VLSI 物理设计方法学

发布 *V0.1*

QIAMKING

2021 年 08 月 14 日

Contents:

CHAPTER 1

前言

1.1 To do

1. 单元库的建立（具体）
2. 单元库的类型介绍（具体）
3. 时序库的建立-> SPICE、器件延时理论模型、线负载模型
4. 功耗库
5. 噪声库
6. 单元 *LEF* 文件的分类（具体）
7. 工艺角
8. 天线效应
9. PVT 条件
10. PDK
11. Physical Design Implementation Style
12. ESD
13. electromigration
14. IR Drop
15. technology file (*.tf*)
16. TLU file
17. port 与 pin 的区别

1.2 基本概念

1.2.1 工艺角

最常用的工艺角为：

- 最佳 / 最快–BC / best case / fast
- 典型 / 正常–TC / typical case / normal
- 最差 / 最满–WC / worst case / slow

1.2.2 PVT 条件

process（工艺）、temperature（温度）、voltage（电压）

1.2.3 PDK

PDK（process design kit）主要包括技术文件（technology file）、器件模型、原理图符号（schematic symbol）、参数单元（P-Cell）、验证文件集（DRC/LVS/EXT）

1.3 时钟信号

系统时钟、全局时钟的概念

1.3.1 时钟信号的生成

1.3.2 时钟信号的特性

1.3.2.1 时钟信号的定义

1.3.2.2 时钟信号的延滞（latency）

时钟信号的延滞（latency），又被称为插入延迟（insertion delay），包括两部分：

- 时钟源插入延迟：来自时钟源到当前芯片时钟根节点之间的延迟
- 时钟网络插入延迟：时钟树的延迟

1.3.2.3 时钟信号的抖动

时钟信号本身的抖动（jitter）定义为信号时间与理想时间时间的偏差（deviation），又称时钟抖动为 uncertainty，即为不确定性

1.3.2.4 时钟信号的偏差

同一时钟信号从时钟根到达任意两个寄存器的延迟（delay）之差称为偏差（skew）

1.4 设计约束

注解： 设计约束始终贯穿着集成电路物理实现的流程中，也是整个实现流程中体现物理设计方案质量的核心之一，本质是 EDA 工具通过开放设计约束入口，驱动内置封装的组合优化算法进行有方向有侧重的优化，从而尽可能实现设计者的目的

因此物理实现的最终电路结果是与设计者施加的约束条件密切相关，同时由于物理实现的工具繁多，每个工具使用过程中中均会引入相应的设计约束，将各个阶段的设计约束进行汇总分析的工作尚未完成（**待查**）

P.S. 为了检验电路是否满足约束，自然会各类分析手段，诸如时序分析、功耗分析、信号完整性分析

1.4.1 设计约束文件格式 sdc.rst

设计约束文件格式 *sdc*（Synopsys Design Constraints）文件基于 TCL（tool command language ）格式，由 Synopsys 开发定义，起初用于电路的逻辑综合。1998 年由 Cadence 公司移植用于时序控制的布局布线设计

警告： *sdc* 文件针对电路的时序、功耗、面积等方面进行约束，从而使得芯片满足设计要求的规范。其是系统架构师、前端逻辑设计、验证设计、物理设计的接口文件与指导性要求，其正确性、完成性直接决定了芯片的最终性能

1.4.2 时钟方面的约束

给出“时钟信号”（时钟定义、时钟延滞、不确定性），进而为布局、时钟树综合、布线提供时序约束参考点

注解： 时钟定义将通过时钟树综合来实现，时钟延滞、时钟抖动将在静态时序分析中进行检查

1.5 Design Flow

1.5.1 Implementation

Synthesize ，*Floorplanning*，*Placement*，*CTS*，*Routing*

从布图规划开始，到电源规划到完成布局，每一步规划不仅仅要考虑布线的实现，还要在布局完成之后时提供好的数据环境，供时钟树综合使用

1.5.2 Verification

/chapter9/index

1.6 参考

- 陈春章, 艾霞, and 王国雄. 数字集成电路物理设计. 科学出版社, 2008.
- Golshan, Khosrow. Physical Design Essentials. Springer Science+ Business Media, LLC, 2007.
- IC Compiler Design Planning User Guide Version E-2010.12-SP2, March 2011
- https://www.vlsi-backend-adventure.com/floorplan.html#8
- https://lmr.fi/int/physical-design-pd-interview-questions-floorplanning/

CHAPTER 2

Library

2.1 逻辑单元类别

标准的单元库是由不同的功能电路组成的，根据其芯片中应用场景可以分为三类：

- 标准单元（standard cell）: 放置于芯片的核心区域以实现逻辑功能的粘接（glue logic）
- 宏单元（macro cell）：更为准确一点，称为 memorys and custom libraries，放置于芯片的核心区域，（至少）包括 RAM、ROM、IP、COT、时钟 PLL、DSP 等宏单元
- 输入输出单元（I/O pad cell）：放置于芯片核心区域的周围，用于芯片信号的输入、输出、电源供给

2.1.1 标准单元

2.1.2 宏单元

2.1.3 I/O 单元

输出输入单元（I/O Pad Cell）包括：

- power I/O Pad Cell：电源单元
- ground I/O Pad Cell：接地单元
- signal I/O Pad Cell：输入信号、输出信号、三态、双向单元

注解： 对于输入信号单元，最重要的是要考虑静电放电 ESD (electrostatic discharge) 的防护

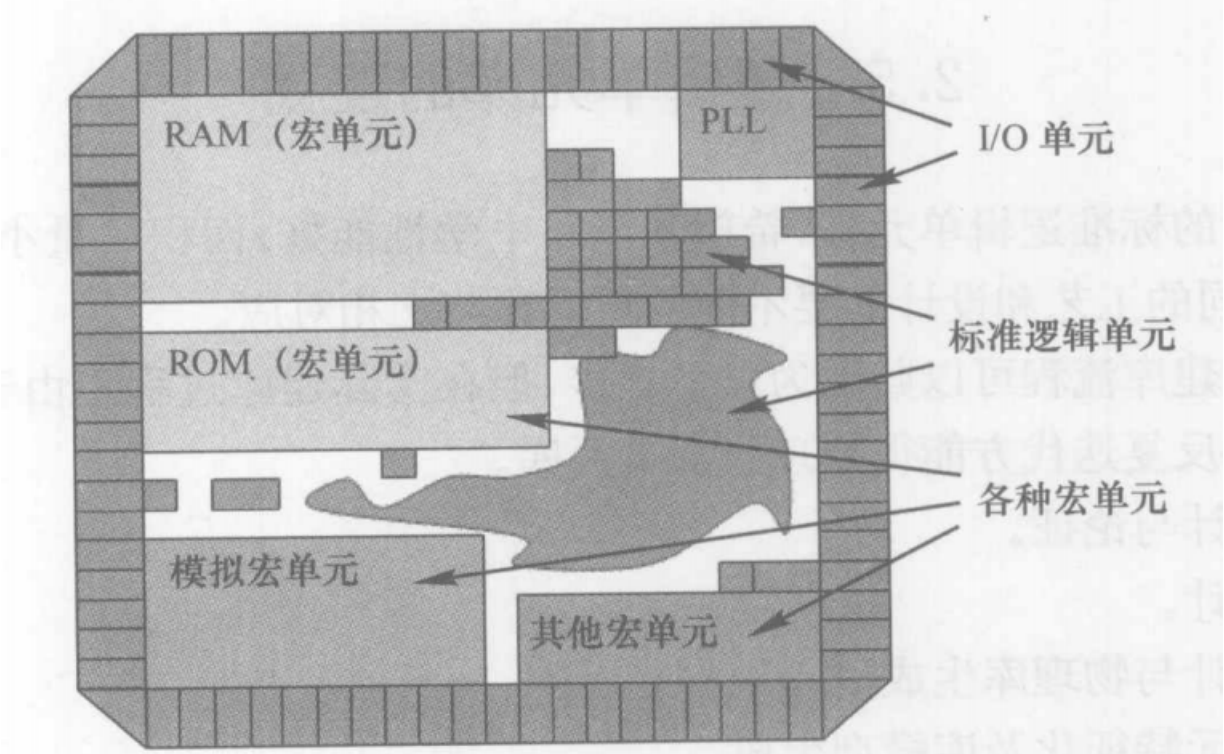


图 2-26　单元库种类

2.2 与单元库相关的标准数据格式文件

2.2.1 物理库交换格式 (LEF) 文件

物理库文件是对版图的抽象描述，使得自动布局布线成功可能且大大提高工具效率——其不考虑逻辑单元版图底层（即晶体管层面的 substrate、diffusion、polysilicon）的具体信息，只关心单元端口的属性，认为单元版图中除了端口外其余所有金属互连线都是不可布线区域

注解： 物理库的业界标准是 Cadence 公司开发的 *LEF* (library exchange format) 文件格式，后缀通常为 *lef* ，为了便于管理和应用，一般将其分为技术 *LEF* （technology LEF）、单元 *LEF* （cell LEF）两部分

2.2.1.1 *LEF* 的建立

需要什么：

- 单元库的 *GDSII* 文件（版图库）
- 逻辑库（时序库）：确定单元连接端口的输入输出属性（版图无法对端口的输入输出属性进行判定）
- 工艺信息：产生技术 *LEF*
- 映射文件：作为 *GDSII* 文件与 *LEF* 文件之间的转换标识文件

根据逻辑单元的类别，从物理单元的版图设计到建库（产生 *LEF* ）同样分为标准单元、I/O 单元、宏单元三种类型

2.2.1.2 技术 *LEF*

定义布局布线的设计规则、Foundry 的工艺信息（包括互连线的最小间距、最小宽度、厚度、典型电阻、电容、电流密度大小、布线轨道宽度、通孔种类等）

2.2.1.3 单元 *LEF*

定义标准单元、宏单元、I/O 单元和各种特殊单元的物理信息，例如针对布局阶段提供单元的仿真区域、对称性、面积大小；针对布线阶段提供单元输入输出端口的布线层、几何形状、不可布线区域、工艺天线效应参数

2.2.2 时序库文件

时序库： 是描述单元库各个单元时序信息的主要库文件，定义了每个单元不同输入情况下各个输入端口到输出端口的传播延时

工具通过仿真不同的工艺角（process corner condition）条件下电路的工作状态得到相应的时序数据，再将数据转换成工具可以识别的库交换文件用于芯片的时序分析

注解： 在建立时序库时，同时会关注时序库的应用条件，即 PVT 条件，常规情况下会考虑三种典型的工艺角条件下的晶体管中作情况

2.2.2.1 时序库的建立

时序库文件与物理库 *LEF* 文件是相互配合的一对文件，其所含有的库单元是根据 *LEF* 所含有的信息，根据器件延时理论模型、SPICE 网表进行仿真计算得来

2.2.2.2 时序库文件格式

- *liberty* 是目前业界广泛使用的时序库文件格式（最早是由 Synopsys 公司开发定义），通常以 *lib* 定义作为扩展名
- *ILM* 是 Synopsys 公司所开发，其时序关系简明准确地反映了接口边界的参数

下具体介绍 *liberty* 的特征：

2.2.2.3 时序库中的线负载模型

时序库中的器件延时与其输入输出信号的转换时间、输出端口负载相关，其采用理想的统计值，并基于线负载模型予以表达

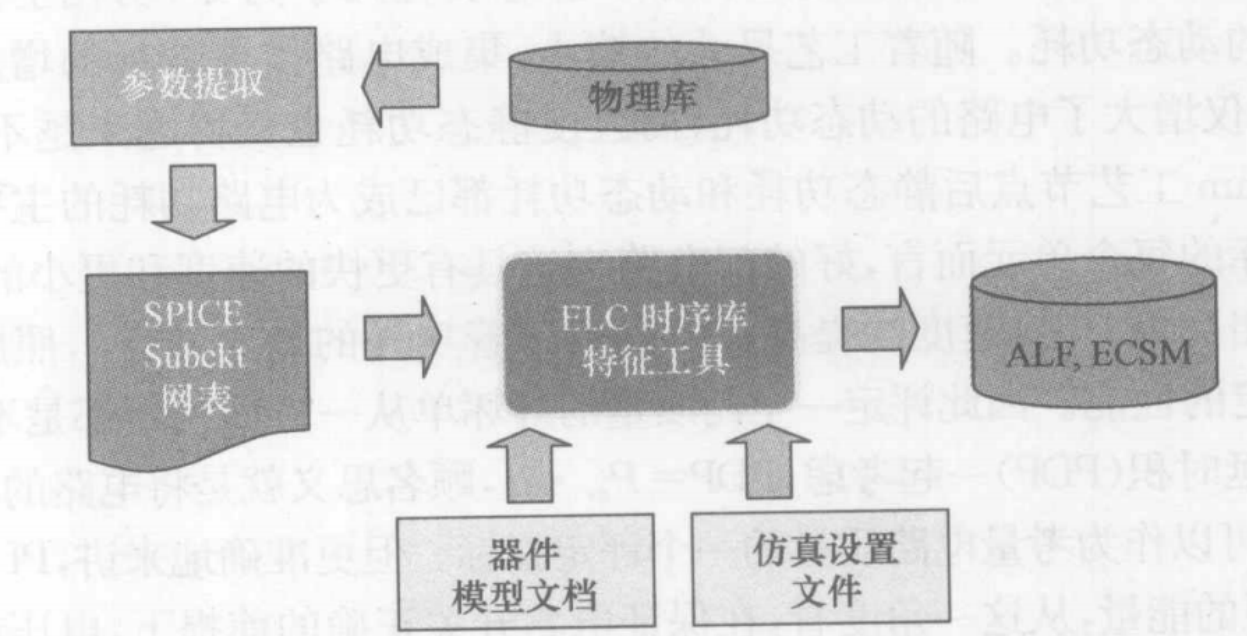


图 2-40 时序库产生的流程图(以 ELC 为例)

2.2.2.4 时序库单元信息

时序库中主要内容由单元信息完成，每个单元与物理库交换格式 *LEF* 文件中的单元一一对应，每个单元的信息包括：

- 不同时序模型下的延时时间表、功耗数值表，表述数据为不同输出复杂、输入转换时间条件下的函数
- 单元的特征，诸如其面积、静态功耗、端口名称
- 端口的逻辑关系
- 噪声

P.S. 端口分别有输入、输出、双向三种类型，其中输入端口包括它的负载电容值

注解： 基于单元库需要建立各种标准数据格式文件（这也是单元库建立流程的一大项工作），除了物理库之外，各种模型库文件用于描述单元库中各类单元电路在不同工作条件下的性能参数，用于电路仿真、逻辑综合、布局布线、时序分析、功耗分析等任务

---

e.g. 逻辑综合产生门级网表、物理设计实现布局布线的过程中需要 *LEF* 物理库文件、*lib* 时序库文件

抽象级别	文件格式	描述
电路级	SPICE/CDL 网表	用于器件级仿真、LVS 检查
符号级	逻辑图	用于逻辑分析, 包括单元名称、符号、输入输出端口
版图级	GDSII	记录版图的完整信息
网表文件	.v/.vhd	/
物理库文件	LEF	版图的抽象文件，主要用于布局、布线
时序库文件	liberty	用于电路综合、时序分析
功耗库	/	/
噪声库	/	用于信号完整性分析

2.3 单元库建立流程

逻辑单元的建库流程可以归纳为以下 5 步[1]：

- 方案设计、论证
- 电路设计
- 版图设计、物理库生成
- 标准单元特征化、库模型生成
- 设计验证

Libraries are collections of the physical layout, abstract views, timing models, simulation or functional models, and transistor level circuit descriptions.

[1] 在实际建库过程中由于不同因素的影响通常需要反复迭代才能得到理想的单元库

CHAPTER 3

Partitioning

3.1 Physical Design Implementation Style

3.1.1 展平式物理设计

采用自下而上的物理设计方案，芯片经过 RTL 设计仿真，通过逻辑综合产生门级网表以及相应的标准时序约束 *SDC* 文件，再调用由上述方法产生的基本单元（标准单元库）和大模块单元（包括 COT、IP、RAM、ROM、DSP），以及这些单元的时序库，通过布局布线实现物理设计，提取 RC 参数进行时序分析，最后产生 *GDSII* 文件

3.1.2 层次化物理设计

将庞大的设计在物理设计时分割（partition）为数个模块 block ，重点处理时序复杂的模块，进而缩短设计收敛的周期，使时序问题局部化

注解： 此处讨论的是自上而下的层次化物理设计方法，先将设计分成数个模块，再对每个模块进行展平化处理，包括独立的布局布线等过程，直到完成相应的建模，最后在顶层完成组装设计

CHAPTER 4

Synthesize

4.1 综合阶段中的设计约束

约束条件主要包括时序约束、面积约束、电路的环境属性、时序和负载在不同模块间的分配 ()

4.1.1 时序约束

4.1.2 面积约束

4.1.3 定义环境属性

4.2 综合结果评估

4.2.1 基于报告的指标分析

综合后可以输出多种结果报告，最常用的是:

- 时序/延迟报告，分析时序约束为例
- 面积报告
- 功耗报告

此时的时序分析是针对对芯片的时序进行理想情况下的 **零线负载模式**（zero WLM / Wire-Load Model）进行的延迟分析，从而验证综合后的网表是否具有较好的时序

4.2.2 综合后门级仿真

以 Design Compiler 为例，其流程如下：

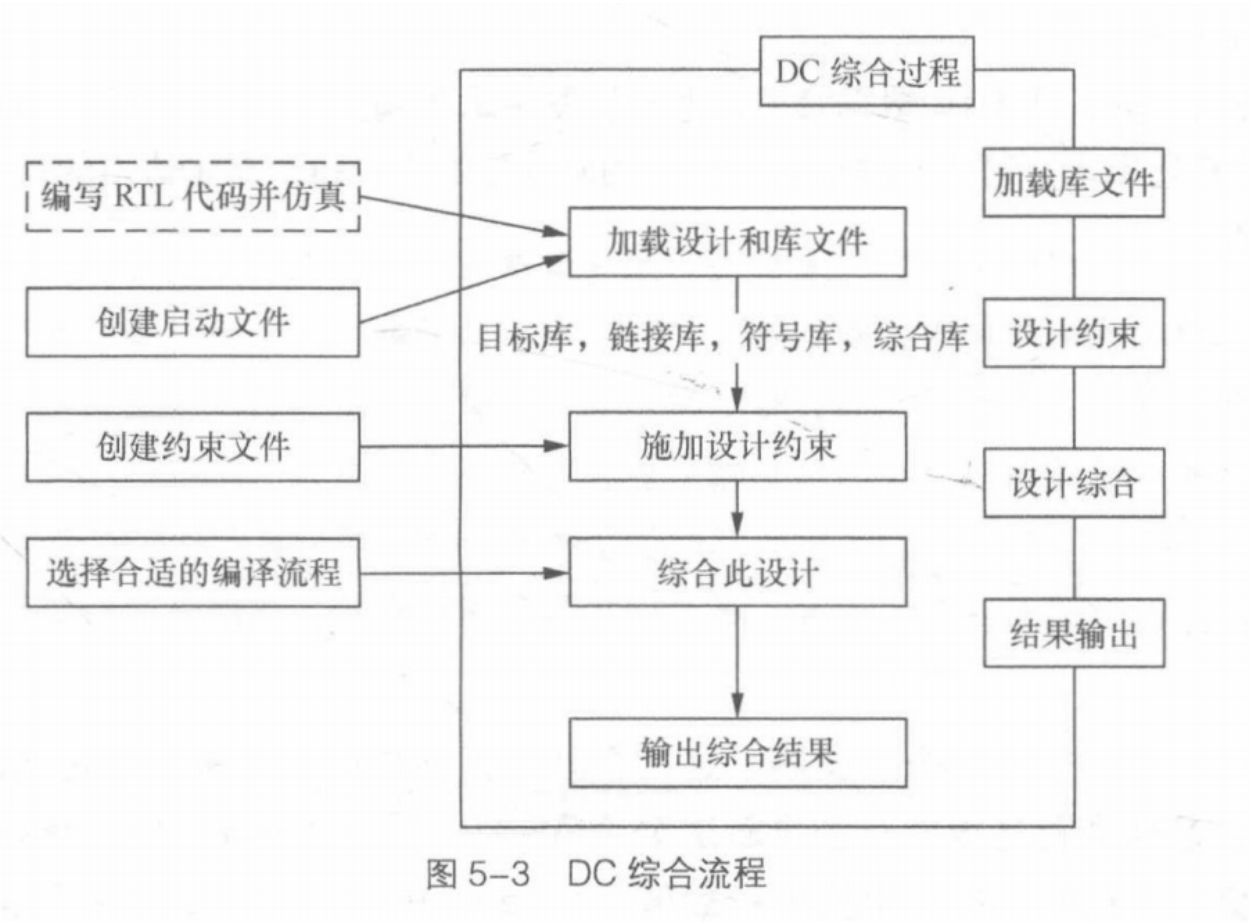


图 5-3 DC 综合流程

综合时基于设计约束、设计文件的信息，将 RTL 模块综合成门级网表，然后对综合出的网表进行评估，检验其是否满足约束条件

> **警告：** 对于整个芯片的综合，其综合过程会与单独的 RTL 模块的有一定的区别（待查）

CHAPTER 5

Floorplanning

5.1 Floorplanning 概述

Floorplanning 是对芯片内部结构的完整规划与设计——Regardless of the physical design implementation style, after physical database creation using the imported netlist and corresponding library and technology files, the first step is **Floorplanning**

5.1.1 Objectives of Floorplanning

- minimize the area
- minimize the Timing
- Reduce the wire length
- Making routing easy
- Reduce IR drop

5.1.2 What to do in Floorplanning

- 确定芯片面积：the size（width and height）of the core
- Macro / Block 的布局
- 标准单元的排列形式
  - the shape and placement of standard cell rows
  - the shape and placement of routing channels
  - standard cell placement constraints
- I/O 单元的布局 + 电源规划：the placement of
  - I/O cells

  - power cells
  - ground cells
  - corner cells
  - filler pad cells

5.2 布局 I/O 单元

It is critical to functional operation of an ASIC design to insure:

- the pads have adequate power and ground connections
- the pads are placed properly

目的：eliminate electromigration and current-switching noise related problems.

警告： 设置 I/O Pad Cell 约束

以 ICC 为例，在 initializing the floorplan 之前，首先为 I/O pad cells 设置约束：

```
$ set_pad_physical_constraints
```

约束内容为指定 the pad cell ordering, orientation, placement side, offset from die edge, and pad-to-pad spacing for each I/O pad

---

注解： 上述工作即为设置管脚约束的 *tdf* 文件了，它指定了每个 IO cell 在整个芯片中的位置和排列顺序——对于 Chip level 的设计而言，该文件主要用来定义设计中所有 IO 以及 IO Corner 的位置（上下左右的方位以及排列顺序，也可以定义具体的坐标）

如果是 Block Level 的设计而言，因为设计中没有 IO cell，所以可以读入 pin 顺序和位置的 *tdf* 文件——对于 Block level 的设计而言，它用来指定所有 pin 的位置和所用的 metal 的层次。

之后在执行

```
$ initialize_floorplan
```

在读入管脚约束文件（*.tdf* 文件）后，可以创建 Floorplan 得到芯片大概的物理形状和尺寸，在这一步执行完毕之后，IO pad cell 或者 pin 就会按照前面的约束进行摆放。

P.S. The *'initialize_floorplan'* command places constrained pads first. Any unconstrained pads are placed next, using any available pad location. The tool does not place unconstrained pads between consecutively ordered constrained pads

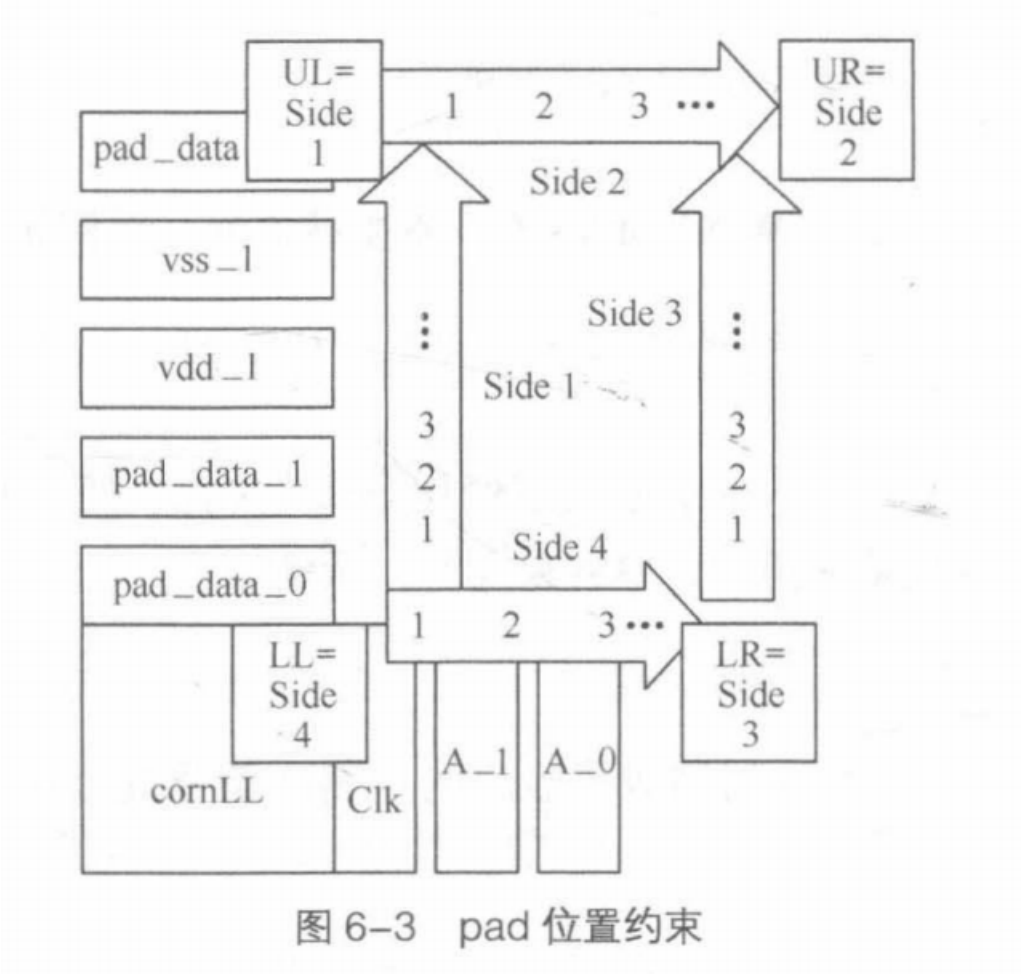
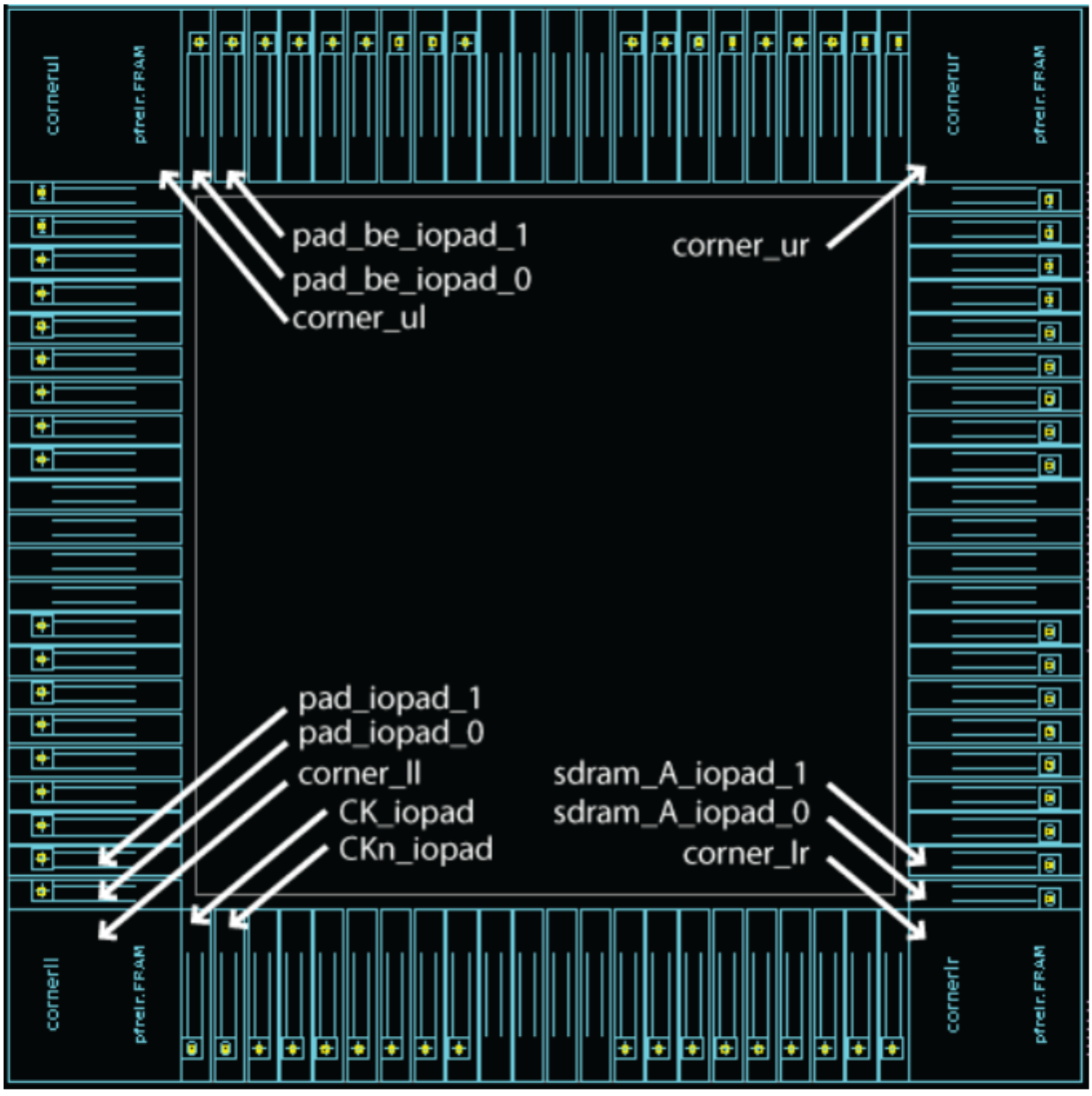

UL=
Side
1
1 2 3 ...
UR=
Side
2
pad_data
Side 2
vss_1
Side 3
vdd_1
Side 1
3
2
1
pad_data_1
3
2
1
Side 4
pad_data_0
1 2 3 ...
LL=
Side
4
LR=
Side
3
A_1
A_0
cornLL
Clk

图 6-3 pad 位置约束

Floorplan With Pad Placement



5.3 确定芯片面积

对芯片面积大小 (die size) 进行确定，主要基于两个方面进行考虑：

- 成本：面积越小，单张 wafer 的产出的裸片（die）数量增加，芯片的平均成本下降
- 布线质量：设定的裸片面积过小，造成布线堵塞（routing congestion），造成长周期的设计迭代

一个合理的面积设定是在保证布线同时尽量节约产品成本

芯片面积的确定体现在布局规划方案所使用的 Floorplan Control Parameters 上，主流上有如下两种方式

5.3.1 Aspect ratio

Aspect ratio is the ratio between vertical routing resources to horizontal routing resources.

$$\text{Aspect Ratio}(Ar) = \frac{\text{Horizontal routing resource}(H)}{\text{Vertical routing resource}(V)} \tag{5.1}$$

Aspect ratio 指定芯片高度和宽度比值的方案——If you specify a ratio of 1.00, the height and width are the same and therefore the core is a square; If you specify a ratio of 3.00, the height is three times the width.

注解： 当设计中不含 Macro 时可以采用 Aspect ratio 布图规划控制参数进行试探，在最开始 Trail 的时候一般设置较小的利用率，看 Timing、DRC、LVS 等结果如何，此时用时较短，能在短时间内给设计者一个参考，让设计者对 Floorplan 方案进行评估。如果设计的问题不大，那么可以逐渐提高利用率，减小芯片的面积。

5.3.2 Width and Height

指定高度和宽度，一般对于含有 Macro 的设计，多用这种方案

5.3.3 评估指标

5.3.3.1 Core Utilization (Cu)

Core Utilization (*Cu*) indicates the amount of core area used for cell placement.

The number is calculated as a ratio of the total cell area (for hard macros and standard cells or soft macro cells) to the core area.

$$\text{Core Utilization(Cu)} = \frac{\text{Standard Cell area}}{\text{Row area + Channel area}} \tag{5.2}$$

A core utilization of 0.8, for example, means that 80% of the core area is used for cell placement and 20 percent is available for routing.

5.3.3.2 Row to Core Ratio (Rcr):

Row to Core Ratio (*Rcr*) indicates the amount of channel space to provide for routing between the cell rows.

The smaller the number, the more space is left for routing. A value of 1.0 leaves no routing channel space.

$$\text{Rcr} = \frac{\text{Row area}}{\text{Core area (H * V)}} \tag{5.3}$$

5.3.3.3 Total utilization T(F)

Total utilization T(F) of floorplan F is derived using the following equation:

$$T(F) = \frac{A(m) + A(p) + A(s)}{A} \tag{5.4}$$

- *A(m)*： Area occupied by macros
- *A(p)*： Area occupied by Pads/ Pad fillers
- *A(s)*： Area occupied by Standard Cells

5.3.3.4 Cell row utilization

Cell row utilization *C(F)* of floorplan F is approximated using the following equation:

$$C(F) = \frac{A(s)}{A(R - \text{union}(B, E, m, p))} \tag{5.5}$$

- *R* = All cell rows
- *B* = All placement blockages
- *E* = Exclusive Regions

5.4 布局 Macro

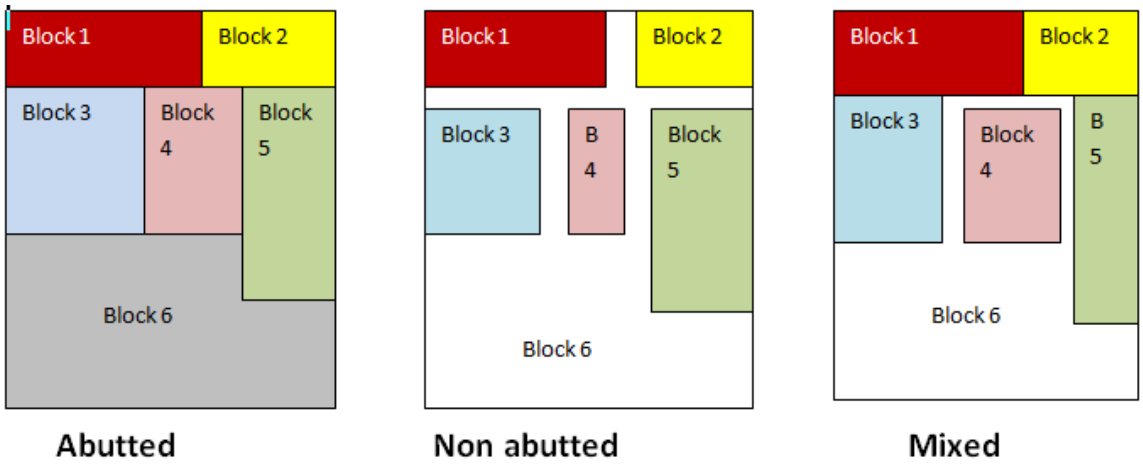
若采用展平式物理设计方案，则设计中仅仅由 Macro 和标准单元构成；若采用层次化物理设计方案，则经过 Partitioning 后，顶层设计主要由 Macro、sub-system 级别的 Block 构成

在 Floorplan 阶段，对于前者主要是考虑 Macro 的放置，后者则是要考虑 Macro、Block 的放置

注解： Before staring of Floorplan, it is better to have **basic design understanding** , data flow of the design, integration guidelines of any special analog hard IPs in the design. And for block/partition level designs understanding the placement & IO interactions of the block in Full chip will help in coming up with good floorplan.

Macro / Block 的放置显著依赖于设计者的经验，摆放它们时要考虑面积、互连线长等传统问题，还需要考虑 Macro / Block 的摆放对于 Place 的影响

5.4.1 (Macro) Floorplan Techniques



- Abutted floorplan : Channel less placement of blocks.
- Non-Abutted / Channeled floorplan : Channel based placement of blocks.
- Mix / Narrow-Channel floorplan : partially abutted with some channels.

5.4.2 Macro 放置原则：边缘摆放

边缘摆放的出发动机主要来源于下面两点：

1. 从目前芯片设计的趋势来看，芯片中除了计算单元外就是随机存储单元 RAM、只读存储单元 ROM 等。这些存储单元占据的芯片面积在有些设计中甚至超过百分之五十
    1. 对于存储单元来说，存在数据端口和存储端口，并且周围需要有一些可测性电路。这使得这些单元引线众多且功耗巨大
    2. 将它们 * 贴边放置 *，不仅有利于这些单元的供电，而且防止这些单元过多的引脚对其他单元的布线造成影响。
2. 标准单元在布局时，按照 Row 所划定的高度一排一排的摆放
    1. 既有利于算法的设计，又有利于工业制造。
    2. 在给各个器件供电时，可以使用横向的电源线将处于同一高度的器件连接在一起统一供电
3. 将标准单元都摆放在芯片区域的中心，而大的 Macro 摆放在四周，就可以使标准单元方便的只用一条电源线连接在一起，而不会被高度不统一的 Macro 打断。对电源网格的设计提供了巨大便利

Macro 的摆放原则基本如下，可以参照下面这张图

1）Macro 尽量摆放在靠近相应输入输出口（IO pad cell）的位置

一般来说对于大型的 Macro，他们不仅仅需要与芯片内部的其他 Macro 或者标准单元进行数据交换，还需要与芯片外部的器件进行通信。

比如，锁相环单元需要接收外部晶振信号，存储单元需要接收外部地址等。这种数据交换就是靠 IO pad cell 进行的，因此摆放在离相应的数据端口附近，有利于减少互联线长度，减少线上延迟，并节约布线资源。

2）大的 Macro 摆放尽量贴近版图的边缘和角落，这样有利于空间的利用，要尽量留出一个连续且尽量接近圆形/方形的 Core 区域来摆放标准单元。

3）Macro 与 Macro 之间要留有一定空隙，给予布线资源。

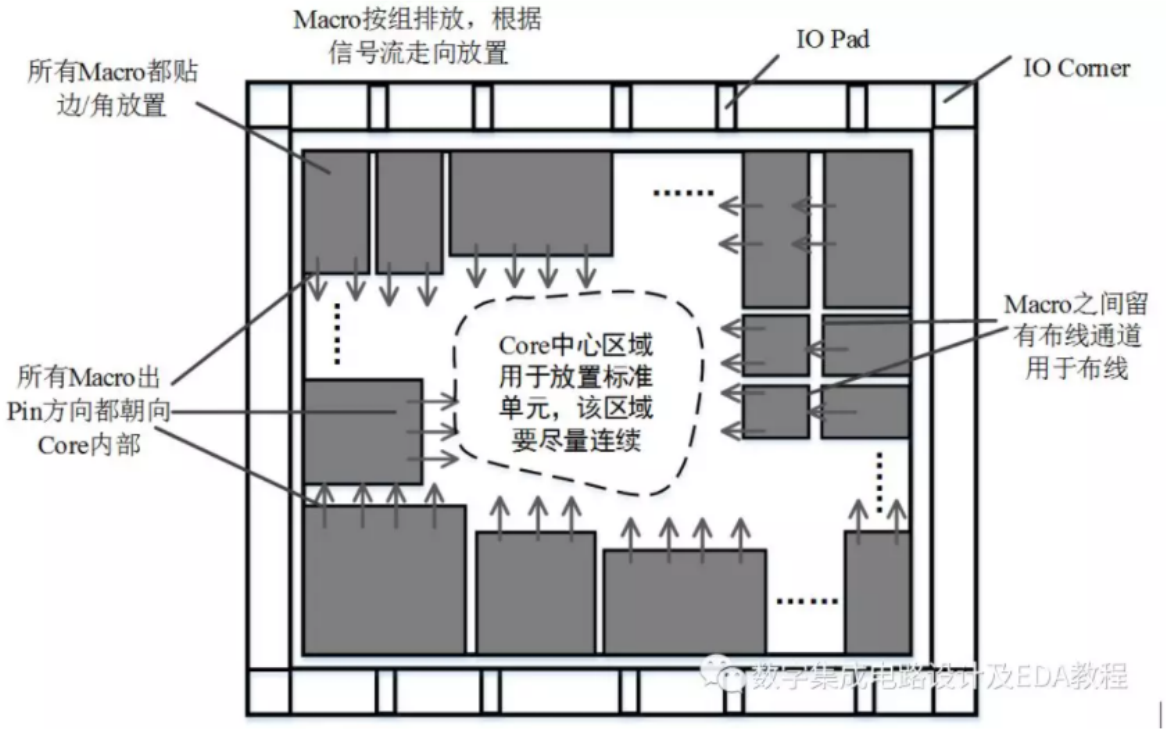


图1 Macro摆放原则

特别是在 Macro 的间隙有端口的时候更是如此，设计者可以通过相邻 Marco 边界上端口的多少来决定留有多大的间隙比较合适，这样才不至于出现 Pin Access 的问题

4）合理设置 Macro 摆放的角度。在考量 Macro 摆放的角度时，不仅仅考虑空间摆放的因素，还要根据端口的连接关系与互连 Macro 的位置来决定。

如前面原理图中存储 Macro 的端口方向朝向中央，因为中间的标准单元需要与存储 Macro 进行数据交换，存在互连关系。在实际设计时，不仅要根据端口与标准单元之间的连接关系，还要考虑 Macro 与 Macro 之间的互连关系进行综合判断。

手工对宏单元进行摆放后，需要将他们设置为 dont_touch 属性，以防止在布局阶段软件对它进行移动。命令为：

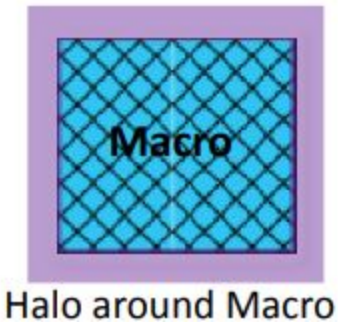
```
set_dont_touch_placement[all_macro_cells]
```

5.4.3 Halo

在使用 EDA 软件的 Floorplan 设计时，同样可以给 Macro 加上 **晕环**（*halo* ）来控制 Macro 与 Macro 之间的距离——设置不允许摆放标准单元晕道的目的是为了能够提供专用的布线空间

注解： 在 ICC 中这被称为 *Keepout Margin*，有 *hard* 、*soft* 之分:

(1)hard 区域不允许任何 Cell 放置在该区域 (2)soft 则在 coarse place 的时候不允许任何 Cell 放入其内，但是在 optimization 以及 legalization 的时候是允许 Cell 放入其内的，也就是只允许 Buffer 加入其中



Keepout Margin 与 Placement Blockage 不同，它并不是独立存在的，而是依附于 Macro 周围，可随 Macro 移动的。所以它是专门用来控制 Macro 和其他单元之间距离的一种功能。

在基于标准单元的设计中，标准单元在布局（place）阶段完成了整个芯片内部的摆放，由于标准单元占据了底层金属的绝大多数布线轨道，当芯片局部出现拥塞时，布线晕道就能够提供更多底层的布线通道

5.5 布局 Block

若采用展平式物理设计方案，则设计中仅仅由 Macro 和标准单元构成；若采用层次化物理设计方案，则经过 Partitioning 后，顶层设计主要由 Macro、sub-system 级别的 Block 构成

在 Floorplan 阶段，对于前者主要是考虑 Macro 的放置，后者则是要考虑 Macro、Block 的放置

注解： 在层次化设计中，对于对于各个预计分割的区域之间也需要进行布线通道的定义。布线通道的宽度需要考虑模块布局的拓扑约束，诸如：

- 子模块到子模块的界限
- 子模块之间的距离、顺序、排列
- 子模块之间的表面比率、网络权重、子模块隔离区

该部分请参考 Placement 部分

5.5.1 布局障碍

blockages : are specific location where placing of cells are blocked, acts like guidelines for placement of std cells.

blockages will not be guiding the tool to place the std cells at some particular area, but it won’t allow the tool to place the std cell in the blocked area

---

注解： 在 Placement、Routing 阶段都存在 Blockage，均有如下分类

1）Hard Blockages

2）Soft Blockages

3）Partial Blockages

分类	特征	目的
Hard Blockage	Complete Standard Cell Blockage	（1）avoid routing congestion at macro corners（2）Restrict std cells to certain regions in the design（3）control power rail generation at macro cells
Soft Blockage	Non-Buffering Blockage	only buffers can be placed and std cells cannot be placed
Partial Blockage	Partial Standard Cell Blockage	（1）used to avoid congestion（2）can Block Standard Cells as per the required percentage value

5.6 电源规划

5.7 时钟规划

Although most ASIC designs use clock tree synthesis, clock tree synthesis may not be sufficient for very high-performance and synchronized designs.

In this case, one needs to implement the distributed clock networks manually in order to minimize the skew between communicating elements due to their line resistance and capacitance.

注解： 注意：Floorplanning 阶段的时钟网络分布与常规的时钟树综合，其依赖于手动布图规划

5.8 Evaluate Floorplanning

在布图规划、布局中需要多次迭代分析，从而实现最佳效果

5.8.1 布局规划阶段的延迟预估

5.8.2 Macro 布放与布线通道的关系

Macro 布放的结果会对前期布图规划的期望目标产生直接影响，其中 **能够保证布线的完成是通过布线通道的分析来进行**

注解： 介绍相关概念：

（1）布线轨道（routing track）：芯片内部专门用于布线的路径

（2）布线通道（routing channel）：每两条 / 多条布线轨道的空间

先前介绍的晕道，实现了预留底层布线通道，解决了当前局部拥塞的问题，但是也会产生其他区域的拥塞问题

5.8.3 Block / Macro 自动放置的算法推进

现代 Soc 设计可以包含数百或更多个 Block，此时就必须借助 EDA 工具来实现 Block 的自动布局

> **警告：** **To do**： fly lines、布局障碍、电源规划、时钟规划、Floorplanning 评估

注解： 在正式开始 Floorplanning 之前，还有两项预备工作，分别是

（1）*Connecting Power and Ground Ports*： create logical connections between power and ground pins on standard cells and macros and the power and ground nets in the design

（2）产生设计中原本不存在但是物理实现必要的单元 *Adding Power, Ground, and Corner Cells*：Physical-only cells for power, ground, and corner placement might not be part of the synthesized netlist and must be added to design.

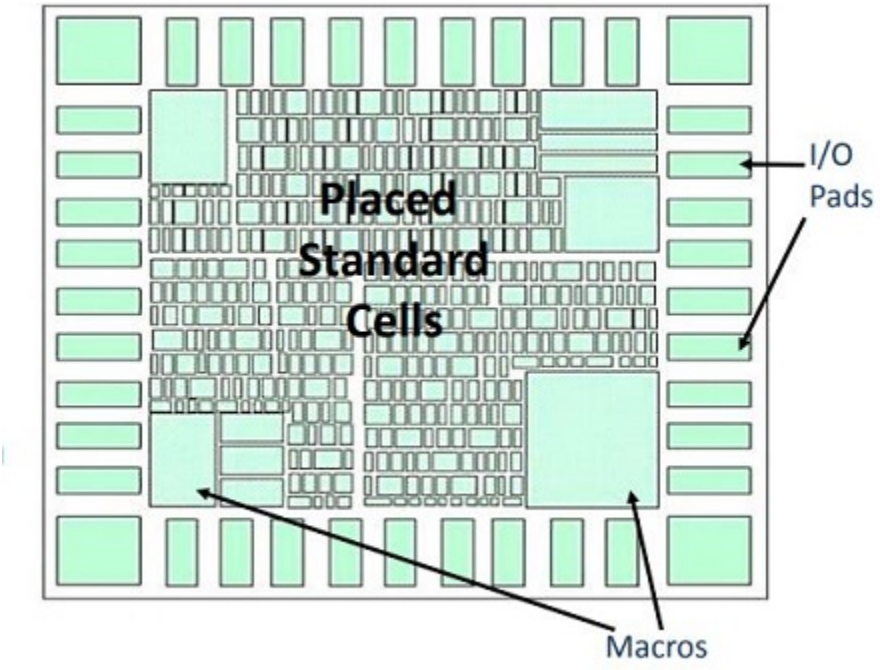
5.9 与布局（Placement）、布线（Routing）的关系

Marco、I/O pad cell 、Block 的布放在结果上属于布局，但是其又在布局规划阶段完成，由于其占据的面积（空间）很大，只有 Marco、I/O pad cell 的布放完成后，电源规划才有意义

CHAPTER 6

Placement

由于 I/O pad cell、macro 的布放都在布图规划时已经完成，对于 **展平式布局**而言，Placement 的剩余任务主要是对标准单元的布局



Placement is the process of placing standard cell in the design, The tool determines the location of each standard cell on the die.

- place the standard cells available in the synthesized netlist
- optimizes the design
- determines the routability of design.

Placement will be driven by different criteria like **timing driven** , **congestion driven** and **power optimization**

6.1 布局过程与算法

6.1.1 Global Placement

6.1.2 Detailed Placement

6.2 布局方案评估

在完成标准单元布局优化完成后，需要对设计进行拥堵分析、静态时序分析、噪声分析、电源分析来评估布局方案的质量

6.2.1 拥塞

查看布局之后的拥塞.. code-block:: console

linenos

$ report_congestion

6.2.2 时序

报告最大路径延时，查看是否存在 Setup Slack

```
$ report_timing
```

报告是否存在时序 DRC 的违反

```
$ report_constraint -all_violators
```

此时设计中最好不要存在 Setup 的违反，可以存在 Max Cap/tran、hold、Min Cap 的违反

6.2.2.1 MCMM

如果设计中有 MCMM，那么可以用下面的命令产生 MCMM 的报告同时产生相应的网页文件，方便查看：

```
$ create_qor_snapshot -name cel_name
```

6.3 层次化布局

CHAPTER 7

CTS

7.1 Terminology

时钟树： 时钟信号在物理设计的实现结果被形象地称为时钟树

根节点： 时钟信号的起点

叶节点： 时钟信号经过一系列分布节点最终到达的寄存器时钟输入端或其他时钟终点

根单元、分布单元、叶单元： 根节点、分布节点、叶节点所依附于的逻辑单元

时钟树综合： 根据芯片时钟网络的设计约束要求，时钟网络从根节点逐级插入驱动器（buffer、inverter）从而到达根节点，进而产生时钟树的过程

7.2 时钟树结构

7.3 时钟树约束文件

根据标准设计约束文件 *sdc* 的要求，产生时钟树约束文件（之后借助于 EDA 工具进行时钟树综合）

7.4 时钟树综合过程及算法

根据时钟树约束文件中对每个时钟定义的参数，时钟树综合工具从标准单元布局开始：

- 计算从时钟根节点到每一个叶节点的延迟
- 插入 buffer / inverter，减小并平衡所有叶节点的延时，使得它们之间的最大差值小于或等于最大插值，同时还要满足其他约束参数条件
- 调用布局工具调整新插入的时钟 buffer / inverter 位置

注解: 布线阶段并没有开始，时钟树综合工具根据时钟树单元布局的位置去估算互连线的总延时——相应的，在布线时优先对时钟树进行，以保证时钟树的最大偏差不致受到影响

7.5 时钟树对电路性能（功能）的影响

从布图规划开始，到电源规划到完成布局之后要提供好的数据环境，供时钟树综合使用

- 物理库、时序库、设计数据、约束参数
- 能够建立时钟树的特殊内容，例如时钟信号和特点、时钟树的特诊和类型、时钟树约束参数

CHAPTER 8

Routing

8.1 Special Routing

8.2 布线过程与算法

8.3 布线方案评估

> **警告：** 按照设计的需求将所有的信号线布通是首要评估指标

衡量布线质量以及后续的优先方向主要以如下几个指标为主

- （消除）布线拥塞 congestion
- （优化）时序（timing）
- （减小）耦合效应（coupling）
- （消除）窜扰（crosstalk）
- （降低）功耗
- （保证）信号完整性（signal integrity）
- （预防）DFM 问题、（提高）良品率

布线的内容是将分布在芯片核内的 Macro、标准单元、I/O pad cell 按照逻辑关系进行互连（百分之百地完成所有逻辑信号的正确互连），且满足设计约束条件