DC 中关于 clock latency 的设置

DC 中，约束的时候，需要添加关于 latency 的约束。约束 latency 的命令是 set_clock_latency。使用 man 命令，查看。这里，有个技巧，可以将 man 的结果重定向到一个临时文件中，可以可以方便查看。

sh 表示使用 shell 命令。

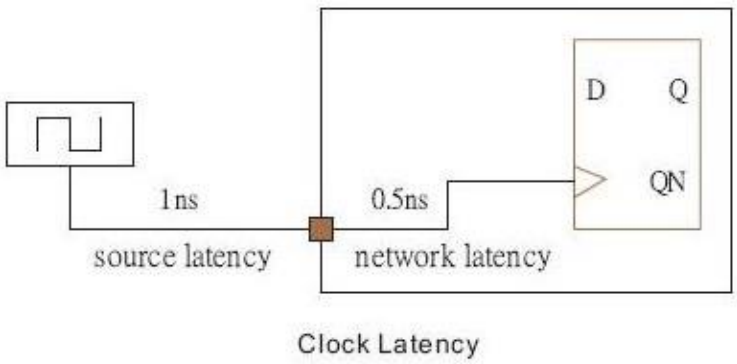
```
dc_shell> man set_clock_latency > tmpfile
dc_shell> sh gvim tmpfile &
2626
dc_shell>
```

使用的方法如图所示，可以看出该命令有很多参数的。

```
2.  Synopsys Commands                                   Command Reference
                              set_clock_latency

NAME
       set_clock_latency
              Specifies clock network latency.

SYNTAX
       status set_clock_latency
                [-rise]
                [-fall]
                [-min]
                [-max]
                [-source]
                [-early]
                [-late]
                [-dynamic dynamic_component_of_delay]
                [-clock clock_list]
                delay
                object_list

   Data Types
```

先来说一下-source 参数。DC 中将 latency 分成了两部分，一个是芯片外的 source latency，一个是芯片内的 network latency。

用下面一个图就可以说明了。



Clock Latency

source latency 是外部 clock 信号来源到芯片的 clock 输入端的 delay，而 network latency 是指芯片 clokc 输入端到 flip-flop clock 输入的 delay。所以对于上面的图，所施加的约束就是

set_clock_latency -source 1 [get_clocks CLK]

set_clock_latency 0.5 [get_clocks CLK]

当不指定-source 参数时，表示是对 network latency 进行约束。

对于-rise 和-fall，这两个参数比较简单，针对时钟的上升沿还是下降沿进行约束，不指定的话，表示对上升沿和下降沿约束的参数是一样的。

对于-min 和-max，这两个参数从命名就知道作用是什么。这里就不解释了。

重点是-early 和-late 参数的理解。在网上找到这样的描述

For setup analysis, Design Compiler uses the late edge for the launching flip-flop and the early edge for the capturing flip-flop. For hold analysis, Design Compiler uses the early edge for the launching flip-flop and the late edge for the capturing flip-flop.

说明这两个参数是对于建立时间和保持时间分析有用的。

在 eetop 上看到有人举例子：

set_clock_latency 4 -source -early ....

set_clock_latency 5 -source -late....

在计算 setup 的时候 ： launch clock 上的 latency 用的是 5，capture clock 上的 latency 用的是 4.

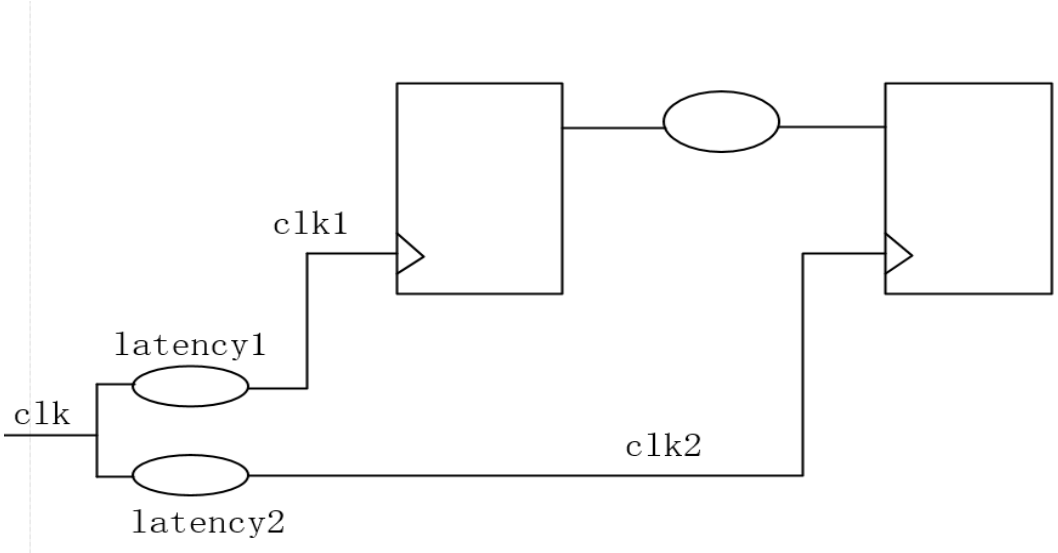
在计算 hold 的时候： launch clock 上的 lactency 用的是 4，capture clock 的 latency 用的值是 5.

其实这两个参数，就是指定在建立时间和保持时间分析的时候，launch clock 和 capture clock 的 latency。有了这两个参数，能加紧约束。

	Setup	Hold
Launch clock	late edge	early edge
capture clock	early edge	late edge

如果上面的例子你没有看懂的话，说明你的建立时间和保持时间分析基本功还不够到位了。

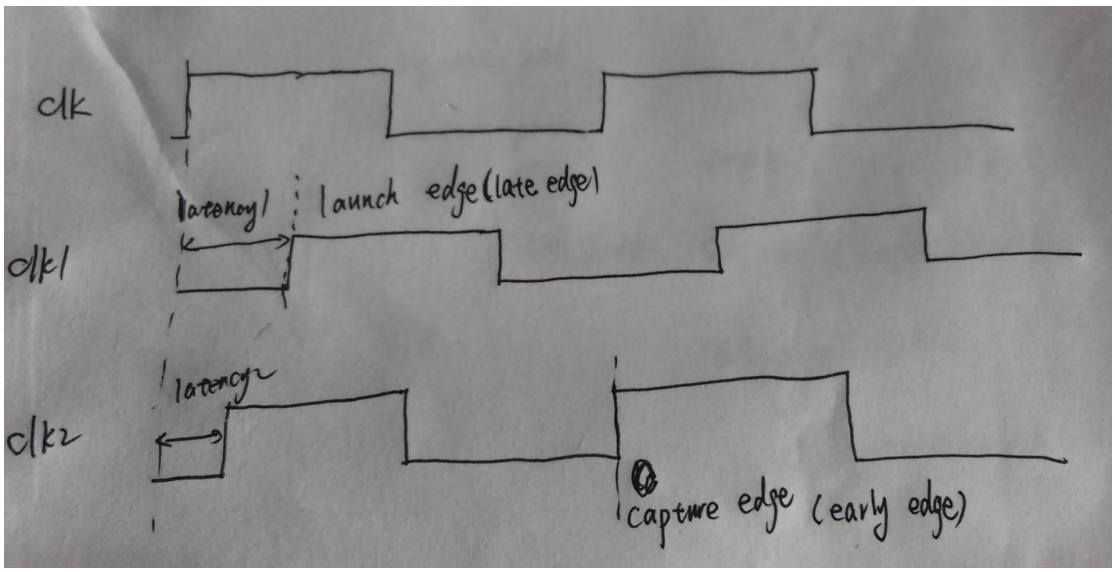
下面，我就来说明一下：



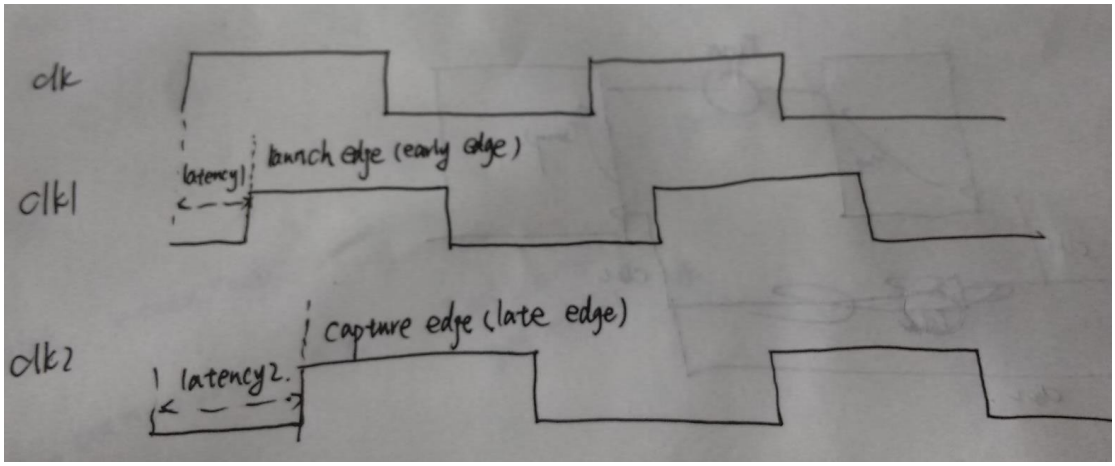
对于建立时间，考虑上面的电路图。从时钟 clk 出来到寄存器的 clk 端口是有延时的，

对于 clk1 的延时是 latency1，对于 clk2 的延时是 latency2。当然这延时是不确定的，有可能 latency1 大于 latency2，也可能反过来。

考虑最差的情况。latency1 大于 latency2。这样的话，有效的时钟周期时间就变成 T-(latency1-latency2)了，那对建立时间的约束就变紧了，因为有效周期小了。



对于保持时间，还是上面的电路图。考虑最差的情况，latency1 小于 latency2。约束的保持时间延时就要大于（latency2- latency1）了，对保持时间的约束也就变紧了。



以上，就是分析了这两个参数的作用。

-dynamic 参数，是指定时钟源的抖动的时间。

```
The dynamic component of clock latency is the amount of jitter  in  the
original clock source, which leads to a reduction in timing slack equal
to the specified amount of time. In the case of a zero-cycle  path,  in
which  the same clock edge both launches and captures data in the path,
jitter does not reduce the slack, which is taken into  account  in  any
clock reconvergence pessimism removal adjustment performed on the path.
Or provide whatever usage information that might help  the  user  takes
advantage of this feature.

The  dynamic  component of any clock latency can be specified using the
-dynamic option.  It can be specified only if the - source  option  and
total clock latency ARE also specified with the command.
```

delay，指定 latency 的延时值。
object_list，指定施加 latency 约束的目标信号
理解了上面的参数的信息，对于 latency 约束就比较容易了。