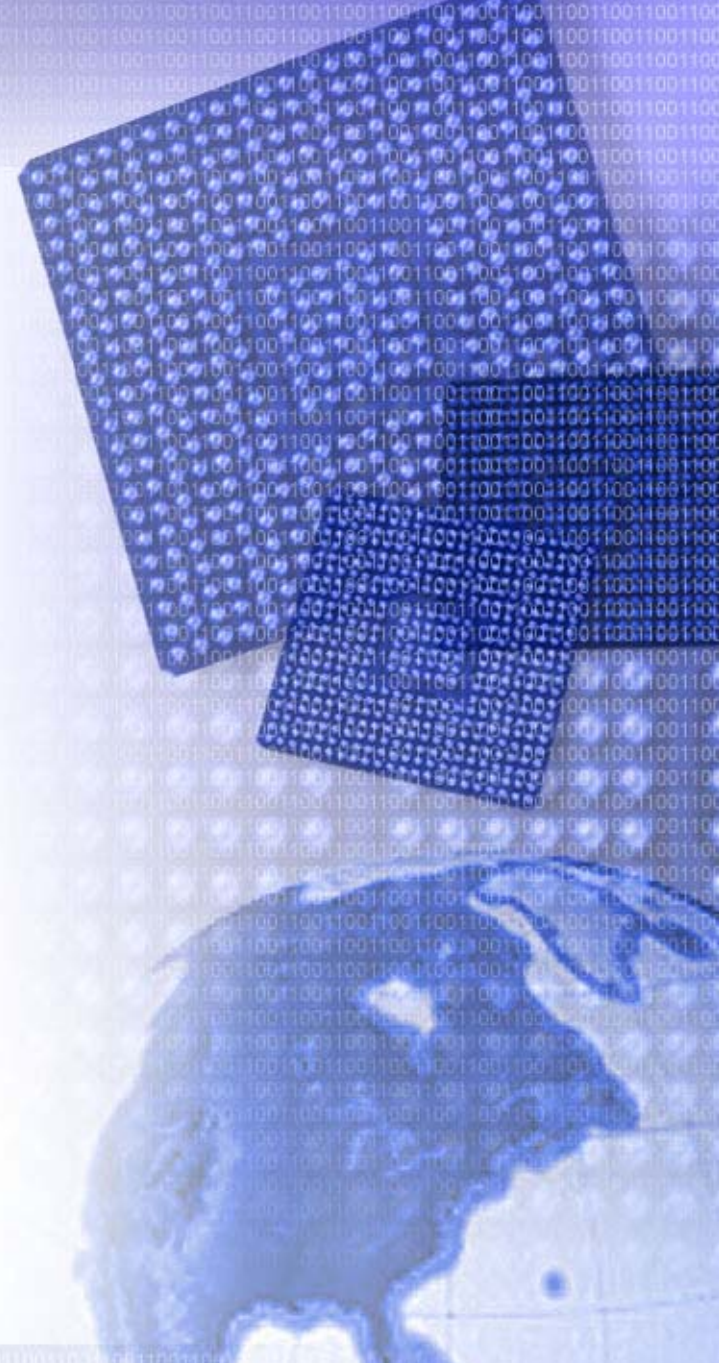




# Advanced Timing Constrain

Xilinx GSD Asia Pacific

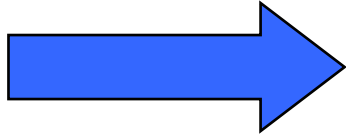


# Agenda



- 约束（Constraints）的一般概念
  - 有哪些约束？
- 对设计进行时序约束
  - 怎么用这些时序约束？

# Overview

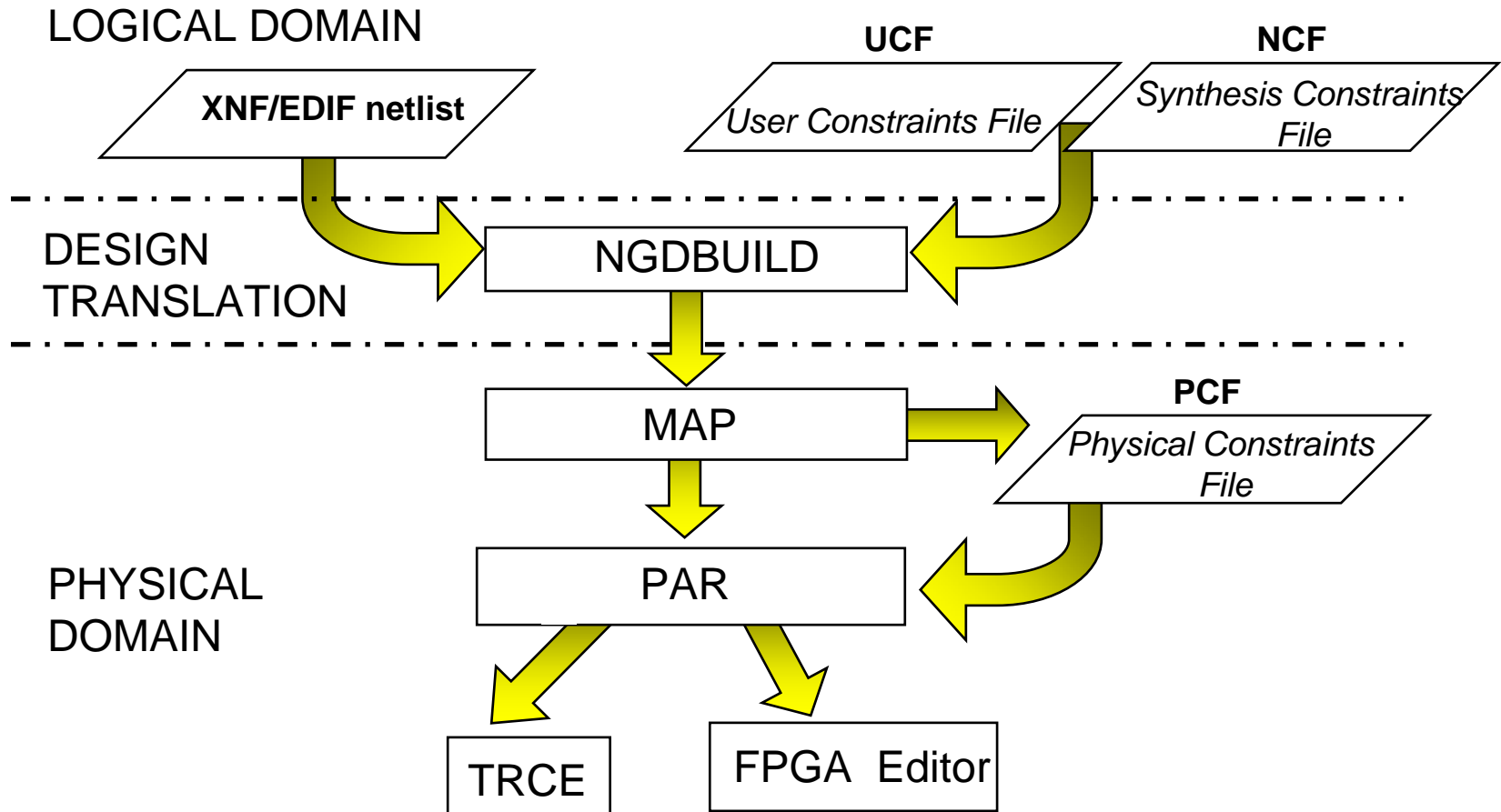


- 介绍
- 基本时序约束
- 建立分组
- 其他约束
- 约束优先级

# The UCF File

- UCF = 用户约束文件 ( User Constraints File )
- 可以用文本编辑器和Xilinx Constraints Editor ( GUI ) 进行编辑的一个简单的文本文件
  - 约束编辑器不支持所有的约束
- 除了Xilinx定义的专门用于约束的关键字：象 PERIOD, HIGH, LOW, ns, ps, 等等外，其余字符是大小写敏感的
- 每条约束以分号";" 结尾
- 以" #"号开头表明接下来的是注释
- 对于约束描述的次序没有特殊要求

# Review of Constraint Flow

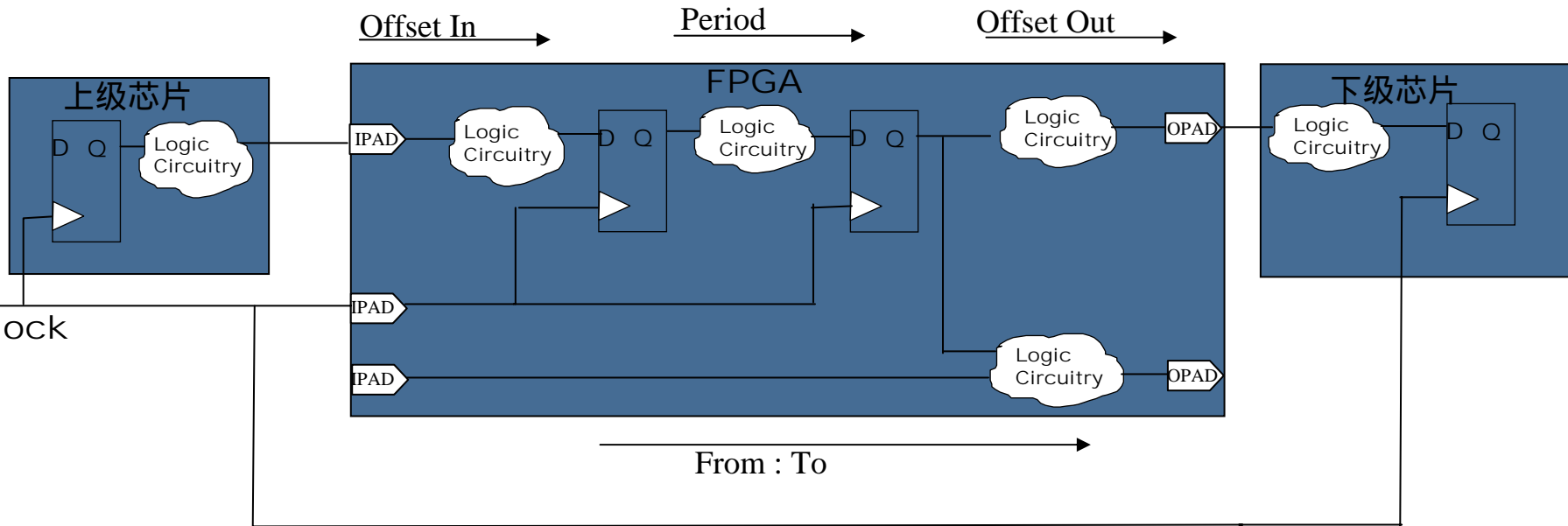


# Overview



- 介绍
- 时序约束
- 建立分组
- 其他约束
- 约束优先级

# Timing Constraint



# Overview

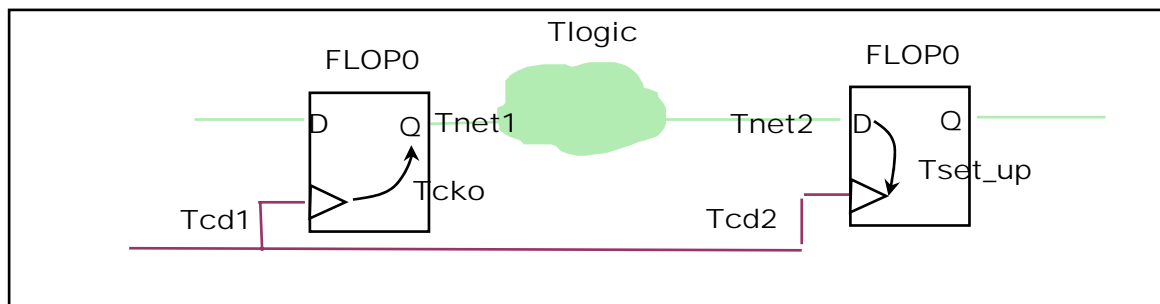


- 时钟约束
  - PERIOD
- 偏移约束
  - 输入偏移约束：OFFSET IN
  - 输出偏移约束：OFFSET OUT
- FROM:TO 约束
  - PAD 到 PAD
  - 特定路径时序约束



# Period Estimation

- 在进行Period约束之前，需要对电路的时钟周期进行估计，不要过松或过紧的约束
- 设计的内部电路所能达到的最高运行频率取决于同步元件本身的建立保持时间，以及同步元件之间的逻辑和布线延迟
  - $T_{clk} = T_{cko} + T_{net1} + T_{logic} + T_{net2} - T_{clk\_skew}$
  - $T_{clk\_skew} = T_{cd2} - T_{cd1}$

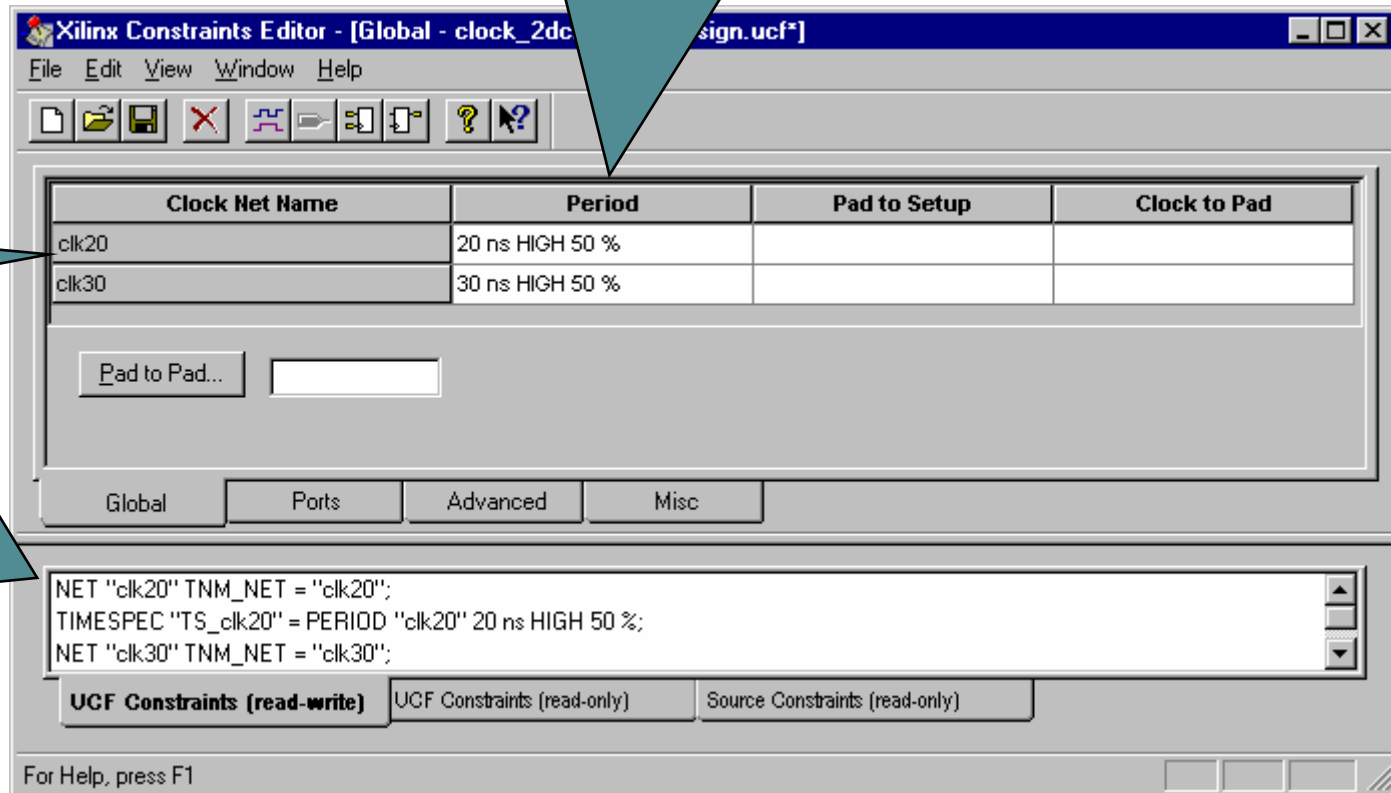


# Entering Period Constraint

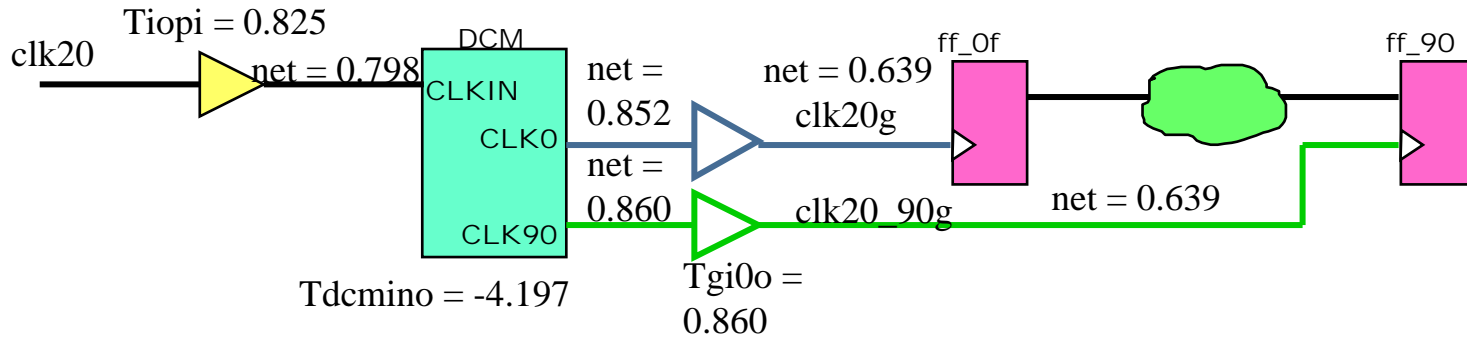
Enter period constraint  
Double click to see more  
options

Clocks in the  
design

Create the  
group with  
TNM\_NET,  
then constrain  
the group with  
a PERIOD  
constraint



# TRCE clock skew



## 目的时钟延时 - 源时钟延时

$$\begin{aligned}
 & (0.825 + 0.798 + -4.197 + 0.860 + 0.860 + 0.639) - \\
 & (0.825 + 0.798 + -4.197 + 0.852 + 0.860 + 0.639) = \\
 & .008 \text{ ns}
 \end{aligned}$$

# Basic Period Report

Basic element  
type is listed

Slack equation

Slack: 18.108 (requirement - (data path - clock skew))  
 Source: [ff 0](#) (FF)  
 Destination: [ff 0](#) (FF)  
 Requirement: 20.000ns  
 Data Path Delay: 1.892ns (Levels of Logic = 1)  
 Clock Skew: 0.000ns  
 Source Clock: clk20g rising at 0.000ns  
 Destination Clock: clk20g rising at 20.000ns

Logic Levels  
Only levels of  
logic, not Clock to  
Out and Setup

Clock  
names and  
time of  
active edge.  
Includes  
Clock Phase

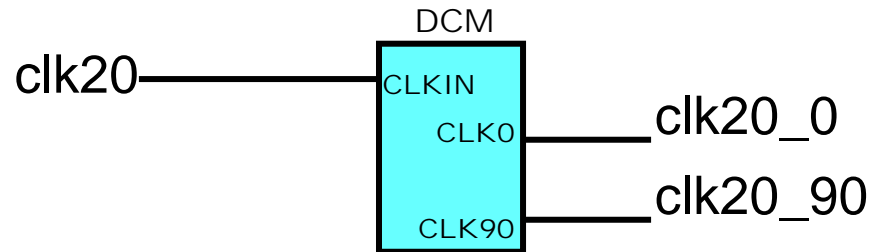
## Data Path: [ff 0](#) to [ff 0](#)

Delay type	Delay(ns)	Logical Resource(s)
<a href="#">Tcko</a>	0.568	<a href="#">ff 0</a>
net (fanout=7)	0.514	<a href="#">ff 0</a>
<a href="#">Tilo</a>	0.439	<a href="#">N 147 i</a>
net (fanout=1)	0.001	<a href="#">N 147 i</a>
<a href="#">Tdxck</a>	0.370	<a href="#">ff 0</a>
Total	1.892ns	(1.377ns logic, 0.515ns route) (72.8% logic, 27.2% route)

Data path  
with Cross  
Probing  
Links to  
Floorplanner  
or Synthesis  
Tool  
(In Timing  
Analyzer)

Web link to  
graphical  
picture of  
delay type!  
(In Timing  
Analyzer)

# Period Constraint with DCM



- 用户对DCM的时钟输入端进行约束

```
NET "clk20" TNM_NET = "clk20";
```

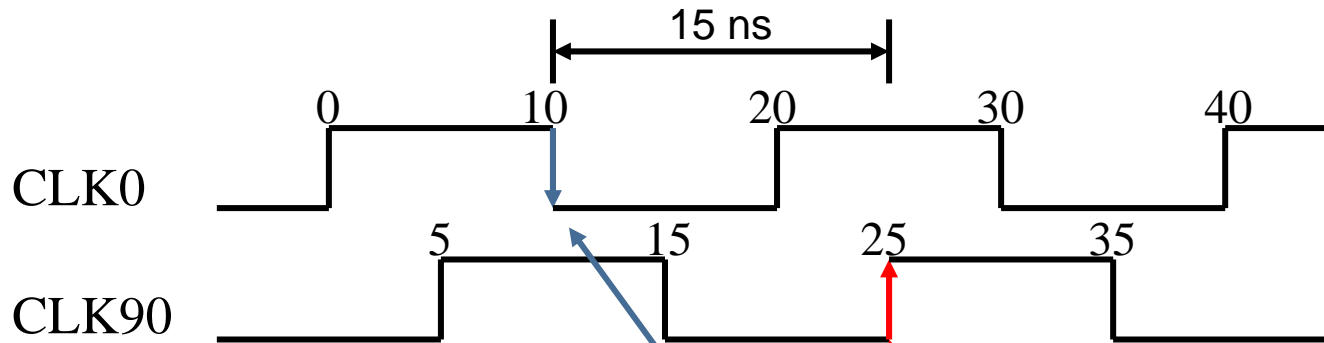
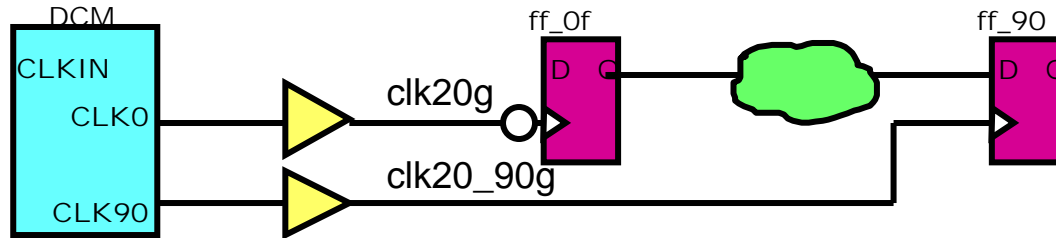
```
TIMESPEC "TS_clk20" = PERIOD "clk20" 20 ns HIGH 50 %;
```

- Translate 时产生DCM的时钟约束

```
CLK0: TS_clk20_0=PERIOD clk20_0 TS_clk20*1.000000 HIGH 50.000000%
```

```
CLK90: TS_clk20_90=PERIOD clk20_90 TS_clk20*1.000000 PHASE +  
5.000000 nS HIGH 50.000000%
```

# Clock Phase Period Example



```
Slack: 11.578ns (requirement - (data path - clock skew))
Source: ff_Of (FF)
Destination: ff_90 (FF)
Requirement: 15.000ns
Data Path Delay: 3.422ns (Levels of Logic = 3)
Clock Skew: 0.000ns
Source Clock: clk20g falling at 10.000ns
Destination Clock: clk20_90g rising at 25.000ns
```

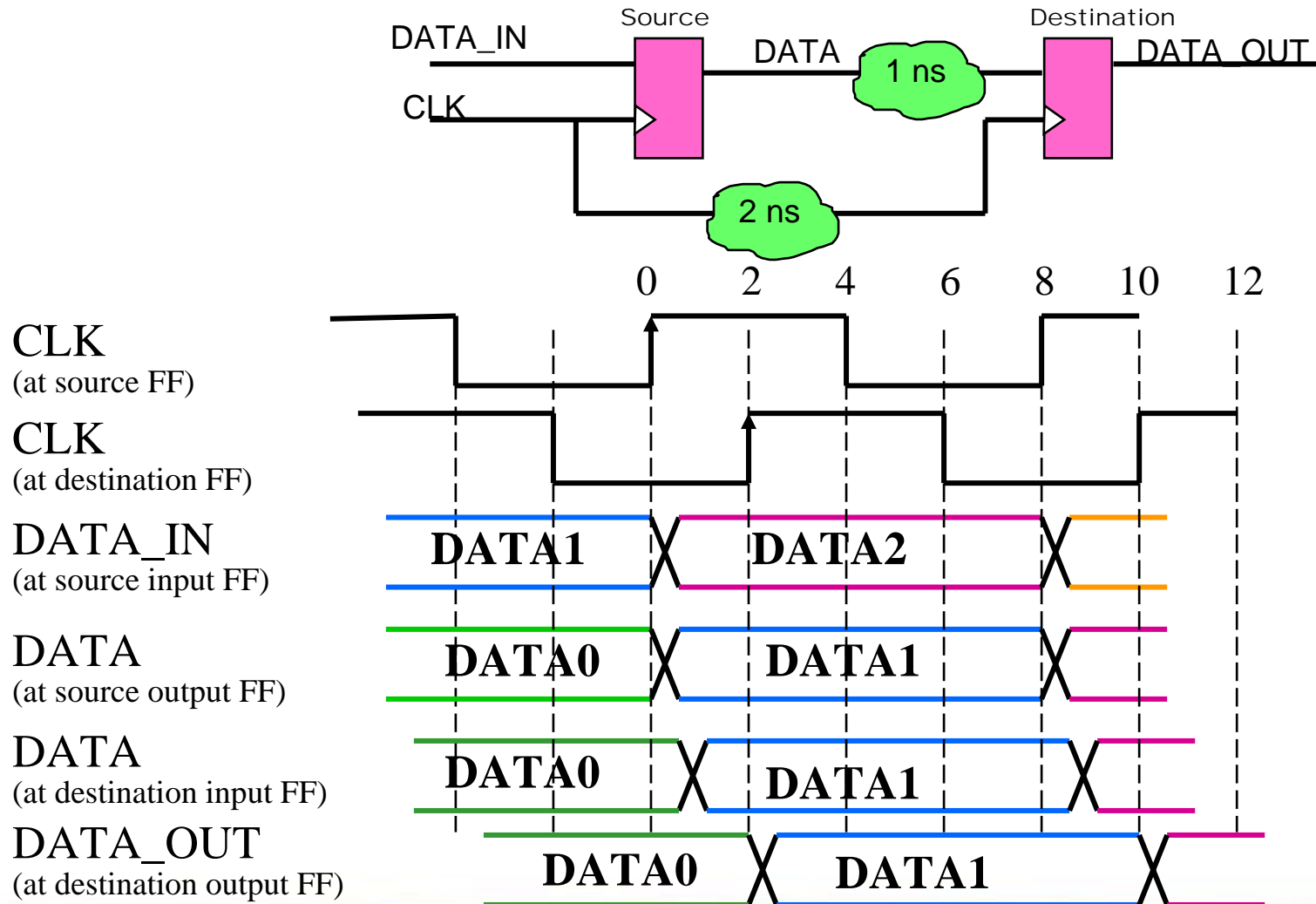
不同时钟沿之间的要求

有效沿的类型和时间

# Hold Calculations

- 当数据在时钟到达目的时序元件以前发生改变，此时就发生了内部的 hold violation
  - 数据延时比正的Clock Skew 要小

# Hold Calculations





# Constraints Overview

- 时钟约束

- PERIOD



- 偏移约束

- 输入偏移约束：OFFSET IN

- 输出偏移约束：OFFSET OUT

- FROM:TO 约束

- PAD 到 PAD

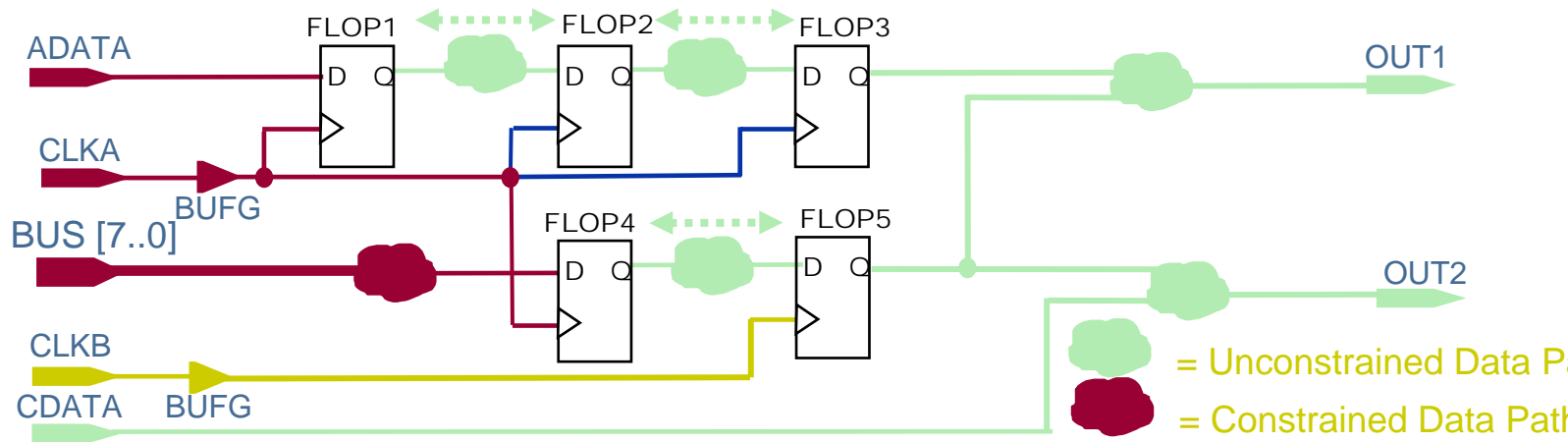
- 特定路径时序约束

# OFFSET Constraint

- 偏移约束用于约束FPGA外部时钟的有效时钟沿和输入输出引脚之间的时序关系（只用于与PAD相连的信号，不能用于内部信号）
- 时钟的有效时钟沿在利用PERIOD 约束描述中的HIGH/LOW 关键词来定义时钟的初始有效时钟沿的类型
  - 有效时钟沿为上升沿 (缺省)
    - TIMESPEC TS\_clock = PERIOD clock 10 ns HIGH 50 %;
  - 有效时钟沿为下降沿
    - TIMESPEC TS\_clock = PERIOD clock 10 ns LOW 50 %;
- OFFSET:
  - OFFSET = IN <delay> ns BEFORE <clk\_group>;
  - OFFSET = OUT <delay> ns AFTER <clk\_group>;

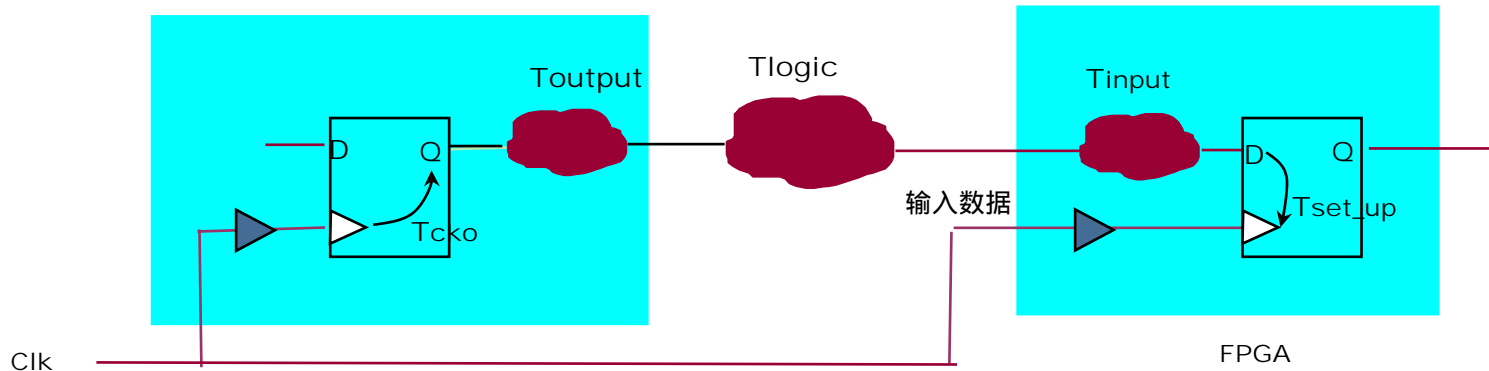
# OFFSET IN Constraint

- 输入偏移约束对输入PAD到同步元件输入之间的路径延时进行约束.
- 输入偏移约束不能对芯片内部同步元件之间的路径进行约束



# OFFSET IN AFTER Constraint

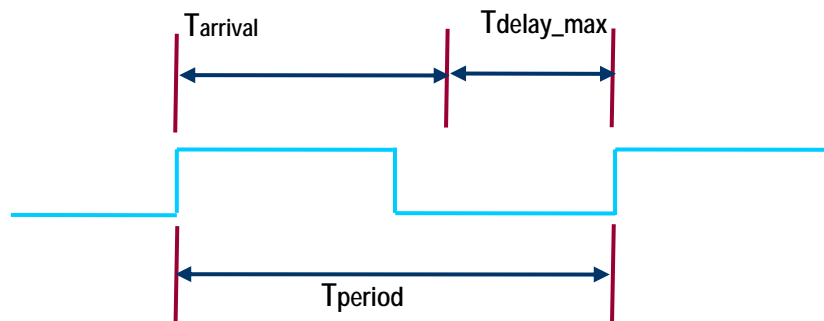
- 定义输入数据在有效时钟沿之后多长时间到达芯片的输入引脚
  - $T_{arrival} = T_{cko} + T_{output} + T_{logic}$
  - NET Data\_In OFFSET=IN Tarrival AFTER CLK



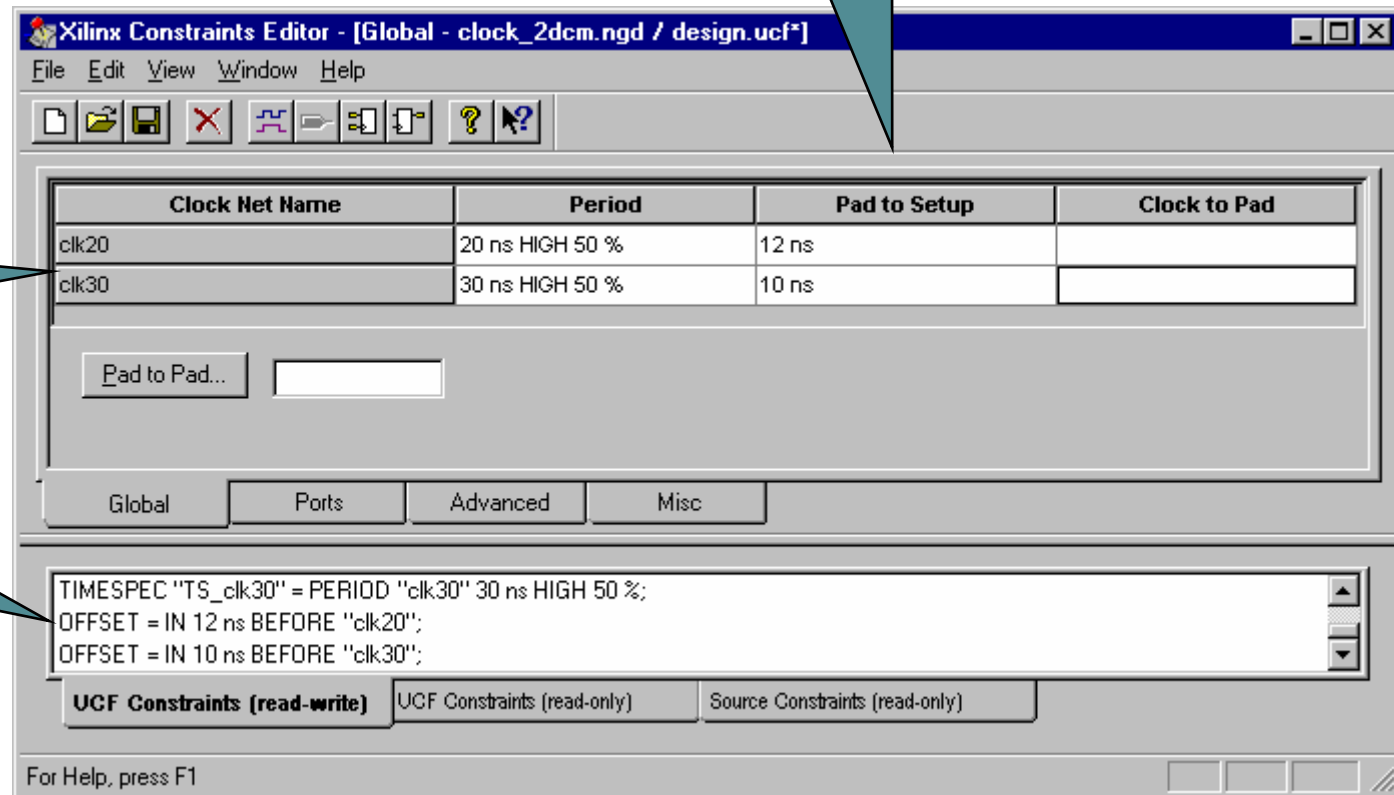
- 综合实现工具会对管脚输入到同步元件的数据端之间的路径进行优化，以使输入延时时间  $T_{input}$ ：
$$T_{arrival} + T_{input} + T_{setup} < T_{clk}$$

# OFFSET IN BEFORE Constraint

- OFFSET\_IN\_BEFORE 和OFFSET\_IN\_AFTER一样都是对芯片输入到内部时序元件的输入端之间的逻辑进行约束
- 定义输入数据应该比有效时钟沿提前多长时间内准备好
  - NET Data\_In OFFSET=IN Tdelay BEFORE CLK
  - Tdelay : 为要求的芯片内部输入延迟
  - 综合实现工具会努力保证 $T_{delay} < T_{period} - T_{arrival}$



# Entering Global OFFSET IN Constraint



# Entering Specific OFFSET IN Constraint

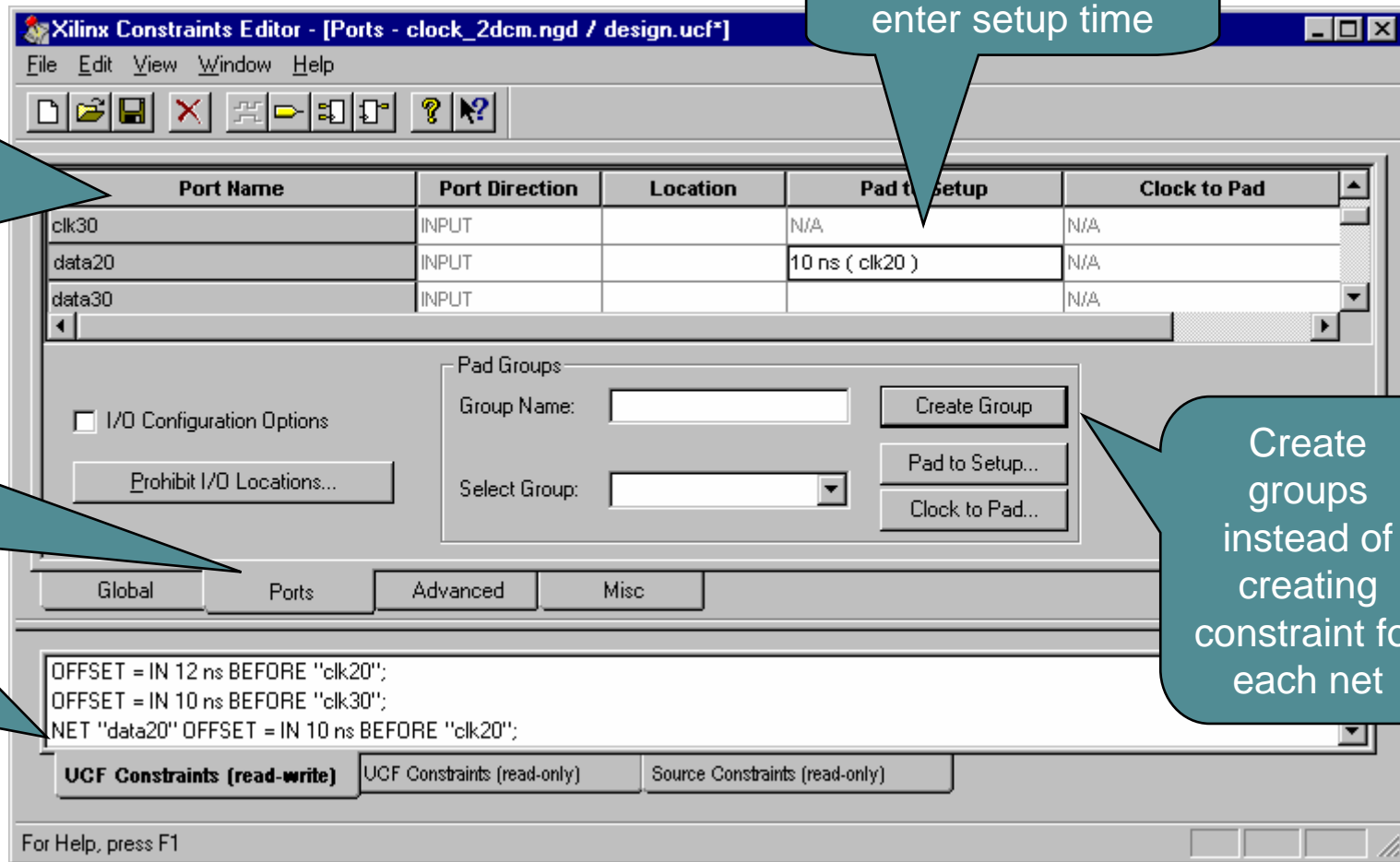
Ports in the design, Double click to change order of Ports

Ports Tab to define port specific timing

Generated UCF constraints

Double click to enter setup time

Create groups instead of creating constraint for each net



# OFFSET IN Report Example

Slack: 4.534ns (requirement - (data path - clock path - clock arrival))  
Source: [sig 0](#) (PAD)  
Destination: [ff 0](#) (FF)  
Destination Clock: clk20g rising at 0.000ns  
Requirement: 8.000ns  
Data Path Delay: 2.985ns (Levels of Logic = 2)  
Clock Path Delay: -0.481ns (Levels of Logic = 3)

Slack equation

## Data Path: [sig 0](#) to [ff 0](#)

Delay type	Delay(ns)	Logical Resource(s)
<hr/>		
<a href="#">Tiopi</a>	0.825	<a href="#">sig 0</a> <a href="#">sig 0 ibuf</a>
net (fanout=2)	1.350	<a href="#">sig 0 c</a>
<a href="#">Tilo</a>	0.439	<a href="#">N 147 i</a>
net (fanout=1)	0.001	<a href="#">N 147 i</a>
<a href="#">Tdxck</a>	0.370	<a href="#">ff 0</a>
<hr/>		
Total	2.985ns	(1.634ns logic, 1.351ns route) (54.7% logic, 45.3% route)

Data Path Delay

## Clock Path: [clk20](#) to [ff 0](#)

Delay type	Delay(ns)	Logical Resource(s)
<hr/>		
<a href="#">Tiopi</a>	0.825	<a href="#">clk20</a> <a href="#">IBUFG CLK20</a>
net (fanout=1)	0.798	<a href="#">clk20 ibufg</a>
<a href="#">Tdcmino</a>	-4.197	<a href="#">clk20 DLL</a>
net (fanout=2)	0.860	<a href="#">clk20 0</a>
<a href="#">Tgi0o</a>	0.589	<a href="#">clk20 BUFG.GCLKMUX</a> <a href="#">clk20 BUFG</a>
net (fanout=4)	0.644	<a href="#">clk20g</a>
<hr/>		
Total	-0.481ns	(-2.783ns logic, 2.302ns route)

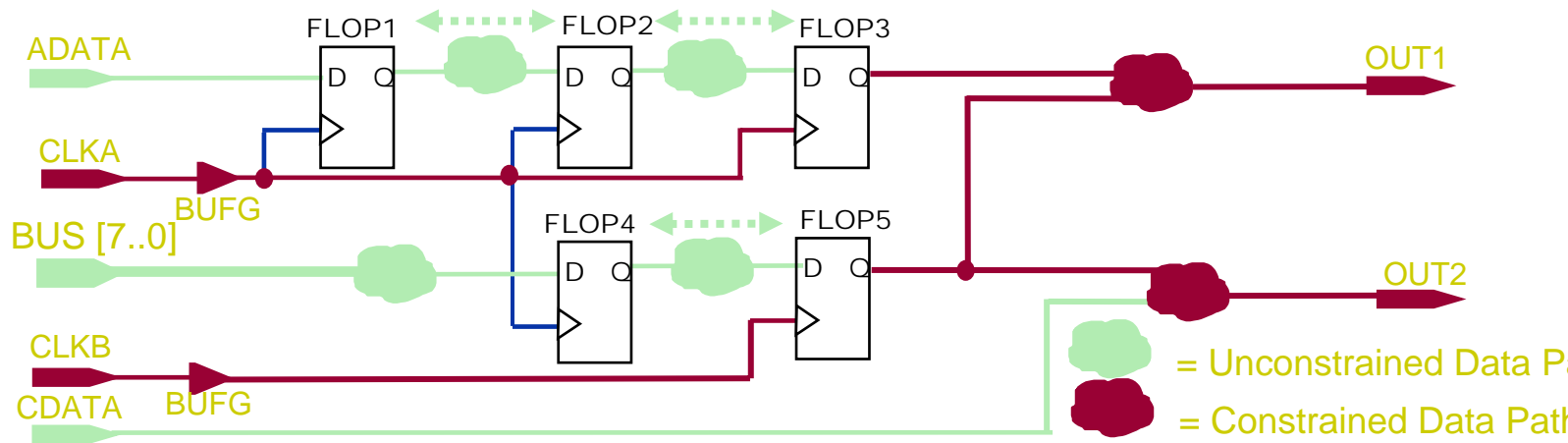
Clock Path Delay

Clock name and time of active edge



# OFFSET OUT Constraint

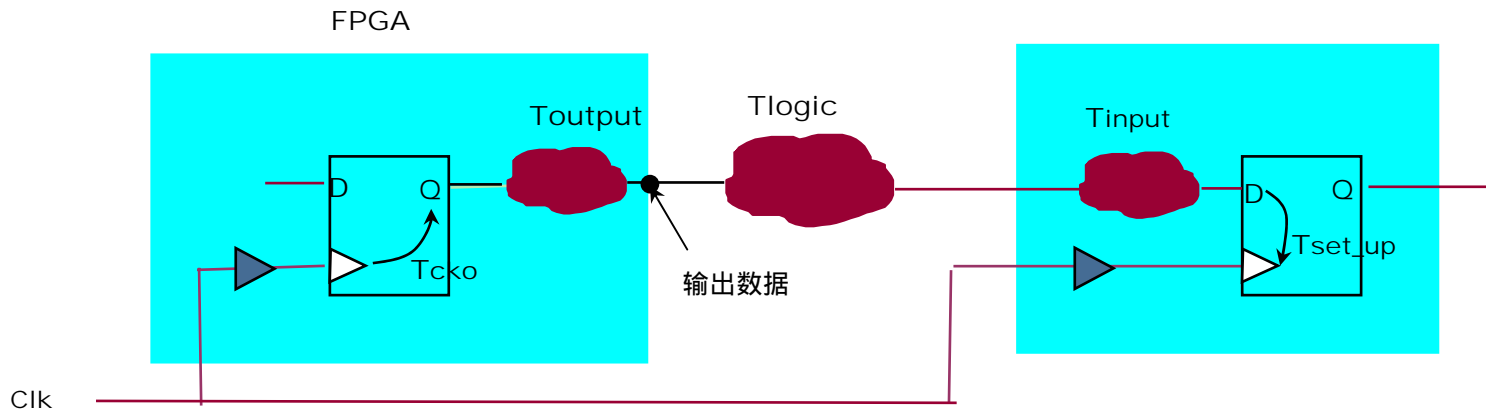
- 输出偏移约束对同步元件的输出到芯片输出PAD之间的路径进行约束
- 输出偏移约束和输入偏移约束一样同样不能对芯片内部时序元件之间的路径进行约束



# OFFSET OUT BEFORE

## Constraint

- 定义下一级芯片的输入数据应在有效时钟沿之前多长时间内准备好
  - $T_{stable} = T_{logic} + T_{input} + T_{setup}$
  - NET Data\_Out OFFSET=OUT  $T_{stable}$  BEFORE CLK

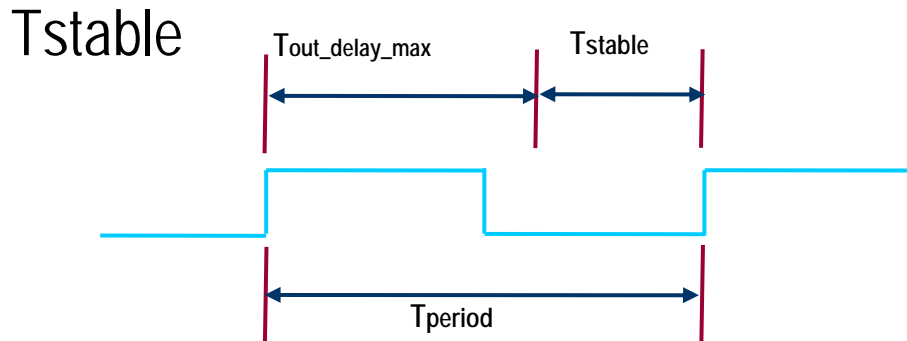


- 综合实现工具对设计同步元件的输出到芯片输出管脚之间的路径进行优化，使输出端的延迟满足如下关系：

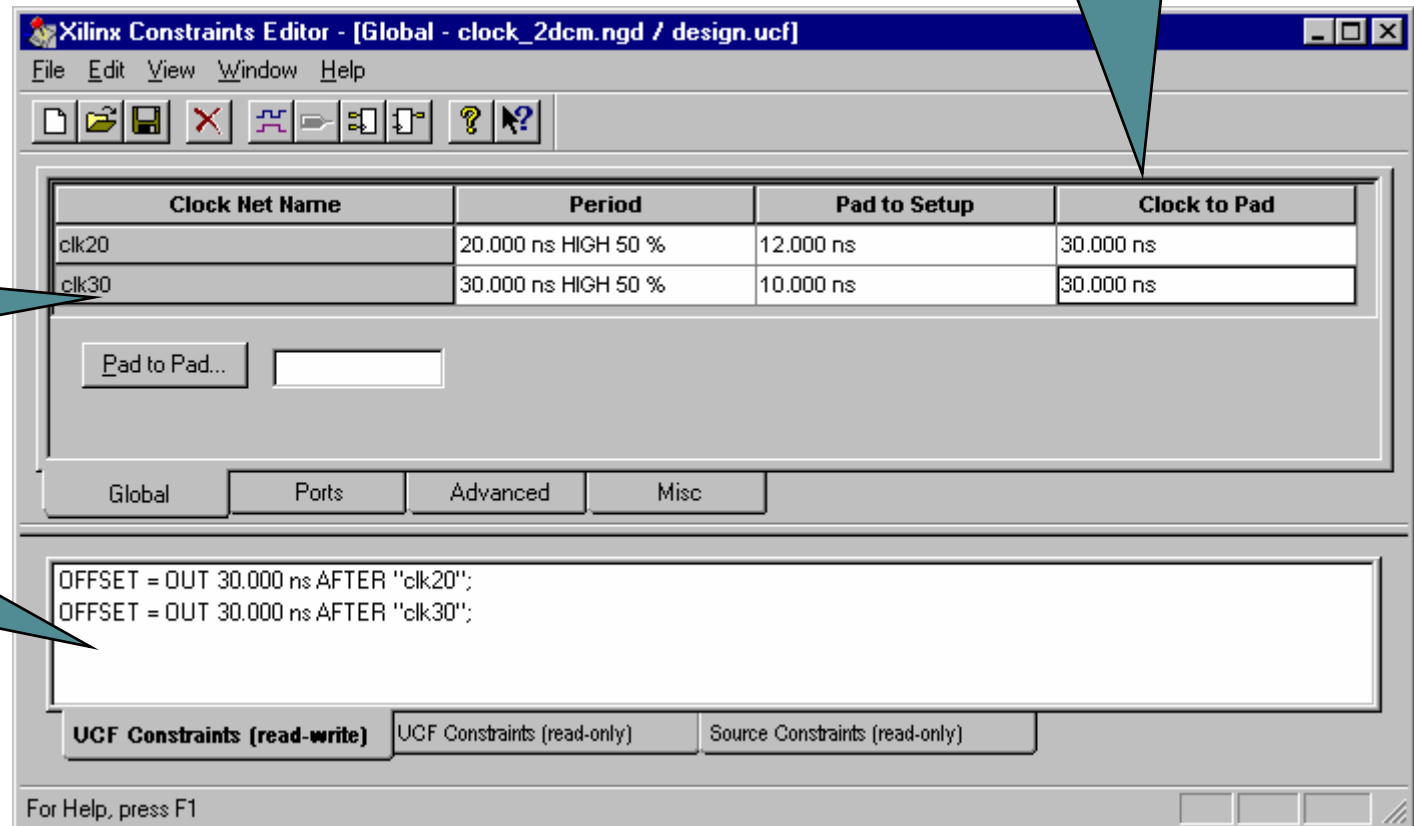
$$T_{cko} + T_{output} + T_{stable} < T_{clk}$$

# OFFSET OUT AFTER Constraint

- OFFSET\_OUT\_BEFORE 和OFFSET\_OUT\_AFTER一样都是对芯片输出到内部时序元件的输出端之间的逻辑实现进行约束
- 定义输出数据应该在有效时钟沿后多长时间稳定下来
  - NET Data\_Out OFFSET=OUT Toutput\_delay BEFORE CLK
  - Toutput\_delay : 为要求的芯片内部输出延迟
  - 综合实现工具会努力保证 $T_{\text{output\_delay}} < T_{\text{period}} - T_{\text{stable}}$



# Entering Global OFFSET OUT Constraint



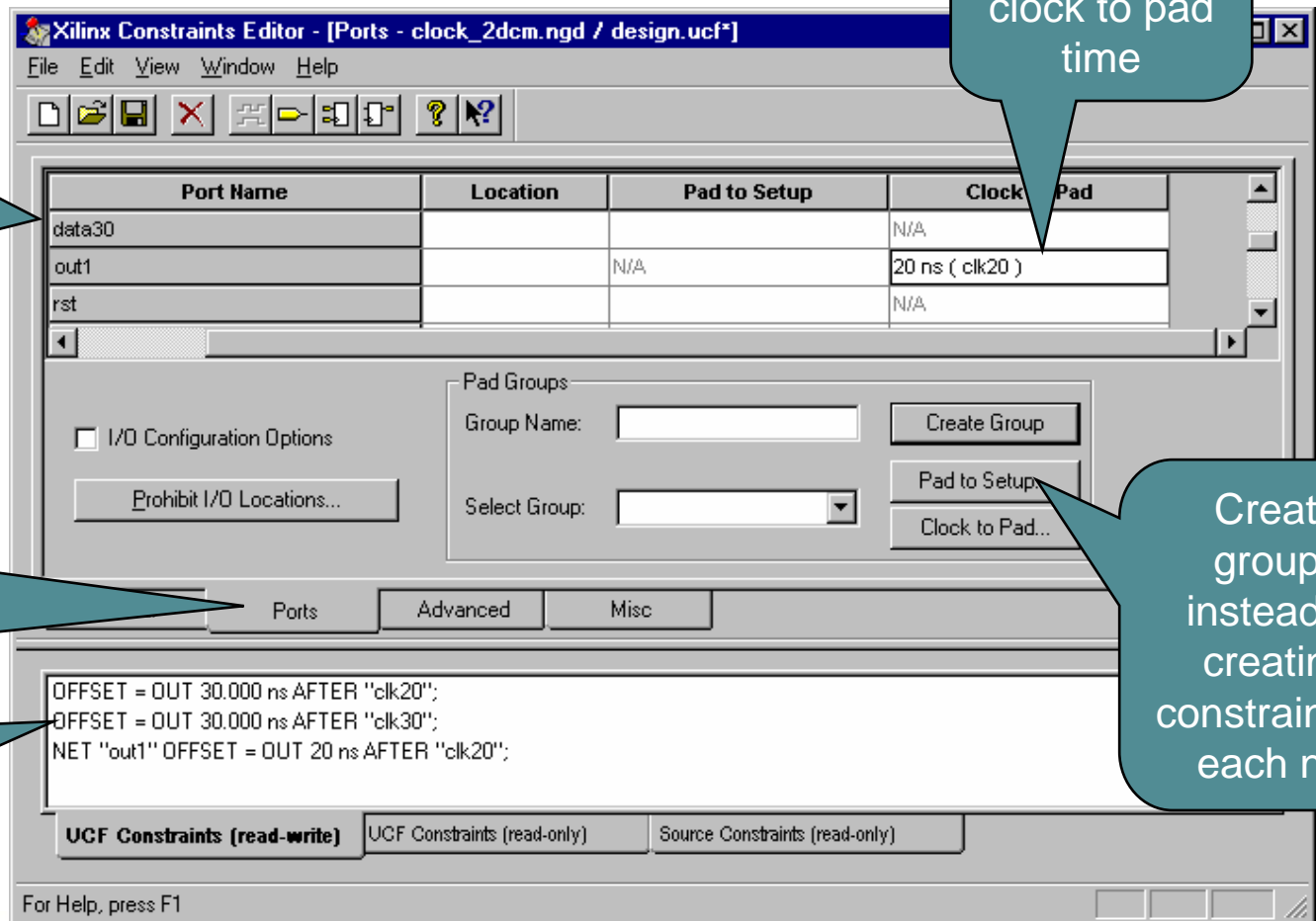
# Entering Specific OFFSET OUT Constraint

Ports in the design, Double click to change order of Ports

Double click to enter clock to pad time

Ports Tab to define port specific timing

Generated UCF constraints



Create groups instead of creating constraint for each net

# OFFSET OUT Report

Slack: 21.714ns (requirement - (clock arrival + clock path + data path))  
 Source: [clk20](#) (PAD)  
 Destination: [out1](#) (PAD)  
 Source Clock: clk20g rising at 0.000ns  
 Requirement: 30.000ns  
 Data Path Delay: 8.770ns (Levels of Logic = 2)  
 Clock Path Delay: -0.484ns (Levels of Logic = 3)

Slack equation

Clock name and time of active edge

## Clock Path: [clk20](#) to [ff 0](#)

Delay type	Delay(ns)	Logical Resource(s)
<a href="#">Tiopi</a>	0.825	<a href="#">clk20</a> <a href="#">IBUFG CLK20</a>
net (fanout=1)	0.798	<a href="#">clk20 ibufg</a>
<a href="#">Tdcmino</a>	-4.197	<a href="#">clk20 DLL</a>
net (fanout=2)	0.860	<a href="#">clk20 0</a>
<a href="#">TgiOo</a>	0.589	<a href="#">clk20 BUFG.GCLKMUX</a> <a href="#">clk20 BUFG</a>
net (fanout=4)	0.641	<a href="#">clk20g</a>
Total	-0.484ns	(-2.783ns logic, 2.299ns route)

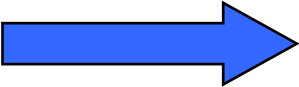
Clock Path Delay

## Data Path: [ff 0](#) to [out1](#)

Delay type	Delay(ns)	Logical Resource(s)
<a href="#">Tcko</a>	0.568	<a href="#">ff 0</a>
net (fanout=6)	0.323	<a href="#">ff 0</a>
<a href="#">Tilo</a>	0.439	<a href="#">N 66 i</a>
net (fanout=1)	1.333	<a href="#">N 66 i</a>
<a href="#">Tioop</a>	6.107	<a href="#">out1 obuf</a> <a href="#">out1</a>
Total	8.770ns	(7.114ns logic, 1.656ns route) (81.1% logic, 18.9% route)

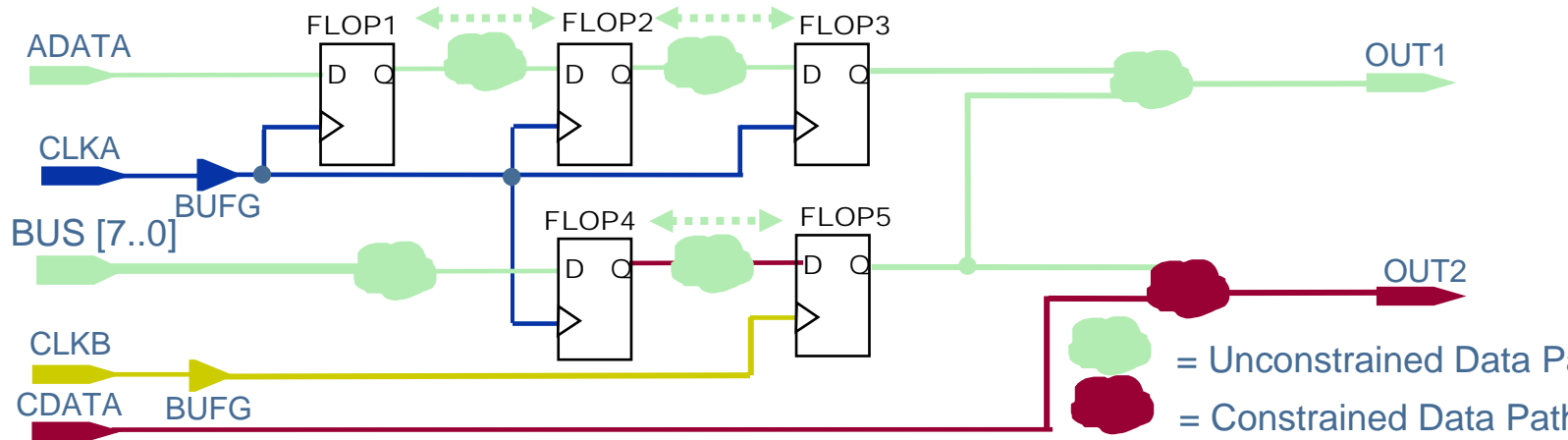
Data Path Delay

# Constraints Overview

- 时钟约束
  - PERIOD
- 偏移约束
  - 输入偏移约束：OFFSET IN
  - 输出偏移约束：OFFSET OUT
-  · FROM:TO 约束
  - PAD 到 PAD
  - 特定路径时序约束

# PAD to PAD Constraint

- 一般用户都不会对I/O Pad之间的纯组合逻辑的延时路径进行约束
- 如果对PAD 到PAD之间的路径延时有要求，可以用FROM : TO进行约束





# FROM:TO Constraint

- Syntax: **TIMESPEC** *TS\_name* = **FROM** *group1* **TO** *group2* *value*;
- *TS\_name* 必须要以 "TS" 起首
  - 后面紧跟ASCII码字符串,由A-Z,a-z,0-9以及下划线组成
- *Group1* 路径的起点
- *Group2* 路径的终点
- *Value* 的缺省单位为ns
  - 可以是 ps 或 MHz 等
  - 也可以是表达式

# Entering a Pad to Pad Constraint

The screenshot shows the Xilinx Constraints Editor window titled "Xilinx Constraints Editor - [Global - clock\_2dcm.ngd / design.ucf\*]". The window has a menu bar (File, Edit, View, Window, Help) and a toolbar with icons for file operations and help. The main area contains a table of constraints and a text area for generated UCF constraints.

Clock Net Name	Period	Pad to Setup	Clock to Pad
clk20	20.000 ns HIGH 50 %	12.000 ns	30.000 ns
clk30	30.000 ns HIGH 50 %	10.000 ns	30.000 ns

Below the table, there is a "Pad to Pad..." button and a text input field containing the value "15".

At the bottom, there are tabs for "Global", "Ports", "Advanced", and "Misc". The "Global" tab is selected, showing the following generated UCF constraints:

```
OFFSET = OUT 30.000 ns AFTER "clk20";  
OFFSET = OUT 30.000 ns AFTER "clk30";  
NET "out1" OFFSET = OUT 20 ns AFTER "clk20";  
TIMESPEC "TS_P2P" = FROM "PADS" TO "PADS" 15 ns;
```

Below the text area, there are three buttons: "UCF Constraints (read-write)", "UCF Constraints (read-only)", and "Source Constraints (read-only)". The "UCF Constraints (read-write)" button is selected.

At the bottom of the window, it says "For Help, press F1".

Pad to Pad  
Timing

Generated  
UCF  
constraints

# PAD to PAD Report

Timing constraint: TS\_P2P = MAXDELAY FROM TIMEGRP "PADS" TO TIMEGRP "PADS" 15 ns ;

1 item analyzed, 0 timing errors detected.

Maximum delay is 11.251ns.

Slack equation

Slack: 3.749ns (requirement - data path)

Source: [data](#) (PAD)

Destination: [out1](#) (PAD)

Requirement: 15.000ns

Data Path Delay: 11.251ns (Levels of Logic = 4)

源和目的元件都是PAD

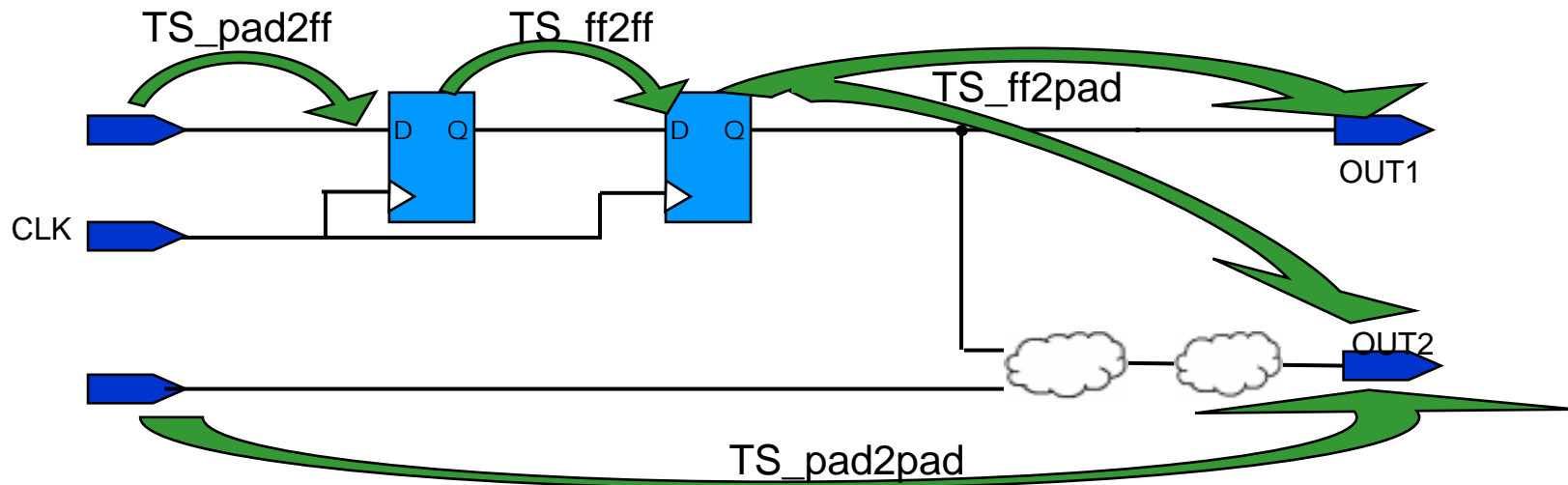
## Data Path: data to out1

Delay type	Delay(ns)	Logical Resource(s)
<a href="#">Tiopi</a>	0.825	<a href="#">data</a> <a href="#">data ibuf</a>
net (fanout=2)	0.984	<a href="#">data c</a>
<a href="#">Tilo</a>	0.439	<a href="#">out1 i and3 42</a>
net (fanout=1)	0.532	<a href="#">out1 i and3 42</a>
<a href="#">Tilo</a>	0.439	<a href="#">N 66 i</a>
net (fanout=1)	1.925	<a href="#">N 66 i</a>
<a href="#">Tioup</a>	6.107	<a href="#">out1 obuf</a> <a href="#">out1</a>
Total	11.251ns	(7.810ns logic, 3.441ns route) (69.4% logic, 30.6% route)

数据路径延时

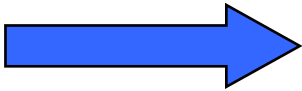
# FROM:TO Examples

- Using pre-defined groups:
  - TIMESPEC TS\_pad2pad = FROM PADS TO PADS 26 ns;
  - TIMESPEC TS\_pad2ff = FROM PADS TO FFS 6 ns;
  - TIMESPEC TS\_ff2ff = FROM FFS TO FFS 10 ns;
  - TIMESPEC TS\_ff2pad = FROM FFS TO PADS 6 ns;



# Overview

- 介绍
- 基本时序约束
- **建立分组**
- 其他约束
- 约束优先级



# Specifying Groups

- 延时路径的起点是芯片的输入和内部有效同步元件的输出，终点是芯片的输出和内部有效同步元件的输入
- 为了对路径进行高效率的约束，路径的起点和终点最好能够被分成一个个组
- 在作时序约束时可以作下列四种分组：
  - 预定义分组 (keywords)
  - 用TNM定义建立用户自定义分组
  - 对当前已经存在的分组重新进行组合
  - 通过模式匹配或网络（NET）名进行的分组

# Predefined Groups

- 时序约束中利用下列关键字定义时序组和端点:
  - PADS: 所有的输入输出PAD
  - FFS: 所有的触发器( flip-flop)
  - LATCHES: 所有的寄存器 ( latche )
  - RAMS: 所有的RAM -包括分布式(distributed) 和块状(Block)RAM
  - BRAMS\_PORTA: 所有双口块状RAM的端口A
  - BRAMS\_PORTB: 所有双口块状RAM的端口B
  - MULTS: 被寄存的乘法器
  - CPUS: Virtex-II Pro CPU
  - HSIOs: Virtex-II Pro Gigabit Transceiver

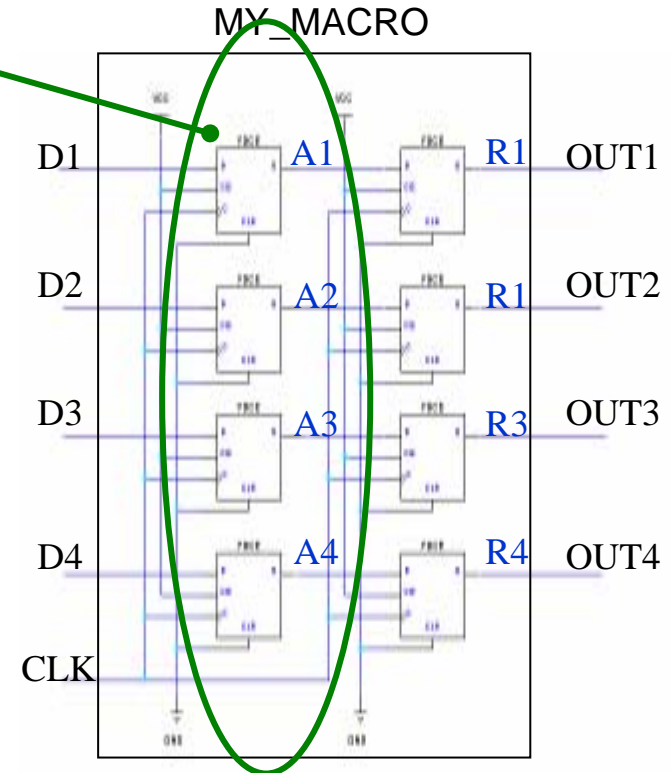
# Qualifiers and Predefined Groups

- 在一个预定义分组中限定一部分元件
- Syntax: *predefined\_group* (name\_qualifier)
  - *predefined\_group* : 预定义分组的一个关键字 (FFS, LATCHES, PADS, or RAMS)
  - name\_qualifier ( Primitive开始的完全层次路径名 )
    - 用星号"\*"表示任意多个字符
    - 用问号"?"表示一个字符



# Qualifier Example

- 分组 FFS(MY\_MACRO/A?) 是 MY\_MACRO 中输出为 A1, A2, A3, and A4 的4个触发器



# Signal/Element Names

- 问题: 怎样才能知道Instance名和net 名?
  - 查看网表 ( 比较麻烦 )
  - 通过约束编辑器来查看
- 建议:
  - 用约束编辑器作初始的约束和分组
  - 编辑UCF文件, 利用通配符来减少约束和增加约束

# User Defined Groups

- 用TNM约束可以在元件中选出构成一个分组的元件
  - 元件：FFS,RAMS,LATCHES ,  
PADS,BRAMS\_PORTA,BRAM\_PORTB,CPUs,MGTs
  - 可以通过Net, Primitive,Macro 以及Macro/Primitive的pin来创建分组

{NET | INST | PIN} "Object\_name" TNM = "identifier"

Object\_name : NET,INST或PIN的名称

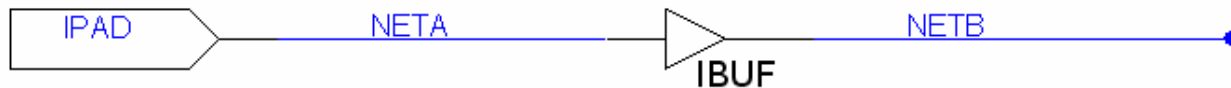
identifer : 分组的名称

# Why Use TNM?

- 第一种方法（隐式）：
  - `TIMESPEC TSmacro2data = FROM FFS(macro?) TO FFS(data*) 9 ns;`
  - `TIMESPEC TSdata2pads = FROM FFS(data*) TO PADS(pad?) 16 ns;`
- 第二种方法（显式）：
  - `INST macro? TNM = macro_ffs;`
  - `INST data* TNM = data_ffs;`
  - `PAD pad? TNM = padgrp;`
  - `TIMESPEC TSmacro2data = FROM macro_ffs TO data_ffs 9 ns;`
  - `TIMESPEC TSdata2pads = FROM data_ffs TO padgrp 16 ns;`
- 如果现在需要增加对从macro ? 到bus\_a\* 到 data\_ffs的约束
  - 第一种方法: 需要添加两行约束描述
  - 第二种方法：只需要利用TNM作一个分组定义

# TNM versus TNM\_NET

- 当TNM约束加在Net上时，这个Net所驱动的所有有效同步元件都会被赋予这个TNM值
- TNM如果加在输入PAD上时不会穿越IBUF和IBUFG，而TNM\_NET可以

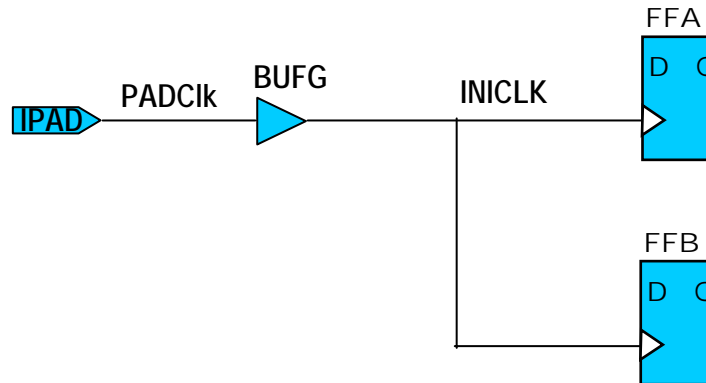


- 例子：将NETA驱动的时序元件分为一个组：
  - NET NETB TNM = myowngroup; or
  - NET NETA TNM\_NET = myowngroup;

# TNM\_NET

- TNM\_NET

- 只加在Net上，把该Net所在路径的所有有效同步元件作为命名分组的一部分
- 与TNM约束加在Net上时的情况基本相同，区别在于，如果TNM被加在PAD上时，TNM的值只能被赋予PAD，而不是该网线所在路径上的所有有效同步元件。



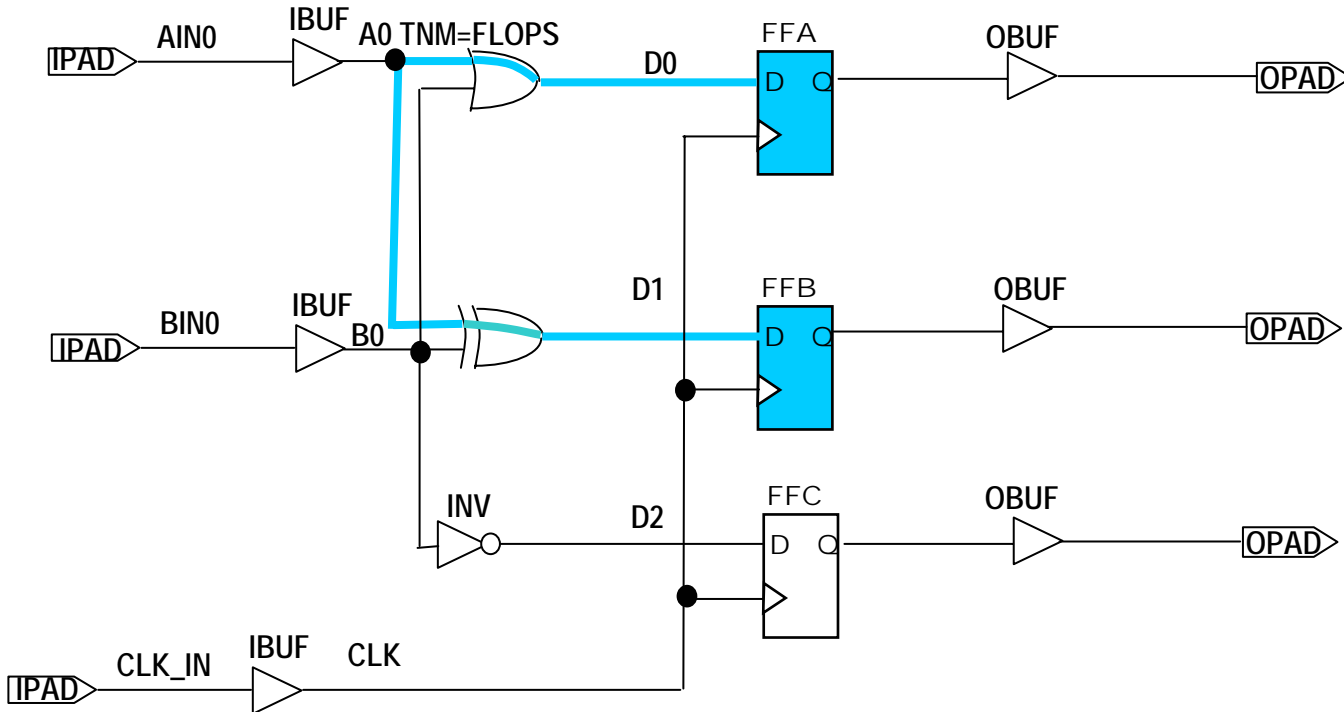
- NET "PADCLK" TNM = "PADGRP"

# TNM on Macros

- 当TNM 约束加在Macro/Primitive 的引脚上时，则该引脚驱动的所有有效同步元件将作为TNM定义的分组成员
- 时当TNM约束加在Macro/Primitive上时，那么这个Macro或Primitive将作为TNM定义的分组
  - Macro或Primitive各个层次上的同步元件都归为TNM定义的分组

# TNM on Net

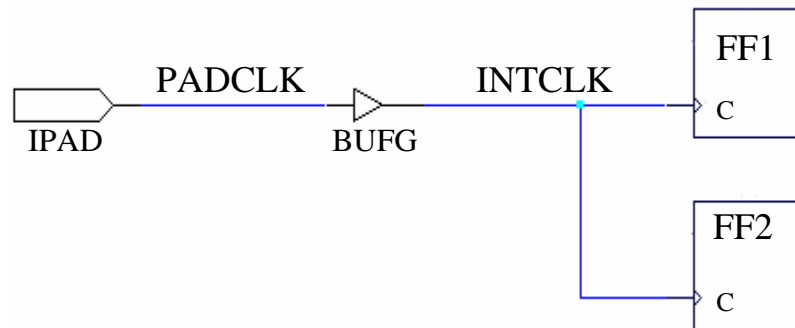
- 当TNM约束加在网线的时候，该网线所在的路径的有效同步元件将被赋予该TNM值





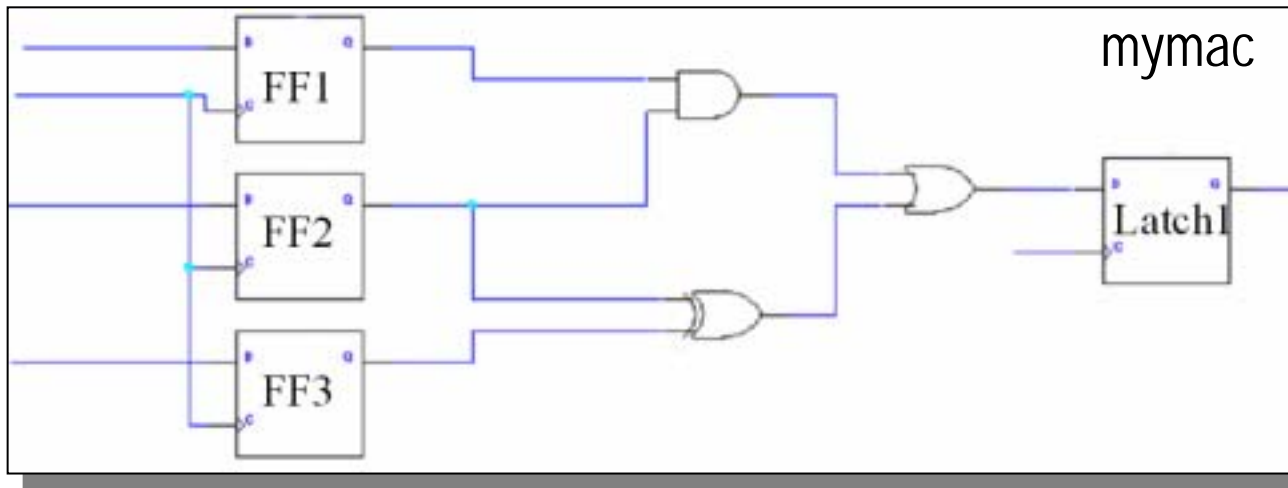
# TNM Examples

- 下列分组中包括哪些元件?
  - `NET PADCLK TNM = PADS padgroup; # 只有IPAD`
  - `NET PADCLK TNM = FFS flopgroup1; # 是一个空的分组，会报错`
  - `NET INTCLK TNM = FFS flopgroup2; # FF1, FF2`
  - `NET PADCLK TNM_NET = FFS flopgroup3; # FF1,FF2`



# TNM Examples

- 下列分组中包括那些元件?
  - `INST mymac TNM = FFS mymac_grp1; #FF1,FF2,FF3`
  - `INST mymac TNM = mymac_grp2; #FF1,FF1,FF3, Latch1`



# Combining Groups

- 除了用TNM约束定义分组外，还可以用TIMEGRP作下列定义
  - 将多个分组合并成一个分组  
`TIMEGRP newgroup = group1 group2 [group3...];`
  - 用排除的方法构成新的分组  
`TIMEGRP newgroup = group1 [group2...] EXCEPT group3 [group4...];`
  - 将触发器分组中有效时钟沿相同的元件定义成新的分组（时触发器分组的子集）  
`TIMEGRP newgroup = [RISING | FALLING] group1;`

# TIMGRP Example

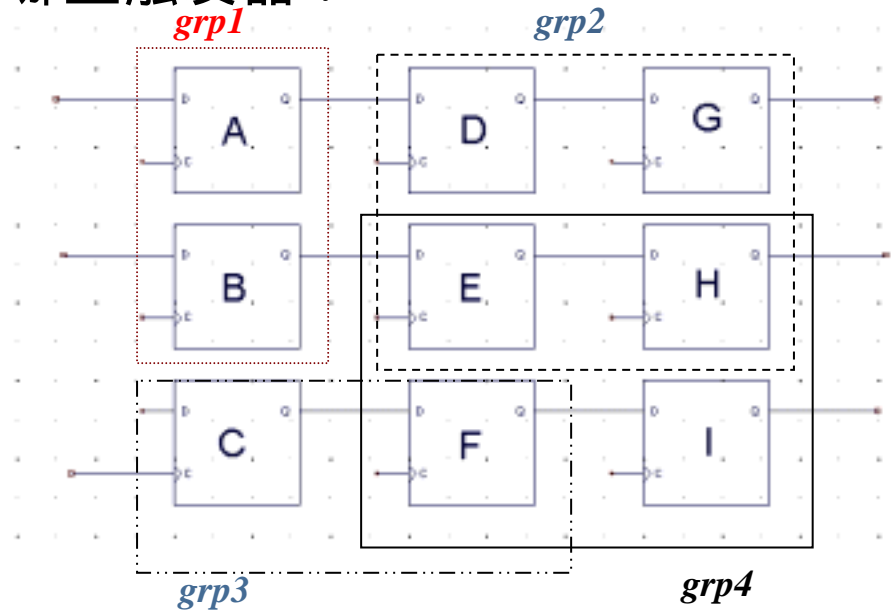
- 下列约束定义的分组中都有哪些触发器？

*grp1* = A, B

*grp2* = D, E, G, H

*grp3* = C, F

*grp4* = E, F, H, I



- `TIMEGRP manyffs = grp1 grp2 grp3 ;`
- `TIMEGRP largeone = grp2 grp3 EXCEPT grp4;`

# Outline

- 介绍
- 基本时序约束
- 建立分组
- 其他约束
- 约束优先级

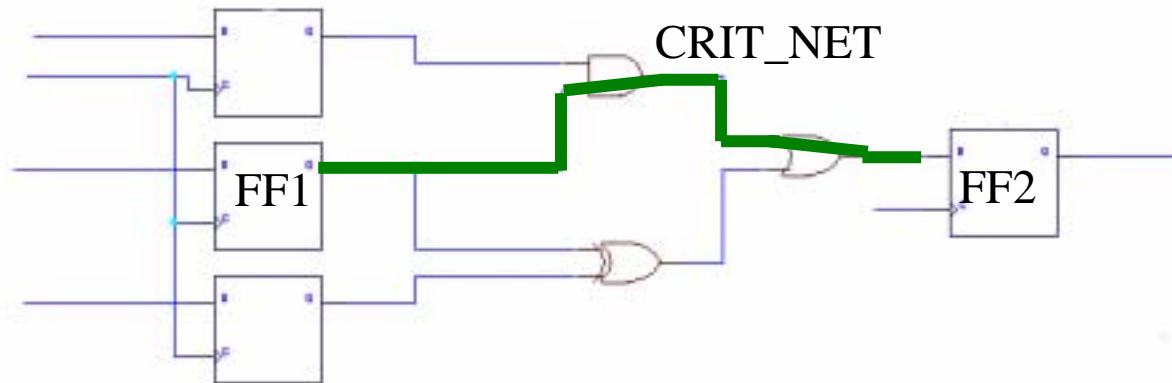


# TPTHRU

- 为什么要用 TPTHU?
  - TPTHU用于定义被约束的路径中的一个或一组中间点
- 约束描述和TNM约束类似
  - [NET|INST|PIN] *object\_name* TPTHU = *thru\_pt1*;
- 可以在基本时序相关约束 ( TIMESEPC ) 中应用
  - TIMESPEC *TSidentifier* = FROM *group1*  
THRU *thru\_pt1* [THRU *thru\_pt2* . . . ] TO *group2 value*;
  - *group1* 和 *group2* 是用户自定义的分组和预定义分组
  - *thru\_pt1* [*thru\_pt2*] 用于精确的表示被约束路径
  - *thru\_pt1* [*thru\_pt2*] 没有先后次序

# TPTHRU Example

- 只对粗的绿线所指示的路径进行约束:
  - INST FF1 TNM = start;
  - INST FF2 TNM = endpt;
  - NET CRIT\_NET TPTHRU = middle;
  - TIMESPEC TSmypath = FROM start THRU middle TO endpt 10 ns;



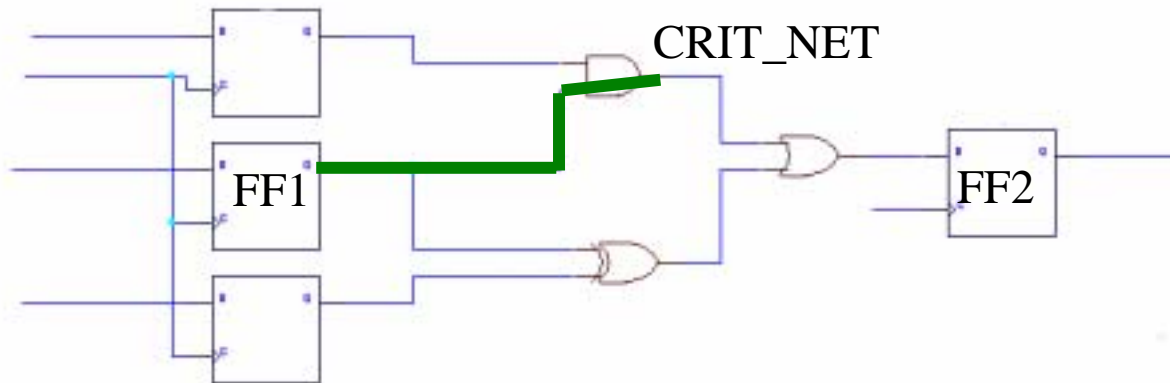
# TPSYNC

- 为什么要用 TPSYNC?
  - 为了将组合逻辑元件定义为时序路径中的起点和终点
- 约束描述和TPTHRU类似
  - `[NET|INST|PIN] object_name TPSYNC = sync_pt1;`
- 可以在基本时序相关约束 ( TIMESEPC ) 中应用
  - `TIMESPEC TSidentifier = FROM group1 TO sync_pt1 value;`
- 被TPSYNC定以为时序路径中的起点和终点的组合逻辑不能放在不同的LUT中
  - 会影响设计密度和性能



# TPSYNC Example

- 只对粗的绿线所指示的路径进行约束
  - `INST FF1 TNM = start;`
  - `NET CRIT_NET TPSYNC = new_endpt;`
  - `TIMESPEC TSmy_sync = FROM start TO new_endpt 5 ns;`



# Ignoring Selected Paths (TIG)

- TIG = Timing IGnore
- Why use TIG?
  - 不是设计中的所有路径都需要定义基本时序特性
  - 减少布线工作为这些路径进行布线所花费的时间，以及有可能产生的副作用
- 描述: `[NET|PIN] object_name TIG [= TSid1, TSid2...];`
  - 将定义在Ignores timing on all paths that contain *object\_name*的时序约束忽略
  - *TSid1, TSid2* are specific TIMESPEC identifiers to ignore
    - All other TIMESPECs will be applied normally to these paths
- TIG 也可以定义将 FROM:TO之间的路径忽略
  - Example: TIMESPEC TS\_ignore = FROM group1 TO group2 TIG;

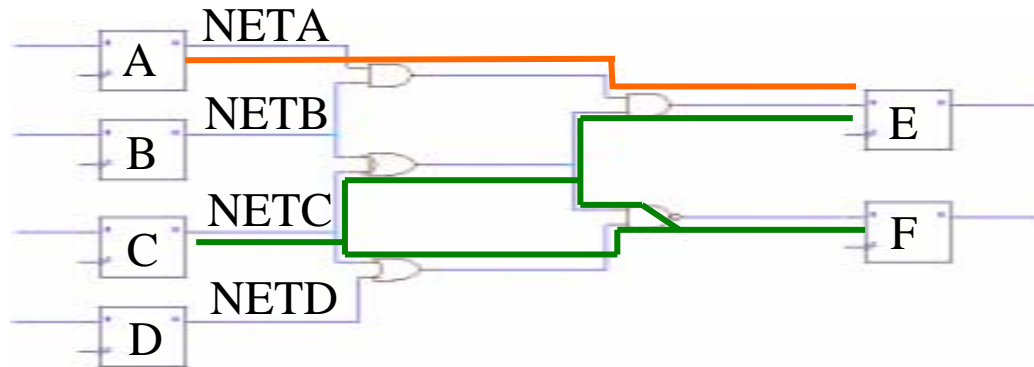
# Ignore Path

- TIG 用于定义无关路径，时序分析时会将加了TIG约束的网线开始的路径忽略
  - 可以是分组之间的路径
  - 也可以是被指定的NET开始的路径

```
TIMESPEC TS_IGNORE = FROM GROUP1 TO GROUP2  
TIG ;  
NET SLOW_NET TIG;
```
- Why use TIG?
  - 不是设计中的所有路径都需要定义基本时序特性
  - 减少布线工具为这些路径进行布线所花费的时间，以及有可能产生的负作用

# TIG Example

- NET NETC TIG;
  - 绿线上的约束都会被去掉
  - 从NETA, NETB, NETD开始的路径有一些部分正好覆盖到要被忽略的网络
    - These paths will NOT be ignored
- TIMESPEC TS\_TIG\_A2E = FROM FFS(A) TO FFS(E) TIG;



- 哪些路径不会被忽略掉?

# MAXDELAY and MAXSKEW

- 用于对时序比较紧张的路径进行约束
  - 全局时钟Buffer的net不需要定义MAXDELAY 和MAXSKEW
  - 主要用于那些没有用全局时钟Buffer的时钟网络
- MAXDELAY约束: `NET netname MAXDELAY = delay_time;`
- MAXSKEW 约束: `NET netname MAXSKEW = delay_time;`
  - *delay\_time* 可以是任何数字
  - *delay\_time* 的缺省单位是ns
- 不要将MAXSKEW设为0
  - 会引起软件错误的

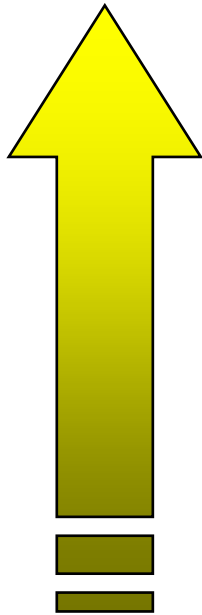
# Outline

- 介绍
- 基本时序约束
- 建立分组
- 其他约束
- 约束优先级



# Timing Constraint Priority

**Highest Precedence**



**Lowest Precedence**

TIG constraint (Timing IGnore)

FROM:THRU:TO constraints

Source and destination defined by user

Source or destination defined by user

Source and destination are pre-defined groups

FROM:TO constraints

Source and destination defined by user

Source or destination defined by user

Source and destination are pre-defined groups

OFFSET constraints

Specific data IOB

Time group of data IOBs

All data IOBs

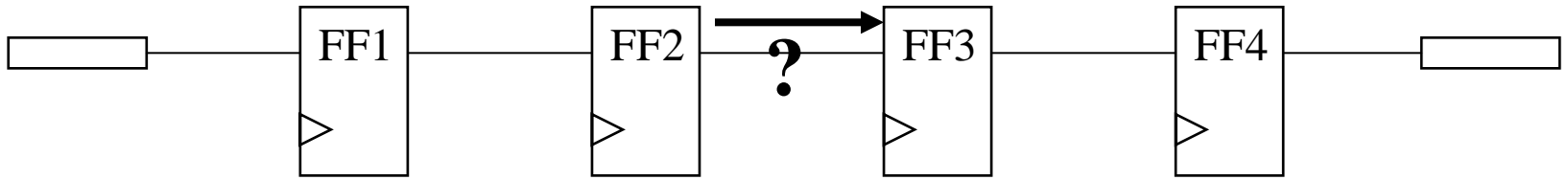
PERIOD constraints

# Priority for Conflicting TIMESPECs

- 定义基本时序约束的优先级是可定义的
  - 除了 MAXDELAY 和 MAXSKEW
- 在一条路径上有多种时序约束的情况下有用
  - 例如不止一个FROM TO 约束
- 约束描述: *<constraint\_definition> **PRIORITY** value;*
  - *value* 代表优先级 (从 1 到100)
    - 数字越小, 优先级越高
- 例子: **TIMESPEC** TS\_01 = **FROM** high **TO** low 8 ns **PRIORITY** 3;



# Priority Example

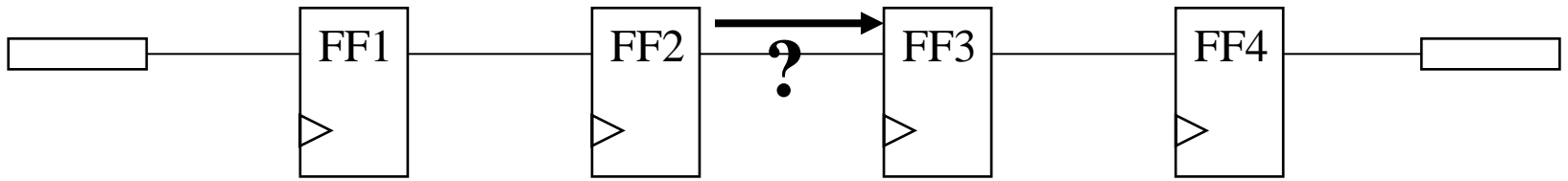


· 例子:

- FF1, FF2, FF3 在fastgroup分组中
- FF2, FF3, FF4 在slowgroup分组中
- TIMESPEC ts\_fast = FROM fastgroup TO fastgroup 3ns PRIORITY 2;
- TIMESPEC ts\_slow = FROM slowgroup TO slowgroup 10ns PRIORITY 1;

· 前面的约束对FF2和FF3之间的路径都作了约束，哪条约束才会真正的被约束到这条路径上呢？

# Priority Example



- FF2和FF3之间的路径被约束了两次（不同的），哪条约束才会真正的被约束到这条路径上呢？
  - 根据PRIORITY的属性定义，这条路径的约束应该是10ns而不是3ns
- 如果没有PRIORITY，也有一些因素会决定到底应该选用那条约束的
  - 可以看 Constraints Guide中Constraints Entry 的Constraints Priority部分
  - 或者在实现以后用 TRCE -tsi 命令产生一个基本时序交互报告(a timing specification interaction report)

# Outline

- 介绍
- 基本时序约束
- 建立分组
- 其他约束
- 约束优先级

# Agenda

- 约束 ( Constraints ) 的一般概念
  - 有哪些约束？
- 对设计进行约束
  - 怎么用这些约束？



# Constraining a Design



- First: 整体约束
- Second: I/O 专门约束
- Third: 内部专门约束
  - TIG
  - Multi-cycle clocks
  - Unrelated clocks

# Using the Constraints Editor

The screenshot shows the Xilinx Constraints Editor window titled "Xilinx Constraints Editor - [Global - clock\_2dcm.ngd / global.ucf\*]". The window has a menu bar (File, Edit, View, Window, Help) and a toolbar. The main area is divided into a table for clock constraints and a text area for generated constraints. Callouts highlight specific features: "Clock Timing" points to the table, "IO Timing" points to the "Clock to Pad" column, "PAD to PAD Timing" points to the "Pad to Pad..." input field, and "Generated Constraints: PERIOD, OFFSET, PAD TO PAD" points to the generated constraint text.

Clock Net Name	Period	Pad to Setup	Clock to Pad
clk20	20.000 ns HIGH 50 % ns HIGH 50 %	12.000 ns	30.000 ns
clk30	30.000 ns HIGH 50 % ns HIGH 50 %	10.000 ns	30.000 ns

Pad to Pad... 15.000 ns

Global Ports Advanced Misc

```
NET "clk20" TNM_NET = "clk20";
NET "clk30" TNM_NET = "clk30";
TIMESPEC "TS_clk20" = PERIOD "clk20" 20.000 ns HIGH 50 % ns HIGH 50 %;
TIMESPEC "TS_clk30" = PERIOD "clk30" 30.000 ns HIGH 50 % ns HIGH 50 %;
OFFSET = IN 10.000 ns BEFORE "clk30";
OFFSET = OUT 30.000 ns AFTER "clk30";
OFFSET = OUT 30.000 ns AFTER "clk20";
OFFSET = IN 12.000 ns BEFORE "clk20";
TIMESPEC "TS_P2P" = FROM "PADS" TO "PADS" 15.000 ns;
```

UCF Constraints (read-write) UCF Constraints (read-only) Source Constraints (read-only)

For Help, press F1

- The Global page will constrain the design

# Constraining a Design



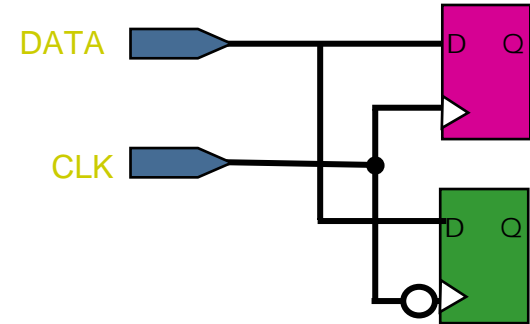
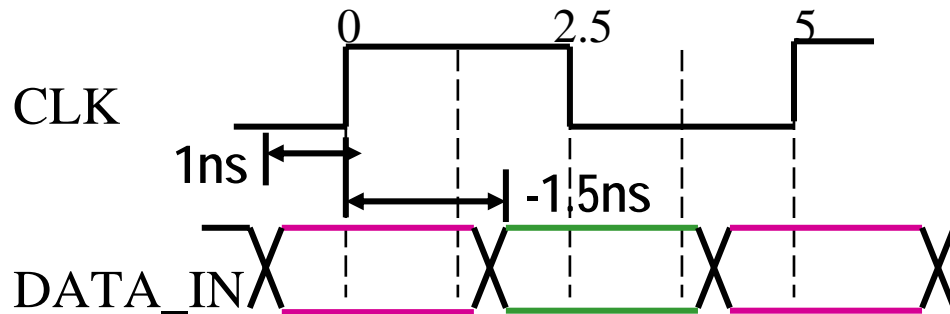
- First: Global constraints
- Second: I/O Details
- Third: Internal Details
  - TIG
  - Multi-cycle clocks
  - Unrelated clocks

# I/O Timing With Internal Clocks

- 偏移约束只能覆盖设计的输入输出和外部输入时钟之间的时序关系
- 用FROM:TO 来约束设计的输入输出和内部产生的时钟（由LUT或者FF产生）之间的时序关系
  - 创建分组  
INST DATA\_OUT[\*] TNM = DATA\_OUT;
  - 创建FROM:TO约束  
TIMESPEC TS\_DATA = FROM FFS TO  
DATA\_OUT 3;



# Constraining DDR inputs



定义时钟周期

```
NET CLK TNM_NET = CLK_GRP;
```

```
TIMESPEC "TS_CLK" = PERIOD "CLK_GRP" 5 ns HIGH 50%;
```

创建分组

```
INST DATA_IN[*] TNM = DATA_IN;
```

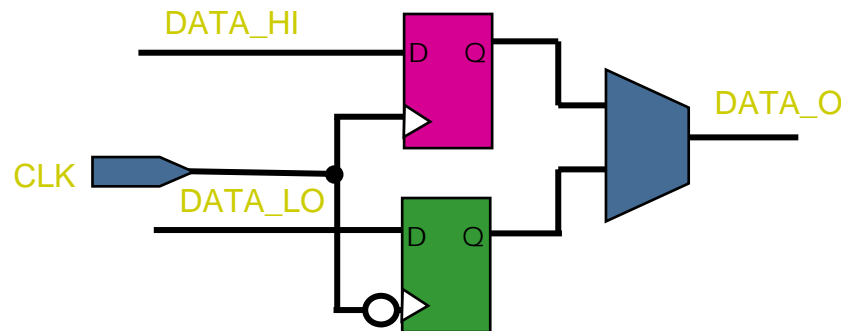
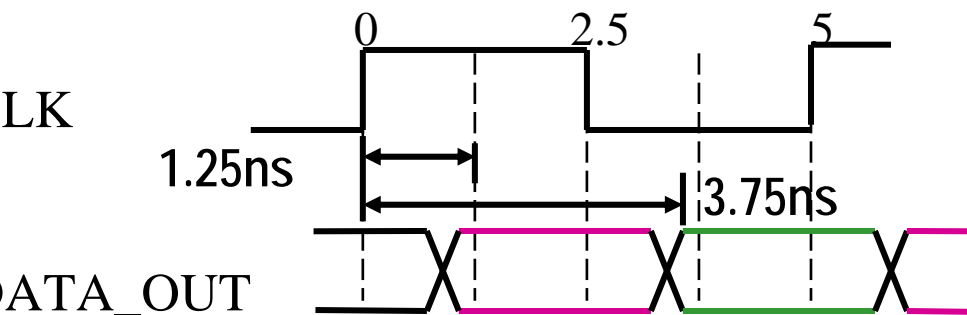
```
TIMEGRP FF_FALLING = FALLING CLK_GRP;
```

加偏移约束

```
OFFSET IN = 1 BEFORE CLK;
```

```
TIMEGRP DATA_IN OFFSET IN = -1.5 BEFORE CLK TIMEGRP  
FF_FALLING;
```

# Constraining DDR outputs



· 定义周期

```
NET CLK TNM_NET = CLK_GRP;
```

```
TIMESPEC "TS_CLK" = PERIOD "CLK_GRP" 5 ns HIGH 50%;
```

· 创建分组

```
INST DATA_OUT[*] TNM = DATA_OUT;
```

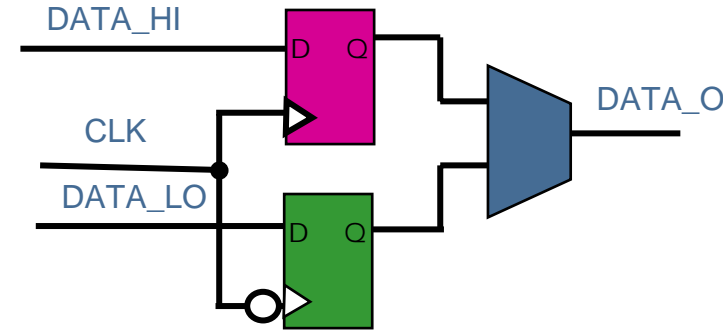
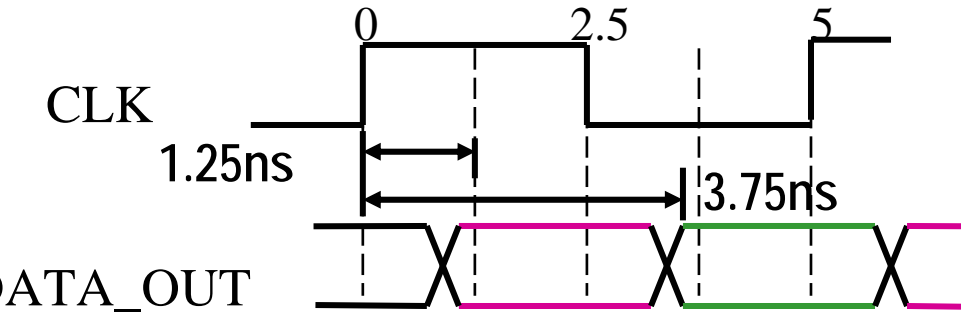
```
TIMEGRP FF_FALLING = FALLING CLK_GRP;
```

· 定义偏移约束

```
OFFSET OUT = 1.25 AFTER CLK;
```

```
TIMEGRP DATA_OUT OFFSET OUT = 3.75 AFTER CLK  
TIMEGRP FF_FALLING;
```

# Local Clock DDR outputs



- 加时钟周期约束

NET CLK MAXSKEW = 2ns;

NET CLK TNM = CLK\_GRP;

TIMESPEC "TS\_CLK" = PERIOD "CLK\_GRP" 5 ns HIGH 50%;

- 创建分组

INST DATA\_OUT[\*] TNM = DATA\_OUT;

- 加 FROM:TO 约束

TIMESPEC TS\_DDR\_OUT = FROM CLK\_GRP TO DATA\_OUT 1.25;

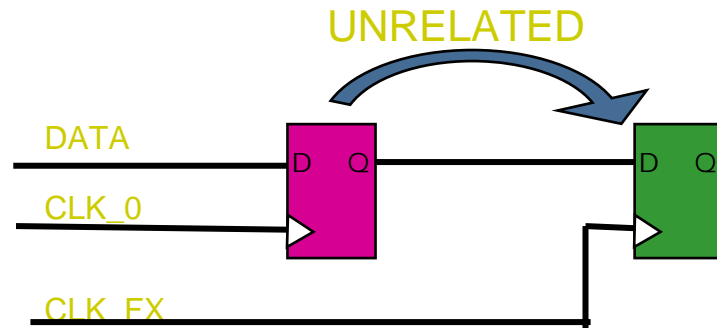
# Constraining a Design

- First: 整体约束
- Second: I/O 专门约束
- Third: 内部专门约束
  - TIG
  - Multi-cycle clocks
  - Unrelated clocks



# CLKDV/CLKFX Cross Clock Domains

- 两个不相关的时钟域之间的路径应该被设为伪路径（在时序分析的时候被忽略）



```
NET CLK_0 TNM = CLK0_GRP;
```

```
NET CLK_FX TNM = CLKFX_GRP;
```

```
TS_TIG0 = TIMESPEC FROM CLK0_GRP TO CLKFX_GRP TIG;
```

```
TS_TIG1 = TIMESPEC FROM CLKFX_GRP TO CLK0_GRP TIG;
```

# DCM Attribute

## CLKIN\_DIVIDE\_BY\_2

- 原始的时钟周期约束为2.5ns

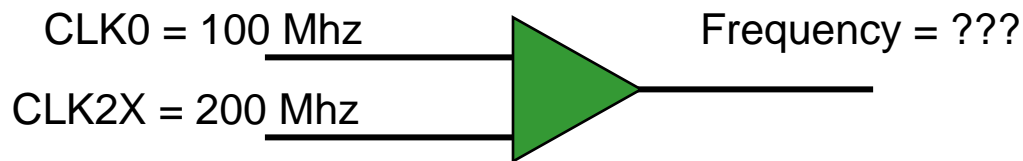
TS\_clk = PERIOD TIMEGRP "clk" 2.5 ns HIGH 50;

- Translate会在上面约束的前提下将CLK0的时钟周期约束为5.0ns

TS\_clk0 = PERIOD TIMEGRP "clk0" TS\_clk\*2 HIGH 50;



# Constraining BUFGMUX Clock



- Frequency是哪个?
  - 在PCF约束文件中的最后一个约束
- Use the PRIORITY keyword

```
NET CLK0 TNM_NET = CLK0_GRP;
```

```
TIMESPEC "TS_CLK0" = PERIOD "CLK0_GRP" 10 ns;
```

```
NET CLK2X TNM_NET = CLK2X_GRP;
```

```
TIMESPEC "TS_CLK2X" = PERIOD "CLK2X_GRP" TS_CLK0/2  
PRIORITY 1;
```

# Known Multi-Cycle Delays

- 对于已知的不需要在一个时钟周期内完成的路径用FROM:TO来约束
- 对于已知的没有时序要求的路径加上TIG约束



# Is the Design Fully Constrained?

- 在加了所有已知的时序约束以后作一个“sanity”检查
- 可以利用时序分析工具产生没有被时序约束覆盖到的路径的一个报告
  - 命令行： `trace -u n`
  - Timing Analyzer: 选中 Report Unconstrained paths

# Where Can I Learn More?

- 从[www.xilinx-china.com](http://www.xilinx-china.com)上技术要点的数据库中，用“timing constraints” 搜索，有很多资料可以参考
- 从[www.xilinx-china.com](http://www.xilinx-china.com) 的技术文档处的软件手册数据的链接中下载 Constraints Guide



# How to get support

- 登陆Xilinx中文技术支持网站，填写详细的问题描述  
<http://www.xilinx.com/cn/support/techsup/china.htm>
- 将详细的问题描述通过Email发给  
[zte-support@xilinx.com](mailto:zte-support@xilinx.com), [china-support@xilinx.com](mailto:china-support@xilinx.com)
- Xilinx 的FAE 和 GSD的工程师
  - 刘晖：0755 - 25882622 ext 3005 , [hans.liu@xilinx.com](mailto:hans.liu@xilinx.com)
  - 胡骏：0755 - 25882622 ext 3007 , [qin.hu@xilinx.com](mailto:qin.hu@xilinx.com)



# Thank You

