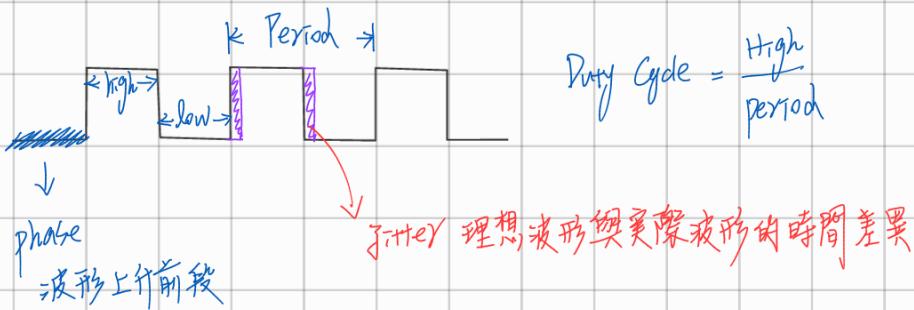


# Clock



## 使用Tcl控制CLK

創建

`create_clock -name <name> -period <period value> -waveform {  
 [clock名字] [nS為單位] [波起 波落]}`

e.g. `create_clock -name clk2 -period 10.0 -waveform {0.1 2.1}`



$$\text{Duty Cycle} = \frac{2.1 - 0.1}{10.1 - 0.1} = \frac{2}{10} = 20\%$$

獲取

`get_clocks <name>` 抓某個 CLK出來，常與其它指令併用

若要獲得某 CLK 的狀態或資訊

`get_property [get_clocks <name>]`

`report_property [get_clocks <name>]`

| report_property [get_clocks clk_samp] |        |           |   |
|---------------------------------------|--------|-----------|---|
| Property                              | Type   | Read-only | Value   |
| CLASS                                 | string | true      | clock   |
| DIVIDE_BY                             | int    | true      | 32  |
| FILE_NAME                             | string | true      | C:/training/Designing FPGAs Using the Vivado Design Suite |
| INPUT_JITTER                          | double | true      | 0.000   |
| IS_GENERATED                          | bool   | true      | 1   |
| IS_INVERTED                           | bool   | true      | 0   |
| IS_PROPAGATED                         | bool   | true      | 1   |
| IS_RENAMED                            | bool   | true      | 0   |
| IS_USER_GENERATED                     | bool   | true      | 1   |
| IS_VIRTUAL                            | bool   | true      | 0   |
| LINE_NUMBER                           | int    | true      | 29  |
| NAME                                  | string | true      | clk_samp  |
| PERIOD                                | double | true      | 0.000   |
| SOURCE                                | pin    | true      | clk_gen i0/BUFGE clk samp i0/I                            |

設定

`set_jitter` system 內所有 CLK 的 jitter

`set_system_jitter <value>`

設定某個 CLK 的 jitter

`set_input_jitter <clock-name> <value>`

) 會被考慮進 STA 裡

(Static Timing Analysis)

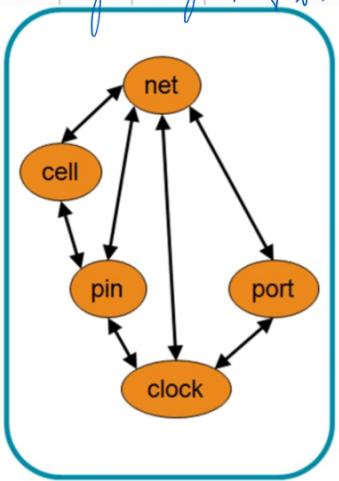
`get_latency`

額外的 clk delay

`set_clock_latency -source <latency> <objects>`

e.g. `set_clock_latency -source { -early 0.2 [get_clocks <clock name>] -late`

Setting latency 的對象

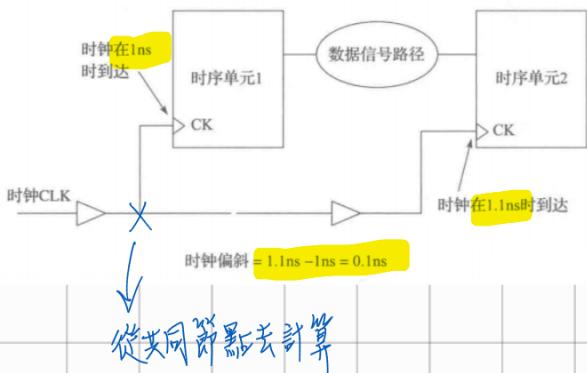


若有 clk, 則設定後的 latency 會添加到此 clk 所有  
的 Destination

若為 port / Pin, 則會添加到與之連接的所有 clk

## Clock Skew

指 clk 到每個 component 的時間不同產生的時間差



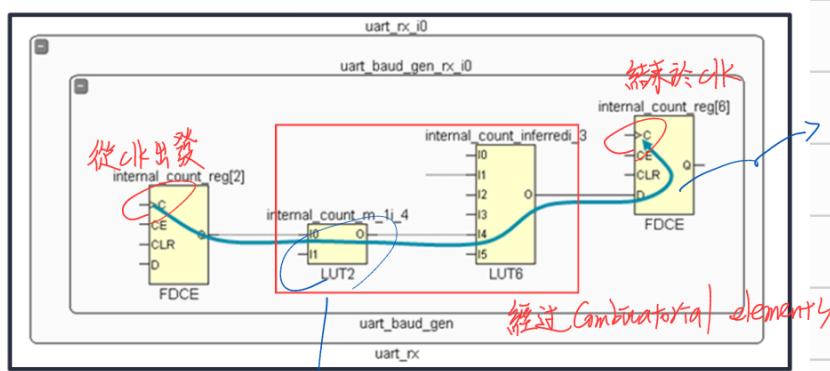
公式  $T_{Cin} - T_{Ui}$

公共節點到 start point clk 的 delay

公共節點到 end point 的 delay

時間差太大会造成電路功能無法正常運作

## Static Time Paths

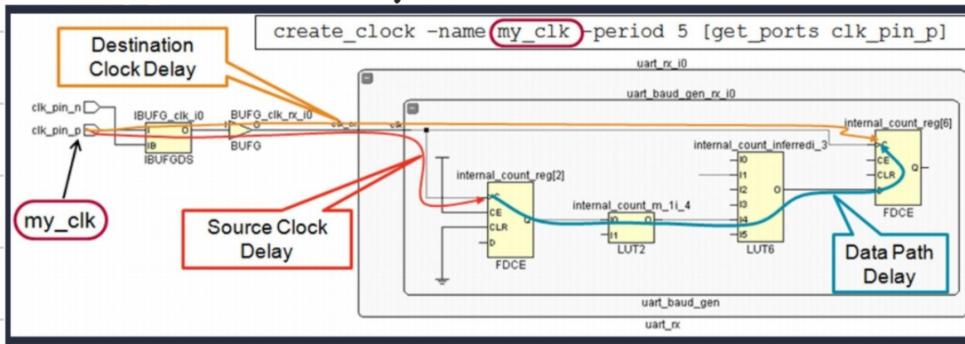


這樣的路徑稱為 static time Path

每塊 logic 在 Vivado 內稱為 BEL (Basic Element logic)

用途是拿來做 STA (Static Time Analysis) 只對 Timing Performance 做驗證, 不對功能做測試

# Static Time Analysis



## Setup Check

$$\text{slack} (\text{餘量}) = \text{Period} + \text{Destination Clock Delay} - \text{Source Clock Delay} - \text{Data Path Delay}$$

- clock Uncertainty

slack > 0 为通过 setup check, 因此

$$T_{\text{period}} > T_{\text{source}} + T_{\text{data}} + T_{\text{uncertainty}} - T_{\text{destination}}$$

$$T_{\text{period}} + T_{\text{destination}} > T_{\text{source}} + T_{\text{data}} + T_{\text{uncertainty}}$$

取 min                          取 max

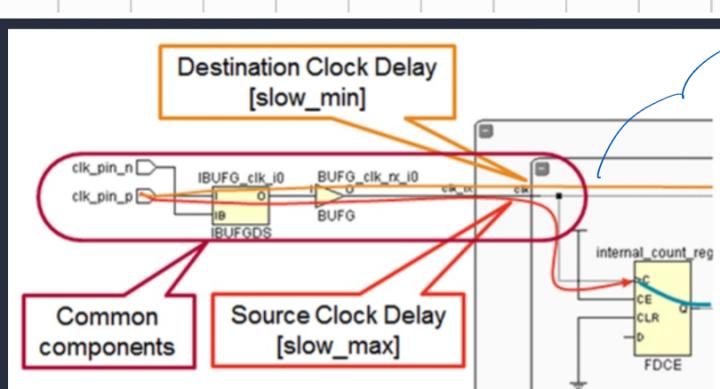
透过上式, 可看出从第一个 clk-pin-p 到第一个 DFF 间的 Delay 加上

第一个 DFF 经过中间资料的传递到最终 CLK 的 Delay

不能大于

从 clk-pin-p 直接到最终 CLK 的 Delay 加上一个 CLK 的 Period

## Clock Pessimism



這段路線為公共路線,

同一條路線實際中不會產生兩種 Delay

clock Pessimism

$$= \text{Common Components} (\text{slow max} - \text{slow min})$$

所有共同路線

用途在於最後  $T_{\text{source}} - T_{\text{destination}}$  的結果, 需要再扣掉 clock Pessimism 才會

偏向實際的 clock skew 值

## Generated Clock

一個複雜的設計中不會只有一個 clock，各單元件可能會工作在不同的 clock 下

generated clock 必定會跟著一 Master clock

可以是 primary clock 或是另一個 generated clock

透過 FPGA pin 脚輸入

因此 generated clock 的用意就在於 CLK 切換

CLK 分頻

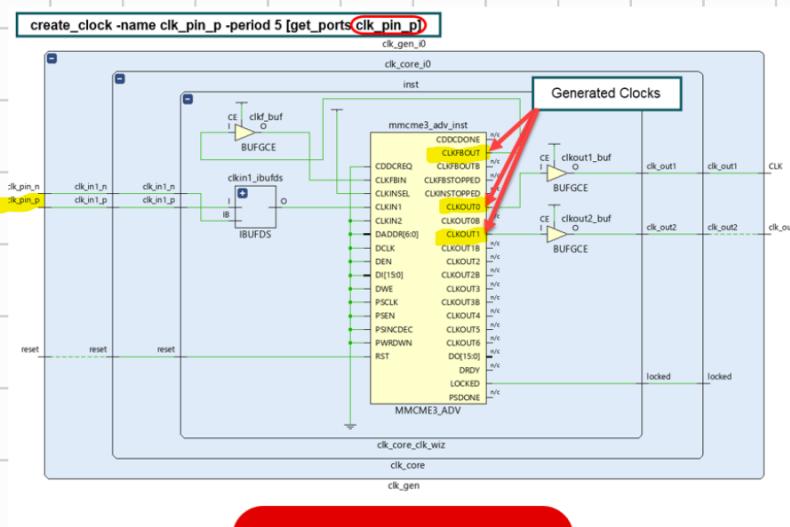
CLK 倍頻

clk reset 等等，可以透過 create-generated-clock 來定義

## 產生 Generated Clock 的方式

1. MMCM, PLL, HSMO Xilinx 將其做成 HSMO wizard，方便使用且不用重構並能達成高速 I/O

CMT (clock Manager File) 能夠基於 input clocks 產生 generated clocks



MMCM 可以做到二輸入的 CLK，多輸出的 CLK，並有到小數點位的倍/除頻

PLL 只能做到整數倍/除頻

2. BUFGCE 帶有 clk enable 的 BUFG

global buffer，輸入是 BUFG 的輸出

global input buffer，板子上給你 input clk 和頻分配於此

3. Non-Clock Resource

能使用 create-generated-clock -name <name> -source <source> <relationship> <objects> 產生

Xilinx 不推薦使用這方式

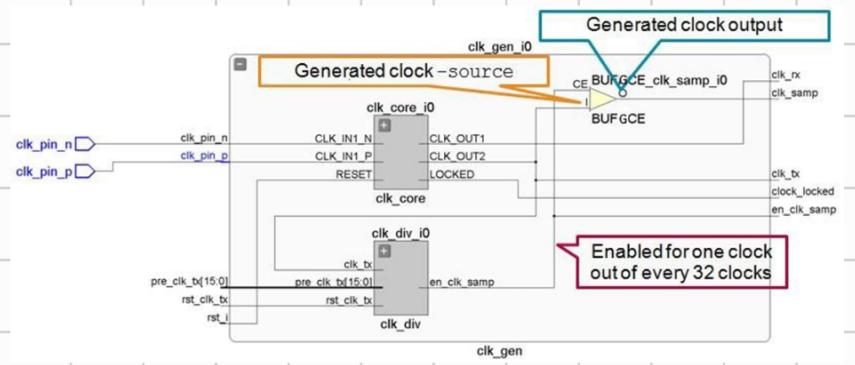
e.g. create-generated-clock -name clk Samp

port or pin 跟 CLK 有關並做為 reference clk

-source [get\_pins clk\_gen\_10/BUFGCE\_CLK\_SAMP\_10/I] -divide\_by 32

[get\_pins clk\_gen\_10/BUFGCE\_CLK\_SAMP\_10/O]

relationship between source & generated clk

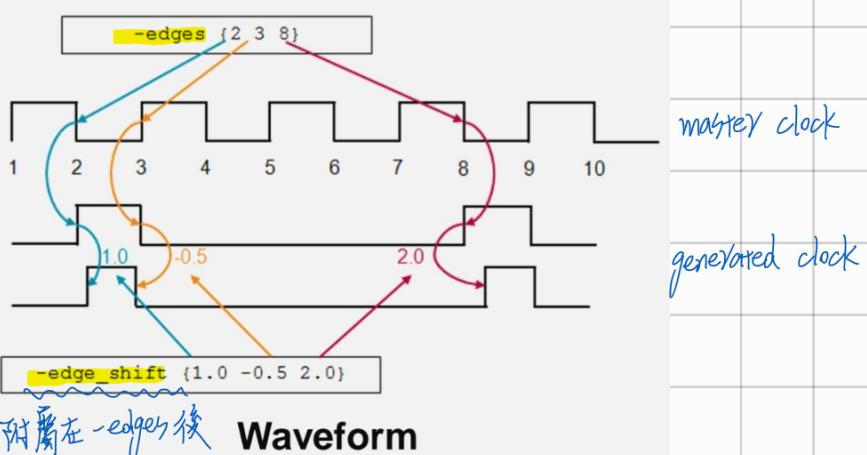


一般來說，不推薦利用 get\_clocks 去抓 clock，因為各 tools 所產生的 clock 名字不一  
建議透過 object 的形式，也就是描述 pin/port 與 clock 的關係來去獲取

## Propagation

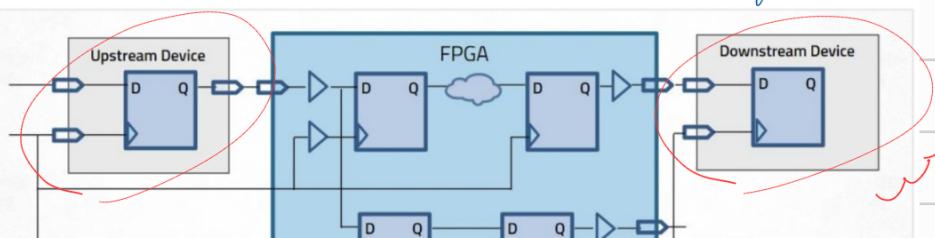
無論是自動產生 (MMCM / PLL...) 還是手動 (create-generated-clocks)，在做 Static Timing Analysis 時，都會追溯回 primary clock start point

補 另一產生 generated clocks 的 tool

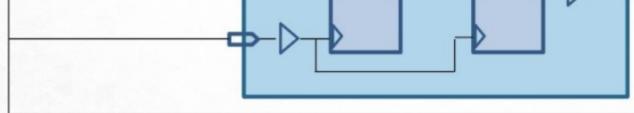


## Virtual Clocks

為了在 FPGA 設計內不存在的 clock，因為 FPGA 與 Board 上其它晶片進行資料互動時，  
其它晶片的 clock 並非由 FPGA 提供，因此 FPGA 在進行時序分析時必須定義一個  
Virtual clock 來代表並約束非FPGA 端的 I/O delay.



左右端非FPGA內部的單元，  
因此要做 I/O constraint 時須先定義



特性

- ① 不會有實體連接
- ② 主要用於定義 input / output delay 的 constraints
- ③ 使用 create-clock 創建，不用指定 source object

e.g. `create_clock -name <name> -period <period>`

```
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]
```

→ 實體 clk

→ 未指定 source object，為 Virtual clk

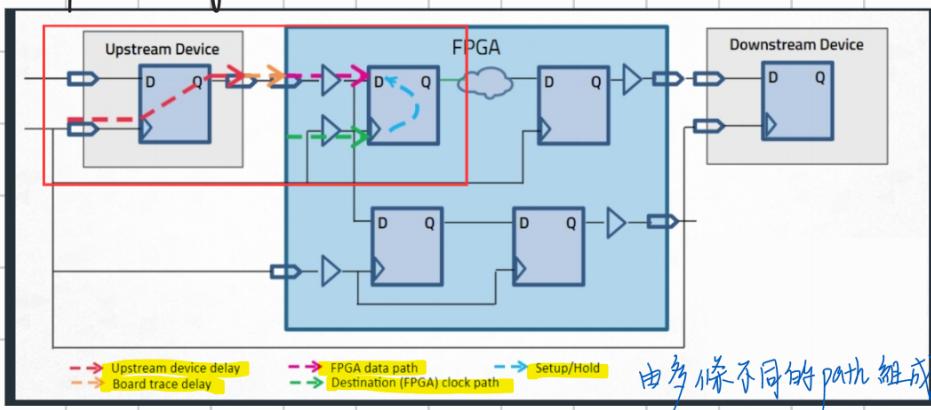
設定 Virtual clk 的 attributes

補：還有以下情況會使用 Virtual clk

- ④ 非內部 FPGA Design clk 的 I/O
- ⑤ FPGA I/O 路徑與內部生成的 clk 有關，卻無法有確定的值
- ⑥ 不希望修改內部 FPGA clk 的特性，但又想為 I/O delay constraints 指定不同參數

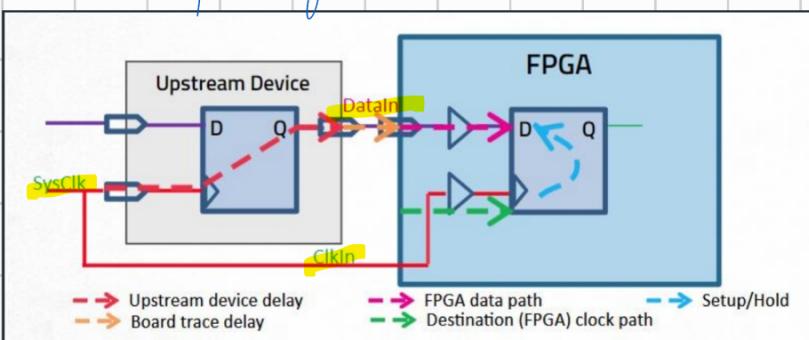
## Input / Output Timing

### Input Timing overview



⇒ 要定義 upstream 的 delay，必須先確定 source 與 destination 的 clock (create-clock)

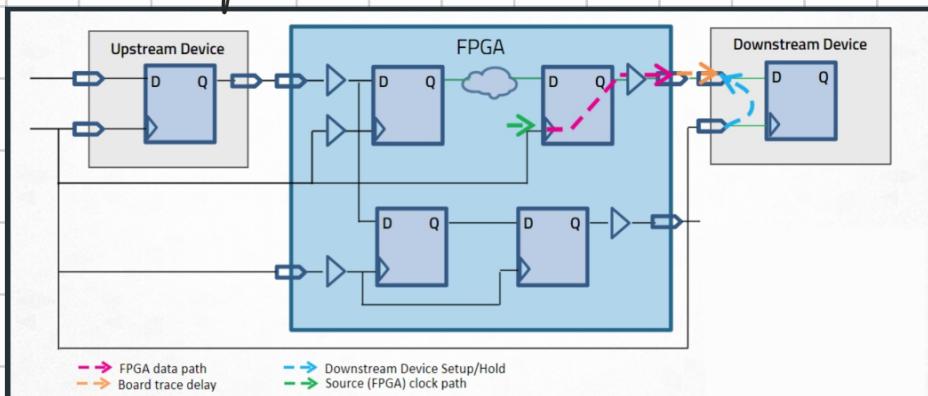
⇒ 利用 set-input-delay 來設定 max setup 和 min hold



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

外部 Device 用的 clk

## Output Timing Overview



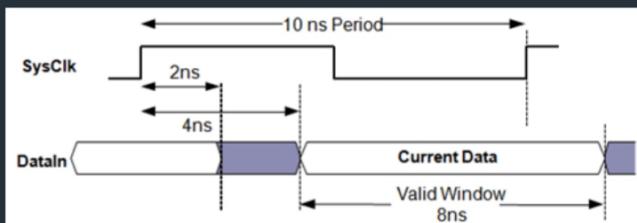
一樣的 create\_clock, 只是換成了 set\_output\_delay -clock <name> <delay> <objects>

一般而言，一個 input port 可以擁有一個 max delay 和 min delay

for setup check      for hold check

eg.

```
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]
```

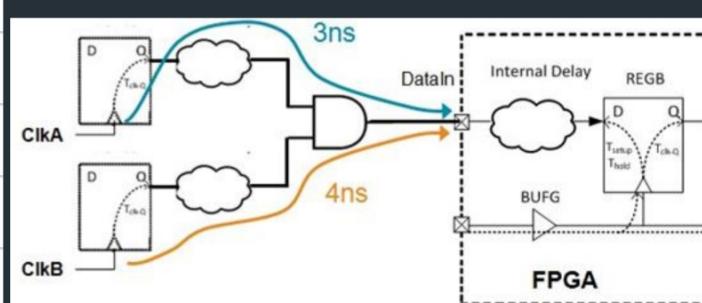


一個 input / output 可以擁有多個 set\_input\_delay 作為多個 static Time Analysis 的用途

一 使用 -add\_delay 做添加

eg.

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



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





# Clock Gating

晶片實際運作中，有些訊號與功能不需要一直開啟，用不到時會將 clock signal 關閉，減少動態功耗。

如何控制 clock 呢？佈署 clock gating？

## Using Clock Enables

→ 每個 logic element 都配置一個 clock enable

優點

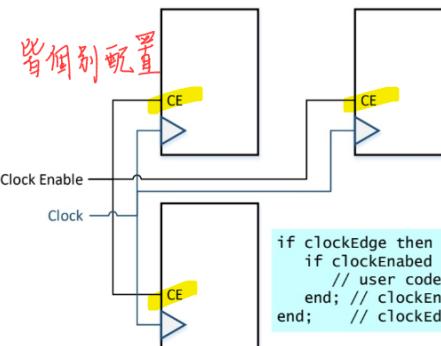
能較詳細的處理哪些 logic 要配置 clock enable

缺點

相同的 logic 通常會 group 在一起，因此若其中有些 elements 有 CE 有些沒有，會造成 routing & utilization issues

設計者必須謹慎選擇 CE 佈署的位置

總體 clock 還是沒被關掉



## Using Regional Clock Control

在同一 clock source 上加入 CE

優點

當該 clock source 被 gated，該整個 clock network 就不會作動，有效降低功耗

缺點

全有全無

無法精細的對每個 logic element 進行配置

