Vivado从此开始（To Learn Vivado From Here）

Vivado
从此开始
To Learn Vivado From Here
高亚军 编著
· 内容翔实全面：涵盖Vivado所有基本功能
· 讲解深入浅出：结合大量案例，帮助读者加强对基本概念的理解
· 描述图文并茂：给出具体操作步骤，易于快速动手实践
中国工信出版集团 电子工业出版社

本书围绕Vivado四大主题

- 设计流程
- 时序约束
- 时序分析
- Tcl脚本的使用

作者：高亚军（Xilinx战略应用高级工程师）

- 2012年2月，出版《基于FPGA的数字信号处理（第1版）》
- 2012年9月，发布网络视频课程《Vivado入门与提高》
- 2015年7月，出版《基于FPGA的数字信号处理（第2版）》
- 2016年7月，发布网络视频课程《跟Xilinx SAE学HLS》

- ◆ 内容翔实全面：涵盖Vivado所有基本功能
- ◆ 讲解深入浅出：结合大量案例，帮助读者加强对基本概念的理解
- ◆ 描述图文并茂：给出具体操作步骤，易于快速动手实践

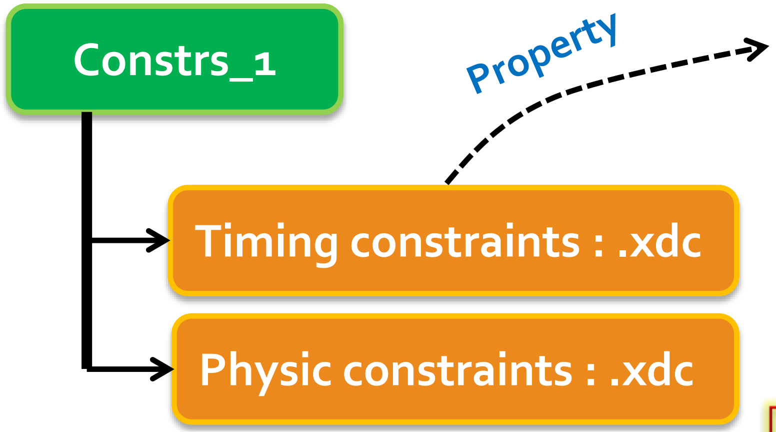
XILINX
ALL PROGRAMMABLE™

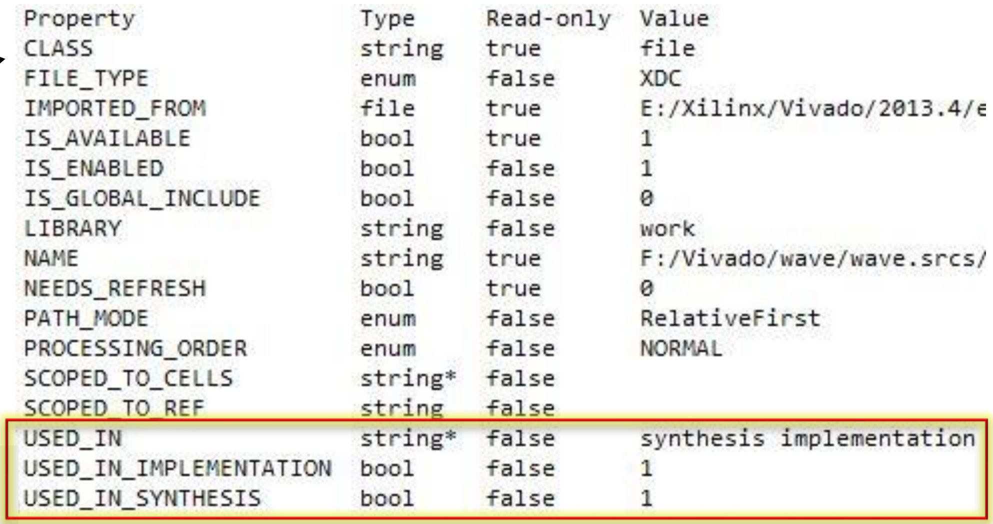
Create Basic Clock Period Constraint

Lauren Gao

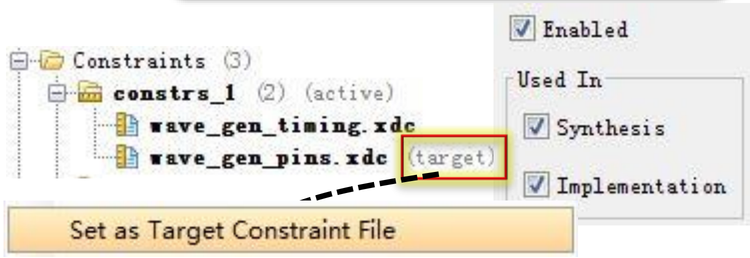
Organizing Your Constraints

Xilinx recommends that you separate timing constraints and physical constraints by saving them into two distinct files





```
Property                  Type      Read-only  Value
CLASS                     string    true       file
FILE_TYPE                 enum      false      XDC
IMPORTED_FROM             file      true       E:/Xilinx/Vivado/2013.4/e
IS_AVAILABLE              bool      true       1
IS_ENABLED                bool      false      1
IS_GLOBAL_INCLUDE         bool      false      0
LIBRARY                   string    false      work
NAME                      string    true       F:/Vivado/wave/wave.srcs/
NEEDS_REFRESH             bool      true       0
PATH_MODE                 enum      false      RelativeFirst
PROCESSING_ORDER          enum      false      NORMAL
SCOPED_TO_CELLS           string*   false
SCOPED_TO_REF             string    false
USED_IN                   string*   false      synthesis implementation
USED_IN_IMPLEMENTATION    bool      false      1
USED_IN_SYNTHESIS         bool      false      1
```

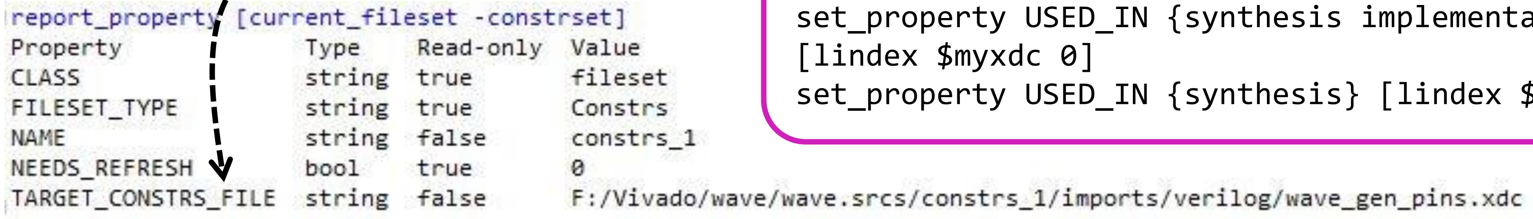


```
⊟ Constraints (3)
  ⊟ constrs_1 (2) (active)
      wave_gen_timing.xdc
      wave_gen_pins.xdc (target)
```

☑ Enabled

Used In
☑ Synthesis
☑ Implementation

Set as Target Constraint File



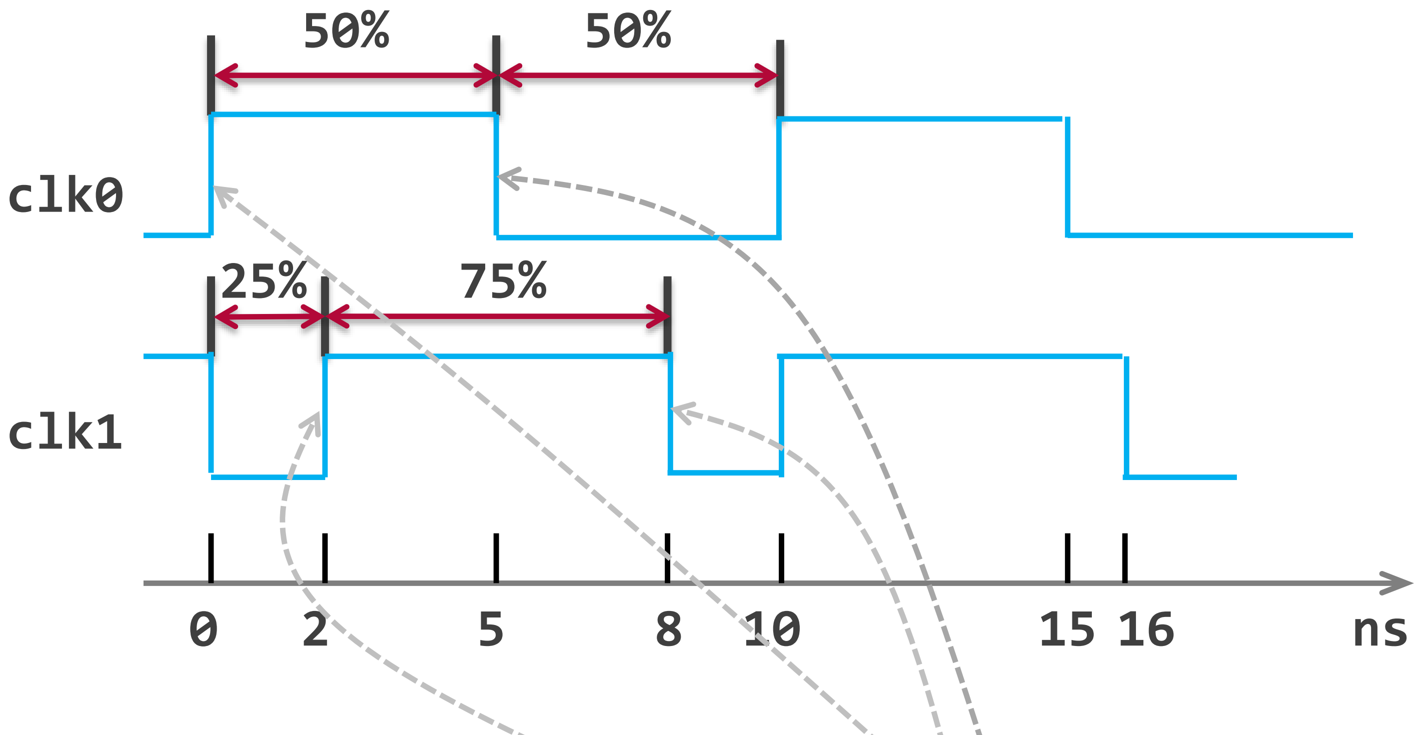
```
report_property [current_fileset -constrset]
Property            Type     Read-only  Value
CLASS               string   true       fileset
FILESET_TYPE        string   true       Constrs
NAME                string   false      constrs_1
NEEDS_REFRESH       bool     true       0
TARGET_CONSTRS_FILE string   false      F:/Vivado/wave/wave.srcs/constrs_1/imports/verilog/wave_gen_pins.xdc
```

```
set myxdc [get_files -of_objects \
[get_filesets constrs_1]]
set_property USED_IN {synthesis implementation } \
[lindex $myxdc 0]
set_property USED_IN {synthesis} [lindex $myxdc 0]
```

XILINX ➤ ALL PROGRAMMABLE.

Clock Description

- Clock period
- Duty cycle
- Phase

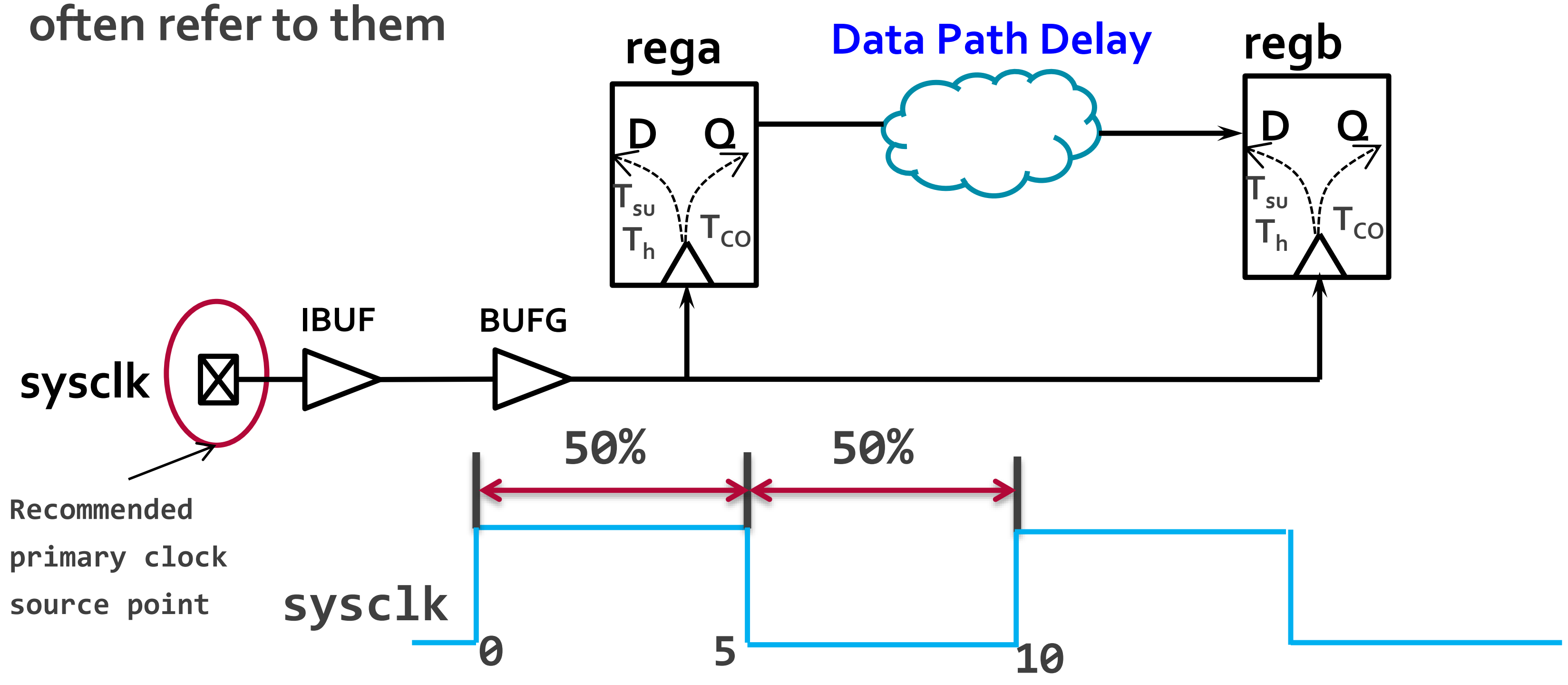


```
clk0: period = 10, waveform = {0 5}
Clk1: period = 8, waveform = {2 8}
```

XILINX ALL PROGRAMMABLE.

Primary Clock

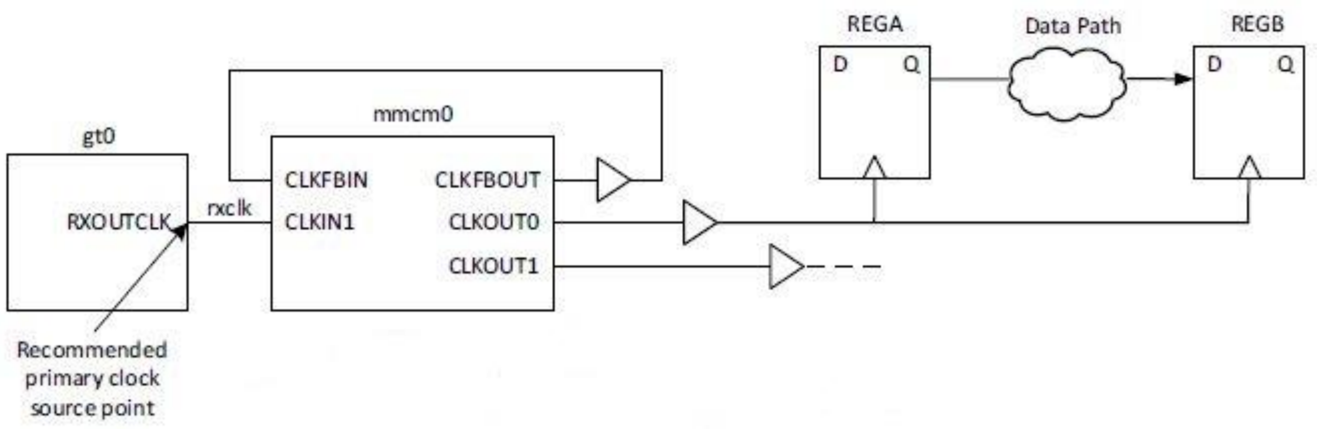
- A primary clock is a board clock that enters the design through:
  - An input port
  - A gigabit transceiver output pin (for example, a recovered clock)
- Primary clocks must be defined first,since other timing constraints often refer to them

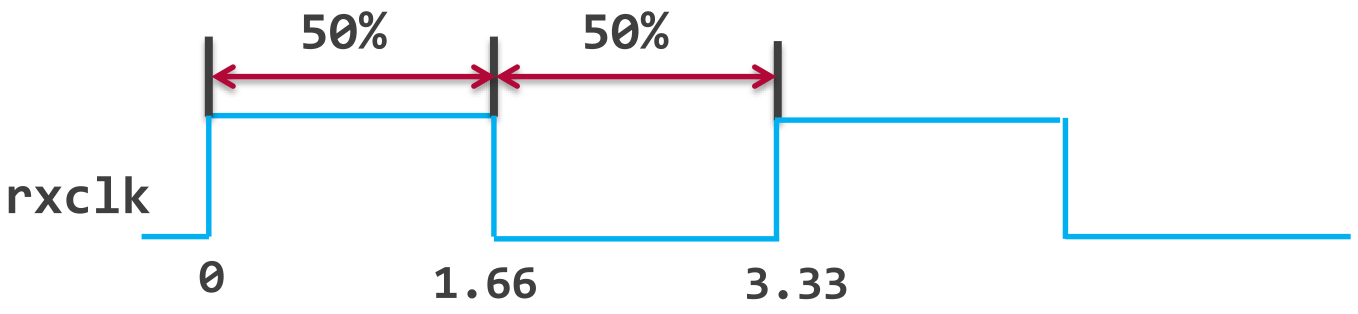


```
create_clock -period 10 [get_ports sysclk]
```

XILINX ALL PROGRAMMABLE.

Primary Clock





```
create_clock -name rxclk -period 3.33 [get_pins gt0/RXOUTCLK]
```

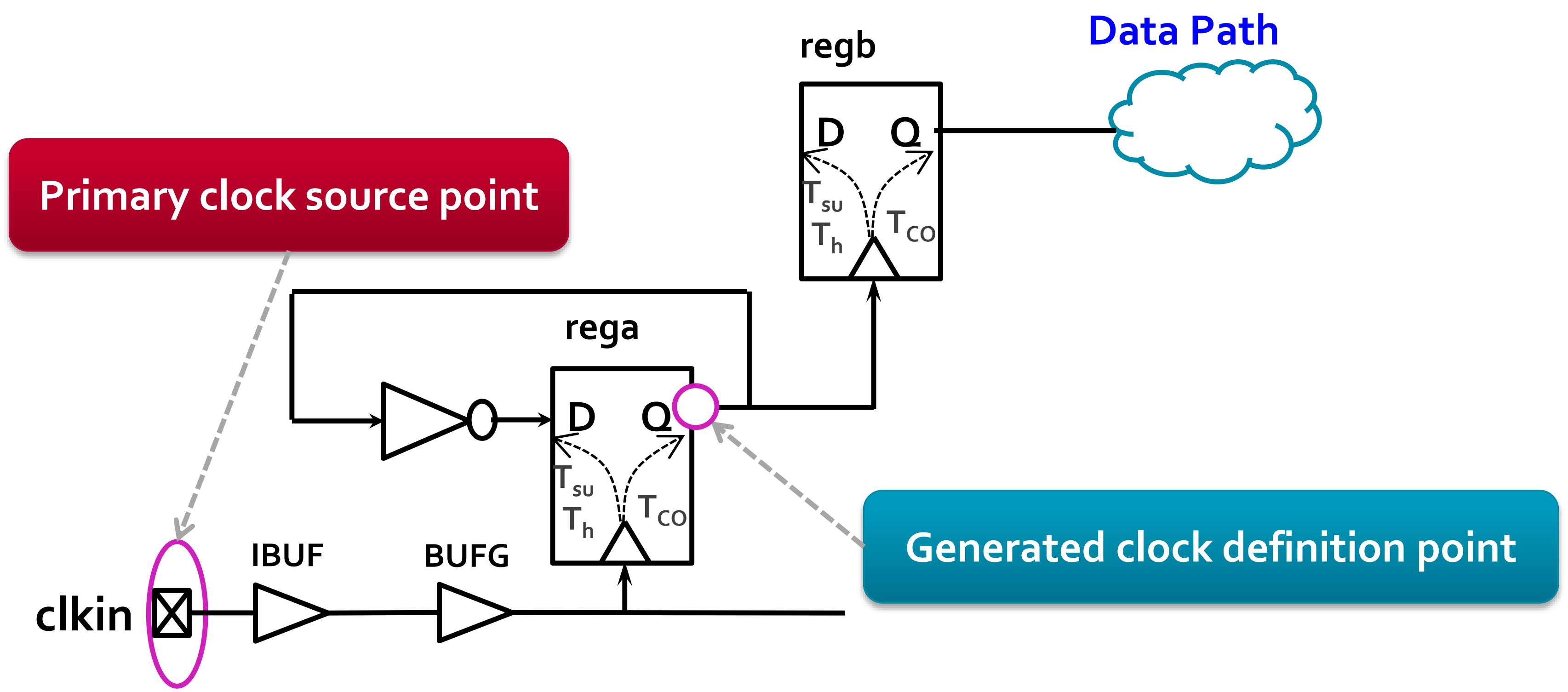
Generated Clock

- **User Defined Generated Clocks**
  - Defined by the **create_generated_clock** command
  - Attached to a netlist object, preferably the clock tree root pin
- **Automatically Derived Clocks**
  - also called auto-generated clocks
  - Their constraint is automatically created by the Vivado IDE on the output pins of the Clock Modifying Blocks (CMB)
  - The CMBs are
    - MMCMx, PLLx or BUFR primitives

create_generated_clock [**-name** *arg*] [**-source** *args*] [**-edges** *args*] [**-divide_by** *arg*] [**-multiply_by** *arg*] [**-combinational**] [**-duty_cycle** *arg*] [**-edge_shift** *args*] [**-add**] [**-master_clock** *arg*] [**-quiet**] [**-verbose**] *objects*

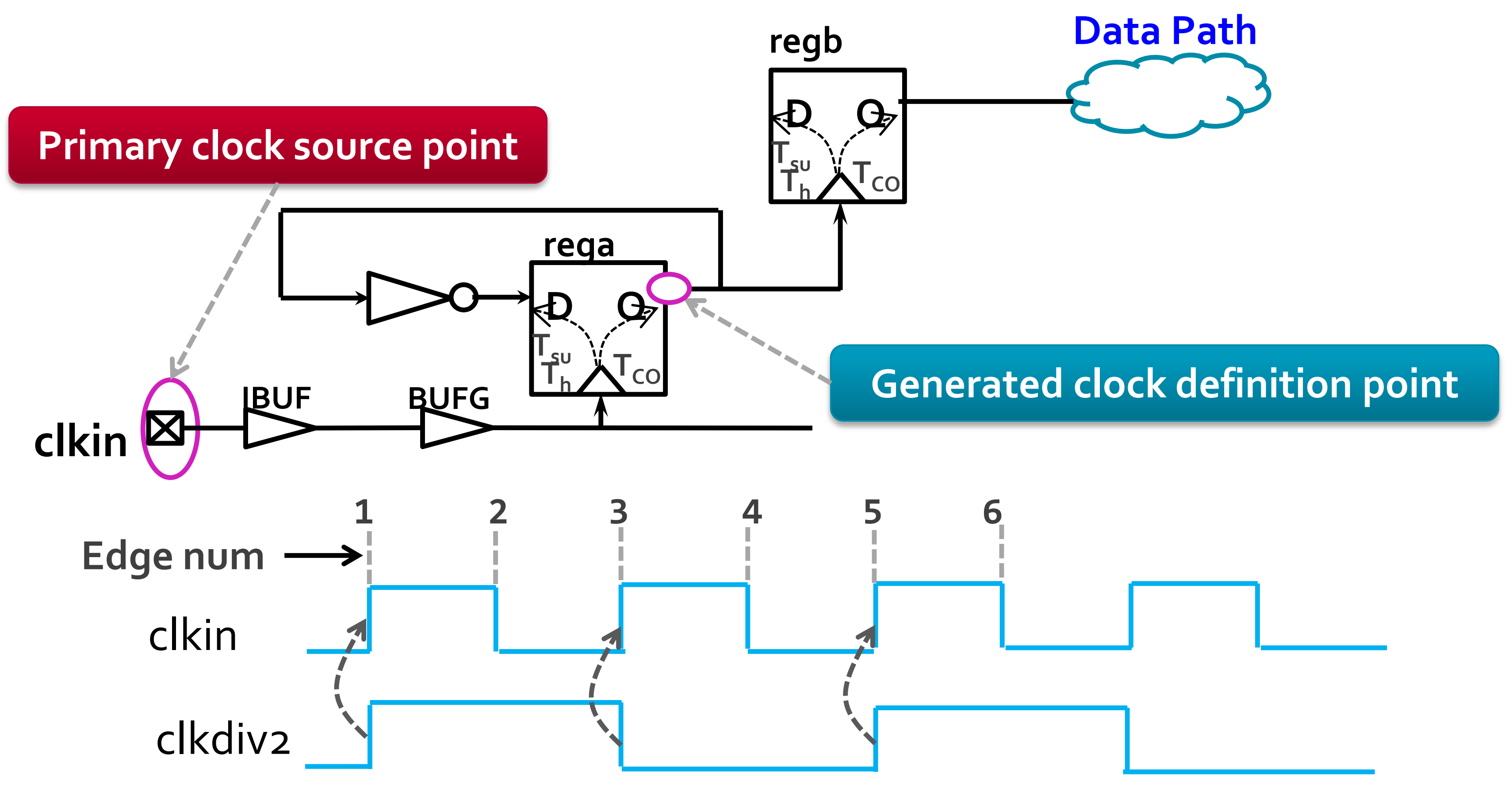
The **-source** option accepts only a pin or port netlist object. It does not accept clock objects

User Defined Generated Clocks



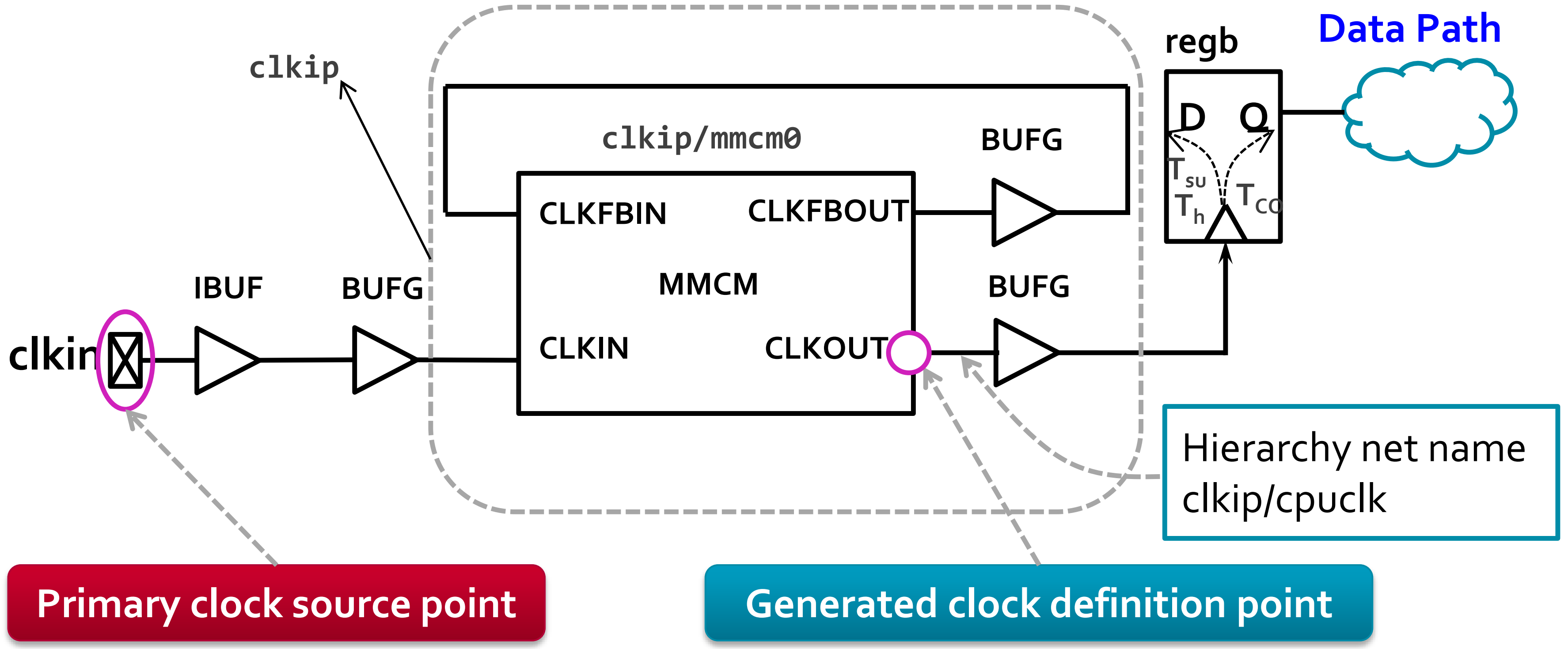
```
create_clock -name clkin -period 10 [get_ports clkin]

# Option 1: master clock source is the primary clock source point
create_generated_clock -name clkdiv2 -source [get_ports clkin] -divide_by 2 \
[get_pins REGA/Q]

# Option 2: master clock source is the REGA clock pin
create_generated_clock -name clkdiv2 -source [get_pins REGA/C] -divide_by 2 \
[get_pins REGA/Q]
```

User Defined Generated Clocks



```
# waveform specified with -edges instead of -divide_by
create_generated_clock -name clkdiv2 -source [get_pins REGA/C] \
-edges {1 3 5} [get_pins REGA/Q]
```

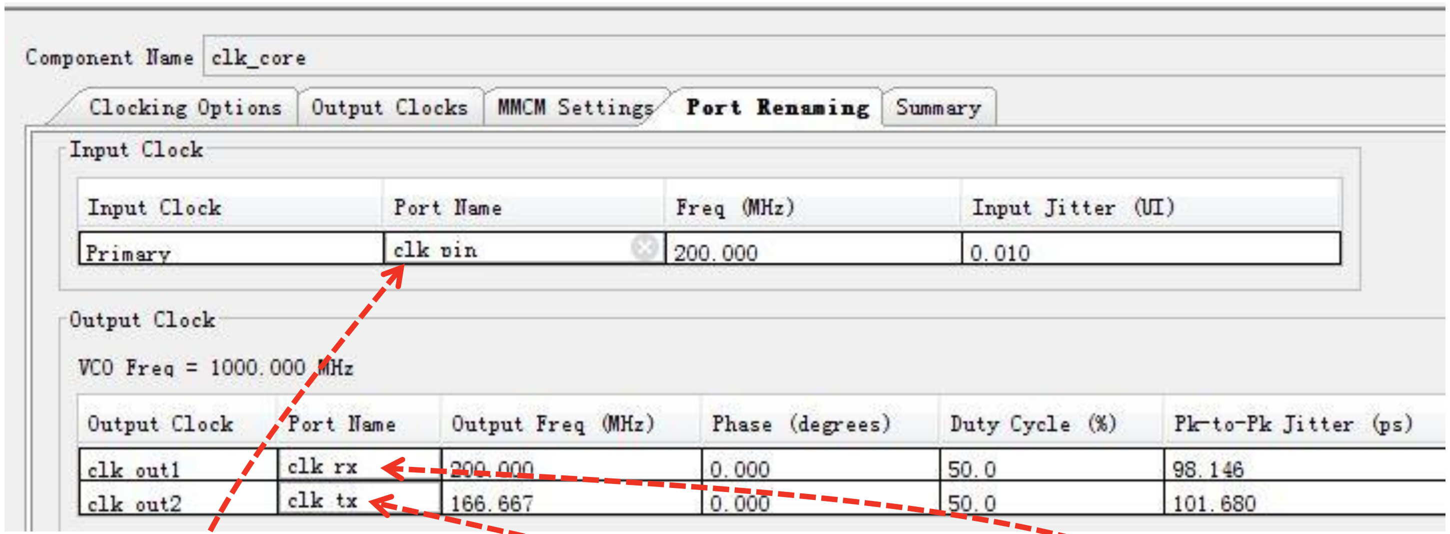
Automatically Derived Clock



- What's the generated clock name here?
  - cpuclk
- How to make your constraint independent of the clock name changes?

```
get_clocks -of_objects [get_pins clkip/mmcm0/CLKOUT]
get_clocks -of_objects [get_nets clkip/cpuclk]
```

Port Renaming in Clocking Wizard



Component Name: clk_core

Clocking Options | Output Clocks | MMCM Settings | **Port Renaming** | Summary

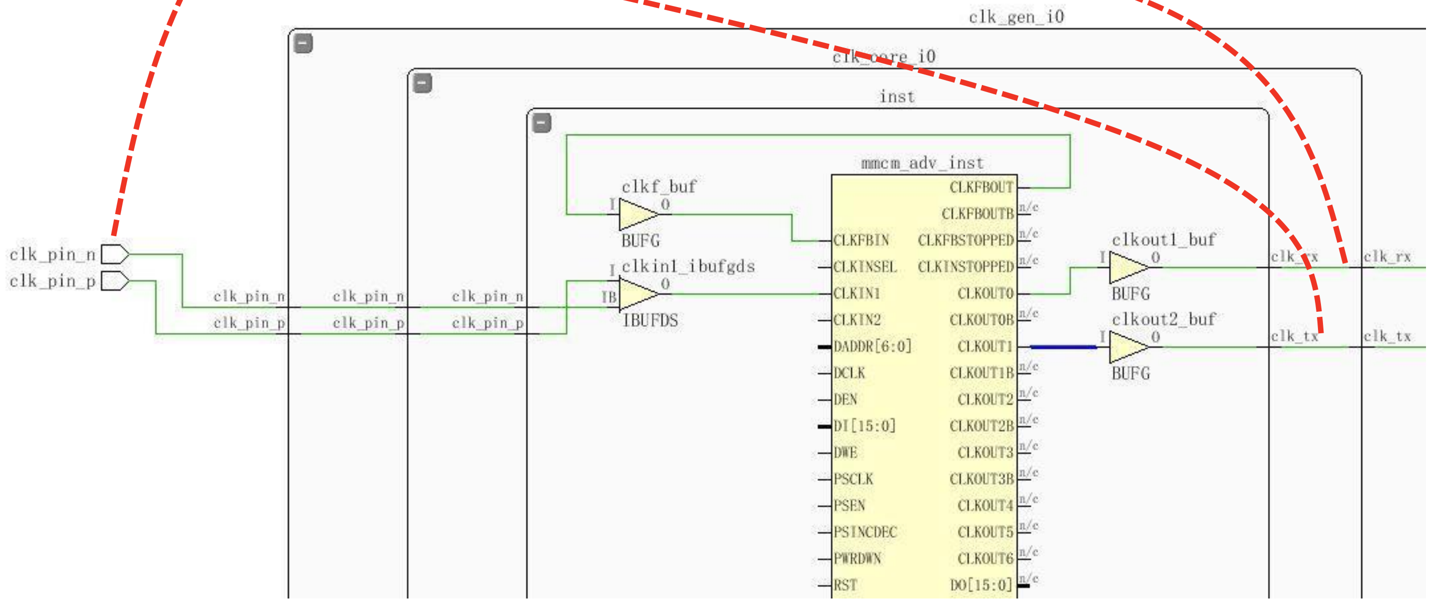
Input Clock

Input Clock	Port Name	Freq (MHz)	Input Jitter (UI)
Primary	clk_pin	200.000	0.010

Output Clock

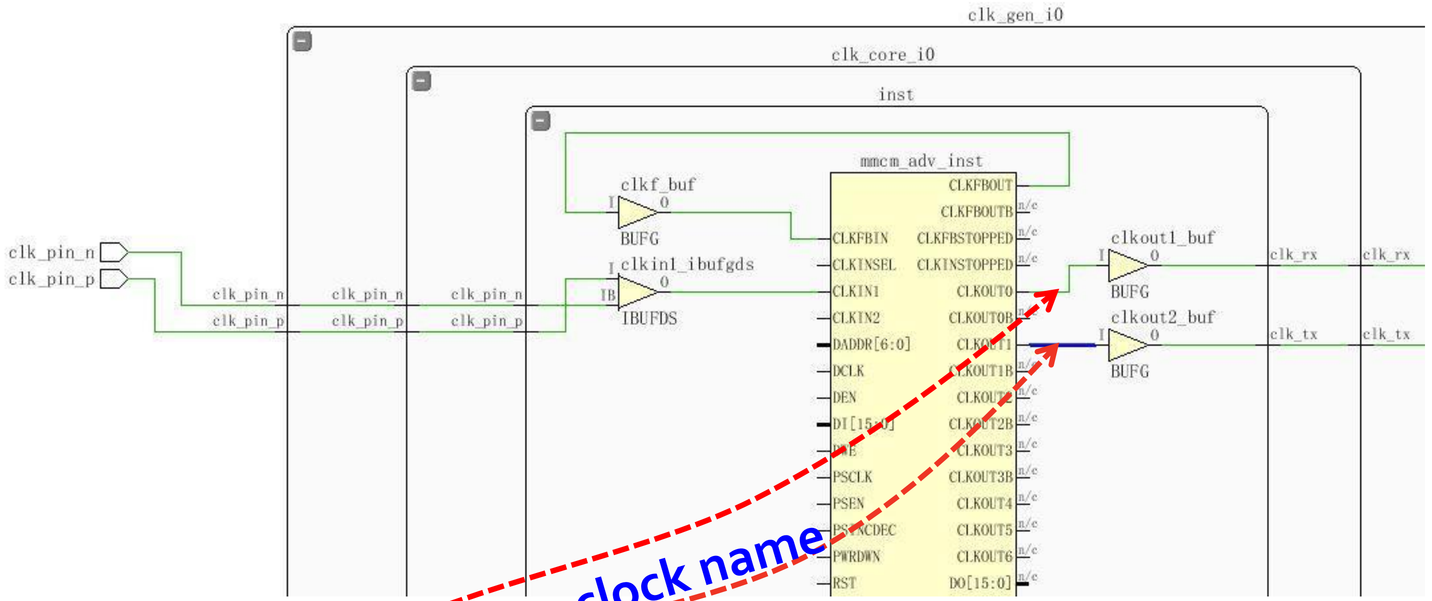
VCO Freq = 1000.000 MHz

Output Clock	Port Name	Output Freq (MHz)	Phase (degrees)	Duty Cycle (%)	Pk-to-Pk Jitter (ps)
clk_out1	clk_rx	200.000	0.000	50.0	98.146
clk_out2	clk_tx	166.667	0.000	50.0	101.680

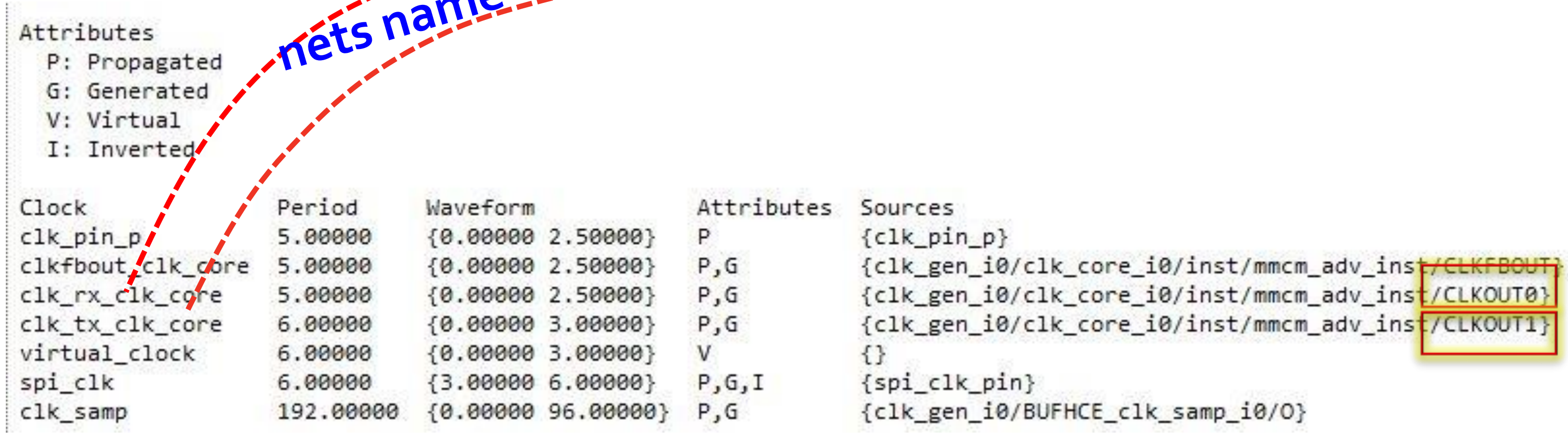


XILINX ➤ ALL PROGRAMMABLE.

report_clocks



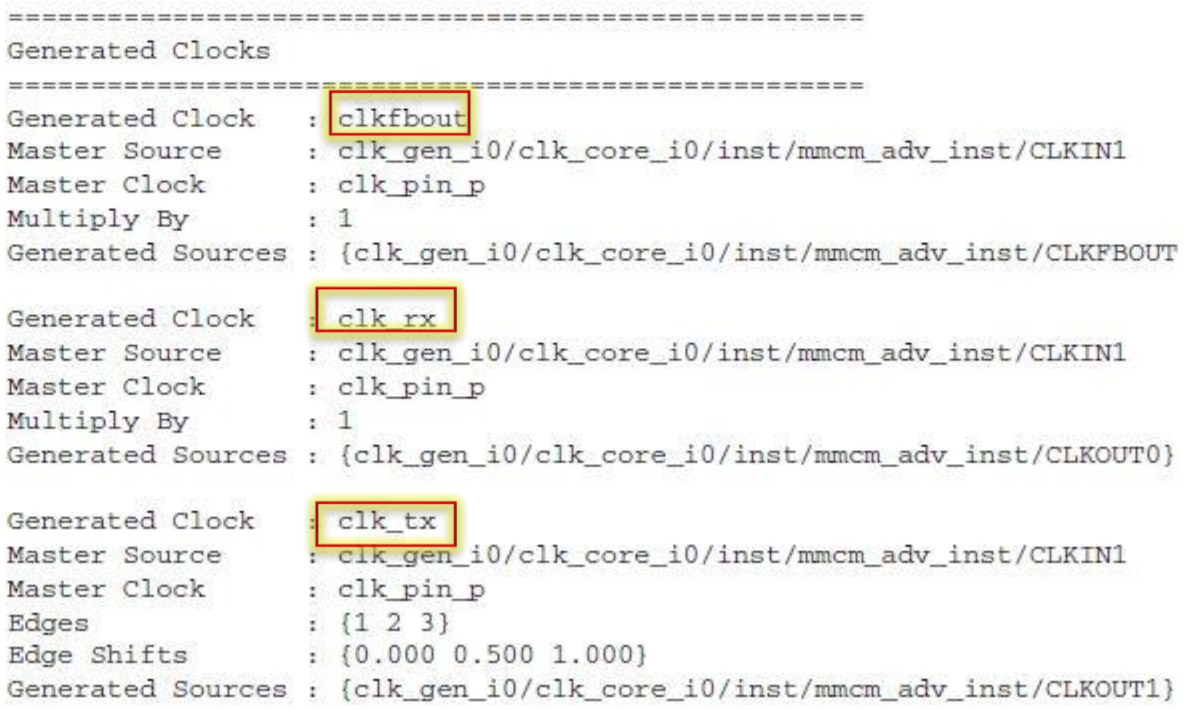
nets name = clock name



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

Clock               Period     Waveform            Attributes  Sources
clk_pin_p           5.00000    {0.00000 2.50000}   P           {clk_pin_p}
clkfbout_clk_core   5.00000    {0.00000 2.50000}   P,G         {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKFBOUT}
clk_rx_clk_core     5.00000    {0.00000 2.50000}   P,G         {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT0}
clk_tx_clk_core     6.00000    {0.00000 3.00000}   P,G         {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT1}
virtual_clock       6.00000    {0.00000 3.00000}   V           {}
spi_clk             6.00000    {3.00000 6.00000}   P,G,I       {spi_clk_pin}
clk_samp            192.00000  {0.00000 96.00000}  P,G         {clk_gen_i0/BUFHCE_clk_samp_i0/O}
```

Rename Tool-Generated Clocks

```
create_generated_clock -name clk_rx \
[get_pins clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT0]
create_generated_clock -name clk_tx \
[get_pins clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT1]
create_generated_clock -name clkfbout \
[get_pins clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKFBOUT]
```

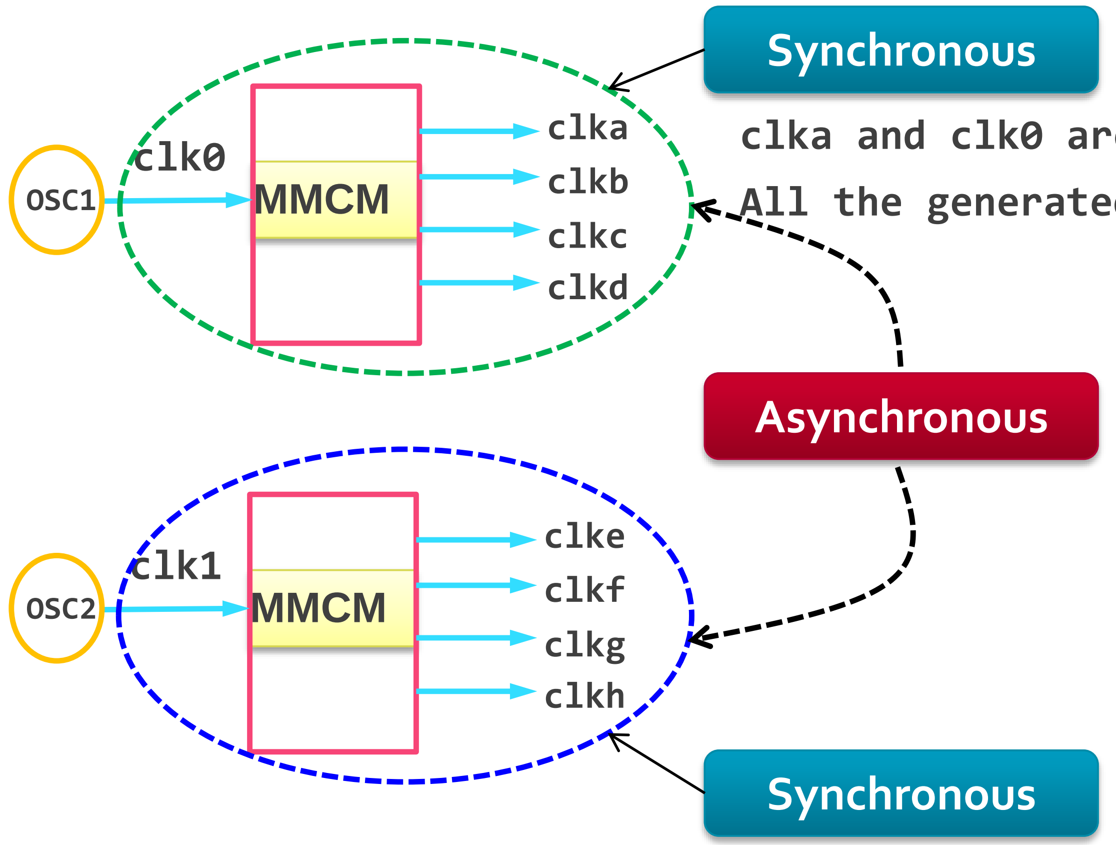


```
=========================================================
Generated Clocks
=========================================================
Generated Clock   : clkfbout
Master Source     : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock      : clk_pin_p
Multiply By       : 1
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKFBOUT

Generated Clock   : clk_rx
Master Source     : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock      : clk_pin_p
Multiply By       : 1
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT0}

Generated Clock   : clk_tx
Master Source     : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock      : clk_pin_p
Edges             : {1 2 3}
Edge Shifts       : {0.000 0.500 1.000}
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT1}
```

Clock Group

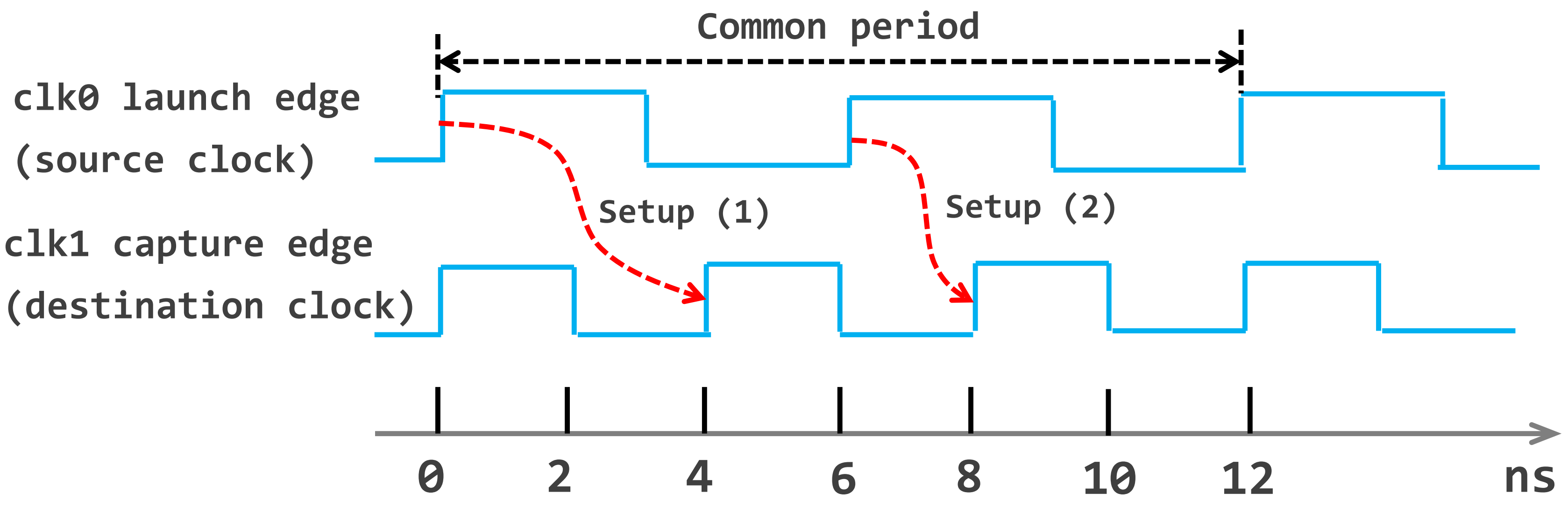
- Clock Categories

- Synchronous Clocks
- Asynchronous Clocks
- Unexpandable Clocks



The Vivado IDE assumes that all clocks are *related* by default

Unexpandable Clocks



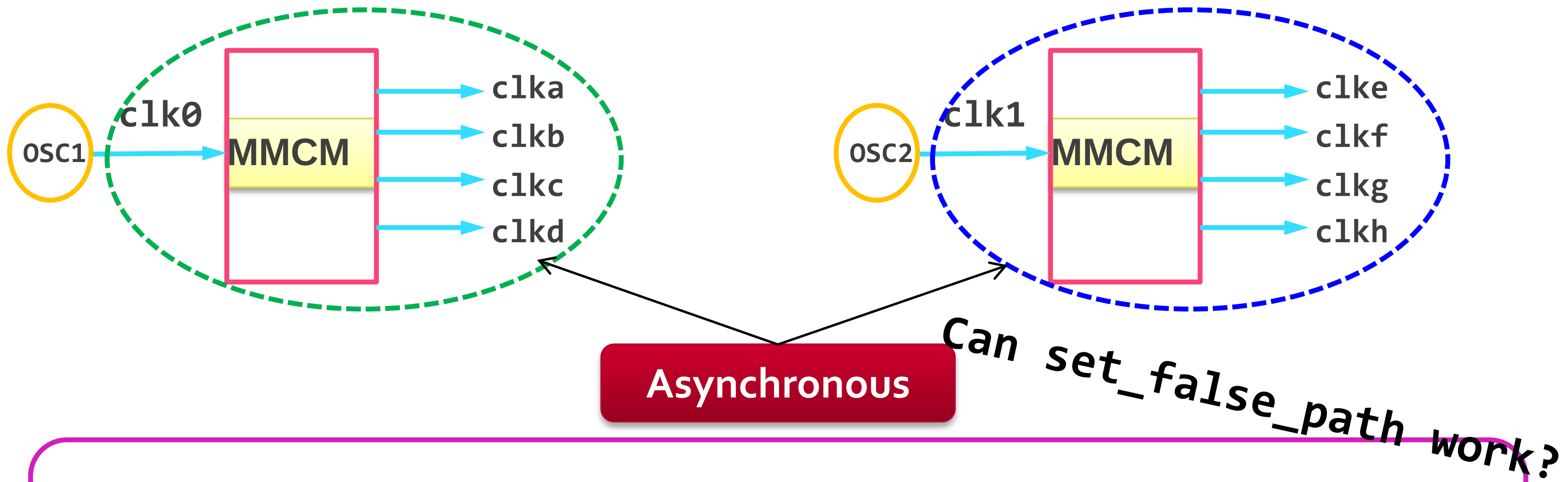
There are two unique source and destination clock edges that qualify for setup analysis: setup(1) and setup(2)

Two clocks are not expandable when the timing engine cannot determine their common period over 1000 cycles

Example:
clk0 has a 5.125ns period
clk1 has a 6.666ns period

Path requirement between two clocks are not reasonable (0.01ns)

Asynchronous Clock Groups

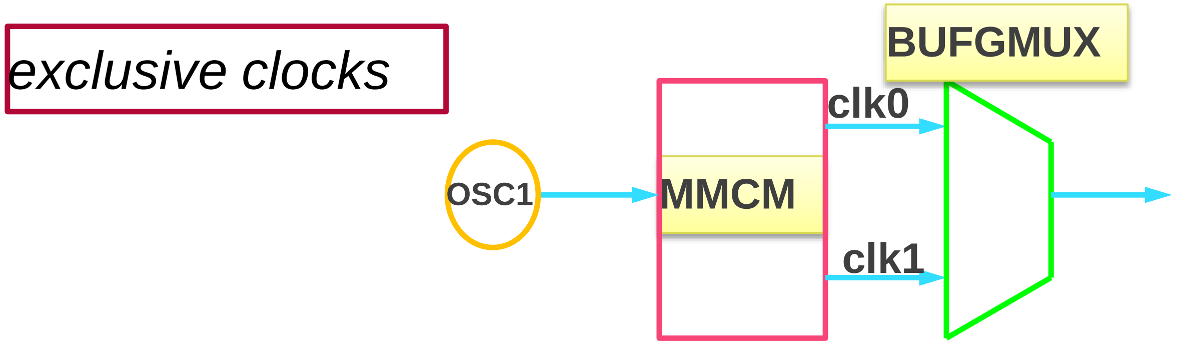


```
set_clock_groups -name async_clk0_clk1 -asynchronous \
-group [get_clocks -include_generated_clocks clk0] \
-group [get_clocks -include_generated_clocks clk1]
```

- The Vivado IDE assumes that all clocks are *related* by default
- The **set_clock_groups** command disables timing analysis between groups of clocks that you identify
- Caution: *Disabling timing analysis between two clocks does not mean that the paths between them will work properly in hardware*

Exclusive Clock Groups

- With the Vivado IDE, several timing clocks can exist on a clock tree at the same time, which is convenient for reporting on all the operation modes at once, but is not possible in hardware.



```
set_clock_groups -name exclusive_clk0_clk1 -physically_exclusive \
-group clk0 -group clk1
```

Clock Relationships

- **All clocks are related by default**
  - – Opposite of UCF defaults
  - – Clock period is expandable to find a common multiple
- **Asynchronous clock domains remove analysis**
  - – XDC command: set_clock_group
  - – Be careful! These paths are valid – need synchronization
- **Unexpandable clocks**
  - – Timing tool could not find common period when expanding two clocks.
  - – Need to be fixed by user (change frequency or false-path)
  - – Reported in check_timing

Migrate from UCF to XDC

Period constraint in UCF

```
NET "clk_ref_p" TNM_NET = TNM_clk_ref;
TIMESPEC "TS_clk_ref" = PERIOD "TNM_clk_ref" 5 ns ;
```

Period constraint in XDC

```
create_clock -name clk_ref_p -period 5 [get_ports clk_ref_p]
```