易 灵 思

易灵思FPGA设计技巧

——时序收敛

陈弘 Bruce Chen

2021年9月

版本

日期	版本	版本描述
2021/9/10	V1.0	初稿发布
2021/9/116	V1.1	修正约束Input Receive Clock Delay公式错误

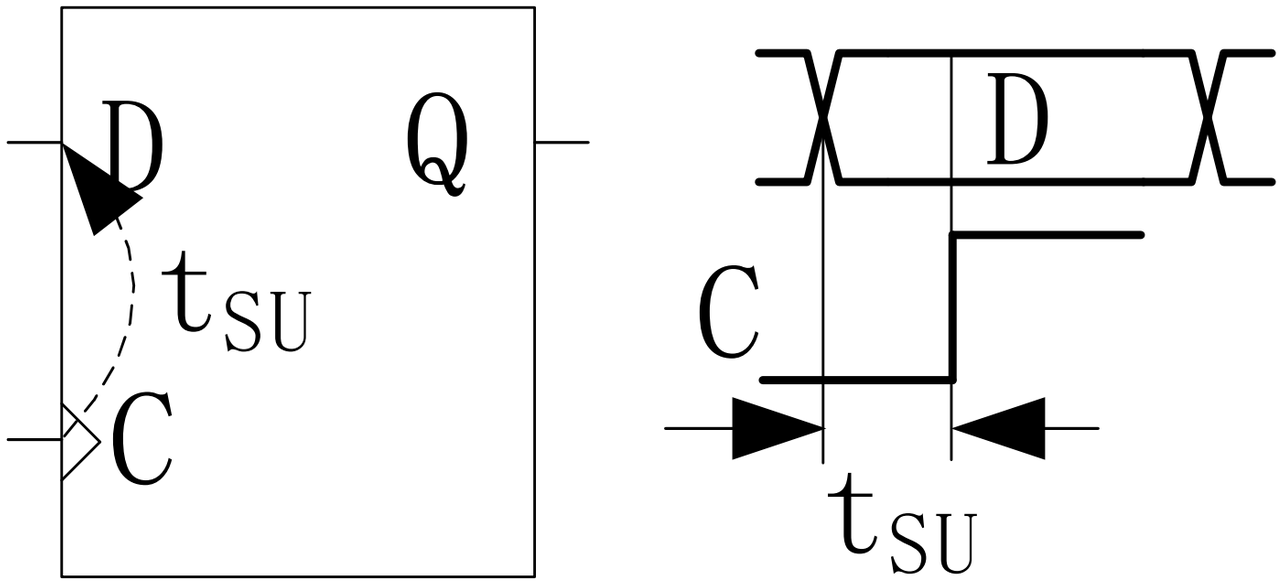
主要内容

1. 理论基础
2. 时序分析
3. 时序约束
4. 时序优化策略

理论基础

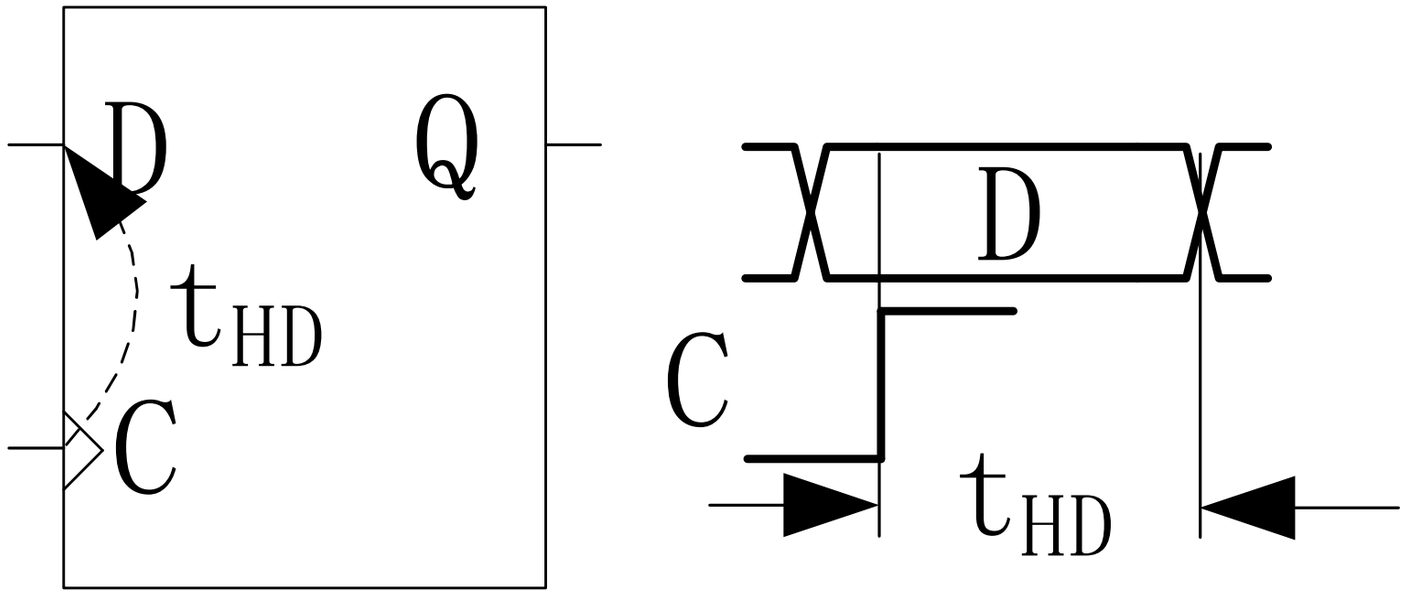
- 时序参数的基本概念
  - 建立时间(t_{SU})
  - 保持时间(t_{HD})
  - 时钟到输出时间(t_{CO})
  - 传播延迟(t_{PD})
  - 时钟的Jitter， Skew和Uncertainty
- 同步时序系统最大工作频率(f_{MAX})的计算原理
- FPGA的四种基本时序路径分析
  - 从FPGA输入引脚到目的寄存器数据输入端口(P2R)
  - 从源寄存器时钟输入端口到目的寄存器数据输入端口(R2R)
  - 从源寄存器时钟输入端口到FPGA输出端口(R2P)
  - 从FPGA输入端口到FPGA输出端口(P2P)

建立时间(t_{SU})



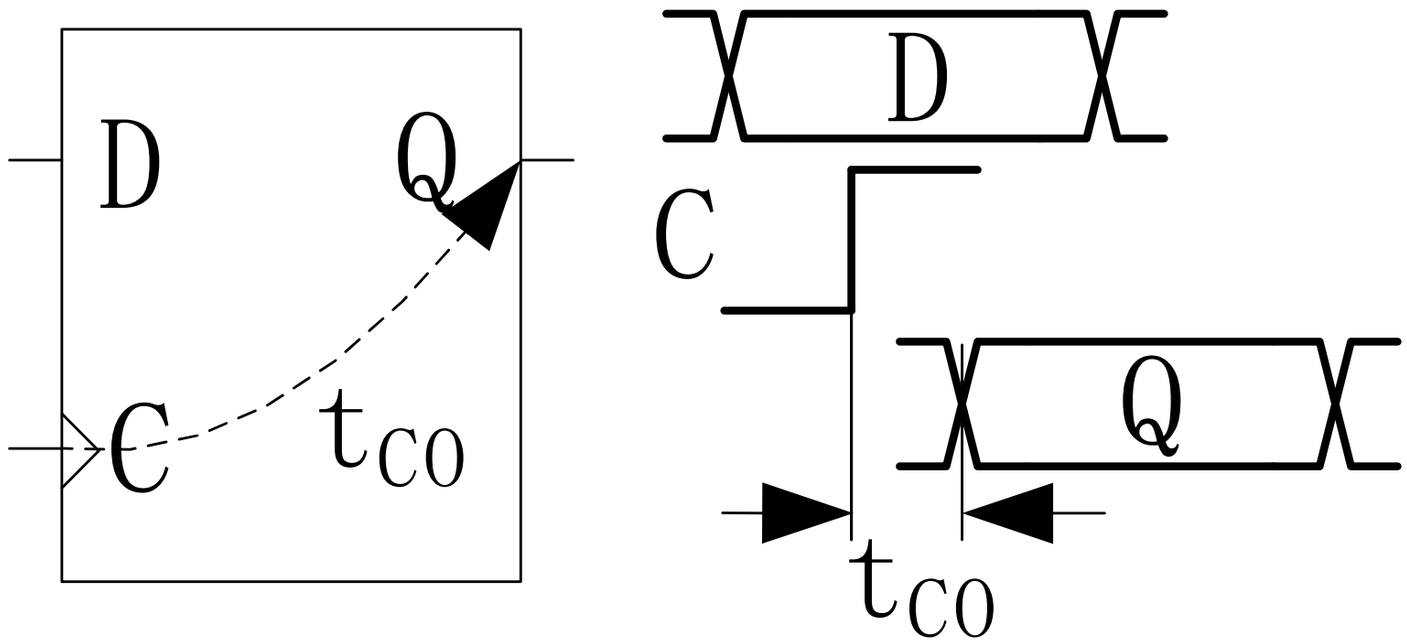
- **t_{SU} 建立时间**
  - 时序单元(FF)的输入数据在时钟有效边沿之前，必须提前稳定的时间。
  - 对于FPGA的时序单元(FF)来说，一旦选定了目标器件，时序模型也就固定了，各时序单元(FF)的t_{SU}建立时间的最低要求也就确定了。

保持时间(t_{HD})



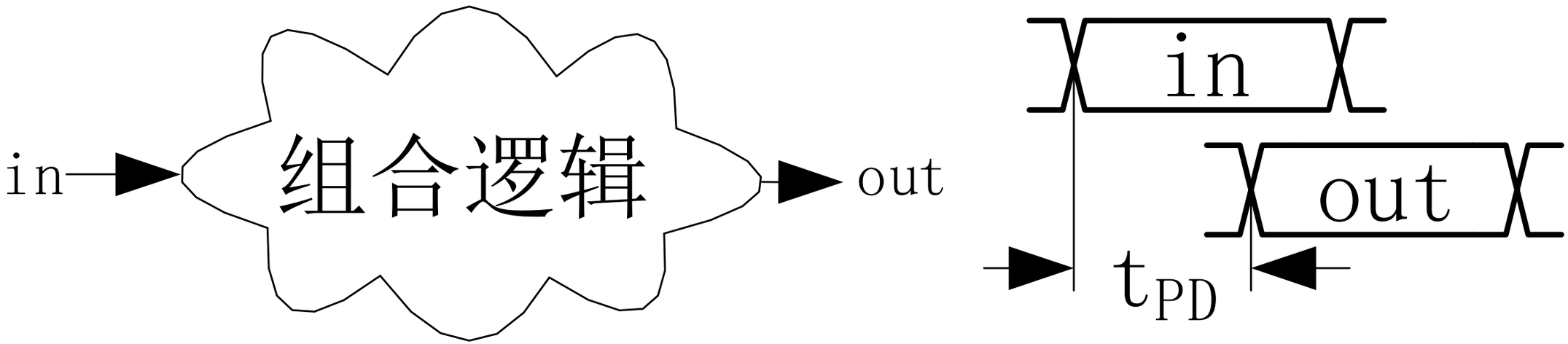
- **t_{HD}保持时间**
  - 时序单元(FF)的输入数据在时钟有效边沿到来之后，需要继续保持稳定的时间。
  - 对于FPGA的时序单元(FF)来说，一旦选定了目标器件，时序模型也就固定了，各时序单元(FF)的t_{HD}保持时间的最低要求也就确定了。

时钟到输出时间(t_{CO})



- **t_{CO}时钟到输出时间**
  - 时序单元(FF)的时钟有效边沿与输出端（Q端）数据稳定的时间差。
  - 对于FPGA的时序单元(FF)来说，一旦选定了目标器件，时序模型也就固定了，各时序单元(FF)的t_{CO}时钟到输出时间也就确定了。

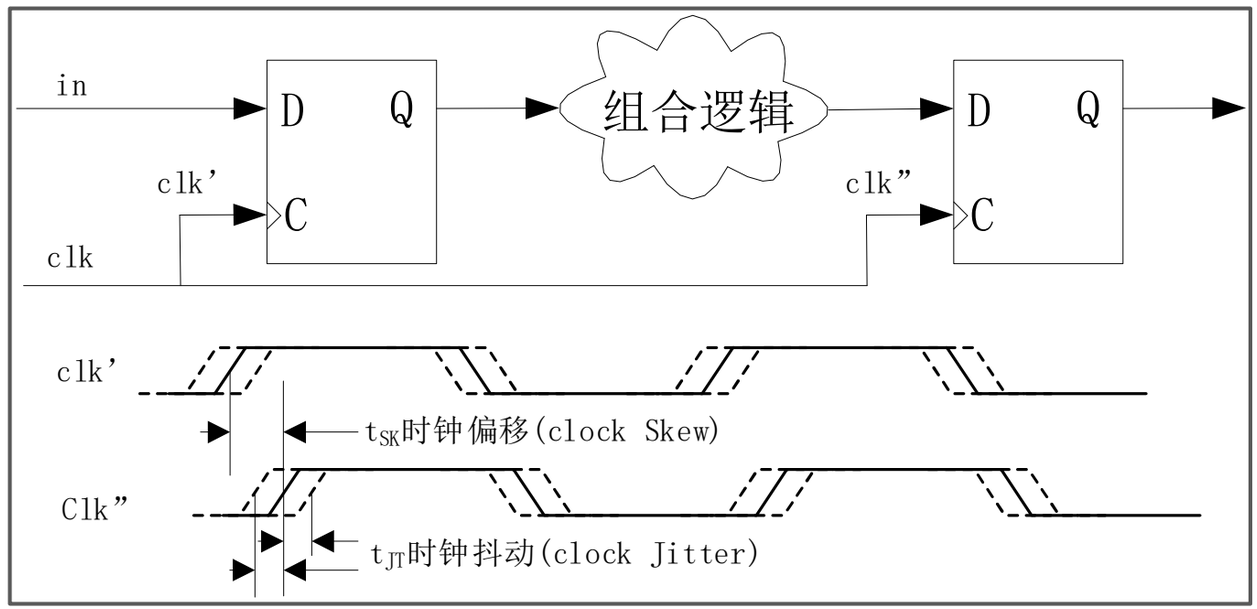
传播延迟(t_{PD})



- **t_{PD}传播延迟**
  - 组合逻辑输入变化到输出变化的延迟。
  - 对FPGA来说，确定时序模型并且布局布线完成以后，组合逻辑的t_{PD}传播延迟就会确定下来。

时钟的Jitter， Skew和Uncertainty

- **时钟抖动(Jitter) t_{JT}**
  - 时钟抖动是指同一时钟，相邻周期间时间不一致的现象。这一误差来源于时钟自身，如：晶振、PLL电路的偏差，与噪声、干扰以及电源变化有关。抖动还可能表现为占空比的改变，称为半周期抖动。综上：可以认为时钟抖动是时钟信号本身在传输过程中的一些偶然和不定的变化之总和。
- **时钟偏移(Skew) t_{SK}**
  - 时钟偏移是因为布线长度和负载不同，导致同一时钟边沿到不同触发器的时间不同。这一时间差，即为时钟偏移。
- **Jitter和Skew的区别**
  - Jitter是时钟源（PLL，OSC）产生的，只和晶振或者PLL以及内部电路有关，布线对其没有影响。Skew是由时钟的布线长度不一致引起时钟边沿到达不同的时序单元延时不同导致的。

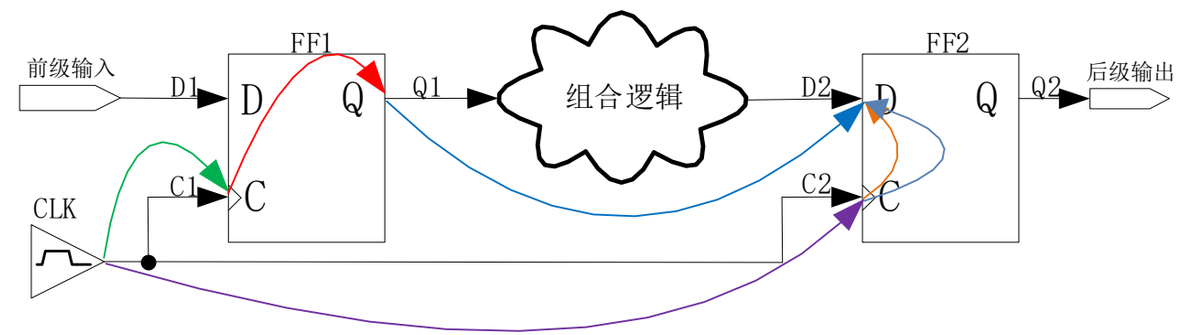


- **时钟不确定性(Uncertainty)**
  - **对于FPGA内部来说**，布局布线完成以后，时钟到达不同的时序单元(FF)的延迟就确定了，在对FPGA内部时序分析的时候会作为时钟路径延迟来计算，所以Uncertainty实际上并没有考虑时钟偏移Skew。
  - **对外设器件进入FPGA的信号来说**，有可能各个单根信号到FPGA的路径不一致，但是对于FPGA的时序分析工具来说，它并不知道这些不一致的信息。所以FPGA的时序分析，既要考虑时钟的Jitter，也要考虑外设器件本身的输出和PCB布线的Skew。

理论基础

- 时序参数的基本概念
  - 建立时间(t_{SU})
  - 保持时间(t_{HD})
  - 时钟到输出时间(t_{CO})
  - 传播延迟(t_{PD})
  - 时钟的Jitter， Skew和Uncertainty
- 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
  - 从FPGA输入引脚到目的寄存器数据输入端口(P2R)
  - 从源寄存器时钟输入端口到目的寄存器数据输入端口(R2R)
  - 从源寄存器时钟输入端口到FPGA输出端口(R2P)
  - 从FPGA输入端口到FPGA输出端口(P2P)

同步系统时序分析原理



Launch Clock Path Delay, t_{LCPD}

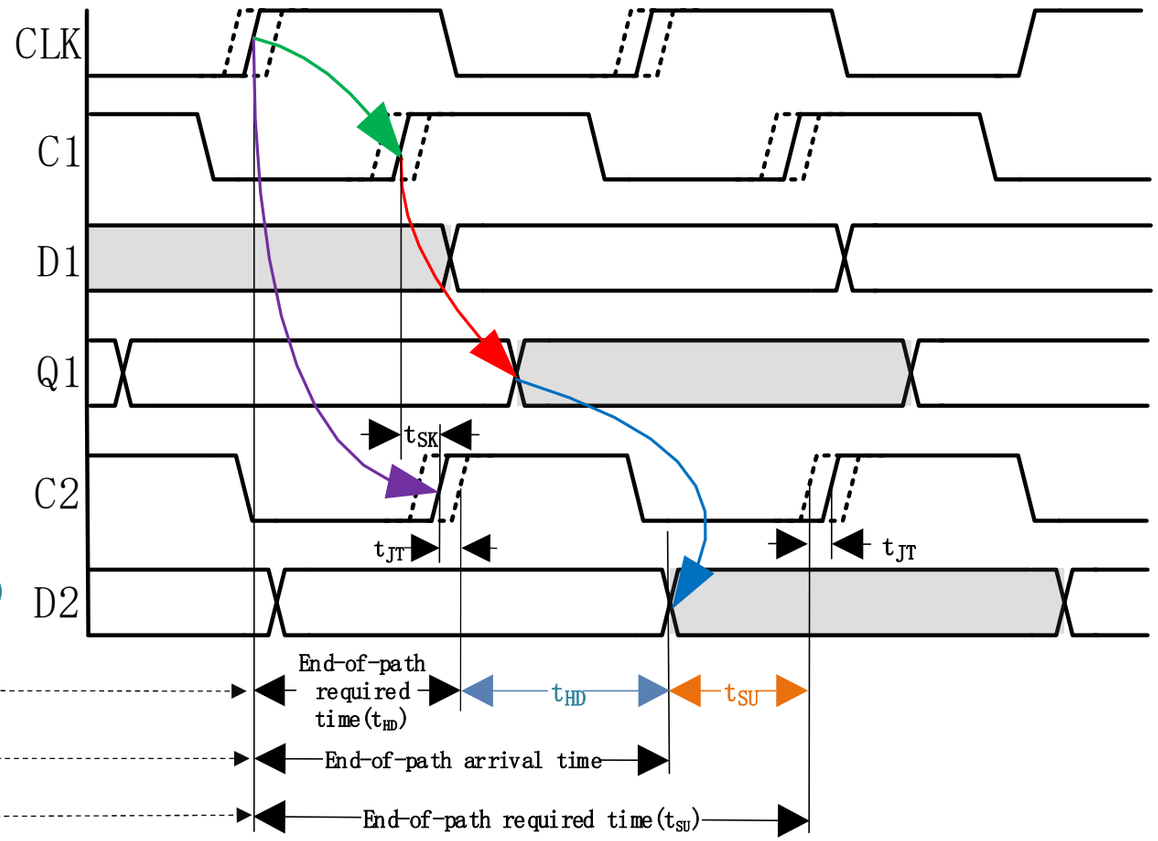
Clock To Q, t_{CO}

Data Path Delay, t_{PD}

Capture Clock Path Delay, t_{CCPD}

t_{SU}=End-of-path required time(t_{SU})-End-of-path arrival time

t_{HD}=End-of-path arrival time - End-of-path required time(t_{HD})

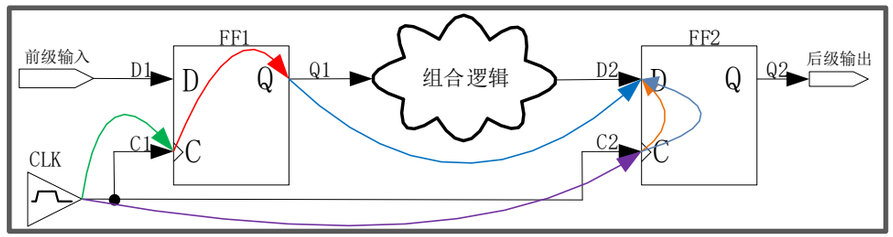


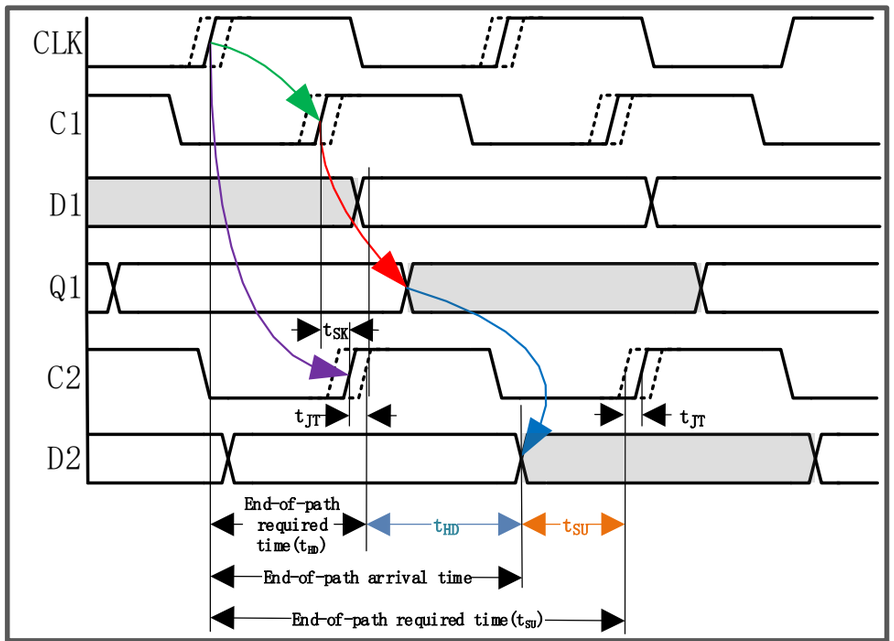
End-of-path required time(t_{HD})=t_{CCPD} + t_{JT}

End-of-path arrival time=t_{LCPD} + t_{CO} + t_{PD}

End-of-path required time(t_{SU})=T + t_{CCPD} − t_{JT}

同步系统时序分析原理





- Launch Clock Path Delay
  - 发起时钟路径延迟，时钟源CLK到FF1的路径延迟，t_{LCPD}
- Capture Clock Path Delay
  - 捕获时钟路径延迟，时钟源CLK到FF2的路径延迟，$t_{CCPD}= t_{LCPD} +t_{SK}$
- Clock To Q
  - 时钟到输出时间，即t_{CO}
- Data Path Delay
  - 数据路径延迟，即传播延迟t_{PD}
- End of path arrival time
  - 数据实际到达时间，$t_{LCPD}+ t_{CO}+ t_{PD}$
- End of path required time
  - 默认的单周期系统中FF2要求在FF1数据发出后的下一个时钟边沿正确采样
  - 数据建立要求到达时间 t_{SU}: $T + t_{CCPD} – Clock\ Uncertainty$，T为时钟周期
  - 数据保持要求到达时间 t_{HD}: $t_{CCPD} + Clock\ Uncertainty$

时序收敛条件推导

- t_{SU}建立时间敛条件推导
  - t_{SU} = End of path required time(t_{SU}) – End of path arrival time => $t_{SU} = (T + t_{CCPD} - t_{JT}) - (t_{LCPD} + t_{CO} + t_{PD})$ =>
  - $t_{SU} = T + (t_{CCPD} - t_{LCPD}) - (t_{CO} + t_{PD} + t_{JT})$ => $t_{SU} = T + t_{SK} - (t_{CO} + t_{PD} + t_{JT})$ =>
  - $T = t_{SU} + t_{CO} + t_{PD} + t_{JT} - t_{SK}$ => $f = 1/T = 1/(t_{SU} + t_{CO} + t_{PD} + t_{JT} - t_{SK})$ =>
  - 设建立时间要求是t_{SUmin}，则有$t_{SU} \geq t_{SUmin}$ => $f \leq 1/(t_{SUmin} + t_{CO} + t_{PD} + t_{JT} - t_{SK})$ =>
  - $f_{MAX} = 1/(t_{SUmin} + t_{CO} + t_{PD} + t_{JT} - t_{SK})$, $t_{PD} \leq T - (t_{SUmin} + t_{CO} + t_{JT}) + t_{SK}$
- t_{HD}保持时间敛条件推导
  - t_{HD} = End of path arrival time - End of path required time(t_{HD}) => $t_{HD} = t_{LCPD} + t_{CO} + t_{PD} - (t_{CCPD} + t_{JT})$ =>
  - $t_{HD} = t_{CO} + t_{PD} - t_{JT} - (t_{CCPD} - t_{LCPD})$ => $t_{HD} = t_{CO} + t_{PD} - t_{JT} - t_{SK}$ =>
  - 设最小保持时间要求是t_{HDmin}，则有$t_{HD} \geq t_{HDmin}$ => $t_{CO} + t_{PD} - t_{JT} - t_{SK} \geq t_{HDmin}$ =>
  - $t_{PD} \geq t_{HDmin} + t_{JT} + t_{SK} - t_{CO}$
- 易灵思FPGA的FF对建立保持时间的要求很小，忽略不计
  - $t_{SUmin} = 0$, $t_{HDmin} = 0$
  - $f_{MAX} = 1/(t_{CO} + t_{PD} + t_{JT} - t_{SK})$, $t_{PD} \leq T - (t_{CO} + t_{JT}) + t_{SK}$
  - $t_{PD} \geq t_{JT} + t_{SK} - t_{CO}$

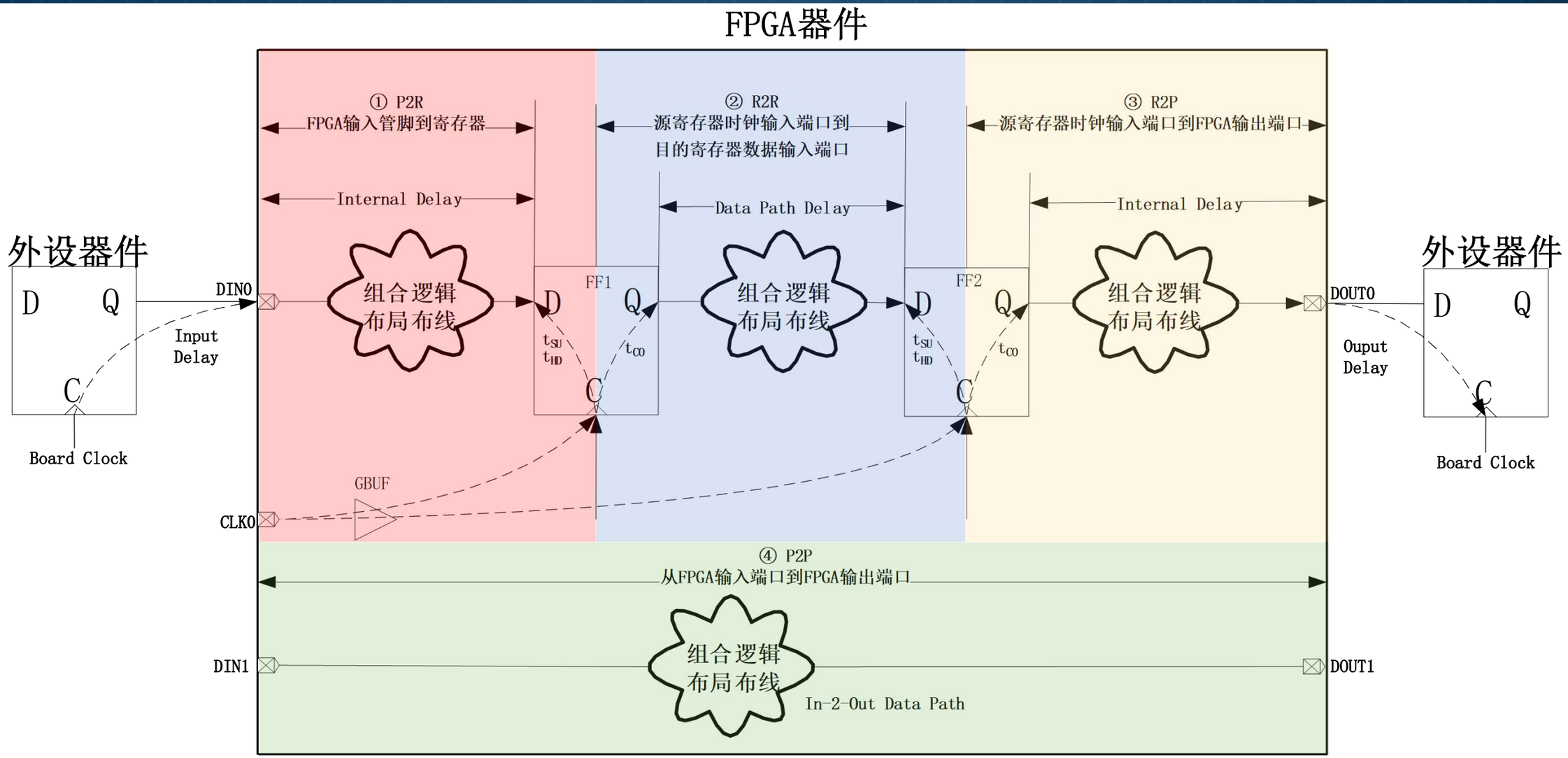
四个结论

- FPGA最大工作频率的计算公式$f_{MAX} = 1/(t_{SUmin} + t_{CO} + t_{PD} - t_{SK} + t_{JT})$。
- 对于FPGA来说，保持时间t_{HD}收敛条件，很容易满足。
- 数据传播延迟t_{PD}越小建立时间t_{SU}越容易满足，取决于最大的路径延迟。
- 而对于保持时间t_{HD}来说，数据传播延迟t_{PD}越大越容易满足，取决于最小的路径延迟。

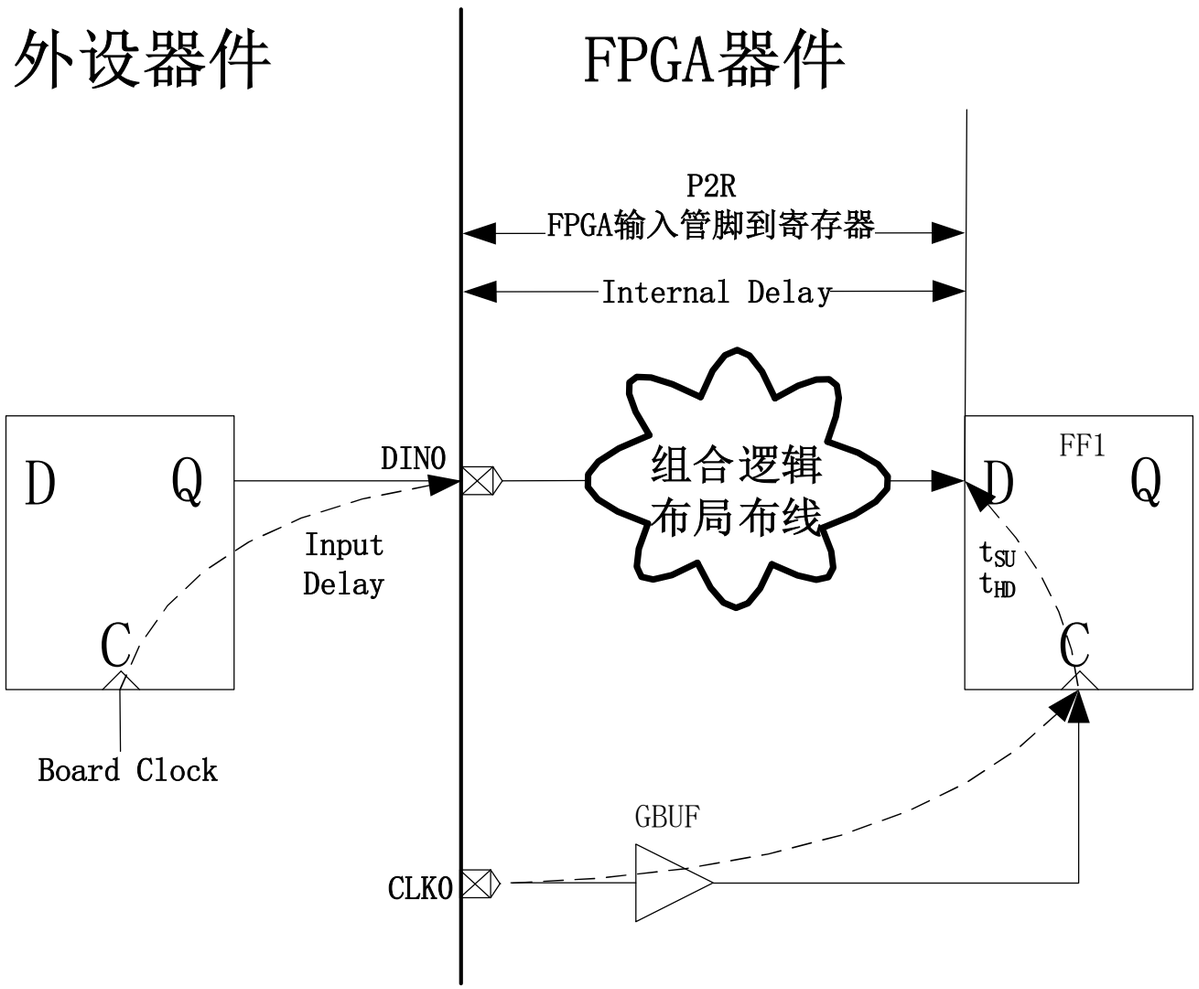
理论基础

- 时序参数的基本概念
  - 建立时间(t_{SU})
  - 保持时间(t_{HD})
  - 时钟到输出时间(t_{CO})
  - 传播延迟(t_{PD})
  - 时钟的Jitter， Skew和Uncertainty
- 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
  - 从FPGA输入引脚到目的寄存器数据输入端口(P2R)
  - 从源寄存器时钟输入端口到目的寄存器数据输入端口(R2R)
  - 从源寄存器时钟输入端口到FPGA输出端口(R2P)
  - 从FPGA输入端口到FPGA输出端口(P2P)

FPGA的四种基本时序路径分析

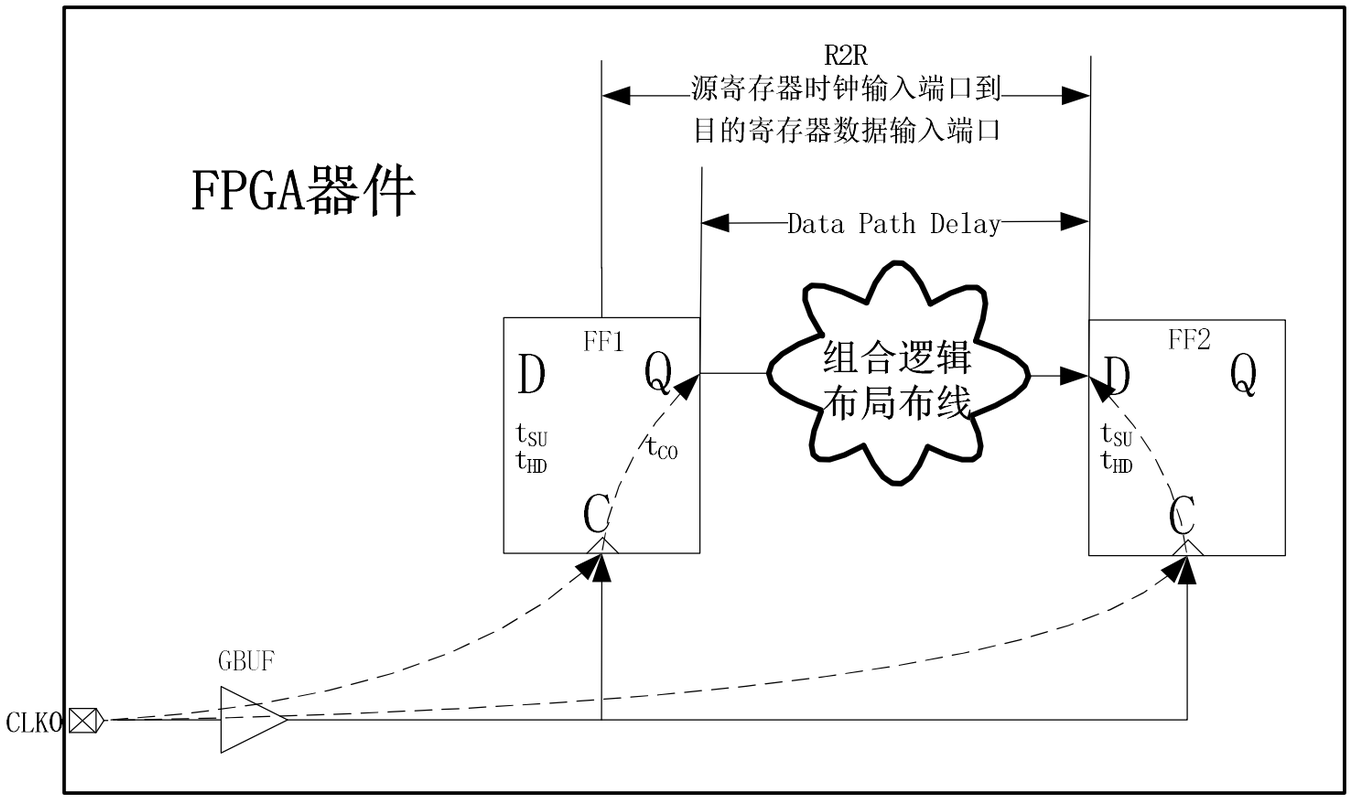


从FPGA输入引脚到目的寄存器数据输入端口(P2R)



- 数据由外设器件的Board Clock发起，经过Input Delay的延迟后到达FPGA的输入引脚。
- 然后经过FPGA的Internal Delay到达目的时钟驱动的目的寄存器。
- FPGA不知道源路径信息。
- 用户需要约束Input Delay和时钟告知时序引擎必要信息，时序引擎才能正确的分析这种路径。

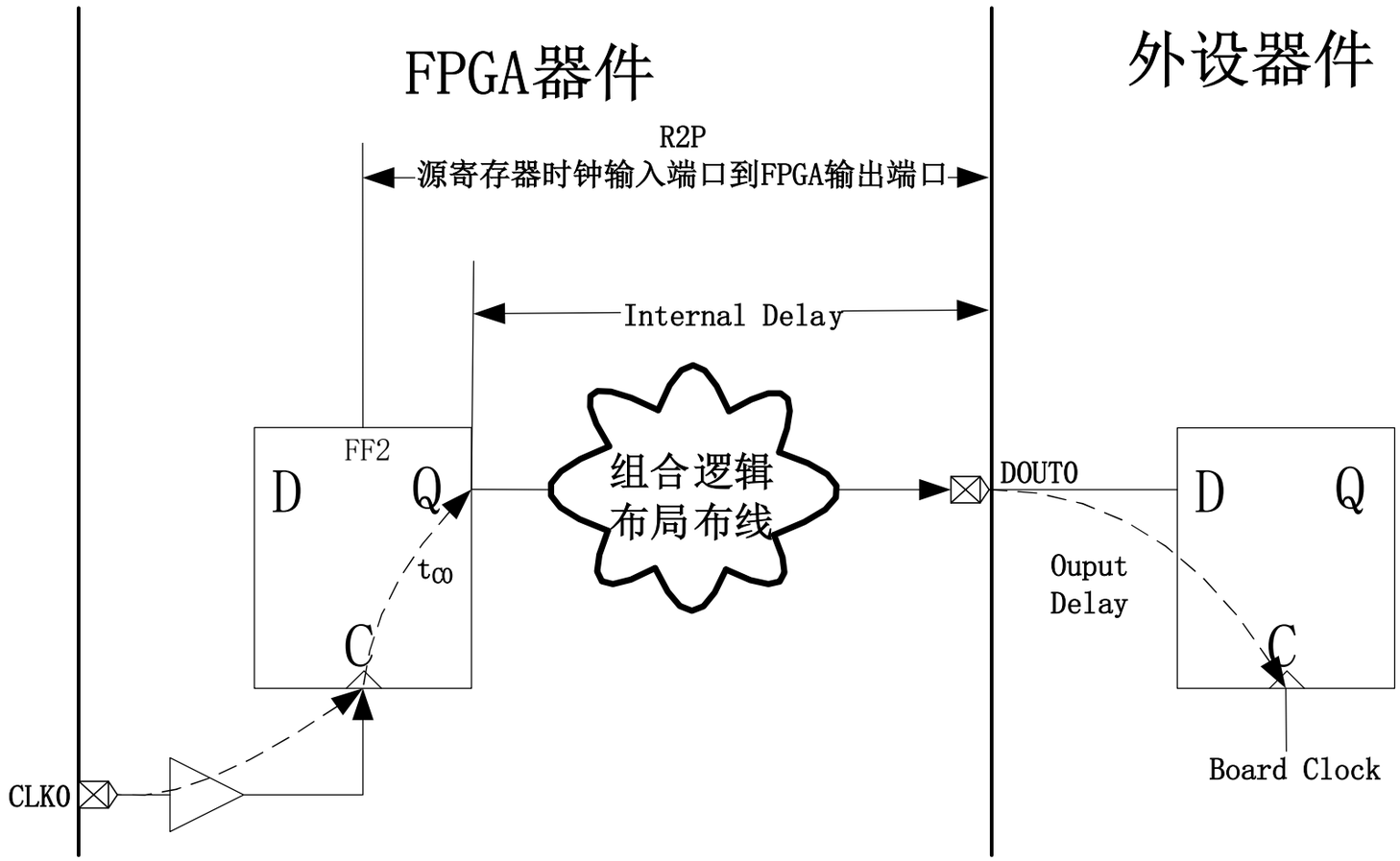
从源寄存器时钟输入端口到目的寄存器数据输入端口(R2R)



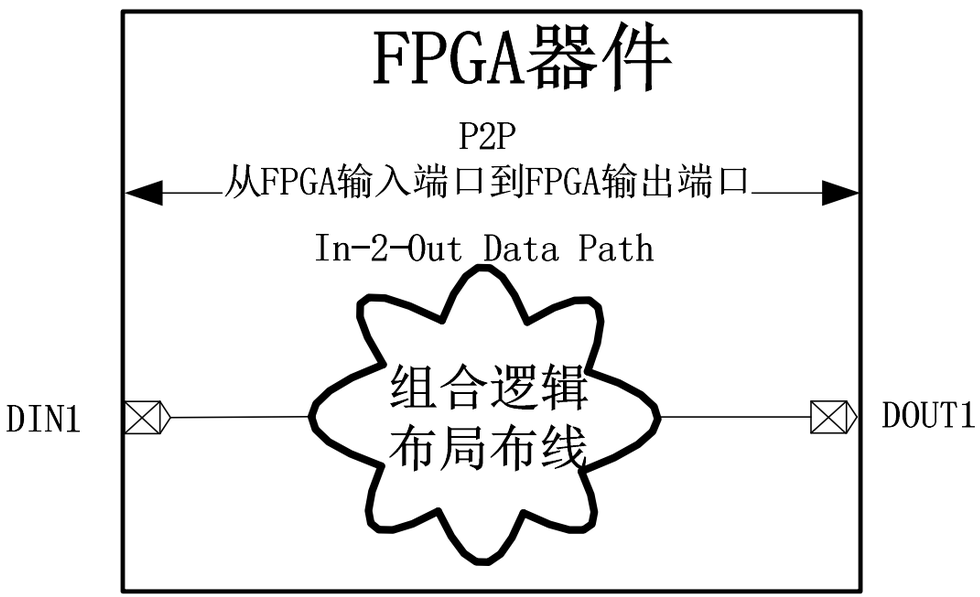
- 数据由源时钟在FPGA内部的源寄存器发起。
- 数据经过Data Path Delay后到达目的时钟驱动的目的寄存器。
- 这种时序路径是FPGA设计中最常见的。
- 用户需要约束源时钟和目的时钟告知时序引擎必要的信息，时序引擎才能正确地分析这种时序路径。

从源寄存器时钟输入端口到FPGA输出引脚(R2P)

- 数据由源时钟在FPGA内部的源寄存器发起。
- 数据经过Internal Delay后到达FPGA的输出引脚。
- 然后数据经过Output Delay后被目的外设器件的时钟Board Clock捕获到。
- FPGA不知道目的路径信息。
- 用户需要约束Output Delay和时钟来告知时序引擎必要信息，时序引擎才能正确地分析这种路径。



从FPGA输入引脚到FPGA输出引脚(P2P)



- 数据直接穿过FPGA，没有经过任何触发器。这种路径也叫In-To-Out Path。
- 路径中只有数据路径。
- 用户需要约束Input Delay和Output Delay，告知时序引擎必要的信息，时序引擎才能正确地分析这种时序路径。
- 使用虚拟时钟作为参考时钟来约束Input Delay和Output Delay 。

小节

- FPGA的最大工作频率计算公式: $f_{MAX} = 1/(t_{SUmin} + t_{CO} + t_{PD} - t_{SK} + t_{JT})$
  - t_{SUmin}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

主要内容

1. 理论基础
2. 时序分析
3. 时序约束
4. 时序优化策略

时序分析

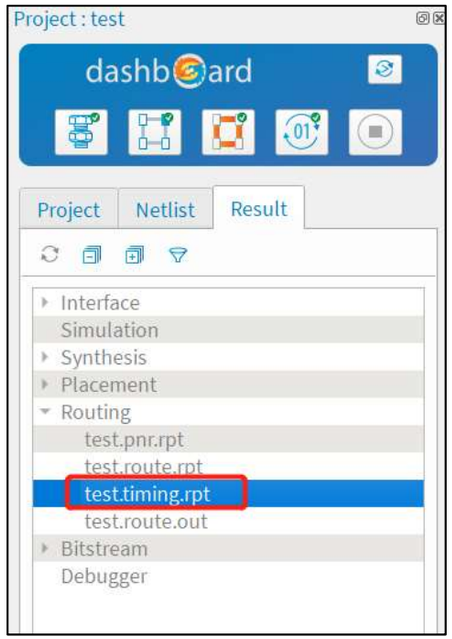
- 时序报告
  - 时序报告简述
  - 详细时序报告(<项目名>.timing.rpt)
    - 时序模型信息
    - 时钟频率简报
    - 时钟关系简报
    - 最长关键路径详细报告
    - 最短关键路径详细报告
- 时序分析TCL命令
  - report_timing
  - 其它命令

时序报告简述

- 时序简报： Dashboard => Result => Timing面板
  - Worst Negative Slack 最差时钟下降沿余量
  - Worst Hold Slack　　最差保持时间余量
  - 时钟最大工作频率f_{MAX}

Timing	
Worst Negative Slack (WNS)	1.421 ns
Worst Hold Slack (WHS)	0.377 ns
pin_clk	326.012 MHz
pll_clk	791.931 MHz

- 详细时序报告： Dashboard => Result => Routing => <项目名>.timing.rpt
  - 详细时序报告
    - 时序模型信息
    - 时钟频率简报
      - 用户的时钟约束列表
      - 时钟最大可能工作频率f_{MAX}
    - 时钟关系简报
    - 最长关键路径详细报告(t_{SU}相关)
    - 最短关键路径详细报告(t_{HD}相关)




```
SDC Filename: D:/Training/advanced_training/timg_closure/Test/test.sdc

Timing Model: C4
     temperature : 0C to 85C
     voltage : 1.2V +/-50mV
     speedgrade : 4
     technology : s40ll
     status : final

---------- Table of Contents (begin) ----------
   1. Clock Frequency Summary
   2. Clock Relationship Summary
   3. Path Details for Max Critical Paths
   4. Path Details for Min Critical Paths
---------- Table of Contents (end) ------------
```

时序分析

- 时序报告
  - 时序报告简述
  - 详细时序报告(<项目名>.timing.rpt)
    - 时序模型信息
    - 时钟频率简报
    - 时钟关系简报
    - 最长关键路径详细报告
    - 最短关键路径详细报告
- 时序分析TCL命令
  - report_timing
  - 其它命令

时序模型信息

- 当前时序报告的时序模型
  - 温度范围
  - 内核电压
  - 速度等级
  - 生产工艺
  - 模型状态

```
12  Timing Model: C4
13        temperature : 0C to 85C
14        voltage : 1.2V +/-50mV
15        speedgrade : 4
16        technology : s4011
17        status : final
```

- 使用get_timing_model命令可在控制台查看当前时序模型
- 使get_available_timing_model命令可在控制台查看当前器件型号所在系列支持的所有时序模型
- 使用set_timing_model命令可重新指定当前工程的时序模型
  - 同其他report命令配合使用，不用重新选择器件型号，重复编译，即可得到其他速度等级器件在当前编译下的时序分析报告

时钟工作频率简报

- 时序约束SDC文件定义的时钟信息
  - 可用来确认时序约束文件中时钟定义是否有效和正确
- 当前工程各时钟实际最大工作频率
  - 对应的最小时钟周期
  - 最大工作频率f_{MAX}
  - R-R代表上升沿发起，上升沿捕获

```
---------- Table of Contents (begin) ----------
  1. Clock Frequency Summary
  2. Clock Relationship Summary
  3. Path Details for Max Critical Paths
  4. Path Details for Min Critical Paths
---------- Table of Contents (end) ------------

---------- 1. Clock Frequency Summary (begin) ----------

User target constrained clocks
Clock Name      Period (ns)    Frequency (MHz)    Waveform    Source Clock Name
pin_clk             5.000          200.000          {0.000 2.500}        pin_clk
pll_clk             5.000          200.000          {0.000 2.500}        pll_clk

Maximum possible analyzed clocks frequency
Clock Name      Period (ns)    Frequency (MHz)    Edge
pin_clk             3.067          326.012      (R-R)
pll_clk             1.263          791.931      (R-R)

Geomean max period: 1.968

---------- Clock Frequency Summary (end) ---------------
```

时钟关系简报

```
19  ---------- Table of Contents (begin) ----------
20     1. Clock Frequency Summary
21     2. Clock Relationship Summary
22     3. Path Details for Max Critical Paths
23     4. Path Details for Min Critical Paths
24  ---------- Table of Contents (end) ------------
42  ---------- 2. Clock Relationship Summary (begin) ----------
43
44  Launch Clock    Capture Clock    Constraint (ns)    Slack (ns)    Edge
45  pin_clk          pin_clk                5.000         1.933     (R-R)
46  pin_clk          pll_clk                2.500         1.421     (R-F)
47  pll_clk          pll_clk                5.000         3.737     (R-R)
48
49  NOTE: Values are in nanoseconds.
50
51  ---------- Clock Relationship Summary (end) ---------------
```

- 关键路径(Critical Path)
  - 一对相关联的发起时钟(Launch Clock)和捕获时钟之间(Capture Clock)所有的路径中最差的一条路径。
  - 决定了捕获时钟是否能正确采样数据，是决定了f_{MAX}的一条路径

- 设计中所有相关联的发起(Launch Clock)时钟和捕获(Capture Clock)时钟所有组合的关键路径时序简报
- 可用来确定设计是否有跨时钟设计
  - 检查时钟关系约束是否合理
- 可用来确定半周期捕获是否合理
  - R-R 上升沿发起，上升沿捕获
  - F-R 下降沿发起，上升沿捕获
  - R-F 上升沿发起，下降沿捕获
- Slack大于0代表时序收敛
- Slack小于0代表时序未收敛

最长关键路径详细报告

```
---------- Table of Contents (begin) ----------
   1. Clock Frequency Summary
   2. Clock Relationship Summary
   3. Path Details for Max Critical Paths
   4. Path Details for Min Critical Paths
---------- Table of Contents (end) ------------
```

```
---------- 2. Clock Relationship Summary (begin) ----------

Launch Clock    Capture Clock    Constraint (ns)    Slack (ns)    Edge
pin_clk          pin_clk               5.000          1.933      (R-R)  ①
pin_clk          pll_clk               2.500          1.421      (R-F)  ②
pll_clk          pll_clk               5.000          3.737      (R-R)  ③

NOTE: Values are in nanoseconds.

---------- Clock Relationship Summary (end) ---------------
```

```
---------- 3. Path Details for Max Critical Paths (begin) ----------


##########################################################################
Path Detail Report (pin_clk vs pll_clk)                                  ②
##########################################################################
```

```
##########################################################################
Path Detail Report (pin_clk vs pin_clk)                                  ①
##########################################################################
```

```
##########################################################################
Path Detail Report (pll_clk vs pll_clk)                                  ③
##########################################################################
---------- Path Details for Max Critical Paths (end) ---------------
```

- 最长关键路径(Max Critical Path)
  - 一对相关联的发起时钟(Launch Clock)和捕获时钟之间(Capture Clock)所有的路径中延时最长的一条路径。
- 最长关键路径详细报告
  - 时序报告第3部分
  - 对应于时序报告第2部分，时钟关系简报列出的各相关时钟的最长路径
  - 决定了设计是否能满足建立时间t_{SU}，是决定f_{MAX}的一条路径

最长关键路径详细报告-基本路径信息

- Path Begin : 路径起点
- Path End : 路径终点
- Launch Clock : 发起时钟
- Capture Clock : 捕获时钟
- Slack : 时序余量 （ Require time – arrival time ）
  要求到达时间 - 实际到达时间
- Delay : 组合逻辑传播延迟(FF Q到FF D)， t_{PD}
- Logic Level : 组合逻辑级数
- Non-global nets on path : 组合逻辑路径非全局信号数
- Global nets on path : 组合逻辑路径全局信号数
- End-of-Path arrival time : 实际到达时间
  - Launch Clock Path Delay : 发起时钟路径延迟，t_{LCPD}
  - Clock To Q : 路径起点FF的时钟到输出时间，t_{CO}
  - Data Path Delay : 组合逻辑传播延迟， t_{PD}
  - Launch Clock Path Delay + Clock To Q + Data Path Delay
  - 实际到达时间 : $t_{LCPD} + t_{CO} + t_{PD}$

```
Path Begin    : in1[1]~FF|CLK
Path End      : out~FF|D
Launch Clock  : pll_clk (RISE)
Capture Clock : pll_clk (RISE)
Slack         : 3.737  (required time - arrival time)
Delay         : 0.743

Logic Level : 1
Non-global nets on path : 1
Global nets on path     : 0

Launch Clock Path Delay        : 4.110
+ Clock To Q + Data Path Delay : 1.143
-----------------------------------------
End-of-path arrival time       : 5.253

Constraint                     : 5.000
+ Capture Clock Path Delay     : 4.110
- Clock Uncertainty            : 0.120
-----------------------------------------
End-of-path required time      : 8.990
```

- End-of-path require time : 要求到达时间
  - Constraint : 对应发起时钟边沿以后捕获时钟的下一个边沿，T
  - Capture Clock Path Delay : 捕获时钟路径延迟， t_{CCPD}
  - Clock Uncertainty : 时钟不确定性，在FPGA里视为Jitter， t_{JT}
  - Constraint + Capture Clock Path Delay – Clock Uncertainty
  - 要求到达时间 : T + t_{CCPD} - Uncertainty

建立时间时序余量(Slack) = 要求到达时间 - 实际到达时间 $=(T + t_{CCPD} - \text{Uncertainty}) - (t_{LCPD} + t_{CO} + t_{PD}) = T - (t_{CO} + t_{PD} + t_{JT}) + t_{SK}$ $\Rightarrow t_{PD} \le T - (t_{CO} + t_{JT}) + t_{SK}$

$f_{MAX} = 1/(t_{CO} + t_{PD} + t_{JT} - t_{SK})$

最长关键路径详细报告-详细路径信息

```
Launch Clock Path
pin name            model name     delay (ns)    cumulative delay (ns)     pins on net    location
=============================================================================================
 pll_clk             inpad           0.000                  0.000            2           (0,118)
 pll_clk             inpad           0.200                  0.200            2           (0,118)
 pll_clk             io              0.000                  0.200            2           (0,118)
   Routing elements:
     Manhattan distance of X:1, Y:0
 CLKBUF__0|IO_in     gbuf_block      0.320                  0.520            2           (1,118)
 CLKBUF__0|I         gbuf            3.590                  4.110            2           (1,118)
 CLKBUF__0|O         gbuf            0.000                  4.110            6           (1,118)
 CLKBUF__0|clkout    gbuf_block      0.000                  4.110            6           (1,118)
 in1[1]~FF|CLK       ff              0.000                  4.110            6           (3,50)

Data Path
pin name            model name     delay (ns)    cumulative delay (ns)     pins on net    location
=============================================================================================
 in1[1]~FF|Q         ff              0.282                  0.282            2           (3,50)
 in1[1]~FF|O_seq     eft             0.476                  0.758            2           (3,50)
   Routing elements:
     Manhattan distance of X:0, Y:13
 out~FF|I[0]         eft             0.267                  1.025            2           (3,63)
 LUT__41|in[0]       lut             0.000                  1.025            2           (3,63)
 LUT__41|out         lut             0.000                  1.025            2           (3,63)
 out~FF|D            ff              0.118                  1.143            2           (3,63)

Capture Clock Path
pin name            model name     delay (ns)    cumulative delay (ns)     pins on net    location
=============================================================================================
 pll_clk             inpad           0.000                  0.000            2           (0,118)
 pll_clk             inpad           0.200                  0.200            2           (0,118)
 pll_clk             io              0.000                  0.200            2           (0,118)
   Routing elements:
     Manhattan distance of X:1, Y:0
 CLKBUF__0|IO_in     gbuf_block      0.320                  0.520            2           (1,118)
 CLKBUF__0|I         gbuf            3.590                  4.110            2           (1,118)
 CLKBUF__0|O         gbuf            0.000                  4.110            6           (1,118)
 CLKBUF__0|clkout    gbuf_block      0.000                  4.110            6           (1,118)
 out~FF|CLK          ff              0.000                  4.110            6           (3,63)
```

- **Launch Clock Path** : 发起时钟路径
- **Data Path** : 数据路径
  - tCO + tPD
- **Capture Clock Path**: 捕获时钟路径
- pin name : 路径节点名(pin)
- model name : 时序模型命名
- delay : 节点延迟
- cumulative : 路径累积延迟
- pins on net : 连接到该节点(pin)的信号(net)一共连接到了多少pin上
  - 表示信号扇出数量
  - 信号扇出越大，该节点延迟越大
- location : 该节点在FPGA Floorplan里的坐标
- Routing elements : 布线单元
  - Manhattan distance 是Quantum架构衡量布线延迟的单位
  - X是横坐标跨度 Y 是纵坐标跨度
  - Manhattan distance跨度越大延迟越大

最短关键路径详细报告

```
---------- Table of Contents (begin) ----------
   1. Clock Frequency Summary
   2. Clock Relationship Summary
   3. Path Details for Max Critical Paths
   4. Path Details for Min Critical Paths
---------- Table of Contents (end) ------------
```

```
---------- 2. Clock Relationship Summary (begin) ----------

Launch Clock    Capture Clock    Constraint (ns)    Slack (ns)    Edge
pin_clk         pin_clk                5.000         1.933      (R-R)  ①
pin_clk         pll_clk                2.500         1.421      (R-F)  ②
pll_clk         pll_clk                5.000         3.737      (R-R)  ③

NOTE: Values are in nanoseconds.

---------- Clock Relationship Summary (end) ---------------
```

```
---------- 4. Path Details for Min Critical Paths (begin) ----------


##########################################################################
Path Detail Report (pll_clk vs pll_clk)                                ③
##########################################################################
##########################################################################
Path Detail Report (pin_clk vs pin_clk)                                ①
##########################################################################
##########################################################################
Path Detail Report (pin_clk vs pll_clk)                                ②
##########################################################################

---------- Path Details for Min Critical Paths (end) ---------------
```

- 最短关键路径(Min Critical Path)
  - 一对相关联的发起时钟(Launch Clock)和捕获时钟之间(Capture Clock)所有的路径中延时最小的一条路径。
- 最短关键路径详细报告
  - 时序报告第4部分
  - 对应于时序报告第2部分，时钟关系简报列出的各相关时钟的最短路径
  - 决定了路径是否能满足保持时间 t_{HD}

最短关键路径详细报告-基本路径信息

- Path Begin : 路径起点
- Path End : 路径终点
- Launch Clock : 发起时钟
- Capture Clock : 捕获时钟
- Slack : 时序余量（arrival time - Require time）
  实际到达时间 - 要求到达时间
- Delay : 组合逻辑传播延迟(FF Q到FF D)， t_{PD}
- Logic Level : 组合逻辑级数
- Non-global nets on path : 组合逻辑路径非全局信号数
- Global nets on path : 组合逻辑路径全局信号数
- End-of-Path arrival time : 实际到达时间
  - Launch Clock Path Delay: 发起时钟路径延迟，t_{LCPD}
  - Clock To Q : 路径起点FF的时钟到输出时间，t_{CO}
  - Data Path Delay : 组合逻辑传播延迟， t_{PD}
  - Launch Clock Path Delay + Clock To Q + Data Path Delay
  - 实际到达时间 : $t_{LCPD} + t_{CO} + t_{PD}$

```
Path Begin     : in1[0]~FF|CLK
Path End       : out~FF|D
Launch Clock   : pll_clk (RISE)
Capture Clock  : pll_clk (RISE)
Slack          : 0.377  (arrival time - required time)
Delay          : 0.296

Logic Level : 1
Non-global nets on path : 1
Global nets on path     : 0

Launch Clock Path Delay        : 2.055
+ Clock To Q + Data Path Delay : 0.437
-----------------------------------------
End-of-path arrival time       : 2.492

Constraint                     : 0.000
+ Capture Clock Path Delay     : 2.055
+ Clock Uncertainty            : 0.060
-----------------------------------------
End-of-path required time      : 2.115
```

- End-of-path require time : 要求到达时间
  - Constraint : 对应发起时钟边沿之后捕获时钟的当前边沿，0
  - Capture Clock Path Delay : 捕获时钟路径延迟， t_{CCPD}
  - Clock Uncertainty : 时钟不确定性，在FPGA里视为Jitter，t_{JT}
  - Constraint + Capture Clock Path Delay + Clock Uncertainty
  - 要求到达时间 : t_{CCPD} + Uncertainty

保持时间时序余量(Slack) = 实际到达时间 - 要求到达时间 = $(t_{LCPD} + t_{CO} + t_{PD}) - (t_{CCPD}$ + Uncertainty) = $t_{CO} + t_{PD} - t_{JT} - t_{SK} \geq 0$ => $t_{PD} \geq t_{JT} + t_{SK} - t_{CO}$

最短关键路径详细报告-详细路径信息

```
Launch Clock Path
pin name            model name     delay (ns)    cumulative delay (ns)     pins on net    location
=============================================================================================
 pll_clk             inpad           0.000                  0.000            2          (0,118)
 pll_clk             inpad           0.100                  0.100            2          (0,118)
 pll_clk             io              0.000                  0.100            2          (0,118)
   Routing elements:
     Manhattan distance of X:1, Y:0
 CLKBUF__0|IO_in     gbuf_block      0.160                  0.260            2          (1,118)
 CLKBUF__0|I         gbuf            1.795                  2.055            2          (1,118)
 CLKBUF__0|O         gbuf            0.000                  2.055            6          (1,118)
 CLKBUF__0|clkout    gbuf_block      0.000                  2.055            6          (1,118)
 in1[0]~FF|CLK       ff              0.000                  2.055            6          (3,61)

Data Path
pin name            model name     delay (ns)    cumulative delay (ns)     pins on net    location
=============================================================================================
 in1[0]~FF|Q         ff              0.141                  0.141            2          (3,61)
 in1[0]~FF|O_seq     eft             0.238                  0.379            2          (3,61)
   Routing elements:
     Manhattan distance of X:0, Y:2
 out~FF|I[3]         eft             0.058                  0.437            2          (3,63)
 LUT__41|in[3]       lut             0.000                  0.437            2          (3,63)
 LUT__41|out         lut             0.000                  0.437            2          (3,63)

Capture Clock Path
pin name            model name     delay (ns)    cumulative delay (ns)     pins on net    location
=============================================================================================
 pll_clk             inpad           0.000                  0.000            2          (0,118)
 pll_clk             inpad           0.100                  0.100            2          (0,118)
 pll_clk             io              0.000                  0.100            2          (0,118)
   Routing elements:
     Manhattan distance of X:1, Y:0
 CLKBUF__0|IO_in     gbuf_block      0.160                  0.260            2          (1,118)
 CLKBUF__0|I         gbuf            1.795                  2.055            2          (1,118)
 CLKBUF__0|O         gbuf            0.000                  2.055            6          (1,118)
 CLKBUF__0|clkout    gbuf_block      0.000                  2.055            6          (1,118)
 out~FF|CLK          ff              0.000                  2.055            6          (3,63)
```

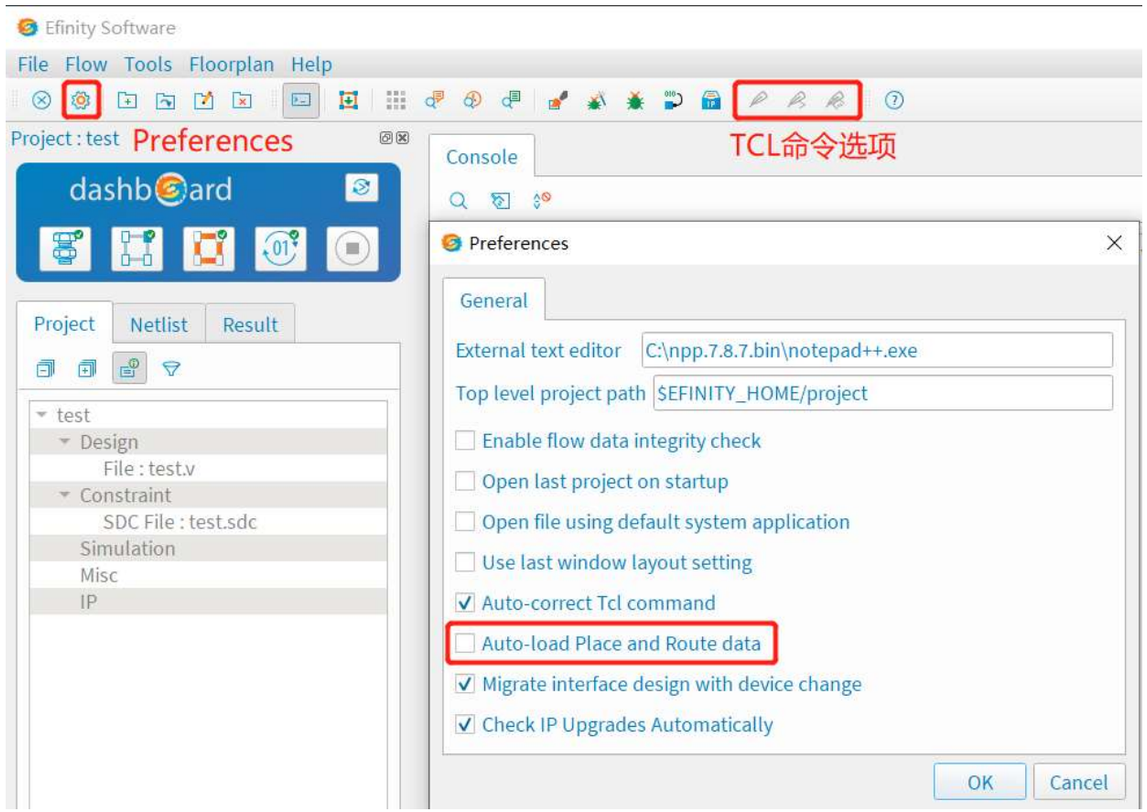
- **Launch Clock Path** : 发起时钟路径
- **Data Path** : 数据路径
  - tCO + tPD
- **Capture Clock Path**: 捕获时钟路径
- pin name : 路径节点名(pin)
- model name : 时序模型命名
- delay : 节点延迟
- cumulative : 路径累积延迟
- pins on net : 该节点(pin)的信号(net)一共连接到了多少pin上
  - 表示信号扇出数量
  - 信号扇出越大，该节点延迟越大
- location : 该节点在FPGA Floorplan里的坐标
- Routing elements : 布线单元
  - Manhattan distance 是Quantum架构衡量布线延迟的单位
  - X是横坐标跨度 Y是 纵坐标跨度
  - Manhattan distance跨度越大延迟越大

时序分析

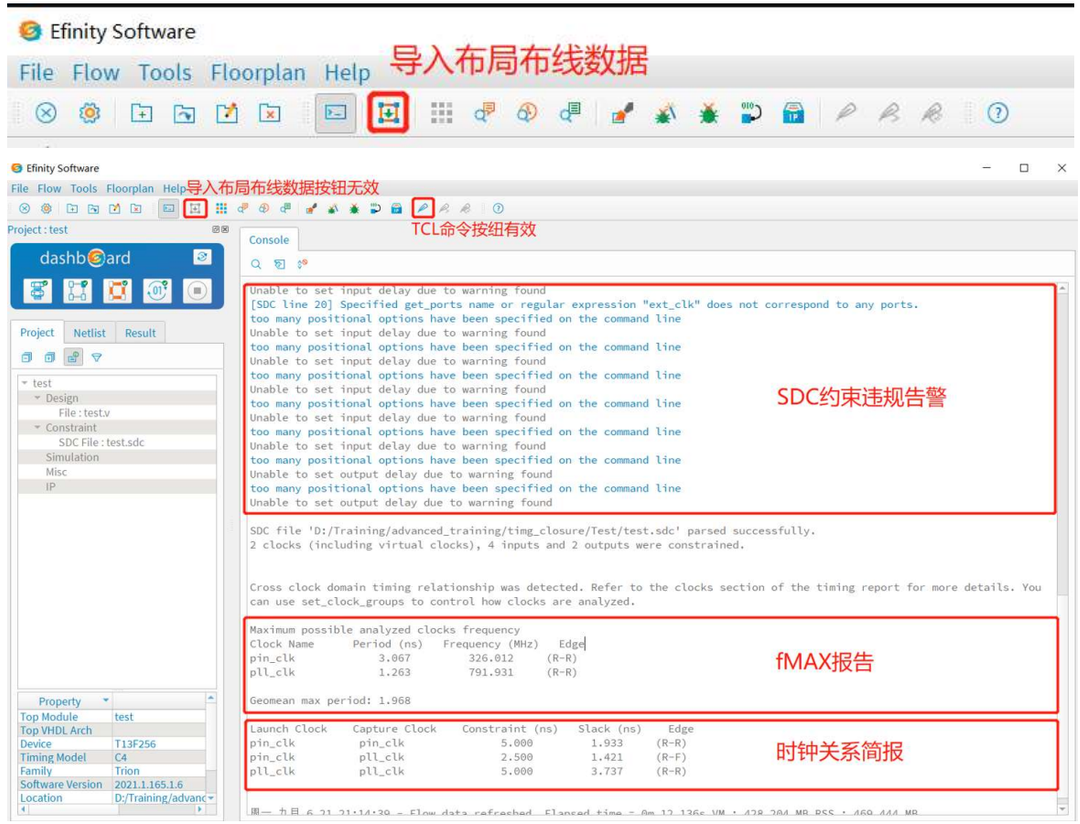
- 时序报告
  - 时序报告简述
  - 详细时序报告(<项目名>.timing.rpt)
    - 时序模型信息
    - 时钟频率简报
    - 时钟关系简报
    - 最长关键路径详细报告
    - 最短关键路径详细报告
- 时序分析TCL命令
  - report_timing
  - 其它命令

时序分析TCL命令

- 为了节约打开已存在项目的数据导入时间和项目编译时间，通常我们会将优选项菜单中的自动加载布局布线数据选项去掉
- 这种情况下，TCL命令行按钮选项是灰色的，处于无法使用状态



- 点击导入布局布线数据按钮，数据导入完成以后按钮变为无效，同时TCL命令按钮变为有效
- 数据导入后，控制台会自动检查SDC约束文件，打印SDC约束违规告警，列出f_{MAX}报告以及时钟关系简报



report_timing

- **report_timing [-detail summary|path_only|path_and_clock|full_path] [-file <*name*>] [-from_clock <*names*>] [-from <*names*>] [-fall_from_clock <*names*>] [-rise_from_clock <*names*>] [-less_than_slack <*slack limit*>] [-npaths <*number*>] [-nworst <*number*>] [-show_routing] [-stdout] [-through <*names*>] [–to <*names*>] [-to_clock <*names*>] [-rise_to_clock <*names*>] [-fall_to_clock <*names*>] [-id <*number*>] [-hold] [-setup]**
  - report_timing时序分析命令是最全面和最有用的一条命令，只要掌握report_timing命令就可以进行全面的详细时序分析。
- **-detail**
  - 指定报告内容, 如果不定义该参数则默认为full_path, 可选变量 summary, path_only, path_and_clock, full_path。
  - 也可使用-detail summary参数，只列出路径的余量，延迟，起始点，发起时钟和捕获时钟。
  - 一般不用定义，默认为full_path，报告列出路径所有细节，格式和timing.rpt相同。
- **-file**
  - 指定存放报告信息的文件名<name>，如果定义则会在outflow里生成包含报告信息以指定文件命名的文本文件。
  - 一般不用定义，默认输出为软件的控制台。
- **-from_clock和-to_clock**
  - 指定报告路径的发起时钟名<name>和捕获时钟名<name>，可使用通配符*匹配多个时钟。
  - 常用，需要指定，对特定发起时钟和捕获时钟的关键路径进行详细分析。

接下一页

report_timing

- **-from和-to**
  - 指定报告路径的起始点名，可以是设计里任意有效的对象。
  - 一般不用定义，通常我们的时序分析针对时钟就足够了。
- **-fall_from_clock, -rise_from_clock 和 -fall_to_clock, -rise_to_clock**
  - 报告只显示指定发起时钟边沿和捕获时钟边沿的路径。
  - 按需定义，如果设计里有半周期捕获，需要分析半周期捕获的关键路径时序是，则需要定义。_from_ 和 _to_是配对使用的。
- **-less_than_slack**
  - 报告只显示时序余量低于本定义的路径。
  - 按需定义。通常用在时序未收敛的情况下，报告余量小于0的路径。
- **-npaths和-nworst**
  - 常用，一般需要定义。
  - -npaths指定报告的路径数。如果没有指定这个参数，则只报告一条最长的路径。
  - -nworst限制每个目标节点报告的路径数量。
  - 如果没有定义-nworst, 每个目标节点报告的路径数量仅受到-npaths限制 。
  - 如果只定义-nworst没定义-npaths, 则-npaths默认和-nworst相同。
  - 对发起时钟和捕获时钟的时序路径分析，在同步时序系统里，通常不会出现多个起点到同一个终点的情况，所以通常-npaths和-nworst只用其中一个即可。

report_timing

- **-show_routing**
  - 路径报告中显示布线信息(Manhattan距离)
  - 有条件使用，如果发现路径组合逻辑级数很少，路径中节点扇出比也正常的情况下，总路径延迟很大，则可使用此参数查看是否布线延迟过大。
- **-stdout**
  - 将报告信息显示到控制台
  - 如果没有使用-file参数，则默认显示到控制台。如果使用-file，则需要使用-stdout才可以同时在控制台显示。
  - 不常用，一般不定义。
- **-though**
  - 只报告通过指定pins或者nets的路径，并且只能是组合逻辑上的pins或者nets。
  - 不常用，一般不用定义。
- **-id**
  - 定义时序报告在 Timing Brower中的序号。如果使用Timing Brower可选用，如果不定义，系统会自动分配序号。
  - 不常用，一般不用定义。
- **-setup和-hold**
  - -setup报告时钟建立时间路径（最长关键路径）。
  - -hold报告时钟保持时间路径（最短关键路径）。
  - 按需定义，如果不定义，则默认按-setup报告路径。

其它命令

- report_timing_summary
  - 报告当前项目时钟的f_{MAX}
  - 报告当前项目时钟之间的关系和最差路径时序余量
- report_clocks
  - 报告当前时钟约束
  - 报告当前项目时钟的f_{MAX}
  - 报告当前项目时钟之间的关系和最差路径时序余量
- report_path
  - 报告指定路径信息
  - report_timing的子集
- delete_timing_results
  - 删除内存中使用report_path和report_timing命令获取的时序报告数据
- get_available_timing_model
  - 在控制台查看当前器件型号所在系列支持的所有时序模型
- get_timing_model
  - 在控制台查看当前工程使用的时序模型
- read_sdc <file>
  - 从指定文件读入时序约束到当前工程
- reset_timing
  - 删除当前工程内存中所有的时序分析数据，约束和路径信息
- set_timing_model
  - 重新指定当前工程的时序模型
- write_sdc <file>
  - 将当前工程内存中的时序约束SDC写到指定文件里

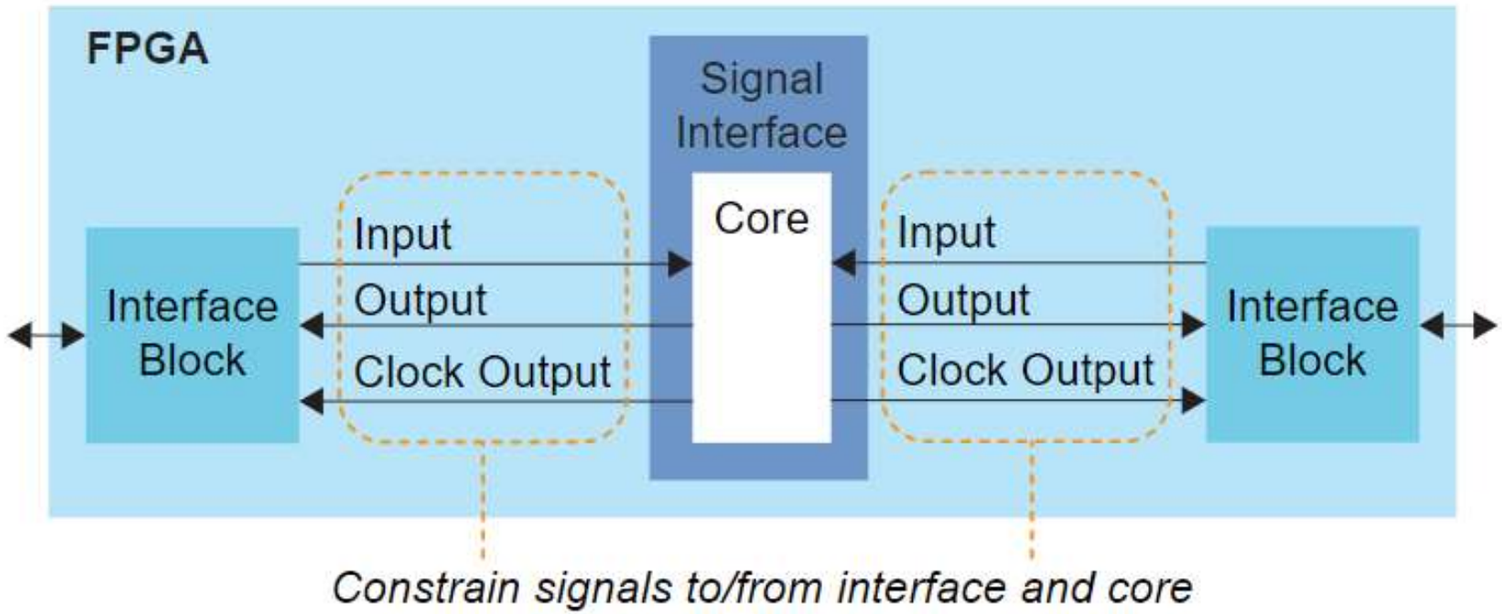
主要内容

1. 理论基础
2. 时序分析
3. 时序约束
4. 时序优化策略

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的时序信息导入针到对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

```
---------- 2. GPIO Timing Report (begin) ----------

Clock Network Delay:
====================

+------------+----------+----------+
| Clock Pin  | Max (ns) | Min (ns) |
+------------+----------+----------+
|  pin_clk   |  4.110   |  2.055   |
| pll_clkout |  4.110   |  2.055   |
+------------+----------+----------+

Clkout GPIO Configuration:
==========================

+---------------+------------+--------------+----------+----------+
| Instance Name | Clock Pin  |  Parameter   | Max (ns) | Min (ns) |
+---------------+------------+--------------+----------+----------+
|     clkout    | pll_clkout | GPIO_CLK_OUT |  7.939   |  3.970   |
+---------------+------------+--------------+----------+----------+

Non-registered GPIO Configuration:
==================================

+---------------+----------+-------------+----------+----------+
| Instance Name | Pin Name |  Parameter  | Max (ns) | Min (ns) |
+---------------+----------+-------------+----------+----------+
|    ext_clk    | ext_clk  |   GPIO_IN   |  1.796   |  0.898   |
|     in[0]     |  in[0]   |   GPIO_IN   |  1.396   |  0.698   |
|     in[1]     |  in[1]   |   GPIO_IN   |  1.396   |  0.698   |
|    pin_clk    | pin_clk  | GPIO_CLK_IN |  1.275   |  0.637   |
|      out      |   out    |   GPIO_OUT  |  3.829   |  1.915   |
+---------------+----------+-------------+----------+----------+

Registered GPIO Configuration:
==============================

+---------------+-----------+------------+-----------+----------------------+----------------------+
| Instance Name | Clock Pin | Setup (ns) | Hold (ns) | Max Clock To Out (ns) | Min Clock To Out (ns) |
+---------------+-----------+------------+-----------+----------------------+----------------------+
|   in_reg[0]   |  pin_clk  |   -2.219   |   1.981   |                      |                      |
|   in_reg[1]   |  pin_clk  |   -2.219   |   1.981   |                      |                      |
|    out_reg    |  pin_clk  |            |           |        8.649         |        4.325         |
+---------------+-----------+------------+-----------+----------------------+----------------------+

---------- 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_OUT②延迟
    - Clkout GPIO Configuration
      - GPIO_CLK_OUT②输出延迟
    - Non-registered GPIO Configuration
      - 未使用IO寄存的GPIO延迟
    - Registered GPIO Configuration
      - Input REG对输入到FPGA管脚信号的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

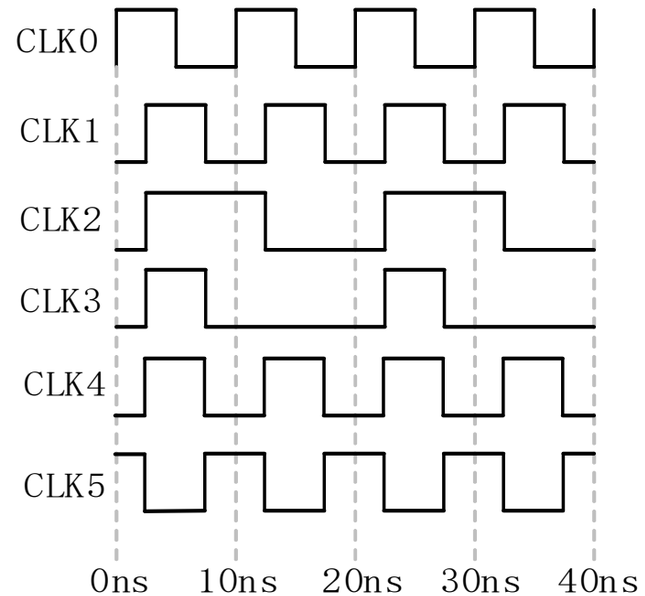
- <项目名>.pt.sdc是Interface Designer生成的针对Core的SDC模板
  - PLL Constraints
    - 提供锁相环输出时钟的完整定义
    - 用户根据Core逻辑具体应用，按实际需要增加时钟关系约束，虚拟时钟约束，生成时钟约束
  - GPIO Constraints
    - 提供GCLK输入的时钟定义模板，由用户根据实际应用计算后定义
    - Registered GPIO
      - 提供IO寄存器和Core之间输出输出延迟的完整约束
    - Non-registered GPIO
      - 提供未寄存GPIO和Core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

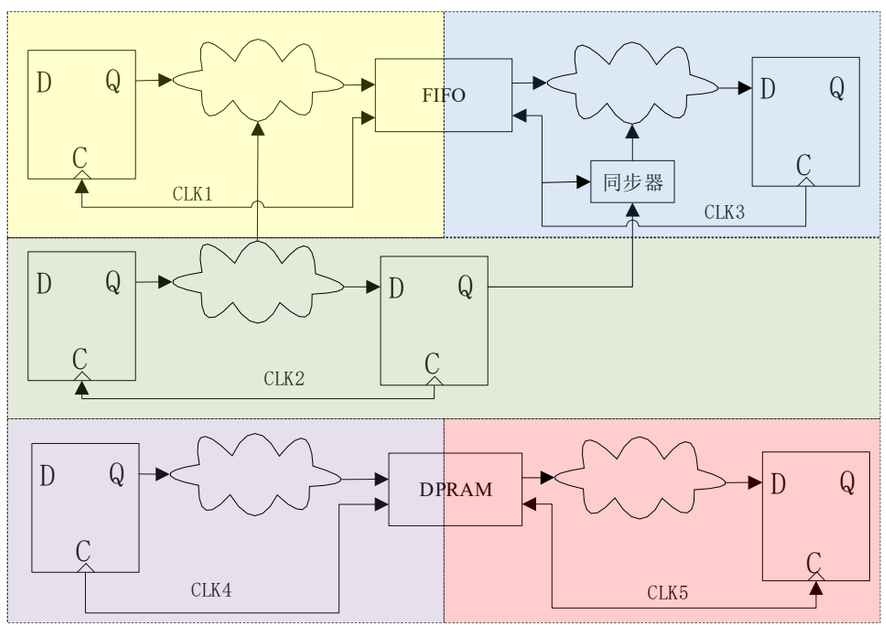
基准时钟 : CLK0,CLK1,CLK3

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

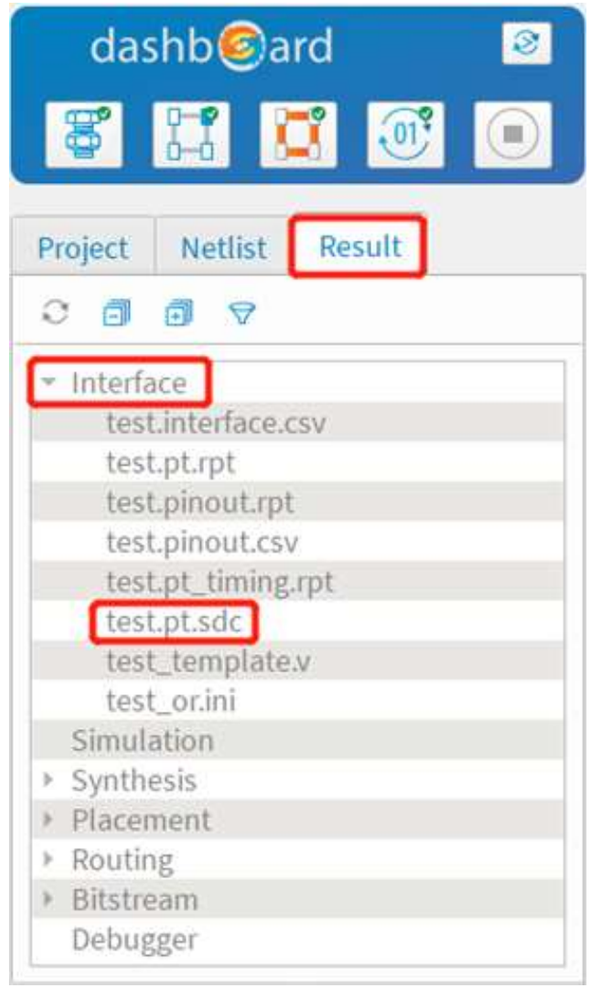
- 针对CORE逻辑的约束
  - Interface Designer时序报告
  - Interface Designer时序约束文件
- 时钟约束
  - 定义时钟
  - 约束时钟之间的关系
  - 定义时钟的不确定性
- IO约束
  - 同步IO和非同步IO
  - 约束同步IO
  - 约束非同步IO
- 约束时序例外
  - 路径例外约束
  - 多周期路径约束
  - 路径延迟约束
- 建立SDC文件
  - 约束文件的基本要求
  - 建立和添加约束文件
  - SDC小知识

同步IO和非同步IO

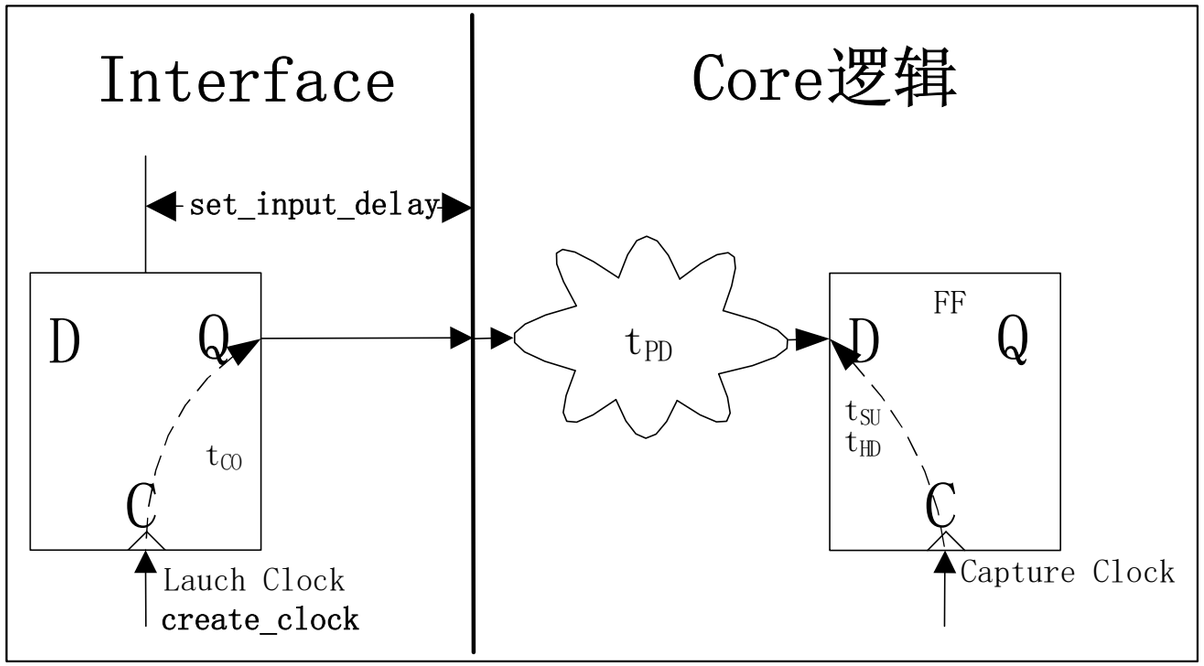
- 同步IO
  - 同步IO是指Interface里有输入输出寄存的IO连接
    - 输入路径起点是Interface
    - 输出路径终点是Interface
  - 几乎所有的Interface同Core的连接方式都是同步IO连接
    - LVDS Serializer/ Deserializer mode
    - 有IO REG的GPIO
    - MIPI硬核
    - DDR硬核控制器
    - MIPI DPHY
    - HyperRAM
    - JTAG
- 非同步IO
  - 非同步IO是指外设器件的信号直接穿过Interface的IO连接
  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



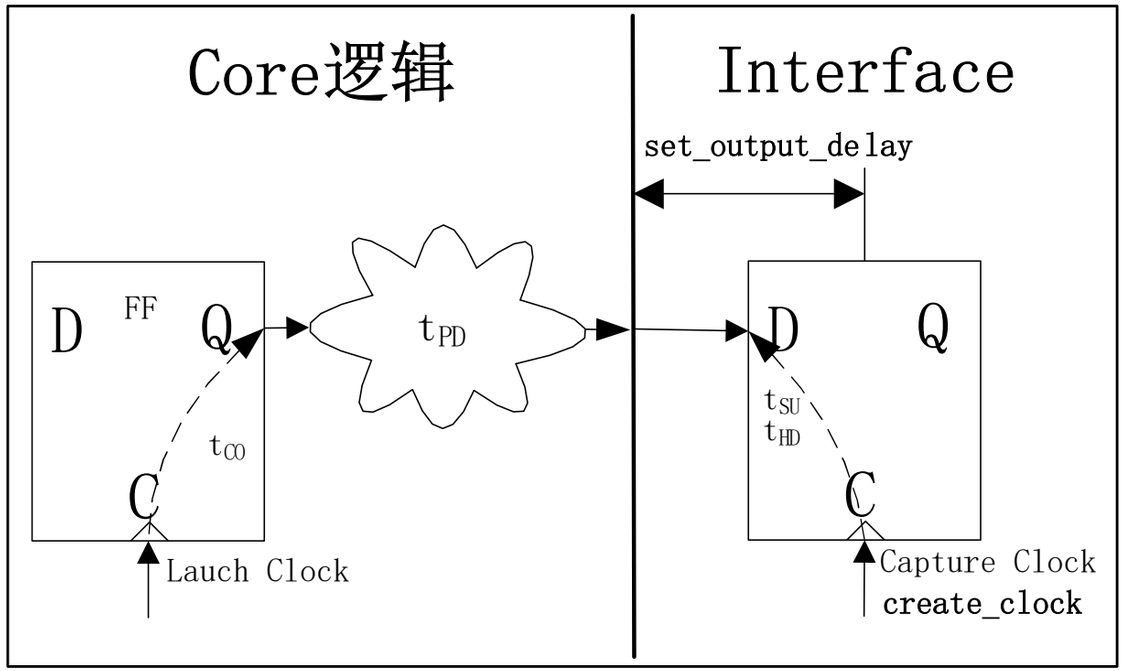
约束同步输入 set_input_delay




```
# Efinity Interface Designer SDC
# Version: 2021.1.165.1.6
# Date: 2021-08-21 01:53

# Copyright (C) 2017 - 2021 Efinix Inc. All rights reserved.

# Device: T13F256
# Project: test
# Timing Model: C4 (final)

# PLL Constraints
#################
create_clock -period 5.00 pll_clk
create_clock -period 5.00 [get_ports {pin_clk}]

# GPIO Constraints
###################
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {ext_clk}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {ext_clk}]
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[0]}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[0]}]
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[1]}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[1]}]
set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[0]}]
set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[0]}]
set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[1]}]
set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[1]}]
# set_output_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {out}]
# set_output_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {out}]
set_output_delay -clock pin_clk -max -3.500 [get_ports {out_reg}]
set_output_delay -clock pin_clk -min -1.971 [get_ports {out_reg}]
```

- 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

约束同步输出 set_output_delay



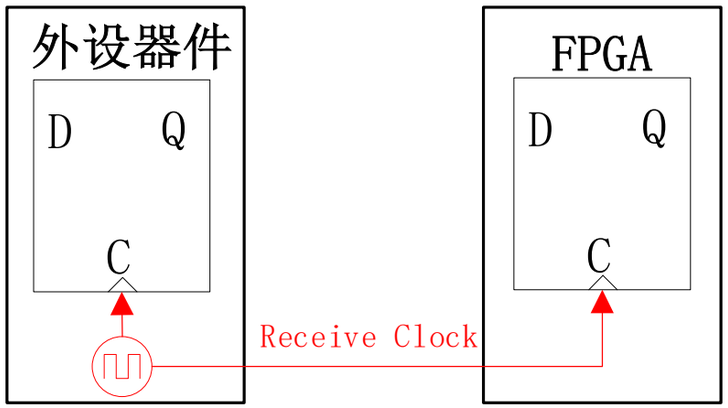
- 路径终点(Capture Path)是Interface
  - Efinity Dashboard->Result->Interface
  - 打开文件：<项目名>.pt.sdc
  - 复制set_output_delay约束到工程SDC文件里

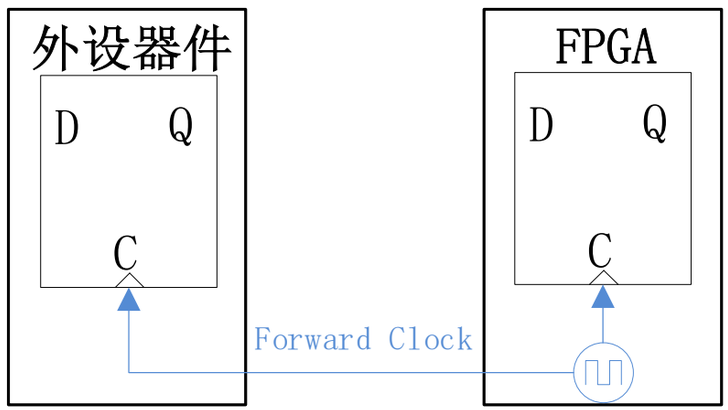
```
# Efinity Interface Designer SDC
# Version: 2021.1.165.1.6
# Date: 2021-08-21 01:53

# Copyright (C) 2017 - 2021 Efinix Inc. All rights reserved.

# Device: T13F256
# Project: test
# Timing Model: C4 (final)

# PLL Constraints
#################
create_clock -period 5.00 pll_clk
create_clock -period 5.00 [get_ports {pin_clk}]

# GPIO Constraints
####################
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {ext_clk}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {ext_clk}]
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[0]}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[0]}]
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[1]}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[1]}]
set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[0]}]
set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[0]}]
set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[1]}]
set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[1]}]
# set_output_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {out}]
# set_output_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {out}]
set_output_delay -clock pin_clk -max -3.500 [get_ports {out_reg}]
set_output_delay -clock pin_clk -min -1.971 [get_ports {out_reg}]
```

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

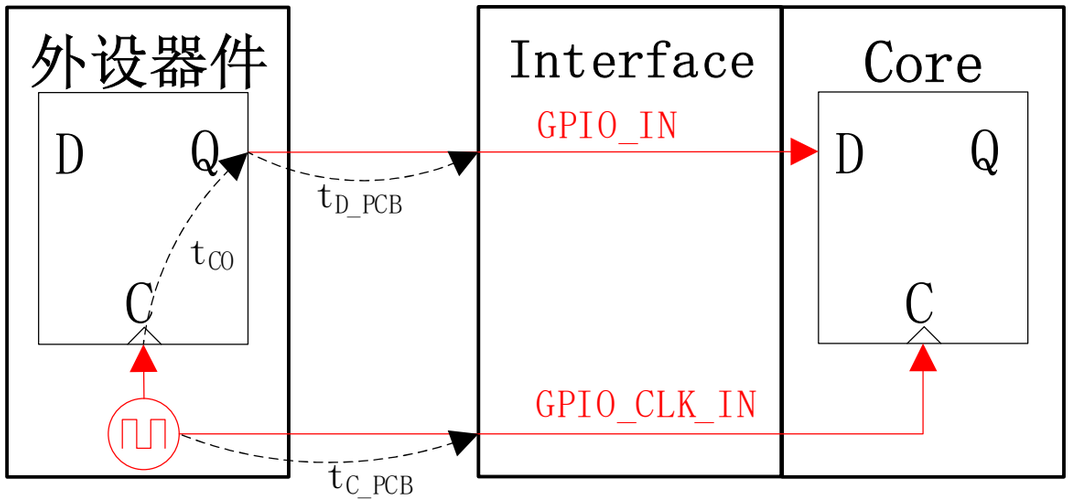
<project name>.pt_timing.rpt

```
Non-registered GPIO Configuration:
==================================

+---------------+----------+-------------+----------+----------+
| Instance Name | Pin Name |  Parameter  | Max (ns) | Min (ns) |
+---------------+----------+-------------+----------+----------+
|    ext_clk    | ext_clk  |   GPIO_IN   |  1.796   |  0.898   |
|     in[0]     |  in[0]   |   GPIO_IN   |  1.396   |  0.698   |
|     in[1]     |  in[1]   |   GPIO_IN   |  1.396   |  0.698   |
|    pin_clk    | pin_clk  | GPIO_CLK_IN |  1.275   |  0.637   |
|      out      |   out    |   GPIO_OUT  |  3.829   |  1.915   |
+---------------+----------+-------------+----------+----------+
```

<project name>.pt.sdc

```
# GPIO Constraints
####################
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {ext_clk}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {ext_clk}]
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[0]}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[0]}]
# set_input_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {in[1]}]
# set_input_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {in[1]}]
set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[0]}]
set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[0]}]
set_input_delay -clock pin_clk -max 4.968 [get_ports {in_reg[1]}]
set_input_delay -clock pin_clk -min 2.484 [get_ports {in_reg[1]}]
# set_output_delay -clock <CLOCK> -max <MAX CALCULATION> [get_ports {out}]
# set_output_delay -clock <CLOCK> -min <MIN CALCULATION> [get_ports {out}]
set_output_delay -clock pin_clk -max -3.500 [get_ports {out_reg}]
set_output_delay -clock pin_clk -min -1.971 [get_ports {out_reg}]

```

约束Input Receive 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

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

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

```
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

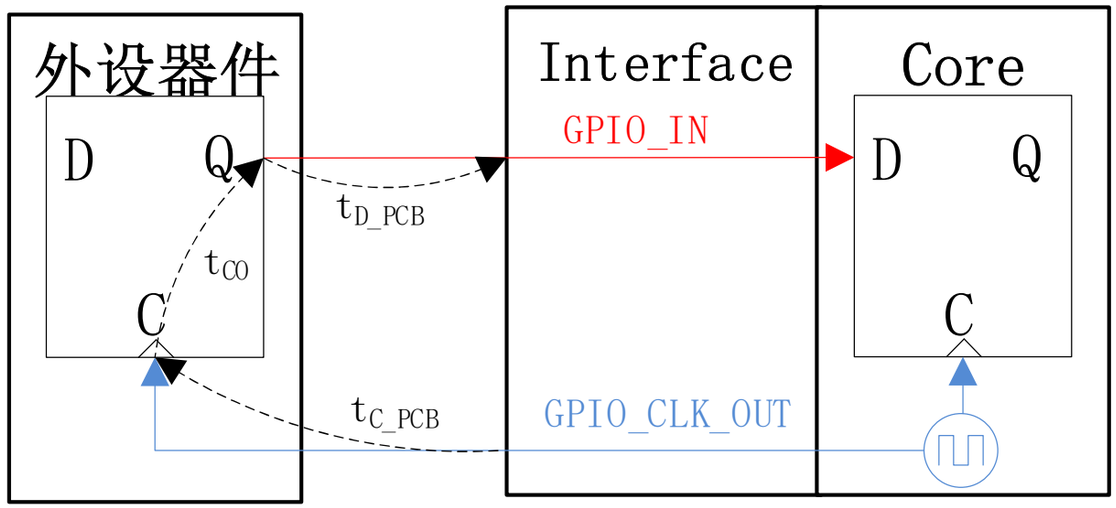
  <min calculation> = 2 + 0.526 - 0.526 = 2

- SDC

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

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

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

<max calculation> = <max board constraint> + GPIO_INmax + GPIO_CLK_OUTmax

<min calculation> = <min board constraint> + GPIO_INmin + 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_{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 +6.834 = 12.788

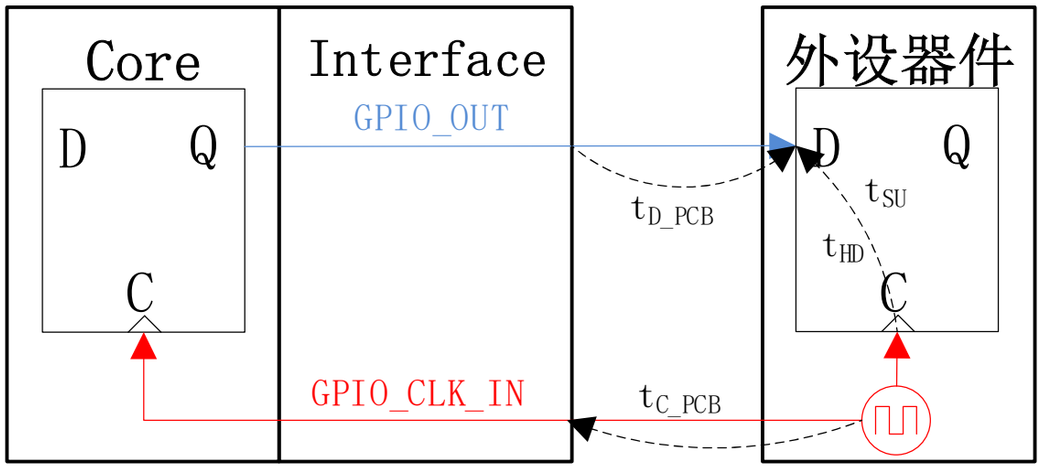
  <min calculation> = 2 + 0.526 + 4.401 = 6.927

- SDC

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

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

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

<max calculation> = <max board constraint> + GPIO_OUTmax + GPIO_CLK_INmax

<min calculation> = <min board constraint> + GPIO_OUTmin + GPIO_CLK_INmin

```
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 + 1.954 = 10.2

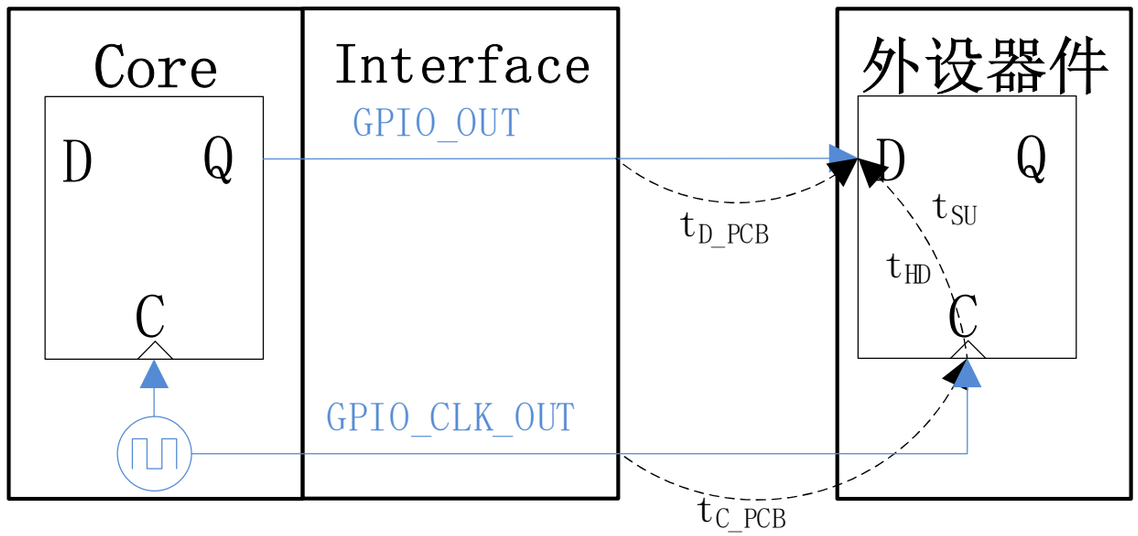
<min calculation> = 2 + 1.081 + 0.526 = 3.607

- SDC

set_output_delay -clock clkin -max 10.2 [get_ports {dout}]

set_output_delay -clock clkin -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}]

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

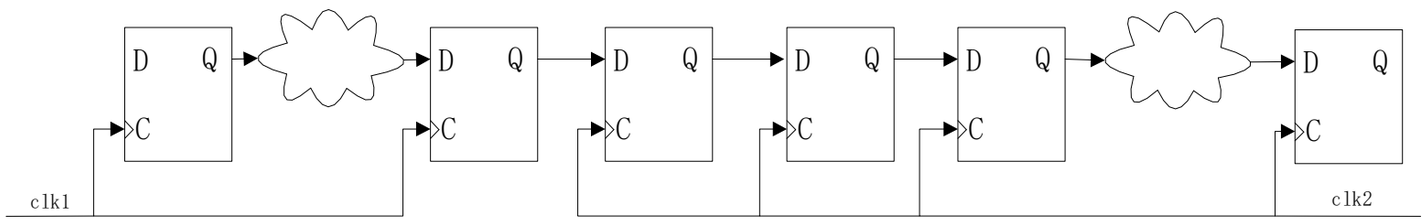
- 针对CORE逻辑的约束
  - Interface Designer时序报告
  - Interface Designer时序约束文件
- 时钟约束
  - 定义时钟
  - 约束时钟之间的关系
  - 定义时钟的不确定性
- IO约束
  - 同步IO和非同步IO
  - 约束同步IO
  - 约束非同步IO
- 约束时序例外
  - 路径例外约束
  - 多周期路径约束
  - 路径延迟约束
- 建立SDC文件
  - 约束文件的基本要求
  - 建立和添加约束文件
  - SDC小知识

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

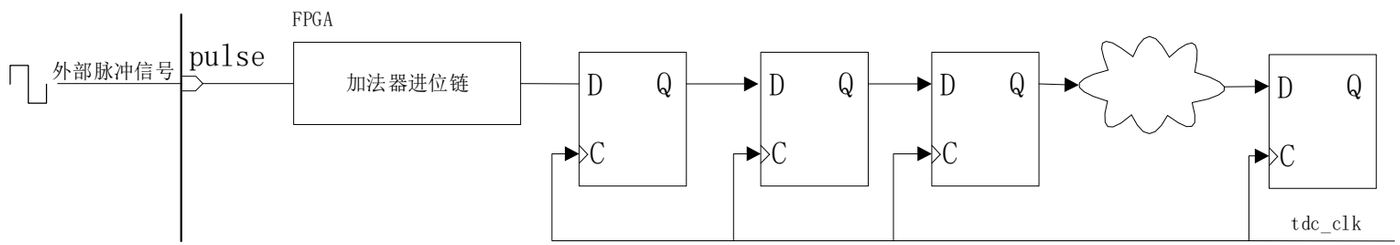
- **set_false_path**
  - 可用于切断时钟之间的关联，使时序引擎不分析异步的发起时钟和捕获时钟之间的时序关系，等效于set_clock_groups约束。这种应用场景，我们推荐使用set_clock_groups.
  - set_false_path在需要针对寄存器或者输入输出端口进行路径例外约束的时候更加有用。或者用在只需要切断两个不同时钟之间某一个方向的路径的时候。
  - Efinity不支持端口到端口的路径例外约束，也就是说在使用set_false_path的时候，from或者to至少有一个是时钟。



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。

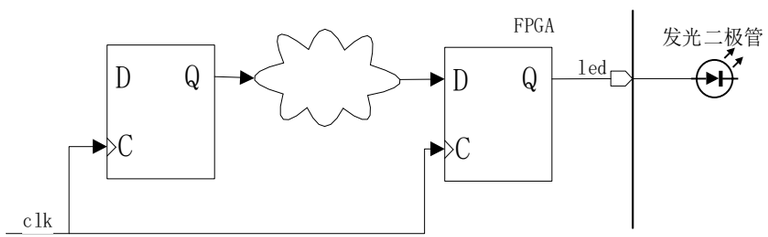
set_false_path –from clk1 –to clk2



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

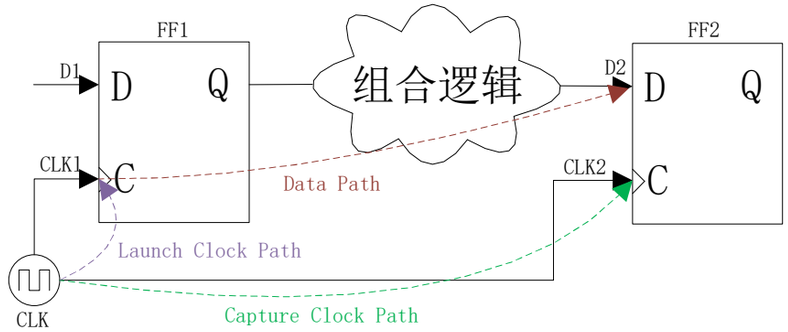


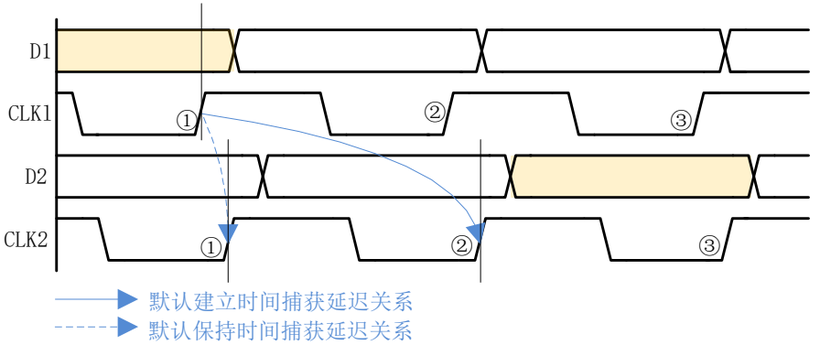
时钟到端口的路径例外约束案例

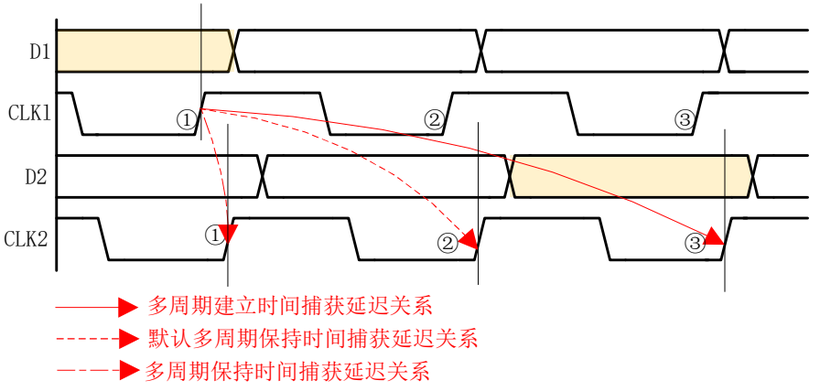
常见的LED闪灯设计，FPGA输出同LED没有任何时钟同步关系，也不需要进行时序分析。

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
  - Data Path-(Capture Clock Path – 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

  隐含红色虚线表示的保持时间捕获延迟关系

  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

易 灵 思

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

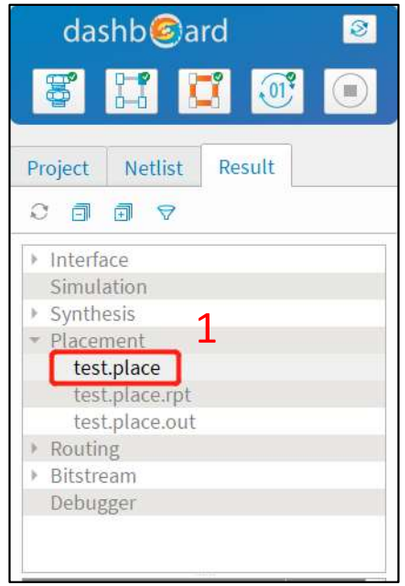
```
set_max_delay -from <clock> -to <clock> <delay>
set_min_delay -from <clock> -to <clock> <delay>
```

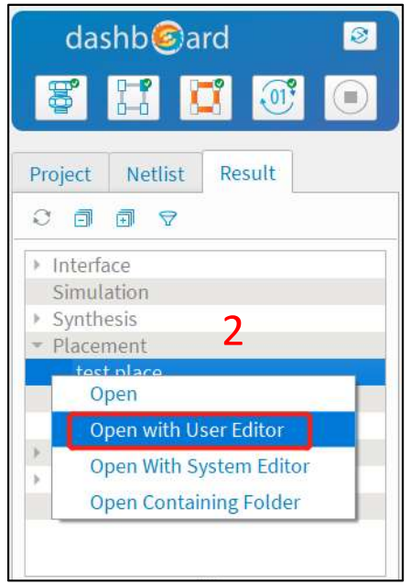
时序约束

- 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

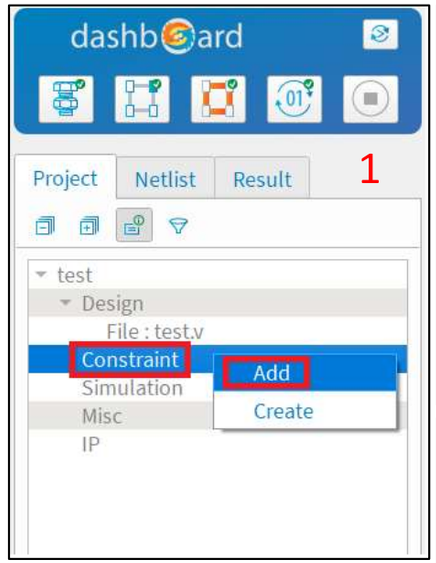


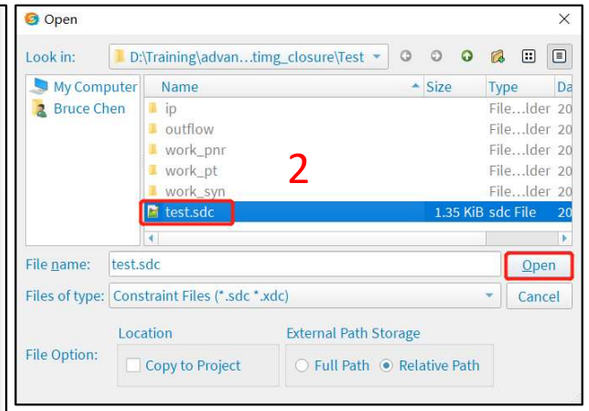


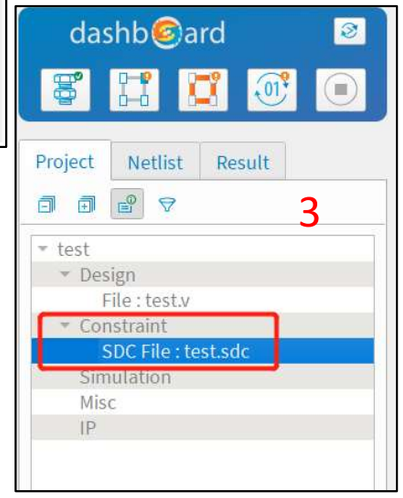

```
66  WARNING(1):
67  1 Error(s) found in the SDC file D:/Training/advanced_training/timg_closure/Test/
    test.sdc
68  Error processing line 15:
69  missing close-brace                    3   <项目名>.place.out
70      while executing
71  "create_clock -period 5.00 [get_ports {"
72
73
74  SDC file 'D:/Training/advanced_training/timg_closure/Test/test.sdc' blank or does
     not contain valid constraint or not found.
75  Using default timing constraint of 1 ns.
76
```

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

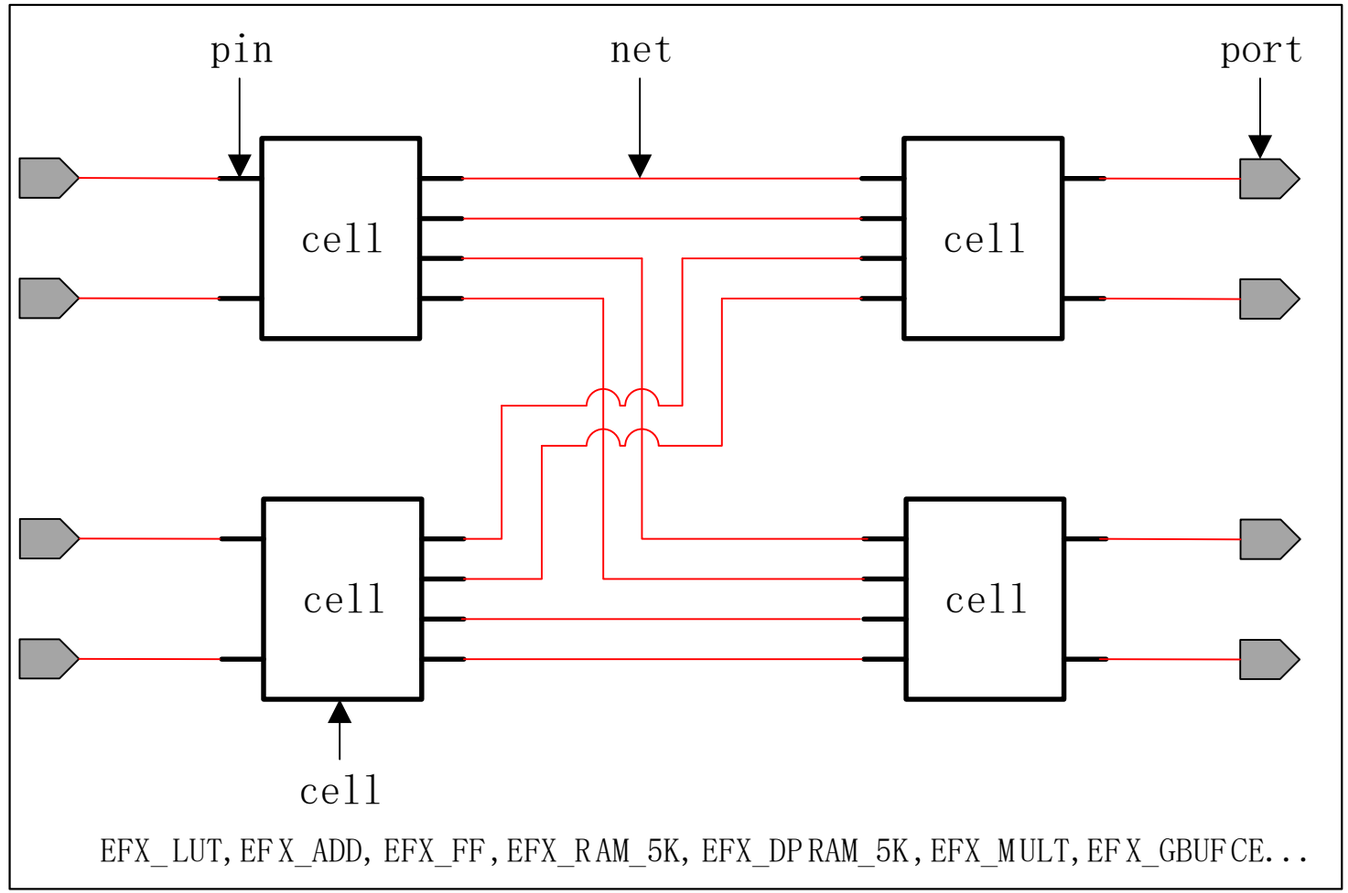






SDC小知识

- # 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. 理论基础
2. 时序分析
3. 时序约束
4. 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）

接下一页

综合策略

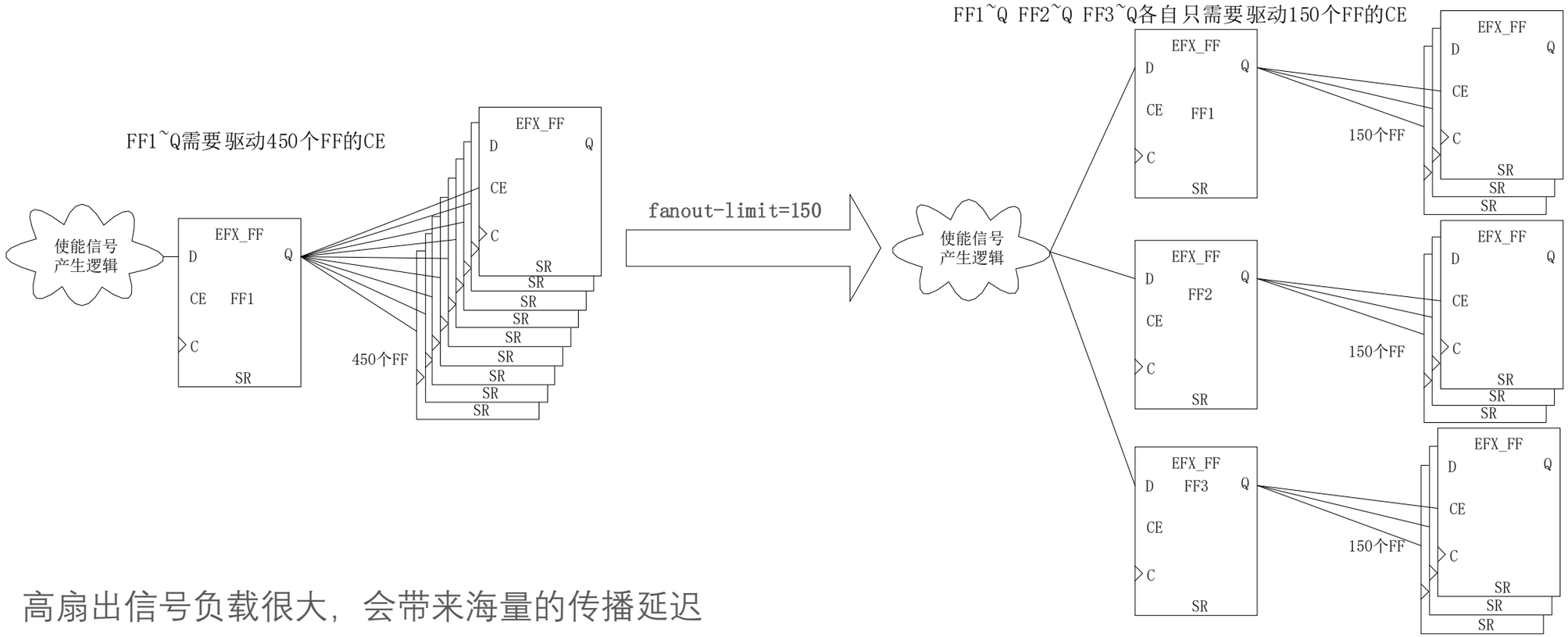
综合策略名称	功能描述	选项	设置项功能描述
infer-sync-set-reset	同步置位复位推断	0	禁止同步置位复位信号推断
		1	打开同步置位复位信号推断（默认）
fanout-limit	信号扇出数量限制	0	不限制扇出数量（默认）
		n	限制信号可扇出的最大数量，超过限制综合器自动复制高扇出信号源; 一般高扇出信号是逻辑产生的使能和复位信号; 在RTL设计的时候，高扇出信号推荐使用寄存器输出。
seq_opt	顺序优化	0	禁止优化
		1	打开优化（默认） 综合器分析寄存器所有可能的状态顺序，并按等效的顺序合并信号。总的来说，这是一个面积优化的选项，能有效减少LUT的使用数量。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。

接下一页

综合策略

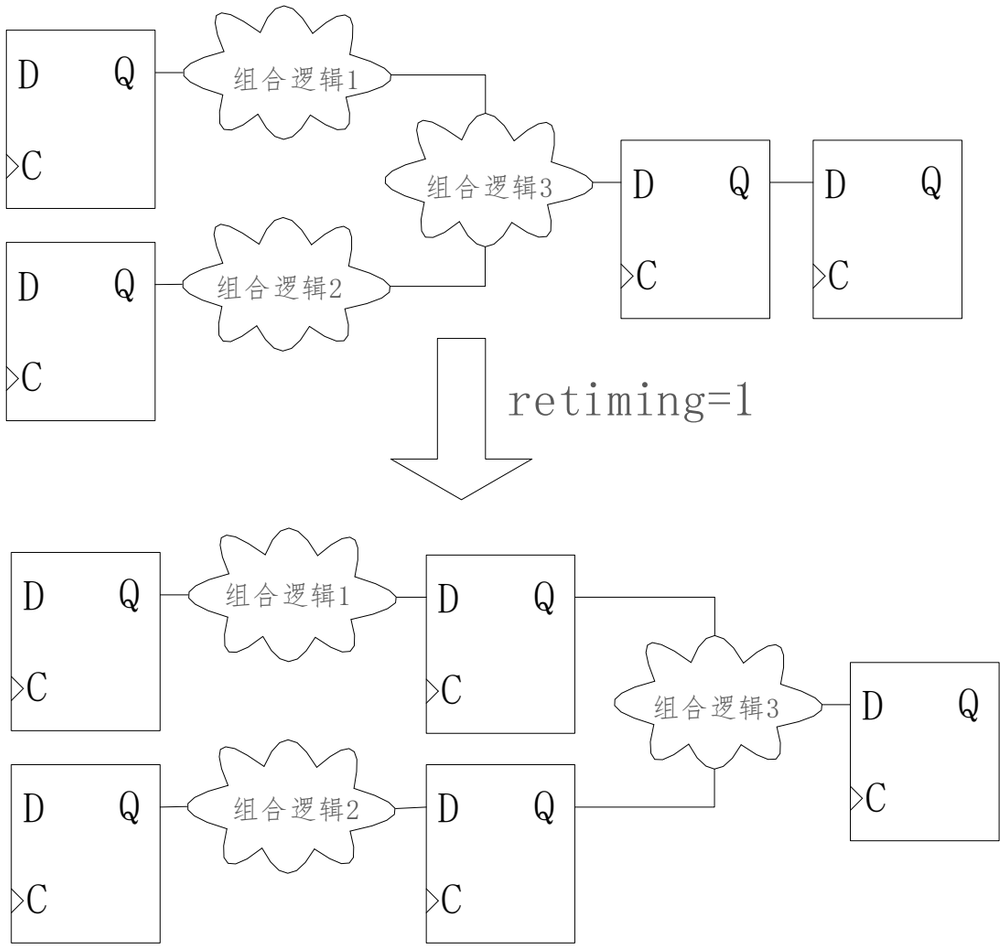
综合策略名称	功能描述	选项	设置项功能描述
blast_const_operand_adders	常数操作数优化	0	禁止优化（默认）
		1	打开优化 如果运算的某操作数为常数，综合器将其综合为组合逻辑。
dsp-mac-packing	DSP乘累加打包	0	禁止优化
		1	打开优化（默认） 允许DSP乘累加打包，综合器将符合MAC描述的加法和乘法运算打包用DSP实现。(钛金)
dsp-input-regs-packing mult_input_regs_packing	DSP输入寄存器打包 乘法器输入寄存器打包	0	禁止优化
		1	打开优化（默认） 允许综合器把同DSP输入直接有关的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

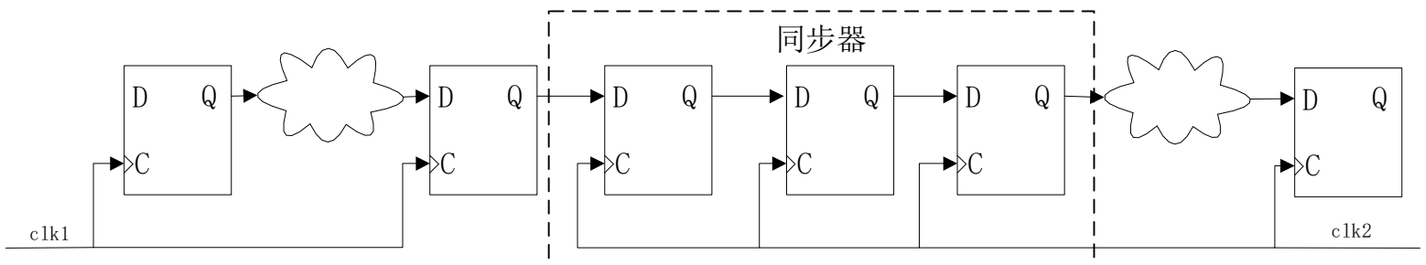
retiming



- 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

- 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综合属性注释



- 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。

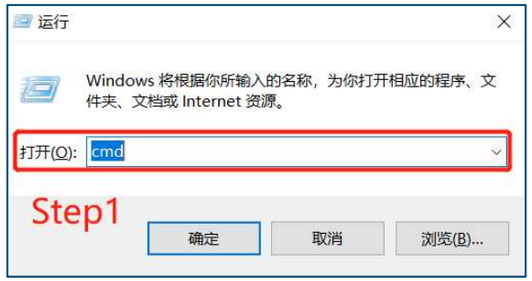
P&R优化策略	选项	功能描述
optimization_level	NULL	禁止优化（默认）
	TIMING_1	非拥塞设计布局布线优化等级1，低
	TIMING_2	非拥塞设计布局布线优化等级2，中
	TIMING_3	非拥塞设计布局布线优化等级3，高
	CONGESTION_1	拥塞设计布局布线优化等级1，低
	CONGESTION_2	拥塞设计布局布线优化等级2，中
	CONGESTION_3	拥塞设计布局布线优化等级3，高
seed	任意整数	种子，随机整数。相当于为布局布线算法设置一个随机的起始位置，不同的起始位置会影响布局布线的结果。

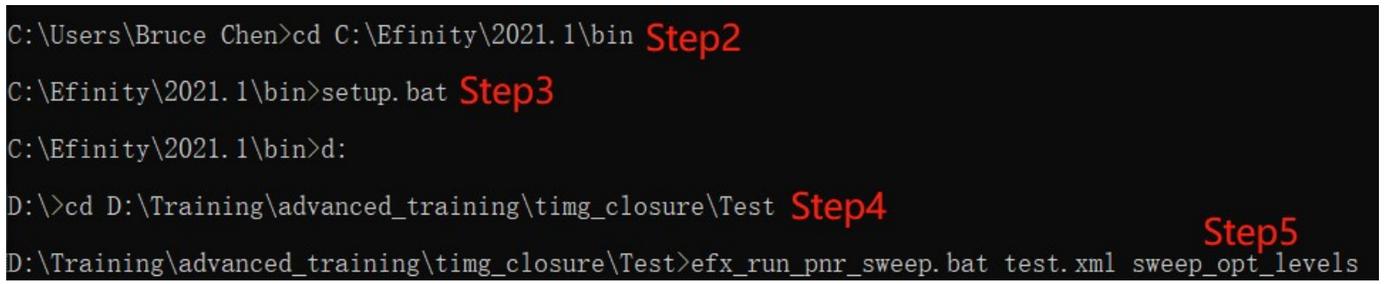
- 随机种子扫描(Seed Sweep)
  - 5个不同的种子产生的时序差异可达10%；
  - 当设计f_{MAX}离设计要求还差10%以内，或者是设计稍作修改f_{MAX}下降10%以内的时候，可以选择随机种子扫描；
  - 使用随机种子扫描，软件自动使用定义范围的种子进行布局布线，并记录包括时序报告在内的所有布局布线结果；
  - 根据结果选择最适合当前设计的种子，而不需要手动逐个尝试。

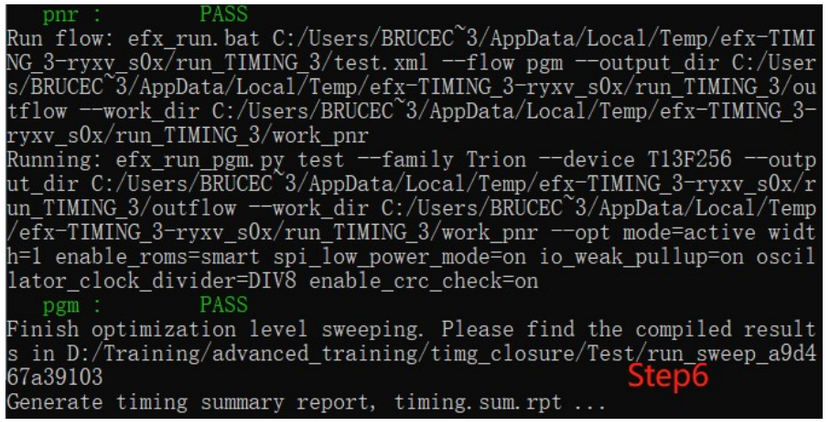
优化策略扫描(Optimization Sweep)

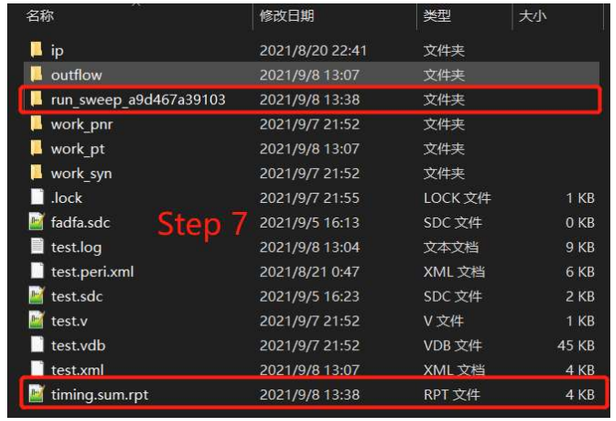
- **Step1** Win+R，运行cmd，进入命令行菜单
- **Step2** cd进入Efinity安装目录bin文件夹
- **Step3** 运行setup.bat
- **Step4** 在cmd命令行进入项目根目录
  - 如果项目和Efinity不在一个盘下，需要先进行换盘符操作，例如在D盘，则在cmd命令提示符键入D:
  - cd进入项目根目录
- **Step5** 在项目根目录下运行策略扫描命令行
  - efx_run_pnr_sweep.bat <项目名>.xml sweep_opt_levels
- **Step6** 等待后台完成优化策略扫描
  - cmd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>

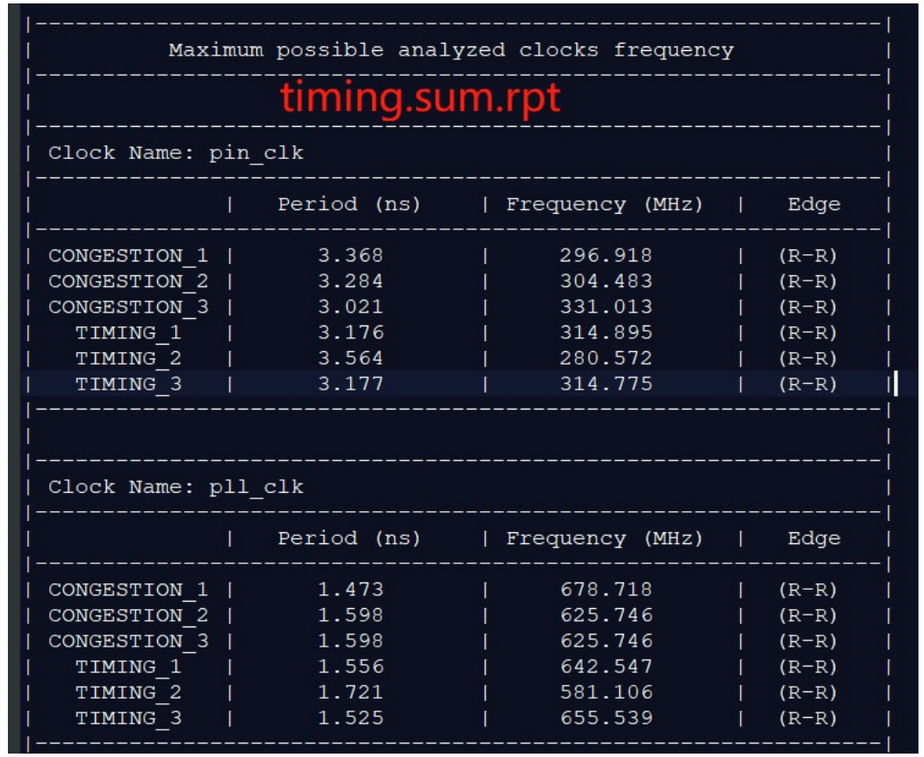
优化策略扫描(Optimization Sweep)











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

易 灵 思

谢 谢 ！