



易 灵 思

易灵思FPGA设计技巧

——时序收敛

陈弘 Bruce Chen

2021年9月

第二部分

——时序约束和时序优化策略



易 灵 思

9/16/2021

版本

日期	版本	版本描述
2021/9/10	V1.0	初稿发布



主要内容

1. 时序约束
2. 时序优化策略



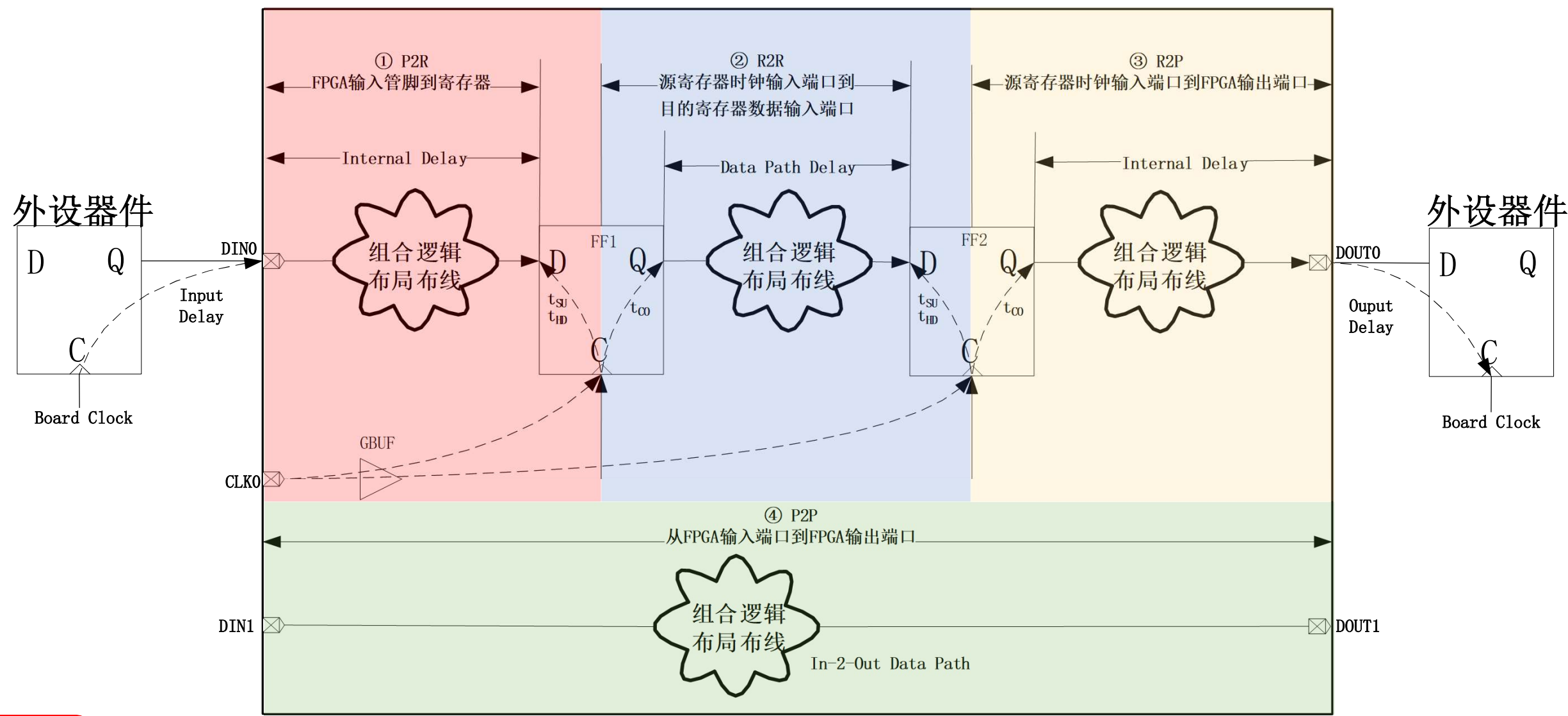
回顾

- FPGA的最大工作频率计算公式: $f_{MAX} = 1/(t_{SUmmin} + t_{CO} + t_{PD} - t_{SK} + t_{JT})$
 - t_{SUmmin} 由FPGA的时序模型决定
 - t_{CO} 由FPGA的时序模型决定
 - t_{PD} 由FPGA的时序模型和综合以后的组合逻辑以及布局布线决定
 - t_{SK} 由FPGA的时序模型和综合以后的时钟树决定
 - t_{JT} 由时钟源决定
- 时序收敛条件
 - f_{MAX} 必须大于等于系统要求
 - 保持时间和建立时间的余量(Slack)都必须大于0
 - 传播延迟越小建立时间越容易满足
 - 传播延迟越大保持时间越容易满足
- 准确和完整地输入时序约束是FPGA设计中最重要的一部分之一
 - 时钟参数: 时钟周期, 相位, 占空比, 抖动Jitter/Uncertainty, 时钟之间的关系
 - 外设和FPGA之间各种接口信号(R2R/R2P/P2R/P2P)的路径延迟
- FPGA设计工具根据输入的时序约束调用对应的时序模型计算并完成布局布线和静态时序分析



FPGA的四种基本时序路径分析

FPGA器件

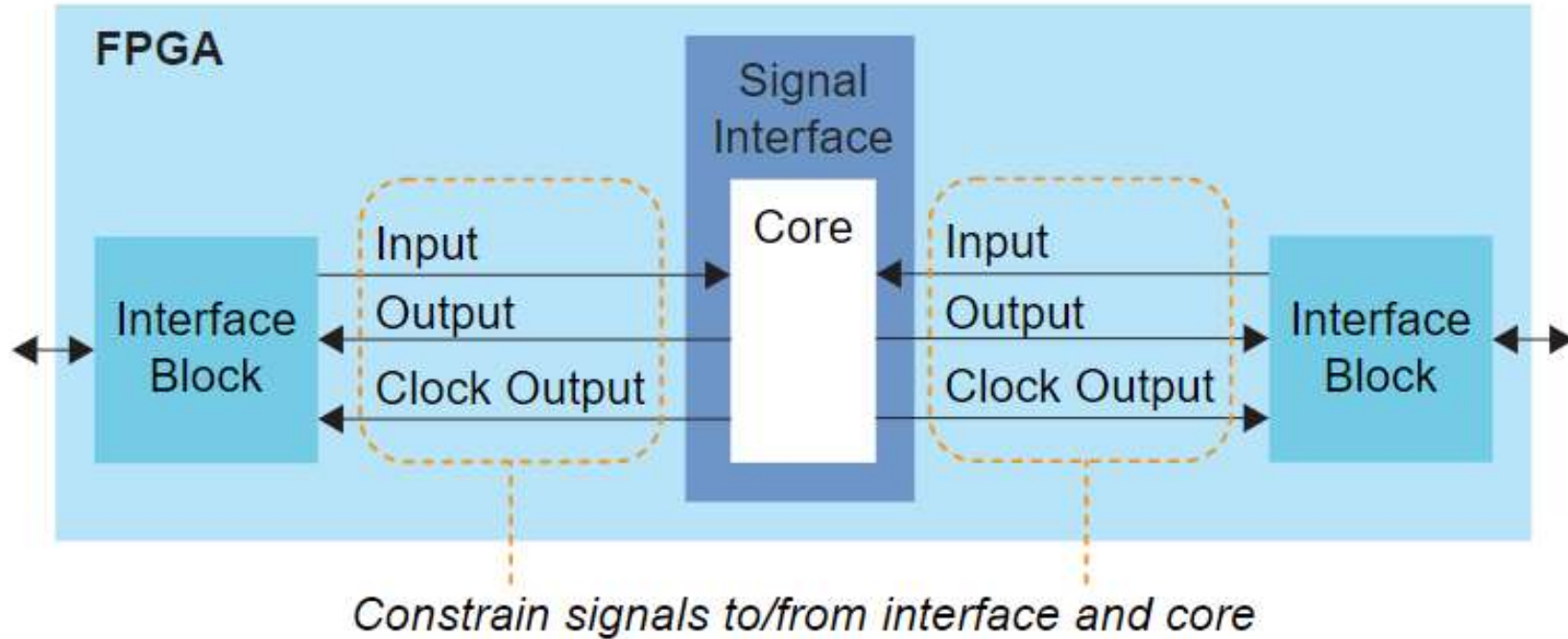


时序约束

- 针对CORE逻辑的约束
 - Interface Designer时序报告
 - Interface Designer时序约束文件
- 时钟约束
 - 定义时钟
 - 约束时钟之间的关系
 - 定义时钟的不确定性
- IO约束
 - 同步IO和非同步IO
 - 约束同步IO
 - 约束非同步IO
- 约束时序例外
 - 路径例外约束
 - 多周期路径约束
 - 路径延迟约束
- 建立SDC文件
 - 约束文件的基本要求
 - 建立和添加约束文件
 - SDC小知识



针对Core逻辑的时序约束



- 时序约束是面向Core逻辑的约束，这一点是和传统的FPGA最大的区别
- 需要将Interface当作外设器件来处理
 - 对Core逻辑来说，Interface就是外设。所以需要将Interface的时序信息导入针对Core的时序约束中
- 当在Interface designer中完成了接口设计后，工程输出文件夹(outflow)里会生成：
 - <项目名> pt_timing.rpt: 当前工程Interface时序报告
 - <项目名> pt.sdc: 当前工程的Interface 时序约束SDC文件



Interface Designer时序报告

```
22
23 ----- 2. GPIO Timing Report (begin) -----
24
25 Clock Network Delay:
26 =====
27
28 +-----+-----+-----+
29 | Clock Pin | Max (ns) | Min (ns) |
30 +-----+-----+-----+
31 | pin_clk   | 4.110    | 2.055    |
32 | pll_clkout| 4.110    | 2.055    |
33 +-----+-----+-----+
34
35 Clkout GPIO Configuration:
36 =====
37
38 +-----+-----+-----+-----+-----+
39 | Instance Name | Clock Pin | Parameter | Max (ns) | Min (ns) |
40 +-----+-----+-----+-----+-----+
41 | clkout        | pll_clkout| GPIO_CLK_OUT | 7.939    | 3.970    |
42 +-----+-----+-----+-----+-----+
43
44 Non-registered GPIO Configuration:
45 =====
46
47 +-----+-----+-----+-----+-----+
48 | Instance Name | Pin Name  | Parameter | Max (ns) | Min (ns) |
49 +-----+-----+-----+-----+-----+
50 | ext_clk       | ext_clk   | GPIO_IN   | 1.796    | 0.898    |
51 | in[0]         | in[0]     | GPIO_IN   | 1.396    | 0.698    |
52 | in[1]         | in[1]     | GPIO_IN   | 1.396    | 0.698    |
53 | pin_clk       | pin_clk   | GPIO_CLK_IN | 1.275    | 0.637    |
54 | out           | out       | GPIO_OUT  | 3.829    | 1.915    |
55 +-----+-----+-----+-----+-----+
56
57 Registered GPIO Configuration:
58 =====
59
60 +-----+-----+-----+-----+-----+-----+
61 | Instance Name | Clock Pin | Setup (ns) | Hold (ns) | Max Clock To Out (ns) | Min Clock To Out (ns) |
62 +-----+-----+-----+-----+-----+-----+
63 | in_reg[0]     | pin_clk   | -2.219     | 1.981     |                       |                       |
64 | in_reg[1]     | pin_clk   | -2.219     | 1.981     |                       |                       |
65 | out_reg       | pin_clk   |             |           | 8.649                 | 4.325                 |
66 +-----+-----+-----+-----+-----+-----+
67
68 ----- GPIO Timing Report (end) -----
```

- pt_timing.rpt
 - PLL Timing Report
 - 锁相环输出时钟的周期和相位信息
 - GPIO Timing Report
 - Clock Network Delay
 - GPIO_CLK_IN^①到 Core全局时钟网络的延迟
 - 锁相环输出到GPIO_CLK_OUT^②延迟
 - Clkout GPIO Configuration
 - GPIO_CLK_OUT^②输出延迟
 - Non-registered GPIO Configuration
 - 未使用IO寄存的GPIO延迟
 - Registered GPIO Configuration
 - Input REG对输入信号到FPGA管脚的Setup和Hold要求
 - Output REG输出到FPGA管脚的延迟信息
 - JTAG Timing Report
 - 使用Debugger的时候做相关时序约束使用
-

① GPIO_CLK_IN : gclk连接类型
② GPIO_CLK_OUT: clkout连接类型



Interface Designer时序约束文件

- <项目名>.pt.sdc是Interface Designer生成的针对Core的SDC模板
 - PLL Constraints
 - 提供锁相环输出时钟的完整定义
 - 用户根据Core逻辑具体应用，按实际需要增加时钟关系约束，虚拟时钟约束，生成时钟约束
 - GPIO Constraints
 - 提供GCLK输入的时钟定义模板，由用户根据实际应用计算后定义
 - Registered GPIO
 - 提供IO寄存器和Core之间输入输出延迟的完整约束
 - Non-registered GPIO
 - 提供未寄存GPIO和Core之间输入输出延迟约束模板，由用户根据实际应用和Interface时序报告计算以后填写
 - LVDS, MIPI硬核, MIPI DPHY, DDR控制器
 - 提供Interface和core之间输入输出延迟的完整约束



时序约束

- 针对CORE逻辑的约束
 - Interface Designer时序报告
 - Interface Designer时序约束文件
- 时钟约束
 - 定义时钟
 - 约束时钟之间的关系
 - 定义时钟的不确定性
- IO约束
 - 同步IO和非同步IO
 - 约束同步IO
 - 约束非同步IO
- 约束时序例外
 - 路径例外约束
 - 多周期路径约束
 - 路径延迟约束
- 建立SDC文件
 - 约束文件的基本要求
 - 建立和添加约束文件
 - SDC小知识



定义时钟

- 时钟源

- 基准时钟

- 锁相环PLL输出
 - 连接类型定义为gclk的全局时钟GPIO_CLK_IN

- 虚拟时钟

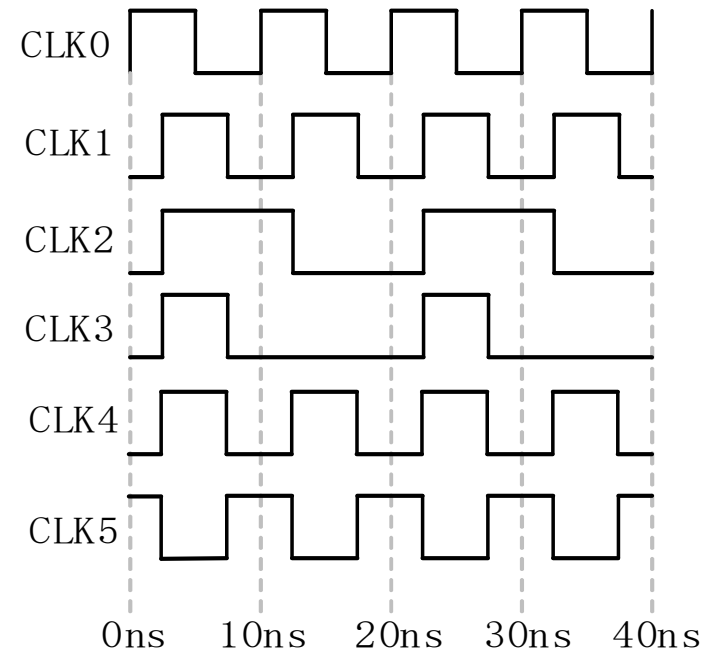
- 虚拟时钟是没有指定时序节点(端口)的时钟
 - 通常用在输入输出接口约束的时候

- 生成时钟

- 基准时钟生成的分频，倍频和反向等时钟

- SDC

- create_clock
 - create_generated_clock



CLK0 : create_clock -period 10 -name CLK0 [get_ports CLK0]

CLK1 : create_clock -period 10 -waveform {2.50 7.50} -name CLK1 [get_ports CLK1]

CLK2 : create_generated_clock -source CLK1 -divide_by 2 CLK2

CLK3 : create_clock -period 20 -waveform {2.50 7.50} -name CLK3 [get_ports CLK3]

CLK4 : create_clock -period 10 -waveform {2.50 7.50} -name CLK4

CLK5 : create_generated_clock -source CLK1 -invert 2 CLK5

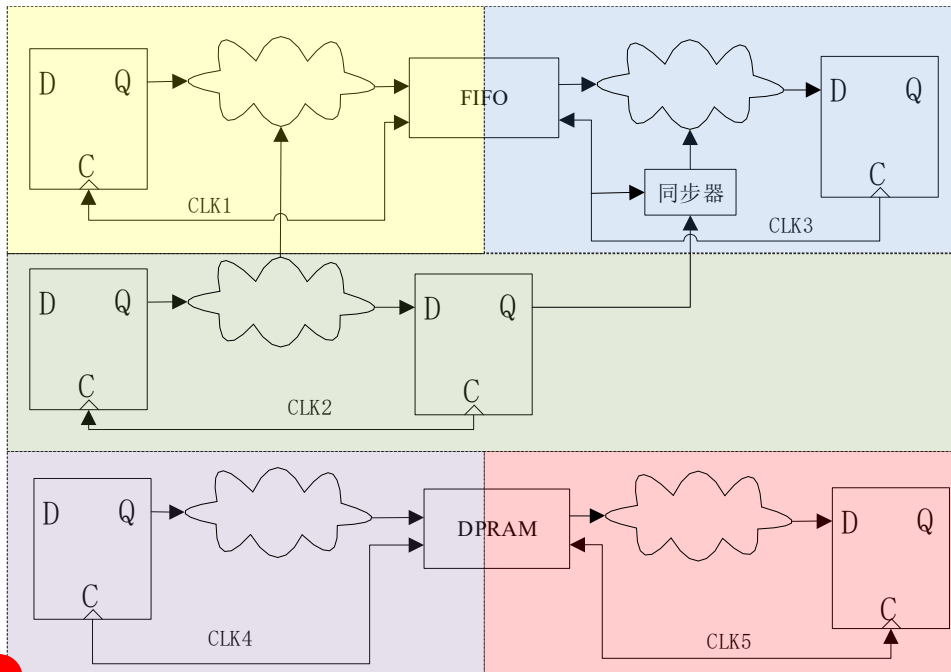
基准时钟 : CLK0,CLK1,CLK3

虚拟时钟 : CLK4, 不指定端口, 但必须加-name

生成时钟 : CLK2,CLK5, (PLL产生的输出时钟不需要使用生成时钟)

约束时钟之间的关系

- Efinity 默认所有的时钟都是有关联的
 - 时序引擎分析所有时钟之间的关系，并执行相关路径优化
 - 增加不必要的布局布线和时序分析运行时间
 - 造成时序收敛困难
 - 必须使用约束对没有相关性的时钟域进行隔离
- SDC
 - set_clock_groups
 - set_false_path



方法一：set_clock_groups

```
set_clock_groups -exclusive -group {CLK1 CLK2} -group {CLK3} -group {CLK4} -group {CLK5}
```

或

```
set_clock_groups -exclusive -group {CLK1 CLK2}
```

```
set_clock_groups -exclusive -group {CLK3}
```

```
set_clock_groups -exclusive -group {CLK4}
```

```
set_clock_groups -exclusive -group {CLK5}
```

方法二：set_false_path

```
set_false_path -from CLK1 -to CLK3 set_false_path -from CLK3 -to CLK1
```

```
set_false_path -from CLK1 -to CLK4 set_false_path -from CLK4 -to CLK1
```

```
set_false_path -from CLK1 -to CLK5 set_false_path -from CLK5 -to CLK1
```

```
set_false_path -from CLK2 -to CLK3 set_false_path -from CLK3 -to CLK2
```

```
set_false_path -from CLK2 -to CLK4 set_false_path -from CLK4 -to CLK2
```

```
set_false_path -from CLK2 -to CLK5 set_false_path -from CLK5 -to CLK2
```

```
set_false_path -from CLK3 -to CLK4 set_false_path -from CLK4 -to CLK3
```

```
set_false_path -from CLK3 -to CLK5 set_false_path -from CLK5 -to CLK3
```

```
set_false_path -from CLK4 -to CLK5 set_false_path -from CLK5 -to CLK4
```

定义时钟的不确定性

- Trion系列和钛金系列FPGA都有默认的时钟不确定性值(Clock Uncertainty)
 - 不同的型号有不同的默认值，可在时序报告文件里查看
 - <项目名>.timing.rpt
 - 如果SDC里没有定义时钟的不确定值，Efinity将使用默认的时钟不确定值来进行建立和保持时间分析
- 如果希望设计时序余量更大，可以通过约束增加时钟的不确定值
 - 可参照外部时钟源的Jitter参数，添加时钟的不确定性约束
 - 系统将按照【器件默认值+约束值】来执行时序分析
 - 增加时序收敛的难度
- SDC
 - set_clock_uncertainty

Trion T8建立时间的时钟不确定值默认是140ps，保持时间的时钟不确定值默认是50ps。如果在SDC文件加上以下两条约束：

```
set_clock_uncertainty -to clk -setup 0.06
```

```
set_clock_uncertainty -to clk -hold 0.05
```

则Efinity将使用200ps时钟不确定值进行建立时间时序分析，使用100ps时钟不确定值进行保持时间时序分析。



时序约束



易 灵 思

9/16/2021

- 针对CORE逻辑的约束
 - Interface Designer时序报告
 - Interface Designer时序约束文件
- 时钟约束
 - 定义时钟
 - 约束时钟之间的关系
 - 定义时钟的不确定性
- IO约束
 - 同步IO和非同步IO
 - 约束同步IO
 - 约束非同步IO
- 约束时序例外
 - 路径例外约束
 - 多周期路径约束
 - 路径延迟约束
- 建立SDC文件
 - 约束文件的基本要求
 - 建立和添加约束文件
 - SDC小知识

同步IO和非同步IO

- 同步IO

- 同步IO是指Interface里有输入输出寄存的IO连接
 - 输入路径起点是Interface
 - 输出路径终点是Interface
- 几乎所有的Interface同Core的连接方式都是同步IO连接
 - LVDS Serializer/ Deserializer mode
 - 有IO REG的GPIO
 - MIPI硬核
 - DDR硬核控制器
 - MIPI DPHY
 - HyperRAM
 - JTAG

- 非同步IO

- 非同步IO是指外设器件的信号直接穿过Interface的IO连接
 - LVDS X1 bypass 模式
 - 没有使用IO REG的GPIO



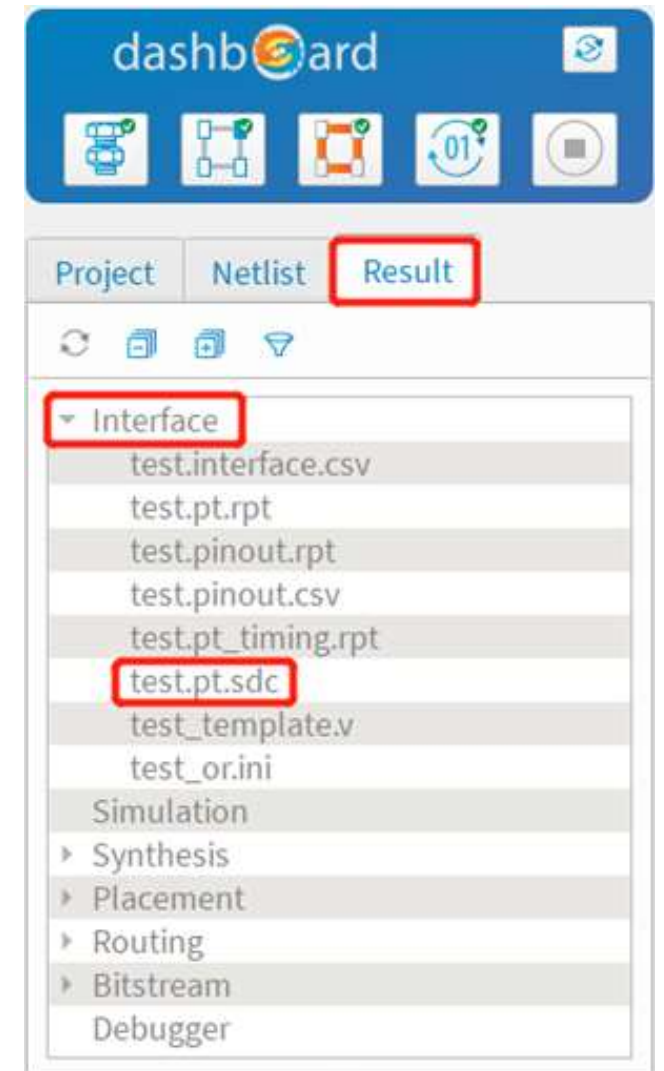
约束同步IO

- 同步IO约束由Efinity自动生成

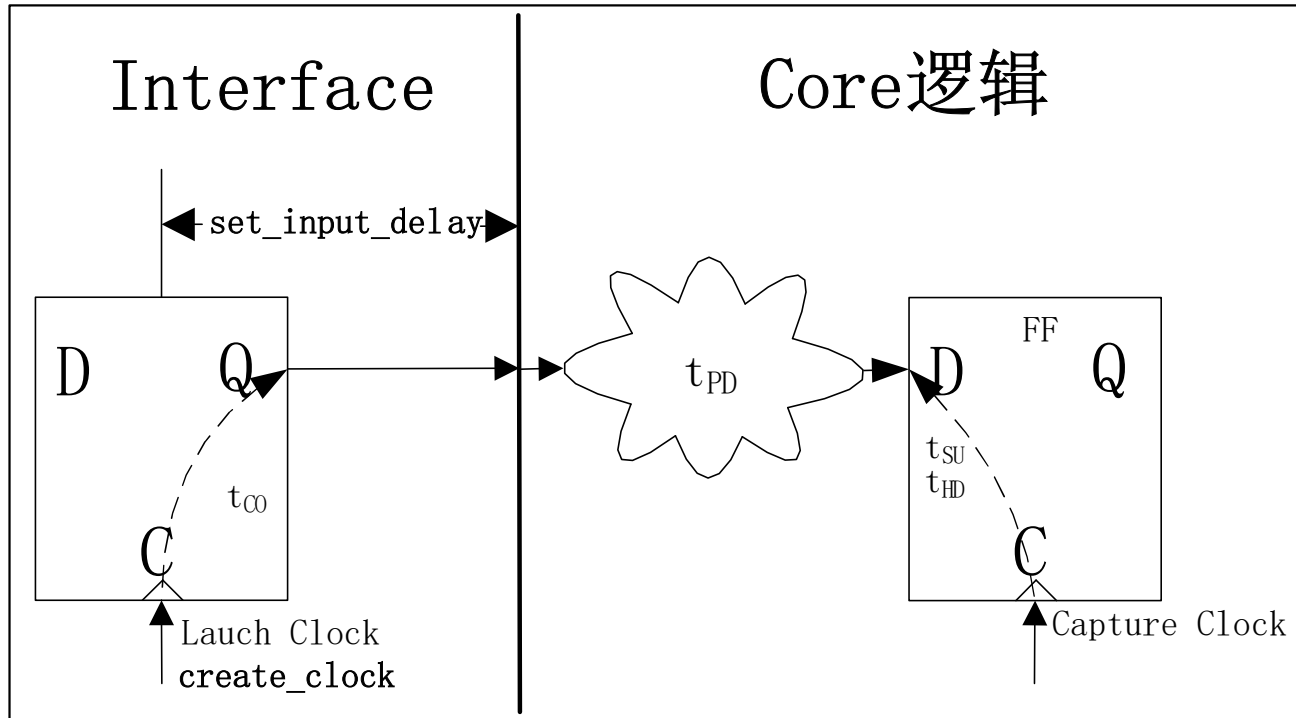
- Efinity的时序约束是针对Core逻辑的时序约束，Interface被当成外设器件
- Interface Designer知道Interface同Core之间时钟和数据的路径信息
 - 输入: 路径起点(Launch Path)是Interface
 - 输出: 路径终点(Capture Path)是Interface
- Efinity自动计算并生成准确的路径延迟信息

- 导入同步IO约束

- Efinity Dashboard->Result->Interface
- 打开文件: <项目名>.pt.sdc
- 复制set_input_delay和set_output_delay约束到工程SDC文件里
- 也可将<项目名>.pt.sdc文件直接拷贝到工程SDC存放的位置作为设计约束模板, 调整和添加完整约束以后直接使用
 - 当前工程文件夹->outflow
 - 不能直接设置成工程时序约束文件, 否则每次修改Interface designer该文件会被自动覆盖, 手动调整和添加的信息会丢失



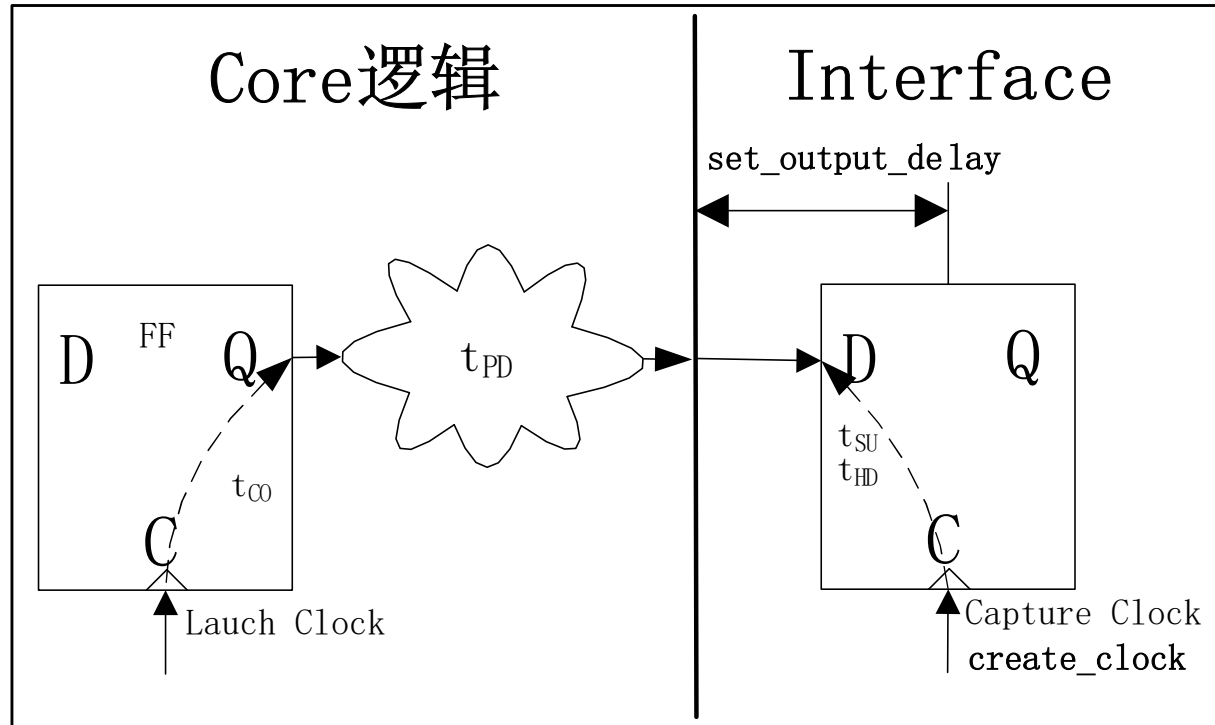
约束同步输入 set_input_delay



- 路径起点(Launch Path)是Interface
 - Efinity Dashboard->Result->Interface
 - 打开文件: <项目名>.pt.sdc
 - 复制set_input_delay约束到工程SDC文件里

```
2 # Efinity Interface Designer SDC
3 # Version: 2021.1.165.1.6
4 # Date: 2021-08-21 01:53
5
6 # Copyright (C) 2017 - 2021 Efinix Inc. All rights reserved.
7
8 # Device: T13F256
9 # Project: test
10 # Timing Model: C4 (final)
11
12 # PLL Constraints
13 #####
14 create_clock -period 5.00 pll_clk
15 create_clock -period 5.00 [get_ports {pin_clk}]
16
17 # GPIO Constraints
18 #####
19 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {ext_clk}]
20 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {ext_clk}]
21 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[0]}]
22 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[0]}]
23 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[1]}]
24 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[1]}]
25 set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[0]}]
26 set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[0]}]
27 set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[1]}]
28 set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[1]}]
29 # set_output_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {out}]
30 # set_output_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {out}]
31 set_output_delay -clock pin_clk -max -3.500 [get_ports {out_reg}]
32 set_output_delay -clock pin_clk -min -1.971 [get_ports {out_reg}]
```

约束同步输出 set_output_delay



- 路径终点(Capture Path)是Interface
 - Efinity Dashboard->Result->Interface
 - 打开文件: <项目名>.pt.sdc
 - 复制set_output_delay约束到工程SDC文件里

```
2 # Efinity Interface Designer SDC
3 # Version: 2021.1.165.1.6
4 # Date: 2021-08-21 01:53
5
6 # Copyright (C) 2017 - 2021 Efinix Inc. All rights reserved.
7
8 # Device: T13F256
9 # Project: test
10 # Timing Model: C4 (final)
11
12 # PLL Constraints
13 #####
14 create_clock -period 5.00 pll_clk
15 create_clock -period 5.00 [get_ports {pin_clk}]
16
17 # GPIO Constraints
18 #####
19 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {ext_clk}]
20 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {ext_clk}]
21 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[0]}]
22 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[0]}]
23 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[1]}]
24 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[1]}]
25 set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[0]}]
26 set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[0]}]
27 set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[1]}]
28 set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[1]}]
29 # set_output_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {out}]
30 # set_output_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {out}]
31 set_output_delay -clock pin_clk -max -3.500 [get_ports {out_reg}]
32 set_output_delay -clock pin_clk -min -1.971 [get_ports {out_reg}]
33
```

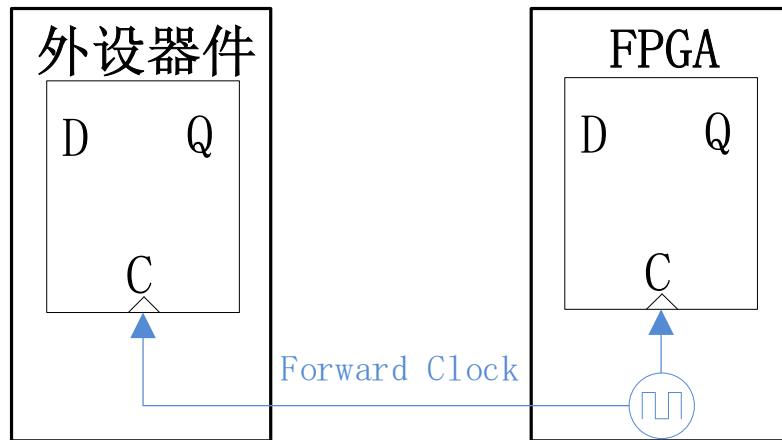
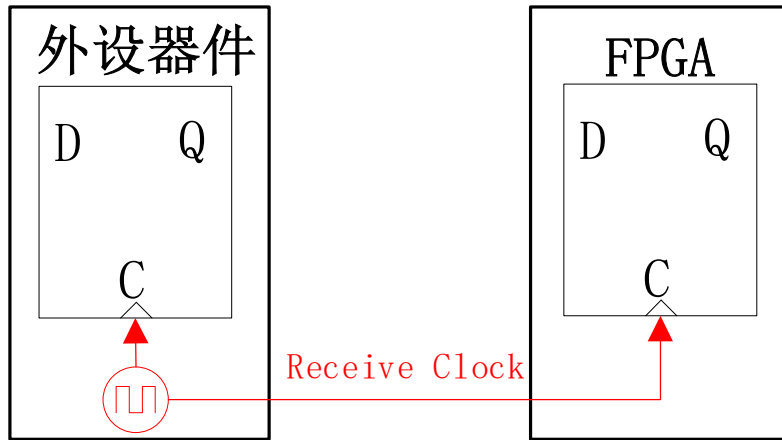


时序约束

- 针对CORE逻辑的约束
 - Interface Designer时序报告
 - Interface Designer时序约束文件
- 时钟约束
 - 定义时钟
 - 约束时钟之间的关系
 - 定义时钟的不确定性
- IO约束
 - 同步IO和非同步IO
 - 约束同步IO
 - 约束非同步IO
- 约束时序例外
 - 路径例外约束
 - 多周期路径约束
 - 路径延迟约束
- 建立SDC文件
 - 约束文件的基本要求
 - 建立和添加约束文件
 - SDC小知识



Receive clock和Forward clock



- Receive clock
 - 由外设器件产生并发起，经过GPIO输入到FPGA
 - FPGA GPIO连接属性设置为gclk的输入管脚——GPIO_IN_CLK
 - GPIO_IN_CLK延迟表现为管脚经过IO Buffer到全局时钟网络的组合逻辑延迟
- Forward clock
 - 由FPGA产生并发起，经过GPIO clkout模式输出到外设器件
 - GPIO_CLK_OUT延迟表现为从FPGA时钟树经过IO buffer到管脚组合逻辑延迟
 - 输出源同步时钟设计，推荐使用普通DDIO的方式输出时钟，所以实际应用中Forward clock的非同步IO约束的应用场景很少
- 非同步IO延迟
 - 没有使用IO Reg
 - 输入信号：从输入管脚经过IO Buffer到Core的延迟，被看作是组合逻辑延迟
 - 输出和输出使能信号：从Core经过IO buffer输出到FPGA管脚的延迟，被看作组合逻辑延迟
- 对于非同步IO，需要将板级延迟和非同步IO延迟一起考虑，计算后得出针对Core的输入输出延迟约束



约束非同步IO

- 导入非同步IO约束

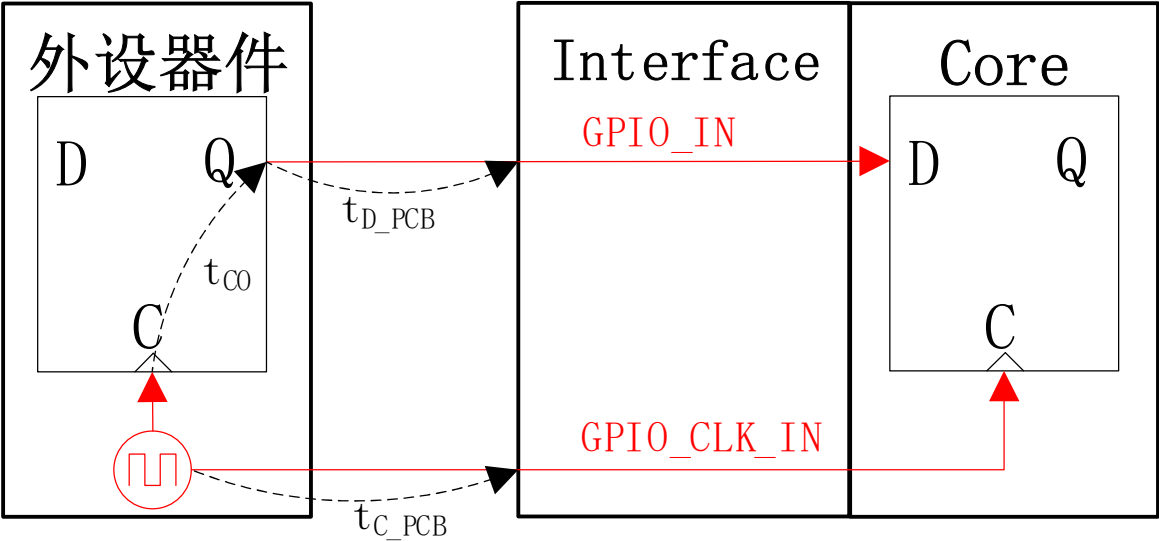
- **Step1** 判断非同步IO约束的模式
 - input receive, input forward, output receive或者output forward
- **Step2** 在Interface Designer时序报告文件<项目名>.pt_timing.rpt中找到对应路径的最小最大时序参数
- **Step3** 使用不同模式对应的公式计算延迟
- **Step4** 将约束添加到SDC文件里
 - Efinity在<项目名>.pt.sdc文件里提供非同步IO约束的模板
 - 将计算后的delay值填入模板中即可

```
33 Non-registered GPIO Configuration:
34 =====
35                                     <project name>.pt_timing.rpt
36 +-----+-----+-----+-----+-----+
37 | Instance Name | Pin Name | Parameter | Max (ns) | Min (ns) |
38 +-----+-----+-----+-----+-----+
39 | ext_clk      | ext_clk  | GPIO_IN   | 1.796     | 0.898     |
40 | in[0]        | in[0]    | GPIO_IN   | 1.396     | 0.698     |
41 | in[1]        | in[1]    | GPIO_IN   | 1.396     | 0.698     |
42 | pin_clk      | pin_clk  | GPIO_CLK_IN | 1.275     | 0.637     |
43 | out          | out      | GPIO_OUT  | 3.829     | 1.915     |
44 +-----+-----+-----+-----+-----+
```

```
17 # GPIO Constraints                                     <project name>.pt.sdc
18 #####
19 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {ext_clk}]
20 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {ext_clk}]
21 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[0]}]
22 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[0]}]
23 # set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[1]}]
24 # set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[1]}]
25 set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[0]}]
26 set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[0]}]
27 set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[1]}]
28 set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[1]}]
29 # set_output_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {out}]
30 # set_output_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {out}]
31 set_output_delay -clock pin_clk -max -3.500 [get_ports {out_reg}]
32 set_output_delay -clock pin_clk -min -1.971 [get_ports {out_reg}]
33
```



约束Input Receive Clock Delay



Non-registered GPIO Configuration:

Instance Name	Pin Name	Parameter	Max (ns)	Min (ns)
clkin	clkin	GPIO_CLK_IN	1.954	0.526
din	din	GPIO_IN	1.954	0.526
dout	dout	GPIO_OUT	4.246	1.081

• 计算过程:

<max board constraint> = $t_{CO} + t_{D_PCB} - t_{C_PCB}$ (假设为4ns)

<min board constraint> = $t_{CO} + t_{D_PCB} - t_{C_PCB}$ (假设为2ns)

<max calculation> = $4 + 1.954 - 1.954 = 4$

<min calculation> = $2 + 0.526 - 0.526 = 2$

• SDC

set_input_delay -clock clkin -max 4 [get_ports {din}]

set_input_delay -clock clkin -min 2 [get_ports {din}]

• SDC模板

set_input_delay -clock <clock> -max <max calculation> <ports>

set_input_delay -clock <clock> -min <min calculation> <ports>

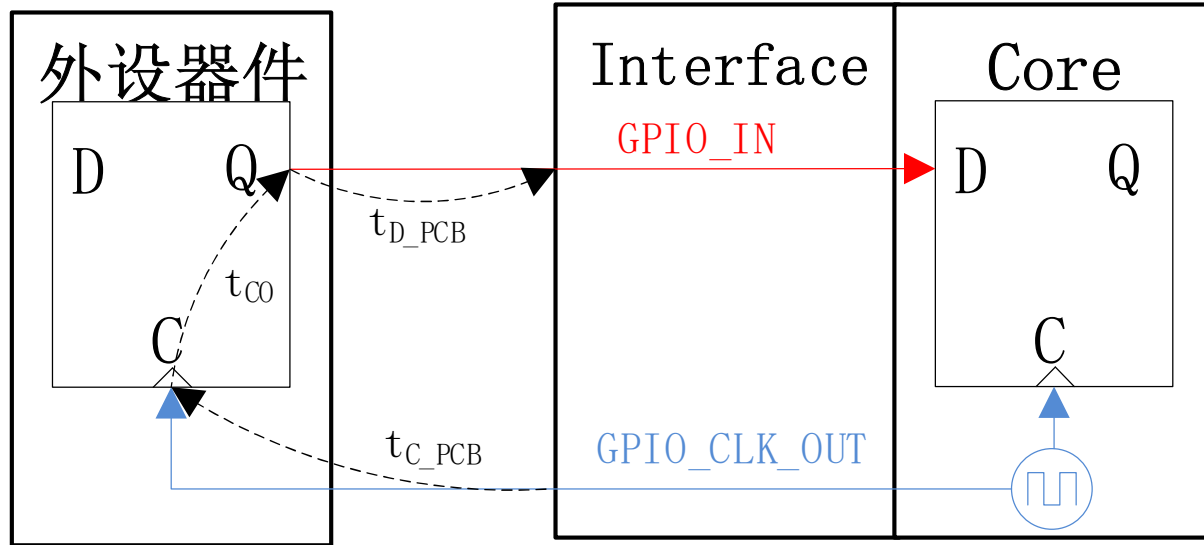
• 计算公式

<max calculation> = <max board constraint> + GPIO_INmax - GPIO_CLK_INmax

<min calculation> = <min board constraint> + GPIO_INmin - GPIO_CLK_INmin



约束Input Forward Clock Delay



• SDC模板

`set_input_delay -clock <clock> -max <max calculation> <ports>`

`set_input_delay -clock <clock> -min <min calculation> <ports>`

• 计算公式

$\text{<max calculation>} = \text{<max board constraint>} + \text{GPIO_INmax} + \text{GPIO_CLK_OUTmax}$

$\text{<min calculation>} = \text{<min board constraint>} + \text{GPIO_INmin} + \text{GPIO_CLK_OUTmin}$

Clkout GPIO Configuration:

=====

Instance Name	Clock Pin	Parameter	Max (ns)	Min (ns)
clkout	pllclk0	GPIO_CLK_OUT	6.834	4.401

Non-registered GPIO Configuration:

=====

Instance Name	Pin Name	Parameter	Max (ns)	Min (ns)
clkin	clkin	GPIO_CLK_IN	1.954	0.526
din	din	GPIO_IN	1.954	0.526
dout	dout	GPIO_OUT	4.246	1.081

• 计算过程:

$\text{<max board constraint>} = t_{CO} + t_{D_PCB} + t_{C_PCB}$ (假设为4ns)

$\text{<min board constraint>} = t_{CO} + t_{D_PCB} + t_{C_PCB}$ (假设为2ns)

$\text{<max calculation>} = 4 + 1.954 + 6.834 = 12.788$

$\text{<min calculation>} = 2 + 0.526 + 4.401 = 6.927$

• SDC

`set_input_delay -clock clkout -max 12.788 [get_ports {din}]`

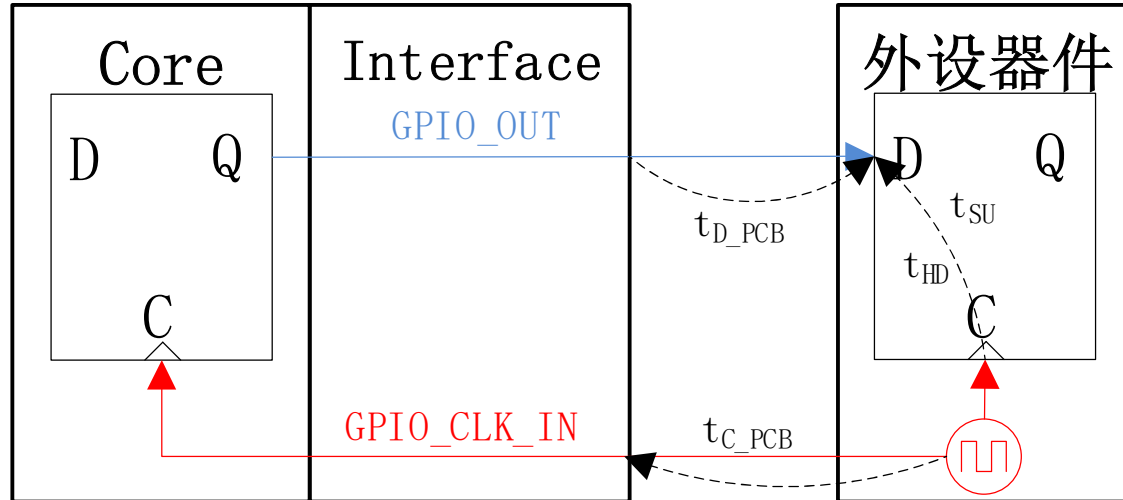
`set_input_delay -clock clkout -min 6.927 [get_ports {din}]`



易 灵 思

9/16/2021

约束Output Receive Clock Delay



- SDC模板

`set_output_delay -clock <clock> -max <max calculation> <ports>`

`set_output_delay -clock <clock> -min <min calculation> <ports>`

- 计算公式

$\text{<max calculation>} = \text{<max board constraint>} + \text{GPIO_OUTmax} + \text{GPIO_CLK_INmax}$

$\text{<min calculation>} = \text{<min board constraint>} + \text{GPIO_OUTmin} + \text{GPIO_CLK_INmin}$

Non-registered GPIO Configuration:

=====

Instance Name	Pin Name	Parameter	Max (ns)	Min (ns)
clk_in	clk_in	GPIO_CLK_IN	1.954	0.526
din	din	GPIO_IN	1.954	0.526
dout	dout	GPIO_OUT	4.246	1.081

- 计算过程:

$\text{<max board constraint>} = t_{D_PCB} + t_{C_PCB} + t_{SU}$ (假设为4ns)

$\text{<min board constraint>} = t_{D_PCB} + t_{C_PCB} - t_{HD}$ (假设为2ns)

$\text{<max calculation>} = 4 + 4.246 + 1.954 = 10.2$

$\text{<min calculation>} = 2 + 1.081 + 0.526 = 3.607$

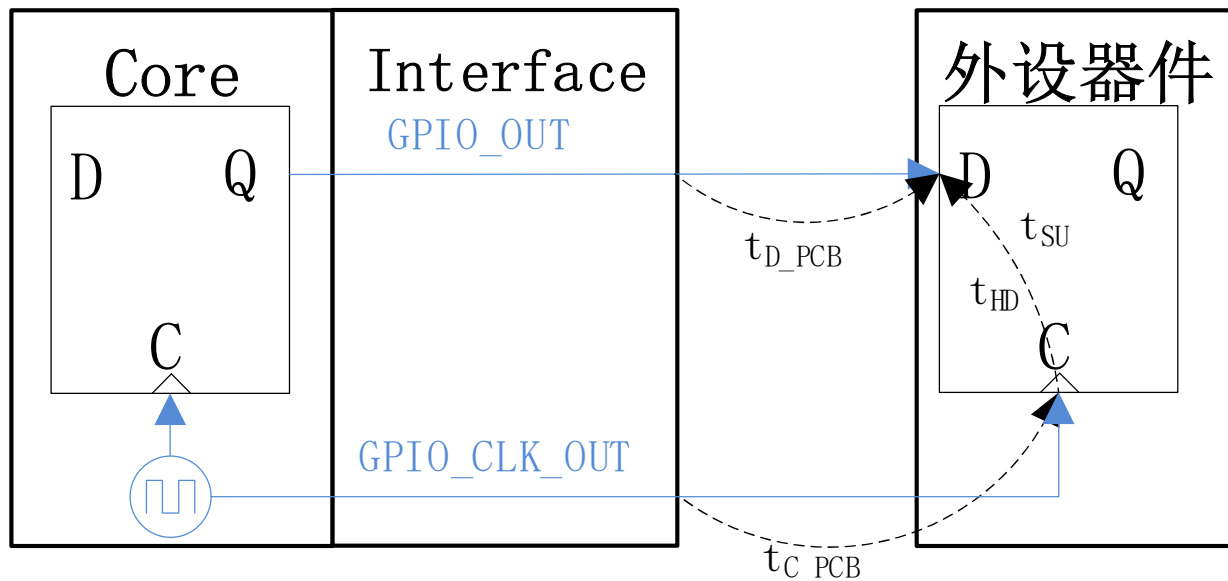
- SDC

`set_output_delay -clock clk_in -max 10.2 [get_ports {dout}]`

`set_output_delay -clock clk_in -min 3.607 [get_ports {dout}]`



约束Output Forward Clock Delay



• SDC模板

```
set_output_delay -clock <clock> -max <max calculation> <ports>
```

```
set_output_delay -clock <clock> -min <min calculation> <ports>
```

• 计算公式

<max calculation> = <max board constraint> + GPIO_OUTmax - GPIO_CLK_OUTmax

<min calculation> = <min board constraint> + GPIO_OUTmin - GPIO_CLK_OUTmin

Clkout GPIO Configuration:

=====

Instance Name	Clock Pin	Parameter	Max (ns)	Min (ns)
clkout	pllclk0	GPIO_CLK_OUT	6.834	4.401

Non-registered GPIO Configuration:

=====

Instance Name	Pin Name	Parameter	Max (ns)	Min (ns)
clkin	clkin	GPIO_CLK_IN	1.954	0.526
din	din	GPIO_IN	1.954	0.526
dout	dout	GPIO_OUT	4.246	1.081

• 计算过程:

<max board constraint> = $t_{D_PCB} - t_{C_PCB} + t_{SU}$ (假设为4ns)

<min board constraint> = $t_{D_PCB} - t_{C_PCB} - t_{HD}$ (假设为2ns)

<max calculation> = $4 + 4.246 - 6.834 = 1.412$

<min calculation> = $2 + 1.081 - 4.401 = -1.32$

• SDC

```
set_output_delay -clock clkout -max 1.412 [get_ports {dout}]
```

```
set_output_delay -clock clkout -min -1.32 [get_ports {dout}]
```



易 灵 思

9/16/2021

IO约束-小节

- 同步IO和非同步IO的区别
- 同步IO的约束方法
 - 定义时钟
 - 拷贝<项目名>.pt.sdc
- 非同步IO的约束方法
 - 定义时钟
 - 根据<项目名>.pt_timing.rpt和板级延迟参数计算总延迟
 - 参考<项目名>.pt.sdc非同步IO约束模板
 - 约束Input Receive Clock Delay
 - 约束Input Forward Clock Delay
 - 约束Output Receive Clock Delay
 - 约束Output Forward Clock Delay



时序约束

- 针对CORE逻辑的约束
 - Interface Designer时序报告
 - Interface Designer时序约束文件
- 时钟约束
 - 定义时钟
 - 约束时钟之间的关系
 - 定义时钟的不确定性
- IO约束
 - 同步IO和非同步IO
 - 约束同步IO
 - 约束非同步IO
- 约束时序例外
 - 路径例外约束
 - 多周期路径约束
 - 路径延迟约束
- 建立SDC文件
 - 约束文件的基本要求
 - 建立和添加约束文件
 - SDC小知识



路径例外约束

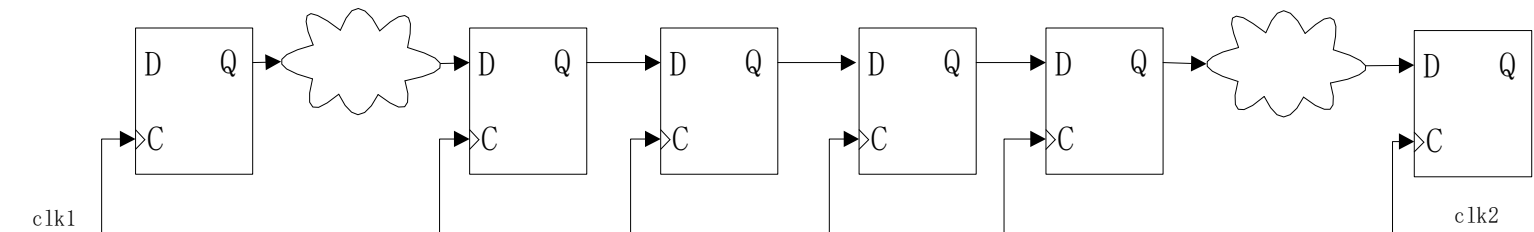
- `set_false_path`
 - 可用于切断时钟域之间的关联，使时序引擎不分析异步的发起时钟和捕获时钟之间的关联，等效于`set_clock_groups`约束。在这种场景下，我们推荐使用`set_clock_groups`。
 - `set_false_path`在需要进行寄存器或者输入输出的例外约束场景更加有用。或者用在只需要切断两个不同时钟之间某一个方向路径的时候。
 - Efinity不支持端口到端口的路径例外约束，也就是说在使用`set_false_path`的时候，`from`或者`to`至少有一个是时钟。



路径例外约束

- **set_false_path**

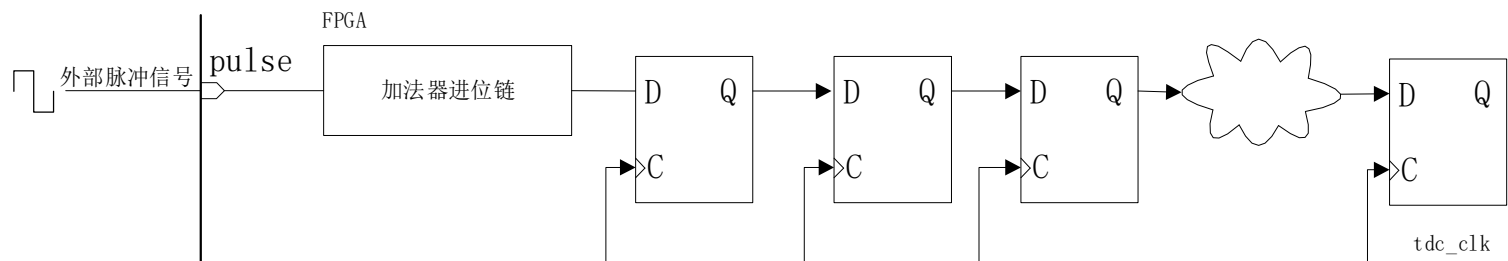
- 可用于切断时钟之间的关联，使时序引擎不分析异步的发起时钟和捕获时钟之间的时序关系，等效于set_clock_groups约束。这种应用场景，我们推荐使用set_clock_groups。
- set_false_path在需要针对寄存器或者输入输出端口进行路径例外约束的时候更加有用。或者用在只需要切断两个不同时钟之间某一个方向的路径的时候。
- Efinity不支持端口到端口的路径例外约束，也就是说在使用set_false_path的时候，from或者to至少有一个是时钟。



单向跨时钟路径例外约束应用案例

clk2频率远高于clk1，利用多级触发器消除亚稳态。

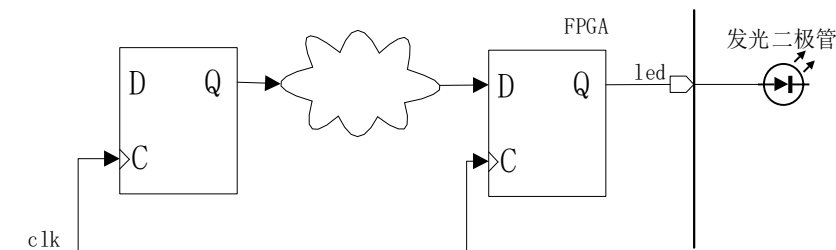
set_false_path -from clk1 -to clk2



端口到时钟的路径例外约束案例

TDC应用，被测试脉冲同测试算法时钟tdc_clk没有时钟同步关系，也不需要进行时序分析。

set_false_path -from [get_ports pulse] -to tdc_clk



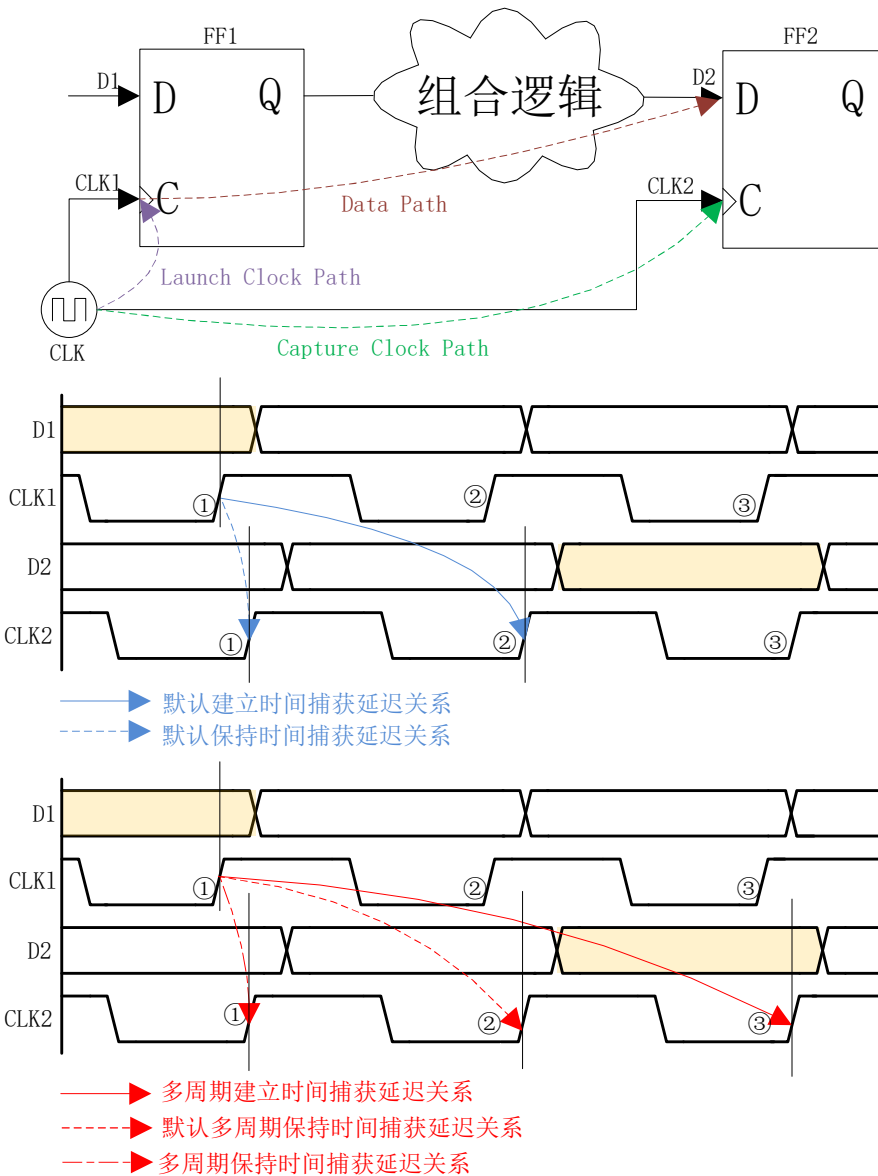
时钟到端口的路径例外约束案例

常见的LED闪灯设计，FPGA输出同LED没有任何时钟同步关系，也不需要进行时序分析。

set_false_path -from clk -to [get_ports led]



多周期路径约束



- 时序引擎默认的Setup和Hold分析，是基于单周期捕获的分析。即FF1发出的数据，到达FF2必须在下一个时钟边沿被正确采样并满足Setup和Hold最低要求（见P11 同步时序系统Fmax计算原理）
- 但是在有些设计中，组合逻辑级数很多并且无法采用流水线截短，路径终点的FF无法保证在下一个时钟边沿被正确采样，这时候我们就需要采用多周期路径约束
 - $\text{Data Path} - (\text{Capture Clock Path} - \text{Launch Clock Path}) > T$
- 多周期路径约束告诉时序引擎，要求布局布线产生的延迟满足FF2在多个周期以后的某时钟边沿正确采样数据，并正确分析时序
- Efinity暂时只能支持基于整个时钟域的多周期路径约束
- 暂时不支持寄存器到寄存器的多周期路径约束

- 左图中，数据D1从FF1被CLK1打出后经过Data Path变为D2，到达目的FF2的D端被CLK2采样
- 数据相对时钟的延迟超过一个周期T，只能被CLK2对应CLK1之后的第2个上升沿采样
- 要保证这样一个设计时序正确，需要多周期约束
- SDC:
 - 1. `set_multicycle_path -setup 2 -from CLK1 -to CLK2`
隐含保持时间捕获延迟关系（红色虚线）
 - 2. `set_multicycle_path -hold 1 -from CLK1 -to CLK2`
将保持时间捕获关系向前移动一个周期（从红色虚线移动到红色点画线）



延迟约束

- 用在需要用自定义的延迟覆盖由create_clock产生的默认时序关系的时候
- Efinity暂时只能支持时钟到时钟的最大最小延迟约束

```
set_max_delay -from <clock> -to <clock> <delay>
```

```
set_min_delay -from <clock> -to <clock> <delay>
```



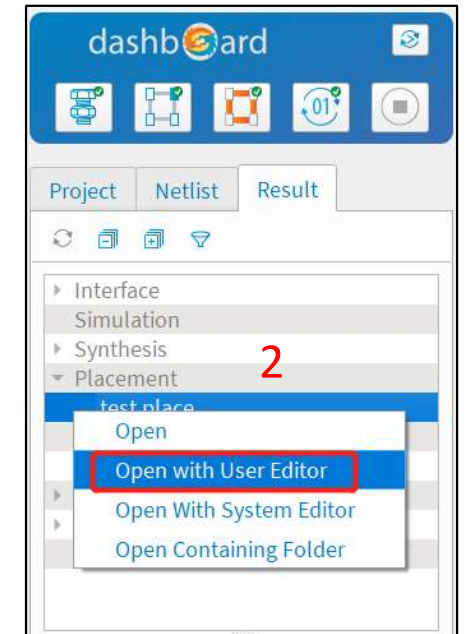
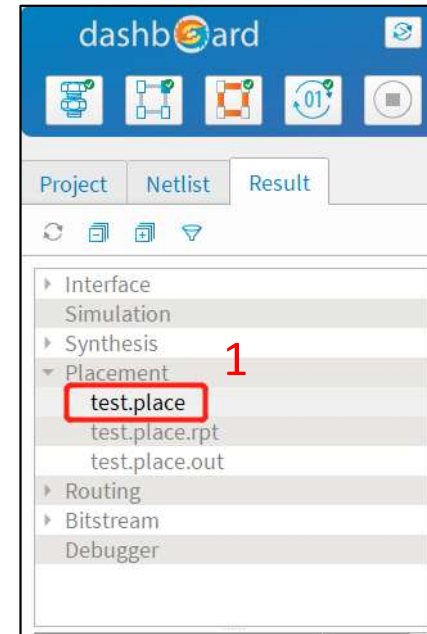
时序约束

- 针对CORE逻辑的约束
 - Interface Designer时序报告
 - Interface Designer时序约束文件
- 时钟约束
 - 定义时钟
 - 约束时钟之间的关系
 - 定义时钟的不确定性
- IO约束
 - 同步IO和非同步IO
 - 约束同步IO
 - 约束非同步IO
- 约束时序例外
 - 路径例外约束
 - 多周期路径约束
 - 路径延迟约束
- 建立SDC文件
 - 约束文件的基本要求
 - 建立和添加约束文件
 - SDC小知识



约束文件的基本要求

- 约束之间有相互依赖关系，书写顺序非常重要
 - 基准时钟 create_clock
 - 虚拟时钟 create_clock
 - 生成时钟 create_generated_clock
 - 时钟之间的关系 set_clock_groups
 - 输入输出延迟 set_input_delay/set_output_delay
 - 约束时序例外
 - Set_false_path
 - Set_max_delay/set_min_delay
 - Set_multicycle_path
- 语法必须正确，否则会被软件忽略
 - 约束对参数书写顺序有要求
 - 如果怀疑SDC没有生效，编译完成后软件会有告警提示
 - <项目名>.place.out
- 时序约束文件SDC，至少需要一个时钟定义
 - 如果没有定义SDC文件，软件将默认所有时钟为1ns



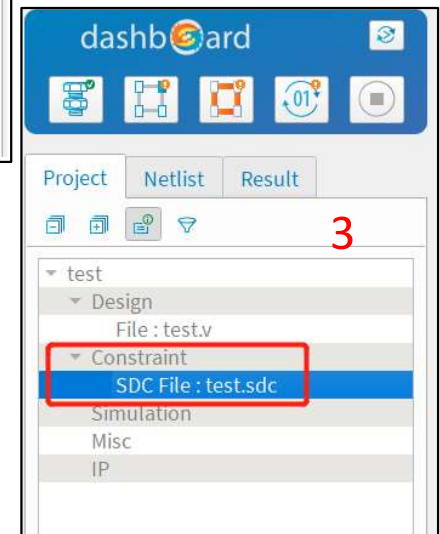
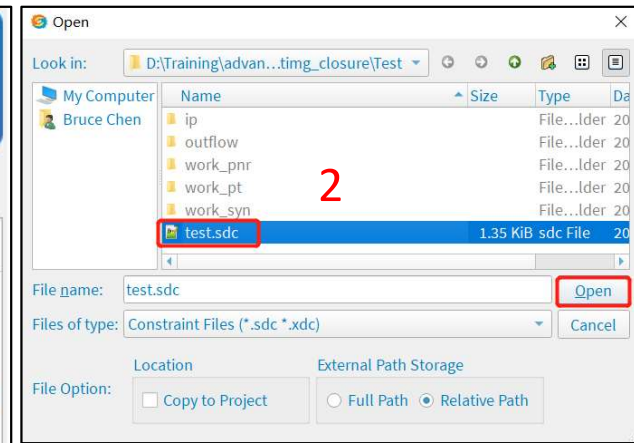
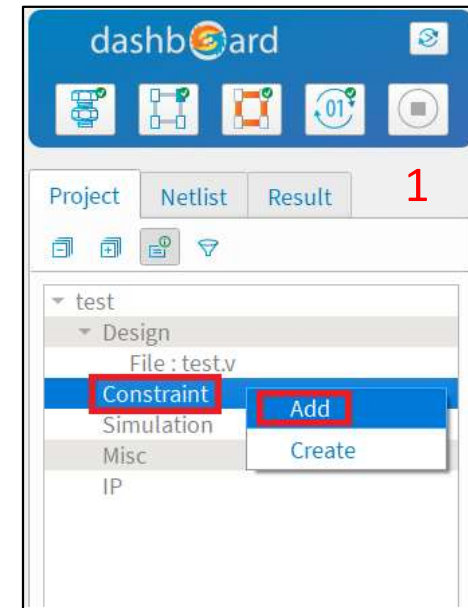
```
66 WARNING(1):
67 1 Error(s) found in the SDC file D:/Training/advanced_training/timg_closure/Test/
68 test.sdc
69 Error processing line 15:
70 missing close-brace
71 while executing
72 "create_clock -period 5.00 [get_ports {"
73
74 SDC file 'D:/Training/advanced_training/timg_closure/Test/test.sdc' blank or does
75 not contain valid constraint or not found.
76 Using default timing constraint of 1 ns.
```

3 <项目名>.place.out



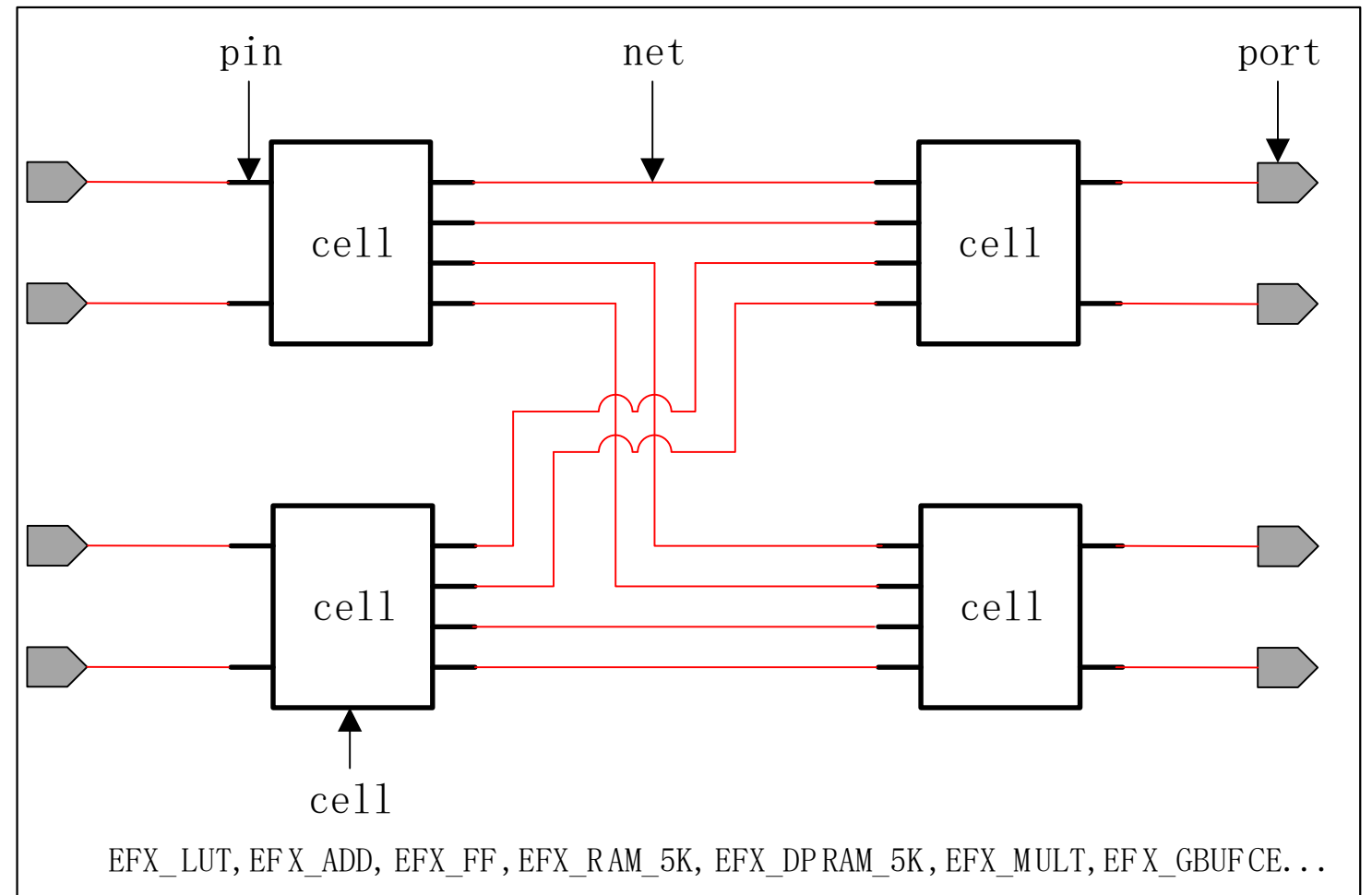
建立和添加约束文件

- 为当前工程编写时序约束文件
 - 使用任意文本编辑器新建扩展名为sdc的文本文件
 - 拷贝Interface Designer产生的<项目名>.pt.sdc中的内容作为时序约束模板
 - PLL输出时钟的完整定义
 - 所有同步IO完整约束
 - DDIO, IO REG
 - LVDS
 - DDR3
 - MIPI
 - ...
 - 使用<项目名>.pt_timing.rpt中的数据, 计算并填写模板中非同步IO的约束值
 - LVDS X1 bypass mode
 - 未寄存的GPIO
 - 根据实际需要添加时钟定义, 生成时钟定义
 - create_clock
 - 根据实际需要添加时钟uncertainty约束和时钟关系约束
 - set_clock_uncertainty
 - set_clock_groups
 - 根据工程需要添加时序例外的路径约束
 - set_false_path
 - set_multicycle_path
- 将sdc文件添加到工程里



SDC小知识

- # 注释符 * 通配符
- \ 换行符, 表示在下一行续写
- [] 对象符 {} 变量符
 - [get_pins {pin1 pin2 pin3}]
 - [get_ports {port1 port2 port3}]
 - [get_nets {nets1 nets2 nets3}]
 - [get_clocks {clock1 clock2 clock3}]
 - [get_cells {cell1 cell2 cell3}]
- 约束对象以综合后网表中的对象命名
 - 使用TCL对象检索命令可以检索网表中正确的命名
- 对象检索命令
 - all_inputs 检索所有的输入
 - all_outputs 检索所有的输出
 - all_clocks 检索所有的时钟
 - all_registers 检索所有的寄存器
 - get_ports 检索指定的端口
 - get_cells 检索指定的单元
 - get_pins 检索指定的管脚
 - get_clocks 检索指定的时钟
 - get_nets 检索指定的网络



主要内容

1. 时序约束
2. 时序优化策略

时序优化策略

- 综合策略
- 布局布线策略
- Seed Sweep
- Optimization Sweep

综合策略

综合策略名称	功能描述	选项	设置项功能描述
mode	综合模式选择	speed	速度优化（默认）
		area	面积优化，减少资源消耗，但是降低f _{MAX} 。
		area2	针对大位宽多路选择器进行优化，兼顾速度优化并且减少大位宽选择器使用的LUT4数量。综合器判断设计里的大位宽选择器，使用LUT4级联的方式以较少的LUT4实现。使用这个选项有可能增加逻辑级数降低f _{MAX} ，但不是必然。
max_ram	BRAM数量限制	-1	不对BRAM的使用进行限制（默认）
		0	禁止使用BRAM
		n	设置可使用的数量
max_mult	乘法器数量限制	-1	不对乘法器的使用进行限制（默认）
		0	禁止使用BRAM
		n	设置可使用的数量
infer-clk-enable	时钟使能推断	0	禁止综合器自动推断时钟使能
		1	低
		2	中
		3	高（默认）

综合策略

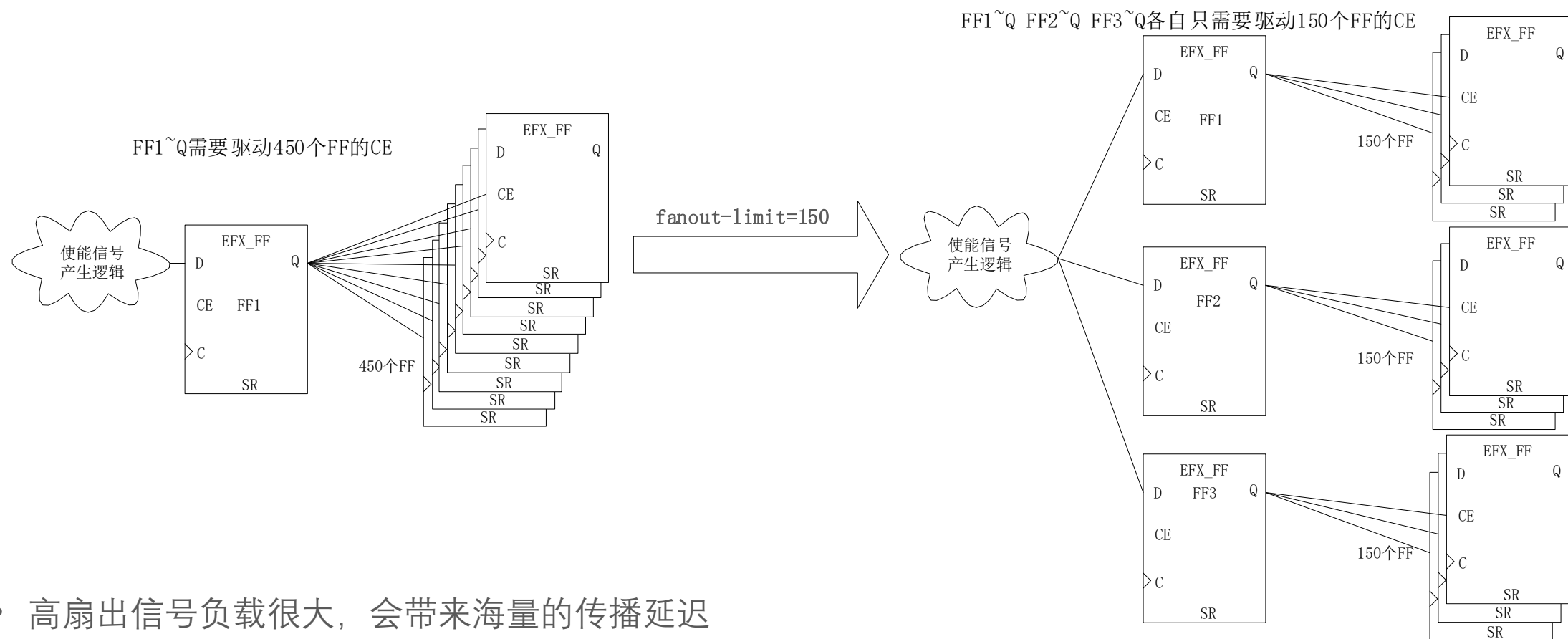
综合策略名称	功能描述	选项	设置项功能描述
infer-sync-set-reset	同步置位复位推断	0	禁止同步置位复位信号推断
		1	打开同步置位复位信号推断（默认）
fanout-limit	信号扇出数量限制	0	不限制扇出数量（默认）
		n	限制信号可扇出的最大数量，超过限制综合器自动复制高扇出信号源；一般高扇出信号是逻辑产生的使能和复位信号；在RTL设计的时候，高扇出信号推荐使用寄存器输出。
seq_opt	顺序优化	0	禁止优化
		1	打开优化（默认） 综合器分析寄存器所有可能的状态顺序，并按等效的顺序合并信号。总的来说，这是一个面积优化的选项，能有效减少LUT的使用数量。在很多情况下，经过顺序优化以后的电路会更有效率， f_{MAX} 通常也会变好。
bram_output_regs_packing	BRAM输出寄存器打包	0	禁止使用BRAM输出寄存器
		1	允许综合器把同BRAM输出直接相关连的寄存器综合为BRAM的输出寄存器。（默认）
retiming	retiming优化	0	禁止优化（默认）
		1	打开优化 综合器分析相关联路径之间的逻辑关系，在不影响功能的前提下移动组合逻辑之间的寄存器，平衡各级寄存器之间的组合逻辑级数，达到降低组合逻辑级数的效果。对 f_{MAX} 有一定帮助。

综合策略

综合策略名称	功能描述	选项	设置项功能描述
blast_const_operand_adders	常数操作数优化	0	禁止优化（默认）
		1	打开优化 如果运算的某操作数为常数，综合器将其综合为组合逻辑。
dsp-mac-packing	DSP乘累加打包	0	禁止优化
		1	打开优化（默认） 允许DSP乘累加打包，综合器将符合MAC描述的加法和乘法运算打包用DSP实现。（钛金）
dsp-input-regs-packing mult_input_regs_packing	DSP输入寄存器打包 乘法器输入寄存器打包	0	禁止优化
		1	打开优化（默认） 允许综合器把同DSP输入直接有关的寄存器综合为DSP的输入寄存器。（钛金） 允许综合器把同乘法器输入直接有关的寄存器综合为乘法器的输入寄存器。（Trion）
dsp-output-regs-packing mult_output_regs_packing	DSP输出寄存器打包 乘法器输出寄存器打包	0	禁止优化
		1	打开优化（默认） 允许综合器把同DSP输出直接相关的寄存器综合为DSP的输出寄存器。（钛金） 允许综合器把同乘法器输出直接相关的寄存器综合为乘法器的输出寄存器。（Trion）

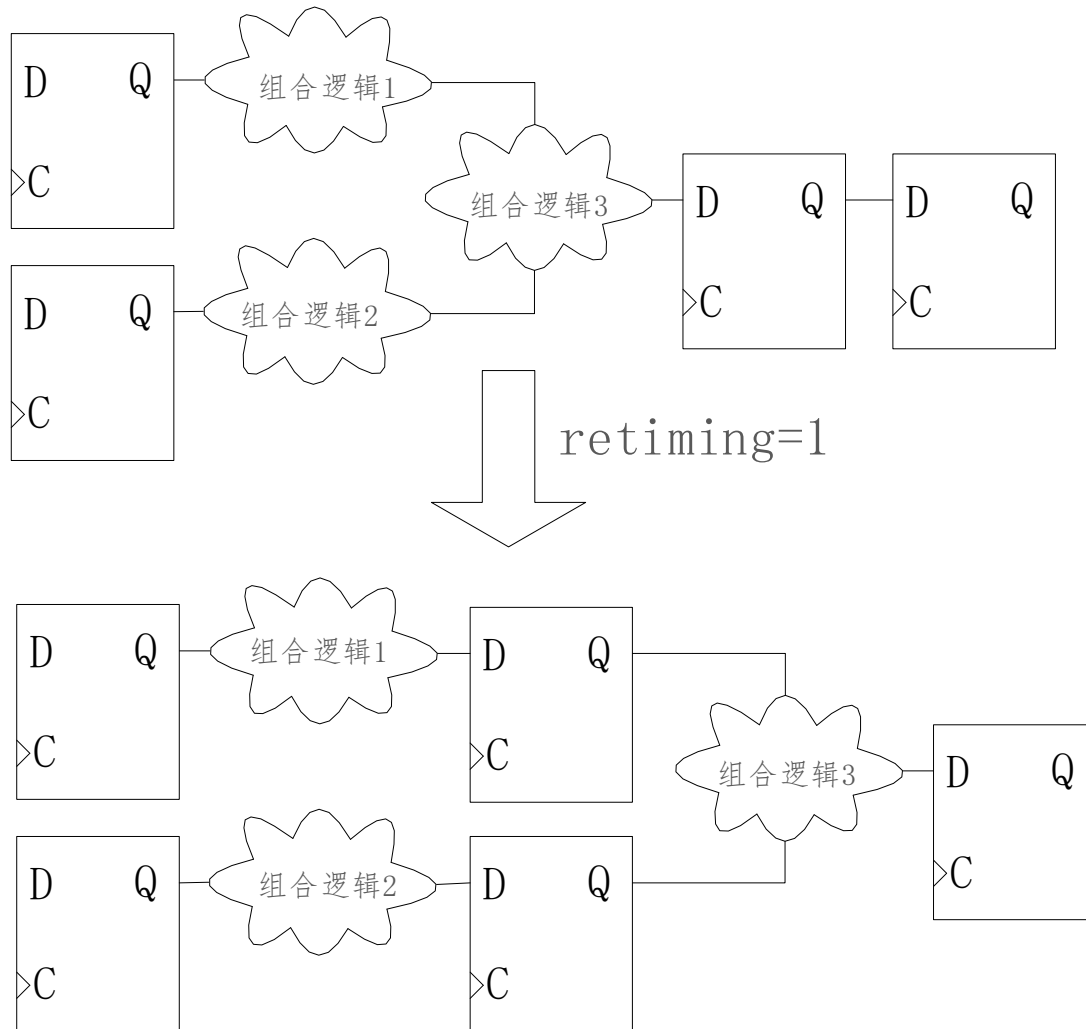


fanout-limit



- 高扇出信号负载很大，会带来海量的传播延迟
 - 通过查看时序报告里的关键路径详细信息里节点的pins on net数量可判断路径里是否存在高扇出信号
- 如果路径里有高扇出信号，通过适当设置fanout-limit限制扇出数量可以大大的降低延迟，对时序收敛有显著的帮助
 - Fanout-limit不是越小越好，需要根据具体设计要求调整，一般设置为100~150左右即可

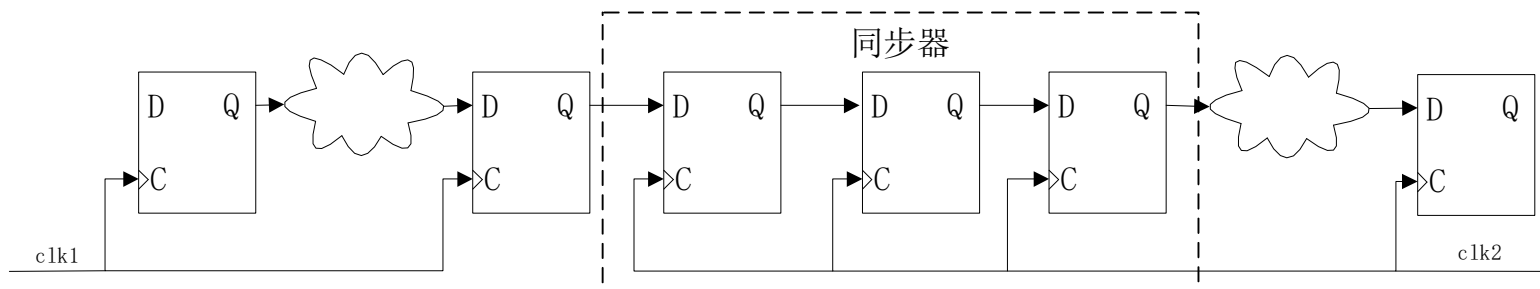
retiming



- 综合器自动判断相关联路径之间的逻辑关系;
- 在不影响功能的前提下移动组合逻辑之间的寄存器;
- 平衡各级寄存器之间的组合逻辑级数, 达到降低组合逻辑延迟的效果;
- 对 f_{MAX} 有一定帮助。

同步器综合属性约束

- 单根信号跨时钟域，使用多级触发器级联消除亚稳态
- 不希望被综合器自动优化
 - 移动(retiming), 复制(fanout-limit), 合并或者被优化掉
- 需要在RTL里使用async_reg综合属性约束



- Verilog HDL

```
(* async_reg = "true" *) reg [1:0] x;
```
- VHDL

```
attribute async_reg: boolean;  
attribute async_reg of x : signal is true;
```



时序优化策略

- 综合策略
- 布局布线策略
- 随机种子扫描
- 优化策略扫描



布局布线策略

- 优化策略(optimization_level)

- 针对非拥塞设计有三个等级的时序优化策略;
- 针对拥塞设计也有三个等级的时序优化策略;
- 帮助改善拥塞问题, 以便布线工具可以专注于改善时序;
- 优化等级越高, P&R算法的迭代次数越多, 所需要的时间也越多;

- 随机种子设置(seed)

- 种子, 相当于布局布线在FPGA中起始的位置, 不同的起始位置会对布局布线的结果产生影响;
- 同样的种子, 在不同的电脑, 不同操作系统, 不同的软件版本, P&R结果都会有所不同。

- 优化策略扫描(Optimization Sweep)

- 根据经验, 通常使用TIMING_3取得的效果相对会比较明显;
- 使用优化策略扫描, 软件自动使用所有的优化策略进行布局布线, 并记录包括时序报告在内的所有布局布线结果;
- 根据结果确定最适合当前设计的优化策略, 而不需要手动逐个尝试。

P&R优化策略	选项	功能描述
optimization_level	NULL	禁止优化 (默认)
	TIMING_1	非拥塞设计布局布线优化等级1, 低
	TIMING_2	非拥塞设计布局布线优化等级2, 中
	TIMING_3	非拥塞设计布局布线优化等级3, 高
	CONGESTION_1	拥塞设计布局布线优化等级1, 低
	CONGESTION_2	拥塞设计布局布线优化等级2, 中
	CONGESTION_3	拥塞设计布局布线优化等级3, 高
seed	任意整数	种子, 随机整数。相当于为布局布线算法设置一个随机的起始位置, 不同的起始位置会影响布局布线的结果。

- 随机种子扫描(Seed Sweep)

- 5个不同的种子产生的时序差异可达10%;
- 当设计 f_{MAX} 离设计要求还差10%以内, 或者是设计稍作修改 f_{MAX} 下降10%以内的时候, 可以选择随机种子扫描;
- 使用随机种子扫描, 软件自动使用定义范围的种子进行布局布线, 并记录包括时序报告在内的所有布局布线结果;
- 根据结果选择最适合当前设计的种子, 而不需要手动逐个尝试。

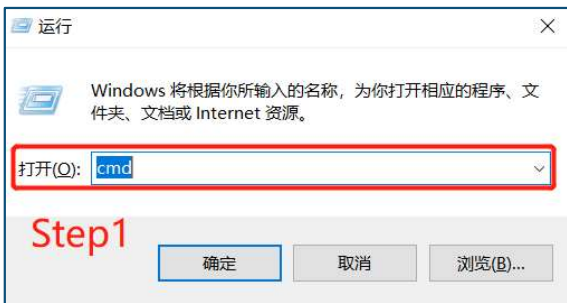


优化策略扫描(Optimization Sweep)

- **Step1** Win+R, 运行cmd, 进入命令行菜单
- **Step2** cd进入Efinity安装目录bin文件夹
- **Step3** 运行setup.bat
- **Step4** 在cmd命令行进入项目根目录
 - 如果项目和Efinity不在一个盘下, 需要先进行换盘符操作, 例如在D盘, 则在cmd命令提示符键入D:
 - cd进入项目根目录
- **Step5** 在项目根目录下运行策略扫描命令行
 - `efx_run_pnr_sweep.bat <项目名>.xml sweep_opt_levels`
- **Step6** 等待后台完成优化策略扫描
 - cmd窗口会提示进度
- **Step7** 查看扫描结果
 - 扫描报告在工程根目录下timing.sum.rpt
 - 详细结果文件包在工程根目录下run_sweep_<12位随机码>



优化策略扫描(Optimization Sweep)



C:\Users\Bruce Chen>cd C:\Efinity\2021.1\bin **Step2**

C:\Efinity\2021.1\bin>setup.bat **Step3**

C:\Efinity\2021.1\bin>d:

D:\>cd D:\Training\advanced_training\timg_closure\Test **Step4**

D:\Training\advanced_training\timg_closure\Test>efx_run_pnr_sweep.bat test.xml sweep_opt_levels **Step5**

pnr : **PASS**
Run flow: efx_run.bat C:/Users/BRUCEC~3/AppData/Local/Temp/efx-TIMING_3-ryxv_s0x/run_TIMING_3/test.xml --flow pgm --output_dir C:/Users/BRUCEC~3/AppData/Local/Temp/efx-TIMING_3-ryxv_s0x/run_TIMING_3/outflow --work_dir C:/Users/BRUCEC~3/AppData/Local/Temp/efx-TIMING_3-ryxv_s0x/run_TIMING_3/work_pnr
Running: efx_run_pgm.py test --family Trion --device T13F256 --output_dir C:/Users/BRUCEC~3/AppData/Local/Temp/efx-TIMING_3-ryxv_s0x/run_TIMING_3/outflow --work_dir C:/Users/BRUCEC~3/AppData/Local/Temp/efx-TIMING_3-ryxv_s0x/run_TIMING_3/work_pnr --opt mode=active width=1 enable_roms=smart spi_low_power_mode=on io_weak_pullup=on oscillator_clock_divider=DIV8 enable_crc_check=on
pgm : **PASS**
Finish optimization level sweeping. Please find the compiled results in D:/Training/advanced_training/timg_closure/Test/run_sweep_a9d467a39103
Generate timing summary report, timing.sum.rpt ... **Step6**

名称	修改日期	类型	大小
ip	2021/8/20 22:41	文件夹	
outflow	2021/9/8 13:07	文件夹	
run_sweep_a9d467a39103	2021/9/8 13:38	文件夹	
work_pnr	2021/9/7 21:52	文件夹	
work_pt	2021/9/8 13:07	文件夹	
work_syn	2021/9/7 21:52	文件夹	
.lock	2021/9/7 21:55	LOCK 文件	1 KB
fadfa.sdc	2021/9/5 16:13	SDC 文件	0 KB
test.log	2021/9/8 13:04	文本文件	9 KB
test.peri.xml	2021/8/21 0:47	XML 文档	6 KB
test.sdc	2021/9/5 16:23	SDC 文件	2 KB
test.v	2021/9/7 21:52	V 文件	1 KB
test.vdb	2021/9/7 21:52	VDB 文件	45 KB
test.xml	2021/9/8 13:07	XML 文档	4 KB
timing.sum.rpt	2021/9/8 13:38	RPT 文件	4 KB

名称	修改日期	类型
CONGESTION_1	2021/9/8 13:34	文件夹
CONGESTION_2	2021/9/8 13:35	文件夹
CONGESTION_3	2021/9/8 13:36	文件夹
TIMING_1	2021/9/8 13:37	文件夹
TIMING_2	2021/9/8 13:38	文件夹
TIMING_3	2021/9/8 13:38	文件夹
timing.sum.rpt	2021/9/8 13:38	RPT 文件

Maximum possible analyzed clocks frequency			
timing.sum.rpt			
Clock Name: pin_clk			
	Period (ns)	Frequency (MHz)	Edge
CONGESTION_1	3.368	296.918	(R-R)
CONGESTION_2	3.284	304.483	(R-R)
CONGESTION_3	3.021	331.013	(R-R)
TIMING_1	3.176	314.895	(R-R)
TIMING_2	3.564	280.572	(R-R)
TIMING_3	3.177	314.775	(R-R)
Clock Name: pll_clk			
	Period (ns)	Frequency (MHz)	Edge
CONGESTION_1	1.473	678.718	(R-R)
CONGESTION_2	1.598	625.746	(R-R)
CONGESTION_3	1.598	625.746	(R-R)
TIMING_1	1.556	642.547	(R-R)
TIMING_2	1.721	581.106	(R-R)
TIMING_3	1.525	655.539	(R-R)



易 灵 思

9/16/2021

48

随机种子扫描(Seed Sweep)

- **Step1** Win+R, 运行cmd, 进入命令行菜单
- **Step2** cd进入Efinity安装目录bin文件夹
- **Step3** 运行setup.bat
- **Step4** 在cmd命令行进入项目根目录
 - 如果项目和Efinity不在一个盘下, 需要先进行更换当前盘操作
 - 例如在D盘, 则在cmd命令提示符键入D:
 - cd进入项目根目录
- **Step5** 在项目根目录下运行策略扫描命令行
 - `efx_run_pnr_sweep.bat <项目名>.xml sweep_seeds [--num_seeds <种子个数>] [--start_seed <起始种子> --end_seed <结束种子>]`
- **Step6** 等待后台完成随机种子扫描
 - cmd窗口会提示进度
- **Step7** 查看扫描结果
 - 扫描报告在工程根目录下timing.sum.rpt
 - 详细结果文件包在工程根目录下run_sweep_<12位随机码>



随机种子扫描(Seed Sweep)

名称	修改日期	类型	大小
ip	2021/8/20 22:41	文件夹	
outflow	2021/9/8 13:07	文件夹	
run_sweep_8b383e68c073	2021/9/8 14:08	文件夹	
work_pnr	2021/9/7 21:52	文件夹	
work_pt	2021/9/8 13:07	文件夹	
work_syn	2021/9/7 21:52	文件夹	
.lock	2021/9/7 21:55	LOCK 文件	1 KB
fadfa.sdc	2021/9/5 16:13	SDC 文件	0 KB
test.log	2021/9/8 13:04	文本文档	9 KB
test.peri.xml	2021/8/21 0:47	XML 文档	6 KB
test.sdc	2021/9/5 16:23	SDC 文件	2 KB
test.v	2021/9/7 21:52	V 文件	1 KB
test.vdb	2021/9/7 21:52	VDB 文件	45 KB
test.xml	2021/9/8 13:07	XML 文档	4 KB
timing.sum.rpt	2021/9/8 14:08	RPT 文件	3 KB

名称	修改日期	类型
seed_1	2021/9/8 14:07	文件夹
seed_2	2021/9/8 14:08	文件夹
timing.sum.rpt	2021/9/8 14:08	RPT 文件

Maximum possible analyzed clocks frequency			
Clock Name: pin_clk			
	Period (ns)	Frequency (MHz)	Edge
seed_1	3.176	314.895	(R-R)
seed_2	3.074	325.349	(R-R)
Clock Name: pll_clk			
	Period (ns)	Frequency (MHz)	Edge
seed_1	1.556	642.547	(R-R)
seed_2	1.416	706.413	(R-R)

总结

- 理论基础
 - 时序参数的基本概念
 - 同步时序系统最大工作频率(f_{MAX})的计算原理
 - FPGA的四种基本时序路径分析
- 时序约束
 - 针对CORE逻辑的约束
 - 时钟约束
 - IO约束
 - 约束时序例外
 - 建立SDC文件
- 时序分析
 - 时序报告
 - 详细时序报告
 - 时序分析TCL命令
 - report_timing
- 时序优化策略
 - 综合策略
 - 布局布线策略
 - 随机种子扫描
 - 优化策略扫描





谢谢！