HWALT

Timing Closure Techniques

Walt Lu

“好的时序是设计出来的，不是约束出来的”

时序就是一种关系

1、这种关系的基本概念有哪些？

2、这种关系需约束吗？

3、各自的详细情况是如何的？

4、约束的方法有哪些？

5、这些约束可以分为那几大类？

6、这种关系仅仅通过约束来维系吗？

HWALT

Agenda

- 1、Vivado基本操作流程
- 2、时序基本概念
- 3、时序基本约束和流程
- 4、Baselining时序约束
- 5、CDC时序约束
- 6、I/O时序
- 7、例外时序约束
- 8、时序收敛优化技术

1、Vivado基本操作流程

Overview

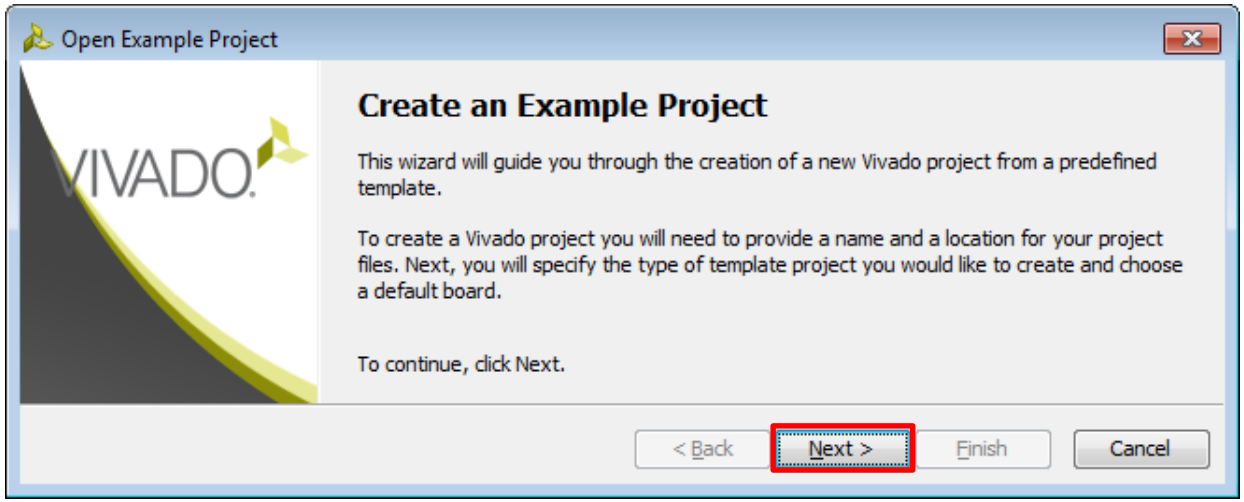
- 使用Vivado建立一个基本的工程
- lab0

Open Example Project

- Launch the Vivado Design Suite
- In the Vivado Getting Started window, click on the **Open Example Project** icon
  - The **Open Example Project - Create an Example Project** window opens

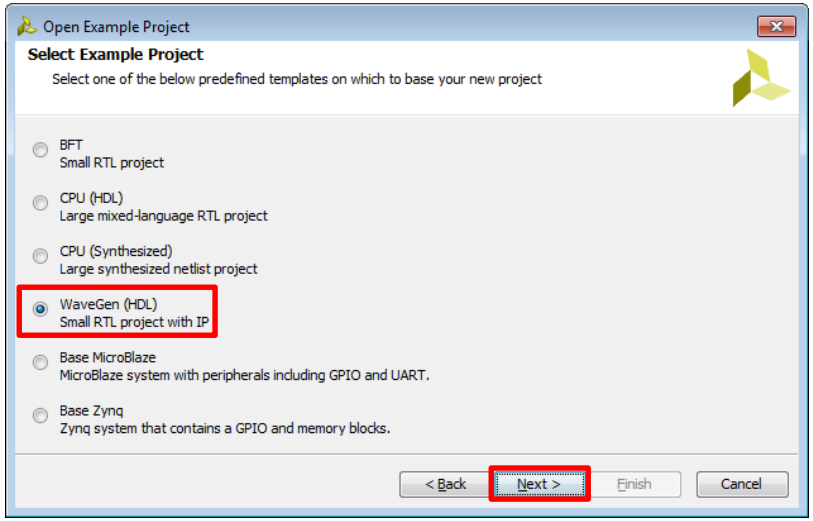
Open Example Project – Create an Example Project

- In the **Open Example Project - Create an Example Project**, click on the **Next >** button
  - The **Open Example Project - Select Example Project** window opens



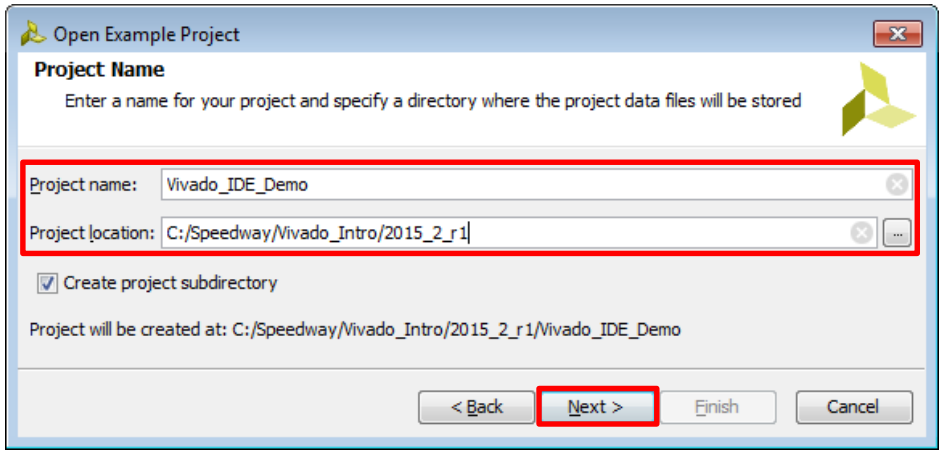
Open Example Project – Project Type

- In the **Open Example Project - Select Example Project** window, click on the **WaveGen (HDL)** radio button
- Click on the **Next >** button
  - The **Open Example Project - Project Name** window opens



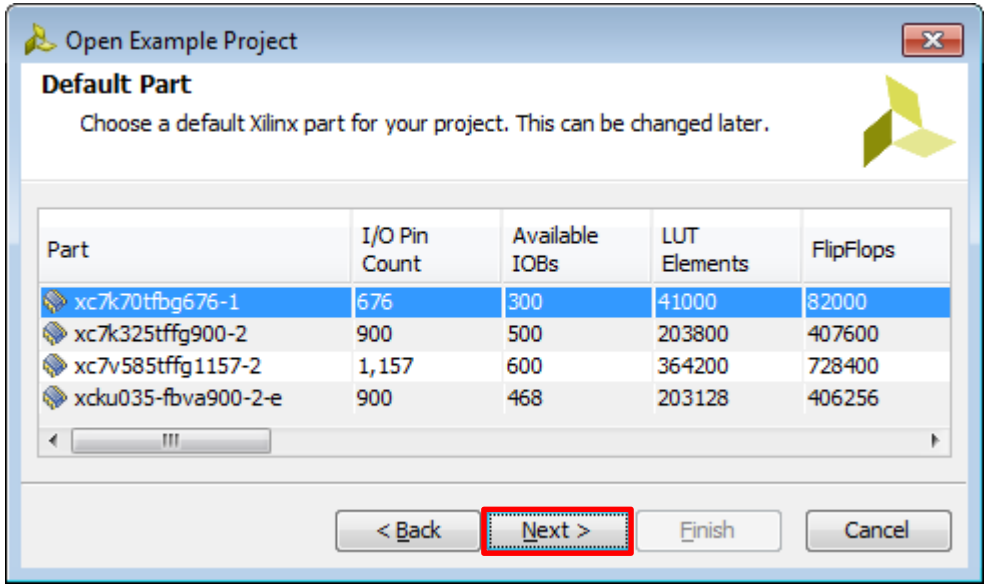
Open Example Project – Project Location

- In the **Open Example Project - Project Name** window, set **Project name:** to ***Vivado_IDE_Demo*** and **Project location:** to ***C:/Speedway/vivado_intro/2015_2_r1***
- Click on the **Next >** button
  - The **Open Example Project - Default Part** window opens



Open Example Project - Part

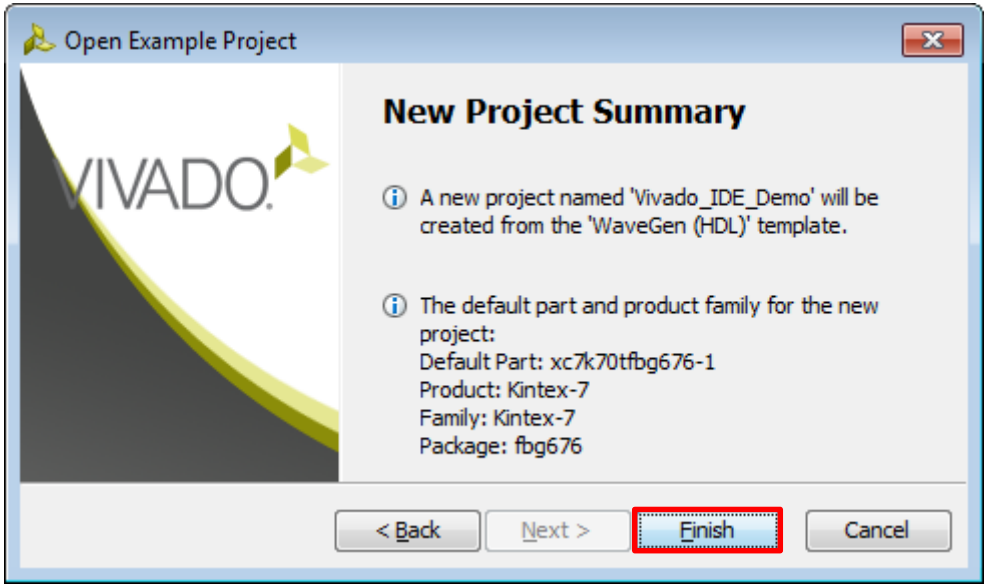
- In the **Open Example Project - Default Part** window, click on the **Next >** button
  - The **Open Example - New Project Summary** window opens



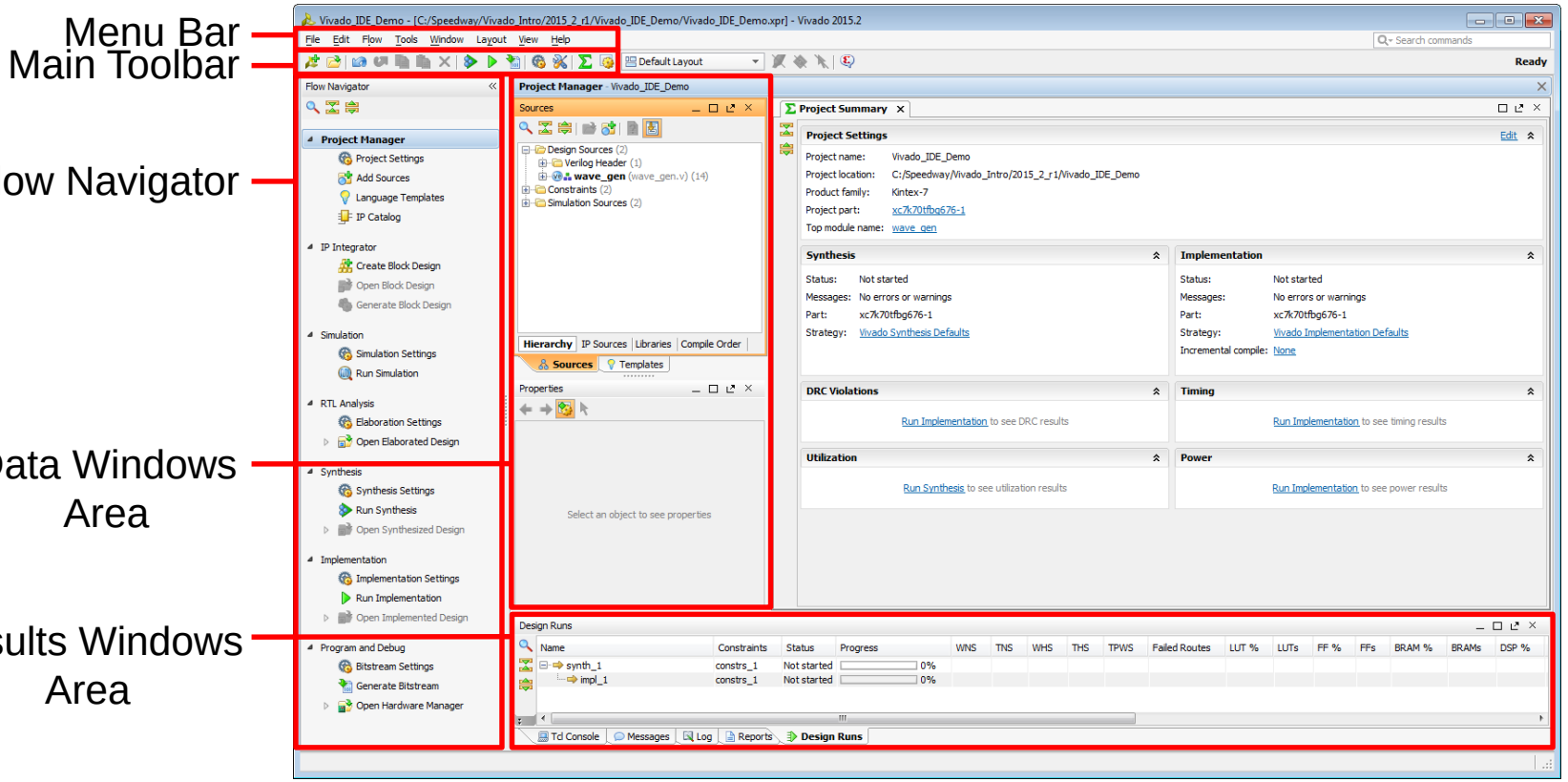
HWALT

Open Example Project - Summary

- In the **New Project Summary** window, click on the **Finish** button
  - The **Vivado IDE** opens

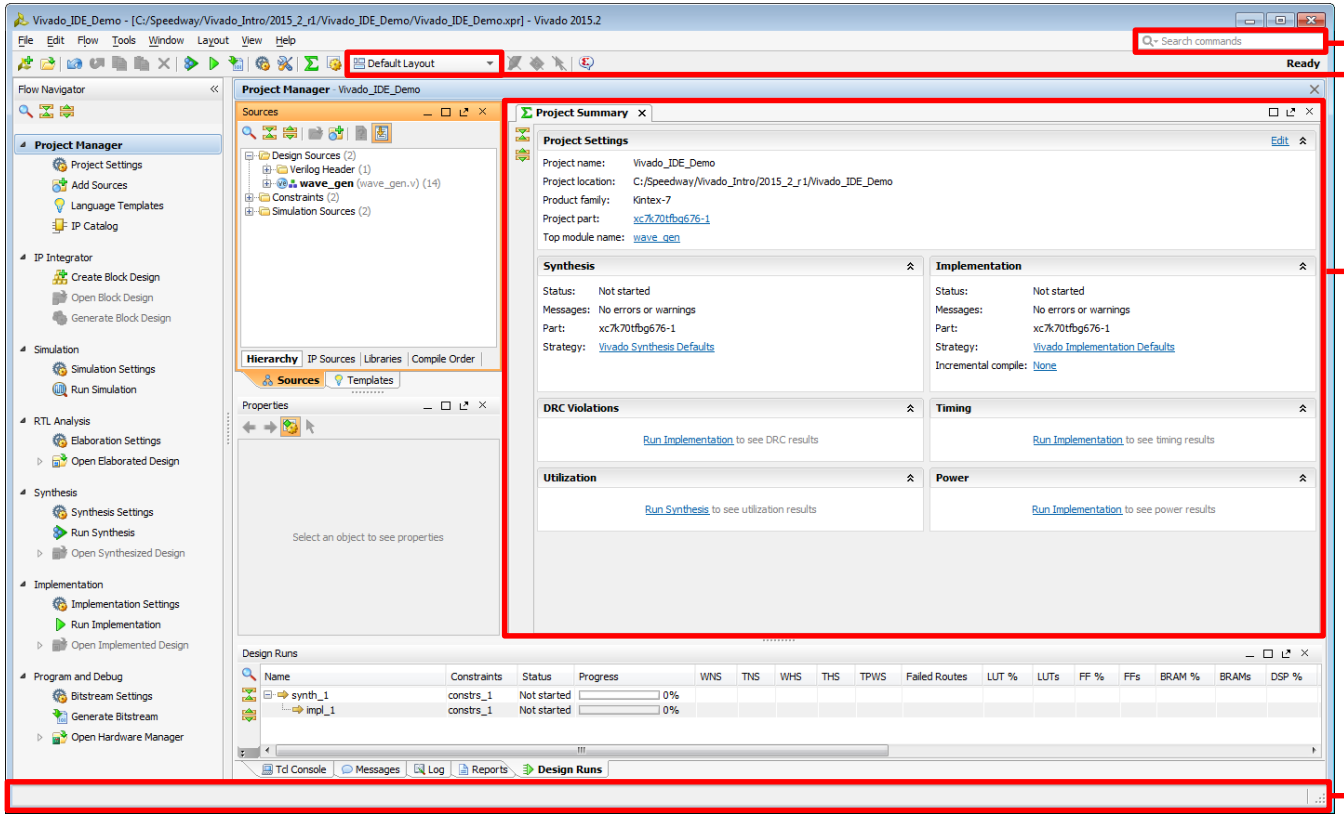


Vivado IDE



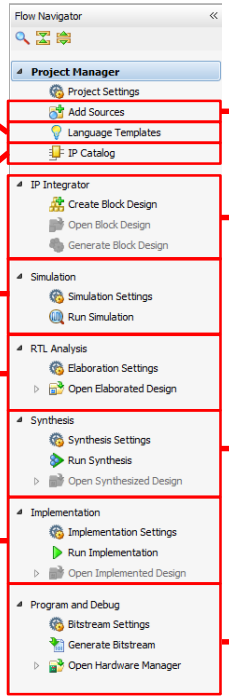
HWALT

Vivado IDE (continued)



Flow Navigator

- The Flow Navigator provides access to commands and tools



HDL Inference, Instantiation, constraints etc

Configure and add IP

HDL and Netlist simulations

RTL netlist, reports, schematic view

Place & route, reports

Add/create HDL, simulation, constraints sources etc.

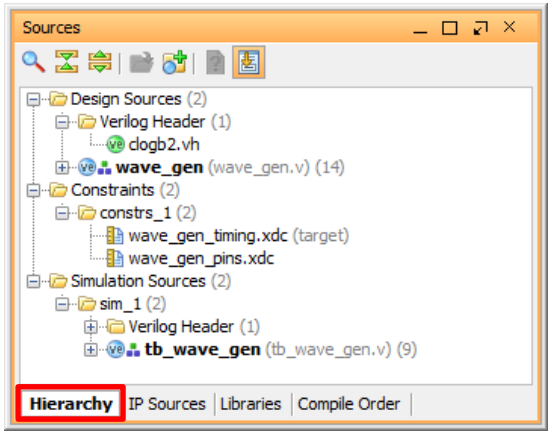
IP-centric design flow, graphically/tcl add/connect IP

Synthesized netlist, constraints, debug, reports, schematic view

Bitstream, program and debug

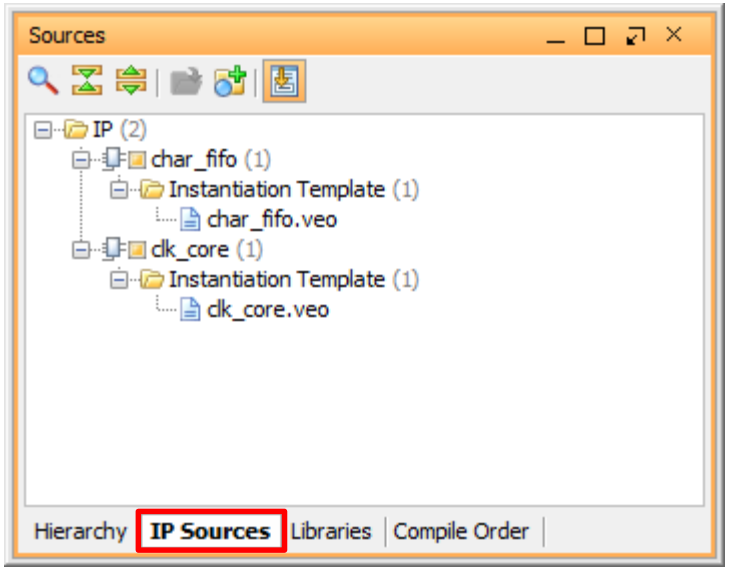
Sources - Hierarchy

- In the **Project Manager**, select the **Source window, Hierarchy** tab.
- The **Sources** window shows design sources, constraints, and simulation sources. Constraints and simulation files are grouped into sets (constr_1 and sim_1 in this case)



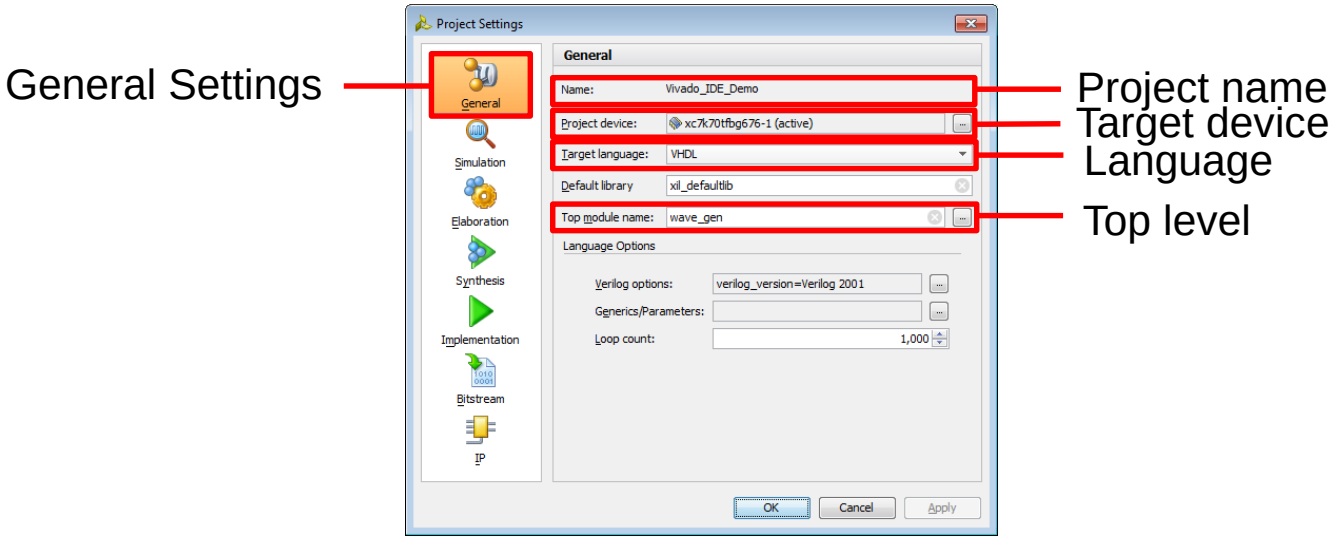
Sources – IP Sources

- In the **Project Manager**, select the **Source window** / **IP Sources** tab.
- IP forms a key part of designing for large capacity FPGA platforms.



Project Settings - General

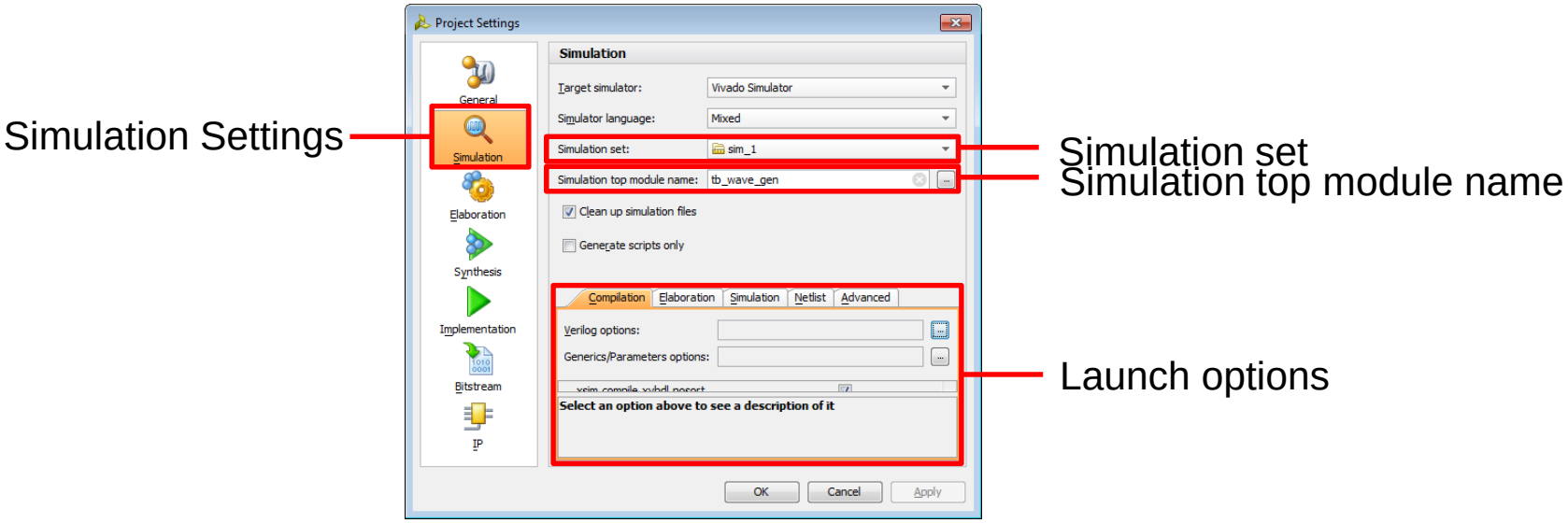
- On the Toolbar, click on the **Project Settings** icon
  - The **Project Settings** window opens
- **General**, shows the project name and enables you to change the part, language and specify top module name



HWALT

Project Settings - Simulation

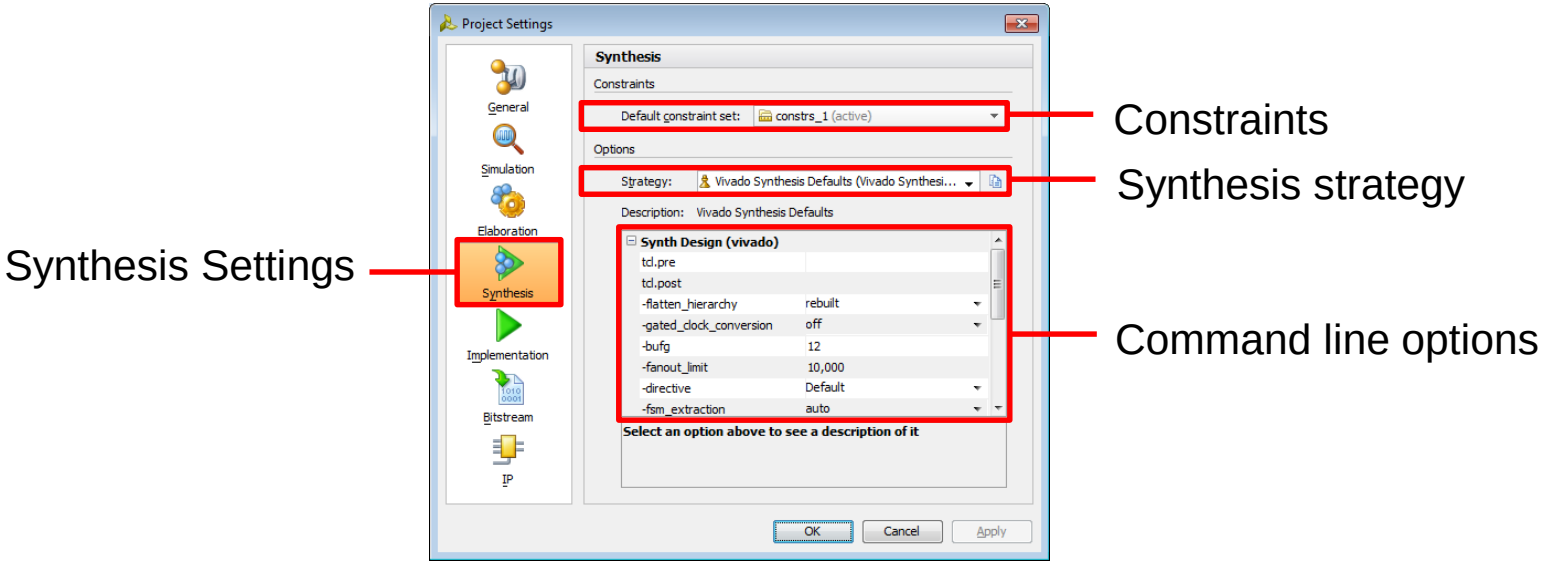
- In the **Project Settings** window, select **Simulation**
- **Simulation**, shows the simulation set, the simulation top module name and a tabbed listing of launch options



HWALT

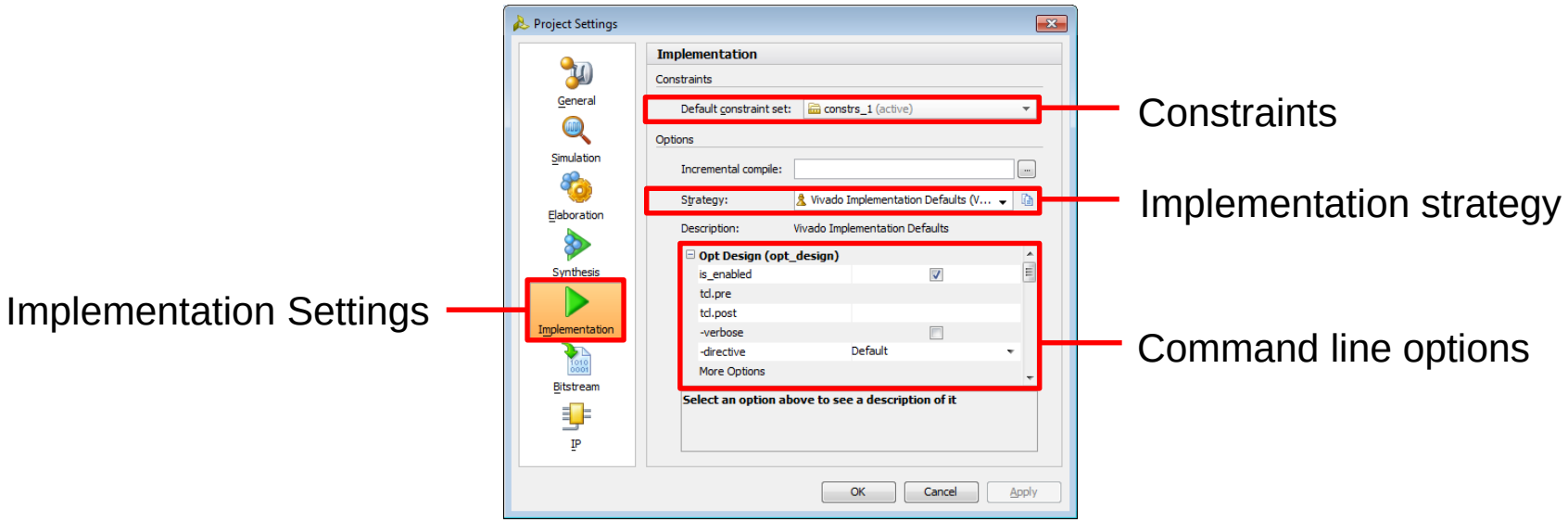
Project Settings - Synthesis

- In the **Project Settings** window, select **Synthesis**
- **Synthesis**, shows the default constraints set, synthesis strategies and synthesis command line options



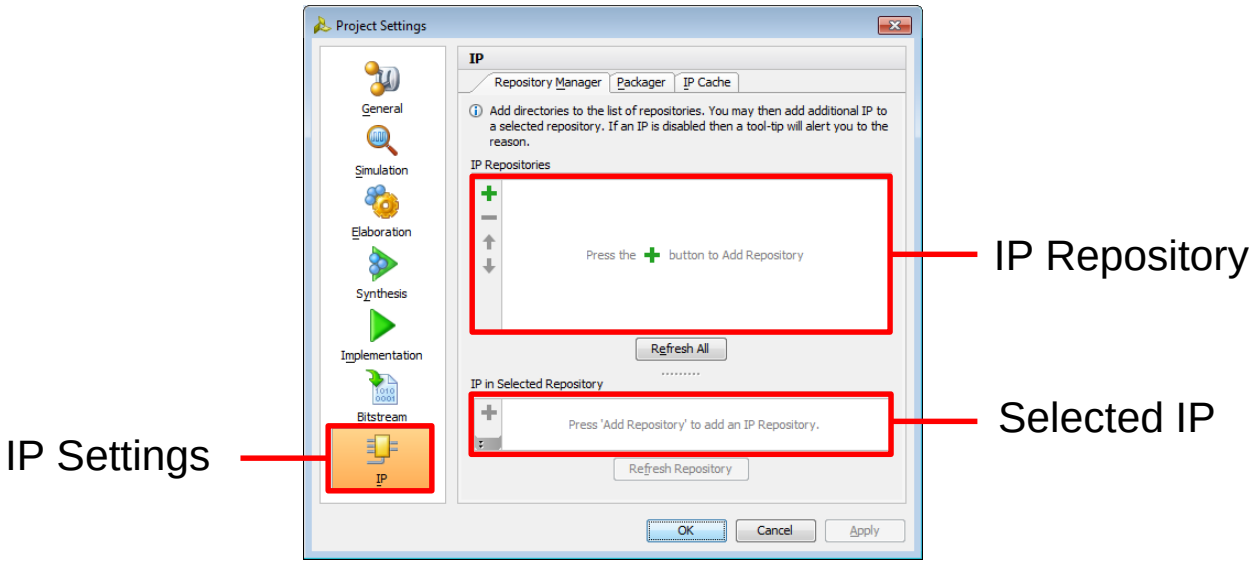
Project Settings - Implementation

- In the **Project Settings** window, select **Implementation**
- **Implementation**, shows the default constraints set, implementation strategies and implementation command line options



Project Settings - IP

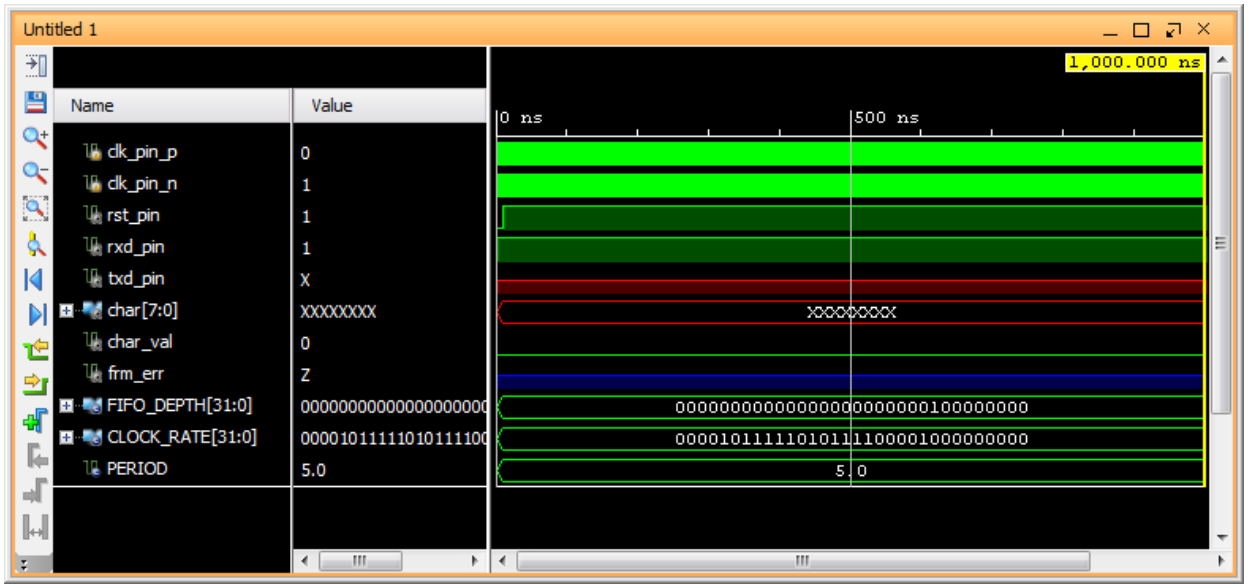
- In the **Project Settings** window, select **IP**
- **IP**, shows the included IP Repositories and selected IP from the repository



HWALT

Behavioral Simulation

- In the **Flow Navigator** pane, click on the **Run Simulation** icon and select **Behavioral Simulation**



HWALT

Overview

- 使用Vivado建立一个基本的工程
- lab0

Lab 0 – Vivado IDE Overview lab

Duration: 15 Minutes

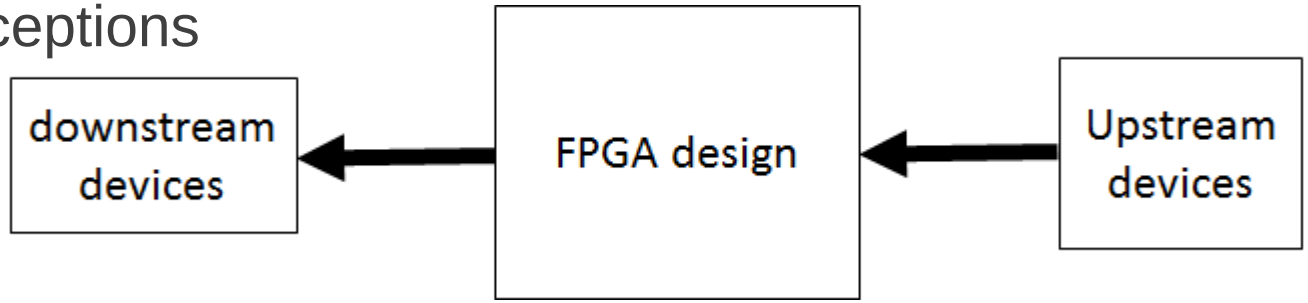
2、时序基本概念

Overview

- 时序路径
- XDC

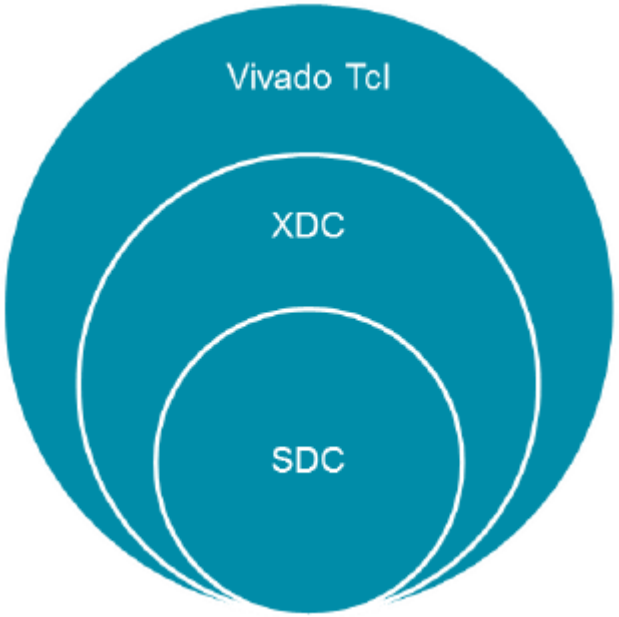
FPGA时序的基本概念

- 静态时序分析（Static Timing Analysis）简称STA，
  - 采用穷尽的分析方法来提取出整个电路存在的所有时序路径，计算信号在这些路径上的传播延时，检查信号的建立和保持时间是否满足时序要求，通过对最大路径延时和最小路径延时的分析，找出违背时序约束的错误并报告。
  - FPGA器件的需求取决于器件内部逻辑及上下游器件
- Input paths
  - 输入时序约束控制外部引脚到内部模块的路径
- Register-to-register paths
  - 寄存器-寄存器的约束，在同步时序电路中，就是周期的约束
- Output paths
  - 输出时序约束控制内部模块到外部引脚的路径
- Path specific exceptions
  - 具体例外路径约束



XDC

- XDC 是Xilinx Design Constraints 的简写
- 基础语法来源于业界统一的约束规范SDC（最早由Synopsys 公司提出，故名Synopsys DesignConstraints）
- XDC 的主要优势包括：
  - 1. 统一了前后端约束格式，便于管理；
  - 2. 可以像命令一样实时录入并执行；
  - 3. 允许增量设置约束，加速调试效率；
  - 4. 覆盖率高，可扩展性好，效率高；
  - 5. 业界统一，兼容性好，可移植性强



3、时序基本约束

Overview

- 时序约束的作用及方法
- 时序约束的三大类别介绍
- 高级时序约束介绍
- Baselining时序约束
- TCW使用及时序报告分析
- Lab1

Poll Question

- 大家是否使用时序约束进行自己的设计？
- 大家是否使用过TCW时序约束向导进行设计？

为什么明确的时序约束必不可少？

- **当时序约束不完善时(clock或者 IO)**
  - 工具会乐观分析路径时序
  - 不会报告违规，有可能导致硬件工作异常
- **当路径被错误约束时**
  - 工具会消耗大量时间优化并非真正需要优化的路径
  - 报告输出的时序违规问题，可能并不会影响硬件正常工作
- **错误的时序约束有可能导致HOLD时序违规**
  - 导致运行时间加长，由于非关键路径被优化导致建立时间违规
  - 工具首先会修复保持时间的违规，因为：
    - 设计无法在有保持时间违规的硬件上正常工作
    - 设计建立时间违规硬件可以工作，只是变慢

创建明确好约束的方法

- **创建时序约束的关键四步**
  1. Create clocks
  2. Define clocks interactions
  3. Set input and output delays
  4. Set timing exceptions

  } Baseline constraints

- **要包含IP的时序约束**

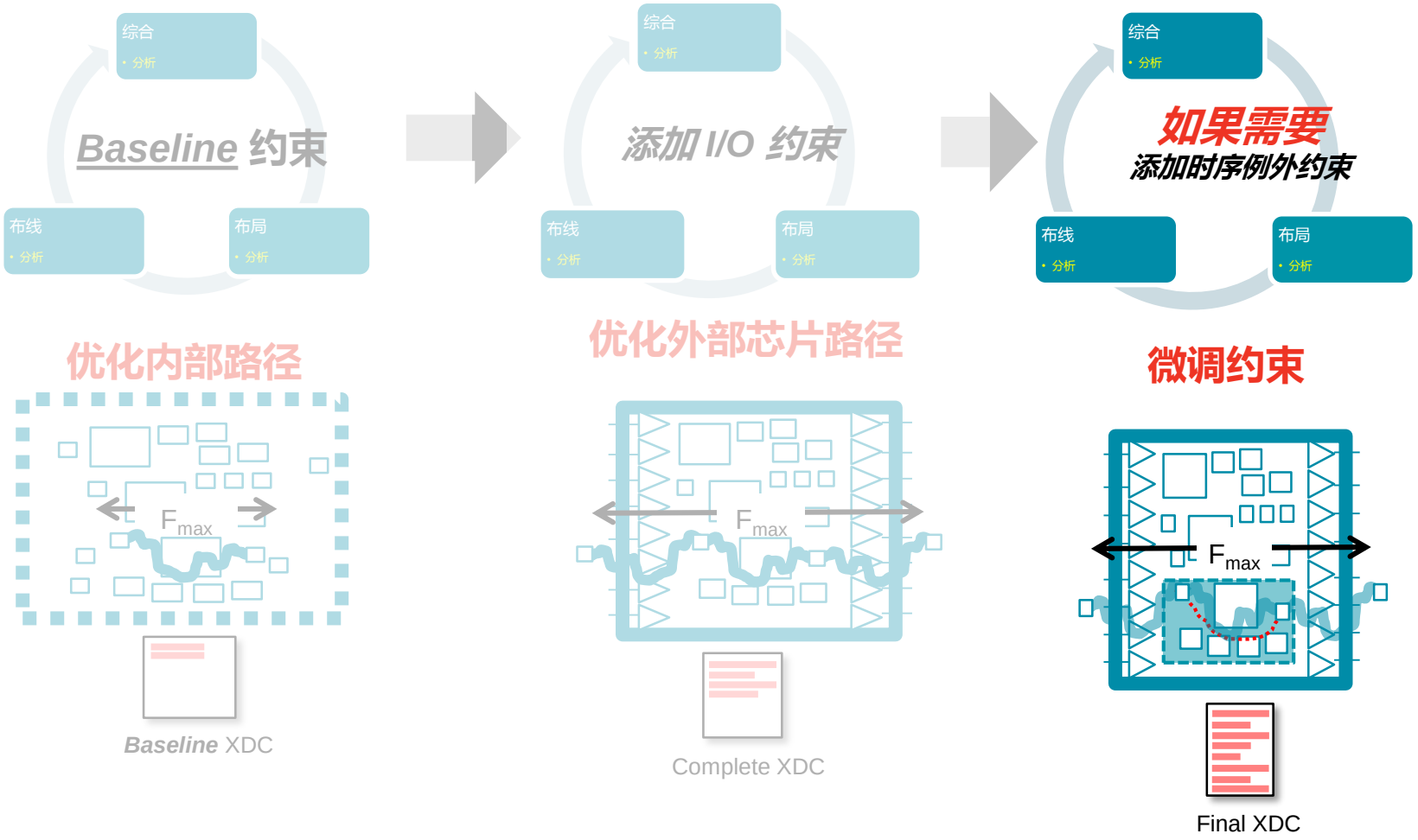
- **各个步骤验证时序约束情况**
  - Monitor unconstrained objects
  - Validate timing
  - Debug constraint issue post-synthesis
    - Analysis will be faster

- ☑ report_timing_summary
- ☑ check_timing
- ☑ report_clocks *(Note: Tcl only)*
- ☑ report_clock_networks
- ☑ report_clock_interaction
- ☑ XDC and TIMING DRCs

Overview

- 时序约束的作用及方法
- 时序约束的三大类别介绍
- 高级时序约束介绍
- Baselining时序约束
- TCW使用及时序报告分析
- lab2

时序约束的三大类别



时钟约束

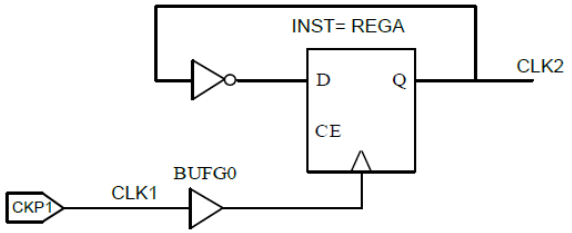
- 时钟约束必须最早创建。
  - 端口进来时钟以及GT 输出RXCLK/TXCLK 都须使用create_clock 自主创建
  - 如果是差分输入时钟，使用create_clock 创建约束P端即可。

```
create_clock -name clk_200 -period 5 [get_ports clk200_p]
```

- Vivado 工具自动推导的衍生时钟
  - 工具自动推导MMCM/PLL/BUFR 输出作为衍生时钟
  - 用户需要使用这些衍生钟的名字来创建I/O 约束、时钟关系或是时序例外等约束

```
create_generated_clock -name my_clk_name  [get_pins  mmcm0/CLKOUT] \
                       -source [get_pins  mmcm0/CLKIN] \
                       -master_clock  main_clk
```

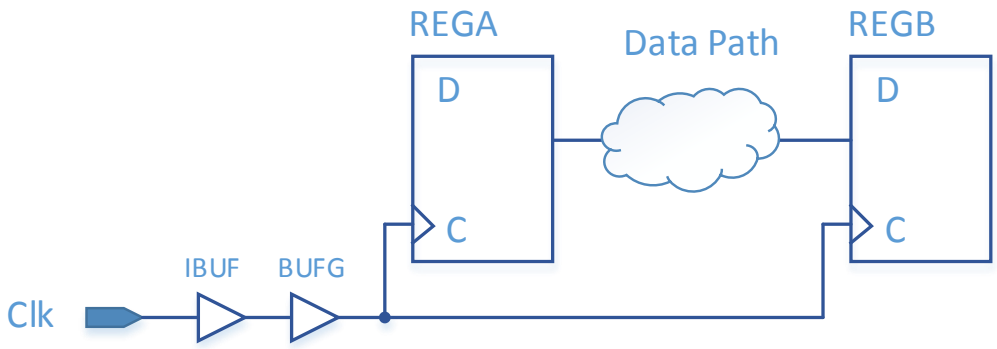
- 用户自定义的衍生时钟
  - 寄存器和组合逻辑搭建的分频器等，须由用户用create_generated_clock 来创建




```
create_clock -name clk1 -period 4 [get_ports CKP1]
create_generated_clock  -name clk2  [get_pins REGA/Q] \
                        -source [get_ports CKP1] -divide_by 2
```

时钟约束

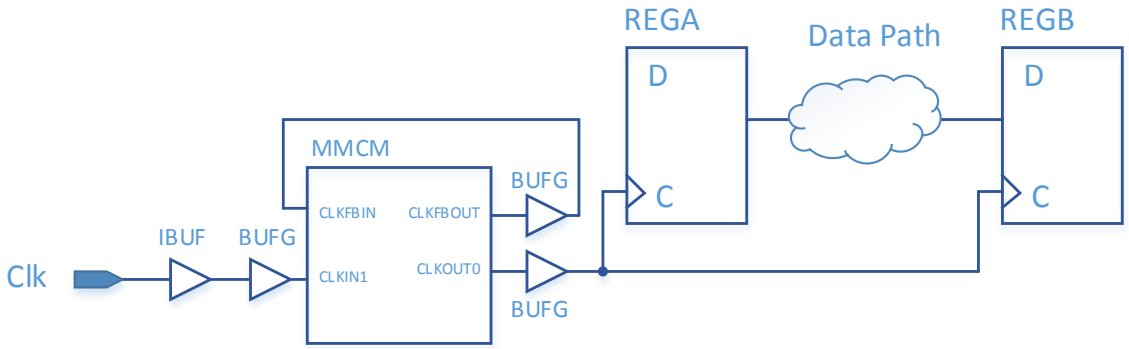
- Primary input clock
  - Input port
  - Propagated through BUFG (P)



- create_clock -name SysClk -period 10 -waveform {0 5} [get_ports Clk]

时钟约束

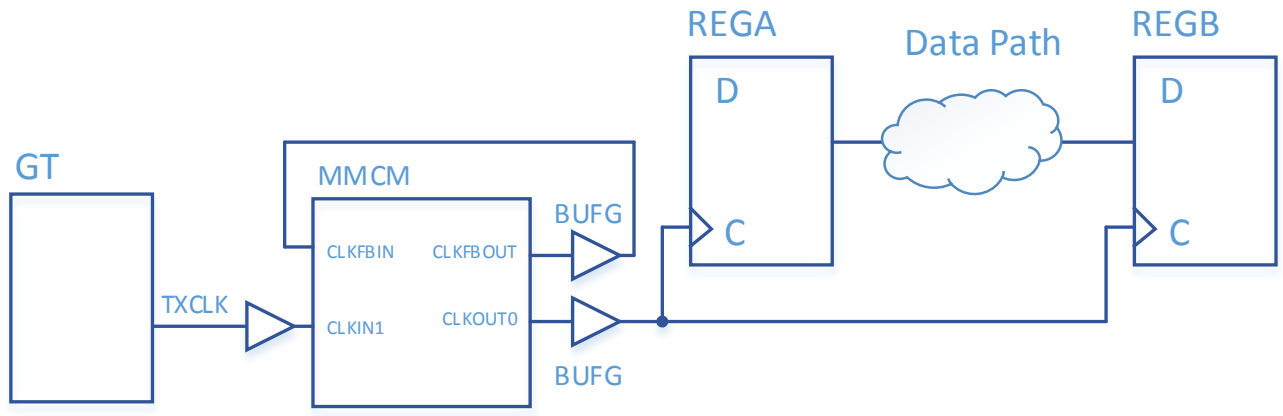
- Primary input clock
  - Input port
  - Propagated/through IBUF, BUFG and generated from MMCM (P,G)



- create_clock -name SysClk -period 10 -waveform {0 5} [get_ports Clk]

时钟约束

- Primary clock transceiver output
  - Gigabit transceiver clock output pin
  - Propagated through BUFG and generated from MMCM (P,G)



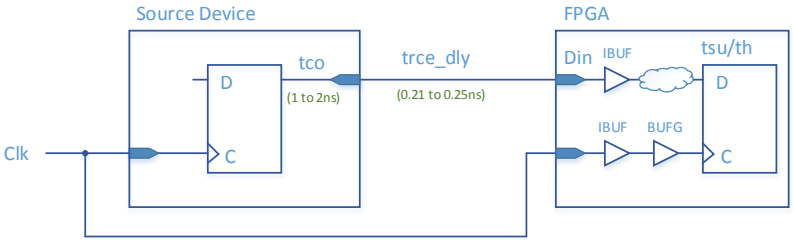
- create_clock -name txclk -period 6.667 [get_pins GT/TXOUTCLK]

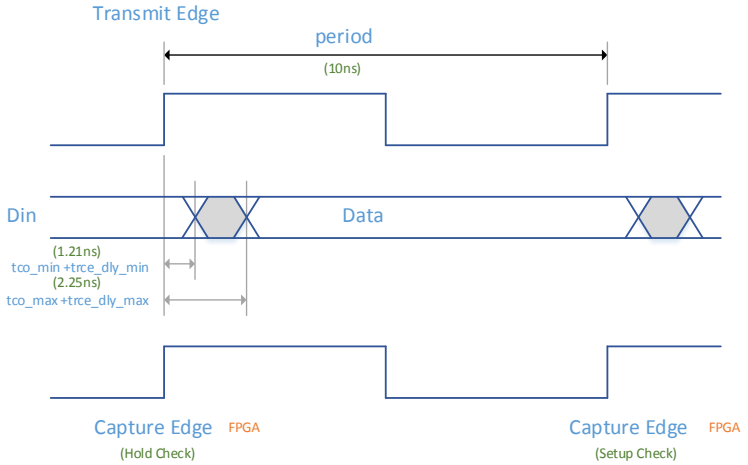
IO约束

- 设计初期可不加IO约束
  - 在设计的初期，可以不加I/O 约束，让工具专注于满足FPGA 内部的时序要求
  - 当时序要求基本满足后，再加上I/O 约束
- 不加任何I/O 约束的端口时序要求被视作无穷大
- XDC 中的set_input_delay / set_output_delay 对应于UCF 中 OFFSET IN / OFFSET OUT
  - OFFSET IN / OFFSET OUT 是从FPGA 内部延时的角度来约束端口时序，set_input_delay /set_output_delay 则是从系统角度来约束。
- 典型的I/O 时序，包括系统同步、源同步、SDR 和DDR 等等

IO约束

- Referenced to clock input
  - Max for setup analysis
  - Min for hold analysis

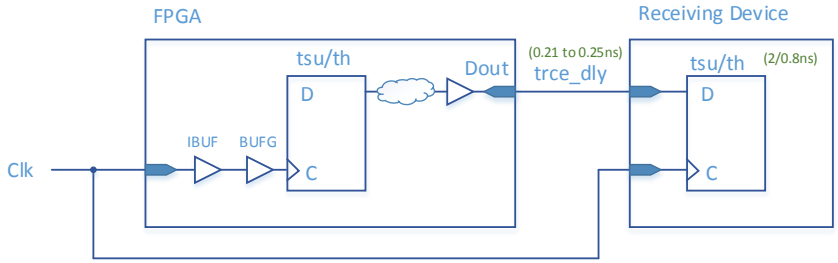


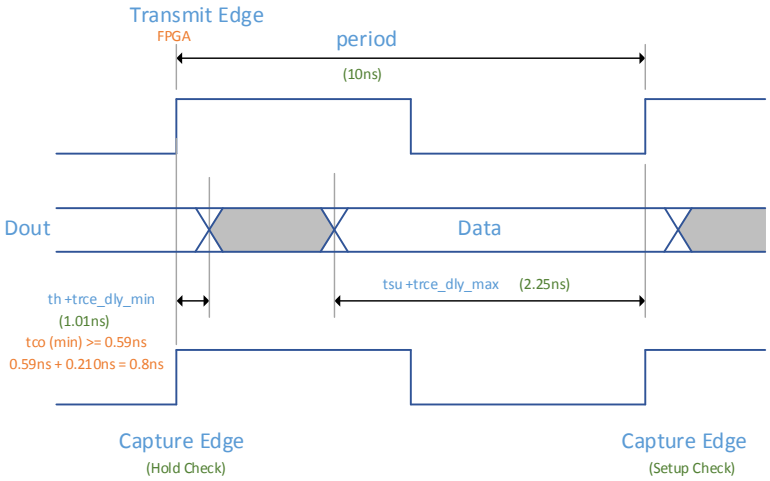


- set_input_delay -clock [get_clocks {Clk}] -min -add_delay 1.21 [get_ports {Din[*]}]
- set_input_delay -clock [get_clocks {Clk}] -max -add_delay 2.25 [get_ports {Din[*]}]

IO约束

- Referenced to clock output
  - Max for setup analysis
  - Min for hold analysis

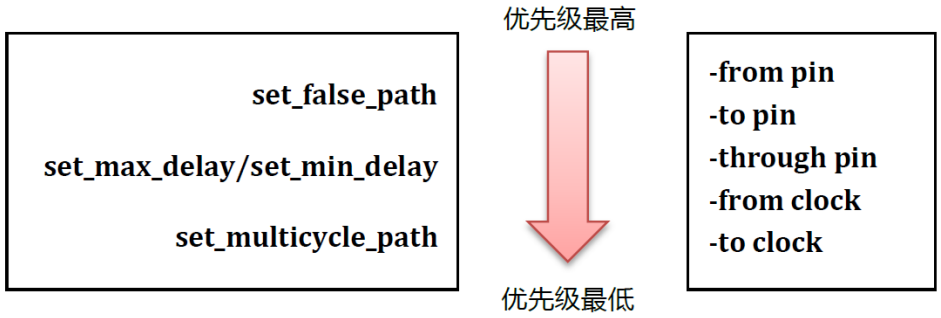


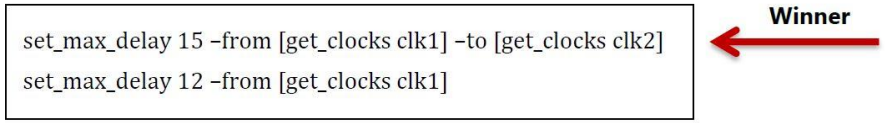


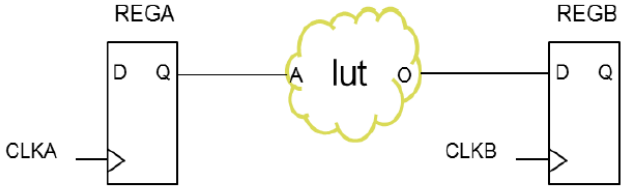
- set_output_delay -clock [get_clocks {Clk}] -min -add_delay -0.59 [get_ports {Dout[*]}]
- set_output_delay -clock [get_clocks {Clk}] -max -add_delay 2.25 [get_ports {Dout[*]}]

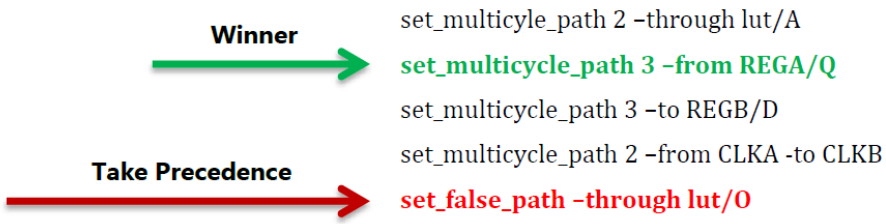
时序例外约束

- 时序例外约束包括
  - set_max_delay/set_min_delay
  - set_multicycle_path
  - set_false_path
- 这些XDC 的先后顺序存在优先级
- 自身也存在优先级限制，
  - 原则是针对同一条路径，对约束目标描述越具体的优先级越高







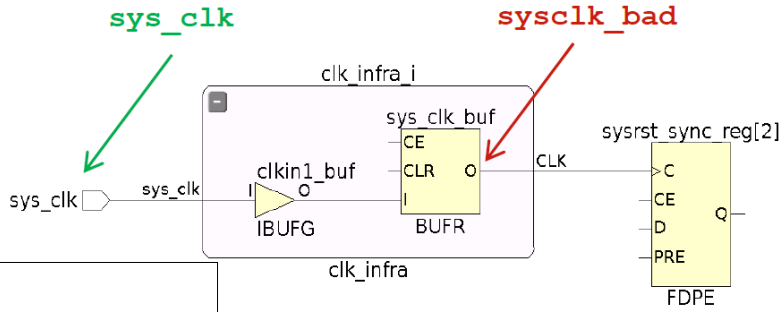


Overview

- 时序约束的作用及方法
- 时序约束的三大类别介绍
- 高级时序约束介绍
- Baselining时序约束
- TCW使用及时序报告分析
- Lab2

高级时钟约束

- 1、时序的零起点
  - create_clock 定义的主时钟的起点即时序的“零起点”，在这之前的上游路径延时都被工具自动忽略

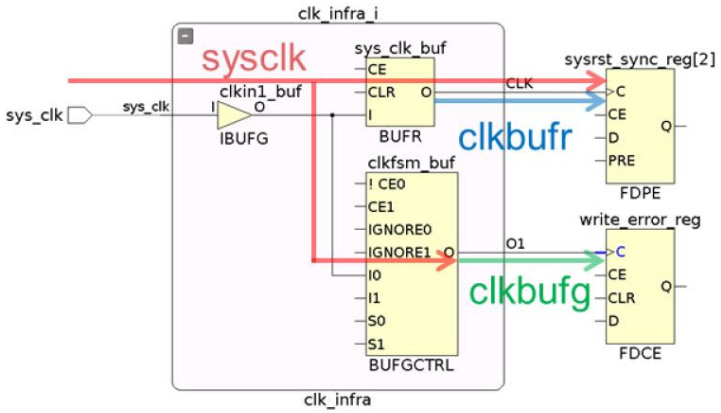



```
create_clock -name sysclk -period 10 [get_ports sys_clk]

Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------    -------------------
(clock sysclk rise edge)     0.000     0.000 r
                             0.000     0.000 r  sys_clk
net (fo=0)                   0.000     0.000    clk_infra_i/sys_clk
IBUFG (Prop_ibufg_I_O)       0.766     0.766 r  clk_infra_i/clkin1_buf/O
net (fo=3, unplaced)         1.109     1.875    clk_infra_i/clkin1
BUFR (Prop_bufr_I_O)         0.314     2.189 r  clk_infra_i/sys_clk_buf/O
net (fo=10, unplaced)        1.109     3.298    sys_clk_int
                                     r  sysrst_sync_reg[2]/C


create_clock -name sysclk_bad -period 10 [get_pins clk_infra_i/sys_clk_buf/O]

Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------    -------------------
(clock sysclk_bad rise edge)
                             0.000     0.000 r
BUFR                         0.000     0.000 r  clk_infra_i/sys_clk_buf/O
net (fo=10, unplaced)        1.109     1.109    sys_clk_int
                                     r  sysrst_sync_reg[2]/C
```

高级时钟约束

2、时钟定义的先后顺序

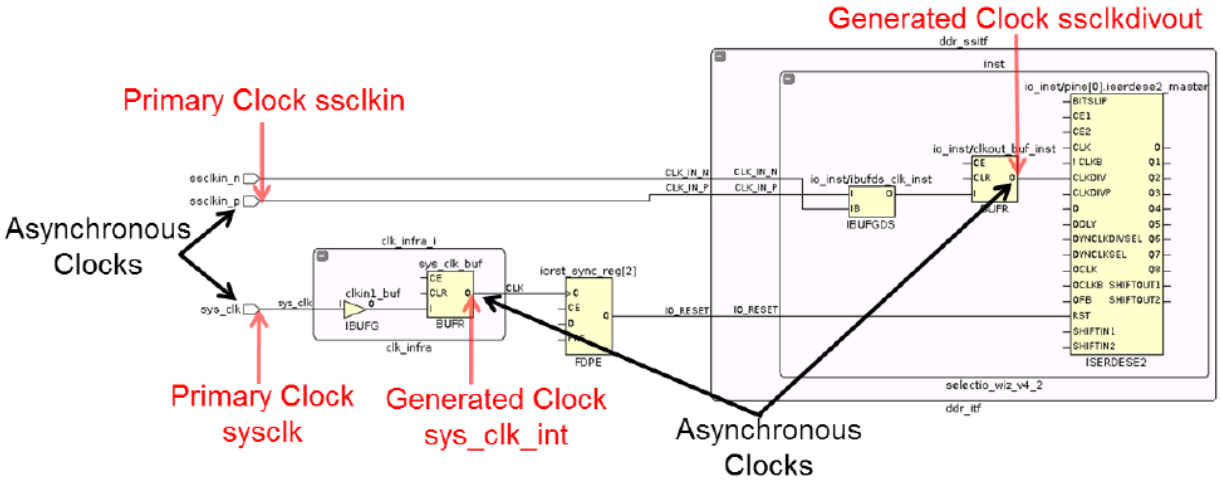
- 时钟的定义也遵从XDC/Tcl 的一般优先级
- 同一个点上，用户定义的时钟会覆盖工具自动推导的时钟，
- 后定义的时钟会覆盖先定义的时钟。
- 若要二者并存，必须使用 -add 选项。




```
create_clock -name sysclk -period 10 [get_ports sys_clk]
create_generated_clock -name clkbufg -source [get_ports sys_clk] -divide_by 1 \
            [get_pins clk_infra_i/clkfsm_buf/O]
create_generated_clock -name clkbufr -source [get_ports sys_clk] -divide_by 1 \
            [get_pins clk_infra_i/sys_clk_buf/O] -add -master_clock sysclk
```

高级时钟约束

- 3、同步时钟和异步时钟
  - 所有的时钟都会被约束缺省时，默认为是相关的
  - 即网表中所有存在的时序路径都会被Vivado 分析
  - FPGA 设计人员必须通过约束告知工具，哪些路径是无需分析的，哪些时钟域之间是异步的。

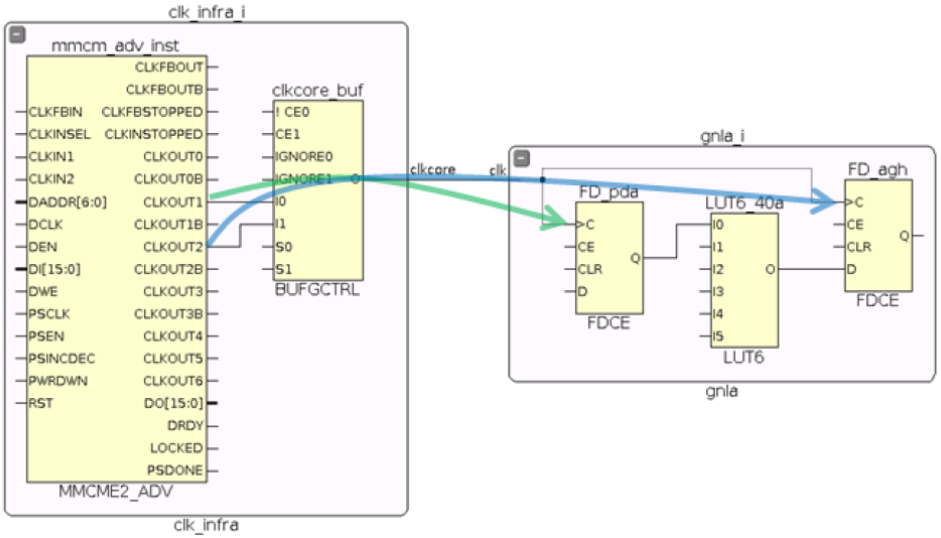



```
set_clock_groups -name sys_ss_async -asynchronous \
                 -group [get_clocks -include_generated_clocks sysclk] \
                 -group [get_clocks -include_generated_clocks ssclkin]
```

高级时钟约束

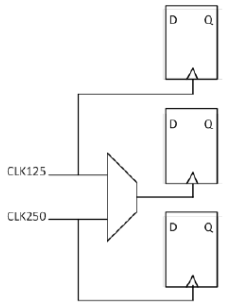
4、重叠（单点多个）时钟

– 重叠时钟是指多个时钟共享完全相同的时钟传输网络，例如两个时钟经过一个MUX选择后输出的时钟，在有多种运行模式的设计中很常见




```
set_clock_groups -physically_exclusive \
                 -group [get_clocks clk125] \
                 -group [get_clocks clk250]
```

```
create_generated_clock -name clk125_bufgctrl \
            -divide_by 1 [get_pins bufgctrl_i/O] \
            -source [get_ports bufgctrl_i/I0]
create_generated_clock -name clk250_bufgctrl \
            -divide_by 1 [get_pins bufgctrl_i/O] \
            -source [get_ports bufgctrl_i/I1] \
            -add -master_clock clk250

set_clock_groups -physically_exclusive \
                 -group clk125_bufgctrl \
                 -group clk250_bufgctrl
```



Overview

- 时序约束的作用及方法
- 时序约束的三大类别介绍
- 高级时序约束介绍
- Baselining时序约束及TCW使用
- 时序报告分析
- Lab2

Baselining约束

- Create clocks
  - 首先是时钟约束，确保所有的时钟都被约束
- Define clocks interactions
  - 检查时钟交互，定义时钟的相互关系，确保CDC跨时钟域路径的安全
- 尽早创建beselining约束
- 生成最简单的时序约束集
  - Clocks including generated clocks包含内部产生的所有时钟
  - Clock domain crossing检查时钟域交互
  - Clock relationships定义时钟相互关系
- 覆盖大部分的时序路径
  - FPGA内部的时序路径
- baselining 约束还要包括IP的约束
- 根据分析情况改善约束
  - Baselining 和时序约束验证流程

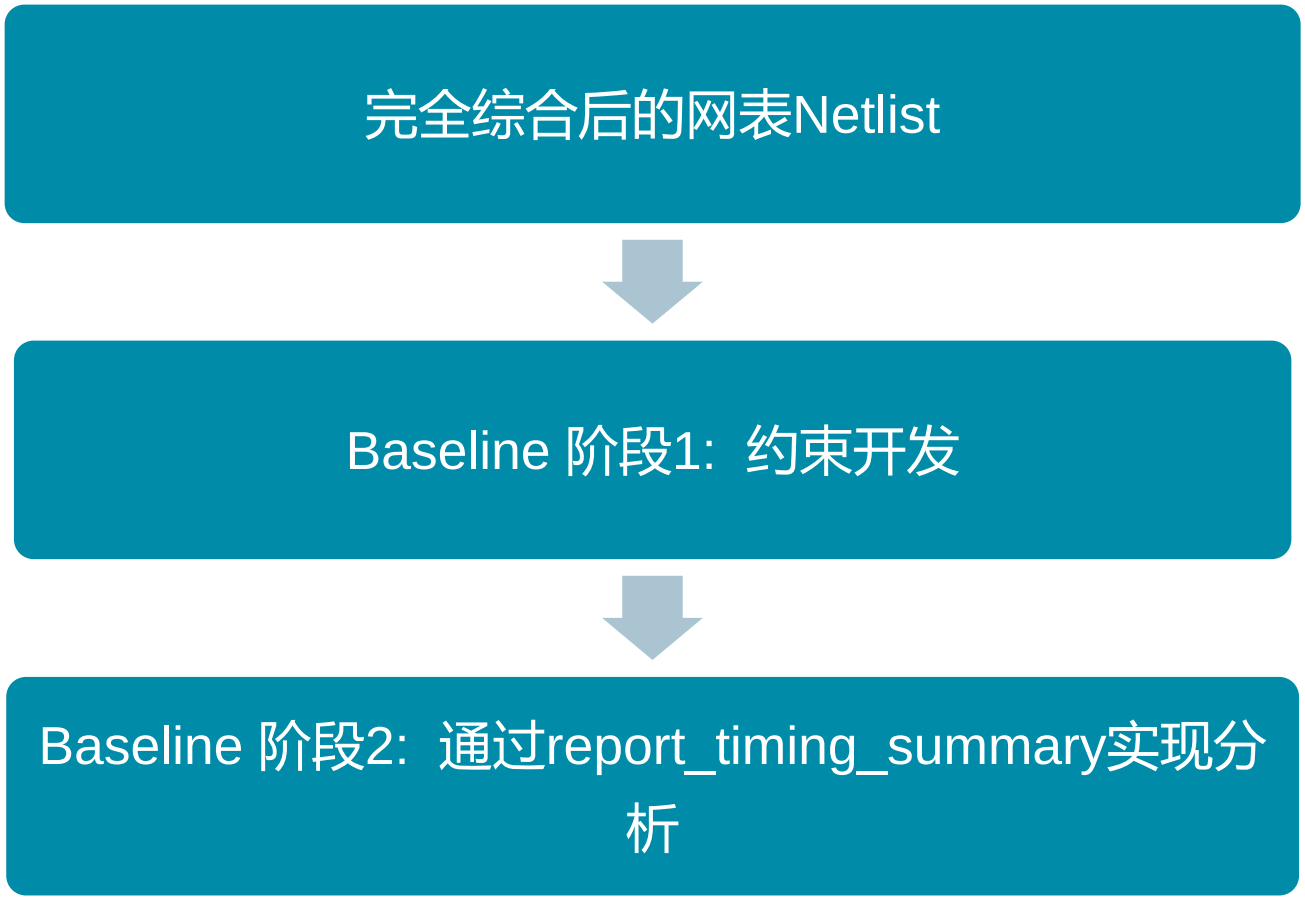
两主要问题

- 设计是否已经准确并适当地进行了约束？
- 期望的时序收敛是否合理？
  - 过约束会导致更多的资源及运行时间，甚至出现失败违规时序
  - 不进行约束也许没有时序错误，有可能导致硬件工作异常
- 如果上述两个问题答案都是否定的，那么就没有必要运行 implementation
  - 因为得到的结果不是工具真正能够实现的结果

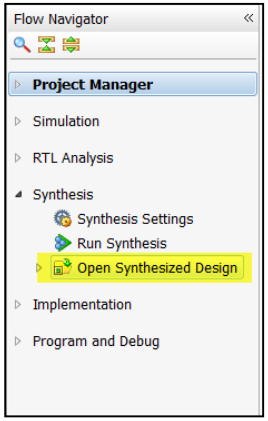
如何使用时序约束向导TCW进行baselining约束

- Disable 用户的XDC文件
- 创建新的baseline XDC文件，并设置成target
- 运行时序约束向导TCW
  - 约束所有时钟及时钟相互关系
  - 运行CDC跨时钟域报告检查CDC问题
- 第一遍先跳过IO约束
- 迭代各个阶段，验证各个阶段的时序问题
  - 根据迭代结果添加必要的例外约束
  - 确保内部时序满足要求
- 在随后的流程中再添加IO和其他例外约束
  - 迭代各个阶段，验证各个阶段的时序问题

使用Vivado进行Baselining 设计

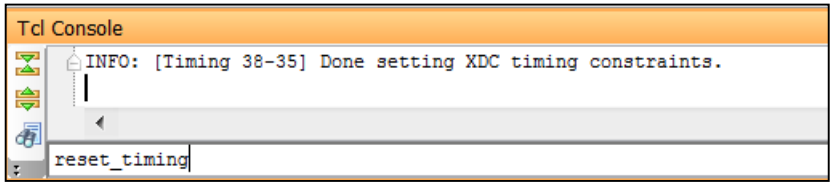


阶段1 从头进行约束开发的过程

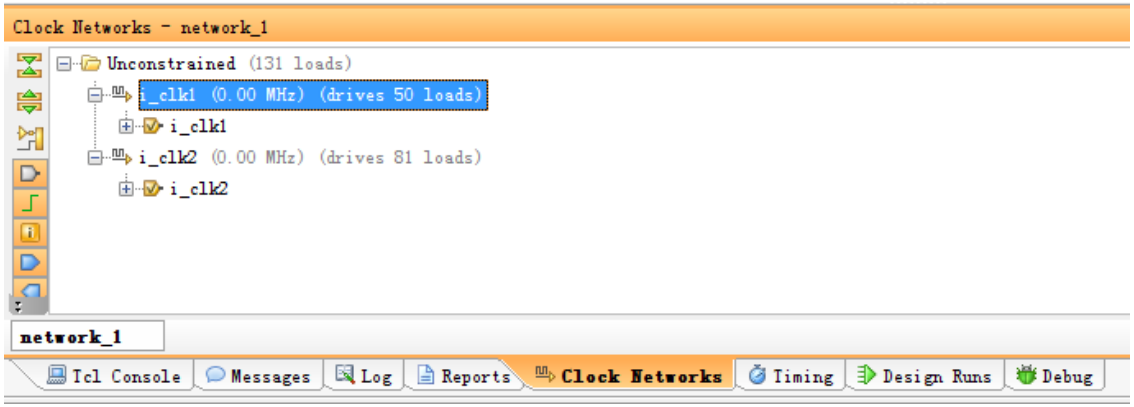


- Q. 如何开始？
  - A. Open synthesized design in Vivado IDE

- Q. 如何确保从头启动约束？
  - A. tcl_console> reset_timing

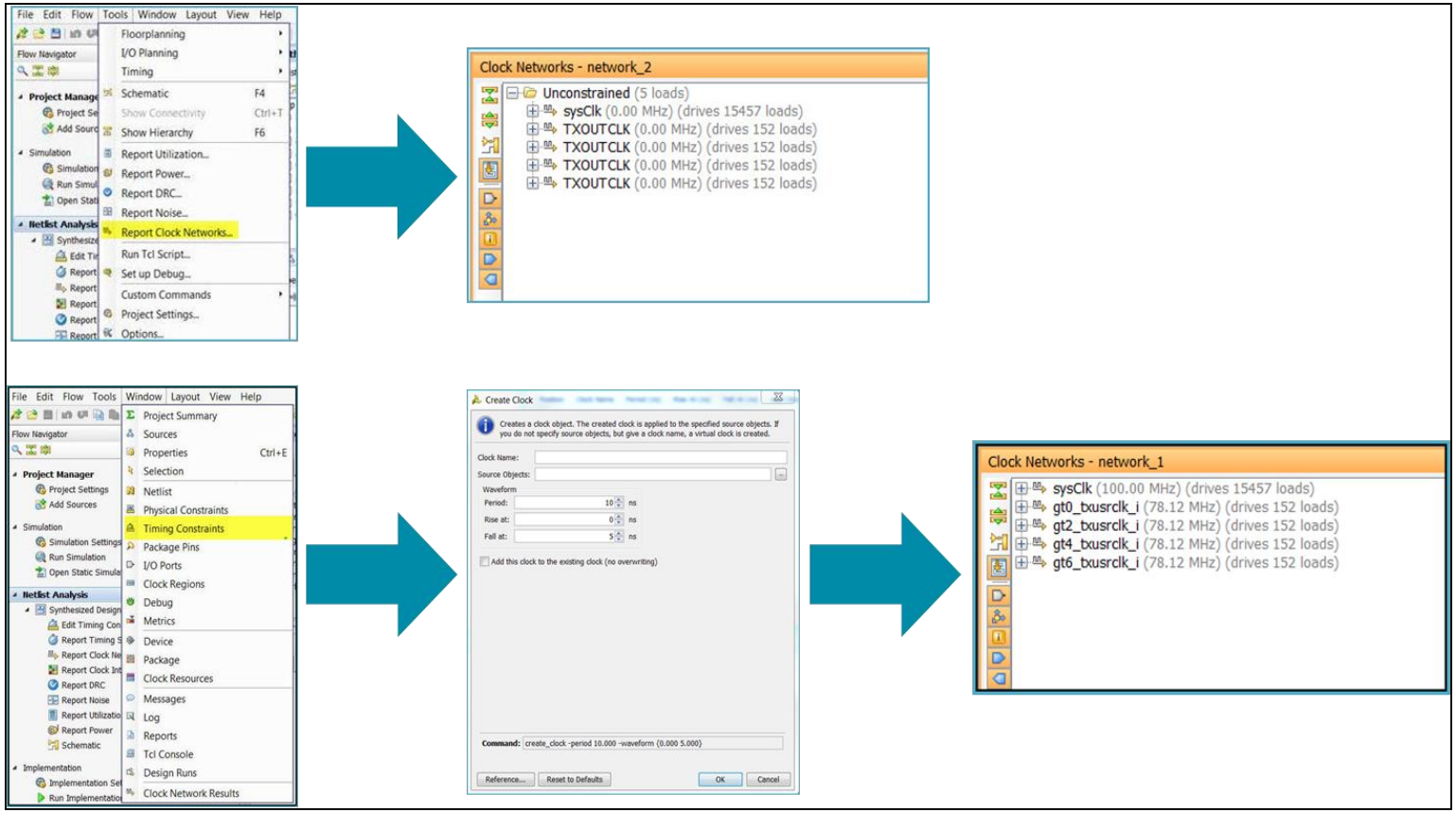


- Q. 如何了解要进行什么约束？
  - A. report_clock_networks



阶段1 从头进行约束开发的过程

- Q. 如何知道要进行哪些时钟的约束？
  - A. 直到report_clock_networks 显示no unconstrained networks时



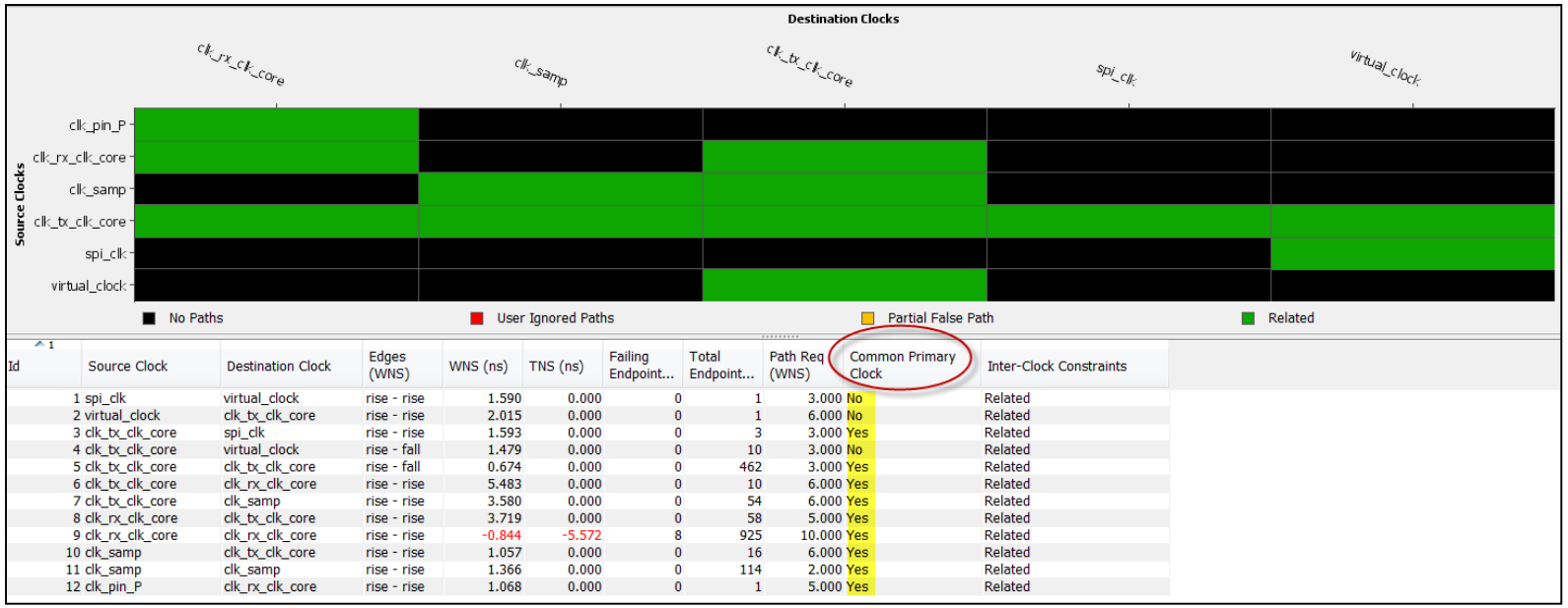
阶段1 从头进行约束开发的过程

- Q. 如何确保时钟是正确的？
  - A. report_clocks – 显示设计中所有时钟周期和波形

```
****************************************************************************************
* Report  : Clocks
* Design  : top
* Part    : Device=7k70t, Package=fbg676, Speed=-2
* Version : Vivado v2012.3 (64-bit) Build 209282 by xbuild on Thu Oct 18 20:50:53 MDT 2012
* Date    : Thu Dec 06 10:10:19 2012
****************************************************************************************


Attributes
  P: Propagated
  G: Generated
  V: Virtual
  I: Inverted

Clock           Period    Waveform            Attributes  Sources
sysClk          10.00000  {0.00000 5.00000}   P           {sysClk}
gt0_txusrclk_i  12.80000  {0.00000 6.40000}   P           {mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt0_ROCKETIO_WRAPPER_TILE_i/gtxe2_i/TXOUTCLK}
gt2_txusrclk_i  12.80000  {0.00000 6.40000}   P           {mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt2_ROCKETIO_WRAPPER_TILE_i/gtxe2_i/TXOUTCLK}
gt4_txusrclk_i  12.80000  {0.00000 6.40000}   P           {mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt4_ROCKETIO_WRAPPER_TILE_i/gtxe2_i/TXOUTCLK}
gt6_txusrclk_i  12.80000  {0.00000 6.40000}   P           {mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt6_ROCKETIO_WRAPPER_TILE_i/gtxe2_i/TXOUTCLK}
clkfbout        10.00000  {0.00000 5.00000}   P,G         {clkgen/mmcm_adv_inst/CLKFBOUT}
cpuClk          20.00000  {0.00000 10.00000}  P,G         {clkgen/mmcm_adv_inst/CLKOUT0}
wbClk           20.00000  {0.00000 10.00000}  P,G         {clkgen/mmcm_adv_inst/CLKOUT1}
usbClk          10.00000  {0.00000 5.00000}   P,G         {clkgen/mmcm_adv_inst/CLKOUT2}
phyClk0         10.00000  {0.00000 5.00000}   P,G         {clkgen/mmcm_adv_inst/CLKOUT3}
phyClk1         10.00000  {0.00000 5.00000}   P,G         {clkgen/mmcm_adv_inst/CLKOUT4}
fftClk          10.00000  {0.00000 5.00000}   P,G         {clkgen/mmcm_adv_inst/CLKOUT5}
```

阶段1 从头进行约束开发的过程

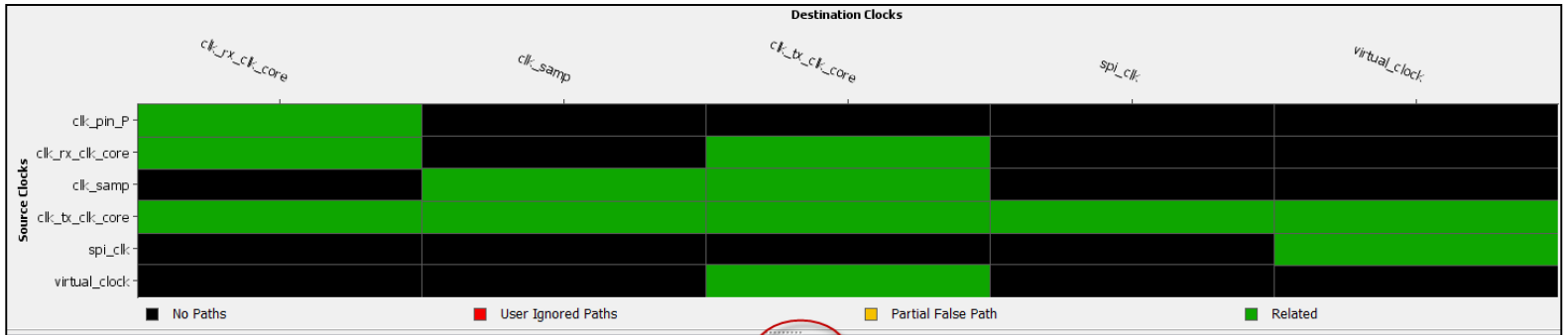
- Q. 如何确定哪些时钟必须明确相关关系？
  - A. report_clock_interaction – sort by Common Primary Clock



Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoint...	Total Endpoint...	Path Req (WNS)	Common Primary Clock	Inter-Clock Constraints
1	spi_clk	virtual_clock	rise - rise	1.590	0.000	0	1	3.000	No	Related
2	virtual_clock	clk_tx_clk_core	rise - rise	2.015	0.000	0	1	6.000	No	Related
3	clk_tx_clk_core	spi_clk	rise - rise	1.593	0.000	0	3	3.000	Yes	Related
4	clk_tx_clk_core	virtual_clock	rise - fall	1.479	0.000	0	10	3.000	No	Related
5	clk_tx_clk_core	clk_tx_clk_core	rise - fall	0.674	0.000	0	462	3.000	Yes	Related
6	clk_tx_clk_core	clk_rx_clk_core	rise - rise	5.483	0.000	0	10	6.000	Yes	Related
7	clk_tx_clk_core	clk_samp	rise - rise	3.580	0.000	0	54	6.000	Yes	Related
8	clk_rx_clk_core	clk_tx_clk_core	rise - rise	3.719	0.000	0	58	5.000	Yes	Related
9	clk_rx_clk_core	clk_rx_clk_core	rise - rise	-0.844	-5.572	8	925	10.000	Yes	Related
10	clk_samp	clk_tx_clk_core	rise - rise	1.057	0.000	0	16	6.000	Yes	Related
11	clk_samp	clk_samp	rise - rise	1.366	0.000	0	114	2.000	Yes	Related
12	clk_pin_P	clk_rx_clk_core	rise - rise	1.068	0.000	0	1	5.000	Yes	Related

阶段1 从头进行约束开发的过程

- 如何确定约束中有不切实际的路径的要求?
  - A. report_clock_interaction – sort by Path Req (WNS)



Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoint...	Total Endpoint...	Path Req (WNS)	Common Primary Clock	Inter-Clock Constraints
11	clk_samp	clk_samp	rise - rise	1.366	0.000	0	114	2.000	Yes	Related
1	spi_clk	virtual_clock	rise - rise	1.590	0.000	0	1	3.000	No	Related
4	clk_tx_clk_core	virtual_clock	rise - fall	1.479	0.000	0	10	3.000	No	Related
3	clk_tx_clk_core	spi_clk	rise - rise	1.593	0.000	0	3	3.000	Yes	Related
5	clk_tx_clk_core	clk_tx_clk_core	rise - fall	0.674	0.000	0	462	3.000	Yes	Related
8	clk_rx_clk_core	clk_tx_clk_core	rise - rise	3.719	0.000	0	58	5.000	Yes	Related
12	clk_pin_P	clk_rx_clk_core	rise - rise	1.068	0.000	0	1	5.000	Yes	Related
7	clk_tx_clk_core	clk_samp	rise - rise	3.580	0.000	0	54	6.000	Yes	Related
6	clk_tx_clk_core	clk_rx_clk_core	rise - rise	5.483	0.000	0	10	6.000	Yes	Related
2	virtual_clock	clk_tx_clk_core	rise - rise	2.015	0.000	0	1	6.000	No	Related
10	clk_samp	clk_tx_clk_core	rise - rise	1.057	0.000	0	16	6.000	Yes	Related
9	clk_rx_clk_core	clk_rx_clk_core	rise - rise	-0.844	-5.572	8	925	10.000	Yes	Related

阶段1 Baselining时序约束开发技巧

- 创建一个最小但充分的时钟约束
  - 大部分时序问题是内部寄存器间的问题 (register-to-register)
  - 该阶段不要考虑IO时序问题
  - 在baselining阶段是不需要添加IO时序约束
  - 针对跨时钟域CDC路径使用适当的例外时序约束 (set_false_path, set_max_delay –datapath_only, set_clock_groups)

阶段1 Baselining时序约束开发技巧

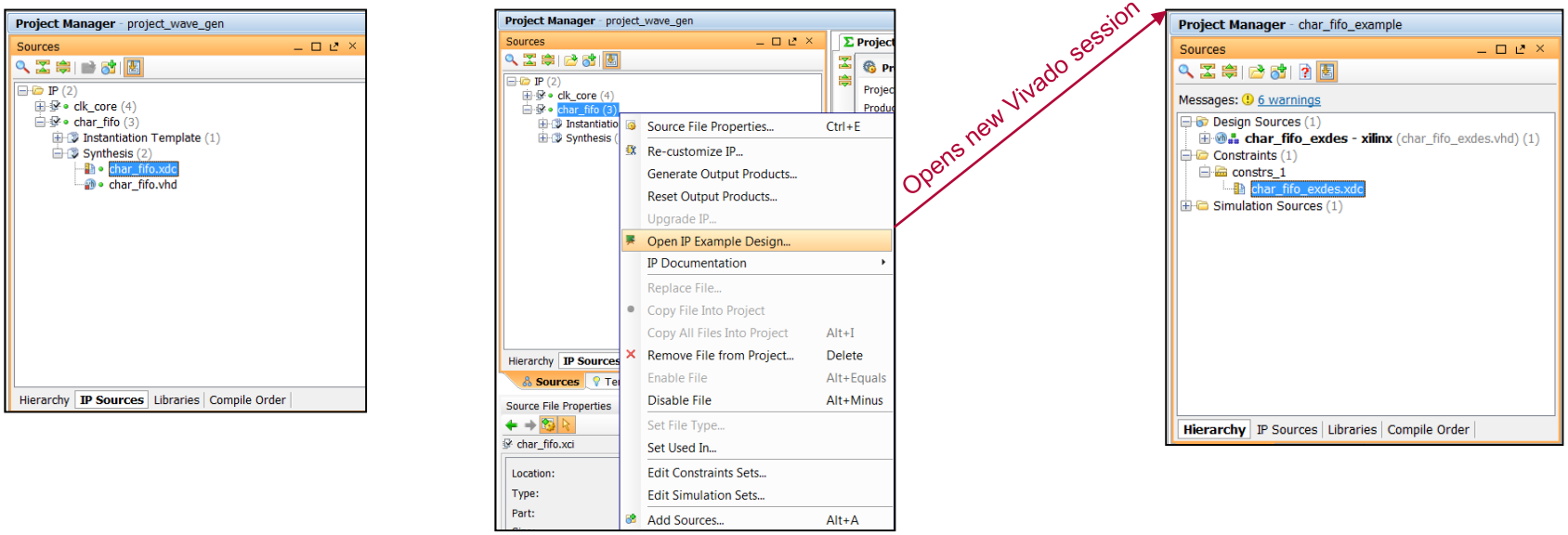
- 不要忘记包含IP的时序约束
  - 很多IP核都包含一些他们自己很重要的例外时序路径的约束 (PCIE, MIG, 2-clock distributed FIFOs…)
  - 非Vivado自带的IP，特别是只提供网表的客户IP非常容易遗漏
  - Vivado自带的，使用命令report_compile_order –constraints 来识别所有约束的源文件

Tcl Console

```
report_compile_order -constraints


Synthesis Constraint Evaluation Order (sources_1 & constrs_1)
Index  File Name            Used_In       Scoped_To_Ref  Scoped_To_Cells  Processing_Order  Full Path Name
-----  -------------------  ------------  -------------  ---------------  ----------------  ----------------------------------------------------------------------------------------------------
1      clk_core.xdc         Synth & Impl  clk_core       inst             EARLY             c:/projects/project_wave_gen/project_wave_gen.srcs/sources_1/ip/clk_core/clk_core.xdc
2      wave_gen_timing.xdc  Synth & Impl                                  NORMAL            C:/projects/project_wave_gen/project_wave_gen.srcs/constrs_1/imports/verilog/wave_gen_timing.xdc
3      char_fifo.xdc        Synth & Impl  char_fifo      U0               NORMAL            c:/projects/project_wave_gen/project_wave_gen.srcs/sources_1/ip/char_fifo/char_fifo/char_fifo.xdc


Implementation Evaluation Compile Order (sources_1 & constrs_1)
Index  File Name            Used_In       Scoped_To_Ref  Scoped_To_Cells  Processing_Order  Full Path Name
-----  -------------------  ------------  -------------  ---------------  ----------------  ----------------------------------------------------------------------------------------------------
1      clk_core.xdc         Synth & Impl  clk_core       inst             EARLY             c:/projects/project_wave_gen/project_wave_gen.srcs/sources_1/ip/clk_core/clk_core.xdc
2      wave_gen_timing.xdc  Synth & Impl                                  NORMAL            C:/projects/project_wave_gen/project_wave_gen.srcs/constrs_1/imports/verilog/wave_gen_timing.xdc
3      wave_gen_pins.xdc    Impl                                          NORMAL            C:/projects/project_wave_gen/project_wave_gen.srcs/constrs_1/imports/verilog/wave_gen_pins.xdc
4      char_fifo.xdc        Synth & Impl  char_fifo      U0               NORMAL            c:/projects/project_wave_gen/project_wave_gen.srcs/sources_1/ip/char_fifo/char_fifo/char_fifo.xdc
```

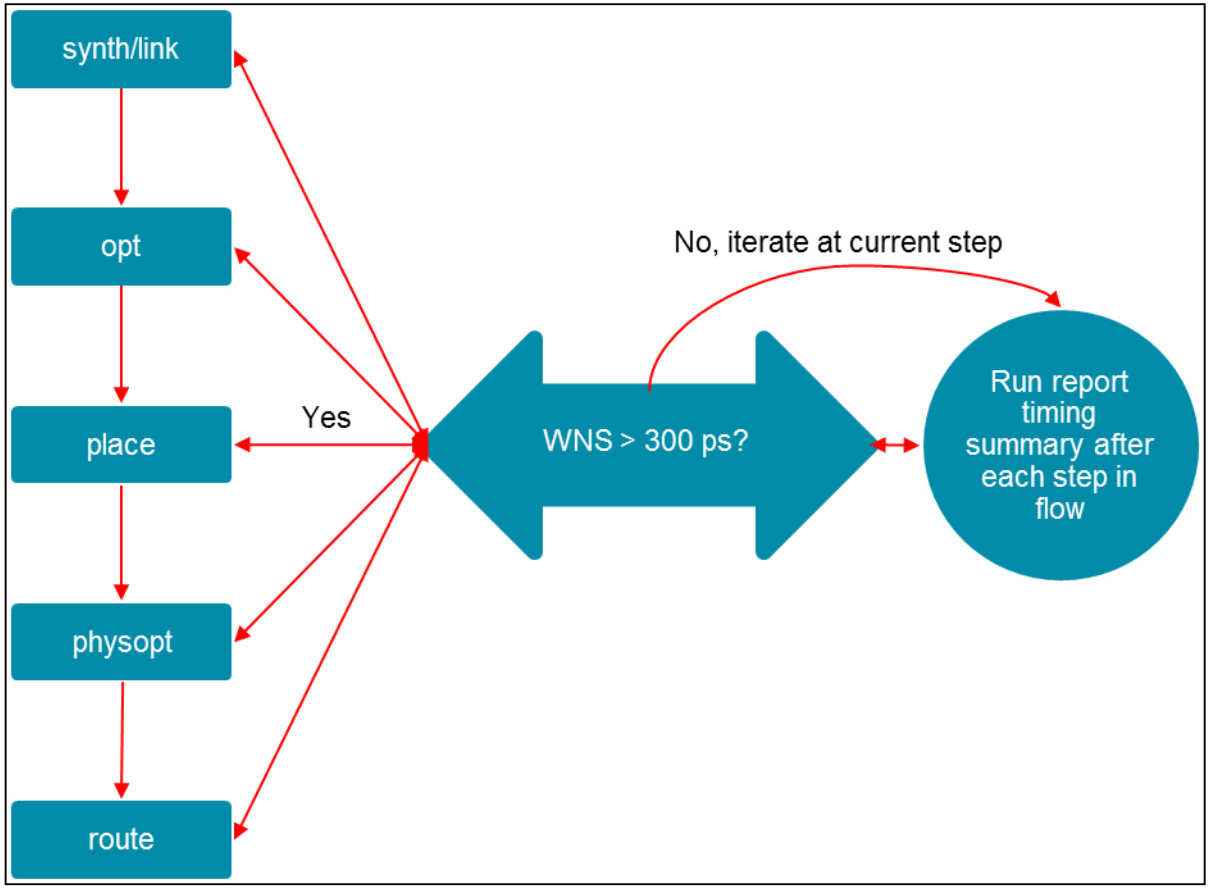
阶段1 Baselining时序约束开发技巧

- 不要忘记包含IP的时序约束
  - Vivado自带的IP:，重新确认IP核自带的XDC约束以及example project中时序例外的XDC文件



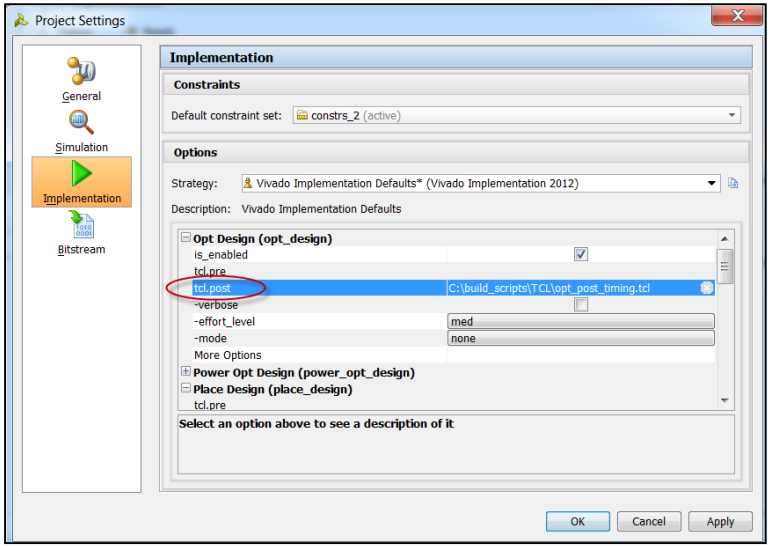
Baseline 阶段2

- 每个阶段都运行 report_timing_summary
- 确保WNS < 300 ps



Setting Up report_timing_summary

- GUI



opt_post_timing.tcl file:
report_timing_summary -file opt_timing.rpt

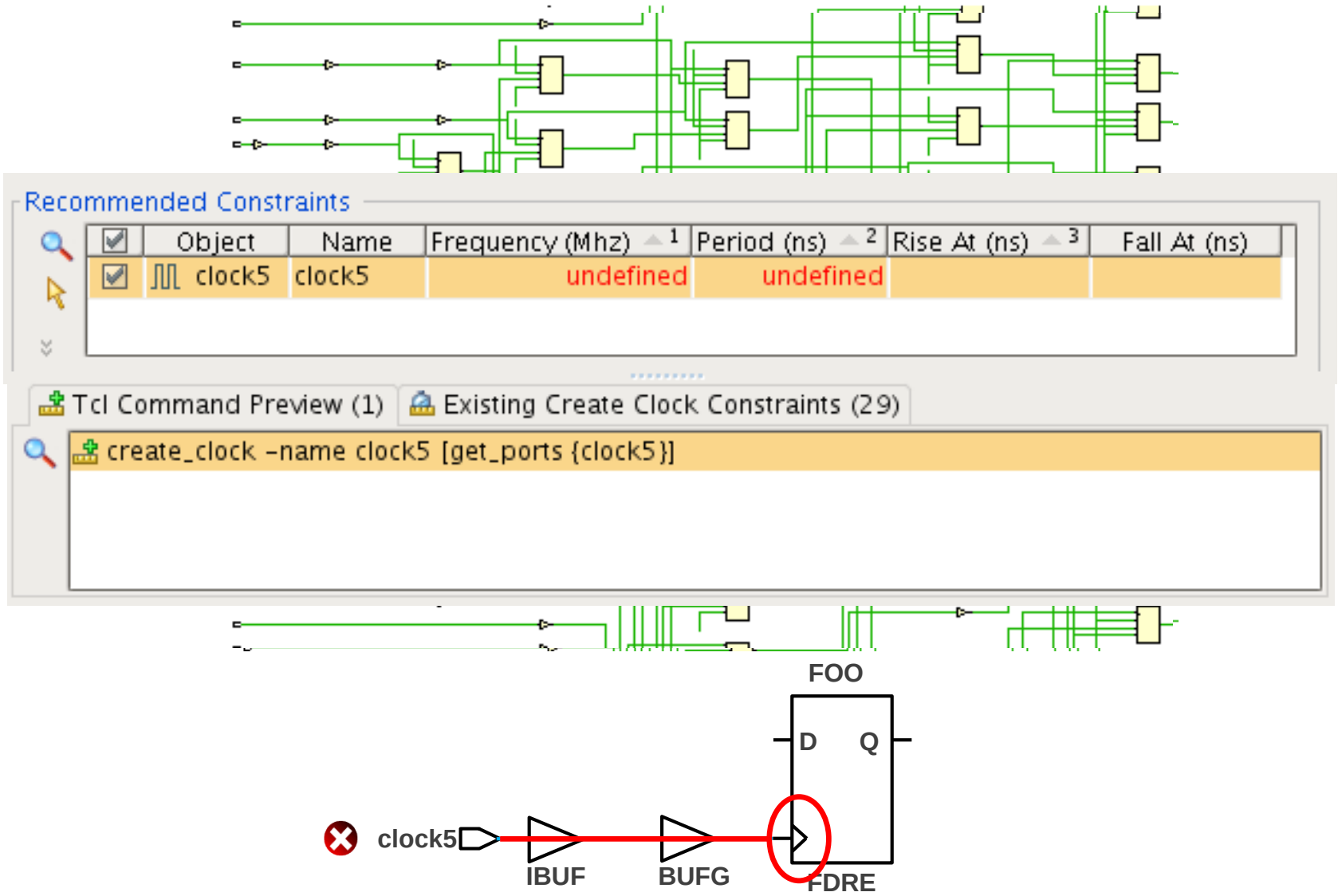
- Batch

Build.tcl file:

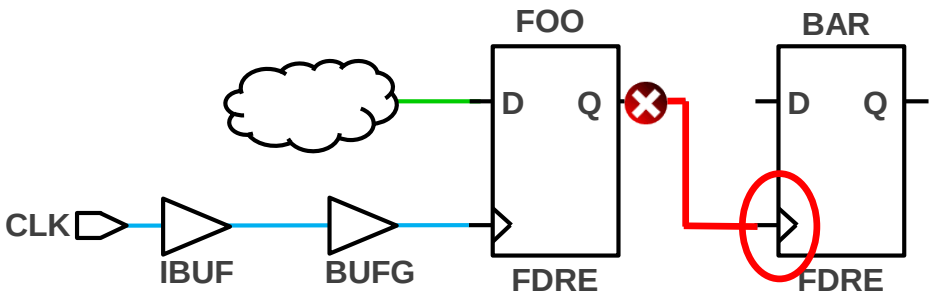
```
link_design -name top -part xc7vx1140tflg1928-2
read_xdc top.xdc

opt_design
report_timing_summary -file opt_timing.rpt
write_checkpoint -force opt.dcp

place_design
report_timing_summary -file place_timing.rpt
write_checkpoint -force place.dcp

phys_opt_design
report_timing_summary -file popt_timing.rpt
write_checkpoint -force popt.dcp

route_design
report_timing_summary –file routed_timing.rpt
write_checkpoint –force routed.dcp
```

TCW基本使用步驟

Missing Primary Clocks



Generated Clocks in the Fabric



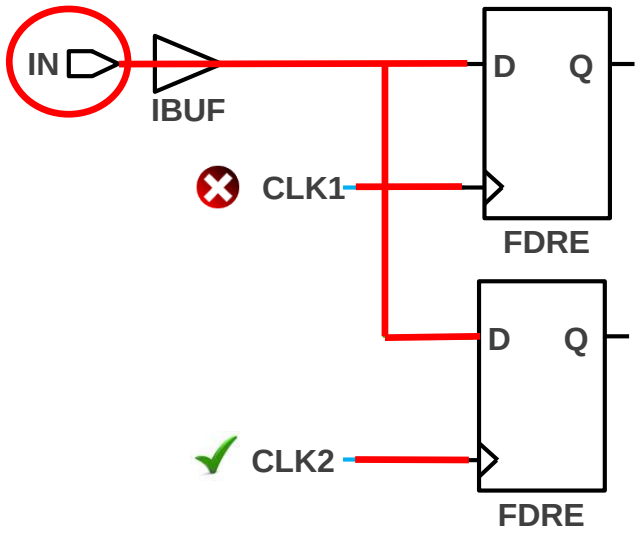
Recommended Constraints

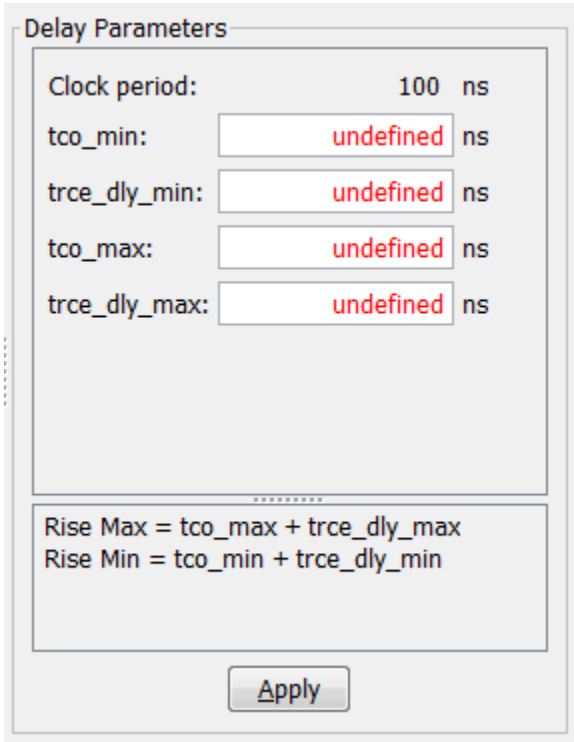
Port/Pin	Generated Clock Name	Source Clock	Divide By
divcl...reg/Q	divclk/CLK	clkdiv	

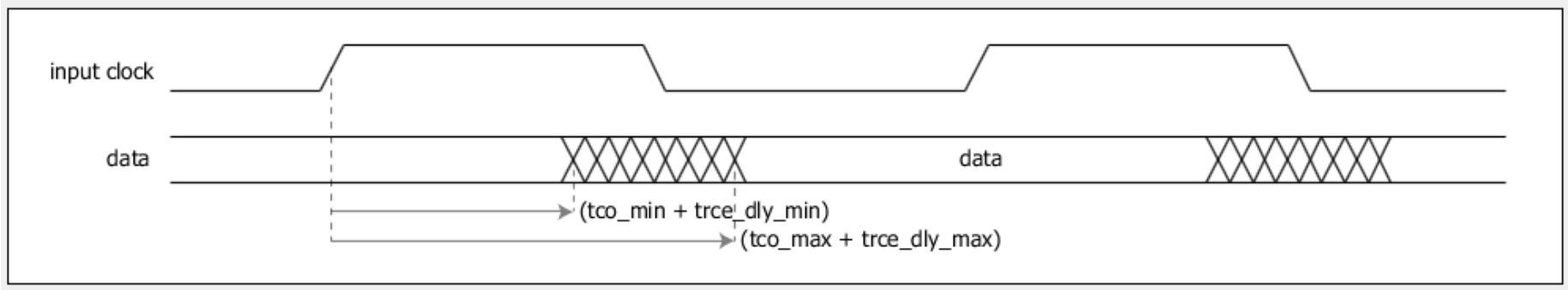
Tcl Command Preview (1) | Existing Create Generated Clock Constraints (0)

```
create_generated_clock -name divclk/CLK -source [get_ports {clkdiv}] [get_pins {divclk/FDIV_reg/Q}]
```

Input Constraints



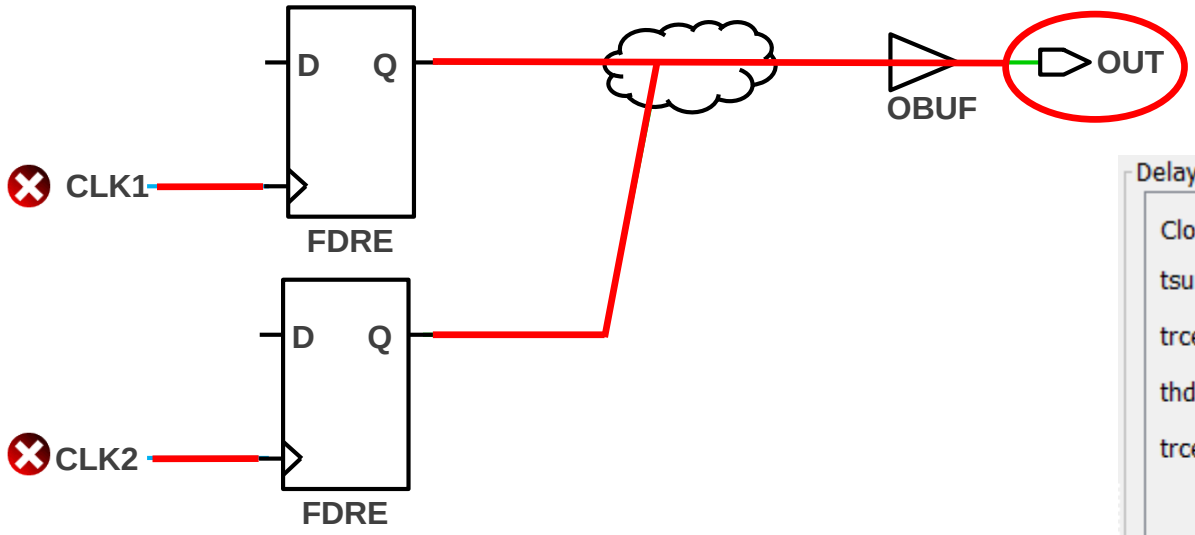


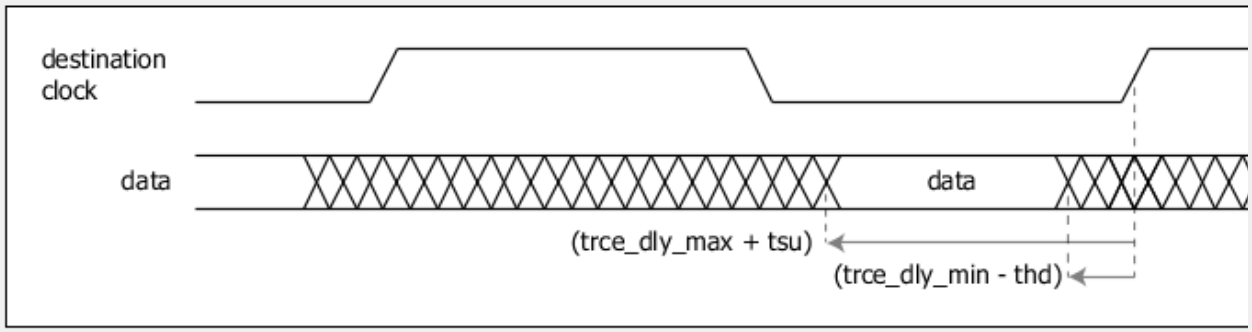


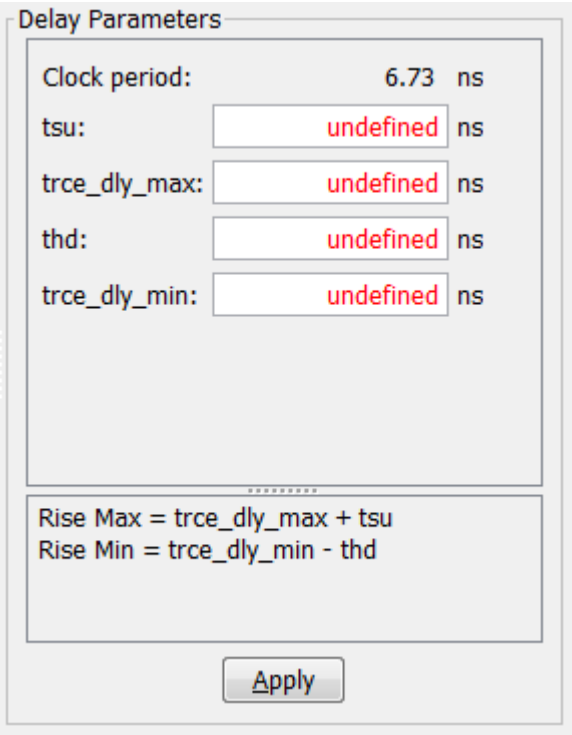
set_input_delay –clock [get_clocks {CLK1}] –max –add_delay 0.1 [get_ports {IN}]

set_input_delay –clock [get_clocks {CLK1}] –min –add_delay 0.1 [get_ports {IN}]

Output Constraints







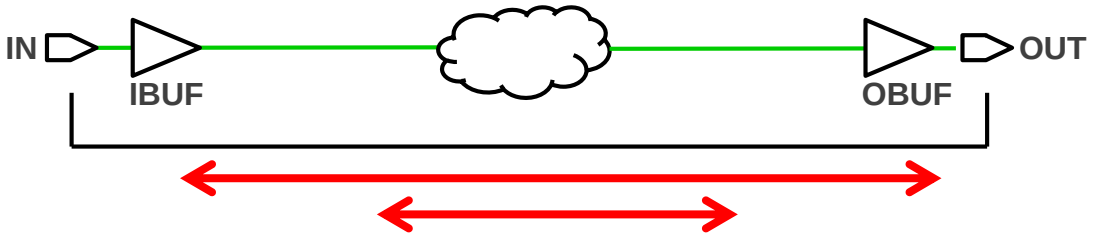
set_output_delay –clock [get_clocks {CLK1}] –max –add_delay 0.2 [get_ports {OUT}]

set_output_delay –clock [get_clocks {CLK2}] –max –add_delay 0.5 [get_ports {OUT}]

set_output_delay –clock [get_clocks {CLK1}] –min –add_delay 0.1 [get_ports {OUT}]

set_output_delay –clock [get_clocks {CLK2}] –min –add_delay 0.1 [get_ports {OUT}]

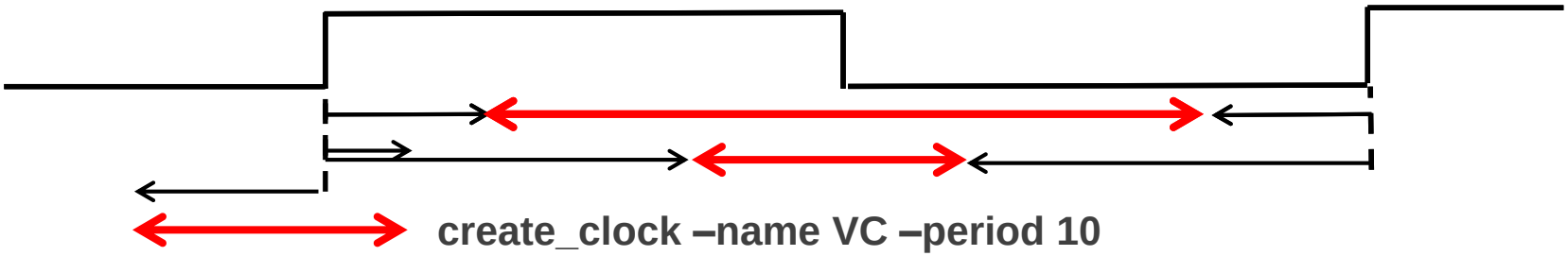
Combinational Delays



slowest set_max_delay 5.0 –from [get_ports {IN}] –to [get_ports{OUT}]

fastest set_min_delay 2.0 –from [get_ports {IN}] –to [get_ports {OUT}]

Virtual Clock



create_clock –name VC –period 10

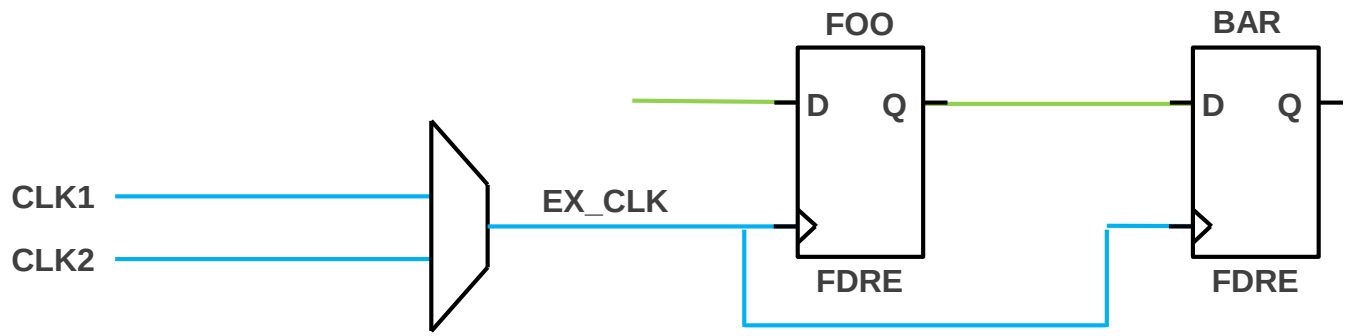
set_input_delay –clock [get_clocks {VC}] –max 2.5

set_output_delay –clock [get_clocks {VC}] –max 2.5

set_input_delay –clock [get_clocks {VC}] –min 1.0

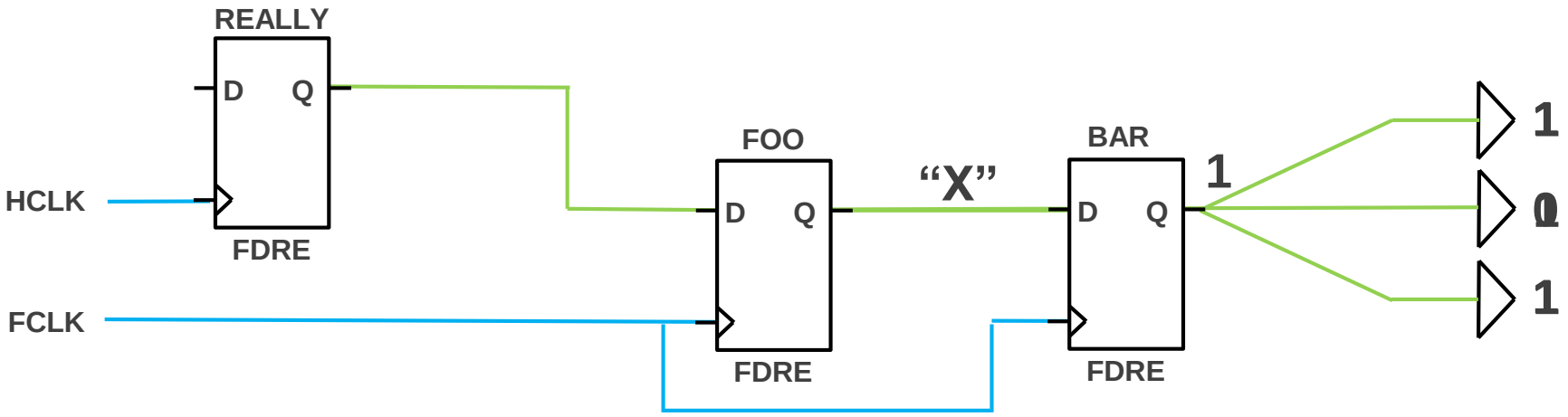
set_output_delay –clock [get_clocks {VC}] –min -3.0

Exclusive Clocks



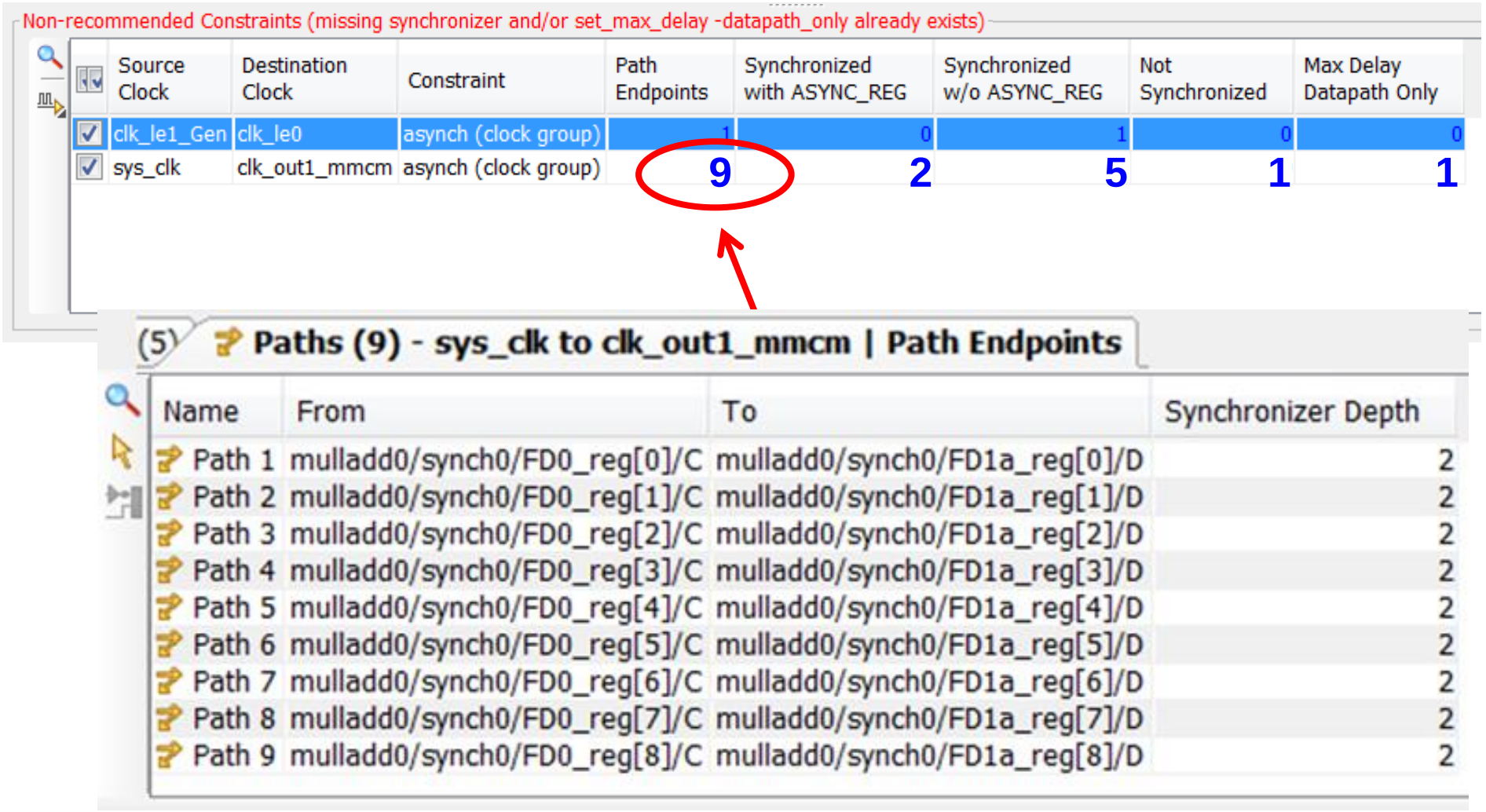
set_clock_groups –logically_exclusive –group [get_clocks –include_generated_clocks {CLK1}]
–group [get_clocks -include_generated_clocks {CLK2}]

Asynchronous Clock Domain Crossings - Synchronizers



- **HCLK and FCLK have no common primary clock**
- **Flop Foo may violate setup / hold requirements**
- **Flop Bar will resolve any meta-stability in the receiving clock domain**
- **Wizard assumes stable after two synchronizing flops**
- **To improve MTBF - FOO and BAR should be placed close together**
  - ASYNC_REG property achieves this

Wizard Analysis of Clock Domain Crossings



Overview

- 时序约束的作用及方法
- 时序约束的三大类别介绍
- 高级时序约束介绍
- Baselining时序约束及TCW使用
- 时序报告分析
- Lab2

时序命令与报告

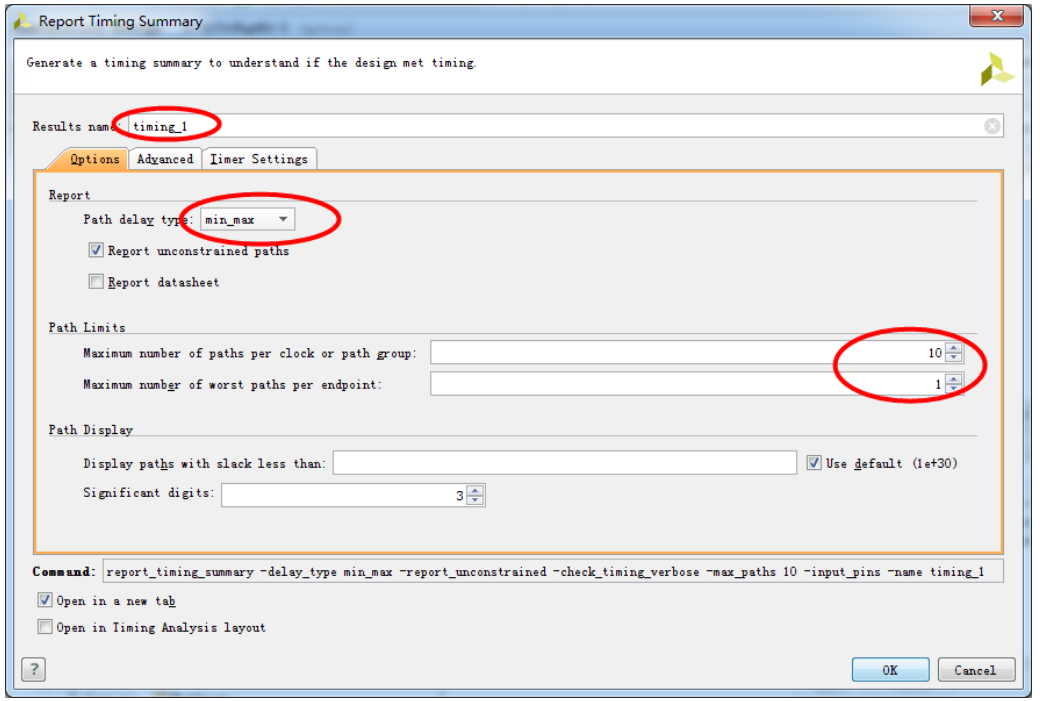
- **report_timing_summary**
  - 主要用于实现后的timing sigh-off
  - 不仅在布局布线后，在综合后甚至是更具体的实现过程中的每一小步之后都可以运行，从而得到一个全局的时序报告。
- **report_timing**
  - 主要用于交互式的约束验证以及更细致具体的时序报告与分析
  - 用户可以在设计的任何阶段使用report_timing，甚至是一边设置XDC，一边用其来验证约束的可行性与优先级

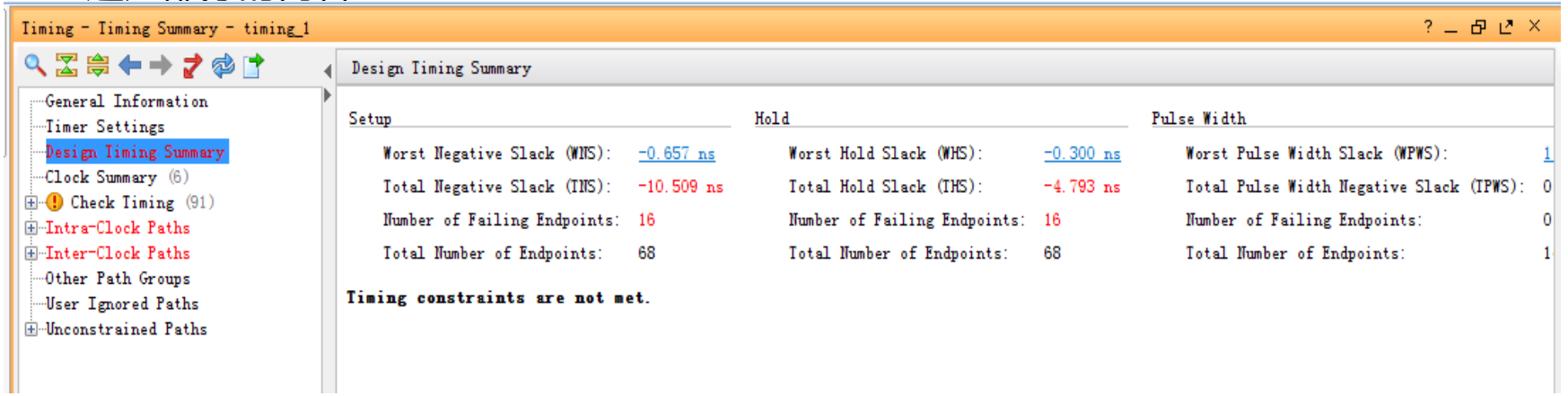
时序报告的术语

- **Setup**
  - Worst Negative Slack (WNS)
    - Path with worst setup/recovery slack
  - Total Negative Slack (TNS)
    - Sum of setup/recovery violations for the entire design
- **Hold**
  - Worst Hold Slack (WHS)
    - Path with worst hold/removal slack
  - Total Hold Slack (THS)
    - Sum of hold/removal violations for the entire design

时序命令与报告

- **report_timing_summary**
  - Vivado IDE 中点击Report Timing Summary后可以改变报告的内容，例如每个时钟域报告的路径条数，是否setup 和hold 全都报告等等。
  - report_timing_summary 实际上隐含了report_timing、report_clocks 、check_timing 以及部分的report_clock_interaction 命令，所以我们最终看到的报告中也包含了这几部分的内容





时序命令与报告

report_timing_summary

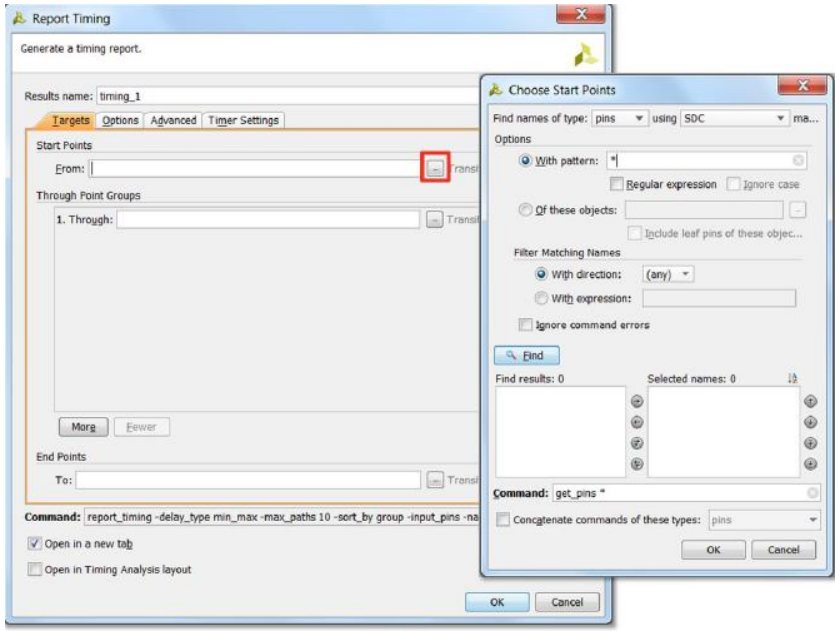
- 每条路径具体的报告会分为Summary、Source Clock Path、Data Path 和Destination Clock Path 几部分，详细报告每部分的逻辑延时与连线延时
- Slack 显示路径是否有时序违例，Source 和Destination 显示源驱动时钟和目的驱动时钟及其时钟频率， Requirement 显示这条路径的时序要求是多少，Data Path 显示数据路径上的延时，Logic Level 显示这条路径的逻辑级数，而Clock Path Skew 和 Clock Uncertainty 则显示时钟路径上的不确定性
- 这是一条clk 时钟域内的路径，时钟周期为3.125ns，这条路径有0.268ns 的时序违例。违例的主要原因是逻辑级数较高导致的数据链路延时较大，但连线延时的比例也较高，所以可以仔细看看这条路径的数据路径上有没有可能改进布局、降低扇出或者是减少逻辑级数的优化方向。

Summary

Name	Path 1
Slack	-0.657ns
Source	tmp_a_reg[3]/C (rising edge-triggered cell FDRE clocked by clk_out1_mmcm_clk1 {rise@0.000ns fall@2.000ns period=4.000ns})
Destination	cd1_xor_reg[0]/D (rising edge-triggered cell FDRE clocked by clk_out1_mmcm_clk1 {rise@0.000ns fall@2.000ns period=4.000ns})
Path Group	clk_out1_mmcm_clk1
Path Type	Setup (Max at Slow Process Corner)
Requirement	4.000ns (clk_out1_mmcm_clk1 rise@4.000ns - clk_out1_mmcm_clk1 rise@0.000ns)
Data Path Delay	4.484ns (logic 3.146ns (70.158%) route 1.338ns (29.842%))
Logic Levels	4 (CARRY4=2 DSP48E1=1 LUT2=1)
Clock Path Skew	-0.145ns
Clock Uncertainty	0.063ns

时序命令与报告

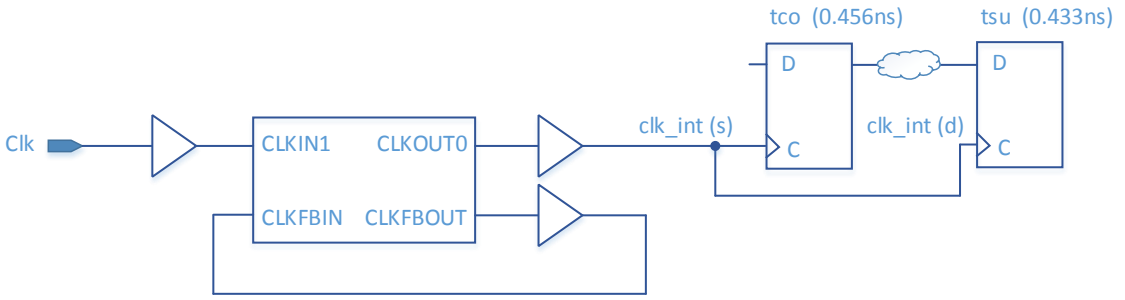
- **report_timing**
  - 更具体的时序报告命令，经常用来报告某一条或是某些共享特定节点的路径
  - Vivado IDE 中可由Tools > Timing > Report Timing 调出其图形化设置窗口
  - 在Targets 一栏
  - 还可以设置需要报告路径的起始点/途经点/结束点，可以三个都设置或是仅设置其中任何一项，每一项都支持通配符匹配甚至是正则表达式查找
  - 仅输入report_timing 而不加任何option，Vivado 便会报告出时序违例最严重的那条路径，方便我们快速了解当前设计的WNS，找到最差的那条路径。
  - 在验证I/O 约束时也常常用到report_timing，只要指定某个输入-from或是-to 某个输出便可以快速验证当前设计在接口上的时序。



时序报告- Summary

- Path Name
- Slack
- Source
- Destination
- Path Type
- Requirement
- Data Path Delay
- Logic Levels
- Clock Path Skew
- Clock Uncertainty

Summary	
Name	Path 7
Slack	6.533ns
Source	UART_RX_I/over_sample_cnt_reg[0]/C (rising edge-triggered cell FDSE clocked by clk_out1_cl
Destination	UART_RX_I/bit_cnt_reg[0]/R (rising edge-triggered cell FDRE clocked by clk_out1_clock {rise@
Path Group	clk_out1_clock
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (clk_out1_clock rise@10.000ns - clk_out1_clock rise@0.000ns)
Data Path Delay	2.920ns (logic 0.869ns (29.760%) route 2.051ns (70.240%))
Logic Levels	2 (LUT4=1 LUT5=1)
Clock Path Skew	-0.040ns
Clock Uncertainty	0.074ns



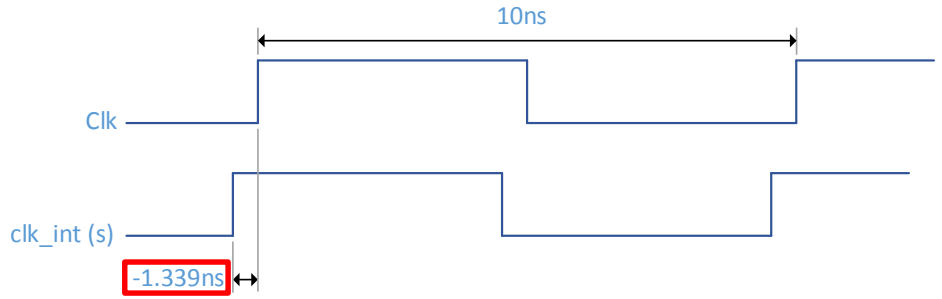
HWALT

时序报告- Source Clock

- Delay from the Clk input to source Clock input

Source Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_out1_clock rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: E3	Clk
net (fo=0)	0.000	0.000		Clk
			Site: E3	IBUFG_I/I
IBUF (Prop_ibuf_I_O)	(r) 1.489	1.489	Site: E3	IBUFG_I/O
net (fo=3, unplaced)	0.800	2.289		CLOCK_TRUE_GEN.CLOCK_I/clk_in1
				CLOCK_TRUE_GEN.CLOCK_I/MMCM_ADV_I/CLKIN1
MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)	(r) -5.324	-3.035		CLOCK_TRUE_GEN.CLOCK_I/MMCM_ADV_I/CLKOUT0
net (fo=1, unplaced)	0.800	-2.235		CLOCK_TRUE_GEN.CLOCK_I/clk_out1_clock
				CLOCK_TRUE_GEN.CLOCK_I/CLKOUT1_BUFG_I/I
BUFG (Prop_bufg_I_O)	(r) 0.096	-2.139		CLOCK_TRUE_GEN.CLOCK_I/CLKOUT1_BUFG_I/O
net (fo=376, unplaced)	0.800	-1.339		UART_RX_I/clk_out1
FDSE				UART_RX_I/over_sample_cnt_reg[0]/C

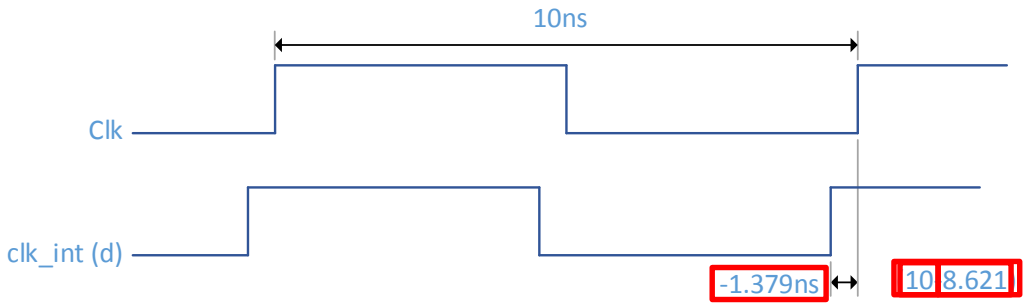


时序报告- Destination Clock

- Delay from the Clk input to destination Clock input

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_out1_clock rise edge)	(r) 10.000	10.000		
	(r) 0.000	10.000	Site: E3	Clk
net (fo=0)	0.000	10.000		Clk
IBUF			Site: E3	IBUFG_I/I
IBUF (Prop_ibuf_I_O)	(r) 1.418	11.418	Site: E3	IBUFG_I/O
net (fo=3, unplaced)	0.760	12.178		CLOCK_TRUE_GEN.CLOCK_I/clk_in1
MMCME2_ADV				CLOCK_TRUE_GEN.CLOCK_I/MMCM_ADV_I/CLKIN1
MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)	(r) -5.699	6.479		CLOCK_TRUE_GEN.CLOCK_I/MMCM_ADV_I/CLKOUT0
net (fo=1, unplaced)	0.760	7.239		CLOCK_TRUE_GEN.CLOCK_I/clk_out1_clock
BUFG				CLOCK_TRUE_GEN.CLOCK_I/CLKOUT1_BUFG_I/I
BUFG (Prop_bufg_I_O)	(r) 0.091	7.330		CLOCK_TRUE_GEN.CLOCK_I/CLKOUT1_BUFG_I/O
net (fo=376, unplaced)	0.760	8.090		UART_RX_I/clk_out1
FDRE				UART_RX_I/bit_cnt_reg[0]/C
clock pessimism	0.531	8.621		
clock uncertainty	-0.074	8.547		
FDRE (Setup_fdre_C_R)	-0.433	8.114		UART_RX_I/bit_cnt_reg[0]
Required Time		8.114		

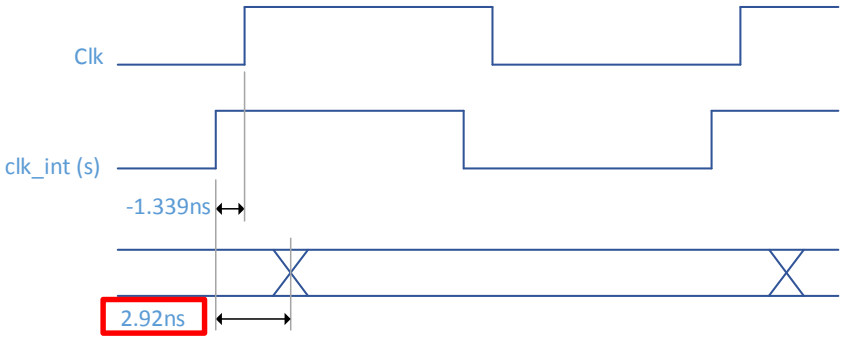


时序报告- Data Path

- Delay from source FF to destination FF input

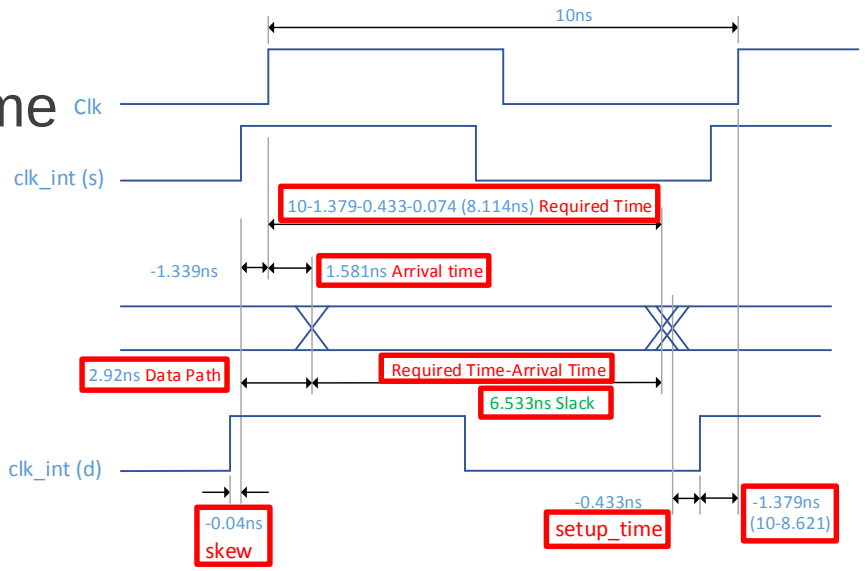
Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDSE (Prop_fdse_C_Q)	(f) 0.456	-0.883		UART_RX_I/over_sample_cnt_reg[0]/Q
net (fo=6, unplaced)	0.773	-0.110		UART_RX_I/over_sample_cnt_reg_n_0_[0]
				UART_RX_I/state[1]_i_2/I1
LUT4 (Prop_lut4_I1_O)	(r) 0.289	0.179		UART_RX_I/state[1]_i_2/O
net (fo=5, unplaced)	0.477	0.656		UART_RX_I/state[1]_i_2_n_0
				UART_RX_I/bit_cnt[3]_i_1/I0
LUT5 (Prop_lut5_I0_O)	(r) 0.124	0.780		UART_RX_I/bit_cnt[3]_i_1/O
net (fo=4, unplaced)	0.801	1.581		UART_RX_I/bit_cnt[3]_i_1_n_0
FDRE				UART_RX_I/bit_cnt_reg[0]/R
Arrival Time		1.581		



时序报告- Slack

- Clock path skew is the difference between source and destination clocks
- Clock uncertainty reduces slack
- Arrival time is data path delay with respect to the Clk input
- Required time is the requirement - clock delay, setup time and clock uncertainty
- Slack is required time – arrival time



HWALT

Overview

- 时序约束的作用及方法
- 时序约束的三大类别介绍
- 高级时序约束介绍
- Baselining时序约束及TCW使用
- 时序报告分析
- Lab2

Lab1: Baselining with TCW & Report Design Analysis labs

Duration: 45 Minutes

6、CDC时序约束

Overview

- CDC基本概念
- 工具中分析CDC
- CDC路径结构

CDC 的定义

- CDC 是Clock Domain Crossing 的简称
- CDC 时序路径指的是起点和终点由不同时钟驱动的路径
- 在电路设计中对这些跨时钟域路径往往需要进行特别的处理来避免亚稳态的产生，例如使用简单同步器、握手电路或是FIFO来隔离。

安全和不安全CDC路径

- **安全的CDC 路径**
  - 所谓安全CDC 路径指那些源时钟和目标时钟拥有相同的来源，在FPGA 内部共享部分时钟网络的时序路径。
  - 这里的安全指的是时钟之间的关系对Vivado®来说是全透明可分析的。
- **不明确安全的CDC路径**
  - 不安全的CDC路径则表示源时钟和目标时钟不同，且由不同的端口进入FPGA，在芯片内部不共享时钟网络
  - 这种情况下，Vivado的报告也只是基于端口处创建的主时钟,在约束文件中所描述的相位和频率关系来分析，并不能代表时钟之间真实的关系。

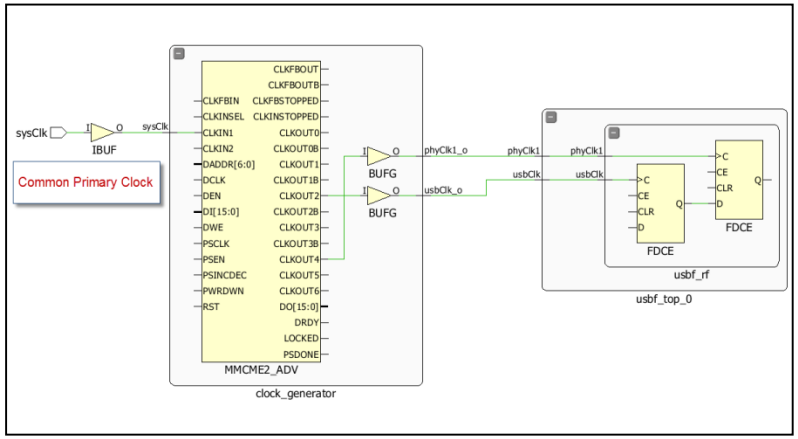


Figure 1: Safe CDC

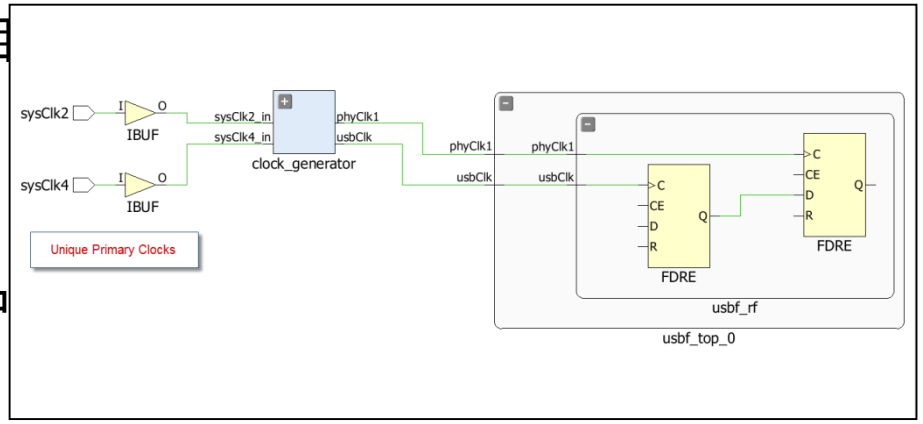


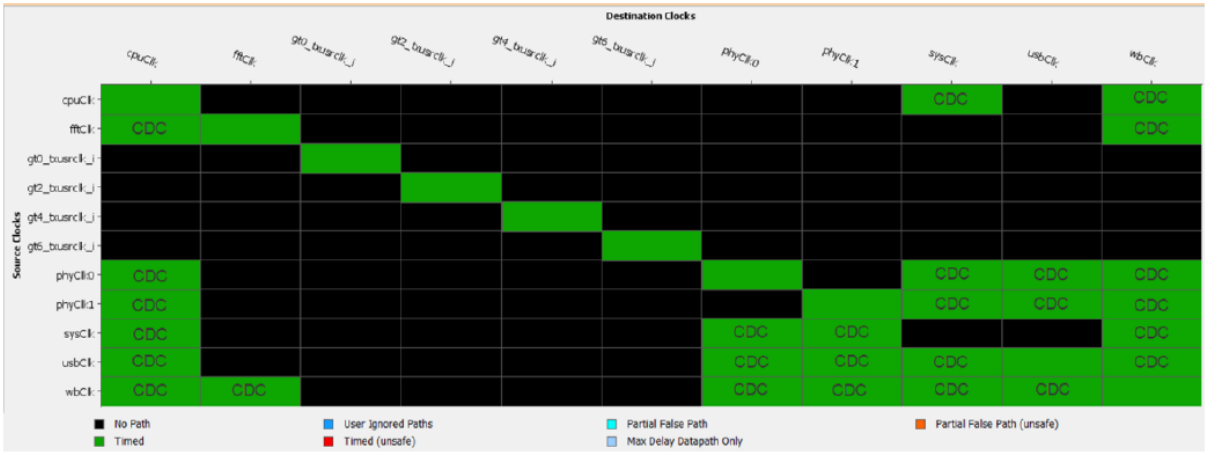
Figure 2: Unknown CDC

Overview

- CDC基本概念
- 工具分析CDC
- CDC路径结构

在Vivado中分析CDC

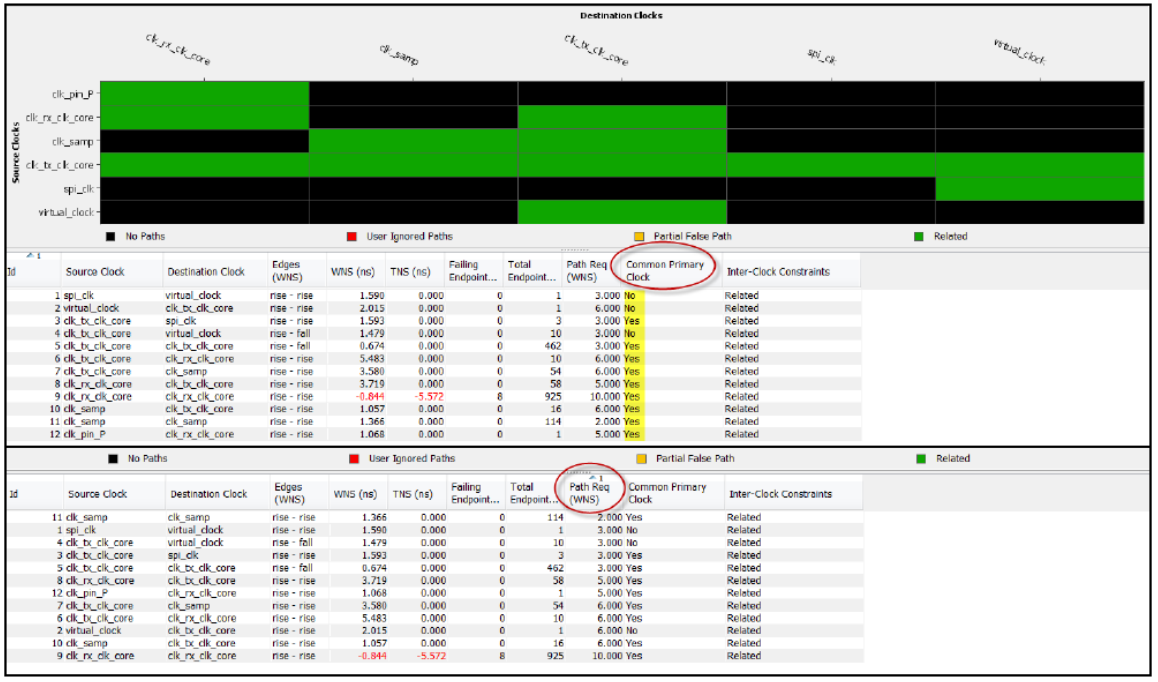
- **使用report_clock_interaction命令（GUI支持）**
  - 鉴别和报告设计中所有的时钟关系
  - 执行命令后会生成一个矩阵图，其中对角线上的路径表示源时钟与目标时钟相同的时钟内部路径，其余都是CDC路径recommended and commonly
- **根据网表和已读入的约束分析出CDC路径的约束情况，分颜色表示**
  - 绿色代表有时序约束
  - 红色代表不安全的CDC路径但是没有约束时序例外
  - 蓝色表示有部分路径已约束为false path的不安全CDC路径。
  - 黑色表示没有交互



HWALT

在Vivado中分析CDC

- **矩阵下方是时钟关系表格鉴别和报告设计中所有的时钟关系**
  - 对“Common Primary Clock”排序（显示为Yes 或No），可以快速鉴别出那些安全和不安全的CDC路径
  - 观察对应的“Inter-Clock Constraints”栏内的内容，判断已读入的XDC中是否对这类路径进行了合理的约束
  - 对“Path Req (WNS)”由小到大进行排序



Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoint...	Total Endpoint...	Path Req (WNS)	Common Primary Clock	Inter-Clock Constraints
1	spi_clk	virtual_clock	rise - rise	1.590	0.000	0	1	3.000	No	Related
2	virtual_clock	clk_tx_clk_core	rise - rise	2.015	0.000	0	1	6.000	No	Related
3	clk_tx_clk_core	spi_clk	rise - rise	1.593	0.000	0	3	3.000	Yes	Related
4	clk_tx_clk_core	virtual_clock	rise - fall	1.479	0.000	0	10	3.000	No	Related
5	clk_tx_clk_core	clk_tx_clk_core	rise - fall	0.674	0.000	0	462	3.000	Yes	Related
6	clk_tx_clk_core	clk_rx_clk_core	rise - rise	5.483	0.000	0	10	6.000	Yes	Related
7	clk_tx_clk_core	clk_samp	rise - rise	3.580	0.000	0	54	6.000	Yes	Related
8	clk_rx_clk_core	clk_tx_clk_core	rise - rise	3.719	0.000	0	58	5.000	Yes	Related
9	clk_rx_clk_core	clk_rx_clk_core	rise - rise	-0.844	-5.572	8	925	10.000	Yes	Related
10	clk_samp	clk_tx_clk_core	rise - rise	1.057	0.000	0	16	6.000	Yes	Related
11	clk_samp	clk_samp	rise - rise	1.366	0.000	0	114	2.000	Yes	Related
12	clk_pin_P	clk_rx_clk_core	rise - rise	1.068	0.000	0	1	5.000	Yes	Related

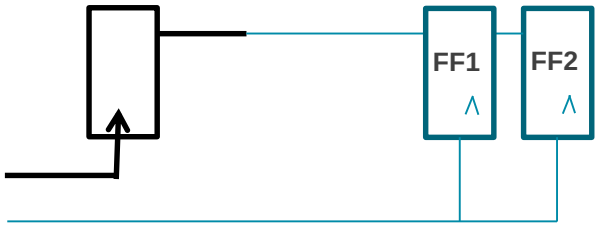
Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoint...	Total Endpoint...	Path Req (WNS)	Common Primary Clock	Inter-Clock Constraints
11	clk_samp	clk_samp	rise - rise	1.366	0.000	0	114	2.000	Yes	Related
1	spi_clk	virtual_clock	rise - rise	1.590	0.000	0	1	3.000	No	Related
4	clk_tx_clk_core	virtual_clock	rise - fall	1.479	0.000	0	10	3.000	No	Related
3	clk_tx_clk_core	spi_clk	rise - rise	1.593	0.000	0	3	3.000	Yes	Related
5	clk_tx_clk_core	clk_tx_clk_core	rise - fall	0.674	0.000	0	462	3.000	Yes	Related
8	clk_rx_clk_core	clk_tx_clk_core	rise - rise	3.719	0.000	0	58	5.000	Yes	Related
12	clk_pin_P	clk_rx_clk_core	rise - rise	1.068	0.000	0	1	5.000	Yes	Related
7	clk_tx_clk_core	clk_samp	rise - rise	3.580	0.000	0	54	6.000	Yes	Related
6	clk_tx_clk_core	clk_rx_clk_core	rise - rise	5.483	0.000	0	10	6.000	Yes	Related
2	virtual_clock	clk_tx_clk_core	rise - rise	2.015	0.000	0	1	6.000	No	Related
10	clk_samp	clk_tx_clk_core	rise - rise	1.057	0.000	0	16	6.000	Yes	Related
9	clk_rx_clk_core	clk_rx_clk_core	rise - rise	-0.844	-5.572	8	925	10.000	Yes	Related

Overview

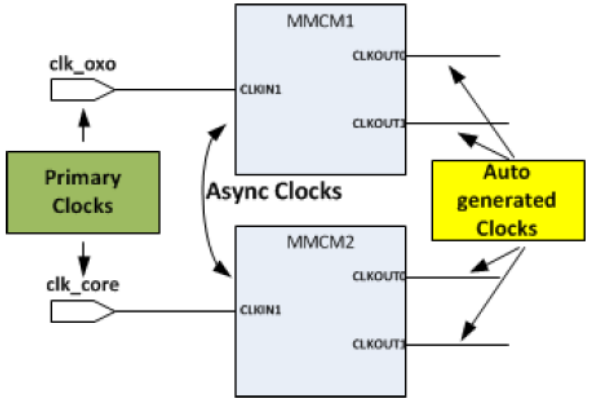
- CDC基本概念
- 工具中分析CDC
- CDC路径结构

CDC结构支持

- **单bit时，采用两级或者更多级进行同步转换**
  - Synchronizers(FF1 and FF2) have ASYNC_REG Property
  - ASYNC_REG的约束，把简单同步器的多个寄存器放入同一个SLICE，以降低走线延时
  - 在XDC中，对于此类设计的CDC路径，可以采用set_clock_groups来约束



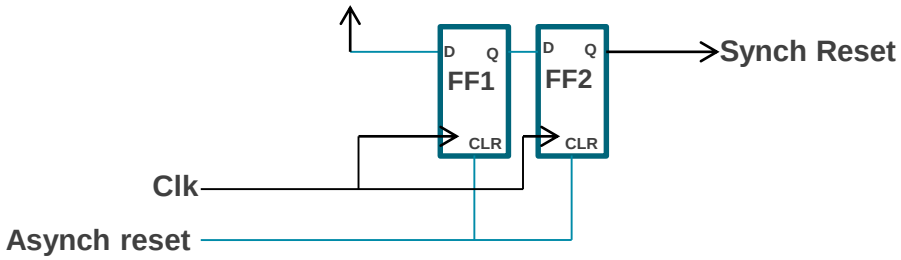
- **多bit时，采用两级或者更多级进行同步转换**
  - 用FIFO隔离CDC
  - 采用set_clock_groups来约束




```
set_clock_groups -asynchronous  -group [get_clocks  -include_generated_clocks  clk_oxo ] \
                                -group [get_clocks  -include_generated_clocks  clk_core ]
```

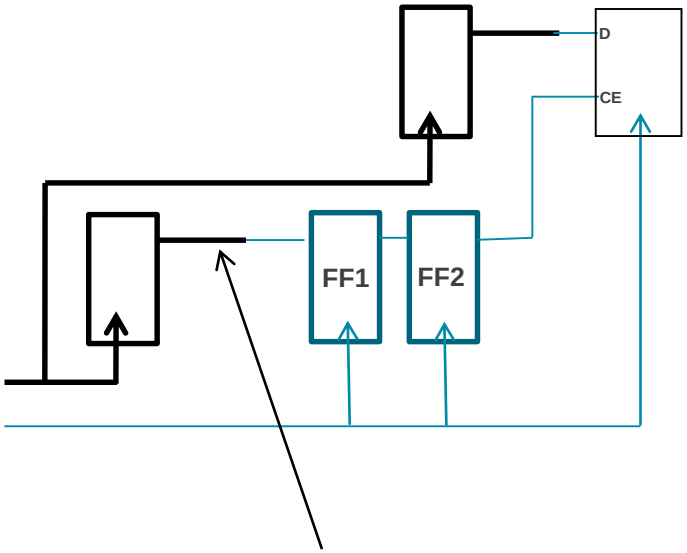
CDC结构支持

- **异步复位的同步**
  - Asserted Asynchronously, Synchronously Removed



CDC结构支持

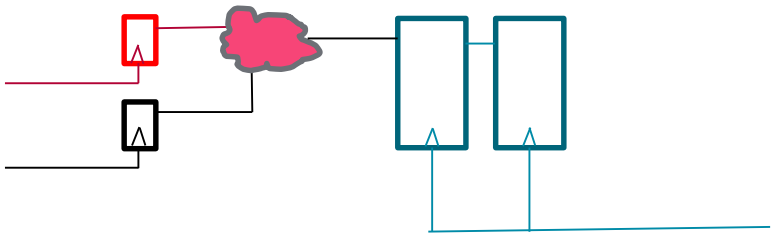
- **Clock-enable Controlled CDC**



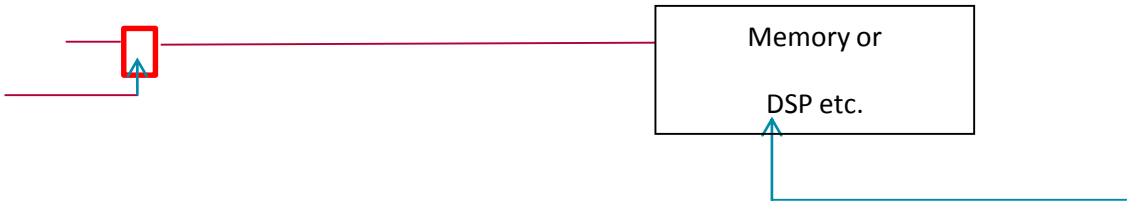
Add a max delay datapath only constraint

CDC structures supported

- **Multi-Clock Fan-in to synchronizer**



- **CDC on a non-FD primitive**



The report_cdc Command Line

- **report_cdc**

  -summary (Default and optional)

  -verbose (Optional)

  -file <File Name>

- **File name not specified, output in TCL Console**

CDC约束方案的对比

- 全部忽略的约束
  - 最大化全部忽略CDC路径的约束，即采用set_clock_groups 或是set_false_path对时钟关系进行约束，从而对跨时钟域的路径全部忽略。
  - 示例：set_clock_groups -asynchronous -group clkA -group clkB
  - set_false_path -from [get_clocks clkA] -to [get_clocks clkB]
  - 优势：简单、快速、执行效率高。
  - 劣势：会掩盖时序报告中所有的跨时钟域路径，容易误伤，不利于时序分析。
- 使用datapath_only约束
  - datapath_only在XDC中必须作为一个选项跟set_max_delay配合使用，可以约束在时钟之间，也可以对具体路径进行约束。
  - 示例：set_max_delay 10 -datapath_only -from clkA -to clkB
- 逐条进行时序例外约束
  - 对设计中的CDC路径分组或逐条分析，采用不同的时序例外约束，如set_false_path，set_max_delay和set_multicycle_path等来约束。
  - 示例：set_false_path -from [get_cells a/b/c/*_meta*] -to [get_cells a/b/c/*_sync*]

Lab2 Report CDC Demo

Duration: 20 Minutes

4、I/O时序约束

Objectives

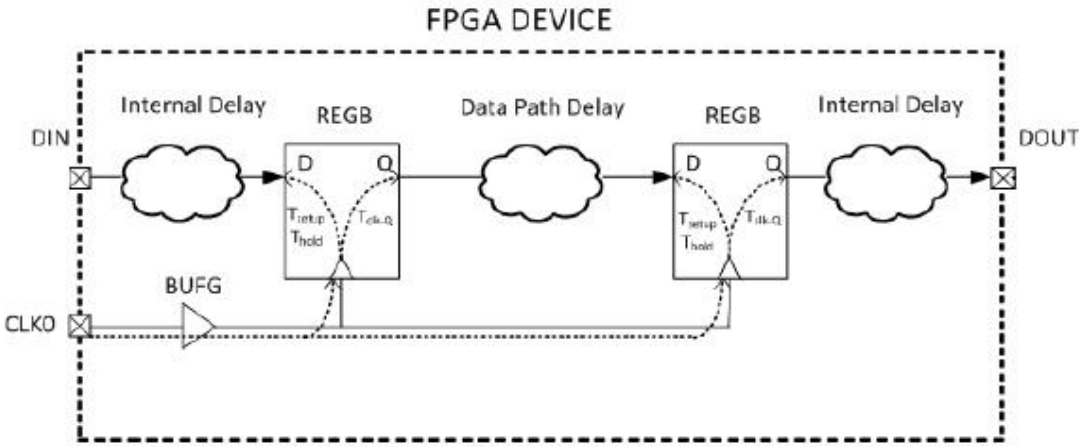
- **完成该培训课程后，可以达到如下效果:**
  - 创建合适的input delays和output delays
  - 在input delays和output delays中使用使用虚拟时钟
  - 描述及理解输入和输出的时序报告

Overview

- Overview
- Creating Input Delays
- Creating Output Delays
- Using Virtual Clocks

静态时序路径和I/O

- **静态时序路径就是从一个clocked elements 到另外一个clocked elements**
  - 其中内部寄存器路径就是一个寄存器到另外一个寄存器的路径，通过时钟约束确定时序情况
- **FPGA是Inputs 和outputs 不是静态时序路径的起点或者终点**
  - 默认情况下，任何I/O和内部逻辑之间不属于一完整的静态时序路径
  - 如果不添加额外命令，I/O上不会进行任何的时序分析



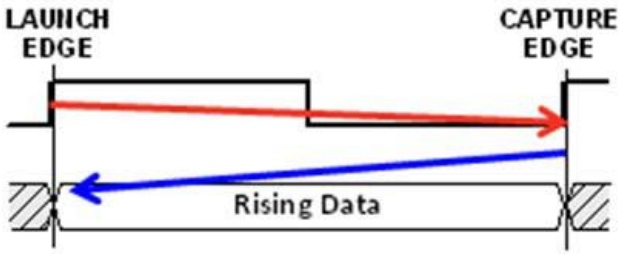
HWALT

I/O 约束的语法

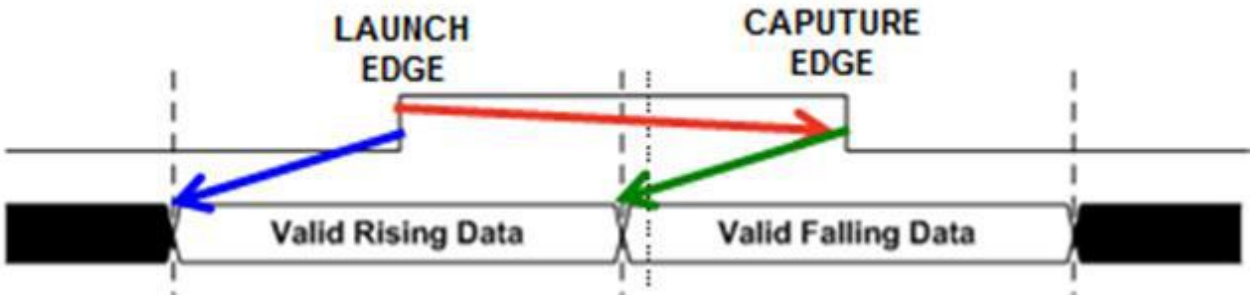
- **set_input_delay / set_output_delay**
  - 如果对 FPGA 的 I/O 不加任何约束，工具会认为时序要求为无穷大
  - 综合和实现时不会考虑 I/O 时序，而且在时序分析时也不会报出这些未约束的路径。
- **set_max_delay / set_min_delay**
  - 只有那些从 FPGA 管脚进入和/或输出都不经过任何时序元件的纯组合逻辑路径使用
  - **或者内部**

Setup/Hold 时序分析

- **Setup 时序分析**
  - 同步电路设计中，数据在时钟上升沿发送，在下一个时钟上升沿接收
  - 发送的时钟沿称作Launch Edge，接收沿称作Capture Edge
  - 时序分析中的Setup Check 跟Capture Edge 的选择息息相关。
- 在SDR 接口的setup 分析中
  - 工具如下图这样识别发送和接收时钟沿



- 而在DDR 接口的setup 分析中
  - 因为数据是双沿采样，所以发送和接收时钟沿变成上升（下降）沿发送，下降（上升）沿接收。

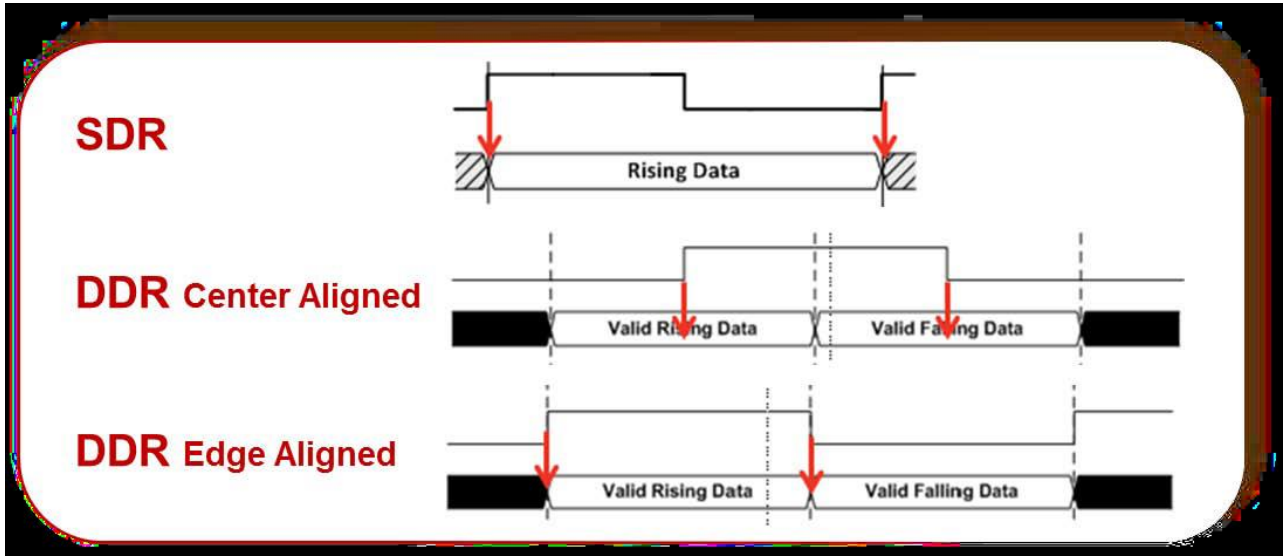


Setup/Hold 时序分析

Hold时序分析

– Hold Check 主要是保证数据在接收（采样）端时钟沿之后还能稳定保持的时间

– 对Hold 分析而言，同一个时钟沿既是Launch Edge 也是Capture Edge，这一点对SDR 和DDR（不论是中心对齐还是边沿对齐）都一样。

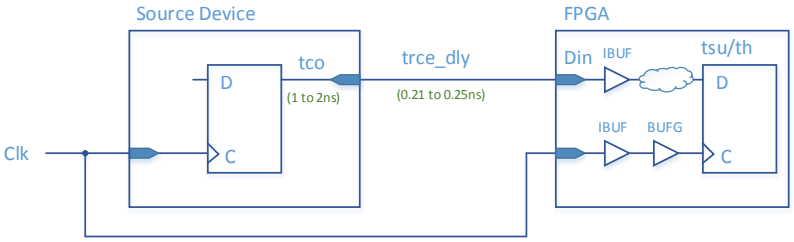


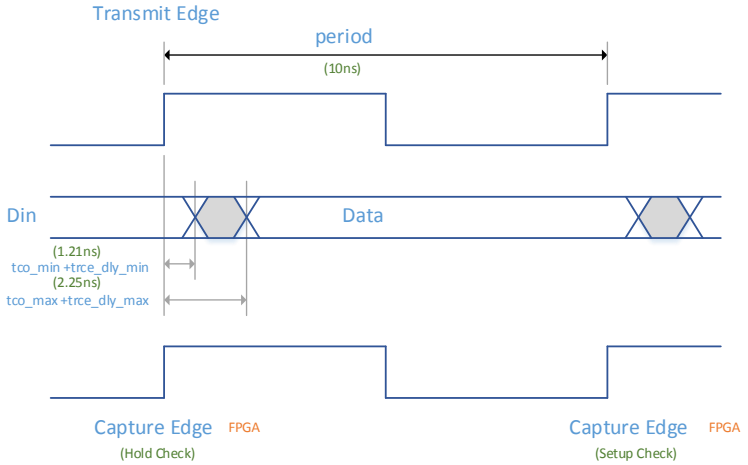
Overview

- Overview
- Creating Input Delays
- Creating Output Delays
- Using Virtual Clocks

IO约束

- Referenced to clock input
  - Max for setup analysis
  - Min for hold analysis



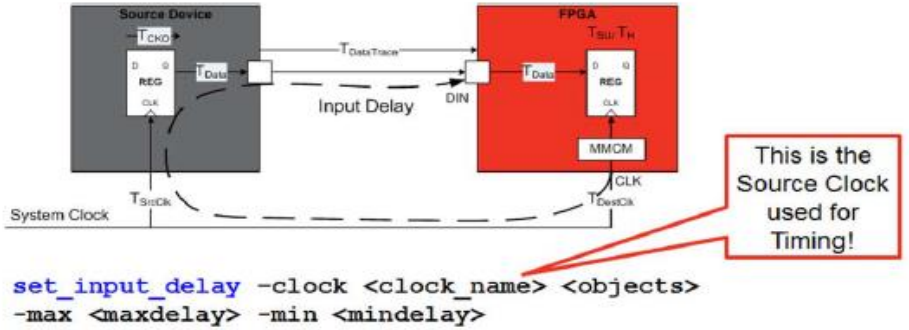


- set_input_delay -clock [get_clocks {Clk}] -min -add_delay 1.21 [get_ports {Din[*]}]
- set_input_delay -clock [get_clocks {Clk}] -max -add_delay 2.25 [get_ports {Din[*]}]

Creating Input Delays

set_input_delay

- set_input_delay –clock <clock_name> <delay> <objects>
- <objects>是想要设定 input 约束的端口名，可以是一个或数个 port
- -clock 之后的<clock_name>指明了对<objects>时序进行分析所用的时钟，
- 可以是一个 FPGA 中真实存在的时钟
- 也可以是预先定义好的虚拟时钟
- -max 之后的<maxdelay>描述了用于 setup 分析，包含有板级走线和外部器件的延时
- -min 之后的<mindelay>描述了用于 hold 分析，包含有板级走线和外部器件的延时。
- -add_delay 和 -clock_fall 用于DDR 接口的约束

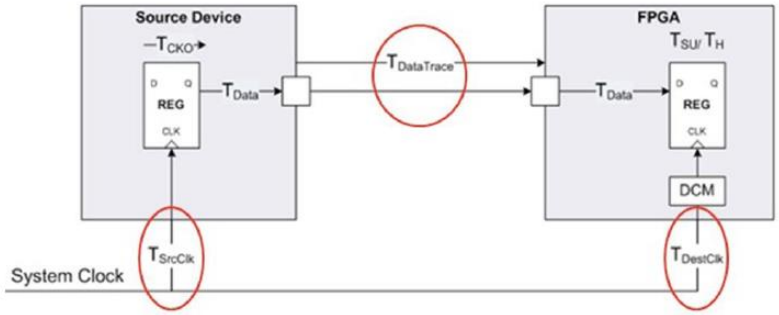


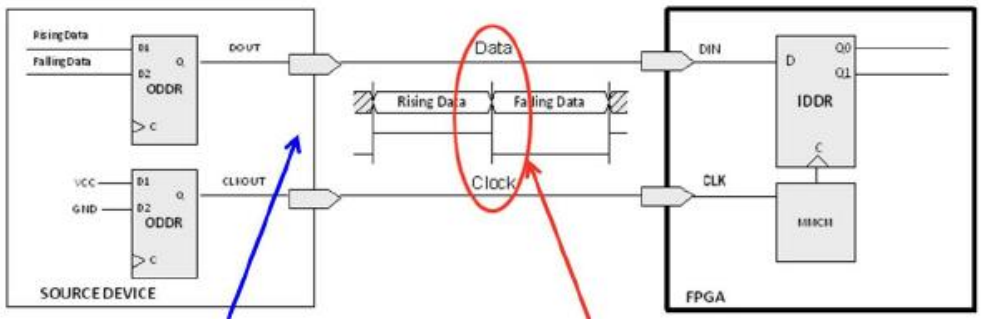
- **create_clock -name SysClk -period 10 [get_ports ClkIn]**
- **set_input_delay -clock SysClk 4 [get_ports DataIn]**

Creating Input Delays

数据接口同步方式

- FPGA 的数据接口同步根据系统级设计方式来讲可以分为系统同步和源同步两种
- 系统同步，时钟信号在系统级上同源，板级走线的延时也要对齐，无法达到更高速的设计要求，所以大部分情况也仅仅应用SDR 方式，（System Synchronous Interface）
- 源同步，在发送端将数据和时钟同步传输，在接收端用时钟沿脉冲来对数据进行锁存，重新使数据与时钟同步（Source Synchronous Interface）
- 源同步接口最大的优点就是大大提升了总线的速度，可以是SDR方式，也可以是DDR方式

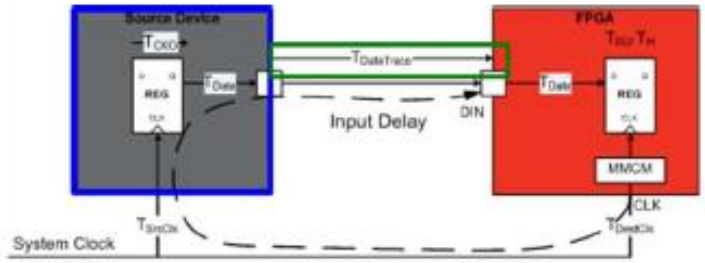


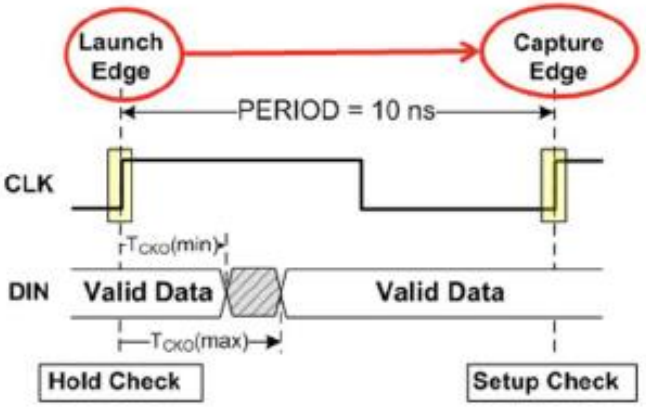


HWALT

Creating Input Delays

- 系统同步方式
  - 对系统同步接口做Input 约束，只需考虑上游器件的Tcko 和数据在板级的延时即可。
  - SDR 上升沿采样系统同步接口的Input 约束示例。





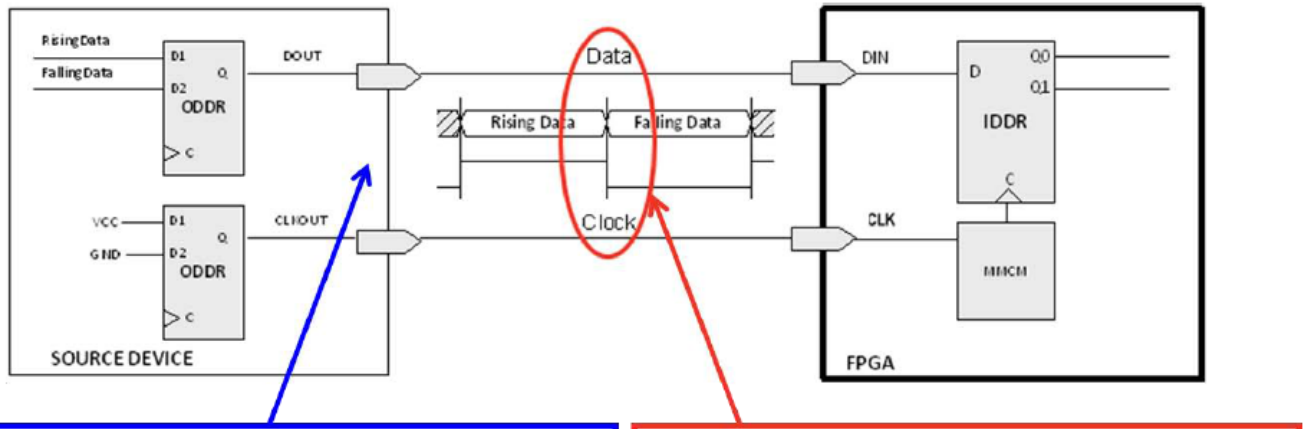
SOURCE DEVICE	
T_{CKO} (min)	=1ns
T_{CKO} (max)	=2ns

TRACE DELAY	
$T_{TRACE(min)}$	=.3ns
$T_{TRACE(max)}$	=.4ns

```
create_clock -name sysclk -period 10    [get_ports CLK];
set_input_delay -clock sysclk -max 2.4 [get_ports DIN];
set_input_delay -clock sysclk -min 1.3 [get_ports DIN];
```

Creating Input Delays

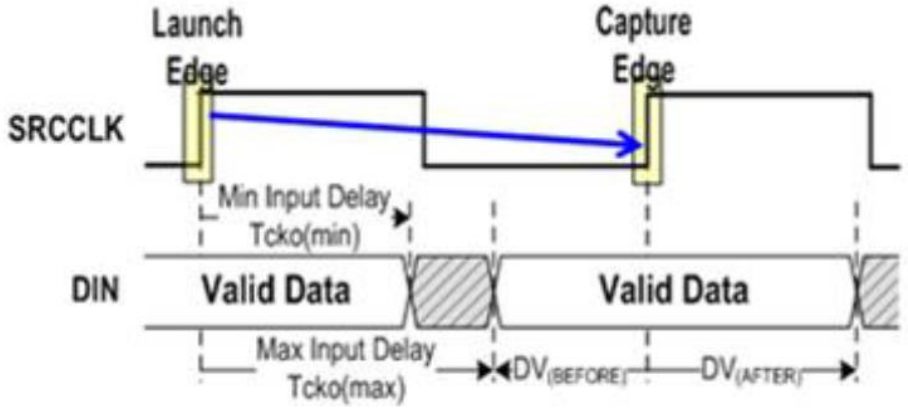
- 源同步方式
  - 源同步接口的约束设置相对复杂，
  - 一则是因为有SDR、DDR、中心对齐（Center Aligned）和边沿对齐（Edge Aligned）等多种方式，
  - 二则可以根据客观已知条件，选用与系统同步接口类似的系统级视角的方式，或是用源同步视角的方式来设置约束。



- 方法 1: System Method
  - 已知上游器件的Tcko值
  - 无需接口的数据有效窗口
- 方法 2: Source Sync Method
  - 已知接口的数据有效窗口
  - 源同步接口约束的典型方式

Creating Input Delays

- SDR源同步方式
  - 对源同步接口进行Input 约束可以根据不同的已知条件，选用不同的约束方式
  - 不论以何种方式来设置Input 约束，作用是一样
  - System method
  - source sync method



System Method Parameters	
Tcko Max	3ns
Tcko Min	2ns
Period	5ns

```
create_clock -name sysclk -period 5     [get_ports CLK];
set_input_delay -clock sysclk -max 3    [get_ports DIN];
set_input_delay -clock sysclk -min 2    [get_ports DIN];
```

Src Sync Method Parameters	
$DV_{(BEFORE)}$	2ns
$DV_{(AFTER)}$	2ns
Period	5ns

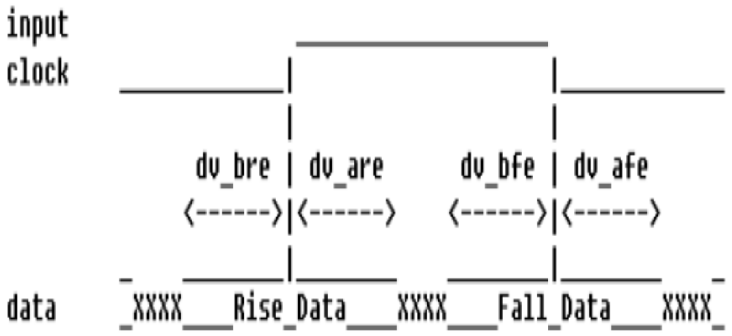
Max = (Period – DV_{BEFORE})

```
create_clock -name sysclk -period 5     [get_ports CLK];
set_input_delay -clock sysclk -max 3    [get_ports DIN];
set_input_delay -clock sysclk -min 2    [get_ports DIN];
```


HWALT

Creating Input Delays

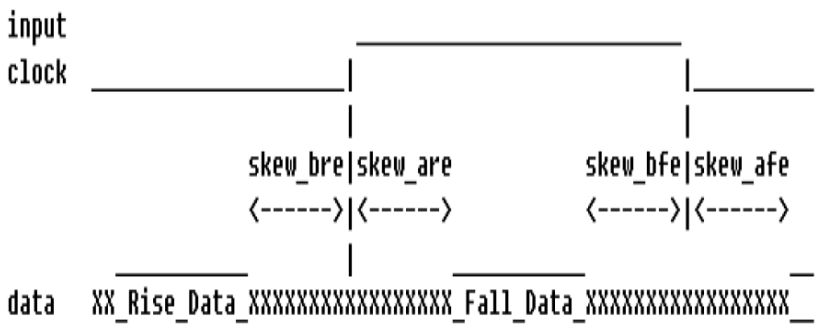
- DDR源同步方式
  - DDR 源同步接口的约束需要将上升沿和下降沿分别考虑和约束
  - 输入接口数据为中心对齐或边沿对齐的方式两种
  - Maximum delay is used for the setup check
  - Minimum delay is used for the hold check
  - set_input_delay 中-max/-min 的定义，时钟采样沿到达之后最大与最小的数据有效窗口
- DDR 源同步中心对齐输入接口
  - 已知条件如下：
    - 时钟信号 src_sync_ddr_clk 的频率： 100 MHz
    - 数据总线： src_sync_ddr_din[3:0]
    - 上升沿之前的数据有效窗口 (dv_bre) ： 0.4 ns
    - 上升沿之后的数据有效窗口 (dv_are) ： 0.6 ns
    - 下降沿之前的数据有效窗口 (dv_bfe) ： 0.7 ns
    - 下降沿之后的数据有效窗口 (dv_afe) ： 0.2 ns




```
set period 10.0;
create_clock -period $period -name clk [get_ports src_sync_ddr_clk];
set_input_delay -clock clk -max    [expr $period/2 – 0.7] [get_ports src_sync_ddr_din[*]] ;
set_input_delay -clock clk -min 0.6 [get_ports src_sync_ddr_din[*]] ;
set_input_delay -clock clk –max [expr $period/2 – 0.4] \
                                    [get_ports src_sync_ddr_din[*]] -clock_fall -add_delay ;
set_input_delay -clock clk -min 0.2 [get_ports src_sync_ddr_din[*]] -clock_fall -add_delay;
```

Creating Input Delays

- DDR 源同步边沿对齐输入接口
  - 已知条件如下：
    - 时钟信号 src_sync_ddr_clk 的频率： 100 MHz
    - 数据总线： src_sync_ddr_din[3:0]
    - 上升沿之前的数据skew (skew_bre) ： 0.6 ns
    - 上升沿之后的数据skew (skew_are) ： 0.4 ns
    - 下降沿之前的数据skew (skew_bfe) ： 0.3 ns
    - 下降沿之后的数据skew (skew_afe) ： 0.7 ns




```
create_clock -period 10.0 -name clk [get_ports src_sync_ddr_clk];
set_input_delay -clock clk -max 0.4 [get_ports src_sync_ddr_din[*]] ;
set_input_delay -clock clk -min -0.6 [get_ports src_sync_ddr_din[*]] ;
set_input_delay -clock clk -max 0.7 [get_ports src_sync_ddr_din[*]] -clock_fall -add_delay ;
set_input_delay -clock clk -min -0.3 [get_ports src_sync_ddr_din[*]] -clock_fall -add_delay;
```

Creating Input Delays

- DDR 源同步边沿对齐输入接口
  - set_input_delay 中-max/-min 的定义，时钟采样沿到达之后最大与最小的数据有效窗口
  - 数据是边沿对齐，只要有jitter 跟skew 的存在，最差情况下，数据有效窗口在到达时钟采样沿之前就已经结束，所以会有负数出现在-min 之后
  - FPGA 用作输入的边沿对齐DDR 源同步接口的情况下，真正用来采样数据的时钟会经过一个MMCM/PLL 做一定的相移，从而把边沿对齐变成中心对齐
  - 经过MMCM/PLL 相移后的采样时钟跟同步接口输入的时钟之间需要做set_false_path 的约束（如下述例子）而把那些伪路径从时序报告中剔除

```
set_false_path –setup –rise_from [get_clocks adc_dclk_p] –rise_to [get_clocks clk_outp0_adc_pll_1]
set_false_path –setup –fall_from [get_clocks adc_dclk_p] –fall_to [get_clocks clk_outp0_adc_pll_1]
set_false_path –hold –fall_from [get_clocks adc_dclk_p] –rise_to [get_clocks clk_outp0_adc_pll_1]
set_false_path –hold –rise_from [get_clocks adc_dclk_p] –fall_to [get_clocks clk_outp0_adc_pll_1]
```

Input Setup Timing Report Summary

```
Slack (MET) :           1.898ns
  Source:               DataIn[0]
                            (input port clocked by SysClk {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:          data_p1_reg[0]/D
                            (rising edge-triggered cell FDCE clocked by clkout0 {rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:           clkout0
  Path Type:            Max at Slow Process Corner
  Requirement:          10.000ns
  Data Path Delay:      1.925ns  (logic 0.847ns (43.997%)  route 1.078ns (56.003%))
  Logic Levels:         2  (IBUF=1 LUT4=1)
  Input Delay:          4.000ns
  Clock Path Skew:      -2.005ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):  -2.005ns
    Source Clock Delay      (SCD):  0.000ns
    Clock Pessimism Removal (CPR):  0.000ns
  Clock Uncertainty:    0.172ns  ((TSJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter     (TSJ):  0.071ns
    Discrete Jitter          (DJ):  0.129ns
    Phase Error              (PE):  0.099ns
```

Input Setup Timing Report Detailed Paths

Location	Delay type	Incr(ns)	Path(ns)	Netlist Resource(s)
	(clock SysClk rise edge)	0.000	0.000 r	
	input delay	4.000	4.000	
U17		0.000	4.000 r	DataIn[0]
U17	IBUF (Prop_ibuf_I_O)	0.829	4.829 r	DataIn_IBUF[0]_inst/O
SLICE_X0Y42	LUT4 (Prop_lut4_I1_O)	1.131	5.960 r	p_0_out_inferredi_0/O
SLICE_X0Y42	net (fo=1, routed)	0.000	5.960 r	data_p1_reg[0]/D
SLICE_X0Y42	FDCE (Setup_fdce_C_D)	-0.035	5.925	data_p1_reg[0]
	(clock clkout0 rise edge)	10.000	10.000 r	
V20		0.000	10.000 r	ClkIn
V20	IBUFG (Prop_ibufg_I_O)	0.741	10.741 r	clk_core_i0/inst/clknetwork_int/clkin1_buf/O
MMCME2_ADV_X0Y0	MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)			
		-6.583	4.158 r	clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0
BUFGCTRL_X0Y1	BUFG (Prop_bufg_I_O)	2.345	6.503 r	clk_core_i0/inst/clknetwork_int/clkout1_buf/O
SLICE_X0Y42	net (fo=4, routed)	1.492	7.995 r	data_p1_reg[0]/C
	clock pessimism	0.000	7.995	
	clock uncertainty	-0.172	7.823	
	required time		7.823	
	arrival time		-5.925	
	slack		1.898	

Source Clock Delay

Data Path Delay

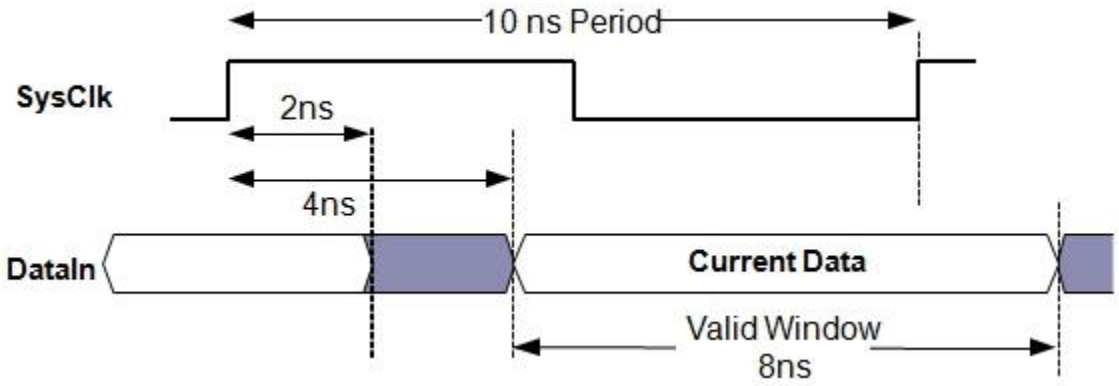
Slack Calculation

Destination Clock Delay

Minimum and Maximum Delays

- **By default, each input port can have one maximum delay and one minimum delay**
  - Maximum delay is used for the setup check
  - Minimum delay is used for the hold check
- **Without the `–max` or `–min` option, the value supplied is used for both**

```
create_clock –name SysClk –period 10 [get_ports ClkIn]
set_input_delay –clock SysClk 4       [get_ports DataIn]
set_input_delay –clock SysClk 2 –min  [get_ports DataIn]
```



Input Hold Timing Report Summary

```
Slack (MET) :             3.107ns
  Source:                 DataIn[0]
                            (input port clocked by SysClk) (rise@0.000ns fall@5.000ns period=10.000ns))
  Destination:            data_p1_reg[0]/D
                            (rising edge-triggered cell FDCE clocked by clkout0) (rise@0.000ns fall@5.000ns period=10.000ns))
  Path Group:             clkout0
  Path Type:              Min at Fast Process Corner
  Requirement:            0.000ns
  Data Path Delay:        0.809ns  (logic 0.343ns (42.391%)  route 0.466ns (57.609%))
  Logic Levels:           2  (IBUF=1 LUT4=1)
  Input Delay:            2.000ns
  Clock Path Skew:        -0.470ns (DCD - SCD - CPR)
    Destination Clock Delay (DCD):  -0.470ns
    Source Clock Delay      (SCD):  0.000ns
    Clock Pessimism Removal (CPR):  -0.000ns
  Clock Uncertainty:      0.172ns  ((TSJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter     (TSJ):  0.071ns
    Discrete Jitter          (DJ):  0.129ns
    Phase Error              (PE):  0.099ns
```

Input Hold Timing Report Detailed Paths

Location	Delay type	Incr(ns)	Path(ns)	Netlist Resource(s)
	(clock SysClk rise edge)	0.000	0.000 r	
	input delay	2.000	2.000	
U17		0.000	2.000 r	DataIn[0]
U17	IBUF (Prop_ibuf_I_0)	0.375	2.375 r	DataIn_IBUF[0]_inst/0
SLICE_X0Y42	LUT4 (Prop_lut4_I1_0)	0.494	2.869 r	p_0_out_inferredi_0/0
SLICE_X0Y42	net (fo=1, routed)	0.000	2.869 r	data_p1_reg[0]/D
SLICE_X0Y42	FDCE (Hold_fdce_C_D)	-0.060	2.809	data_p1_reg[0]
	(clock clkout0 rise edge)	0.000	0.000 r	
V20		0.000	0.000 r	ClkIn
V20	IBUFG (Prop_ibufg_I_0)	0.545	0.545 r	clk_core_i0/inst/clknetwork_int/clkin1_buf/0
MMCME2_ADV_X0Y0	MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)			
		-2.719	-2.174 r	clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0
BUFGCTRL_X0Y1	BUFG (Prop_bufg_I_0)	0.843	-1.331 r	clk_core_i0/inst/clknetwork_int/clkout1_buf/0
SLICE_X0Y42	net (fo=4, routed)	0.861	-0.470 r	data_p1_reg[0]/C
	clock pessimism	0.000	-0.470	
	clock uncertainty	0.172	-0.298	
	required time		0.298	
	arrival time		2.809	
	slack		3.107	

Source Clock Delay

Data Path Delay

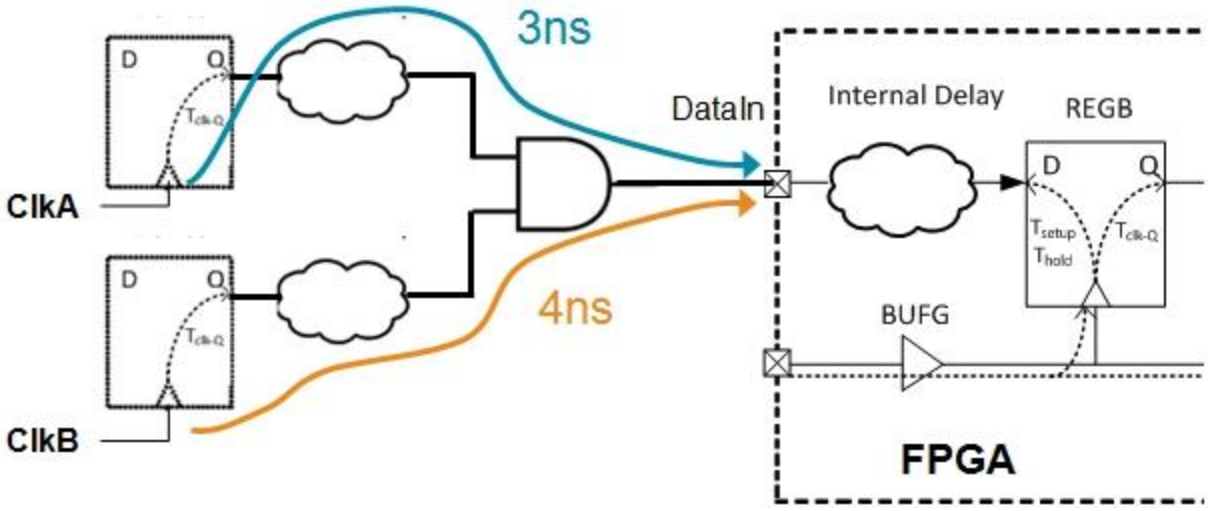
Destination Clock Delay

Slack Calculation

Multiple Input Delays on the Same Port

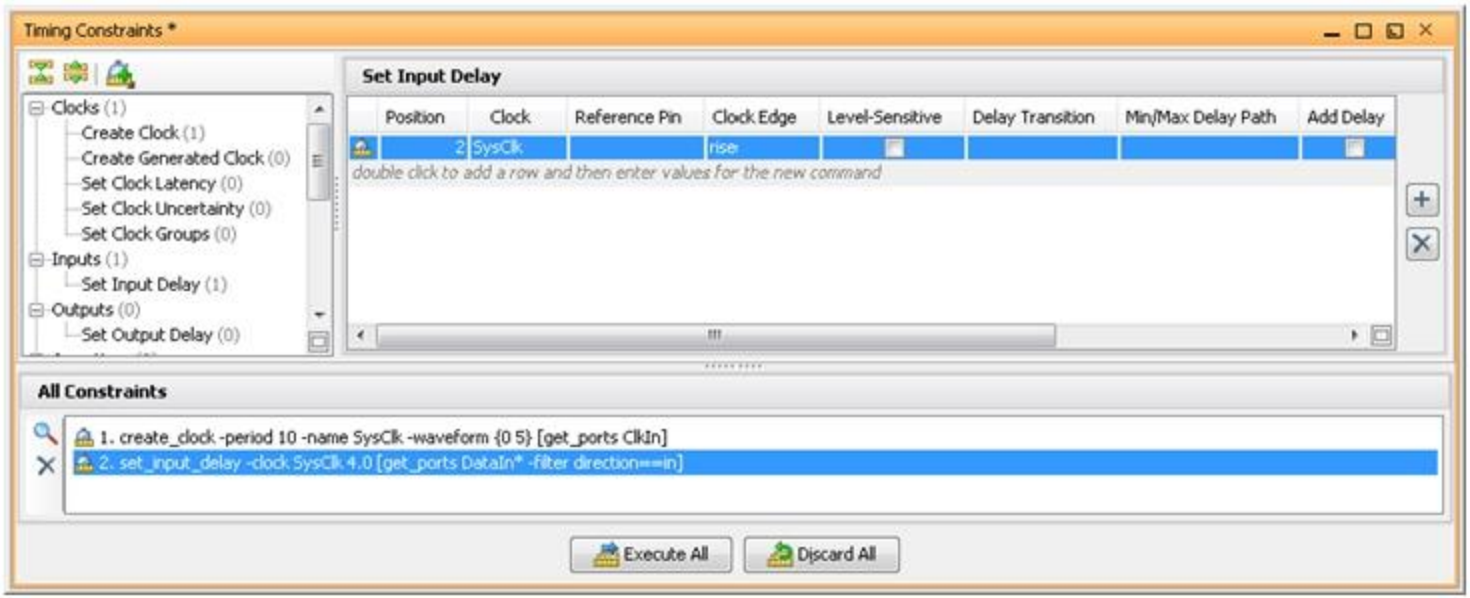
- **An input can have multiple `set_input_delay` commands associated with it**
  - Use the `–add_delay` option
  - Results in multiple static timing paths to check

```
set_input_delay -clock ClkA 3 [get_ports DataIn]
set_input_delay -clock ClkB 4 [get_ports DataIn] -add_delay
```



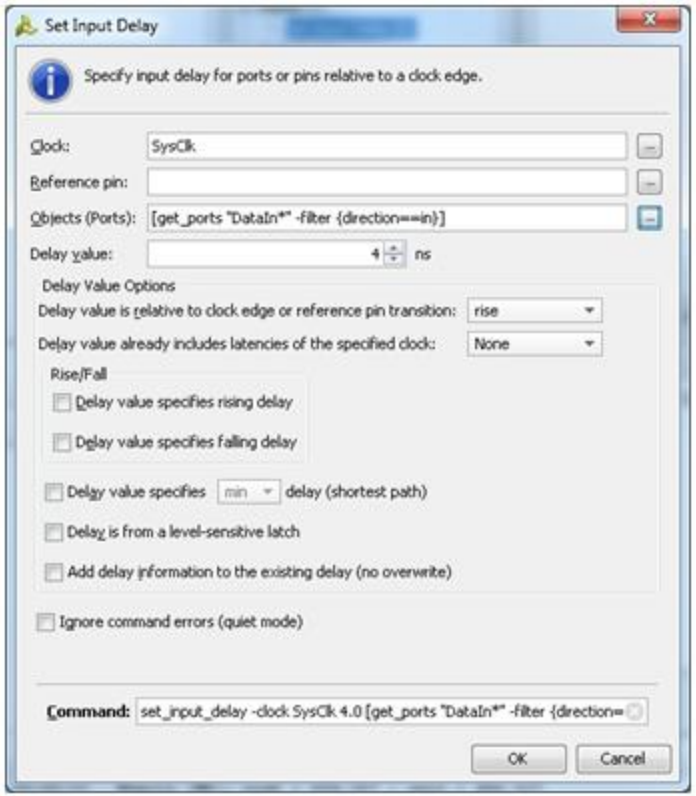
Creating Input Delays Using the GUI

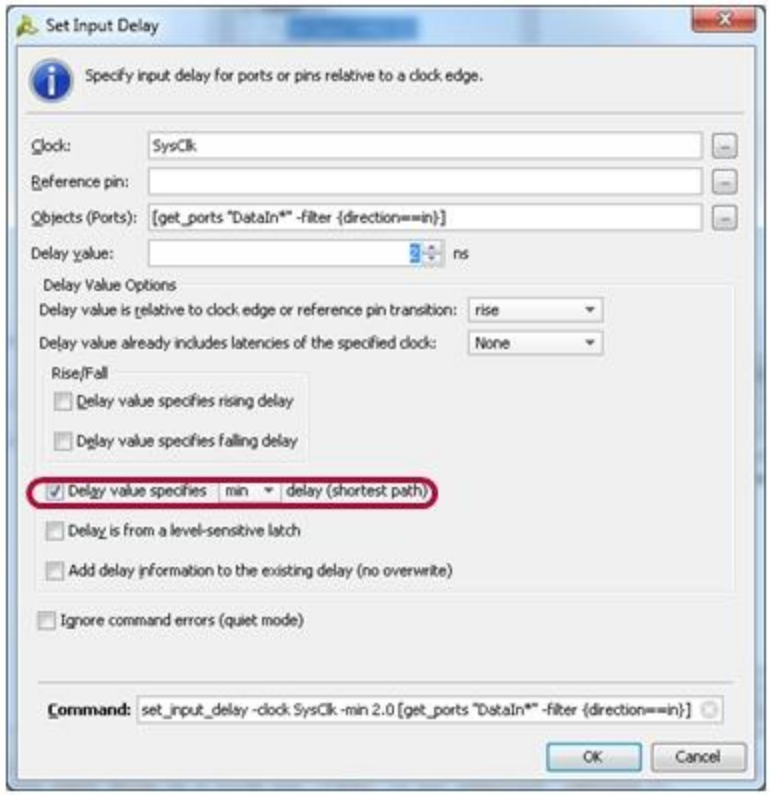
- **Timing Constraint window can be opened by selecting Window > Timing Constraints**
  - Clock can be created by double-clicking **Set Input Delay**, or a new row in the Set Input Delay table
- **Alternatively a clock can be created directly by selecting Tools > Timing > Create Timing Constraint > Inputs > Set Input Delay**



Set Input Delay Wizard

- **Separate constraint is required for the maximum and minimum**
  - Wizard retains its options between invocations, making it easy to specify the minimum after the maximum has been specified



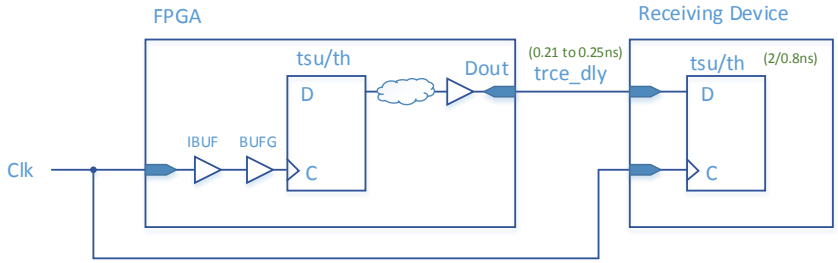


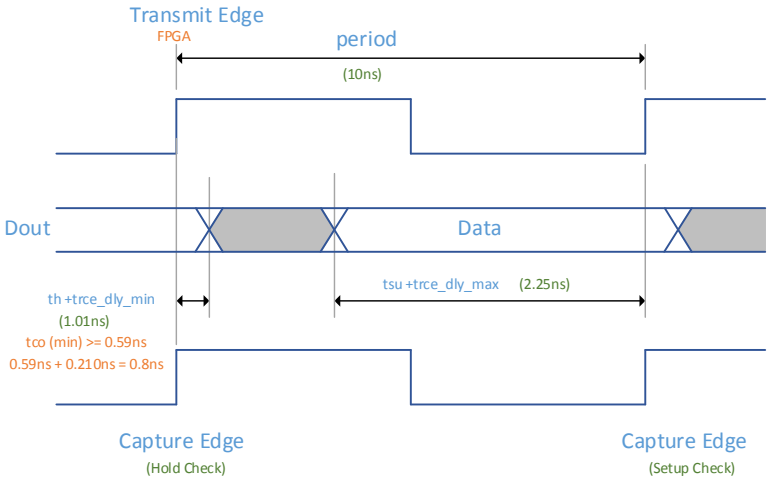
Overview

- Overview
- Creating Input Delays
- Creating Output Delays
- Using Virtual Clocks

IO约束

- Referenced to clock output
  - Max for setup analysis
  - Min for hold analysis



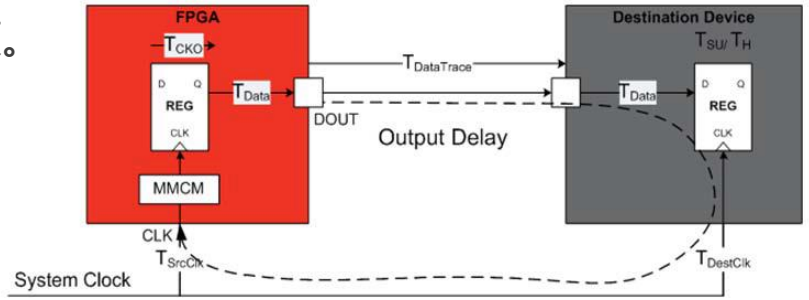


- set_output_delay -clock [get_clocks {Clk}] -min -add_delay -0.59 [get_ports {Dout[*]}]
- set_output_delay -clock [get_clocks {Clk}] -max -add_delay 2.25 [get_ports {Dout[*]}]

Creating output Delays

set_output_delay

- <objects>
  - 是想要设定 output约束的端口名，可以是一个或数个 port
- -clock 之后的<clock_name>指明了对<objects>时序进行分析所用的时钟
  - 可以是一个 FPGA 中真实存在的时钟
  - 也可以是预先定义好的虚拟时钟
- -max 之后的<maxdelay>描述了用于 setup 分析的包含有板级走线和外部器件的延时
- -min 之后的<mindelay>描述了用于 hold 分析的包含有板级走线和外部器件的延时
- -add_delay 和 -clock_fall 用于DDR 接口的约束。




```
set_output_delay -clock <clock_name> <objects>
-max <maxdelay> -min <mindelay>
```

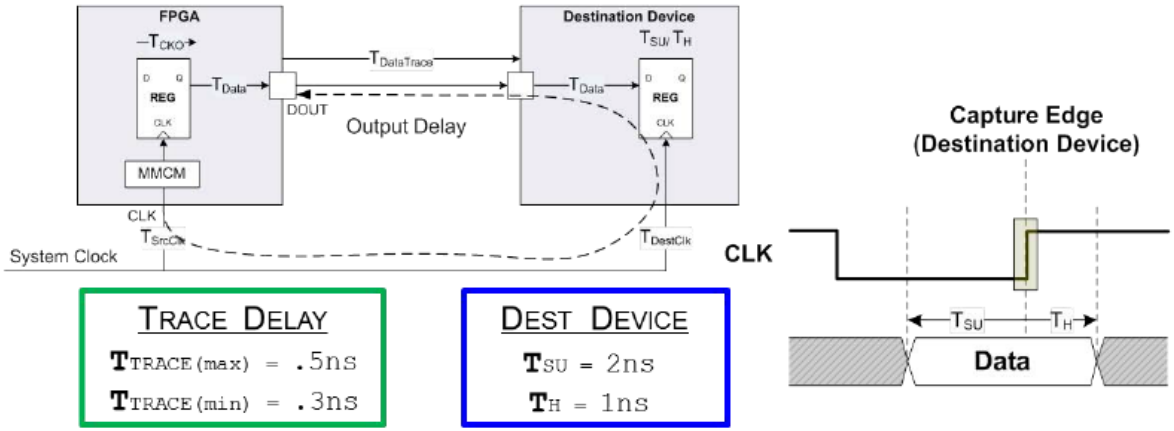
```
create_clock -name SysClk -period 10 [get_ports ClkIn]
set_output_delay -clock SysClk 4 [get_ports DataIn]
```

Creating output Delays

- 数据接口同步方式
  - FPGA 做Output 的接口时序同样也可以分为系统同步与源同步
  - 在源同步接口中，定义接口约束之前，需要用create_generated_clock 先定义送出的随路时钟

Creating output Delays

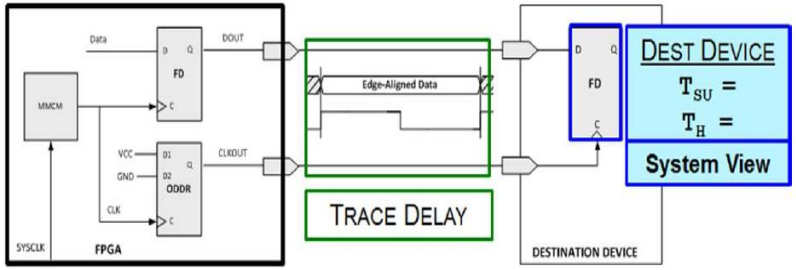
- 系统同步方式
  - FPGA 做Output 接口的系统同步设计，芯片间只传递数据信号，时钟信号的同步完全依靠板级设计来对齐
  - 设置约束时候要考虑的仅仅是下游器件的Tsu/Th 和数据在板级的延
  - 时。




```
create_clock -name sysclk -period 10      [get_ports  CLK];
set_output_delay -clock sysclk -max 2.5   [get_ports  DOUT];
set_output_delay -clock sysclk -min -0.7  [get_ports  DOUT];
```

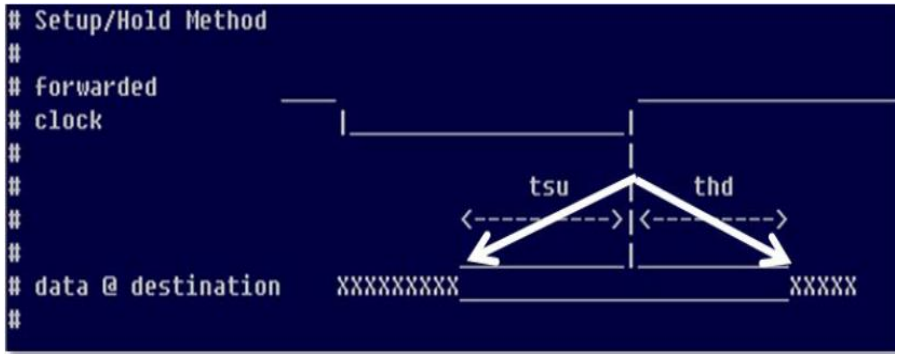
Creating output Delays

- 源同步方式
  - Vivado® IDE 的Language Templates 中关于源同步输出接口的XDC 约束模板包含了以上两种方式的设置方法。
  - set_output_delay 中 -max/-min 的定义，即时钟采样沿到达之前最大与最小的数据有效窗口。
  - 用于setup 分析的 -max 之后跟着正数，表示数据在时钟采样沿之前就到达，而用于 hold 分析的 -min 之后跟着负数，表示数据在时钟采样沿之后还保持了一段时间
- 方法1 Setup/Hold Based Method
  - 仅需要了解下游器件用来锁存数据的触发器的Tsu 与Th 值与系统板级的延时便可以设置



set_output_delay -max: T_{TRACE}(max) + Dest Register T_{SU}

set_output_delay -min: T_{TRACE}(min) - Dest Register T_H

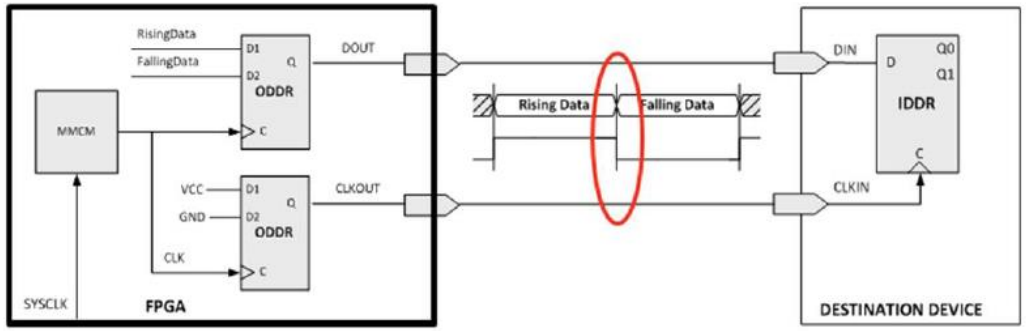


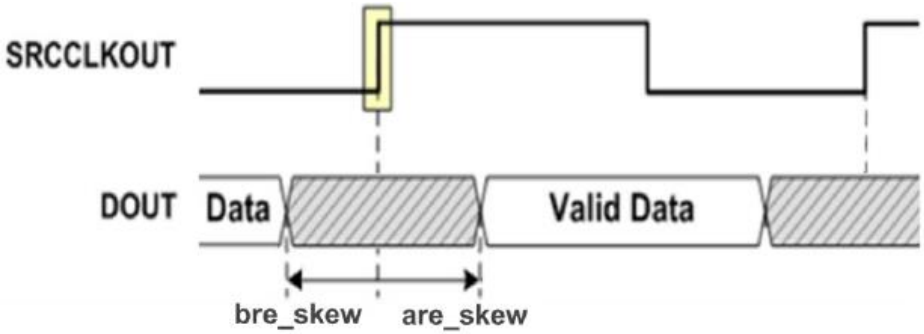
HWALT

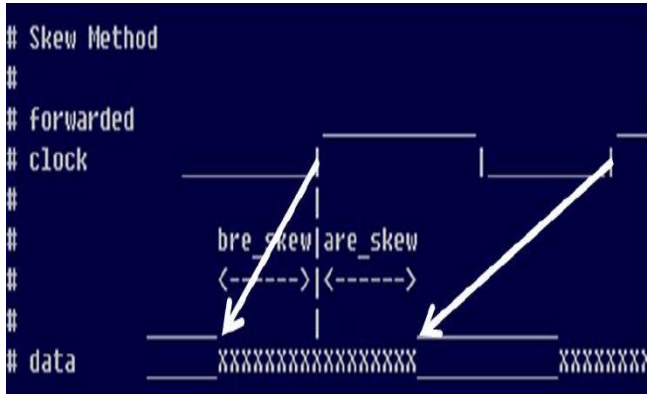
Creating output Delays

- 方法2 Skew Based Method
  - 时需要了解FPGA 送出的数据相对于时钟沿的关系，根据Skew 的大小和时钟频率来计算如何设置 Output约束
  - set_output_delay 中 -max/-min

的定义，即时钟采样沿到达
之前最大与最小的
数据有效窗口








```
set_output_delay -max: SRC Clock Period - are_skew
set_output_delay -min: bre_skew
```

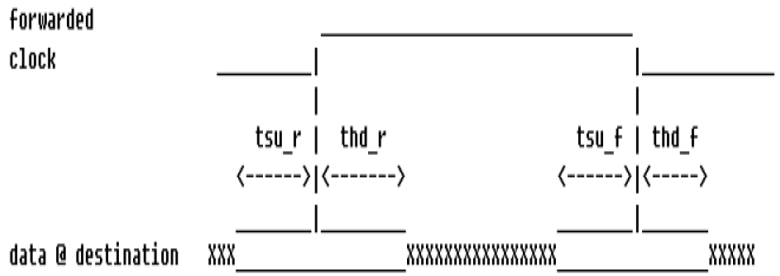

HWALT

Creating output Delays

DDR 源同步方法1 Setup/Hold Based

– 已知条件如下：

- 时钟信号 src_sync_ddr_clk 的频率： 100 MHz
- 随路送出的时钟src_sync_ddr_clk_out 的频率： 100 MHz
- 数据总线： src_sync_ddr_dout[3:0]
- 接收端的上升沿建立时间要求 (tsu_r) ： 0.7 ns
- 接收端的上升沿保持时间要求 (thd_r) ： 0.3 ns
- 接收端的下降沿建立时间要求 (tsu_f) ： 0.6 ns
- 接收端的下降沿保持时间要求 (thd_f) ： 0.4 ns
- 板级走线延时：0 ns




```
create_clock -period 10.0    -name clk [get_ports src_sync_ddr_clk];
create_generated_clock     -name clk_out    [get_ports ddr_src_sync_clk_out] \
                          -source [get_ports src_sync_ddr_clk] -divide_by 1;
set_output_delay -clock clk_out -max 0.7 [get_ports src_sync_ddr_dout[*]] ;
set_output_delay -clock clk_out -min -0.3 [get_ports src_sync_ddr_dout[*]] ;
set_output_delay -clock clk_out -max 0.6 [get_ports src_sync_ddr_dout[*]]-clock_fall -add_delay;
set_output_delay -clock clk_out -min -0.4 [get_ports src_sync_ddr_dout[*]] -clock_fall -add_delay;
```

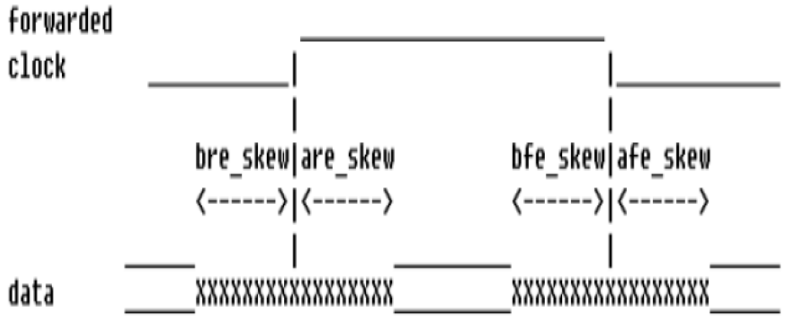

HWALT

Creating output Delays

DDR 源同步方法2 Skew Based Method

– 已知条件如下：

- 时钟信号 src_sync_ddr_clk 的频率： 100 MHz
- 随路送出的时钟src_sync_ddr_clk_out 的频率： 100 MHz
- 数据总线： src_sync_ddr_dout[3:0]
- 上升沿之前的数据skew (bre_skew) ： 0.4 ns
- 上升沿之后的数据skew (are_skew) ： 0.6 ns
- 下降沿之前的数据skew (bfe_skew) ： 0.7 ns
- 下降沿之后的数据skew (afe_skew) ： 0.2 ns

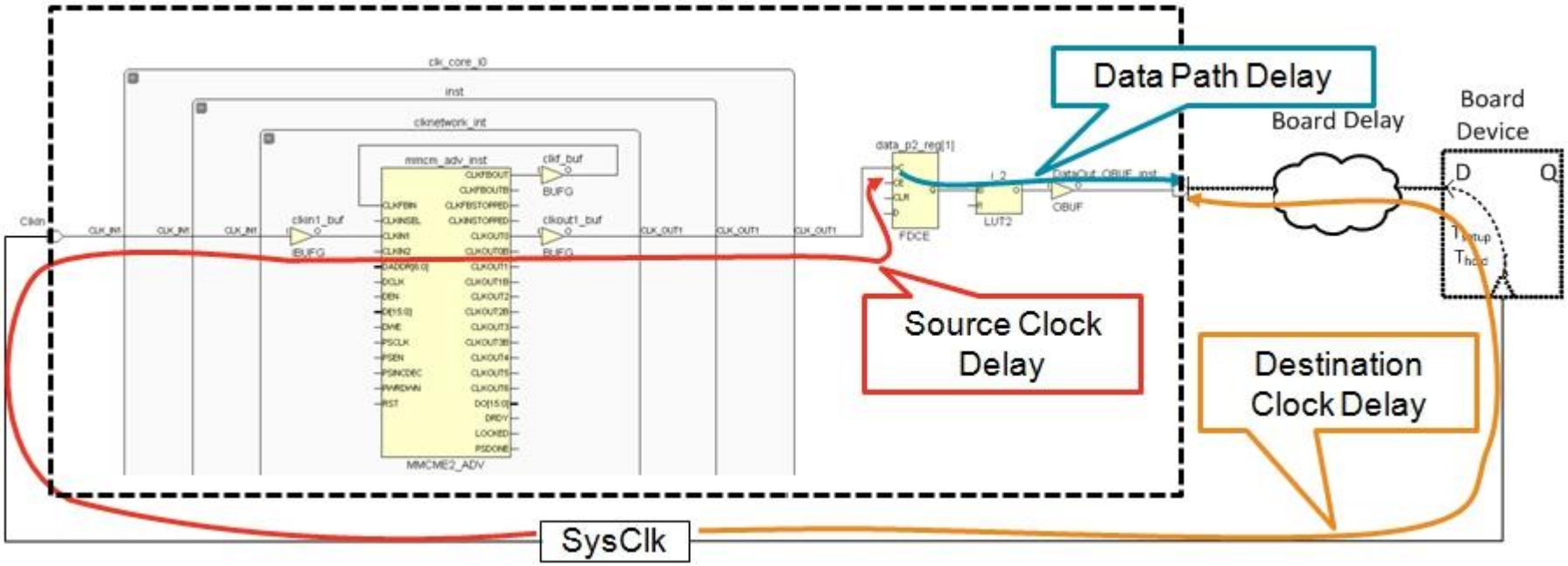



```
set period 10.0;
create_clock -period $period -name clk [get_ports src_sync_ddr_clk];
create_generated_clock     -name clk_out    [get_ports ddr_src_sync_clk_out] \
                                -source [get_ports src_sync_ddr_clk] -divide_by 1;
set_output_delay -clock clk_out -max [expr $period/2 – 0.2] [get_ports src_sync_ddr_dout[*]] ;
set_output_delay -clock clk_out -min 0.4 [get_ports src_sync_ddr_dout[*]] ;
set_output_delay -clock clk_out -max [expr $period/2 – 0.6] [get_ports src_sync_ddr_dout[*]] \
                                -clock_fall -add_delay ;
set_output_delay -clock clk_out -min 0.7 [get_ports src_sync_ddr_dout[*]] -clock_fall -add_delay;
```


HWALT

Complete Output Static Timing Path

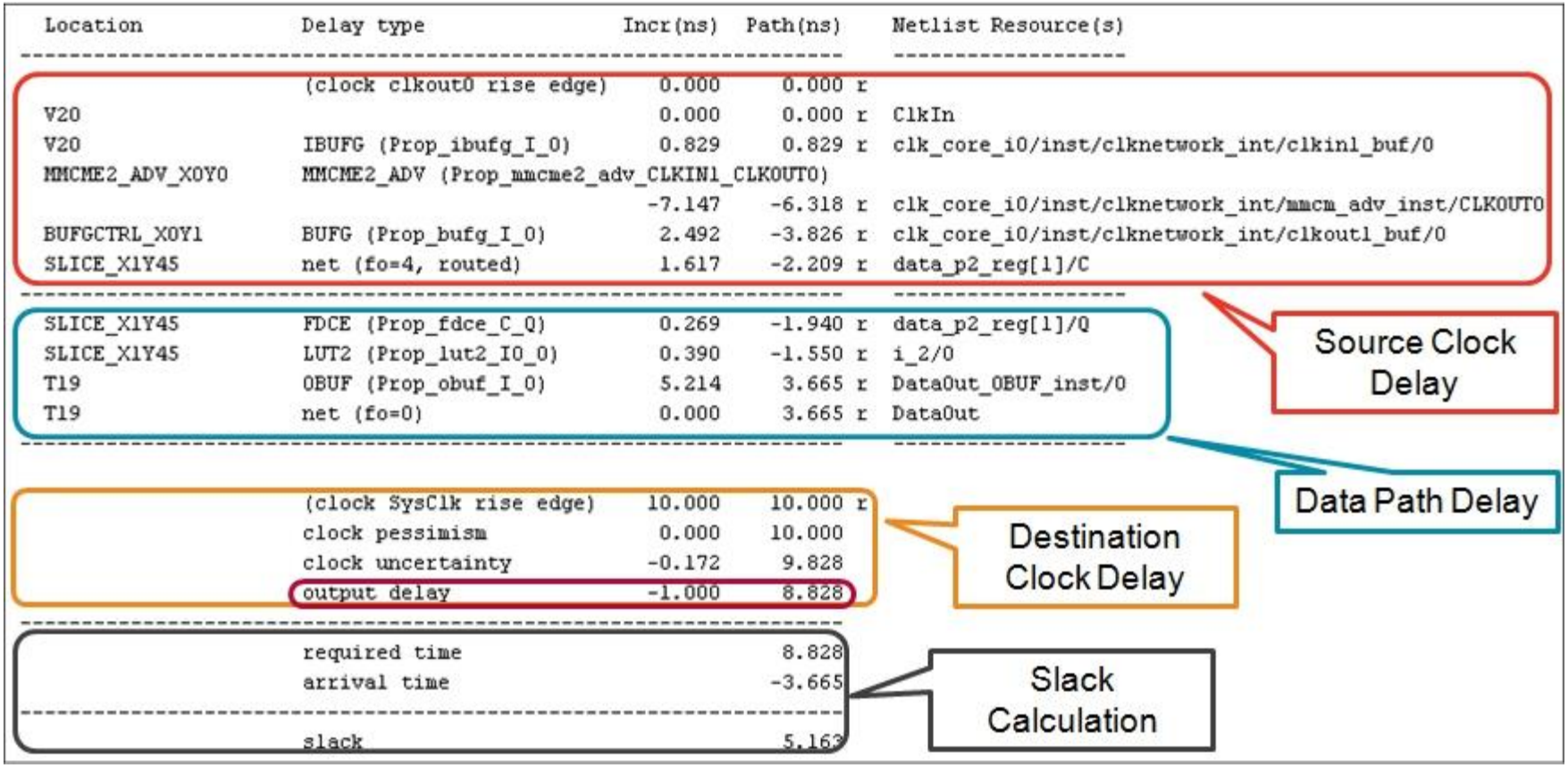
- **Output static timing path is segmented slightly differently**
  - Data path delay ends at the port of the FPGA
  - Destination clock path traces back through the board device to the port of the FPGA



Output Setup Timing Report Summary

```
Slack (MET) :             5.163ns
  Source:                 data_p2_reg[1]/C
                            (rising edge-triggered cell FDCE clocked by clkout0) {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            DataOut
                            (output port clocked by SysClk) (rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             SysClk
  Path Type:              Max at Slow Process Corner
  Requirement:            10.000ns
  Data Path Delay:        5.874ns  (logic 4.343ns (73.941%)  route 1.531ns (26.059%))
  Logic Levels:           2  (LUT2=1 OBUF=1)
  Output Delay:           1.000ns
  Clock Path Skew:        2.209ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):  0.000ns
    Source Clock Delay      (SCD):  -2.209ns
    Clock Pessimism Removal (CPR):  0.000ns
  Clock Uncertainty:      0.172ns  ((TSJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter     (TSJ):  0.071ns
    Discrete Jitter          (DJ):  0.129ns
    Phase Error              (PE):  0.099ns
```

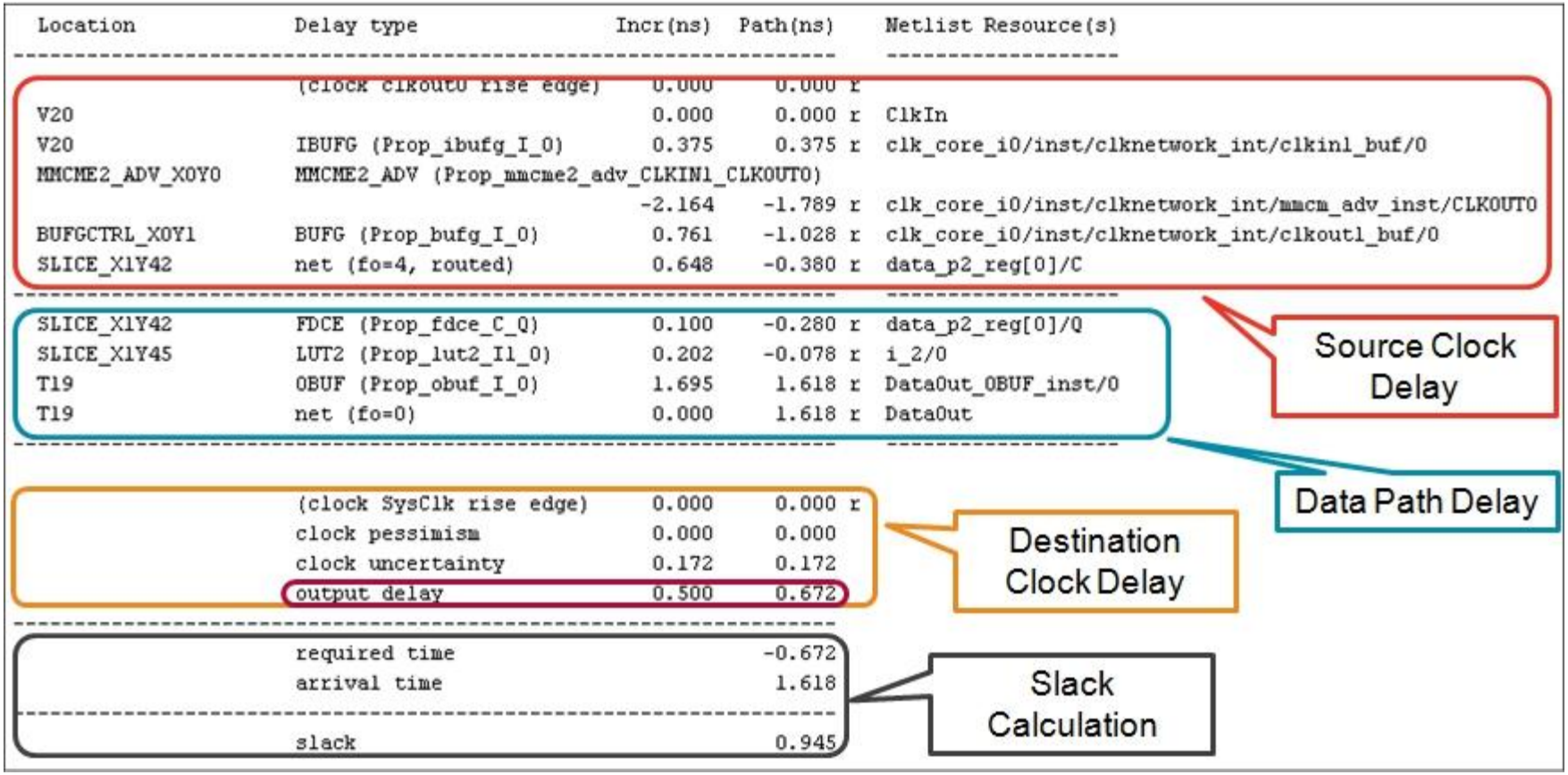
Output Setup Timing Report Detailed Paths



Output Hold Timing Report Summary

```
Slack (MET) :             0.945ns
  Source:                 data_p2_reg[0]/C
                            (rising edge-triggered cell FDCE clocked by clkout0  rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            DataOut
                            (output port clocked by SysClk  {rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             SysClk
  Path Type:              Min at Fast Process Corner
  Requirement:            0.500ns
  Data Path Delay:        1.998ns  (logic 1.334ns (66.783%)  route 0.664ns (33.217%))
  Logic Levels:           2  (LUT2=1 OBUF=1)
  Output Delay:           -0.500ns
  Clock Path Skew:        0.380ns (DCD - SCD - CPR)
    Destination Clock Delay (DCD):  0.000ns
    Source Clock Delay      (SCD):  -0.380ns
    Clock Pessimism Removal (CPR):  -0.000ns
  Clock Uncertainty:      0.172ns  ((TSJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter     (TSJ):  0.071ns
    Discrete Jitter          (DJ):  0.129ns
    Phase Error              (PE):  0.099ns
```

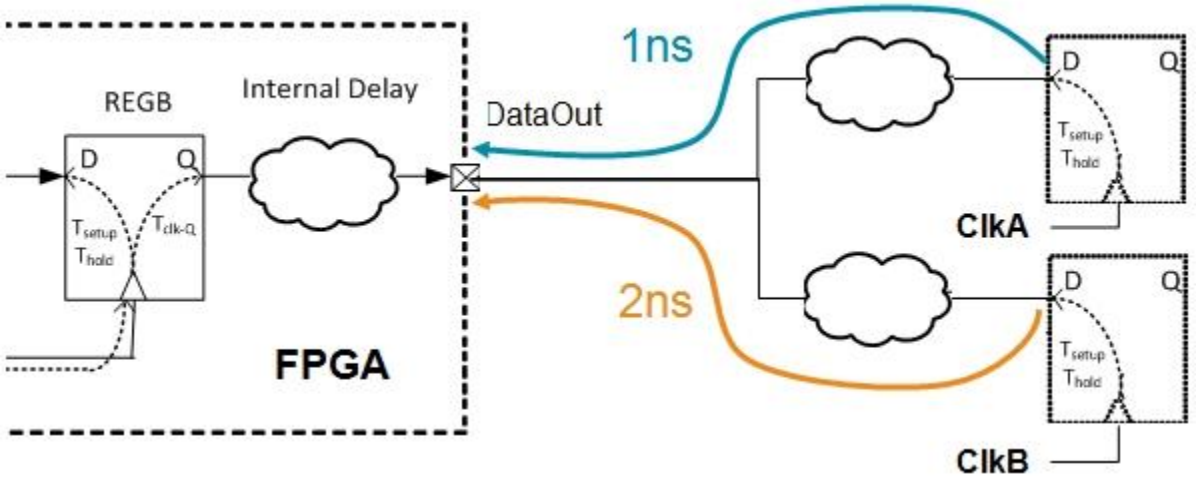
Output Hold Timing Report Detailed Paths



Multiple Output Delays on the Same Port

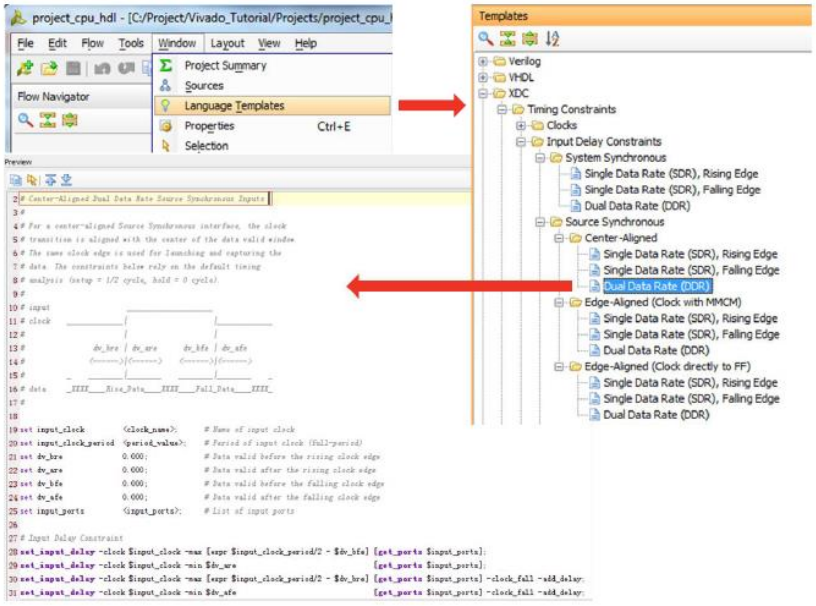
- **An output can have multiple `set_output_delay` commands associated with it**
  - Use the `–add_delay` option
  - Results in multiple static timing paths to check

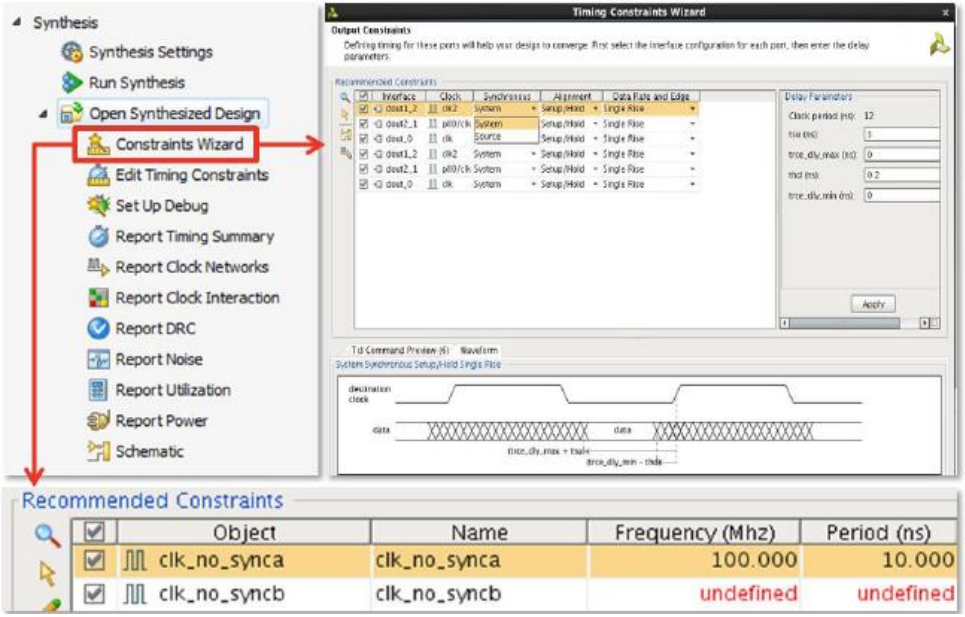
```
set_output_delay -clock ClkA 1 [get_ports DataOut]
set_output_delay -clock ClkB 2 [get_ports DataOut] -add_delay
```



在Vivado 中设置接口约束

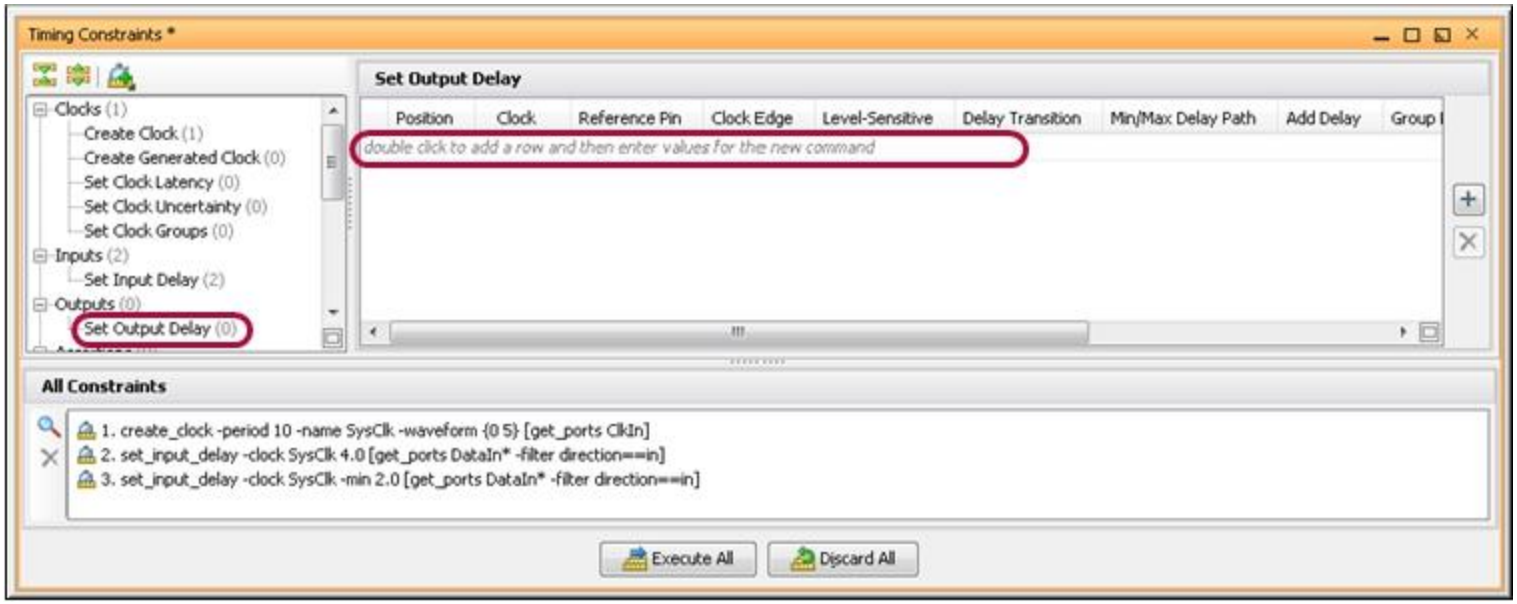
- 参照Vivado IDE 的Language Templates
- Constraints Wizard 的调出方法和界面





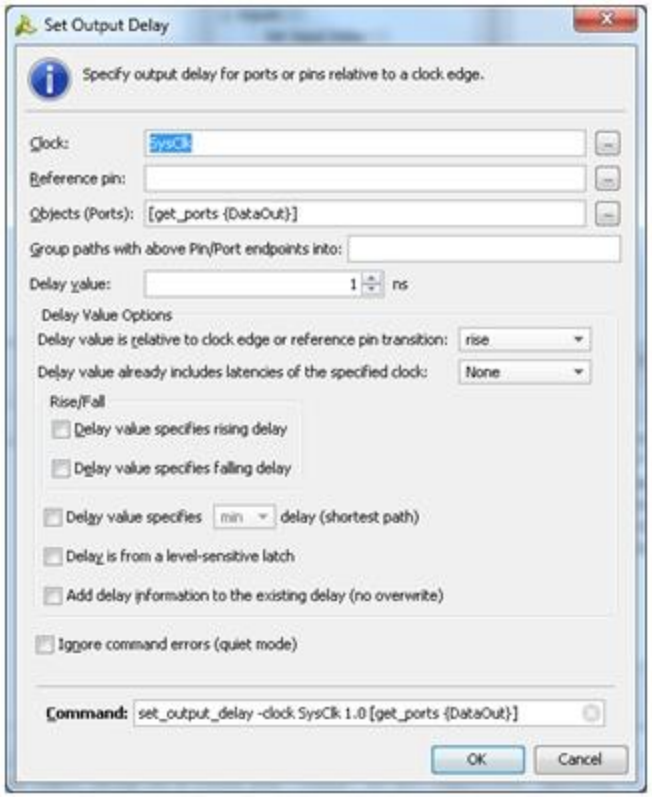
Creating Output Delays Using the GUI

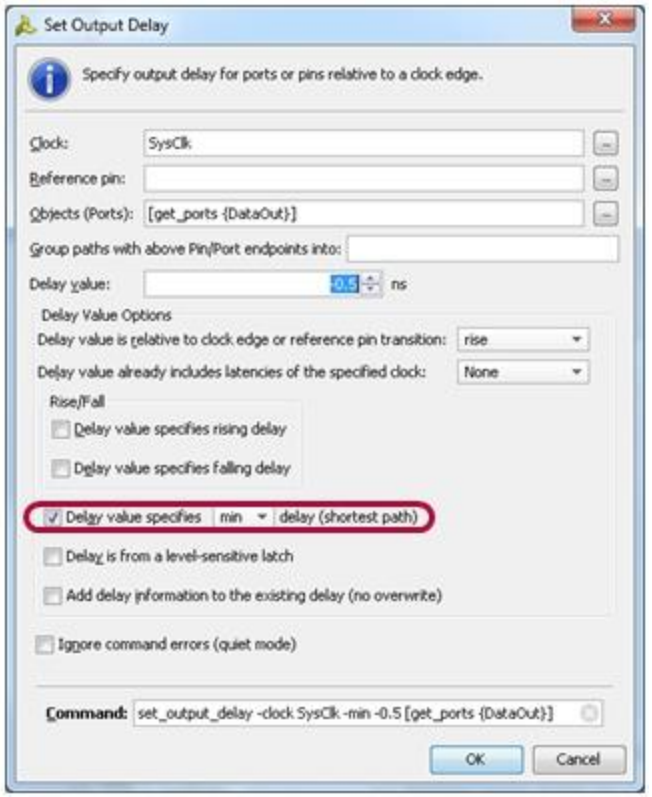
- **Timing Constraint window can be opened by selecting Window > Timing Constraints**
  - Clock can be created by double-clicking **Set Output Delay**, or a new row in the Set Output Delay table
- **Alternatively a clock can be created directly by selecting Tools > Timing > Create Timing Constraint > Output > Set Output Delay**



Set Output Delay Wizard

- **Separate constraint is required for the maximum and minimum**
  - Wizard retains its options between invocations, making it easy to specify the minimum after the maximum has been specified





Overview

- Overview
- Creating Input Delays
- Creating Output Delays
- Using Virtual Clocks

Virtual Clock

- 虚拟时钟Virtual Clock
  - 上游器件送入的数据并不是跟某个FPGA 中已经存在的真实的时钟相关，而是来自于一个不同的时钟，这时就要用到虚拟时钟（Virtual Clock）
  - 举例，上游器件用一个100MHz 的时钟送出数据到FPGA，实际上这个数据每两个时钟周期才变化一次，所以可以用50MHz 的时钟来采样。
  - FPGA 有个100MHz 的输入时钟，经过MMCM 产生一个50MHz 的衍生时钟，并用其来采样上游器件送来的同步数据。
  - 当然，系统级的设计上，必须有一定的机制来保证上游器件中的发送时钟和FPGA 中的接收时钟的时钟沿对齐。

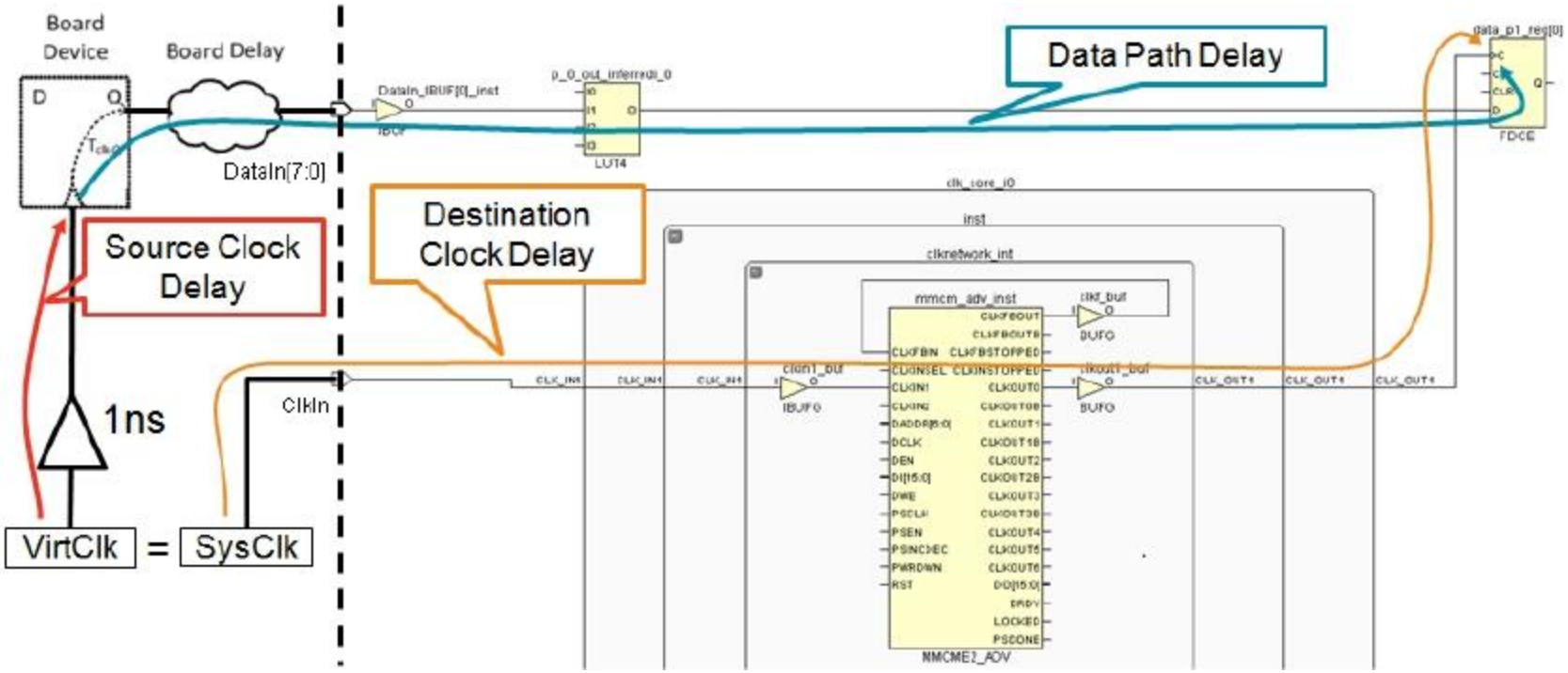
```
create_clock -period 10 -name clk_100 [get_ports i_clk_100MHz] ;
create_clock -period 20 -name clk_50_virtual ;
set_input_delay -max 5.2 -clock clk_50_virtual [get_ports i_data_50] ;
set_input_delay -min 2.0 -clock clk_50_virtual [get_ports i_data_50] ;
```

Showing Virtual Clocks

```
report_clocks
****************************************************************************************************
* Report  : Clocks
* Design  : sdr_example_design
* Part    : Device=7k70t, Package=fbg484, Speed=-1
* Version : Vivado v2012.2.TSC (64-bit) Build 183099 by xbuild on Sat May 19 01:25:01 MDT 2012
* Date    : Mon Jul 09 17:52:49 2012
****************************************************************************************************


Attributes
  P: Propagated
  G: Generated
  V: Virtual
  I: Inverted

Clock     Period    Waveform           Attributes  Sources
SysClk    10.00000  {0.00000 5.00000}  P           {ClkIn}
clkfbout  10.00000  {0.00000 5.00000}  P,G         {clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKFBOUT}
clkout0   10.00000  {0.00000 5.00000}  P,G         {clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0}
VirtClk   10.00000  {0.00000 5.00000}  V           {}
```

Input Static Timing Path with External Buffer

```
create_clock -name SysClk -period 10 [get_ports ClkIn]
create_clock -name VirtClk -period 10
set_clock_latency -source 1 [get_clocks VirtClk]
set_input_delay -clock VirtClk 4 [get_ports DataIn]
```



Input Setup Timing Report Summary with Virtual Clock

```
Slack (MET) :             0.902ns
  Source:                 DataIn[0]
                            (input port clocked by VirtClk  (rise@0.000ns fall@5.000ns period=10.000ns))
  Destination:            data_pl_reg[0]/D
                            (rising edge-triggered cell FDCE clocked by clkout0  (rise@0.000ns fall@5.000ns period=10.000ns))
  Path Group:             clkout0
  Path Type:              Max at Slow Process Corner
  Requirement:            10.000ns
  Data Path Delay:        1.925ns  (logic 0.847ns (43.997%)  route 1.078ns (56.003%))
  Logic Levels:           2  (IBUF=1 LUT4=1)
  Input Delay:            4.000ns
  Clock Path Skew:        -3.005ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):  -2.005ns
    Source Clock Delay      (SCD):  1.000ns
    Clock Pessimism Removal (CPR):  0.000ns
  Clock Uncertainty:      0.168ns  ((TSJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter     (TSJ):  0.050ns
    Discrete Jitter          (DJ):  0.129ns
    Phase Error              (PE):  0.099ns
```

Input Setup Timing Report Detailed Paths with Virtual Clock

Location	Delay type	Incr(ns)	Path(ns)	Netlist Resource(s)
	(clock VirtClk rise edge)	0.000	0.000 r	
	clock source latency	1.000	1.000	
	ideal clock network latency			
		0.000	1.000	
	input delay	4.000	5.000	
U17		0.000	5.000 r	DataIn[0]
U17	IBUF (Prop_ibuf_I_0)	0.829	5.829 r	DataIn_IBUF[0]_inst/0
SLICE_X0Y42	LUT4 (Prop_lut4_I1_0)	1.131	6.960 r	p_0_out_inferredi_0/0
SLICE_X0Y42	net (fo=1, routed)	0.000	6.960 r	data_pl_reg[0]/D
SLICE_X0Y42	FDCE (Setup_fdce_C_D)	-0.035	6.925	data_pl_reg[0]
	(clock clkout0 rise edge)	10.000	10.000 r	
V20		0.000	10.000 r	ClkIn
V20	IBUFG (Prop_ibufg_I_0)	0.741	10.741 r	clk_core_i0/inst/clknetwork_int/clkin1_buf/0
MMCME2_ADV_X0Y0	MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)			
		-6.583	4.158 r	clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0
BUFGCTRL_X0Y1	BUFG (Prop_bufg_I_0)	2.345	6.503 r	clk_core_i0/inst/clknetwork_int/clkout1_buf/0
SLICE_X0Y42	net (fo=4, routed)	1.492	7.995 r	data_pl_reg[0]/C
	clock pessimism	0.000	7.995	
	clock uncertainty	-0.168	7.827	
	required time		7.827	
	arrival time		-6.925	
	slack		0.902	

Source Clock Delay

Data Path Delay

Slack Calculation

Destination Clock Delay

Overview

- Overview
- Creating Input Delays
- Creating Output Delays
- Using Virtual Clocks
- lab3

Lab3 IO Constraints

Duration: 20 Minutes

5、例外时序约束Timing Exceptions

Objectives

完成这部分后主要可以掌握:

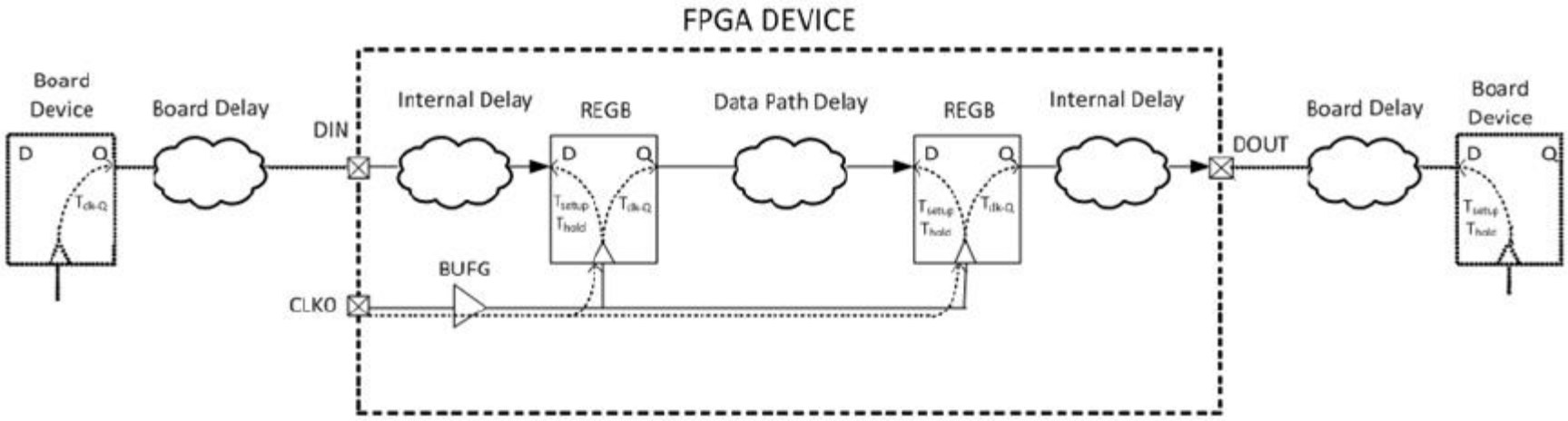
- 识别设计中那部分需要例外约束
- 掌握如何定义multicycle路径
- 掌握如何定义false path路径
- 掌握如何定义path delays路径
- 掌握约束和例外的优先级

Overview

- Overview
- Multicycle Paths
- False Paths
- Max/Min Delay Exceptions

静态时序检查Static Timing Checks

- 所有的静态时序路径都会进行Setup和hold 的检查
  - FPGA内部的所有clocked elements间的时序路径
  - 使用set_input_delay 和set_output_delay 命令描述约束的路径
- 检查时钟驱动的起点与终点之间的需求



时序例外Timing Exceptions

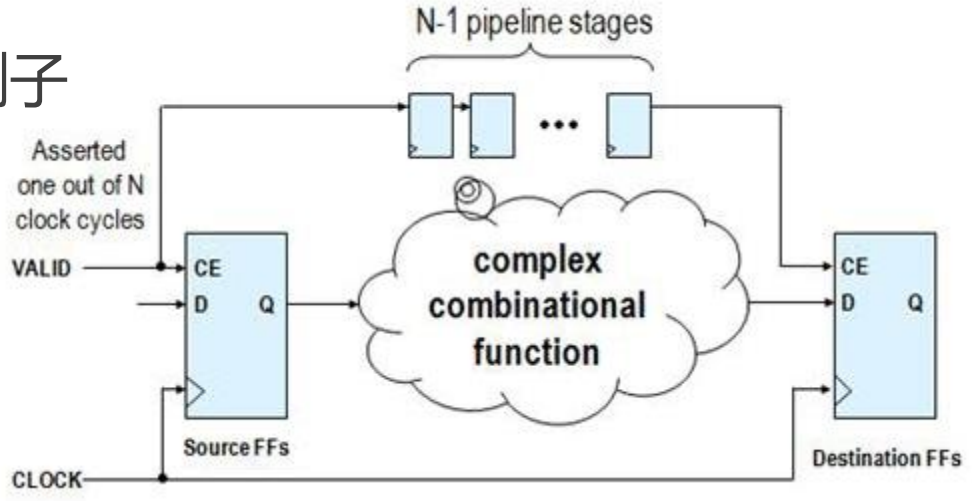
- 通用的setup 和hold检查在一部分设计中不正确的状况是存在的
  - 约束的架构的一些方面需要修改这些检查这通过时序例外约束来完成
- 时序例外Timing exceptions 修改已分析出来的setup/hold检查
  - 为设计者提供一个方法，来准确地描述在静态时序路径上的"非默认"定时关系

Multicycle Paths

- **Overview**
- **Multicycle Paths**
- **False Paths**
- **Max/Min Delay Exceptions**

Multicycle Paths

- 多周期约束改变默认的setup和hold的默认检查方式
- 设计中有些逻辑并不要求在单周期内完成稳定采样，在寄存器间不是按照连续时钟周期改变的时候就多时钟周期路径问题
  - 通常出现在存在时钟使能信号的情况
  - 从源寄存器、目的寄存器要求数据更新的时间是N个时钟周期
- 可以使得工具关注于更加难满足timing的path，便于时序收敛；工具更加关注于真实的关键路径。
- Reduce total Runtime.
- 下面就是一个N个时钟周期的例子



创建多周期路径例外Creating a Multicycle Path Exception

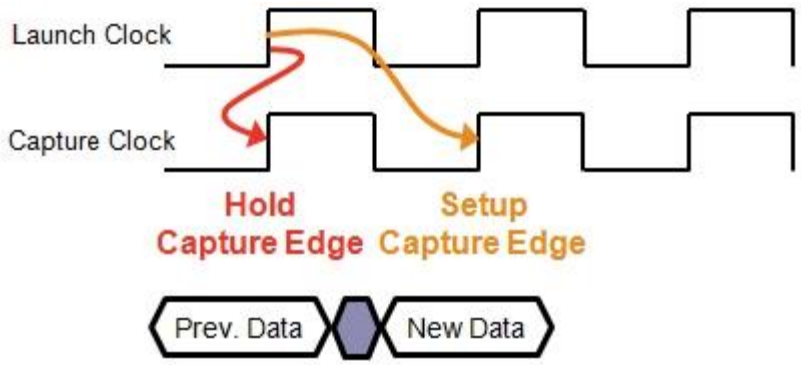
- 多周期路径通过set_multicycle_path命令进行定义
  - `set_multicycle_path <path_enumeration> <check_type> \<multiplier>`
    - `<path_enumeration>` 标志识别哪些路径使用 `set_multicycle_path` 命令
    - o Uses the `-from`, `-through`, and `-to` options (among others)
    - `<check_type>` 决定静态时序检查的类型
    - o `-setup` or `-hold`
    - `<multiplier>` 决定multicycle 路径有多少个时钟周期

  – set_multicycle_path 2 -setup -from [get_pinsdata0_reg/C] -to [get_pins data1_reg/D]
    set_multicycle_path 1 -hold -from [get_pinsdata0_reg/C] -to [get_pins data1_reg/D]

常规的建立保持时间检查Normal Setup and Hold Checks

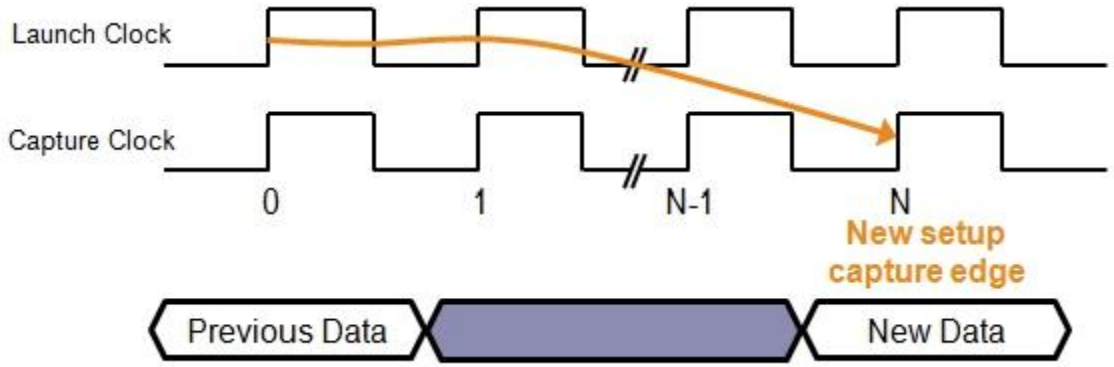
- 静态时序路径就是从一个时钟的元素开始和结束，并且是同一个时钟驱动的周期内
  - Setup check is performed at destination 1 clock cycles after the launch clock edge
  - Hold check is performed at destination 0 clock cycles after the launch clock

- 因此针对 multicycle的路径就要重新定义launch edges和capture



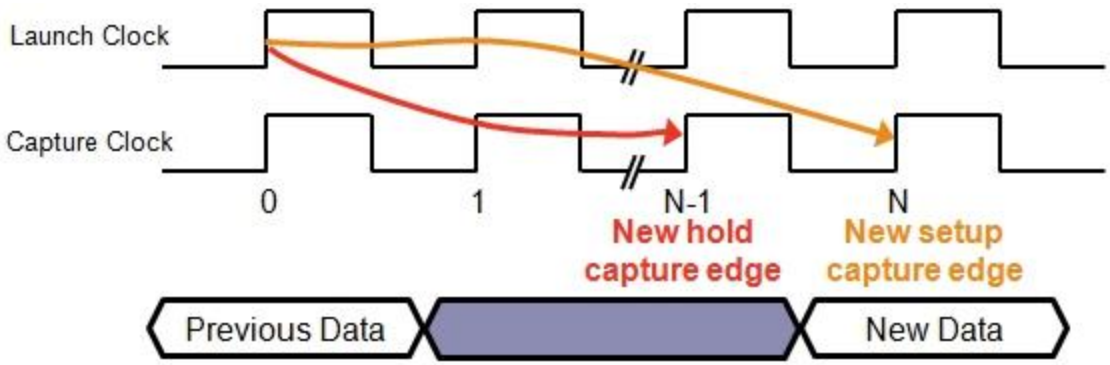
Modifying the Setup Capture Edge

- 为了允许更长的传输时延setup capture edge 需要作出修改
- set_multicycle_path –from $from_list –to $to_list <N>
  - 新的捕获沿setup capture edge 将变成发射沿launch edge后的第 <N>
  - For normal paths N=1 正常不约束是实际上默认为1



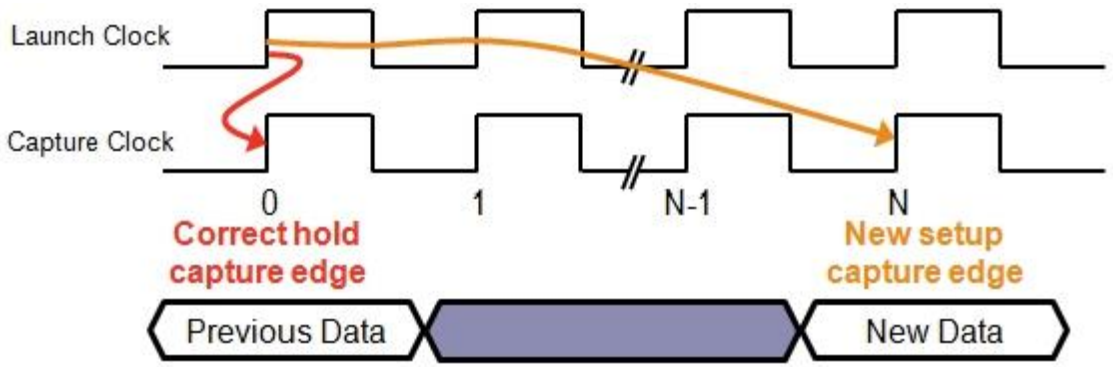
Consequences on the Hold Capture Edge

- 默认情况下hold capture edge捕获沿仅比setup capture edge早一个时钟周期
  - 所以下面这种情况不是正确的检查关系；
  - 这种情况下基本上都是会失败的。



Modifying the Hold Capture Edge

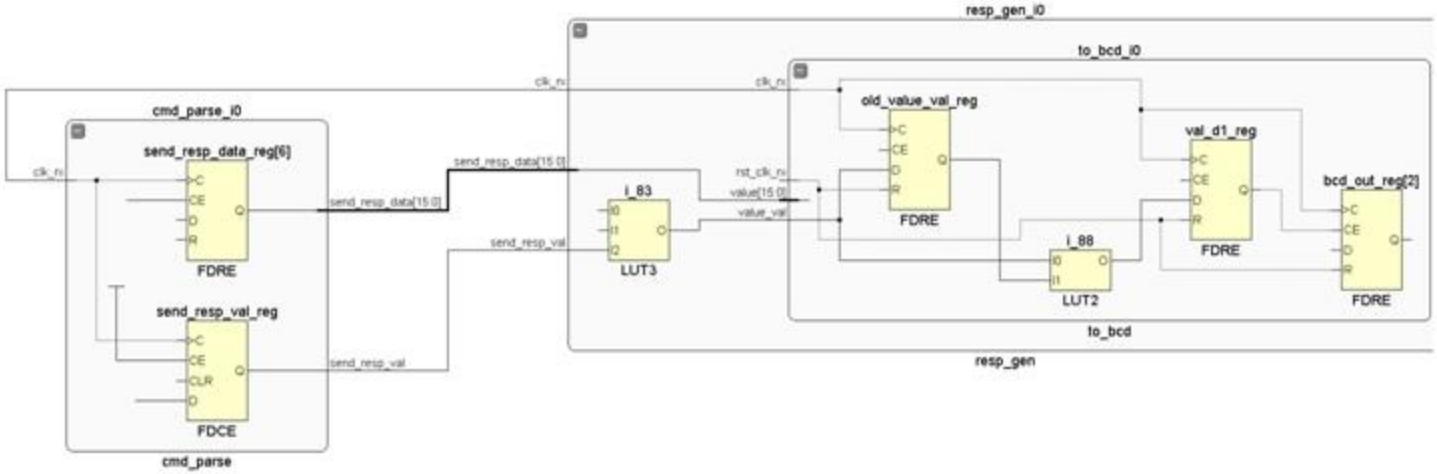
- hold capture edge需要作出修改
  - `set_multicycle_path –hold` **command**
  - **set_multicycle_path –from $from_list –to $to_list –hold <N-1>**



Multicycle Path Example

- **Signal send_resp_data changes infrequently**
  - Signal send_resp_val indicates when send_resp_data is valid
    - After edge detection and one cycle of delay, this is the enable for bcd_out which captures the result of the BCD calculation of send_resp_data
- **Intentionally coded two-cycle, multicycle path**

```
create_clock -name my_clk -period 5 [get_ports clk_pin_p]
set_multicycle_path -from [get_cells
cmd_parse_i0/send_resp_data_reg[*]] \
-to [get_cells resp_gen_i0/to_bcd_i0/bcd_out_reg[*]] 2
```



Multicycle Path Timing Report Summary

- Due to the `set_multicycle_path` command, the requirement for this path is twice the period of the clock

```
Slack (MET) :             2.198ns
  Source:                 cmd_parse_i0/send_resp_data_reg[6]/C
                            (rising edge-triggered cell FDRE clocked by clkout0  {rise@0.000ns fall@2.500ns period=5.000ns})
  Destination:            resp_gen_i0/to_bcd_i0/bcd_out_reg[2]/D
                            (rising edge-triggered cell FDRE clocked by clkout0 {rise@0.000ns fall@2.500ns period=5.000ns})
  Path Group:             clkout0
  Path Type:              Max at Slow Process Corner
  Requirement:            10.000ns
  Data Path Delay:        7.641ns  (logic 2.159ns (28.256%)  route 5.482ns (71.744%))
  Logic Levels:           19  (CARRY4=6 LUT2=2 LUT3=1 LUT4=1 LUT5=1 LUT6=8)
  Clock Path Skew:        -0.104ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):  -1.718ns
    Source Clock Delay      (SCD):  -1.757ns
    Clock Pessimism Removal (CPR):  -0.143ns
  Clock Uncertainty:      0.057ns  ((TSJ^2 + DJ^2)^1/2) / 2 + PE
    Total System Jitter     (TSJ):  0.071ns
    Discrete Jitter          (DJ):  0.090ns
    Phase Error              (PE):  0.000ns
  Timing Exception:       MultiCycle Path   Setup -end   2
```

Multicycle Path Timing Report SCD and DP

- **The source clock and datapath delays are the same as a single cycle path**
  - The datapath delay is long due to the large number of cells in the path

Location	Delay type	Incr(ns)	Path(ns)	Netlist Resource(s)
	(clock clkout0 rise edge)	0.000	0.000 r	
AD12		0.000	0.000 r	clk_pin_p
AD12	IBUFGDS (Prop_ibufgds_I_0)			
		0.917	0.917 r	clk_gen_i0/clk_core_i0/inst/clknetwork_int/clkin1_buf/0
MMCME2_ADV_X1Y1	MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)			
		-6.769	-5.852 r	clk_gen_i0/clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0
BUFGCTRL_X0Y2	BUFG (Prop_bufg_I_0)	2.444	-3.408 r	clk_gen_i0/clk_core_i0/inst/clknetwork_int/clkout1_buf/0
SLICE_X7Y54	net (fo=258, routed)	1.651	-1.757 r	cmd_parse_i0/send_resp_data_reg[6]/C
SLICE_X7Y54	FDRE (Prop_fdre_C_Q)	0.223	-1.534 r	cmd_parse_i0/send_resp_data_reg[6]/Q
SLICE_X13Y57	LUT4 (Prop_lut4_I1_0)	0.684	-0.850 r	resp_gen_i0/to_bcd_i0/i_112/0
SLICE_X12Y57	LUT6 (Prop_lut6_I1_0)	0.422	-0.428 r	resp_gen_i0/to_bcd_i0/i_90/0
⋮	⋮	⋮	⋮	
SLICE_X15Y60	LUT6 (Prop_lut6_I0_0)	0.577	5.919 r	_gen_i0/to_bcd_i0/p_10_out_inferredi_42/0
SLICE_X15Y60	net (fo=1, routed)	0.000	5.919 r	resp_gen_i0/to_bcd_i0/bcd_out_reg[2]/D
SLICE_X15Y60	FDRE (Setup_fdre_C_D)	-0.035	5.884	resp_gen_i0/to_bcd_i0/bcd_out_reg[2]

Arrival Time

Multicycle Path Timing Report DCD and Slack

- **The destination clock delay starts at the requirement**
  - Twice the clock period due to the multicycle path

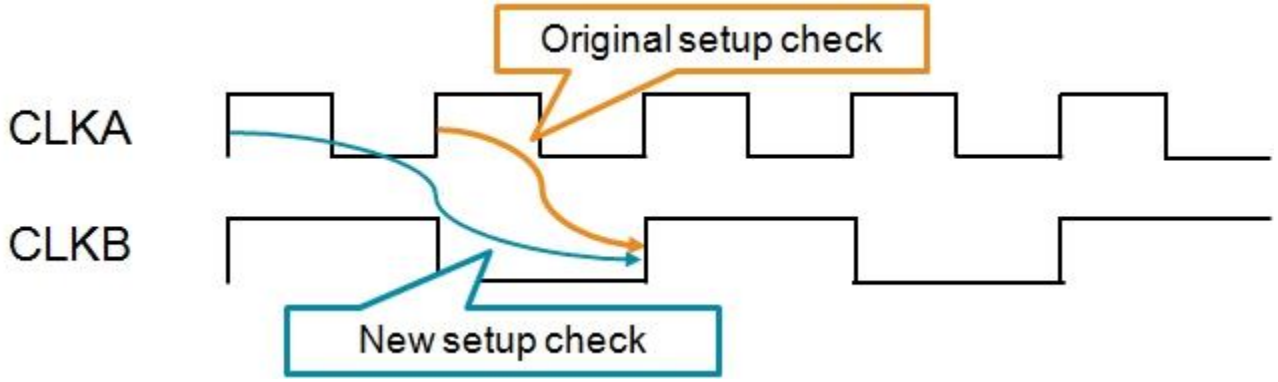
```
                     (clock clkout0 rise edge)   10.000     10.000 r
  AD12                                            0.000     10.000 r  clk_pin_p
  AD12               IBUFGDS (Prop_ibufgds_I_0)
                                                 0.835     10.835 r  clk_gen_i0/clk_core_i0/inst/clknetwork_int/clkin1_buf/O
  MMCME2_ADV_X1Y1    MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
                                                -6.193      4.642 r  clk_gen_i0/clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0
  BUFGCTRL_X0Y2      BUFG (Prop_bufg_I_0)        2.221      6.863 r  clk_gen_i0/clk_core_i0/inst/clknetwork_int/clkout1_buf/O
  SLICE_X15Y60       net (fo=258, routed)        1.419      8.282 r  resp_gen_i0/to_bcd_i0/bcd_out_reg[2]/C
                     clock pessimism            -0.143      8.139
                     clock uncertainty          -0.057      8.082
-------------------------------------------------------------------
                     required time                          8.082
                     arrival time                          -5.884
-------------------------------------------------------------------
                     slack                                  2.198
```

Required Time

Modifying the Launch Edge

- **By default the `set_multicycle_path` command moves the capture edge forward as specified by the multiplier**
  - This is equivalent to using the `-end` option
- **In some circumstances it is necessary to move the launch edge backwards instead**
  - This can be the case when the source and destination clocks are not the same frequency
    - Use the `-start` option

```
set_multicycle_path -from [get_clocks CLKA] \
-to [get_clocks CLKB] -start 2
```



Overview

- Overview
- Multicycle Paths
- False Paths
- Max/Min Delay Exceptions

Disabling Setup and Hold Checks

- 默认情况下所有建立及保持时间的静态时序路径都会进行检查
- 有些静态时序路径的检查是没有必要的
  - 常量及伪常量信号不需要分析
  - 常量或者伪常量通过系统
  - 互斥的子路径或者互斥的时钟
  - 使用异步时钟
- 在这种情况下，需要通过一种办法让工具不分析这些静态时序路径

Creating a False Path Exception

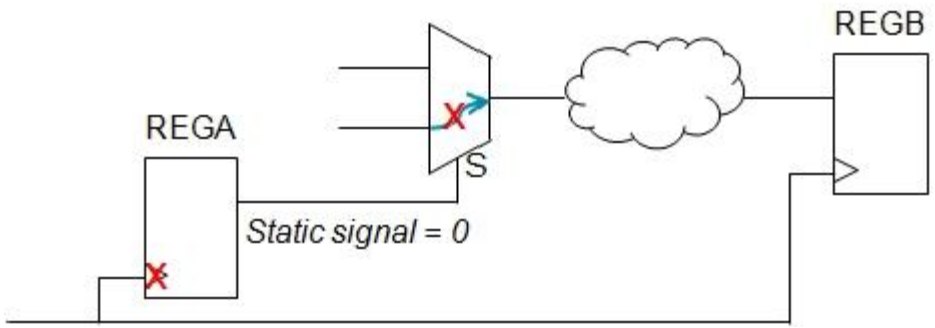
- 通过**set_false_path** 命令创建
  - set_false_path <path_enumeration> <check_type>
  - <path_enumeration>使用制定路径
  - <check_type> 制定类型-setup 或者 –hold
  - 不指定时建立setup和保持hold检查都受到影响

  - set_false_path –through lut/O

常量或者准常量Constants and Pseudo-Constants

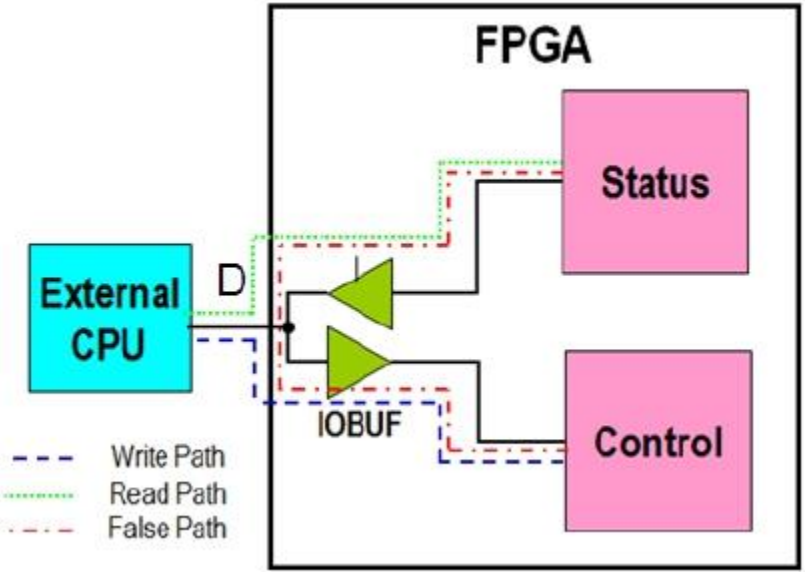
- 设计中信号不需要进行任何转换传输的
- 当net是一个常量时，有些时序可以不分析

```
-set_false_path -from [get_cells REGA]
-set_false_path -through [get_pins MUXA/I1]
```



互斥子路径Mutually Exclusive Sub-Paths

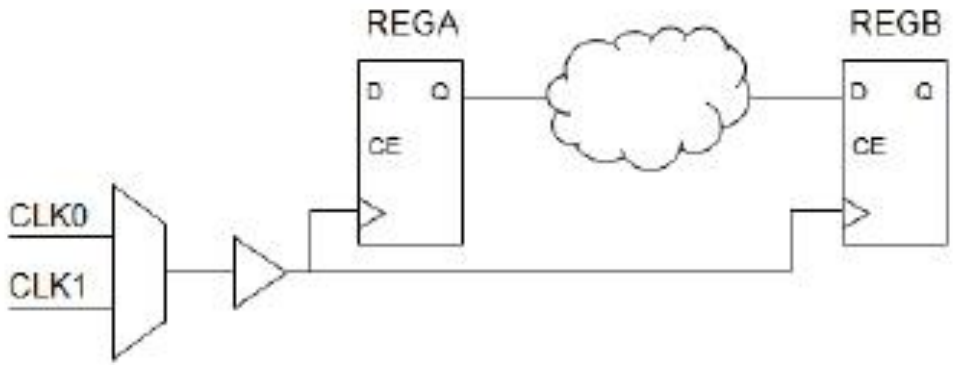
- 两个路径是互斥的
  - 常常出现在双端口的接口上
  - `set_false_path from [get_cells Status/*] through [get_ports D[*]] to [get_cells Control/*]`



Exclusive Clock Groups

- 静态时序路径static timing path分析所有时钟下的路径
- 可以通过 set_false_path 或者set_clock_groups 命令
  - set_false_path –from [get_clocks CLK0] –to [get_clocks CLK1]
  - set_false_path –from [get_clocks CLK1] –to [get_clocks CLK0]

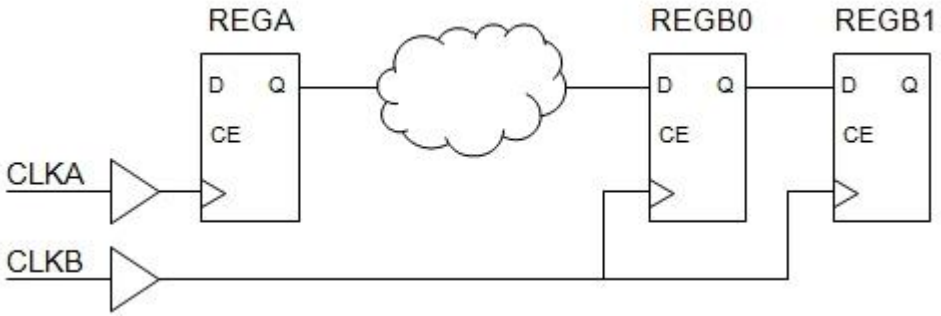
  或者
  - set_clock_groups –physically_exclusive –group CLK0 –group CLK1



Asynchronous Clock Groups

- 当两个时钟之间存在路径时，the launch 和capture edges的建立 setup检查要求是非常严格的

```
-set_clock_groups  -asynchronous -group CLK0 -group CLK1
-set_false_path -[get_clocks CLK0 ]-to[get_clocks   CLK1]
```



False Path Timing Report

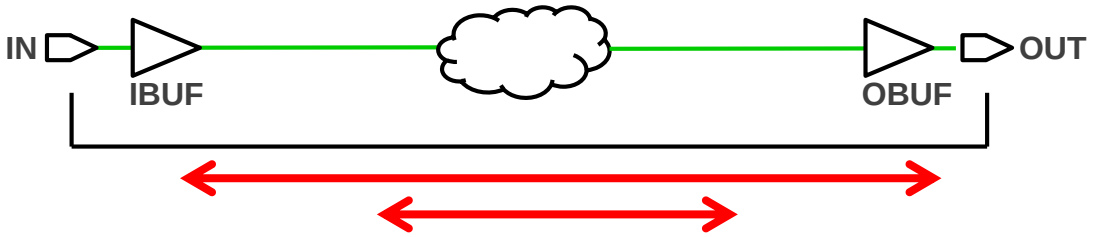
- **False paths will not show up in any timing report unless a `report_timing` command specifically enumerates a false path**
  - If explicitly enumerated, the slack will be reported as `inf`

```
Slack:                    inf
  Source:                 data_p1_reg[0]/C
                            (rising edge-triggered cell FDCE clocked by clkout0  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            data_p2_reg[0]/D
  Path Group:             (none)
  Path Type:              Max at Fast Process Corner
  Data Path Delay:        0.517ns  (logic 0.124ns (23.973%)  route 0.393ns (76.027%))
  Logic Levels:           0
  Clock Path Skew:        0.470ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    0.000ns
    Source Clock Delay      (SCD):    -0.470ns
    Clock Pessimism Removal (CPR):    0.000ns
  Timing Exception:       False Path

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
                         (clock clkout0 rise edge)    0.000     0.000 r
    V20                                               0.000     0.000 r  ClkIn
    V20                  IBUFG (Prop_ibufg_I_O)       0.545     0.545 r  clk_core_i0/inst/clknetwork_int/clkin1_buf/O
    MMCME2_ADV_X0Y0      MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
                                                     -2.719    -2.174 r  clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0
    BUFGCTRL_X0Y1        BUFG (Prop_bufg_I_O)         0.843    -1.331 r  clk_core_i0/inst/clknetwork_int/clkout1_buf/O
    SLICE_X0Y42          net (fo=4, routed)           0.861    -0.470 r  data_p1_reg[0]/C
-------------------------------------------------------------------    -------------------
    SLICE_X0Y42          FDCE (Prop_fdce_C_Q)         0.124    -0.346 r  data_p1_reg[0]/Q
    SLICE_X1Y42          net (fo=1, routed)           0.393     0.047 r  data_p2_reg[0]/D
    SLICE_X1Y42          FDCE                         0.000     0.047    data_p2_reg[0]
-------------------------------------------------------------------    -------------------


                         (clock clkout0 rise edge)    0.000     0.000 r
    V20                                               0.000     0.000 r  ClkIn
    V20                  IBUFG (Prop_ibufg_I_O)       0.375     0.375 r  clk_core_i0/inst/clknetwork_int/clkin1_buf/O
    MMCME2_ADV_X0Y0      MMCME2_ADV (Prop_mmcme2_adv_CLKIN1_CLKOUT0)
                                                     -2.164    -1.789 r  clk_core_i0/inst/clknetwork_int/mmcm_adv_inst/CLKOUT0
    BUFGCTRL_X0Y1        BUFG (Prop_bufg_I_O)         0.761    -1.028 r  clk_core_i0/inst/clknetwork_int/clkout1_buf/O
    SLICE_X1Y42          net (fo=4, routed)           0.648    -0.380 r  data_p2_reg[0]/C
```

Overview

- Overview
- Multicycle Paths
- False Paths
- Max/Min Delay Exceptions

Combinational Delays



slowest **set_max_delay 8.0 –from [get_ports {IN}] –to [get_ports{OUT}]**

fastest **set_min_delay 2.0 –from [get_ports {IN}] –to [get_ports {OUT}]**

Timing Report from Port to Port

```
Slack (MET) :           0.900ns
  Source:               CombIn[0]
                            (input port)
  Destination:          CombOut
  Path Group:           **default**
  Path Type:            Max at Slow Process Corner
  Requirement:          8.000ns
  Data Path Delay:      7.100ns  (logic 4.903ns (69.060%)  route 2.197ns (30.940%))
  Logic Levels:         3  (IBUF=1 LUT2=1 OBUF=1)
  Output Delay:         0.000ns
  Timing Exception:     MaxDelay Path 8.000ns

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
    V22                                               0.000     0.000 r  CombIn[0]
    V22                  IBUF (Prop_ibuf_I_0)         0.829     0.829 r  CombIn_IBUF[0]_inst/0
    SLICE_X0Y46          LUT2 (Prop_lut2_I1_0)        1.091     1.920 r  i_3/0
    T21                  OBUF (Prop_obuf_I_0)         5.179     7.100 r  CombOut_OBUF_inst/0
    T21                  net (fo=0)                   0.000     7.100 r  CombOut
  -------------------------------------------------------------------    -------------------

                         max delay                    8.000     8.000
                         clock pessimism              0.000     8.000
                         output delay                -0.000     8.000
  -----------------------------------------------------------------------
                         required time                          8.000
                         arrival time                          -7.100
  -----------------------------------------------------------------------
                         slack                                  0.900
```

Lab4 Timing Exceptions

Duration: 20 Minutes

6、时序收敛Synthesis/RTL优化技术

节制使用综合指令

- 需要重新检查其他综合工具或者旧版本设置的属性
- 避免使用KEEP, dont_touch, syn_preserve, max_fanout attributes…
- 参考“Vivado and UltraFast in Competitive Situations”
  https://thesource.gosavo.com/Document/Document.aspx?id=33778929&view

时序例外比对检查Timing Exception Comparison

- 有些用户拒绝使用false path timing exceptions
  - They prefer set_max_delay –datapath_only exceptions between asynchronous clock domains对于异步时钟域使用max delay约束
- 这会导致一些额外的消耗…
  - QoR cost – Tool expends effort to close timing paths that are false
  - Runtime cost – Tool has to process and deal with additional timing constraints when set_max_delay –datapath_only is used exclusively

Design Runs

Name	Description	Elapsed	Constraints	Progress	WNS	TNS
impl_11	fifo mod	03:00:55	constrs_2	100%	-0.226	-2.977
impl_12	change cdc script to false path instead of max_delay	01:33:33	constrs_2	100%	-0.074	-0.184

Poll Question

- What kind of reset is this code snippet below?

a. Synchronous

b. Asynchronous

```
always @(posedge clk or negedge reset_n)
  if(~reset_n)
```

OR

```
always @(posedge clk or posedge reset)
  if(reset)
```

Resets

- Avoid Resets, if possible
- Synchronous resets are preferred
- Don’t mix resets in a design
  - Active high and active low resets
  - Asynchronous and Synchronous resets
- Recommend a reset strategy to the customer
  - Global Reset
  - Modular (or Pipelined) Reset

Synchronous Reset:

```
always @(posedge clk)
  if(~reset_n)
```

```
always @(posedge clk)
  if(reset)
```

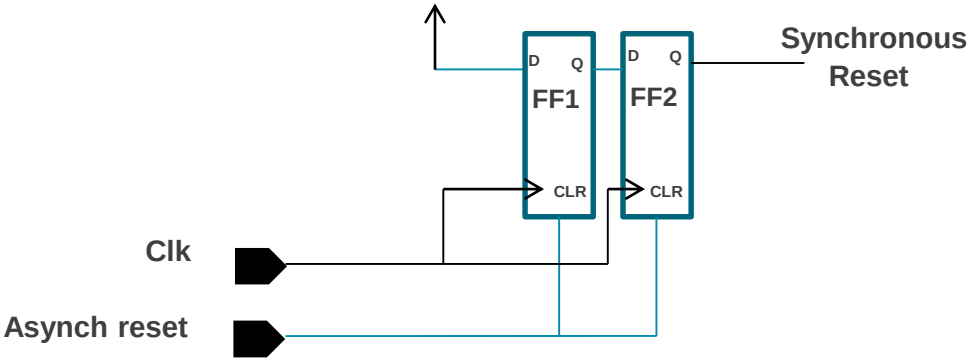
Asynchronous Reset:

```
always @(posedge clk or negedge reset_n)
  if(~reset_n)
```

```
always @(posedge clk or posedge reset)
  if(reset)
```

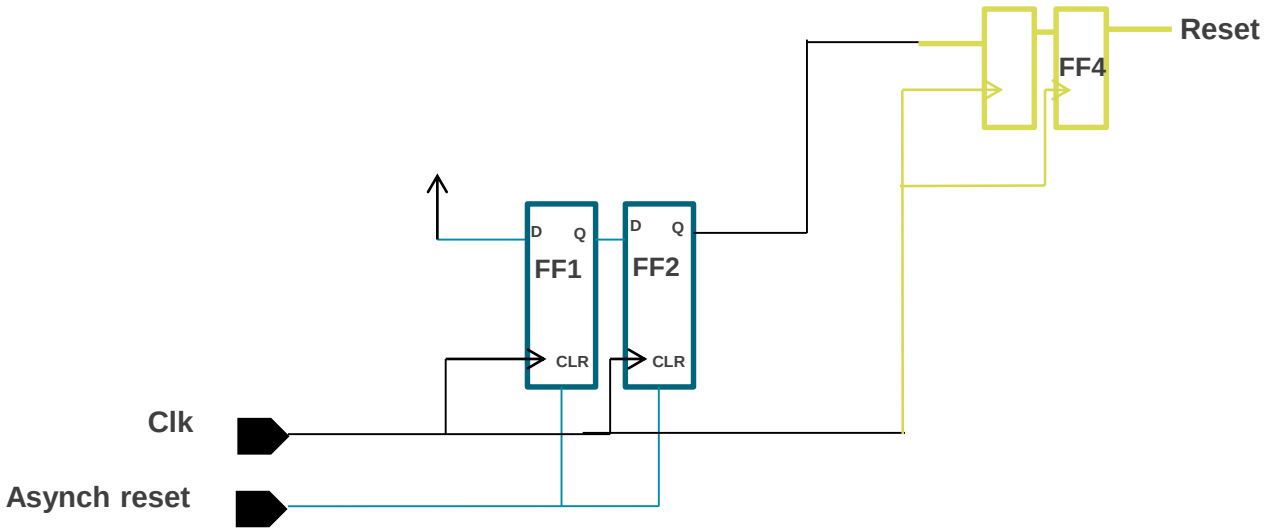
Asynchronous Reset Synchronization

- Asynchronous Reset Synchronization
  - Asynchronous Assertion
  - Synchronous Removal



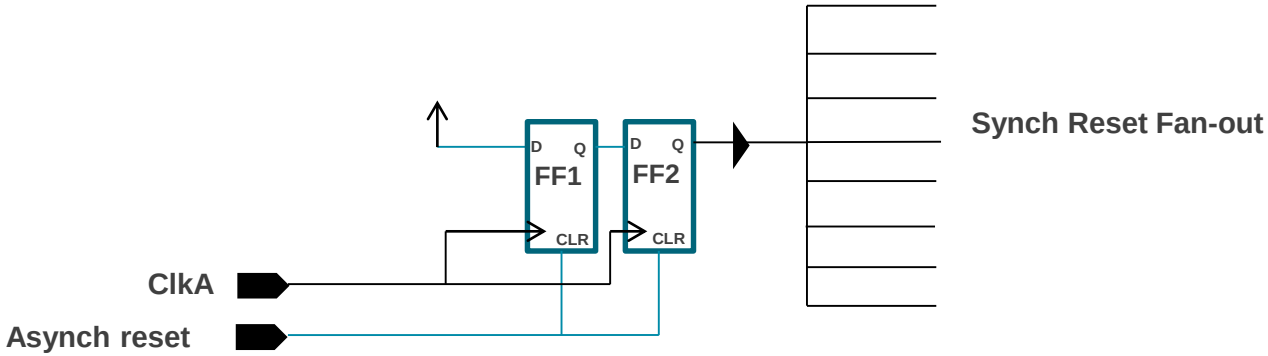
Better Reset

- Structure below gives synchronous assertion and de-assertion
  - **Great solution for the 7-series BRAM issue!**
  - **Added latency is typically not an issue**
    - **Reset is active over a number of clock cycles**



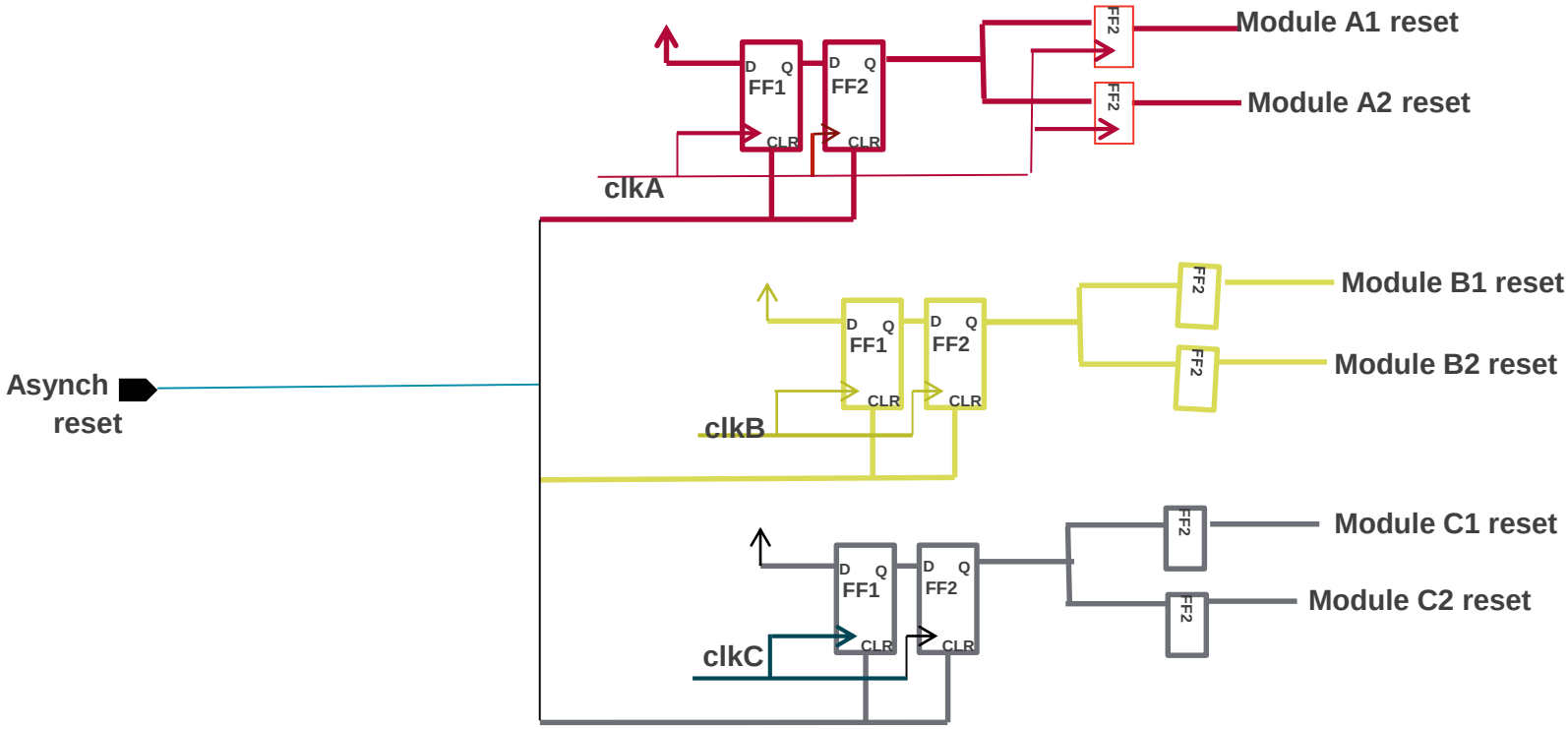
Global Reset

- Typically
  - Asynchronous reset is synchronized
  - Synchronized reset fans out to all modules in a clock domain
- Global resets might stress routing resources

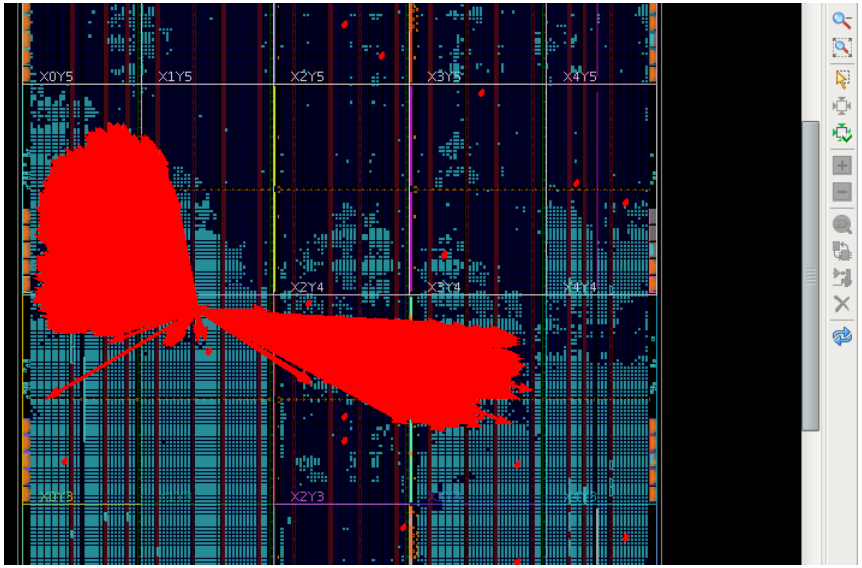


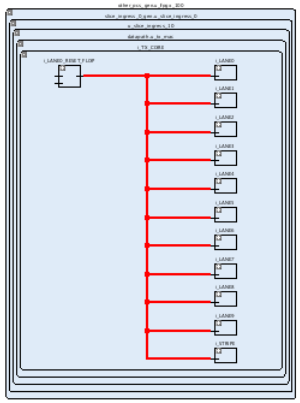
Modular Reset

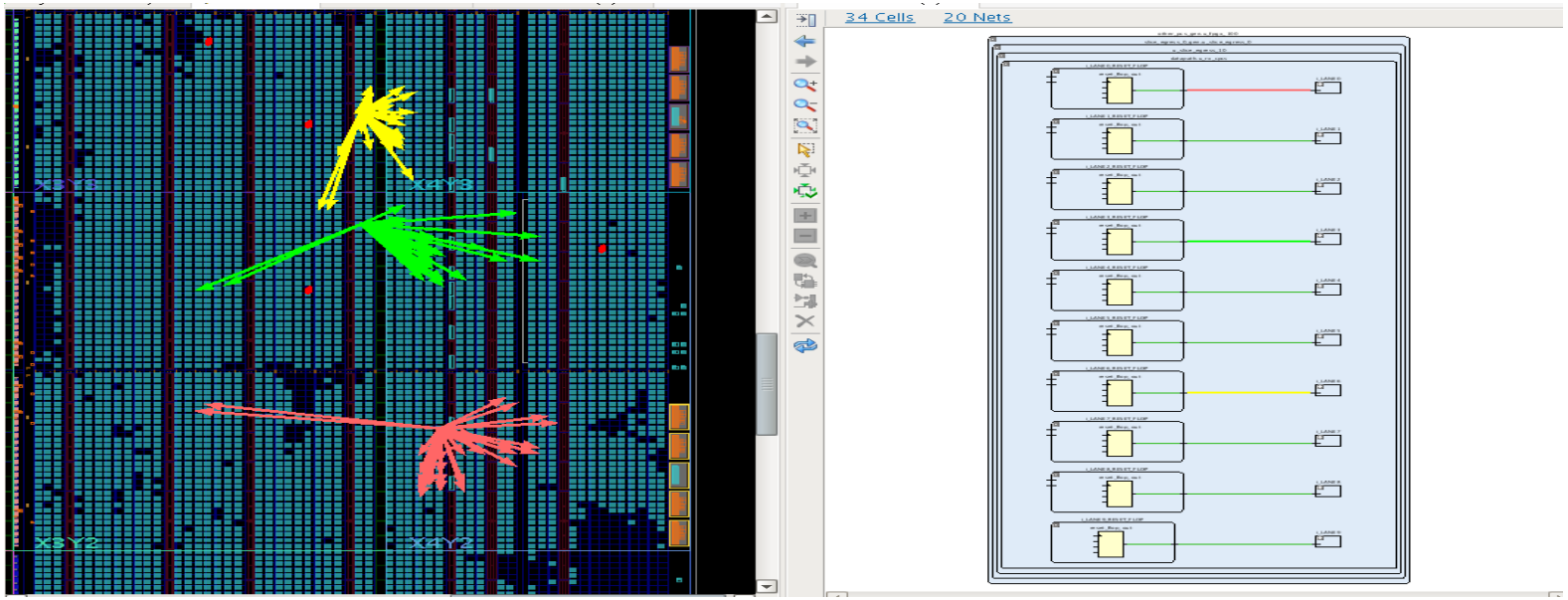
- Asynchronous reset synchronized in every clock domain
- Synchronized reset replicated and fans out in each clock domain



Global Reset vs. Modular Reset

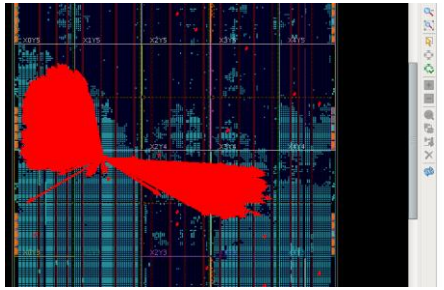


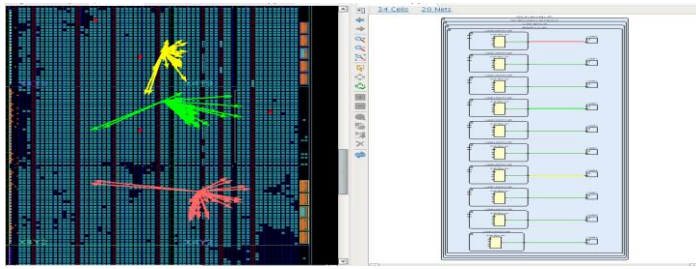




Global Reset vs. Modular Reset

- Modular resets are often designed in customer RTL, but get swallowed up in synthesis due to equivalent register removal.
  - Use KEEP (not DONT_TOUCH) to preserve modular resets in RTL





```
(* keep="true" *) reg my_modular_reset1;
(* keep="true" *) reg my_modular_reset2;
(* keep="true" *) reg my_modular_reset3;
always @(posedge clkA) begin
  my_modular_reset1 <= synchronized_reset;
  my_modular_reset2 <= synchronized_reset;
  my_modular_reset3 <= synchronized_reset;
end
```

Verify Expected Fanout of Reset Networks

- Leverage Vivado Tcl to ensure that reset networks are sized as expected
- Simple script below assumes that the reset registers have common name across hierarchy

```
# Identify Reset pins
set my_reset_pins [get_pins -hier * -filter {NAME =~ */reset_flop_out/Q}]
# For each reset pin, report the fanout
foreach reset_pin $my_reset_pins {set my_reset_net [get_nets -of $my_reset_pin]; puts "Fanout of [expr [get_property FLAT_PIN_COUNT $my_reset_net] - 1] for reset $my_reset_net}
# Result looks like this:
Fanout of 304 for reset modular_reset1
Fanout of 217 for reset modular_reset2
Fanout of 708 for reset modular_reset3
```

Pipelining

- Too many logic levels affects performance (general ideas)
  - 8 to 12 levels of logic for designs around 100MHz reasonable
  - Higher frequency typically requires lower levels of logic
  - Use report_design_analysis to review levels of logic in the design
  - Add pipeline stages to increase performance
    - New command: report_pipeline_analysis in 2015.3

Take The Survey!

Feedback:

Thank you!