# .synopsys\_dc.setup文件

designer = "gtic\_liangx"

company = "GTIC"

search\_path = "/home/project/eda\_soft/opt/SYNOPSYS/syn2007/libraries/syn/"

search\_path = search\_path + "./src" + "./db"

link\_library = {"\*", "class.db", "and\_or.db", "dw\_foundation.sldb"}

target\_library = {"class.db", "and\_or.db"}

symbol\_library = {"class.sdb", "generic.sdb"}

synthetic\_library = {"dw\_foundation.sldb", "standard.sldb"}

define\_design\_lib WORK -path ./work

search\_path变量告诉DC在哪些目录搜索设计文件、约束文件和库文件。

link\_library变量定义了被调用的设计（例如：工艺库中的IO单元、第三方IP如RAM、DW库中的基本单元、工艺库中的基本单元）的集合。

注：

在定义link\_library的时候总是会首先加上“\*”，这是很有必要的，表示DC首先会在已经载入内存的数据库中搜索调用。

此外，只有当Verilog源代码包含了link\_library中单元的例化，才需要使用link\_library。link\_library的后缀名可以是“.db”或“.sldb”。

target\_library即目标工艺库，根据目标工艺库将设计映射成目标工艺库中基本单元的组合。

在实际项目目录下的.synopsys\_dc.setup文件主要给出了搜索路径信息（search\_path）、目标工艺库文件名（target\_library）、链接库文件名（link\_library），其实DW库不需要指定，Synopsys DC软件会自动加上去。

目标工艺库的具体信息都包含在目标工艺库文件中，我们知道我们需要指定最好和最坏工作条件，设计的环境属性，这就需要从工艺库文件中获取这些信息。

那么，如何来获取目标工艺库内包含的信息呢？

首先，我们可以从target\_library变量中获取目标工艺库文件名，然后使用read\_db命令读入所想查找的工艺库的具体信息，read\_db命令使用格式如下：

read\_db <target library file name>

将目标工艺库读入到内存后，使用list\_libs命令查看当前读入了哪些目标工艺库，list\_libs命令的结果中显示了读入的目标工艺库的名字，通过目标工艺库名使用report\_lib命令获取该目标工艺库的详细信息，由于信息一般会分为很多页，所以建议使用redirect命令将report\_lib命令输出的结果写入一个文本文件中以方便查阅。

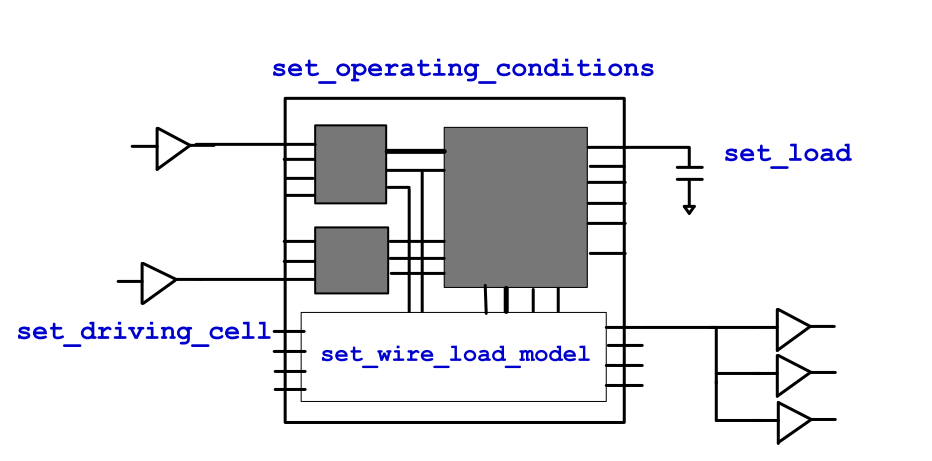
list\_libs

report\_lib <target library name>

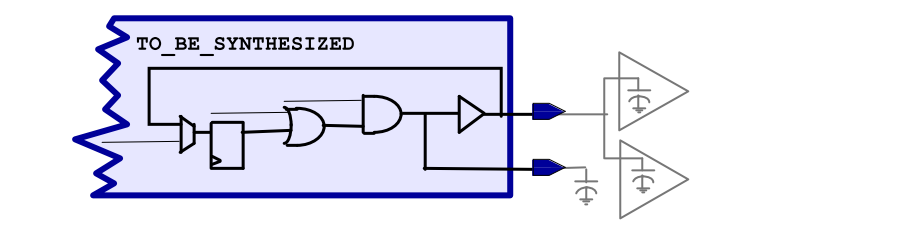
redirect –file lib.rpt {report\_lib <target library name>}

注意：report\_lib命令使用的是目标工艺名，前面read\_db命令使用的是目标工艺文件名。

# 对设计周边环境的描述



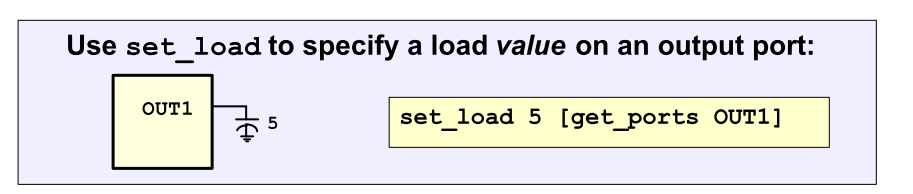
## 输出容性负载的指定



为了精确计算输出端口的时序，DC需要知道输出端口驱动的总的容性负载的大小。

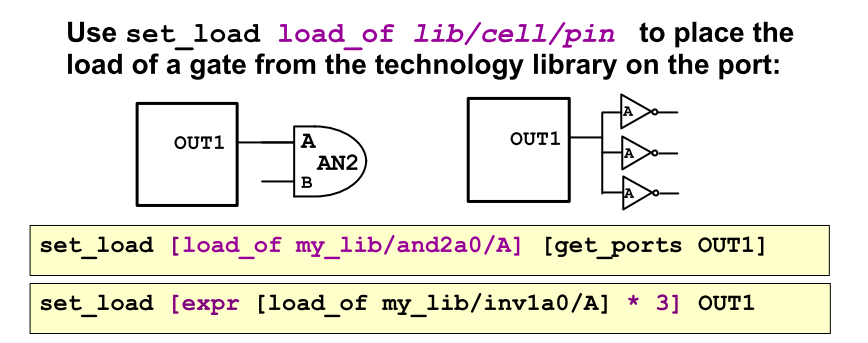
用户可以使用set\_load命令指定输出端口驱动的电容大小。需要注意的是，在默认情况下，即用户不指定输出端口驱动电容大小时，DC认为输出端口的负载为0。下面几个例子介绍set\_load命令的使用方法。

例1：直接指定输出端口所驱动的电容值：



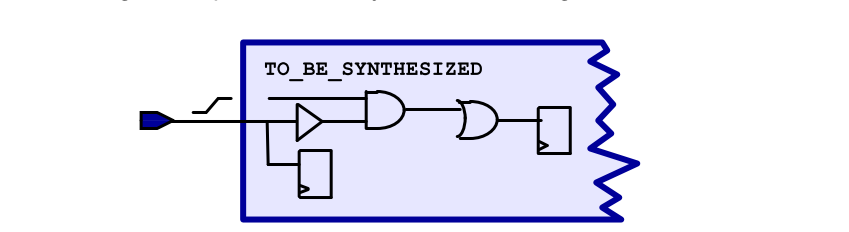
例2：指定输出端口驱动的负载为工艺库中某个单元的某个输入pin。

命令格式为：set\_load load\_of lib/cell/pin [get\_ports OUT1]

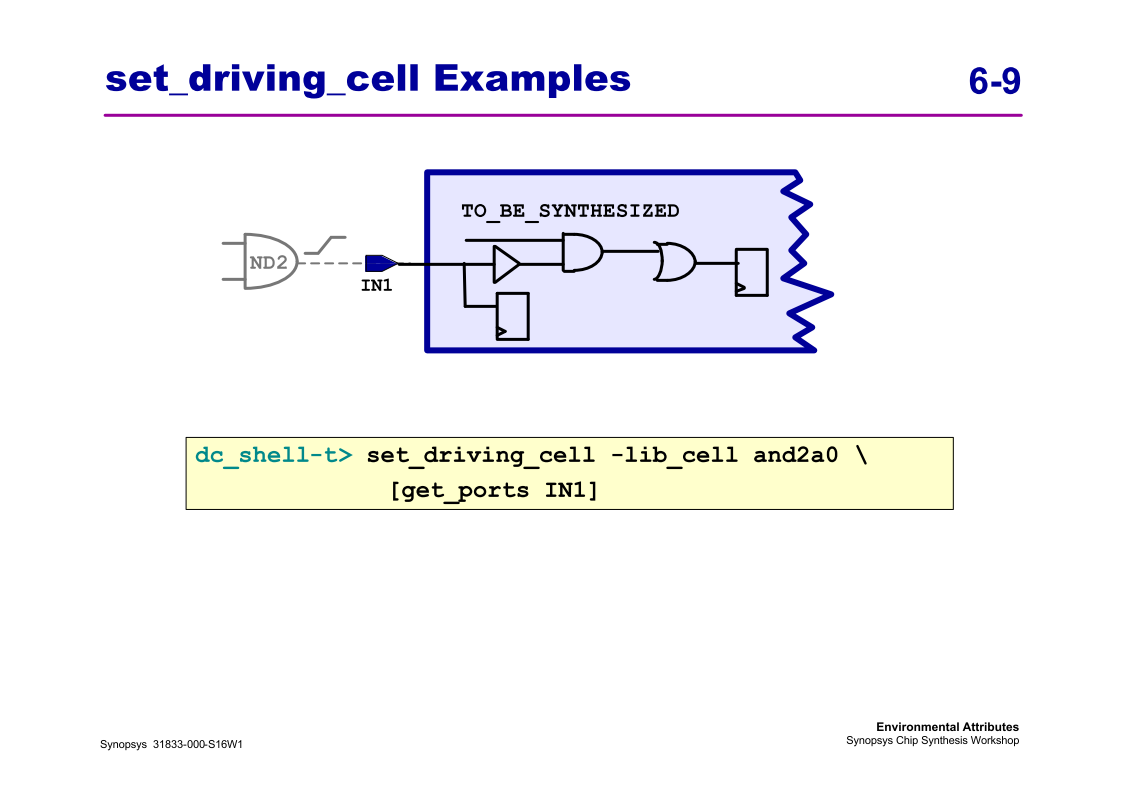


## 指定驱动输入端口的外部驱动强度

为了精确计算输入端口电路的时序，DC需要知道输入信号到达输入端口时的transition time。



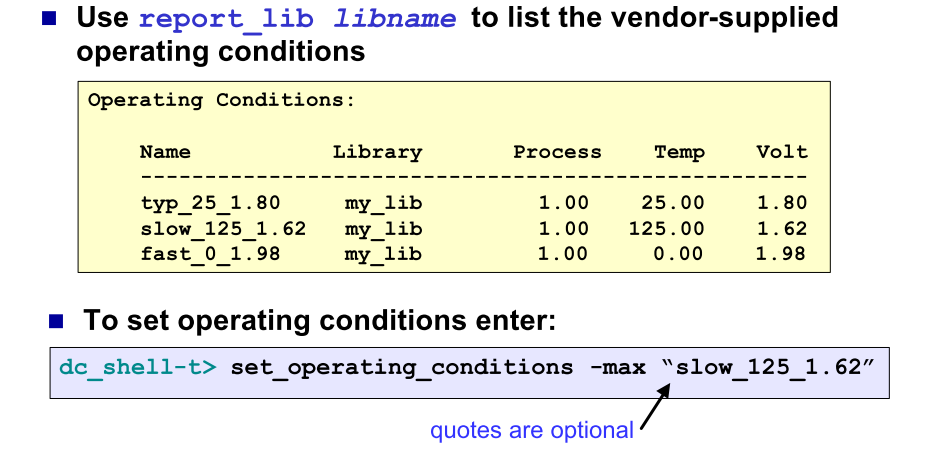
用户可以使用set\_driving\_cell命令来指定一个实际的外部单元来驱动输入端口。默认情况，DC认为输入端口处的信号的transition time为0。在标准单元库中选取一个单元来驱动输入端口，DC将会根据这个驱动单元计算出对应的达到输入端口的信号的transition time。如下面的例子中所述：



## 指定设计的工作条件

这里的工作条件是指温度范围、电源电压范围和工艺偏差范围。温度越低，延迟越小；电压越高，延迟越小；工艺越小，延迟越小。

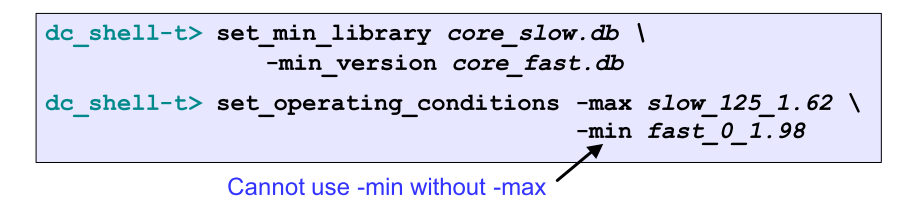
通过set\_operating\_conditions命令来指定工作条件。DC不会为设计指定默认的工作条件，需要用户手动指定。工艺厂商提供的工作条件可以通过report\_lib lib\_name来查看。下面这个例子介绍了如何指定工作条件。



工艺厂商可能会提供多个工艺库，这些工艺库分别定义：

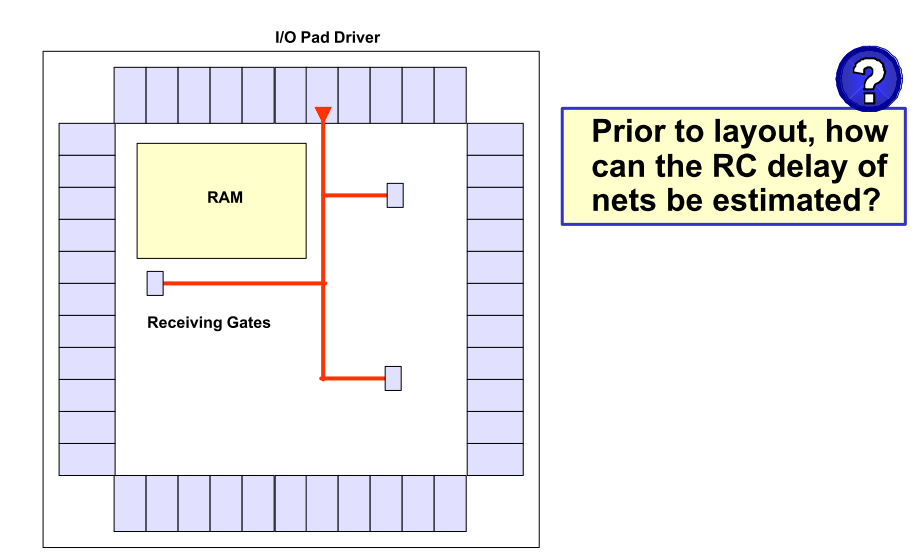
* 最好情况的工作条件和最坏情况的工作条件；
* 最乐观的WLM和最悲观的WLM；
* 最小时序延迟和最大时序延迟。

因此可以同时指定最好和最坏情况的工作条件和工艺库。如下面的例子中所述。



## 指定连线的延迟模型

在layout之前，该怎么估计连线的RC延迟呢？



对于一个balanced clock tree（e.g. 3 levels），第一level通常是一个由pad驱动的buffer。为了减小clock skew，第一level的buffer通常被放置在芯片的中心位置附近。这样摆放的好处是：第一level的buffer到第二level的buffer的连线长度相等。这样就形成了一个环形的结构，第一level的buffer在环的中心。第三level的buffer围绕着第二level的buffer。这样，第一、第二、第三level之间距离最短。但是，这样做也会带来一个问题，第一level的buffer摆放的位置离pad很远。如果使用最小线宽的连线连接pad到第一level的buffer，会带来一个较大的RC延迟，从而影响到clock latency。因此，很有必要将pad到第一level的buffer的输入的连线加宽，从而降低连线的电阻，从而降低总体延迟。取决于设计的规模和level的数量，用户可能在其他level上执行加宽连线的操作。

为了最小化clock skew，layout工具应该具备在时钟树任意一级抽取信号的能力。这一点对于包含gated clock的设计十分重要。如果同一个时钟还同时用于ungated flip-flops，那么gated flip-flops和ungated flip-flops之前的clock skew因为gated clock引入的延迟将会增大。如下图所示，从图中可以看出，到达gated flip-flops的时钟经过的延迟比到达ungated flip-flops的时钟经过的延迟多了一级延迟。因此，如果从时钟树的上一级中抽取时钟信号给gated clock能够减少clock skew。

