Procise 约束手册

（版本号：V2.1）

上海复旦微电子集团股份有限公司

(86-21)6565 5050

(86-21)6565 9115

200433

上海市国泰路 127 号复旦国家大学科技园 4 号楼

修订记录

日期	修订版本	描述	作者
2021-08-10	1.0	初稿	
2022-03-01	2.1	增加了 IOBDELAY 约束	
2023-04-01	2.1	修改 set_clock_groups 命令的一处错误	宋功明

目录

1 关于手册.....1

2 约束的组织.....1

2.1 使用 FDC 约束文件.....1

2.2 约束文件的组织.....1

2.3 约束文件加载顺序.....1

2.4 约束文件作用范围.....2

2.5 指定约束文件的生效阶段.....2

3 对象获取.....2

3.1 网表对象.....2

3.1.1 get_ports.....2

3.1.2 get_nets.....3

3.1.3 get_cells.....3

3.1.4 get_pins.....3

3.1.5 all_inputs.....3

3.1.6 all_outputs.....4

3.1.7 all_clocks.....4

3.2 器件对象.....4

3.2.1 get_iobanks.....4

3.2.2 get_sites.....4

3.2.3 get_bels.....5

3.3 使用建议.....5

4 时序约束.....5

4.1 定义时钟.....5

4.1.1 create_clock.....5

4.1.2 create_generated_clock.....7

4.1.3 set_clock_uncertainty.....9

4.1.4 set_clock_latency.....10

4.1.5 set_clock_groups.....11

4.1.6 set_external_delay.....12

4.2 设置输入/输出延时.....13
4.2.1 set_input_delay.....13
4.2.2 set_output_delay.....14
4.3 时序例外.....16
4.3.1 set_false_path.....16
4.3.2 set_multicycle_path.....17
4.3.3 set_max _delay.....18
4.3.4 set_min_delay.....20
4.4 时序约束优先级.....20
5 物理约束.....20
5.1 I/O 约束.....20
5.1.1 PACKAGE_PIN.....21
5.1.2 IOSTANDARD.....21
5.1.3 DRIVE.....24
5.1.4 SLEW.....24
5.1.5 IN_TERM.....24
5.1.6 DIFF_TERM.....24
5.1.7 PULLTYPE.....25
5.1.8 VCCAUX_IO.....25
5.1.9 DCI_CASCADE.....25
5.1.10 INTERNAL_VREF.....26
5.1.11 IODELAY_GROUP.....26
5.1.12 IOBDELAY.....26
5.2 布局约束.....27
5.2.1 LOC.....27
5.2.2 BEL.....27
5.3 配置约束.....27
5.3.1 PROHIBIT.....27
5.3.2 CLOCK_DEDICATED_ROUTE.....28

1 关于手册

本手册旨在向 Procise 用户说明该软件支持的约束语法，包括时序约束和物理约束两大类。

2 约束的组织

2.1 使用 FDC 约束文件

FDC 是 Fudan Design Constraints 的简称。

FDC 约束是行业标准 SDC(Synopsys Design Constraints, version 1.9)的一个子集和复旦微专有物理约束的组合。FDC 约束本质上是一系列 Tcl 命令，和其他 Tcl 命令一样被 Procise 软件解析执行。可以通过以下几种方式使用 FDC 约束：

1. 将约束保存于 FDC 约束文件，通过工程向导或 read_fdc 命令添加 FDC 文件。
2. 将约束保存于 FDC 约束文件，通过 source 命令执行 FDC 文件。
3. 输入单个约束命令执行。

2.2 约束文件的组织

用户可以添加多个 FDC 约束文件，比如有的文件定义时序，有的文件定义物理约束等。可以指定其中一个为 active 的 FDC 文件，用户通过 Procise 软件的约束向导工具修改的约束文件是指定的 active 的 FDC 约束文件。

2.3 约束文件加载顺序

当存在多个约束文件时，需要注意加载次序。有的约束依赖于别的约束，因此有的需要先加载，有的后加载。Procise 软件无法帮用户分析约束文件的优先顺序，需要用户特别注意。软件默认情况是根据用户添加的 FDC 文件依次加载。

2.4 约束文件作用范围

Procise 软件暂不支持约束文件的作用范围。

2.5 指定约束文件的生效阶段

Procise 软件暂不支持约束文件的指定生效阶段。

3 对象获取

用户在使用时序约束命令和物理约束命令之前，需要通过对象获取命令来得到被约束的对象，这些对象包括用户网表中的引脚（Port）、线网（Net）、单元（Cell）以及单元引脚（Pin），还包括 FPGA 器件的 Site、Bel、IO bank。

3.1 网表对象

3.1.1 get_ports

获取用户网表中的引脚。

命令格式为：

```
get_ports [-regexp] [-nocase] [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

命令示例 1（单个 pattern）：

```
get_ports clk
get_ports *clk*
```

命令示例 2（多个 pattern）：

```
get_ports {data_in?  sys*}
```

命令示例 3（指定 of_objects）：

```
get_ports -of_objects [get_nets clk]
```

3.1.2 get_nets

获取用户网表中的线网。

命令格式为：

```
get_nets [-regexp] [-nocase] [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

命令用法同上。

3.1.3 get_cells

获取用户网表中的单元。

命令格式为：

```
get_cells [-regexp] [-nocase] [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

命令用法同上。

3.1.4 get_pins

获取用户网表中的单元引脚。

命令格式为：

```
get_pins [-regexp] [-nocase] [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

命令用法同上。

3.1.5 all_inputs

获取用户网表中的所有输入引脚。

命令格式为：

```
all_inputs [-quiet] [-verbose]
```

3.1.6 all_outputs

获取用户网表中的所有输出引脚。

命令格式为：

```
all_outputs [-quiet] [-verbose]
```

3.1.7 all_clocks

获取用户网表中的所有时钟。

命令格式为：

```
all_clocks [-quiet] [-verbose]
```

该命令等同于

```
get_clocks *
```

3.2 器件对象

3.2.1 get_iobanks

获取当前器件的 IO banks。

命令格式为：

```
get_iobanks [-quiet] [-verbose] [<patterns>]
```

命令示例：

```
get_iobanks 12
```

3.2.2 get_sites

获取当前器件的 sites。

命令格式为：

```
get_sites [-quiet] [-verbose] [<patterns>]
```

命令示例：

```
get_sites LC_X10Y20
get_sites LC_X1*Y*
```

3.2.3 get_bels

获取当前器件的 bels。

```
get_bels [-quiet] [-verbose] [-of_objects <args>]
         [<patterns>]
```

命令示例：

```
get_sites ALUT5
get_sites AFF -of_objects [get_sites LC_X10Y20]
```

3.3 使用建议

当用户知道某一对象的名字时，建议用户用精确匹配，可以提高命令的效率。

4 时序约束

4.1 定义时钟

4.1.1 create_clock

create_clock 命令对 FPGA 主时钟（primary clock）和虚拟时钟（virtual clock）进行约束。

命令格式为：

```
create_clock -period <float> [-name <string>]
             [-waveform <string>] [-add] <objects>
```

各参数定义如下：

-period	定义时钟周期。
-name	定义 clock 名称，若不指定-name 值，则 clock 名称为 objects 名。

-waveform	定义时钟波形占空比和相移，参数值个数为偶数个，第一个值对应上升沿，第二个值对应下降沿。若不指定-waveform 值，等同于-waveform{0 period/2}。
-add	使用 add 参数给同一个端口定义多个时钟，若不添加-add 参数则以本次定义的时钟来覆盖已经定义在时钟源上的所有时钟。
<objects>	定义时钟源，可以是 pins 或 ports，若不指定时钟源，则为虚拟时钟约束。

约束举例：

1. 全局时钟管脚作为主时钟（primary clock）

1) 定义时钟周期为 20ns，占空比为 50%，相移为 0 的时钟约束：

```
create_clock -period 20 [get_ports clk_in]
```

2) 定义时钟周期为 8ns，占空比为 25%，相移为 90 度的时钟约束：

```
create_clock -name clk -period 20
             -waveform {5 10} [get_ports clk_p]
```

3) 同一个时钟源产生多种时钟：

```
create_clock -name C1 -period 10 [get_ports CLK]
create_clock -name C2 -period 15 -add [get_ports CLK]
```

2. 千兆位高速收发器引脚的输出作为主时钟（primary clock）

定义高速收发器 gt0 的引脚 RXOUTCLK 生成时钟周期为 5ns 的时钟：

```
create_clock -name rxclk0 -period 5 [get_pins gt0/RXOUTCLK]
```

3. 虚拟时钟（virtual clock）

定义时钟周期为 10ns 的虚拟时钟：

```
create_clock -name clk_virt -period 10
```

4.1.2 create_generated_clock

用于创建生成时钟或对系统自动推断出的生成时钟做重命名。

命令格式为：

```
create_generated_clock [-name <string>]  [-source <objs>]
                       [-edges <string>] [-divide_by <float>]
                       [-multiply_by <float>][-combinational]
                       [-duty_cycle <float>] [-invert]
                       [-edge_shift <string>]
                       [-master_clock <string>] [-add]
                       <objects>
```

各参数定义如下：

-name	指定生成时钟（generated clock）名称。
-source	指定主时钟（master clock）传播的端口（port）或引脚（pin）。
-edges	-edges 接的参数是一个包含三个整数值的索引列表，指定生成时钟（generated clock）的第 1 个上升沿，第 1 个下降沿和第 2 个上升沿对应主时钟的第几个沿（从 1 开始计数）。
-divide_by	对主时钟频率进行分频。
-multiply_by	对主时钟频率进行倍频。
-combination	创建“-divide_by 1”的生成时钟
-duty_cycle	配合-multiply_by 选项使用，用以调节占空比。
-invert	使生成时钟的波形反转。
-edge_shift	将指定-edges 选项中三条时钟沿的偏移量，因此

	将包含 3 个正负皆可的浮点数，单位为基本时间单位 ns
-master_clock	如果主时钟的源点上定义了多个时钟，则需要指定哪个时钟作为本生成时钟的主时钟。
-add	表示不要覆盖<objects>上已经定义的时钟，将新时钟添加进去。不使用-add 表示以本生成时钟去覆盖<objects>上已经定义的所有时钟。
<objects>	指定生成时钟的源点。

约束举例：

1. 用-divide_by 参数使生成时钟频率为主时钟频率的一半。

```
create_clock -name clkin -period 10 [get_ports clkin]
create_generated_clock -name clkdiv2 -source [get_ports clkin]
                       -divide_by 2 [get_pins REGA/Q]
```

2. 用-edges 参数使生成时钟为主时钟频率的一半。

```
create_generated_clock -name clkdiv2
                       -source [get_pins REGA/C]
                       -edges {1 3 5} [get_pins REGA/Q]
```

3. 用-edges 和-edge_shift 参数调节生成时钟的占空比和相移。

```
create_clock -name clkin -period 10 [get_ports clkin]
create_generated_clock -name clkshift
                       -source [get_pins mmcm0/CLKIN]
                       -edges {1 2 3} -edge_shift {2.5 0 2.5}
                       [get_pins mmcm0/CLKOUT]
```

4.1.3 set_clock_uncertainty

用于定义时钟的不确定度，用于增加时序裕量，也可以只定义某些特定的方向和时序路径。

命令格式为：

```
set_clock_uncertainty [-setup] [-hold] [-from <objs>]
                      [-rise_from <objs>]
                      [-fall_from <objs>] [-to <objs>]
                      [-rise_to <objs>]
                      [-fall_to <objs>] <uncertainty>
                     [<objects>]
```

各参数定义如下：

-setup	表示时钟不确定性是用在建立时序分析过程
-hold	表示时钟不确定性是用在保持时序分析过程。
-from	表示 source clock
-rise_from	表示 source clock 的活动沿是上升沿
-fall_from	表示 source clock 的活动沿是下降沿
-to	表示 dest clock
-rise_to	表示 dest clock 的活动沿是上升沿
-fall_to	表示 dest clock 的活动沿是上升沿
<uncertainty>	定义时钟不确定的值，单位 ns。
<objects>	指定 uncertainty 应用到哪些时钟上。

约束举例：

```
set_clock_uncertainty 2.0 -from [get_clocks clk1]
                          -to [get_clocks clk2]
set_clock_uncertainty 1.0 [get_clocks clk1]
```

说明：复杂的时钟不确定性优于简单的时钟不确定性，尽管后面一个约束限制了 clk1 1ns 的时钟不确定性约束，从 clk1 到 clk2 依旧有 2ns clock uncertainty 的约束限制。

4.1.4 set_clock_latency

用于定义时钟或者 port 的 source latency。

命令格式为：

```
set_clock_latency [-clock <args>] [-rise]
                  [-fall] [-min] [-max] [-source]
                  [-late] [-early] [-quiet] [-verbose]
                  <latency> <objects>
```

各参数定义如下：

-clock	指定<latency>作用在哪些时钟上，如果不指定次选项，则表示作用在<objects>上定义的所有时钟。
-rise	指定 clock rise 时的 latency。
-fall	指定 clock fall 时的 latency。
-max/ -min	指定最大的/最小的 latency。
-source	选项说明指定的为 source latency，否则为 network latency。
-late	表示时钟沿到得多晚。

-early	表示时钟沿到得多早。
-latency	时钟延迟总量。
-objects	指定 lantency 适用的对象，可以是 clock, port, pin。

约束举例：

1. 定义 sysClk 时钟最小的时钟延迟为 0.2ns：

```
set_clock_latency -source -early 0.2 [get_clocks sysClk]
```

2. 定义 sysClk 时钟最大的时钟延迟为 0.5ns：

```
set_clock_latency -source -late 0.5 [get_clocks sysClk]
```

4.1.5 set_clock_groups

设置不同的时钟组，不同组内的不做时序分析，同组内的同步关系，包括单时钟点上 overlap 的情况等。

命令格式为：

```
set_clock_groups [-name <arg>] [-logically_exclusive]
                 [-physically_exclusive]
                 [-asynchronous] [-group <args>]
                 [-quiet] [-verbose]
```

各参数定义如下：

-name	时钟组名称。
-logically_exclusive	表示这些时钟组在逻辑上不会同时出现。
-physically_exclusive	表示这些时钟组在物理上不会同时出现。
-asynchronous	完全不相干的时钟。

-group	时钟列表。

约束举例：

```
set_clock_groups -name exclusive_clk0_clk1
                 -physically_exclusive
                 -group [get_clocks clk0] -group [get_clocks
        clk1]
```

4.1.6 set_external_delay

用于设置片外输出到输入的反馈延时。

命令格式为：

```
set_external_delay -from <args> -to <args>
                   [-min] [-max] [-add]
                   [-quiet] [-verbose]
                   <delay_value>
```

各参数定义如下：

-from	输出端口名称。
-to	输入端口名称。
-min	为 hold time 设置最小延迟时间。
-max	为 setup time 设置最大延迟时间。
-add	添加额外的延迟时间。
delay_value	反馈延迟时间，单位为 ns,默认为 0。

约束举例：

定义 ClkOut 和 ClkFb 之间的反馈延迟为 1ns。

```
set_external_delay 1.0 -from [get_ports ClkOut]
                       -to [get_ports ClkFb]
```

4.2 设置输入/输出延时

4.2.1 set_input_delay

设置输入端口的片外延时。

命令格式为：

```
set_input_delay  [-clock <objs>] [-reference_pin <objs>]
                 [-clock_fall] [-rise] [-fall] [-max]
                 [-min] [-add_delay]
                 [-network_latency_included]
                 [-source_latency_included]
                 <delay> <objects>
```

各参数定义如下：

-clock	输入延时相对于哪些时钟的活动沿。缺省指上升沿，但是可以用-clock_fall 来指定下降沿
-reference_pin	输入延时相对于哪些 pin 上出现的时钟信号活动沿
-clock_fall	表明输入延时相对于时钟的下降沿
-rise	表明输入延时在<objects>上的信号 rise 时起作用
-fall	表明输入延时在<objects>上的信号 fall 时起作用
-max	表明输入延时仅在做 setup 分析时用到
-min	表明输入延时仅在做 hold 分析时用到

-add_delay	将<delay>加到<objects>已经定义的 delay 上，不用此选项表示替代<objects>已经定义的 delay。
-network_latency_included	表示 delay 里面已经包含了 clock 的 network 延时
-source_latency_included	表示 delay 里面已经包含了 clock 的 source 延时
<delay>	延迟时间。单位 ns
<objects>	表示<delay>作用在哪些输入端口。

约束举例：

约束虚拟时钟 clk_port_virt 到达 DIN 输入端口的时钟延迟。

```
create_clock -name clk_port_virt -period 10
set_input_delay -clock [get_clocks clk_port_virt] 2 [get_ports DIN]
```

注意指定-clock 选项时要使用[get_clocks clkname]，不要仅仅使用 clkname

4.2.2 set_output_delay

设置输出端口的片外延时。

命令格式为：

```
set_output_delay [-clock <objs>] [-reference_pin <objs>]
                 [-clock_fall] [-rise] [-fall] [-max]
                 [-min] [-add_delay]
                 [-network_latency_included]
                 [-source_latency_included]
                 [-quiet] [-verbose] <delay> <objects>
```

各参数定义如下：

-clock	输出延时相对于哪些时钟的活动沿。缺省指上升沿，但是可以用-clock_fall 来指定下降沿
-reference_pin	输出延时相对于哪些 pin 上出现的时钟信号活动沿
-clock_fall	表明输出延时相对于时钟的下降沿
-rise	表明输出延时在<objects>上的信号 rise 时起作用
-fall	表明输出延时在<objects>上的信号 rise 时起作用
-max	表明输出延时仅在做 setup 分析时用到
-min	表明输出延时仅在做 hold 分析时用到
-add_delay	将<delay>加到<objects>已经定义的 delay 上，不用此选项表示替代<objects>已经定义的 delay。
-network_latency_included	表示 delay 里面已经包含了 clock 的 network 延时
-source_latency_included	表示 delay 里面已经包含了 clock 的 source 延时
<delay>	延迟时间。单位 ns
<objects>	表示<delay>作用在哪些输出端口。

约束举例：

约束系统时钟 sysClk 到达输出端口 DOUT 的输出延迟。

```
create_clock -name sysClk -period 10 [get_ports CLK0]
set_output_delay -clock [get_clocks sysClk] 6 [get_ports DOUT]
```

注意指定`-clock`选项时要使用`[get_clocks clkname]`，不要仅仅使用`clkname`

4.3 时序例外

4.3.1 set_false_path

忽略定路径的时序分析。

命令格式为：

```
set_false_path [-setup] [-hold] [-rise] [-fall] [-reset_path]
               [-from <objs>] [-rise_from <objs 指>]
               [-fall_from <objs>] [-to <objs>]
               [-rise_to <objs>] [-fall_to <objs>]
               [-through <objs>] [-rise_through <objs>]
               [-fall_through <objs>]
               [-quiet] [-verbose]
```

各参数定义如下：

-setup/-hold	指定在 setup/hold check 时不分析指定路径。
-rise	表示在 path 的信号 rise 时不做分析
-fall	表示在 path 的信号 fall 时不做分析
-reset_path	先清除 path 上的约束，然后添加此约束
-from	指定 path 的起点或 clock。
-rise_from	指起点信号 rise，或 clock rise
-fall_from	指起点信号 fall，或 clock fall
-to	指定 path 的终点或 clock。
-rise_to	指终点信号 rise，或 clock rise

-fall_to	指终点信号 fall，或 clock fall
-through	指定时序路径必须通过的中间点， pin，cell，net 的列表。
-rise_through	指中间点信号 rise
-fall_through	指中间点信号 fall

约束举例：

将所有从 reset 端口出发的时序路径不做分析。

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

4.3.2 set_multicycle_path

用来改变默认的单周期 setup/hold 关系分析，且 hold 关系是与 setup 紧密联系的。因此，在大部分情况下，setup 关系调整后，hold 也要做相应地调整。主要的一个例外是相移时钟间同步的 CDC path，只有 setup 需要调整。

命令格式为：

```
set_multicycle_path [-setup] [-hold] [-rise] [-fall]
                    [-start] [-end] [-reset_path]
                    [-from <objs>] [-rise_from <objs>]
                    [-fall_from <objs>]
                    [-to <objs>] [-rise_to <objs>]
                    [-fall_to <objs>]
                    [-through <objs>] [-rise_through <objs>]
                    [-fall_through <objs>] <multiplier>
```

各参数定义如下：

-setup	只设定 setup 的多周期数。
-hold	只设定 hold 的多周期数。

-rise	指 path 终点 rise 时约束起作用
-fall	指 path 终点 fall 时约束起作用
-start/end	指多周期数对应于路径的起点/终点
-reset_path	先清除 path 上的约束，然后添加此约束
-from	指定 path 的起点或 clock。
-rise_from	指起点信号 rise，或 clock rise
-fall_from	指起点信号 fall，或 clock fall
-to	指定 path 的终点或 clock。
-rise_to	指终点信号 rise，或 clock rise
-fall_to	指终点信号 fall，或 clock fall
-through	指定时序路径必须通过的中间点， pin，cell，net 的列表。
-rise_through	指中间点信号 rise
-fall_through	指中间点信号 fall
<multiplier>	时钟周期数。

约束举例：

```
set_multicycle_path 2 -from [all_inputs] -to [get_clocks clk1]
```

4.3.3 set_max _delay

用于设定对应 timing path 间的最大延时。

命令格式为：

```
set_max_delay [-rise] [-fall] [-reset_path] [-from <objs>]
              [-rise_from <objs>] [-fall_from <objs>]
```

```
[-to <objs>] [-rise_to <objs>]
[-fall_to <objs>] [-through <objs>]
[-rise_through <objs>] [-fall_through <objs>]
[-datapath_only] [-quiet] [-verbose] <delay>
```

各参数定义：

-rise	<delay>适用于 path 的终点 rise
-fall	<delay>适用于 path 的终点 fall
-reset_path	先清除 path 上的约束，然后添加此约束
-from	指定 path 的起点或 clock。
-rise_from	指起点信号 rise，或 clock rise
-fall_from	指起点信号 fall，或 clock fall
-to	指定 path 的终点或 clock。
-rise_to	指终点信号 rise，或 clock rise
-fall_to	指终点信号 fall，或 clock fall
-through	指定时序路径必须通过的中间点， pin，cell，net 的列表。
-rise_through	指中间点信号 rise
-fall_through	指中间点信号 fall
-datapath_only	计算时不考虑 clock skew 与 jitter
<delay>	延时数值。

约束举例：

定义路径之间的最大延迟为 15ns。

```
set_max_delay 15 -from [get_pins */aclk_dpram_reg*/*/CLK}]
                 -to [get_cells ~*/bclk_dout_reg*]
                 -datapath_only
```

4.3.4 set_min_delay

用于设定对应 timing path 间的最小延时，同 set_max_delay。

4.4 时序约束优先级

Procise 软件支持的时序约束命令的优先级从高到低为：

° clock group (set_clock_groups)

° false path (set_false_path)

° Maximum/Minimum Delay Path(set_max/min_delay)

° multicycle path (set_multicycle_path)

对于同一优先级的命令，按先后次序，后出现的命令会覆盖前面的命令。

5 物理约束

Procise 支持的所有的物理约束都通过 set_property 命令来设置。

可以每次设置一个属性：

```
set_property <property> <value> <object list>
```

也可以单次设置多个属性：

```
set_property -dict {<property> <value> …} <object list>
```

5.1 I/O 约束

用户可以给 I/O 引脚(Port)或连接引脚的线网(Net)添加 I/O 约束。

Procise 软件支持的 I/O 约束如下：

5.1.1 PACKAGE_PIN

设置 I/O 引脚的位置。

具体的管脚位置信息根据芯片封装的不同而不同。

LOC 约束示例：

```
set_property PACKAGE_PIN B8 [get_ports STATUS]
```

5.1.2 IOSTANDARD

设置 I/O 引脚的电气标准。

Procise 软件支持的 I/O 标准有：

BLVDS_25

DIFF_HSTL_I_DCI

DIFF_HSTL_I_DCI_18

DIFF_HSTL_II_DCI

DIFF_HSTL_II_DCI_18

DIFF_HSTL_II_T_DCI

DIFF_HSTL_II_T_DCI_18

DIFF_HSUL_12_DCI

DIFF_MOBILE_DDR

DIFF_SSTL12

DIFF_SSTL12_DCI

DIFF_SSTL12_T_DCI

DIFF_SSTL135_DCI

DIFF_SSTL135_R

DIFF_SSTL135_T_DCI

DIFF_SSTL15_DCI

DIFF_SSTL15_R

DIFF_SSTL15_T_DCI

DIFF_SSTL18_I_DCI

DIFF_SSTL18_II_DCI

DIFF_SSTL18_II_T_DCI

HSLVDCI_15

HSLVDCI_18

HSTL_I_12

HSTL_I_DCI

HSTL_I_DCI_18

HSTL_II_DCI

HSTL_II_DCI_18

HSTL_II_T_DCI

HSTL_II_T_DCI_18

HSUL_12_DCI

LVCMOS12

LVCMOS15

LVCMOS18

LVCMOS25

LVCMOS33

LVDCI_15

LVDCI_18

LVDCI_DV2_15

LVDCI_DV2_18

LVDS

LVDS_25

LVTTL

MINI_LVDS_25

MOBILE_DDR

PCI33_3

PPDS_25

RSDS_25

SSTL12

SSTL12_DCI

SSTL12_T_DCI

SSTL135_DCI

SSTL135_R

SSTL135_T_DCI

SSTL15_DCI

SSTL15_R

SSTL15_T_DCI

SSTL18_I_DCI

SSTL18_II_DCI

SSTL18_II_T_DCI

TMDS_33

IOSTANDARD 约束示例：

```
set_property IOSTANDARD LVCMOS12 [get_ports STATUS]
```

5.1.3 DRIVE

设置 I/O 引脚的驱动能力。

Procise 软件支持的 DRIVE 值为 < 2 | 4 | 6 | 8 | 12 | 16 | 24 >

DRIVE 约束示例：

```
set_property DRIVE 4 [get_ports STATUS]
```

5.1.4 SLEW

设置 I/O 引脚的抖动速率。

Procise 软件支持的 SLEW 值为 < SLOW | FAST >

SLEW 约束示例：

```
set_property SLEW FAST [get_ports STATUS]
```

5.1.5 IN_TERM

设置 I/O 引脚的内部阻抗。支持固定 40 欧、50 欧和 60 欧。

Procise 软件支持的 IN_TERM 值为：

° UNTUNED_SPLIT_40

° UNTUNED_SPLIT_50

° UNTUNED_SPLIT_60

IN_TERM 约束示例：

```
set_property IN_TERM UNTUNED_SPLIT_40 [get_ports IN]
```

5.1.6 DIFF_TERM

设置 I/O 引脚的内置差分电阻（100 欧）。

Procise 软件支持的 DIFF_TERM 值为 < TRUE | FALSE>，默认为 FALSE

设置该约束的前提是 I/O 引脚的电气标准必须为以下几种：

° LVDS，LVDS_25，MINI_LVDS_25

° PPDS_25

° RSDS_25

DIFF_TERM 约束示例：

```
set_property DIFF_TERM TRUE [get_ports IN]
```

5.1.7 PULLTYPE

设置 I/O 引脚的弱上拉/弱下拉/弱保持电阻。

Procise 软件支持的 PULLTYPE 值为 ＜PULLUP | PULLDOWN | KEEPER>

PULLTYPE 约束示例：

```
set_property PULLTYPE PULLUP [get_ports IN]
```

5.1.8 VCCAUX_IO

设置 VCCAUX_IO 管脚的工作电压。

在一个 bank 中，如果 VCCAUX_IO 管脚供电为 2.0V，则在该 bank 至少一个 I/O 管脚将 VCCAUX_IO 属性约束为 HIGH，其他所有 I/O 可以约束为 HIGH 或者 DONTCARE。

Procise 软件支持的 VCCAUX_IO 值为 ＜DONTCARE | NORMAL | HIGH>

VCCAUX_IO 约束示例：

```
set_property VCCAUX_IO HIGH [get_ports IN]
```

5.1.9 DCI_CASCADE

设置高速（HP）I/O banks 之间的主从（master-slave）关系。

DCI_CASCADE 约束格式为：

```
set_property DCI_CASCADE {slave_banks} [get_iobanks master_bank]
```

DCI_CASCADE 约束示例：

```
set_property DCI_CASCADE {15 16} [get_iobanks 14]
```

5.1.10 INTERNAL_VREF

设置 I/O bank 的内部参考电压值。

Procise 软件支持的 INTERNAL_VREF 值为：0.6，0.675，0.75，0.9

INTERNAL_VREF 约束示例：

```
set_property INTERNAL_VREF 0.75 [get_ports IN]
```

5.1.11 IODELAY_GROUP

当存在多个 IDELAYCTRL 单元时，用户可以将 IDELAYCTRL 单元和它关联的 IDELAY 单元、ODELAY 单元放到一个群组（IODELAY_GROUP）中。以便于 Procise 的布局工具能合理布局这些单元。

IODELAY_GROUP 约束示例：

```
set_property IODELAY_GROUP GRP1 [get_cells MY_IDELAYCTRL_inst]
set_property IODELAY_GROUP GRP1 [get_cells MY_IDELAY_inst]
set_property IODELAY_GROUP GRP1 [get_cells MY_ODELAY_inst]
```

5.1.12 IOBDELAY

IOBDELAY 约束决定了在处理系统同步输入数据时，hold 分析时是否考虑 ILOGIC 块的延时。

该约束作用的对象可以是 cell，net，port。其取值有 4 种：

NONE：不计算 IBUF 的延时，不计算 IFD 的延时.

IBUF：计算 IBUF 的延时，不计算 IFD 的延时.

IFD：不计算 IBUF 的延时，计算 IFD 的延时.

BOTH：计算 IBUF 的延时，计算 IFD 的延时.

IOBDELAY 约束示例：

```
set_property IOBDELAY BOTH [get_cells my_cell]
set_property IOBDELAY NONE [get_ports my_port]
set_property IOBDELAY NONE [get_nets my_net]
```

5.2 布局约束

用户可以给任意逻辑单元(Cell)加布局约束。

Procise 软件支持的布局约束如下：

5.2.1 LOC

指定逻辑单元在器件上的 site 位置。

LOC 约束示例：

```
set_property LOC LC_X0Y0 [get_cells u_ctrl0/lut1]
```

5.2.2 BEL

指定逻辑单元在器件上的 BEL 位置。

LOC 约束示例：

```
set_property BEL CLUT5 [get_cells u_ctrl0/lut1]
```

建议 BEL 和 LOC 一起使用，不要单独使用 BEL 约束。

5.3 配置约束

5.3.1 PROHIBIT

指定器件上的 SITE 或 BEL 不允许使用。

PROHIBIT 约束示例：

```
set_property PROHIBIT TRUE [get_sites LC_X1Y32]
```

5.3.2 CLOCK_DEDICATED_ROUTE

该约束用来指导软件是否遵循时钟配置规则。

Procise 软件支持的 CLOCK_DEDECATED_ROUTE 值为 <TRUE | FALSE>

默认取值为 TRUE。

如果由普通的 IO 管脚驱动全局时钟资源，则需设置为 FALSE。

CLOCK_DEDECATED_ROUTE 约束示例：

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets net1]
```