易灵思

时序优化中的multicycle约束

目录

易灵思

为什么要提出Multicycle?

Multicycle对时序放松的约束方法

03 使用Multicycle进行时序优化

为什么要提出Multicycle?

1　默认情况下的setup和hold检查

2　Multicycle提出的背景

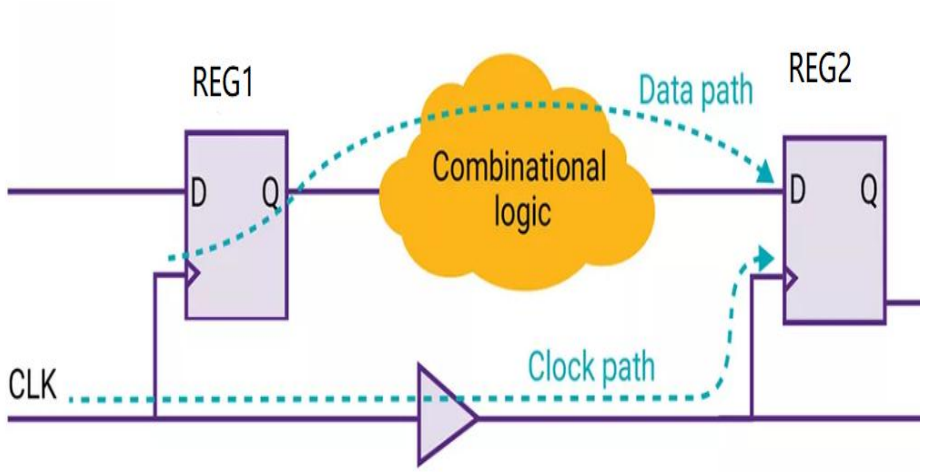
说明：下面提到的Capture clock 和latch clock含义一样，Capture edge 和latch edge 含义一样。

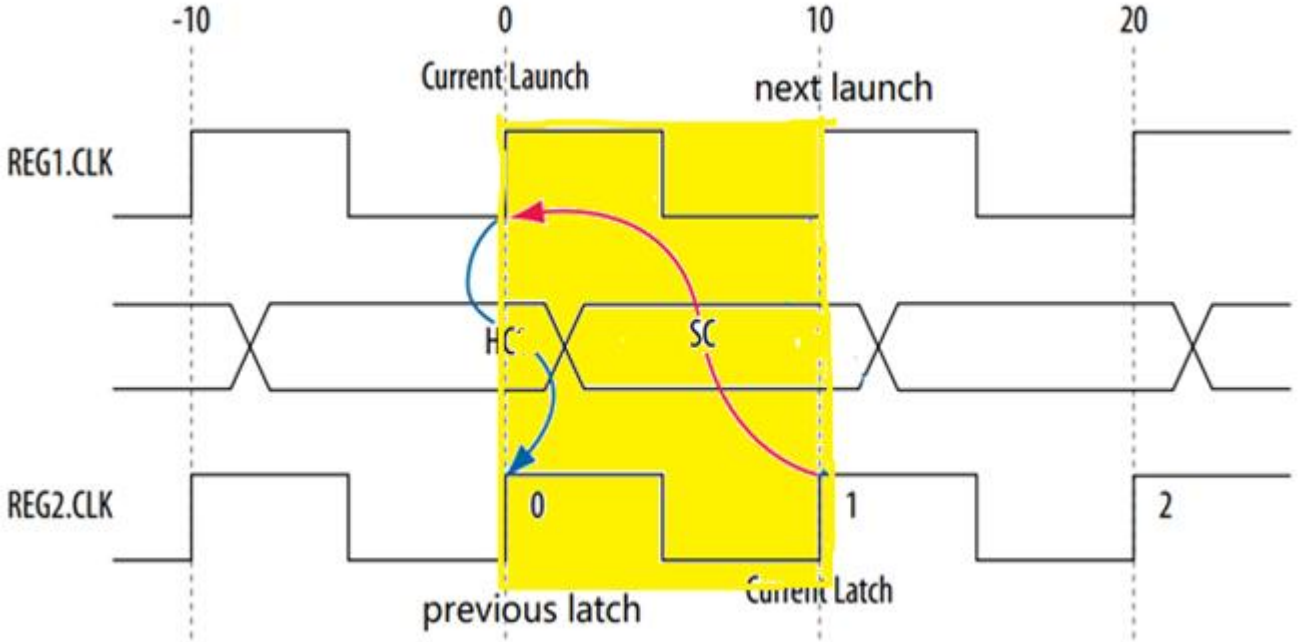
默认情况下的setup和hold检查

- 默认情况下，Effinity对setup和hold进行单周期检查。
- setup关系定义为launch edge和latch edge之间的时钟周期数，默认是1。

  setup check =curremt latch edge – current launch edge
- hold 检查前一个latch edge 与当前launch edge之间的时钟周期数，默认是0。

  hold check = current launch edge-previous capture edge





Multicycle提出的背景

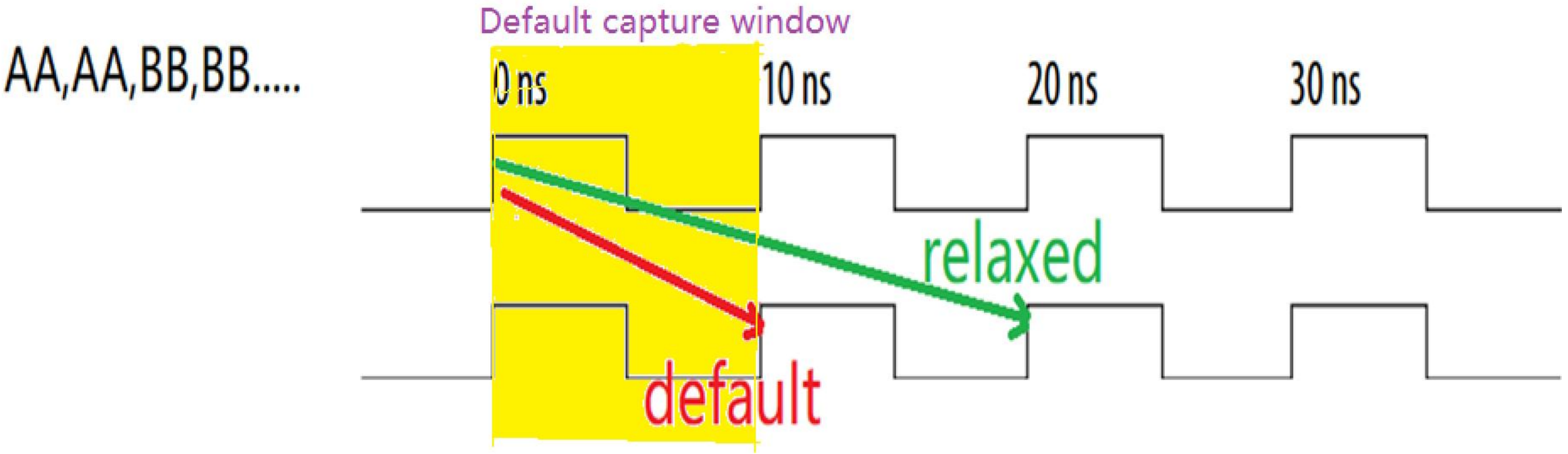
1　源端数据有重复，但数据延时大，setup很难满足

2　源时钟频率比目的端高几倍，但是连续几拍数据想同

3　目的寄存器时钟频率比源寄存器高几倍

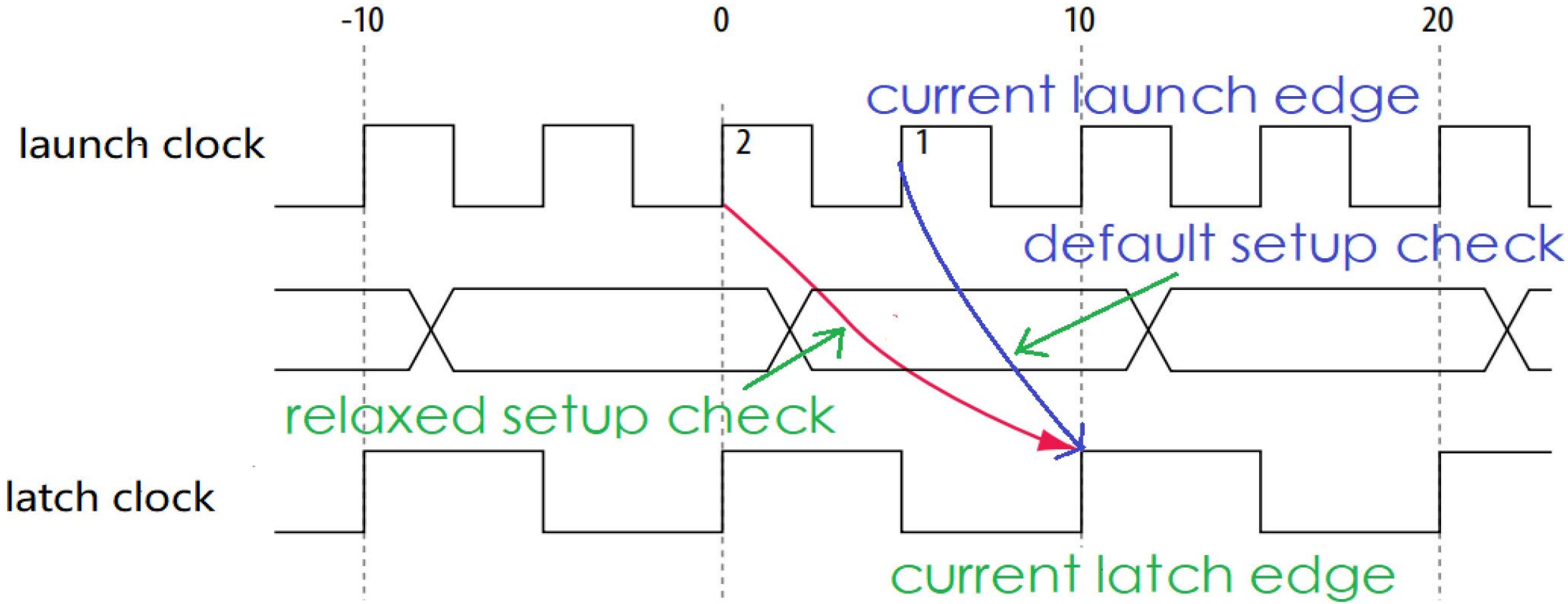
源端数据有重复，但数据延时大，setup很难满足

- 源寄存器和目的寄存器都使用同一个时钟，但是，数据路径延时很大，setup很难满足。如果此时源寄存器的数据每2个或者2个以上的时钟周期才变化一次，此时，就可以采用multicycle来放松对目的寄存器的约束。



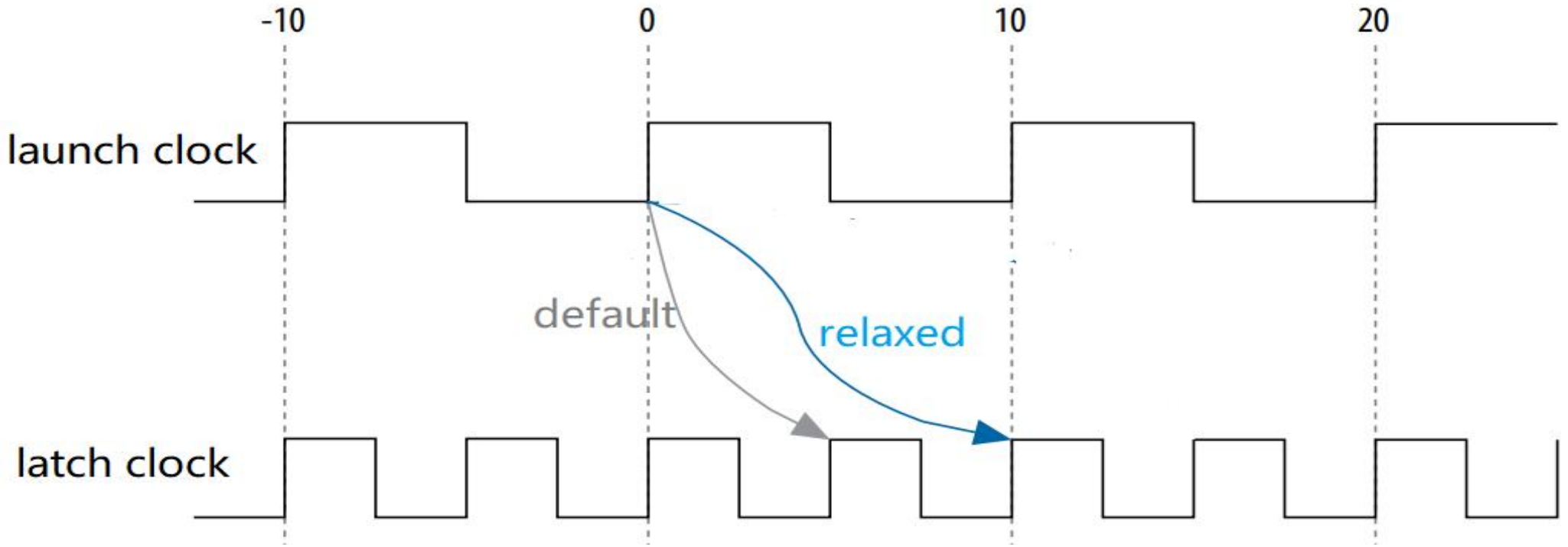
源时钟频率比目的端高几倍，但是连续几拍数据想同

- 源寄存器和目的寄存器的时钟来自于同一个时钟源，但是频率不同。launch clock的频率是Latch clock的n倍（n≥2），源端寄存器连续n拍送出的数据都相同。



目的寄存器时钟频率比源寄存器高几倍

- 源寄存器和目的寄存器的时钟来自于同一个时钟源，但是频率不同。Latch clock的频率是launch clock的n倍（n≥2）



Multicycle对时序放松的约束方法

1、Multicycle在sdc中的通用约束方法

2、Multicycle中setup 的约束方法

3、Multicycle中hold 的约束方法

Multicycle在sdc中的通用约束方法

在sdc文件中，multicycle的通用约束方法如下：

set_multicycle_path path_multiplier [-setup|-hold] [-start|-end] -from <Start Point> [-through <ThroughPoint>] -to <End Point>

- set_multicycle_path是关键字；
- -from 是关键字，后面跟起点；
- -to　　是关键字，后面跟终点；
- -setup 和-hold 只能选择一个，不能同时选择；
- -start 和 -end只能选择一个，不能同时选择；
- Start Point 表示起点，它可以是寄存器的输出，I/O端口或时钟；
- End　Point 表示终点，它可以是寄存器的输入，I/O端口或时钟；
- path_multiplier 一般都是自然数（包括0)；
- [-through <ThroughPoint> 表示起点和终点之间通过某点，是可选项。

Multicycle中setup 的约束方法

1、约束multicycle setup 的两种方法

2、以capture clock为基准的setup约束方法

3、以launch clock为基准的setup 约束方法

约束multicycle setup 的两种方法

- Multicycle setup的约束，是基于source (-start) 或者 destination (-end) clock，用来调整setup 关系的。而path_multiplier是指需要调整的时钟周期个数。
- set_multicycle_path –setup -end <path multiplier >

  以capture clock为基准调整setup关系， path_multiplier是调整的时钟周期数。
- set_multicycle_path -setup -start < path_multiplier >

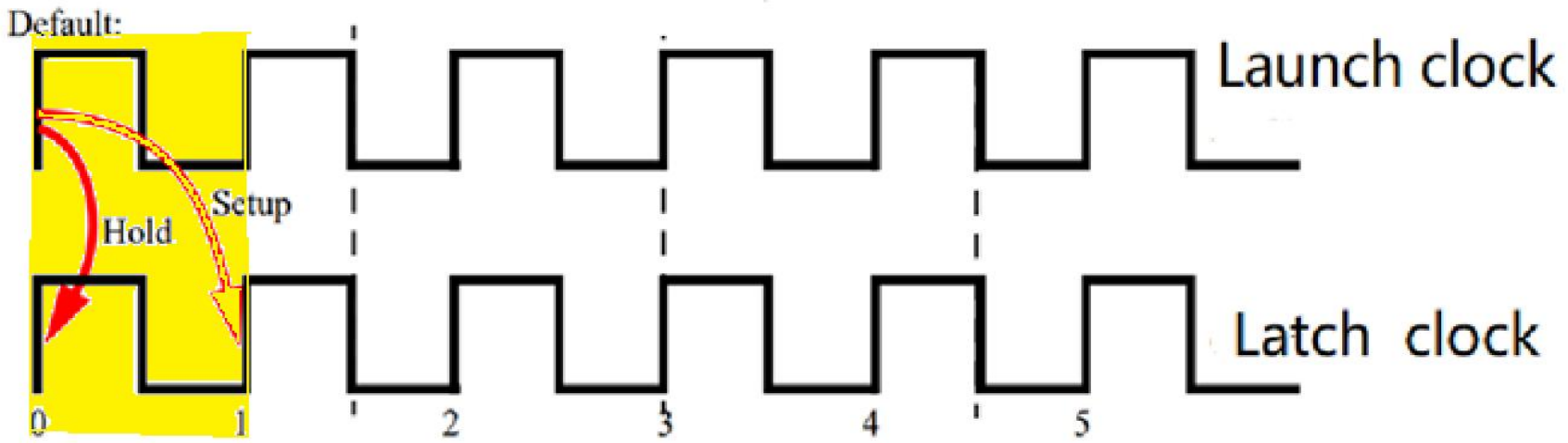
以launch clock为基准调整setup关系， path_multiplier是调整的时钟周期数。

以capture clock为基准的setup 约束方法

- 在默认状态下， setup的检查是以latch clock为参考。它定义为latch edge和launch edge之间的时钟周期数（latch edge –launch edge），默认为1，此时在sdc中不需要使用multicycle来约束。如果要约束，对setup来说，可以这样来约束：

  set_multicycle_path -setup -from a -to b 1

  或 set_multicycle_path -setup -end -from a -to b 1

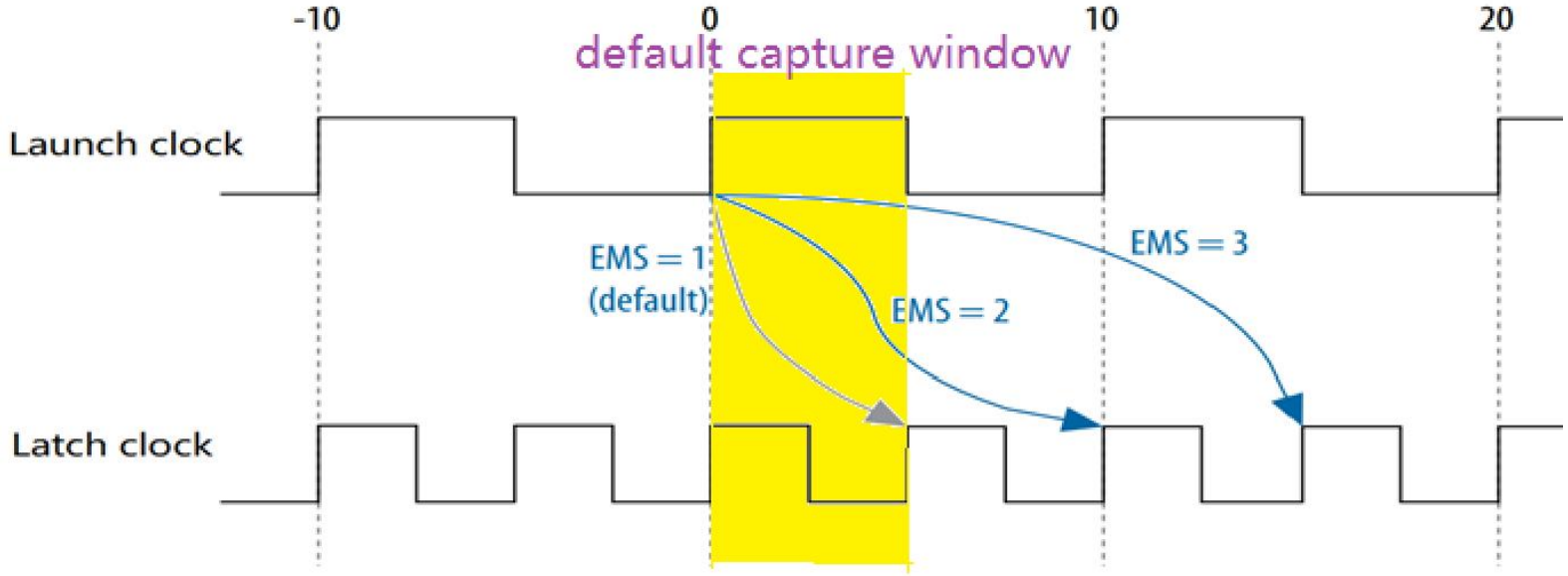


Default capture window

以capture clock为基准的setup 约束方法（续）

- 如果要放松对setup的约束，此时，你就要把 path multiplier 的值从默认的1改成你需要放松到的值。下列图中，EMS（end multicycle setup）从默认的1，分别改到2和3，从而setup的检查就分别放松了1个和2个时钟周期，相应的multicycle的约束如下：

```
set_multicycle_path –setup  -end  -from  a  –to  b  2
set_multicycle_path –setup  -end  -from  a  –to  b  3
```



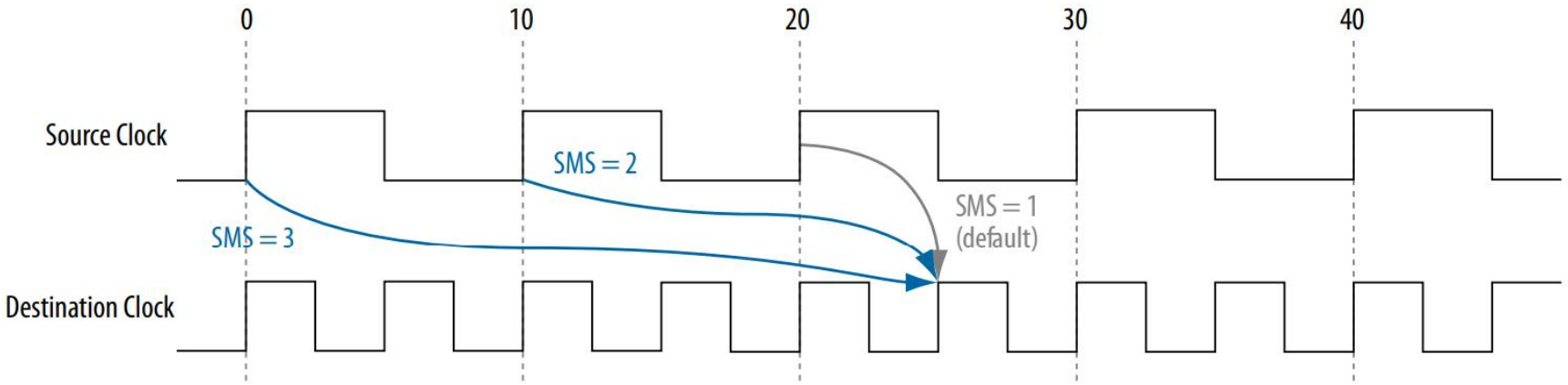
以launch clock为基准的setup 约束方法

- 以launch clock为基准的setup multicycle约束，它根据path multiplier的值，来修改 source clock的launch edge。跟以latch clock为基准的setup multicycle约束不同，它把source clock的launch edge从缺省的地方往左移动path_multiplier-1个launch clock周期。下图中的SMS表示start multicycle setup，对应的约束分别如下：

  set_multicycle_path -setup –start -from a -to b 1 （default）

  set_multicycle_path -setup –start -from a -to b 2

  set_multicycle_path -setup –start -from a -to b 3



Multicycle中hold 的约束方法

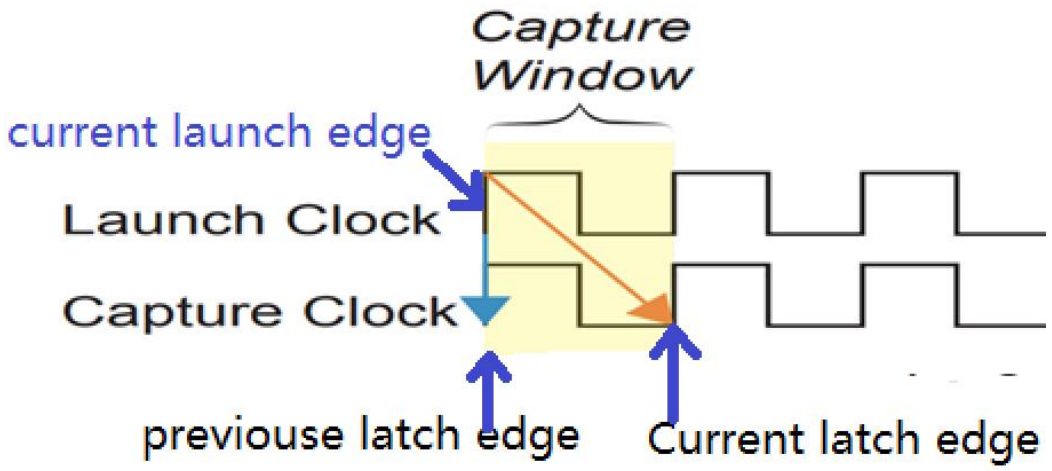
1、Effinity对hold的默认检查

2、约束multicyle中 hold的两种方法

3、以launch/Capture clock为基准的hold 约束方法

Effinity对hold的默认检查

- 在缺省状态下，setup 和 hold 组成一个时钟周期的捕获窗口，此时setup = 1，hold = 0.
- 在Effinity中，默认状态下， hold的检查是以launch clock为参考。它主要检查current launch edge 拍出的数据是否被前一个latch edge捕获到。hold的检查时刻比setup的检查时刻早一个时钟周期。



此时，在sdc中不需要对multicycle进行约束，如果要约束，可以做如下的约束。

```
set_multicycle_path -setup -from a -to b 1
set_multicycle_path -hold -from a -to b 0
```

约束multicyle中 hold的两种方法

- Multicycle hold的约束，是基于source (-start) 或者 destination (-end) clock，用来调整 hold 关系的。而path_multiplier是指需要调整的时钟周期个数。
- set_multicycle_path -hold -start < path_multiplier >

以launch clock为基准调整hold关系， path_multiplier是调整的时钟周期数，在Effinity中，hold默认是以launch clock为参考的。

- set_multicycle_path –hold -end <path multiplier >

以latch clock为基准调整hold关系， path_multiplier是调整的时钟周期数。

以launch /Capture clock为基准的hold 约束方法

1、launch clock和latch clock为同一时钟时的hold约束

2、 launch clock和latch clock为同源不同频时的约束

launch clock和latch clock为同一时钟时的hold约束

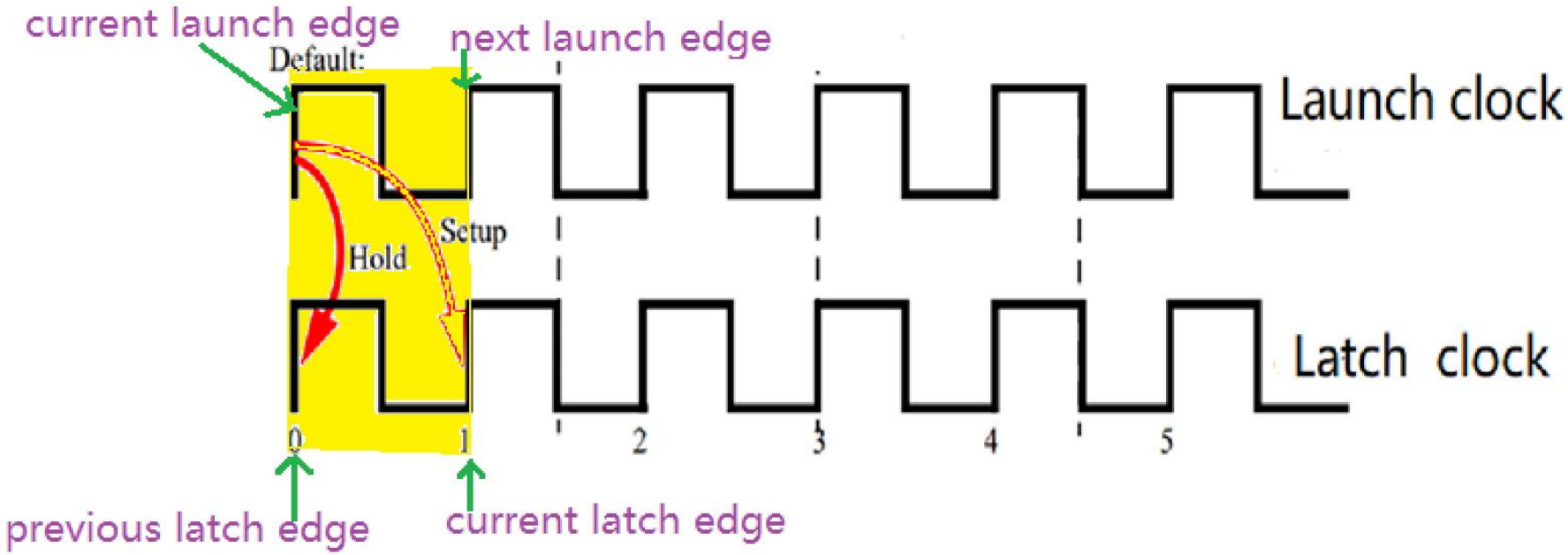
- Setup和hold都为默认值时的hold约束
- 只放宽setup时的hold约束
- 只放宽hold时的hold约束
- Setup和hold都放宽时的hold约束

Setup和hold都为默认值时的hold约束

- 默认情况下，hold和setup的捕获构成一个时钟周期组成的capture window
- multicyle中setup的值是1，hold的值是0。hold在current latch clock edge的前一个时钟沿进行检查。
- 此时，在sdc中不进行multicycle约束，也可以按照下面的方式进行约束。

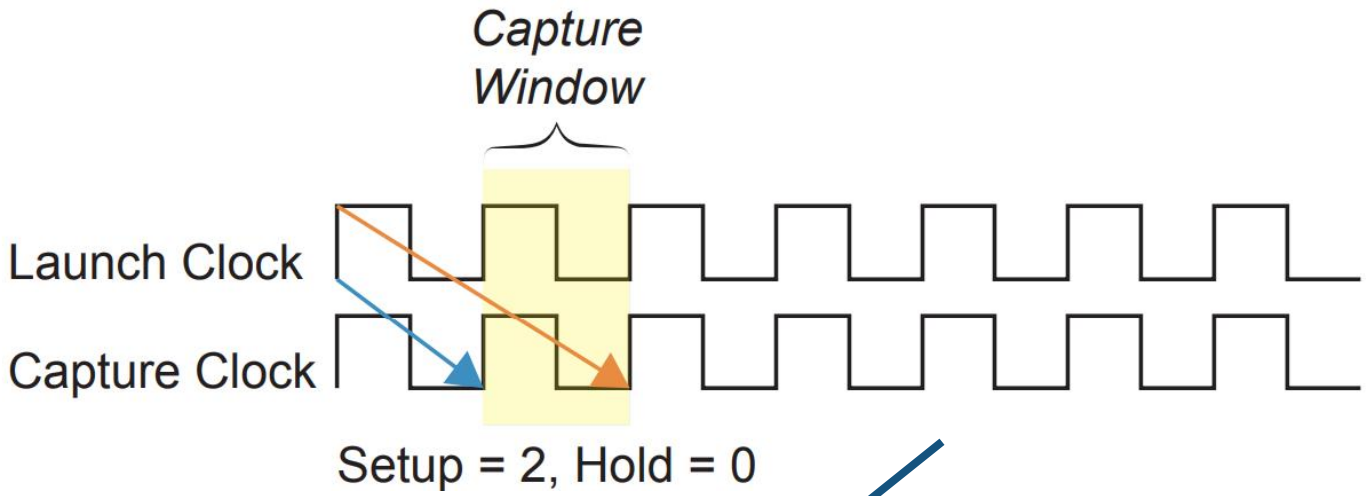
  set_multicycle_path -setup -from a -to b 1

  set_multicycle path -hold -from a -to b 0



只放宽setup时的hold约束

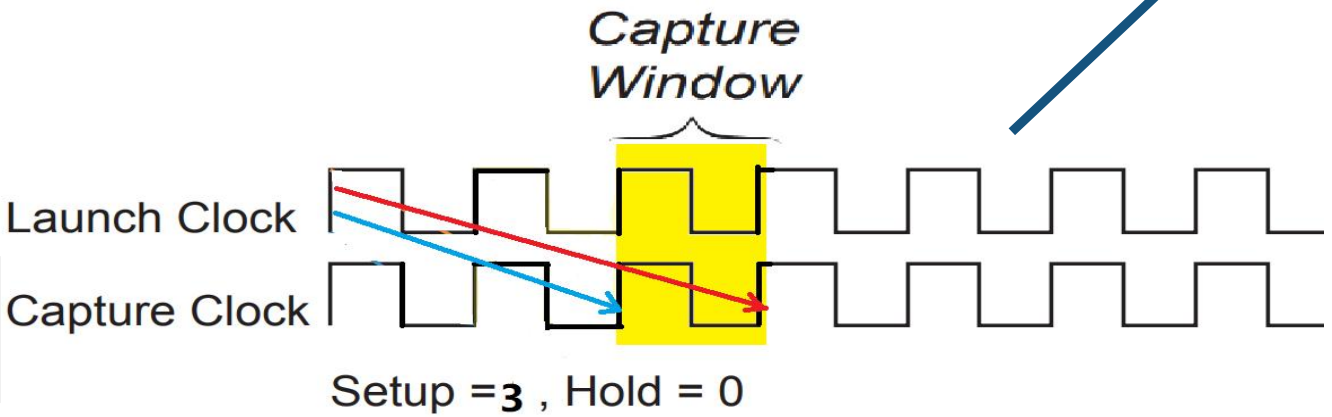
- 下边左图是放宽setup一个时钟周期，捕获窗口不变（此时hold=0）的情况。
- 下边右图是放宽setup两个时钟周期，捕获窗口不变（此时hold=0）的情况。




```
set_multicycle_path -setup -from a -to b 2
set_multicycle_path -hold -from a -to b 0
```

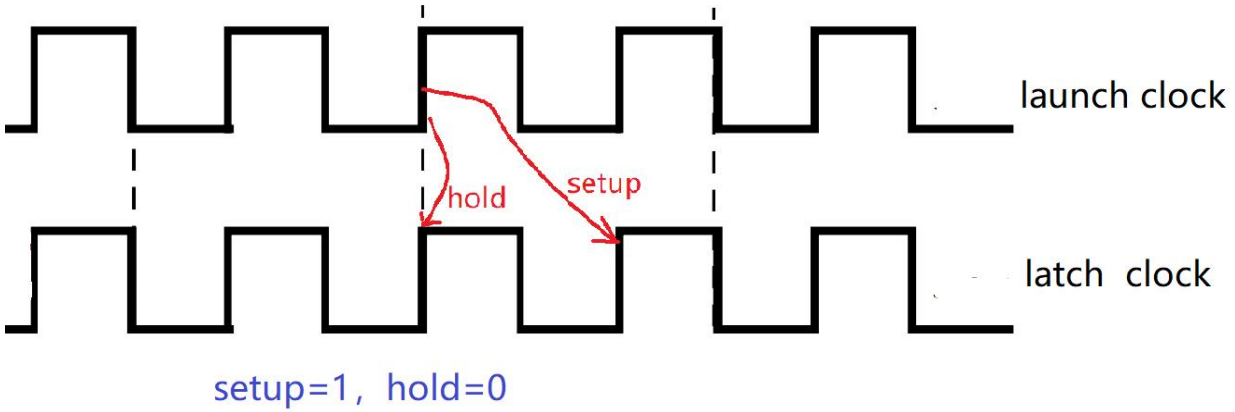
此时，SDC中的约束如下：

```
set_multicycle_path -setup -from a -to b 3
set_multicycle_path -hold -from a -to b 0
```

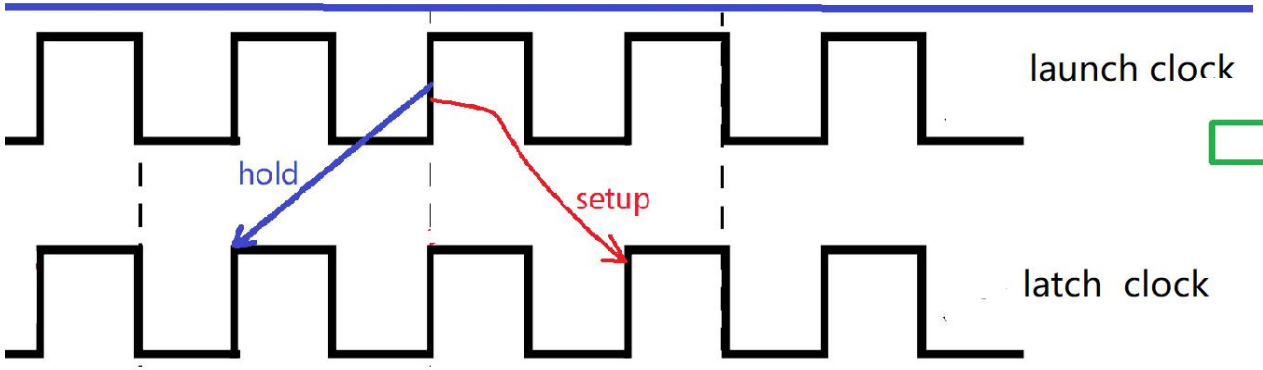


只放宽hold时的hold约束

- 假如把缺省情况的hold值放宽n个周期，于是multicyle中的hold值就是一个负值，而时序分析中不会报告multicyle出现负值的情况。如果在约束中出现了负值，Effinity的时序分析器会把launch edge和 latche edge 向右移动，以便使setup和hold的关系都变成非负。



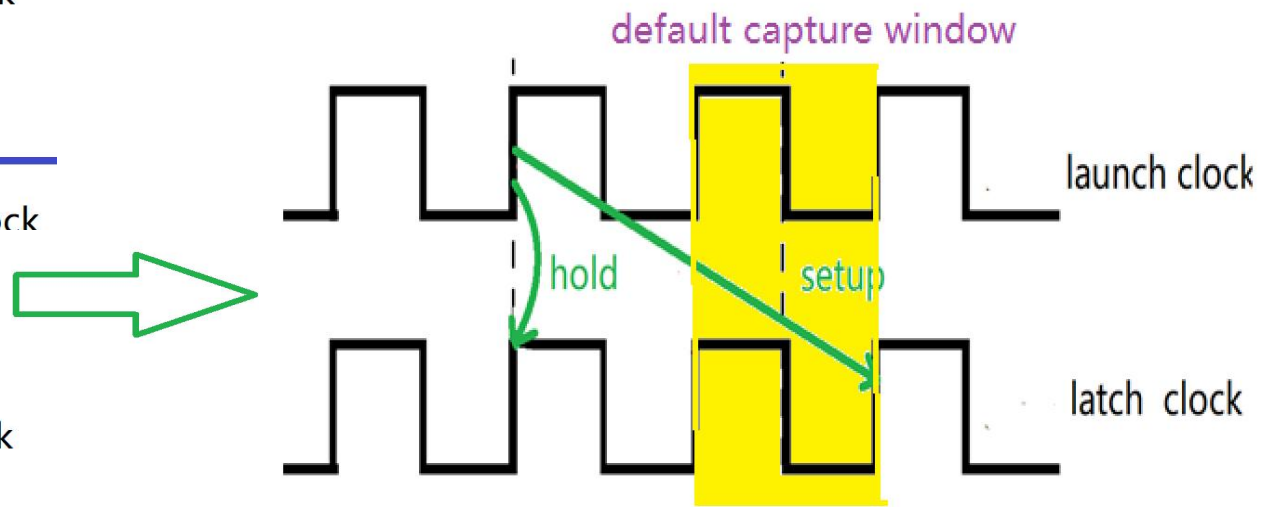
setup=1，hold=0



setup=1，hold=-1

multicycle constraint：

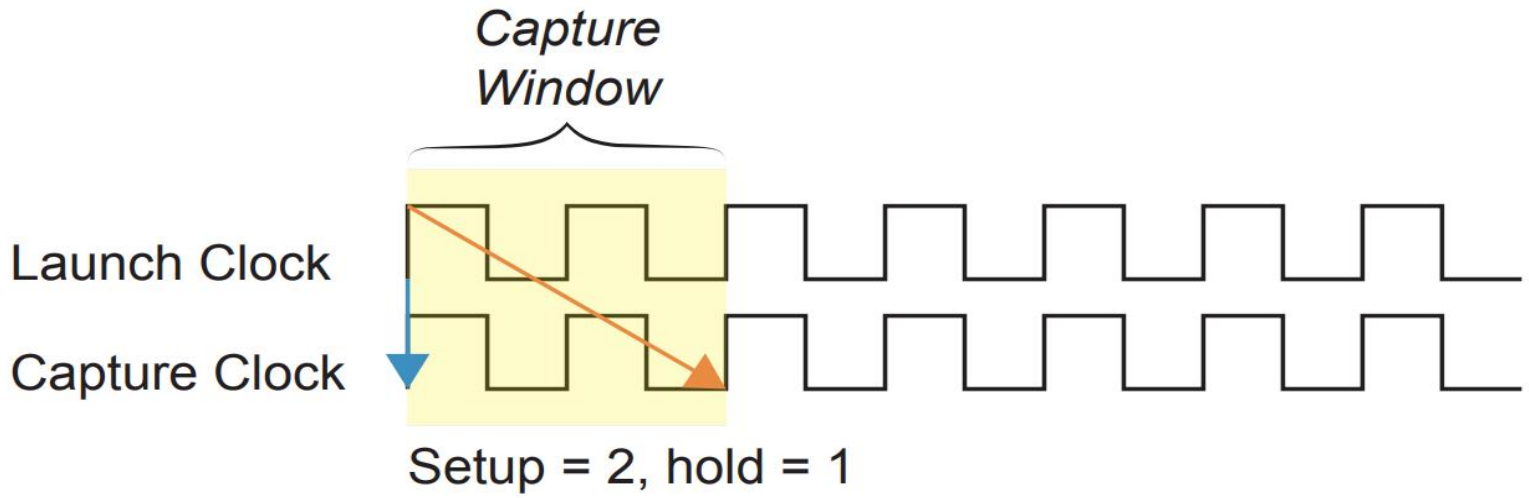
```
set_multicycle_path -setup -from a -to b 2
set_multicycle_path -hold -from a -to b 1
```



setup=2，hold=1

Setup和hold都放宽时的hold约束

- 下图是setup放宽了一个时钟周期，hold也放宽了一个时钟周期的情况。此时，捕获窗口因为hold放宽了一个时钟周期而增大了一倍。



此时sdc中的约束如下：

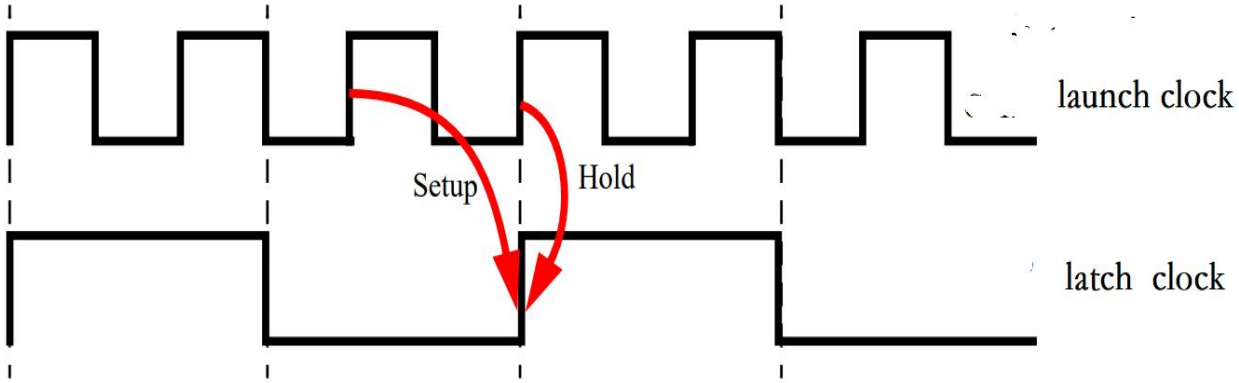
```
set_multicycle_path -setup -from a -to b 2
set_multicycle_path -hold -from a -to b 1
```

launch clock和Capture clock同源不同频时的约束

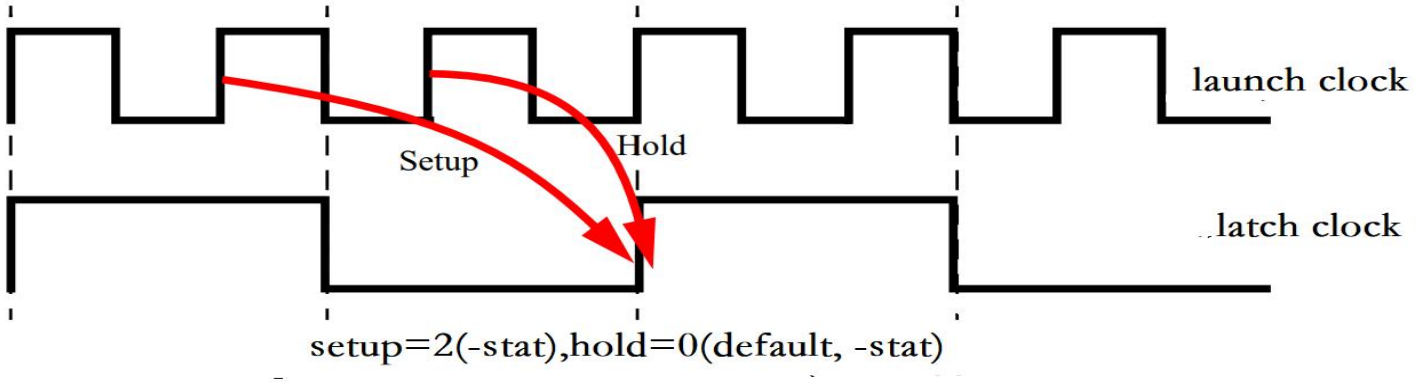
- launch clock频率是latch clock频率的n倍（n为大于1的正整数）
- Capoture clock 频率是launch clock频率的n倍（n为大于1的正整数）

launch clock频率是capture clock频率的n倍

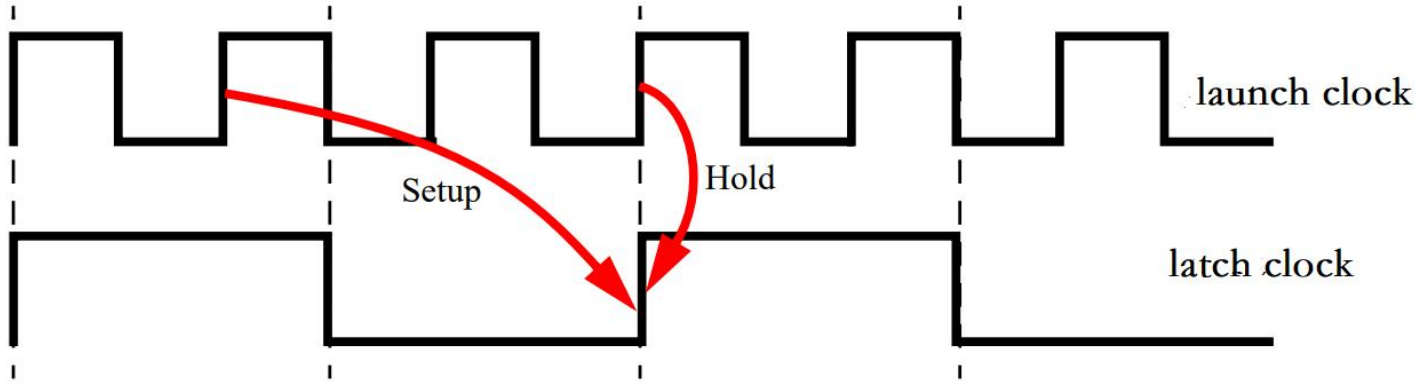
- 此时，使用set_multicycle_path 来放松对setup或者hold约束时，参考时钟使用 launch clock.。对setup约束来说，需要使用关键字“-start.”，对 hold来说，“-start“本来就是缺省设置，你可以加入这个关键字，也可以略去这个关键字。



缺省情况：setup=1（-end）hold=0（-stat）



launch clock频率是latch clock频率的n倍（续）

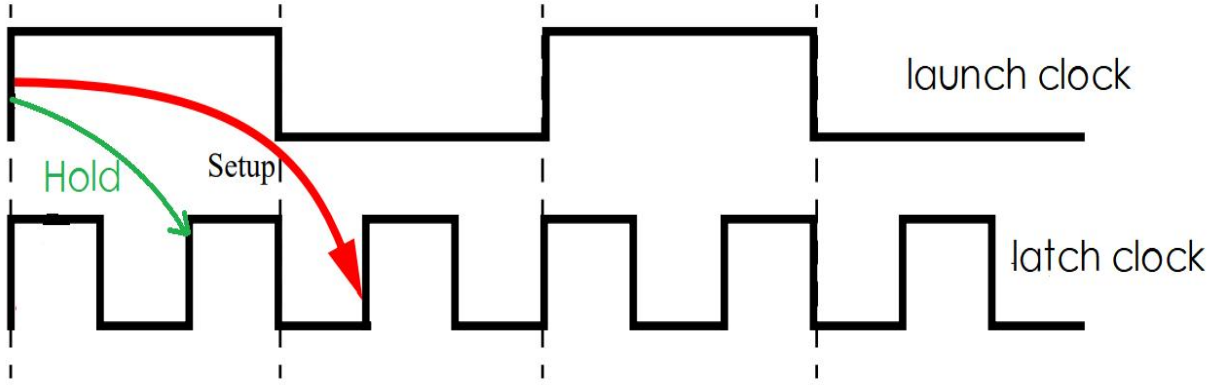


setup=2(-stat),hold=1(-stat)

```
set_multicycle_path 2 -setup -sat -from launch_clk -to latch_clk
set_multicycle_path 1 -hold -stat -from launch_clk -to latch_clk
```

Capture clock频率是launch clock频率的n倍

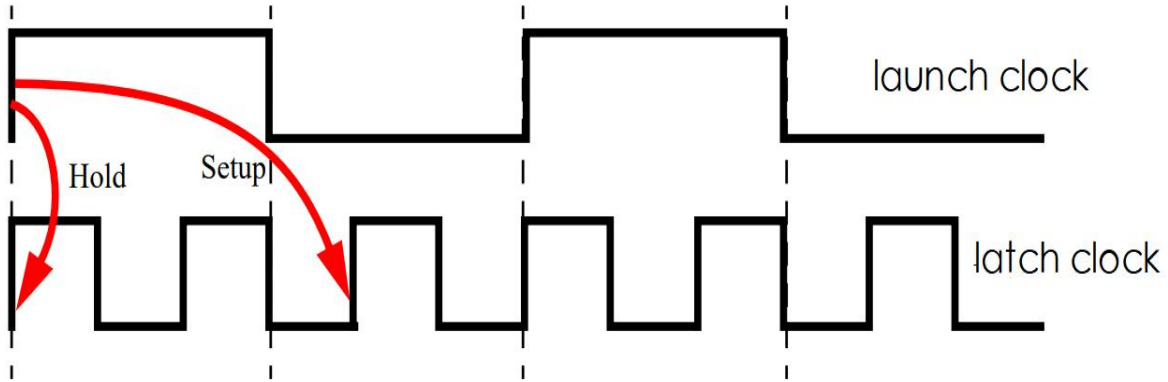
- 当Launch clock比latch clock慢的时候，此时只能使用比较快的latch clock作为set_multicycle_path的约束。此时，对setup约束来说，因为默认的就是以latch clock作为参考，此时可以使用或者不使用“”-end“”关键字。对于hold来说，默认的参考时钟是launch clock，所以，此时必须要使用“”-end“”关键字



setup=2（-end），hold=0（-end）

set_multicycle_path 2 -setup -end -from launch_clk -to latch_clk

set_multicycle_path 0 -hold -end -from launch_clk -to latch_clk



setup=2（-end），hold=1（-end）

set_multicycle_path 2 -setup -end -from launch_clk -to latch_clk

set_multicycle_path 1 -hold -end -from launch_clk -to latch_clk

使用Multicycle进行时序优化

主要内容

1、Multicycle应用的场合

2、使用方法举例

Multicycle应用的场合

1　同一个时钟域，源端数据有重复，但时序很难满足

2　目的端时钟频率比源端高几倍，时序很难满足

3　源时钟频率比目的端高几倍，但是连续几拍数据想同，时序很难满足

使用方法举例

- 同源时钟，目的端时钟频率是源端的几倍。

Multicycle 应用举例.docx

易灵思

感谢您的观看

www.elitestek.com

关注公众号

关注哔哩哔哩