钟周期(en_cnt＝00)，也就是数据到达时间需要卡在两条虚线 A、B 之间。事实上，reg_DATA 输出的数据早到了也没用，下次使能 en(en_cnt＝01)时，reg_DATA1 才锁存该输出值。也就是说，reg_DATA 输出数据到达时间早于下一次使能 en(en_cnt＝01)即可，也就是两条虚线 A、C 之间。因此可以设置多周期路径约束。

有没有发现，慢时钟域到快时钟域的时序和同频同相的时序相似？

在工程中添加多周期路径约束：

```
set_multicycle_path 2   -end   -setup -from [get_pins reg_DATA_reg/C] -to [get_pins reg_
DATA1_reg/D]
set_multicycle_path 1   -end   -hold -from [get_pins reg_DATA_reg/C] -to [get_pins reg_
DATA1_reg/D]
```

这里的周期数 2 或 1 是由逻辑功能决定的，功能决定数据到达时间只能在 2 个周期之内；约束指令中，周期数 2 或 1 是基于目标时钟，指令中均添加 end 选项。保持时间多周期路径受到建立时间多周期路径影响，需要添加 set_multicycle_path 1 -end -hold 指令维持保持时间关系不变。

编译后的慢时钟域到快时钟域的多周期路径建立时间分析报告如图 5-51 所示。

Summary

Name	Path 5
Slack	2.998ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {rise@0.000ns fall@2.500ns period=5.000ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m {rise@0.000ns fall@1.250ns period=2.500ns})
Path Group	clk_out2_ip_pll_20m
Path Type	Setup (Max at Slow Process Corner)
Requirement	5.000ns (clk_out2_ip_pll_20m rise@5.000ns - clk_out1_ip_pll_20m rise@0.000ns)
Data Path Delay	1.377ns (logic 0.223ns (16.196%) route 1.154ns (83.804%))
Logic Levels	0
Clock Path Skew	-0.336ns
Clock Un...rtainty	0.267ns
Timing ...ception	MultiCycle Path Setup -end 2

图 5-51　慢时钟域到快时钟域的多周期路径建立时间分析报告

慢时钟域到快时钟域的多周期路径建立时间分析如图 5-52 所示。这里只分析多周期路径，不再分析具体的延时信息。

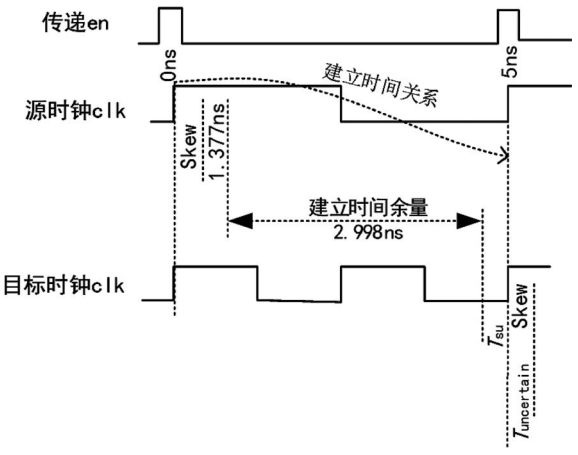


图 5-52　慢时钟域到快时钟域的多周期路径建立时间分析图

图 5-51 和图 5-52 中，建立时间余量为 2.998ns，满足时序要求；数据需求时间为 5ns，启动沿为 0ns，锁存沿为 5ns，也就是 2 个目标时钟周期。在时序例外窗口标明该路径为多周期路径 Setup 时序分析，周期数为 2，目标时钟(-end)为参考时钟。

慢时钟域到快时钟域的多周期路径保持时间分析报告如图 5-53 所示。

Summary	
Name	Path 6
Slack (Hold)	0.210ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {rise@0.000ns fall@2.500ns period=5.000ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m {rise@0.000ns fall@1.250ns period=2.500ns})
Path Group	clk_out2_ip_pll_20m
Path Type	Hold (Min at Fast Process Corner)
Requirement	0.000ns (clk_out2_ip_pll_20m rise@0.000ns - clk_out1_ip_pll_20m rise@0.000ns)
Data Path Delay	0.815ns (logic 0.100ns (12.272%) route 0.715ns (87.728%))
Logic Levels	0
Clock Path Skew	0.298ns
Clock Un...rtainty	0.267ns
Timing ...ception	MultiCycle Path Setup -end 2 Hold -end 1

图 5-53　慢时钟域到快时钟域的多周期路径保持时间分析报告

慢时钟域到快时钟域的多周期路径保持时间分析如图 5-54 所示。

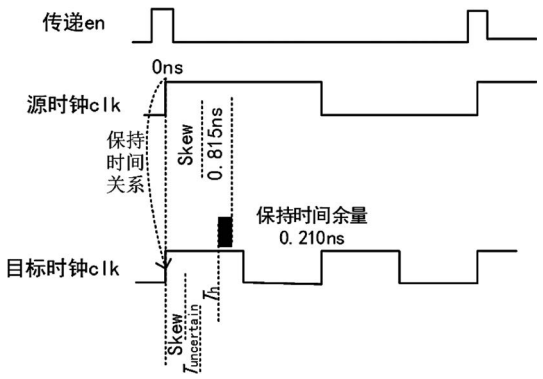


图 5-54　慢时钟域到快时钟域的多周期路径保持时间分析图

图 5-53 和图 5-54 中，保持时间余量为 0.210ns，满足时序要求；数据需求时间为 0ns，启动沿为 0ns，锁存沿为 0ns。由于保持时间分析也受到建立时间多周期约束影响，与保持时间多周期约束也有关，这里标明建立时间周期数为 2，目标时钟(-end)为参考时钟，保持时间周期数为 1。以该工程为例，看一下多周期路径的约束指令是如何扩展路径的多周期的，多周期路径约束扩展多周期示意如图 5-55 所示。

图 5-55 中，未使用多周期路径约束时，建立时间锁存沿是离启动沿最近的上升沿，建立时间需求时间为 1 个时钟周期，保持时间需求时间为 0 个时钟周期。首先，set_multicycle_path 2 -setup -end 指令将建立时间和保持时间的锁存沿右移 1 个周期(建立时间本来有 1 个时钟周期)，建立时间多周期约束也会影响保持时间；其次，set_multicycle_path 1 -hold -end 指令将保持时间的锁存沿左移 1 个周期。该波形就是多周期路径的设计期望。

5.4.5　快时钟域到慢时钟域多周期路径约束

创建一个快时钟域到慢时钟域的多周期路径数据传递工程 project21，其工程代码如

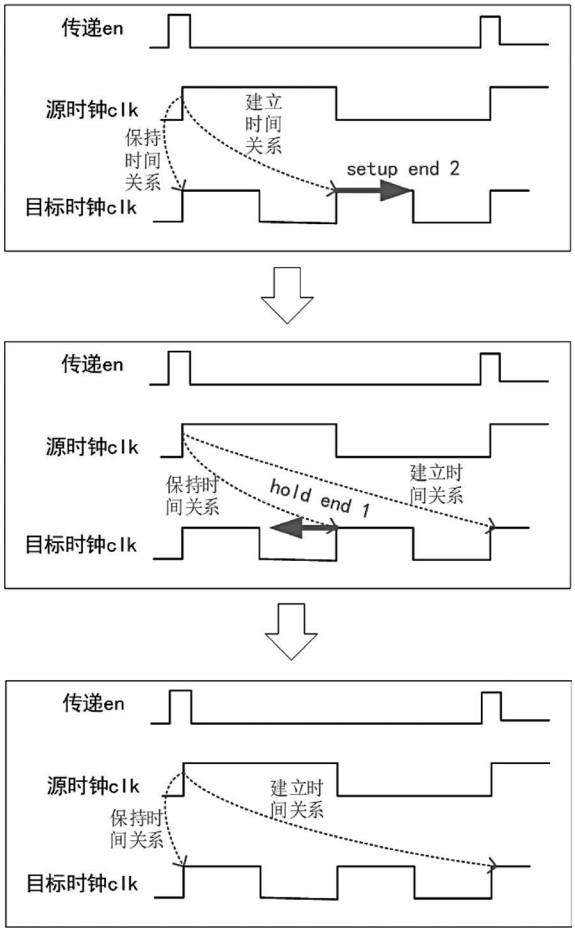


图 5-55　多周期路径约束扩展多周期示意图

project21.v 所示。

```
project21.v

    module project1
    (
    input             I_ad_data,
    input             I_clk,
    output            O_D
     );

    wire    S_clk_200m;
    wire    S_clk_400m;

    ip_pll_20m  inst_ip_pll_200m
         (
        .clk_in1              (I_clk),
        .clk_out1             (S_clk_200m),
        .clk_out2             (S_clk_400m),
        .reset                (1'b0),
        .locked               ( )
          );
```

```
reg     vld;
always@(posedge S_clk_200m) begin
    vld         <=     1'b1;
end

reg  [1:0]   en_cnt;
always@(posedge S_clk_400m) begin
    if(vld)
         begin
          if(en_cnt == 2'b01)  // 2 3 4
               en_cnt <= 2'b00;
          else
               en_cnt <= en_cnt + 1'b1;
          end
    else
         en_cnt  <=   2'b00;
end

reg     reg_DATA;
always@(posedge S_clk_400m) begin
    if(en_cnt   ==   2'b01)
         reg_DATA    <=     I_ad_data;
    else
         reg_DATA    <=     reg_DATA;
end

reg     reg_DATA1;
always@(posedge S_clk_200m) begin
        reg_DATA1   <=     reg_DATA;
end

assign  O_D   = reg_DATA1;

endmodule
```

主时钟为 20MHz，衍生时钟分别为 200MHz 和 400MHz，数据由 400MHz 时钟域传递到 200MHz 时钟域。快时钟域到慢时钟域的多周期路径时序如图 5-56 所示。

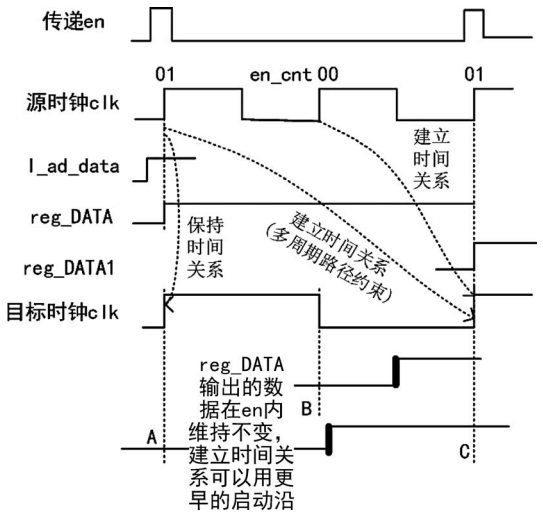


图 5-56　快时钟域到慢时钟域的多周期路径时序图

图 5-56 中，依据逻辑设计慢时钟一个周期(5ns)更新一次数据，快时钟 2 个时钟周期(2×2.5ns)更新一次数据(en_cnt=01)，所以 reg_DATA 寄存器输出的值会锁存 5ns 不变，之后传递给 reg_DATA1 寄存器，时序图与逻辑设计的期望一致。仅有使能 en(en_cnt=01)时，也就是 2 个源时钟周期，源寄存器才更新数据。

默认布局布线时，建立时间分析最糟糕的时序路径，启动沿为源时钟上升沿(en_cnt00)，锁存沿为目标时钟上升沿(en_cnt01)；reg_DATA 寄存器在源时钟(en_cnt00)锁存数据后，数据到达时间需要卡在两条虚线 B、C 之间。事实上，reg_DATA 在 2 个周期内输出的数据维持不变，启动沿可以更早一点，将启动沿提前到源时钟上升沿(en_cnt01)即可。此时，reg_DATA 寄存器在源时钟(en_cnt01)锁存数据后，数据到达时间要卡在两条虚线 A、C 之间，不仅放宽了该路径的时序，而且不影响时序功能。因此可以设置多周期路径约束。

在工程中添加多周期路径约束：

```
set_multicycle_path 2   -start   -setup -from [get_pins reg_DATA_reg/C] -to [get_pins reg
_DATA1_reg/D]
set_multicycle_path 1   -start   -hold -from [get_pins reg_DATA_reg/C] -to [get_pins reg_
DATA1_reg/D]
```

这里的周期数 2 或 1 是由逻辑功能决定的，建立时间分析时，启动沿只能左移一个周期，启动沿参考时钟为源时钟(-start)；保持时间多周期路径受到建立时间多周期路径影响，需要添加 set_multicycle_path 1 -start -hold 使保持时间的启动沿(源时钟 start)右移一个时钟，维持保持时间关系不变。

编译后的快时钟域到慢时钟域的多周期路径建立时间分析报告如图 5-57 所示。

Summary

Name	Path 5
Slack	2.975ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m {rise@0.000ns fall@1.250ns period=2.500ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {rise@0.000ns fall@2.500ns period=5.000ns})
Path Group	clk_out1_ip_pll_20m
Path Type	Setup (Max at Slow Process Corner)
Requirement	5.000ns (clk_out1_ip_pll_20m rise@5.000ns - clk_out2_ip_pll_20m rise@0.000ns)
Data Path Delay	1.399ns (logic 0.223ns (15.935%) route 1.176ns (84.065%))
Logic Levels	0
Clock Path Skew	-0.337ns
Clock Un...rtainty	0.267ns
Timing ...ception	MultiCycle Path Setup -start 2

图 5-57 快时钟域到慢时钟域的多周期路径建立时间分析报告

快时钟域到慢时钟域的多周期路径建立时间分析如图 5-58 所示。

图 5-57 和图 5-58 中，建立时间余量为 2.975ns，满足时序要求；数据需求时间为 5ns，启动沿为 0ns，锁存沿为 5ns，也就是 2 个源时钟周期。在时序例外窗口标明该路径为多周期路径 Setup 时序分析，周期数为 2，源时钟(-start)为参考时钟。

快时钟域到慢时钟域的多周期路径保持时间分析报告如图 5-59 所示。

快时钟域到慢时钟域的多周期路径保持时间分析如图 5-60 所示。

图 5-59 和图 5-60 中，保持时间余量为 0.159ns，满足时序要求；数据需求时间为 0ns，

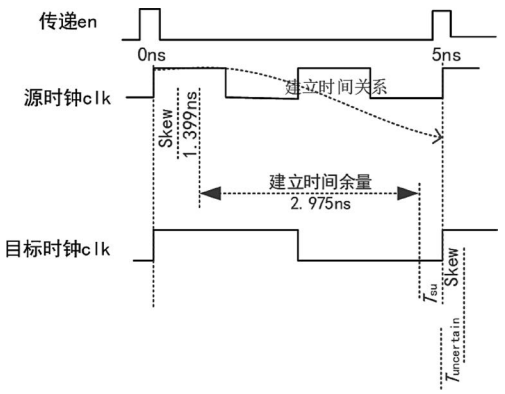


图 5-58　快时钟域到慢时钟域的多周期路径建立时间分析图

Summary	
Name	Path 6
Slack (Hold)	0.159ns
Source	reg_DATA_reg/C (rising edge-triggered cell FDRE clocked by clk_out2_ip_pll_20m {rise@0.000ns fall@1.250ns period=2.500ns})
Destination	reg_DATA1_reg/D (rising edge-triggered cell FDRE clocked by clk_out1_ip_pll_20m {rise@0.000ns fall@2.500ns period=5.000ns})
Path Group	clk_out1_ip_pll_20m
Path Type	Hold (Min at Fast Process Corner)
Requirement	0.000ns (clk_out1_ip_pll_20m rise@0.000ns - clk_out2_ip_pll_20m rise@0.000ns)
Data Path Delay	0.763ns (logic 0.100ns (13.098%) route 0.663ns (86.902%))
Logic Levels	0
Clock Path Skew	0.298ns
Clock Un...rtainty	0.267ns
Timing ...ception	MultiCycle Path Setup -start 2 Hold -start 1

图 5-59　快时钟域到慢时钟域的多周期路径保持时间分析报告

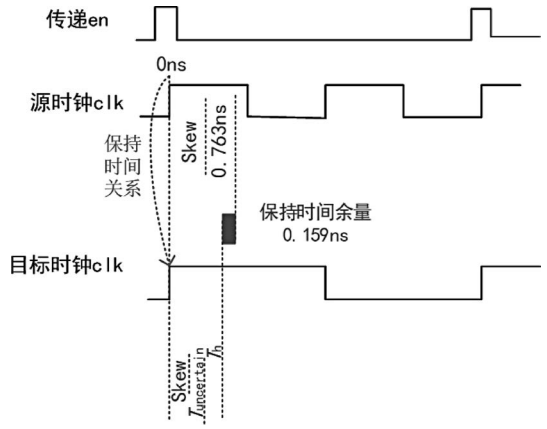


图 5-60　快时钟域到慢时钟域的多周期路径保持时间分析图

启动沿为 0ns，锁存沿为 0ns。由于保持时间分析受到建立时间多周期约束影响，与保持时间多周期约束也有关，这里标明建立时间周期数为 2，源时钟(-start)为参考时钟，保持时间周期数为 1。以该工程为例，看一下多周期路径的约束指令是如何扩展路径的多周期的，多周期路径约束扩展多周期示意如图 5-61 所示。

图 5-61 中，未使用多周期路径约束时，建立时间分析最糟糕的时序路径，启动沿为源时钟上升沿(en_cnt＝00)，锁存沿为目标时钟上升沿(en_cnt＝01)；建立时间需求时间为 1 个

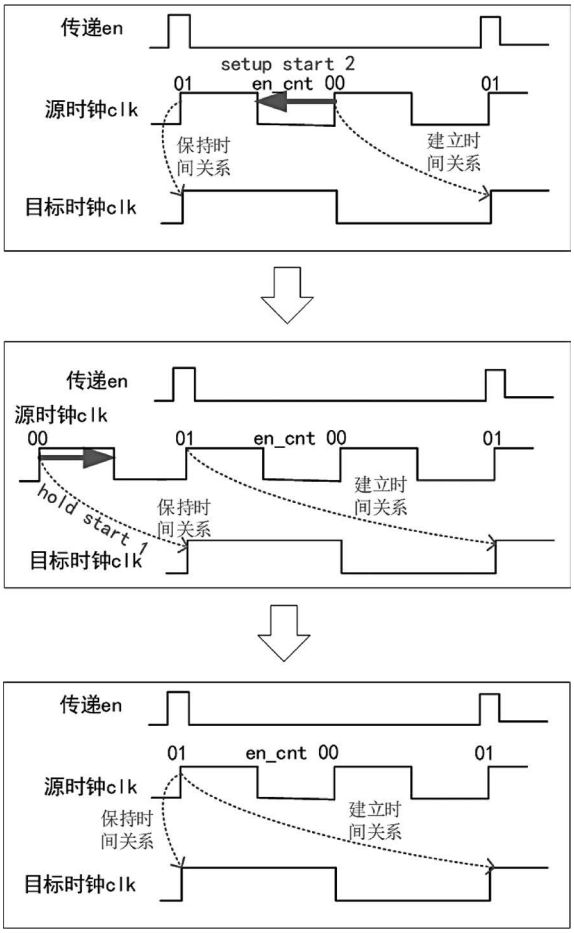


图 5-61　多周期路径约束扩展多周期示意图

时钟周期，保持时间需求时间为 0 个时钟周期。首先，set_multicycle_path 2 -setup -start 指令将建立时间和保持时间的启动沿左移 1 个周期（建立时间本来有 1 个时钟周期），建立时间多周期约束也会影响保持时间；其次，set_multicycle_path 1 -hold -start 指令将保持时间的启动沿右移 1 个周期。该波形就是多周期路径的设计期望。

请结合图 5-37、图 5-44、图 5-55 和图 5-61 中的箭头移动方向，理解 set_multicycle_path 配置参数的意义。

综上，多周期路径约束的基本指令如下：

```
set_multicycle_path N - setup - from [get_pins * /C] - to [get_pins * /D]
set_multicycle_path N - 1 - hold - from [get_pins * /C] - to [get_pins * /D]
```

依据逻辑设计将时序放宽到 N 个周期内，建立时间分析为 N 周期时，保持时间分析一般为 $N-1$；快时钟域到慢时钟域设计，建立/保持时间约束均设计-start 选项；慢时钟域到快时钟域设计，建立/保持时间约束均设计-end 选项。源时钟（-start）和目标时钟（-end）哪个频率更高，则设计哪个选项；源时钟（-start）和目标时钟（-end）频率相同，则均无须设计。

当两个时钟域之间传递复位、初始化、Done 信号、初始参数配置等电平/数据信号时，信号只要在多周期内传过去就行，无论是异步时钟还是同步时钟，都可以设计多周期路径放

宽路径时序要求，注意跨时钟域亚稳态即可。

多周期路径约束不是想设计就设计，它由路径的特性决定；路径本身可以放宽时序约束且逻辑功能不影响时，才可以设计多周期路径，而不是时序违例了就用多周期路径放宽约束消除违例，这样做会影响逻辑功能。观察 project18.v、project20.v 和 project21.v 逻辑代码中的 vld 和 en_cnt 寄存器，这两个寄存器用于对齐时钟，设计者需要明确快慢时钟在哪里对齐，在哪里传递。

5.5　时序例外约束优先级

时钟约束、输入/输出延时约束、时序例外约束是时序约束最常用的三大类约束，一般推荐按照主时钟约束、虚拟时钟、衍生时钟、输入/输出延时约束、时序例外约束的顺序进行约束。

值得注意的是，时序例外约束的优先级由高到低为时钟组约束(set_clock_groups)、伪路径约束(set_false_path)、最大/最小延时约束(set_max_delay/set_min_delay)、多周期路径约束(set_multicycle_path)。上述时序例外约束都可以用来放宽路径的时序要求，只是它们放宽时序的机理不一致。

关于时序例外约束有如下建议。

- 强烈推荐使用时序例外约束约束具体的时序路径，而非约束两个时钟之间的时序路径。set_false_path 指令非常危险，其优先级高，约束范围太大，可能会意外覆盖期望设计。set_false_path -from CLK1 -to CLK2 建议修改为 set_false_path -from */C -to */D，否则可能会影响 CLK1 到 CLK2 的异步 FIFO。
- 强烈推荐使用多周期路径约束代替最大/最小延时约束，伪路径约束和时钟组约束放宽路径的时序要求。当设计者使用优先级高的时序例外约束，约束不规范时，可能会覆盖 Vivado 自带 IP 核约束或其他未注意到的约束，故障难以排查。
- 各类时序约束必须在时钟约束之后，或时钟约束必须优先进行，否则编译将报错。当编译报错“找不到××时钟”时，则将时钟约束置于其他约束之前。如果其他约束和时钟约束不在同一个.xdc 文件中，则优先执行时钟约束。

5.6　时序例外约束对应的逻辑设计

异步时钟或跨时钟域数据传递时，时序路径一般需要设计时序例外约束，这里推荐一些逻辑设计的经验。

跨时钟域传递单比特：单比特电平信号，逻辑设计打拍消除亚稳态，无须考虑快慢时钟。跨时钟域路径可以使用时序例外约束(单比特建议多周期，fifo 格雷码用的是最大延时约束，保证目标寄存器单比特跳变)，打拍同步路径可以使用最大/最小延时约束(set_max_delay/set_min_delay)，避免采到亚稳态数据。单比特脉冲信号，逻辑设计 xpm_cdc_pulse 或 FIFO IP 核，无须考虑时序约束。

跨时钟域传递多比特：数据量较多，逻辑设计 RAM 或 FIFO IP 核，无须考虑时序约束。多比特数据线，设计逻辑协议，读、写、ready、ack 等信号打拍提沿，无特殊设计暂不考虑时序约束。