主要内容

- 综合简介
- 综合设计环境
- 静态时序分析基础知识
- 时序约束

综合简介

数字电路ASIC设计流程

- 工程师根据硬件规格(Specification)用硬件描述语言编写RTL级代码
- 功能正确的高层次RTL设计映射为针对具体制造工艺的更低层次的设计
- 布局布线等一系列后端工作

在这个设计流程中，综合是其中重要的一环，它直接决定了前端设计的功能能否最终有效的在物理上实现，也决定了芯片的各种性能指标。

综合工具种类

- 最著名的综合工具是**Synopsys**公司开发的**FPGA Express,FPGA Compiler,Design Compiler**等。
- 一些**FPGA**公司也开发了自己的**HDL** 综合器，例如**Xilinx**的**XST**
- **Synplicity(Synplify,Amplify,Certify**和**Synplify Asic),**
- **Mentor(Leonardo spectrum)**等公司也有自己的产品。

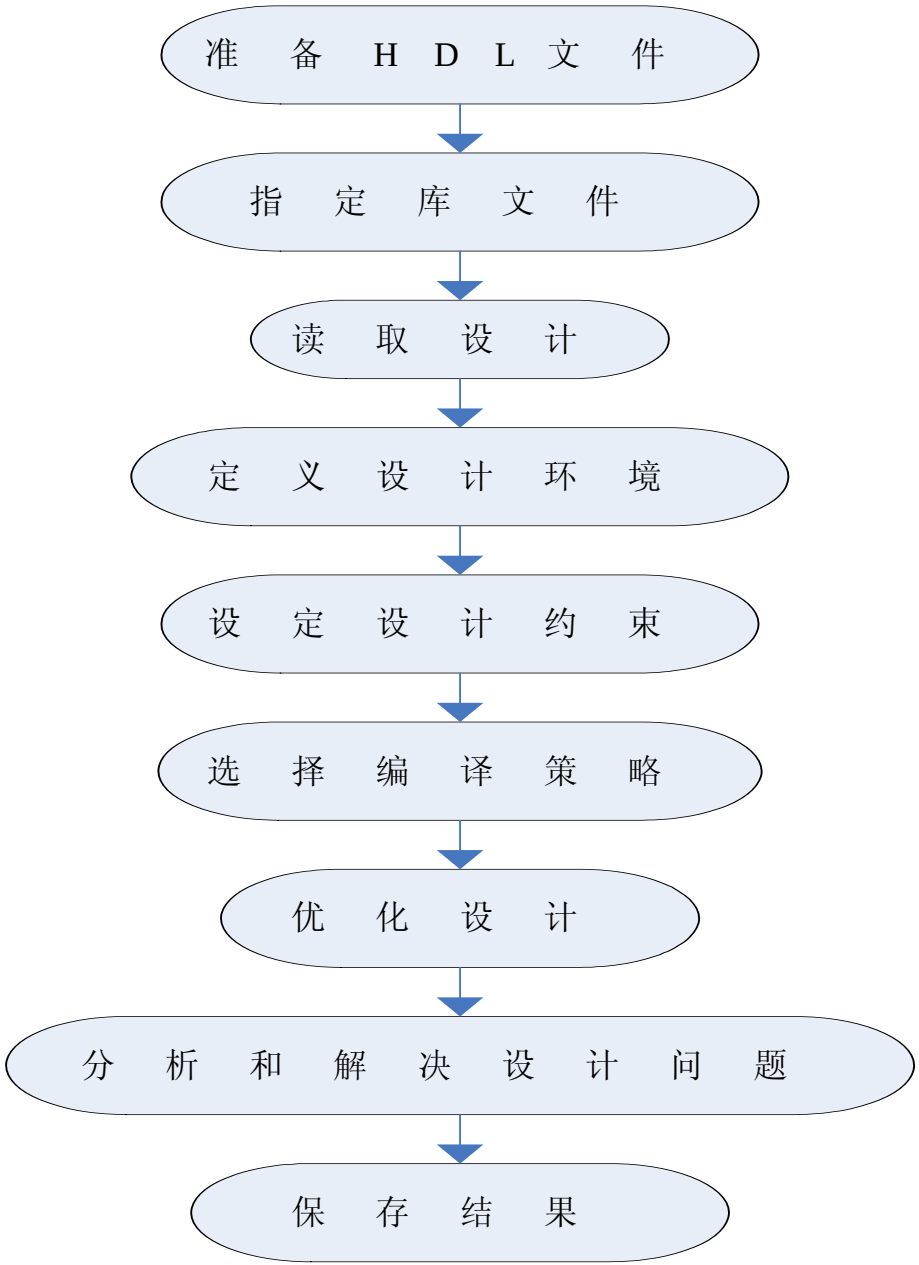
综合技术带来的好处

- 设计师可以采用更高层次的设计方法
- 由于逻辑综合工具的使用，高层次的设计可以很快的转换为门级电路设计
- 逻辑综合技术使与工艺无关的设计成为可能
- 综合工具可以按照约束设置对设计进行自动优化，要得到有不同性能指标的结果，有时候仅仅需要修改综合时的约束设置

对工程师的要求

- 尽管逻辑综合为数字设计带来了显而易见的好处，使设计者再也不用去手工“搭建”自己的产品，但并不等于设计者可以对电路的具体实现毫不关心。
- 为了综合出满足规格定义的产品，工程师在进行代码编写时必须考虑代码的可综合性，良好的代码风格可以得到性能更好的设计
- 逻辑综合本身就是一个复杂的过程，环境和约束的设定、测试和时序问题的分析和解决都需要设计工程师具有专门的知识和技能

具体操作流程


准 备 H D L 文 件
指 定 库 文 件
读 取 设 计
定 义 设 计 环 境
设 定 设 计 约 束
选 择 编 译 策 略
优 化 设 计
分 析 和 解 决 设 计 问 题
保 存 结 果

综合流程

综合包括转译（**Translation**），优化(**Opitimization**)，映射(**Mapping**)三个过程

- 转译把电路的**HDL**描述转化为与工艺无关的功能块组成的逻辑电路的过程。
- 映射是把转译后得到的电路结构用特定目标工艺库中的单元来实现。这时得到的电路包含了具体的制造工艺参数。
- 优化则是综合工具根据设计者施加的时序和面积等约束条件对电路进行改进的过程

DC的主要功能

- 使用用户指定的门阵列或者标准单元库产生速度快、面积小的**ASIC**设计
- 把设计从一种工艺转换为另一种工艺
- 在不同的负载、温度和电压条件下对时序、面积和功耗等设计指标作折中处理
- 综合和优化有限状态机**(FSM)**
- 支持网表输入，支持用来与第三方工具交互的网表或原理图输出
- 自动创建和划分层次化原理图

DC中的工艺库及其配置

- **GTECH库** **GTECH库是Synopsys的通用工艺库。它由DC自带，是独立于厂家工艺的。该库中包含的元件仅代表一定的逻辑功能而不带有任何工艺参数。DC在转译时会先将HDL描述转化为GTECH库单元组成的电路。**(translation)

- 目标工艺库**(target library)** **DC在产生门级网表时必须使用目标工艺库。目标工艺库通常由芯片制造商提供，包含各种单元的物理信息。在映射过程中，DC会自动从目标工艺库中选择合适的标准单元。** **(map)**

工艺库的格式
(以中芯国际的工艺库为例)

- **DC**用到的工艺库是**.db**或者是**.lib**格式的，其中**.lib**格式的文件是可读得，通过此文件可以了解库的详细信息，比如说工作电压，操作温度，工艺偏差等等。**.db**格式的库是二进制的，不可读。**.db**格式的库由**.lib**格式的库通过命令**read_lib**生成

工艺库的格式
(以中芯国际的工艺库为例)

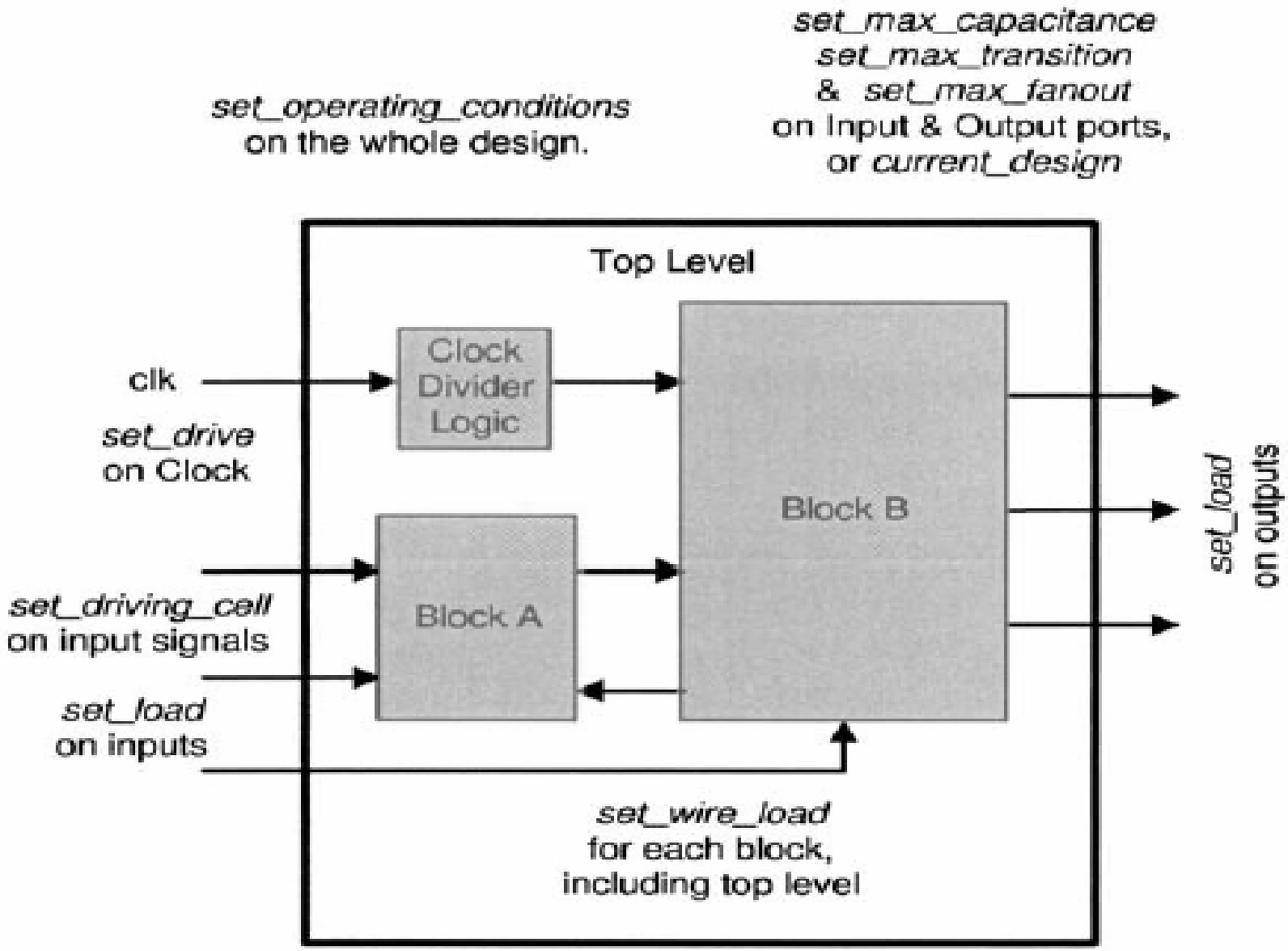
有以下六个文件:

- **smic35os142_max.db**
- **smic35os142_typ.db**
- **smic35os142_min.db**

- **smic35os142_max.lib**
- **smic35os142_typ.lib**
- **smic35os142_min.lib**

综合设计环境

设计环境


set_max_capacitance
set_max_transition
& set_max_fanout
on Input & Output ports,
or current_design
set_operating_conditions
on the whole design.
Top Level
clk
set_drive
on Clock
Clock
Divider
Logic
Block B
Block A
set_driving_cell
on input signals
set_load
on inputs
set_load
on outputs
set_wire_load
for each block,
including top level

操作环境

- 库的开头是一些基本信息，如版本号，各种单位（电压单位，电流单位，电阻单位，电容单位等等）和参数.
- 操作环境：芯片供应商提供的库通常有**max,type,min**三种类型，代表操作环境为最坏**(worst)**，典型**(type)**，最好**(best)**三种情况。芯片的操作环境包括：操作温度，供电电压，制造工艺偏差和**RC**树模型。在手工设置时,应使用set_operating_conditions命令

```
dc_shell-t>set_operating_conditions  WCCOM
  -lib my_lib
```

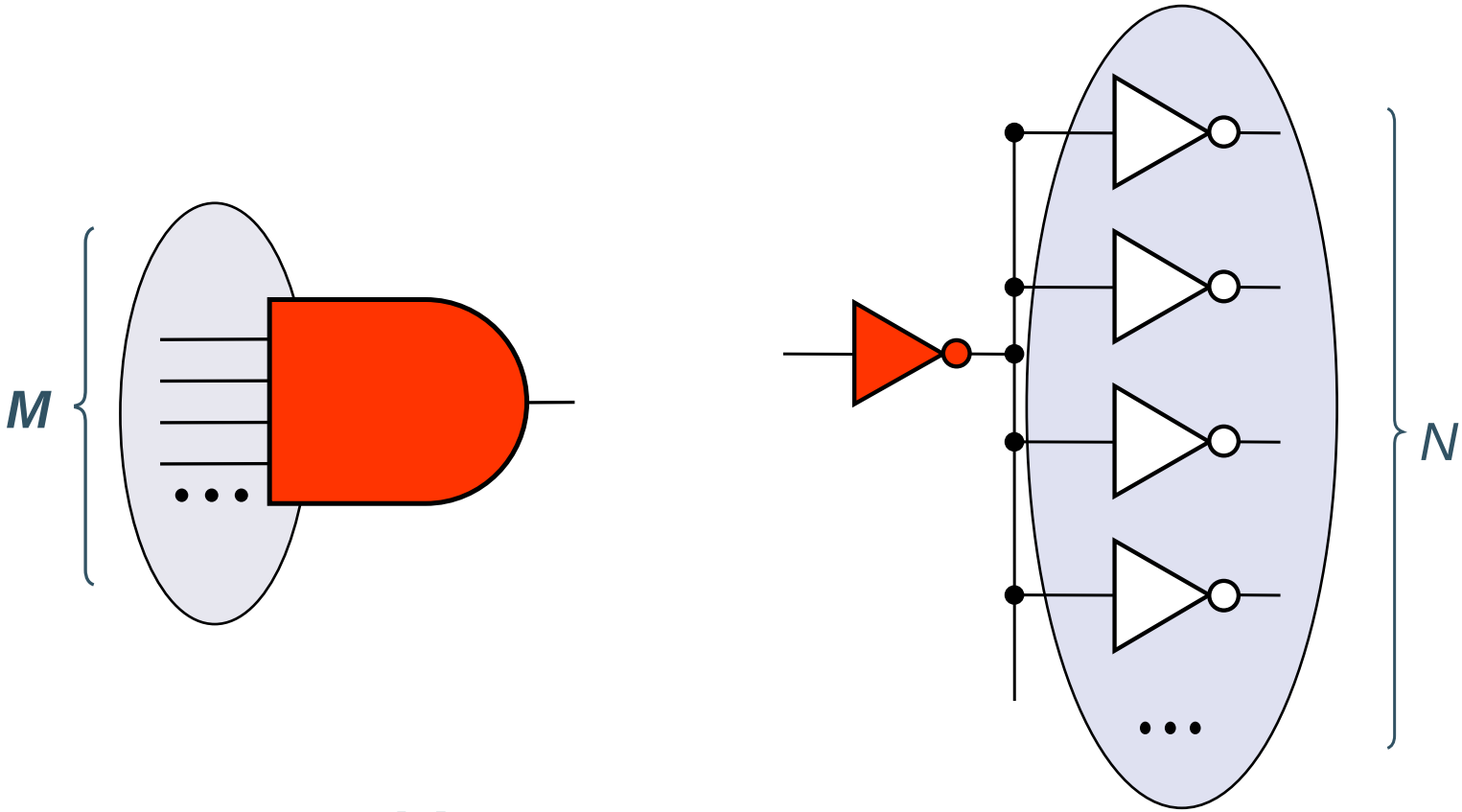
线负载模型

- 在实际电路中，导线都具有一定的电阻和电容，这样信号在导线上传播时均有一定的延时。在深亚微米设计中，导线互连延时对信号造成的影响是不能忽略的。线负载模型可以用来估算电路中导线长度和扇出数对导线电阻、电容和面积的影响。
- 在层次化的设计中，**DC**用顶层**(top)**、包围**(enclosed)**和分段**(segmented)**三种模式来确定跨越层次边界的导线的线负载模型。
- 要手动设置线负载模型和线负载模式，可以使用**set_wire_load_model**和**set_wire_load_mode**命令。

系统接口特性

- 输入端口的驱动特性
- 输入和输出端口的负载
- 输出端口的扇出负载

Fan-in and Fan-out



Fan-in *M*

Fan-out *N*

设定输入端口的驱动

- 驱动能力用驱动电阻值表示，阻值越小驱动能力就越强。如果输入端口采用缺省驱动设置0，则表示有无限大的驱动能力。如果取值不当，综合出来的电路就不能正常工作。如果驱动太大，综合出来的电路的负载很大；驱动太小，信号的变化边沿会很差 DC中有**set_drive**和**set_driving_cell**两条命令来设定端口的驱动。

设定输入和输出端口负载

- 利用端口负载，DC可以为输出端口选择适当大小的驱动能力，也可以用来计算输入端口的延时。如果负载取得过小，下级电路无法正常工作，负载取得过大，会增大上一电路的难度. 在缺省情况下，**DC**假定输入输出端口的容性负载为**0**。我们可以用**set_load**命令设定输入、输出端口的容性负载值。

设定输出端口的扇出负载

- 输出端口外部的扇出负载总和。扇出负载不同于负载，它是一个无量纲的数值。负载则是指电容值的大小 。在**DC**中可以用**set_fanout_load**命令来设定输出端口的扇出负载

综合工艺库单元描述

- 库里面对每个单元都有描述，比如该单元的功能，时间，面积等等。除此之外，对于单元的每个引脚还提供有设计规则约束（**DRC**），这些值由设计工艺决定，只有满足这些条件的单元在运行时才会操作正常，因此在设计中有很高的权重。包括：
- **fanout_load:**输入引脚的扇出负载，这个值跟**max_fanout**有关。
- **Max_fanout:**用于定义输出引脚的值。
- **Max_transition:**用于定义输入引脚的最大转换时间；
- **Max_capacitance:**用于定义输出引脚的最大电容。

综合工艺库单元描述
(中芯国际2输入or门为例)

```
.....................
pin(A2) {
        direction : input;
        capacitance : 0.006;
        fanout_load : 1;
}
internal_power(power_outputs_0)
 {  /* average switching power [in pJ] for pin 'Z' */
    related_outputs : "Z";
    related_inputs  : "A1 A2";
     values( " 0.181, 0.176, 0.174, 0.173, 0.173",\
        " 0.201, 0.196, 0.191, 0.189, 0.188",\
        " 0.267, 0.258, 0.250, 0.246, 0.243",\
        " 0.412, 0.398, 0.384, 0.375, 0.367",\
        " 0.559, 0.543, 0.526, 0.511, 0.500");
 }
```

```
.................
pin(Z) {
    direction : output;
    max_capacitance : 0.150;
    max_fanout : 15;
    max_transition : 3.0;
     function  : "A2+A1";
.................
.........
```

设计规则约束

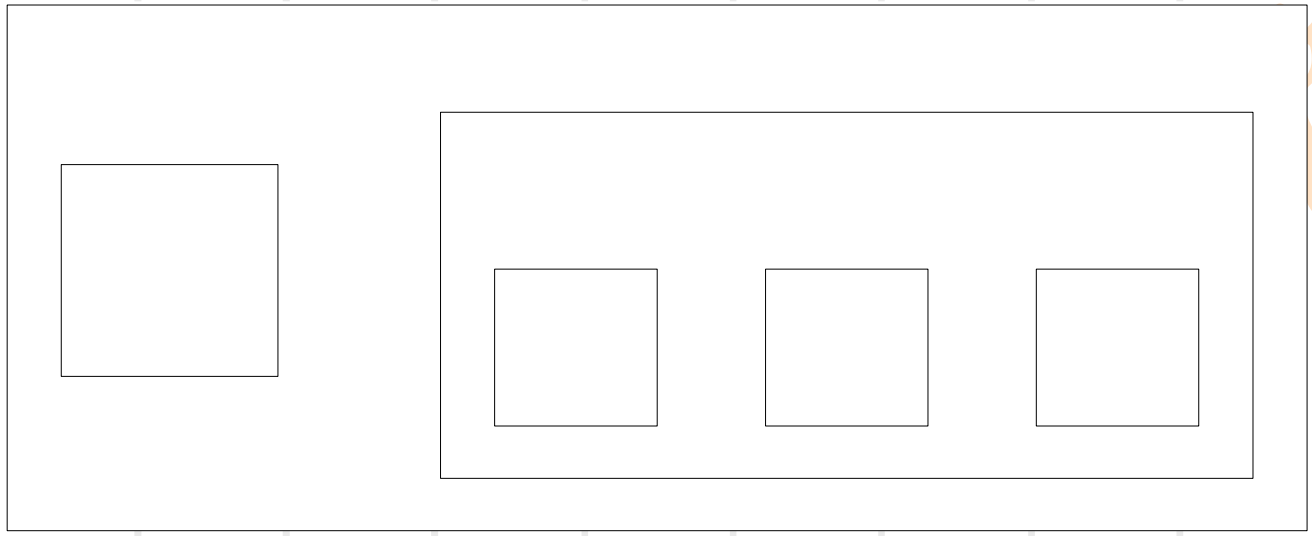
- 在实际的综合中，设计工程师除了要指定设计环境，还应该指定设计约束。
- 设计规则是由芯片制造商提供的关于其工艺的一套规则，反映了该工艺对设计的要求。设计规则通常已经包含在了工艺库中，一般不需要另外定义。如果需要更严格的设计规则约束，可以自己手工定义。包括以下内容:

  ① 转换时间(Transition time)；② 扇出(Fanout load)；③ 电容(Capacitance)；④ 单元退化(Cell degradation)；⑤ 互连类(Connection class)。

综合策略

- 在对层次化设计进行综合时，综合策略的选择对综合速度、综合结果影响极大。层次化设计的基本综合优化策略有两种：自顶向下**(Top-down)**的综合策略和自底向上**(Bottom-up)**的综合策略。

自顶向下策略



- 顶层设计和其所有底层设计会在同一条件下同时综合，一般适合规模比较小的设计

步骤：

- 读入设计；
- 解决多次实例化问题；
- 将约束文件施加到顶层设计；
- 综合。

SUB1

自底向上策略

- 自底向上的策略一般用于中、大规模的设计

其主要步骤如下：

①准备一个全局约束文件和各子模块的约束文件；

②单独综合各个子模块；

③读取顶层设计和未存在于内存中的已综合子模块；

④设置顶层设计为当前设计，链接，施加顶层约束。如果此时设计已满足约束，则综合完成，若不满足，还要继续下面的步骤；

⑤对实例化单元应用**characterize**命令；

⑥使用**write_script**命令生成这些单元的新约束文件；

⑦使用**remove_design –all**命令从内存中清除所有设计，然后读入前面**characterize**后的模块；

⑧设定**characterize**的子模块为**current_design**，并使用步骤⑥中产生的约束文件重新综合；

⑨读入所有综合后的子设计，并链接当前子设计；

⑩选择另一个子模块，重复上面的③到⑧步，直到所有子模块全部综合完毕。

两种策略的比较

- 自底向上的综合策略也有很多优点。对于较大的设计，自底向上的策略可以划分设计且“各个击破”，可以根据设计各个部分的特点和要求单独优化。自底向上还可以节省计算机内存。同时，在设计中的某个部分修改后不需要重新综合整个设计，节省了修改综合的时间。但自底向上的策略也有操作复杂、要手工进行版本控制等等缺点。
- 从设计整体风格、时序规划的角度来说，自顶向下的策略更好一些。但如果设计的规模很大，自底向上的策略也有自己的优势。
- 当然，也可以结合两种策略的优点采用混合综合策略。

静态时序分析
基础知识****

延时的产生

- 1,门延时
- 2,线延时

为什么要进行静态时序分析？

- 说说静态、动态时序模拟的优缺点。（威盛VIA ）
- 传统上是采用动态仿真来验证一个设计的功能和时序。随着设计规模的增大，验证一个设计所需要的测试向量的数量以指数增长，且这种方法难以保证足够的覆盖率。在大型设计中，如果仅用传统的动态仿真的方法，则时间及工作量都难以承受。

为什么要进行静态时序分析？Cont.

- 静态时序分析可以降低验证的复杂性。静态时序分析提供了一种针对大规模设计验证的有效解决方法。它可以检查电路中所有时序路径的时序，测试覆盖率可以达到**100%**。**STA**的方法不需要任何测试向量，分析所需要的时间远远少于门级动态仿真。但静态时序与动态仿真相比，也有自身的缺点。

为什么要进行静态时序分析？Cont.

- **STA不能验证设计的功能**，设计功能验证还必须使用功能仿真来实现。另外，静态时序分析只能验证同步时序电路的时序特性，如果设计中含有较多的异步电路，则应该通过门级动态仿真来验证。一般来说，一个设计的验证应该既包含**RTL**级的功能仿真，也包括静态时序分析和门级动态仿真。静态时序分析和门级动态仿真各有优点，互相补充，一起使用可以有效保证电路的正确性和可靠性。

如何进行静态时序分析？

- 使用静态时序分析的方法分析电路时序时，分析工具会首先将设计分解为很多不同时序路径的集合，然后计算每条时序路径的延时信息，最后检查路径延时，分析其是否满足时序约束。

如何进行静态时序分析？Cont.

- PrimeTime是Synopsys公司的一种静态时序分析工具。
- 现在，静态时序分析已经是ASIC设计流程中最重要的一个环节，它能验证设计在时序上的正确性，并决定设计是否能在要求的频率下运行。
- PrimeTime能够分析设计中时序路径的延时，找出时序冲突，提供分析结果供设计工程师修改设计。PrimeTime适合对大规模的同步数字设计进行分析，而且与综合工具DC有很好的接口，可以在整个设计流程中使用。

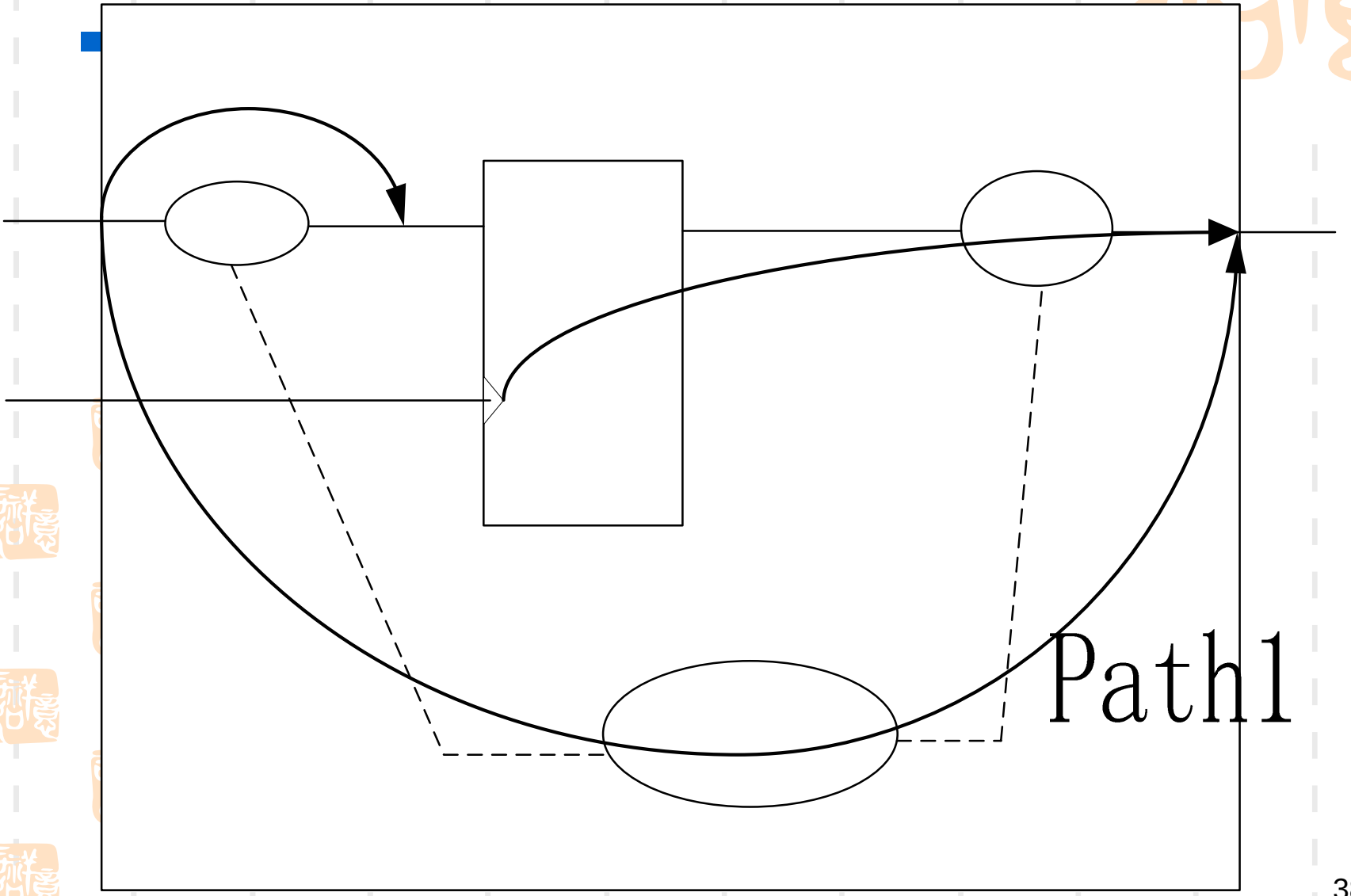
!!!!时序路径(timing path)

- 时序路径的起点只能是内部时序单元的时钟端或设计的输入端口；时序路径的终点只能是内部时序单元的数据输入端或设计的输出端口。电路中的时序路径一般有四种：
- ①从输入端口到触发器；
- ②从触发器到触发器；
- ③从触发器到输出端口；
- ④从输入端口到输出端口。

时序路径(timing path) cont.

- 静态时序分析会分析以上四种时序路径，保证信号在规定的时间传送到各个路径的终点，并在电路正常工作所需要的时间内保持稳定。
- 时间通路（**timing path**）:逻辑信号传递的路线，其起点是所有的基本输入端（**Primary Input**）,和所有时序单元的时钟输入端，终点是所有的基本输出端（**Primary Output**）和所有时序单元的数据输入端。
- **DC**会自动地根据时钟信号分组，划分时间通路，进行时序检查。

时序路径(timing path) cont.



时序路径cont.

- 在每一条时间通路上，DC根据约束设置，调用内部的算法，计算出所有单元和线的延时.

几个重要概念

- 要求到达时间(Required Arrival Time)

  ：指期望信号到达电路中某一点的时间。

- 信号到达时间(Arrival Time)

  ：是指信号到达电路中某一点的真实时间，一般等于信号到达时序路径起点的时间加上信号在该时序路径上传播所用的时间。

- !!!!时序裕度(slack)

  ：指电路中某点处要求到达时间与实际信号到达时间的差值。时序裕度也是作时序检查时我们需要仔细分析的对象。

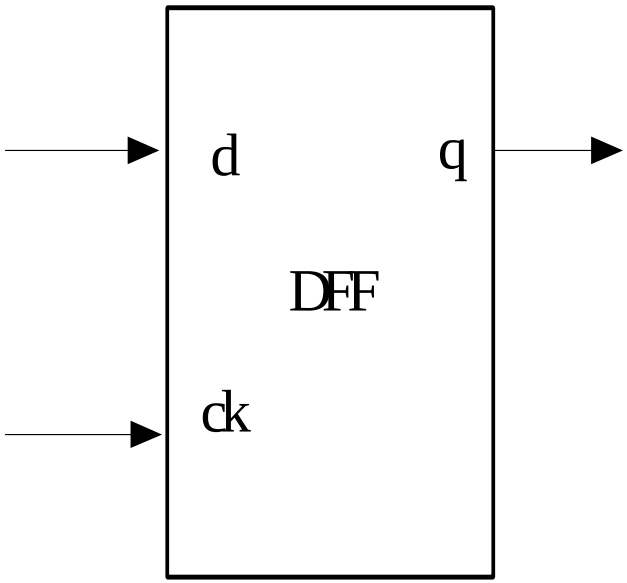
静态时序分析检查内容

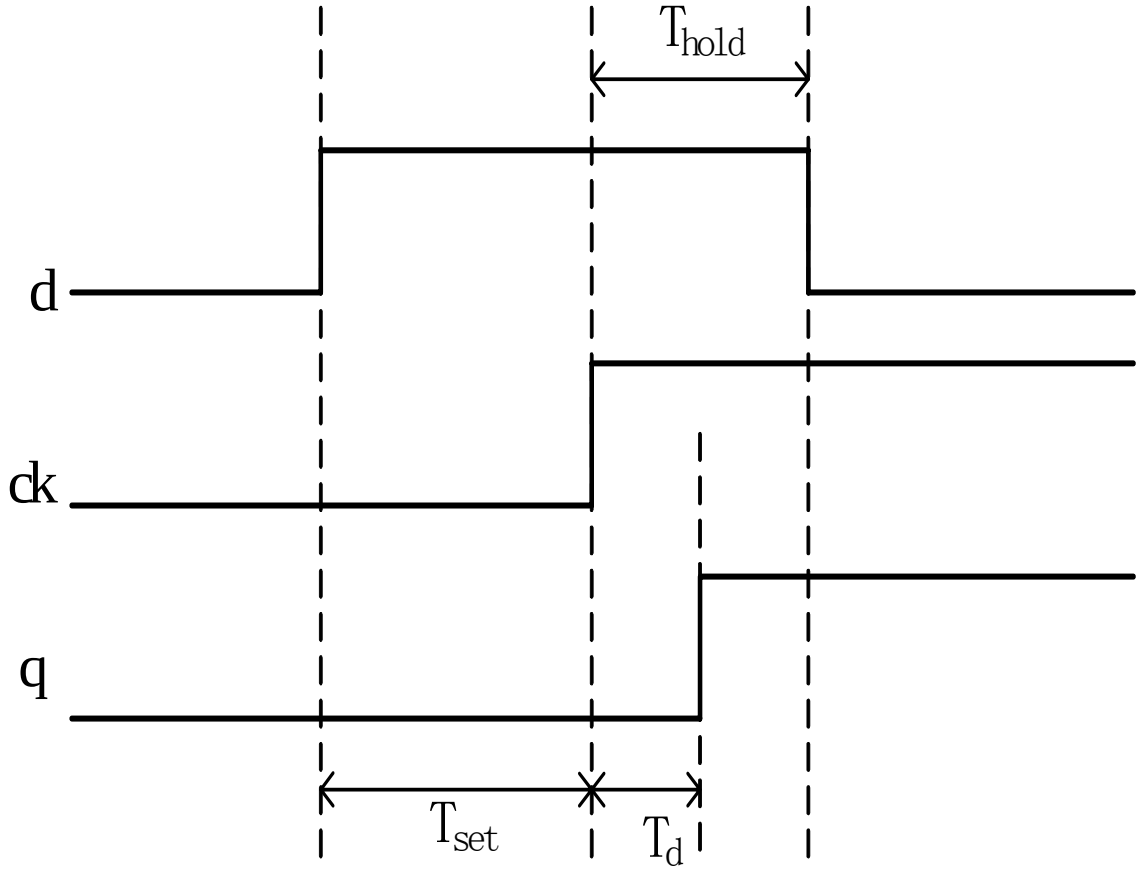
- 主要包括建立时间和保持时间、门控时钟、恢复(Recovery)和(Removal)时间以及时钟脉冲宽度等等。

建立时间和保持时间

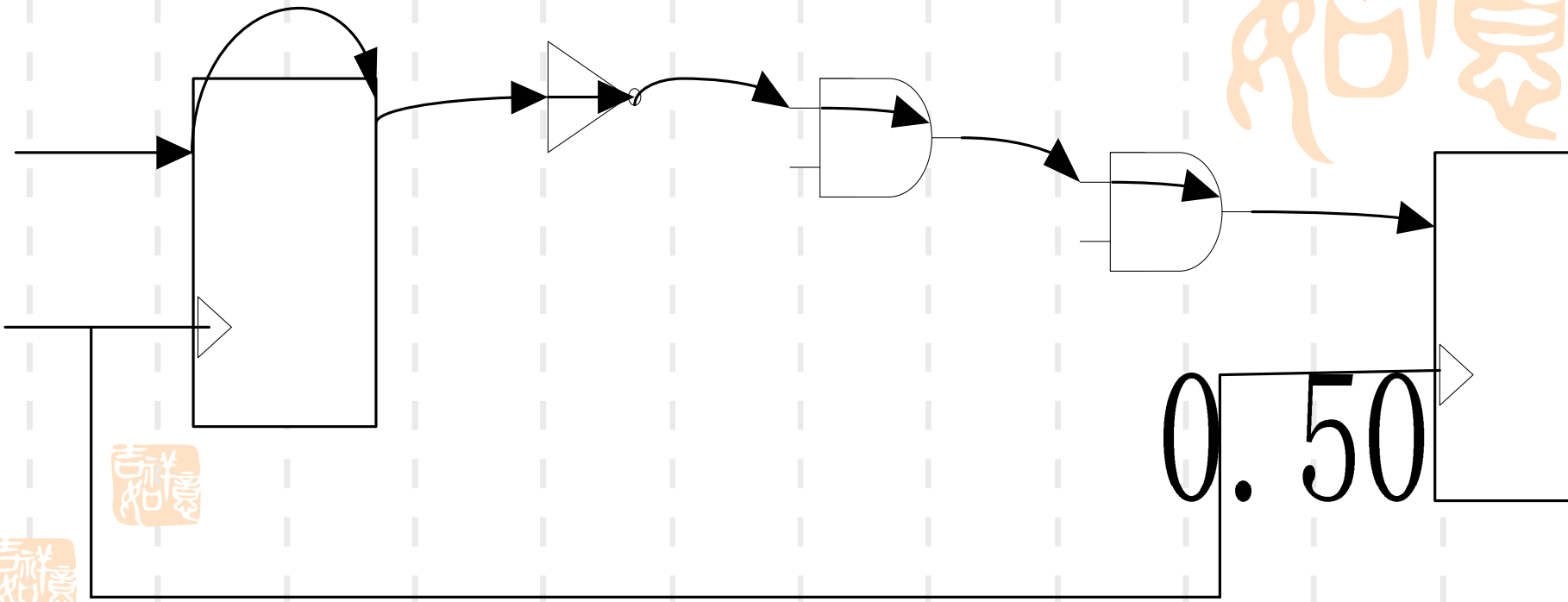
- 建立时间（**setup time**）:数据在时钟信号源到达之前必须要稳定的时间，如果建立时间不满足，数据不能正确打进时序逻辑单元。
- 保持时间（**hold time**）:数据在时钟信号源到达之后必须要稳定的时间，如果保持时间不够，数据被时序逻辑单元正确锁存 。
- 基本单元的延时（**Td**）和线延时 ： 门延时是指信号通过实际的标准单元所需要的时间. 在时序逻辑单元中，反映为从时钟沿开始，到数据输出需要的时间, 线延时是指由于导线的阻容而导致的信号传播延时 。

示意图





图例



- 如：从触发器**DFF1**的**Q**输出端到触发器**DFF2**的数据输入端的路径延时

path_delay=(0.50+1.0+0.54+0.32+0.66+0.23+0.43+0.25)=3.93ns

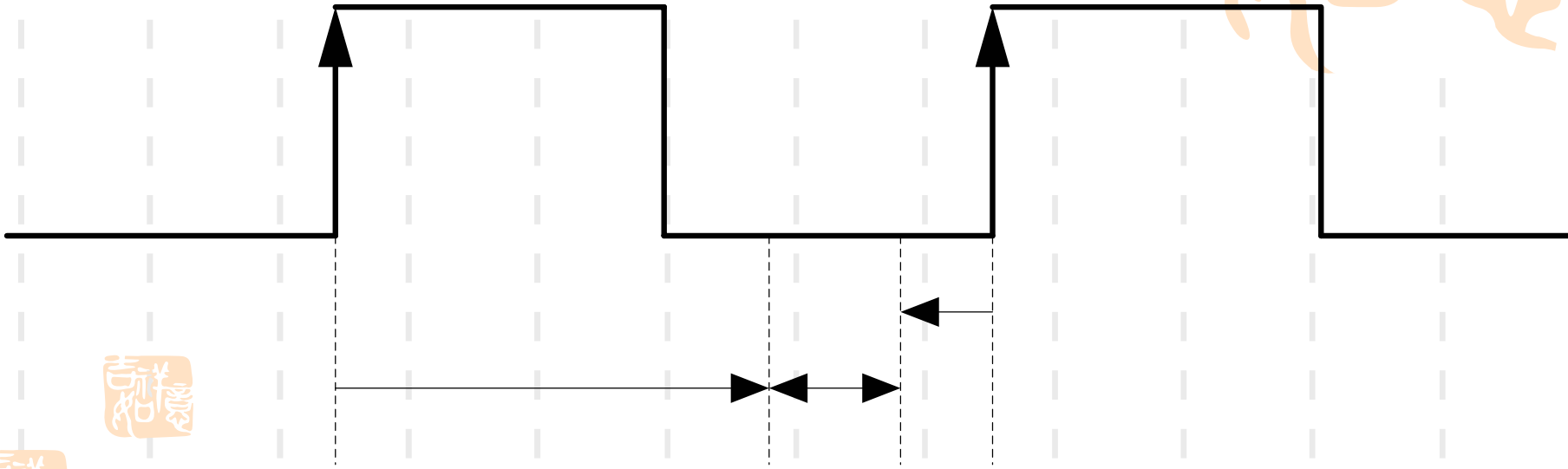
图例 cont.

- 假设在0时刻Data由触发器DFF1的时钟沿触发，经过如图所示的路径在时刻3.93ns到达触发器DFF2的数据端，这个时间就是信号到达时间（Data Arrival Time）.
- 在下一个时钟周期（假设时钟周期为5ns）触发器DFF2的时钟沿到来的时刻，数据打入触发器DFF2.

图例 cont.

- 但是DFF2触发器的数据必须在时钟上升沿到达之前稳定一段时间，即是setup time。假设这个建立时间是0.5ns的话，也就是说数据必须在第二个周期的前0.5ns到达，即数据必须在5－0.5=4.5ns之前到达，因此4.5秒是要求到达时间，如果超过了这个时刻数据就打不进触发器DFF2 .

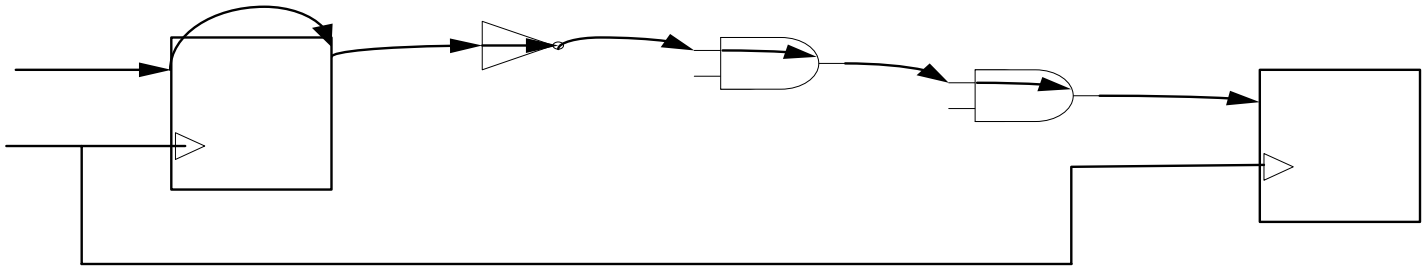
图例 cont.



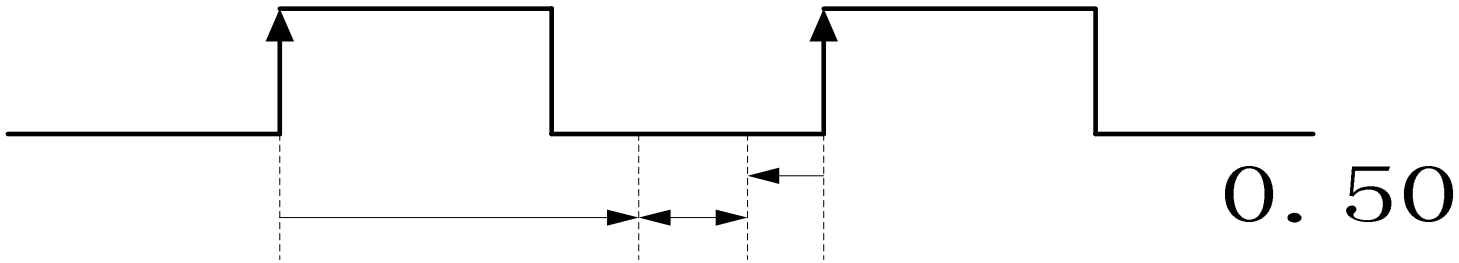
图例 cont.

- 此时数据实际到达的时刻比需要到达的时刻早**4.5-3.93ns,**即时序裕度**slack=4.5-3.93=+0.57ns.**当**slack**为正时，数据就能正确地被触发器采集，反之，当**slack**为负，即数据实际到达的时刻比需要达到的时刻晚，数据不能正确地被采集到，这种情况就叫做建立时间违背（**setup time volations**）.
- 建立时间与保持时间通过对最大路径延时和最小路径延时的分析得到的。

总结：建立时间违背

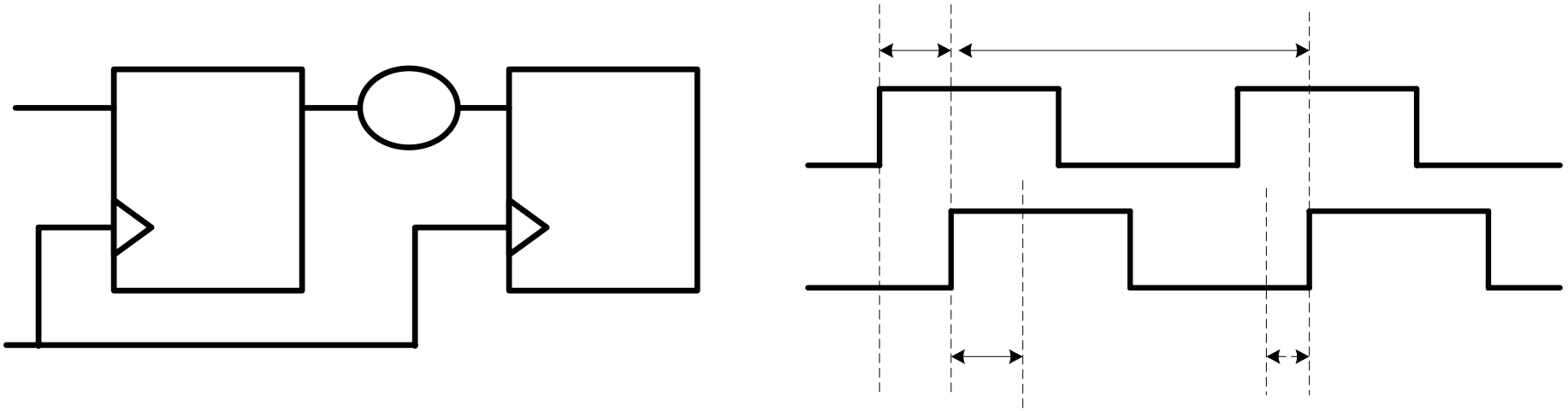


- path_delay=(0.50+1.0+0.54+0.32+0.66+0.23+0.43+0.25)=3.93ns



- 在下一个时钟周期（假设时钟周期为**5ns**）触发器**DFF2**的时钟沿到来的时刻，数据打入触发器**DFF2.**但是**DFF2**触发器的数据必须在时钟上升沿到达之前稳定一段时间，即是**setup time**。假设这个建立时间Ts是**0.5s**的话，也就是说数据必须在第二个周期的前**0.5ns**到达，即数据必须在**5**－**0.5=4.5ns**之前到达，否则数据就打不进触发器**DFF2**。这种情况就是建立时间违背 .

保持时间违背



- 考虑到保持时间的要求，0时刻D1的数据信号经过FF1和组合逻辑到达FF2的数据端D2的时间为T1，T1应比0时刻D2的数据信号到达FF2 的时间晚Th ,即T1>=Tskew+Th,0时刻D2的数据才能被正确打进触发器，否则就会发生保持时间违背。由此可知，保持时间违背也就是数据相对于时钟沿到达的太早了。

FF1

总结:时序约束

- 建立时间与保持时间通过对最大路径延时和最小路径延时的分析得到的。

{Max,Min}Path_Delay+{Max,Min}T_{setup}+{Max,Min}T_{hold}<Tperiod

而芯片的工作时钟频率=1/ T_period

T_period越大,频率就越小,性能就越低.

- **DC的所有时序约束都是针对同步设计而言的，因此对于同步电路DC是比较成熟和完善的，但是对于异步电路，DC目前还没有办法考虑时序。**

回复时间(Recovery)和移除(Removal)

- 恢复(Recovery)和移除(Removal)一般是针对触发器的异步复位/置位端的检查
- 异步复位信号无效后到下一个时钟有效沿之前的最小时间叫做回复时间(Recovery)。
- 而时钟有效沿之后到异步复位信号无效之前的最小时间叫移除时间(Removal)。
- 这两个概念与建立和保持时间类似。通常我们都应该使时钟有效沿与异步复位信号的结束位置尽量远。如果过于紧密，就会出现Recovery和Removal冲突，这时触发器输出可能会出现不定态

PrimeTime工具的主要特点

- ①适用于同步数字电路；
- ②与DC共用相同的延时计算方案；
- ③支持图形和Shell命令行两种界面；
- ④自动排除伪路径，能找到所有的时序冲突；
- ⑤支持SDF文件反标。

关键路径

从延迟的角度来说，关键路径就是指那些总延迟大于相应周期时间的路径。消减关键路径的延迟要从消减路径中的各部分延迟入手，主要方法就是利用综合工具对路径施加约束条件来限制优化，达到减小路径延迟的目的。

对关键路径延迟的主要约束处理方法

通过选择器件的处理方法

从最直观的角度看，时序逻辑和组合逻辑都由基本的电路单元组成，因此，选择延迟小且不影响芯片性能的器件是既简易又高效的处理方法。例如，基本电路单元库中的**DFFXL**寄存器虽然面积较小，但它的延迟相关参数**Tck-q**、**Tsetup**较大，容易形成关键路径，于是可以通过设置**set_dont_use**等约束来禁用它。在一些特殊情况下，基本电路单元库中的器件不能满足要求，这时需要采用自定义的电路单元。

对关键路径延迟的主要约束处理方法 (cont.)

对端口间逻辑的处理方法

这是诸方法中最常用、最有效、最重要的，一般通过**set_input_delay**、**set_output_delay**、**set_max_delay**等来实现，有以下几种情况：

对关键路径延迟的主要约束处理方法(cont.)

对端口间逻辑的处理方法

如果两个寄存器之间的逻辑比较少，那么可以对其输入延迟和输出延迟施加较宽裕的约束，即设置较大的**set_input_delay**和**set_output_delay**值，表明所做逻辑不受压缩，映射电路基本单元库的自由度较大。这样，两者的实际延迟之和将不大于单周期时间(非关键路径)，不仅满足设计要求，而且对其他关键路径的影响很小。

对关键路径延迟的主要约束处理方法(cont.)

对端口间逻辑的处理方法

如果两个寄存器之间的逻辑比较多，那么就要对其输入延迟和输出延迟施加较严厉的约束，即设置较小的**set_input_delay**和**set_output_delay**值，表明所做逻辑需要压缩，而映射电路基本单元库的自由度也较小。但这并不表示越小越好，如果设的值很小**(**甚至为零**)**，那么会使综合器对这条路径的逻辑压缩得过大，而导致其它关键路径的延迟增加，甚至导致其它非关键路径转化为关键路径。因此要凭借经验，不断改变所设的约束值，最终使所有路径的延迟都不大于单周期时间，满足设计要求。

对关键路径延迟的主要约束处理方法(cont.)

对端口间逻辑的处理方法

对于一般芯片设计(中小规模)，在以上两种情况下，对其输入延迟和输出延迟合理施加约束，基本就能满足设计要求。如有个别几条关键路径延迟仍然较长，可以通过设置**set_critical_range**和**group_path**来加以约束。这两种约束对所约束路径的逻辑压缩效果较好，且不会影响其它路径的延迟。采用这样的约束之后，关键路径通常都能被消除了。

对关键路径延迟的主要约束处理方法(cont.)

对端口间逻辑的处理方法

对于一些大规模的芯片设计和上述处理后仍然存在关键路径的情况，就要用**set_max_delay**来进行约束，这种约束的效果非常明显，但会影响其它路径的延迟。因此也要凭借经验，不断改变所设的约束值，最终使所有路径的延迟都能满足设计要求。

对关键路径延迟的主要约束处理方法(cont.)

对层次间边界的处理方法

硬件描述语言描述的**RTL**级电路通常是多层次模块，对其进行综合后得到的电路依然以独立模块的方式存在，即存在边界问题，因此综合中有专门针对边界问题的约束，利用这些约束可以打散边界、保持边界，或重新整合边界，从而优化边界，达到设计要求。

约束**group**用来生成新的层次模块，而约束**ungroup**的作用刚好相反，它用来打散边界，消除层次模块。通常它们都会结合起来使用，但不管以哪种方式选择边界，都应该根据具体的设计要求，参照综合结果，选择最好的方法。

对关键路径延迟的主要约束处理方法 (cont.)

对层次间边界的处理方法

在用硬件描述语言描述**RTL**级电路时，有时会专门设计某些单独模块(类似全定制电路)来简化实现一定的功能，从而缩短延迟并减小面积。在综合中利用约束**set_dont_touch**可以保证这类模块不受影响，保持边界.

对关键路径延迟的主要约束处理方法(cont.)

对电路结构的处理方法

一般情况下，设计者在描述**RTL**级电路时就应该考虑诸多因素，对电路结构进行规划，而在综合中只要将**RTL**代码映射到基本电路单元库几乎就能达到预期要求。但在有些状况下，还是需要利用一些约束来进行优化，这里仅以最常见的**set_structure**和**set_flatten**来加以说明。

对关键路径延迟的主要约束处理方法 (cont.)

对电路结构的处理方法

structuring是综合中默认的逻辑优化策略，它同时考虑了延迟(速度)优化及面积优化；而**flattening**这种策略往往以牺牲面积来达到缩短延迟的目的。

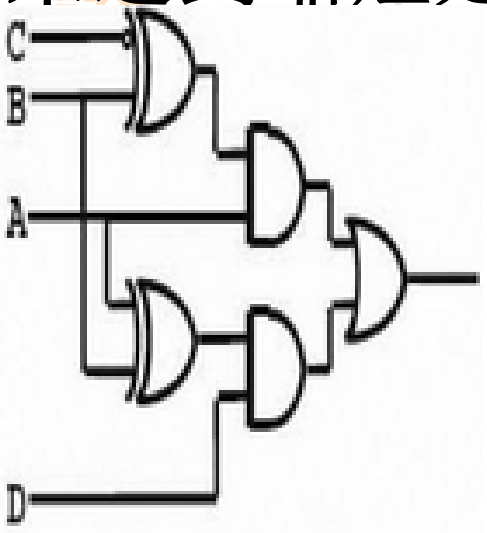


图**3 structuring**策略下某一逻辑的综合电路

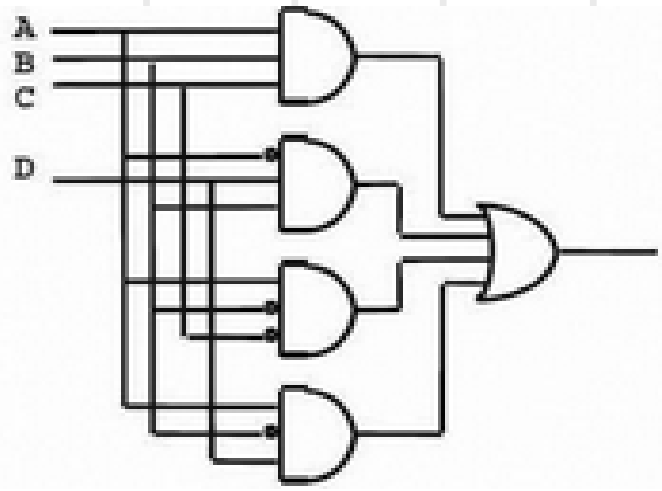


图**4 flattening**策略下同一逻辑的综合电路

对关键路径延迟的主要约束处理方法(cont.)

对电路结构的处理方法

对某一逻辑的**RTL**级描述采用**structuring**和**flattening**两种策略得到的综合电路分别如图**3**、图**4**所示。**flattening**策略下得到的电路只有两级，延迟小于**structuring**策略下得到的三级电路，但电路面积比较大。当今**IC**工艺已经进入深亚微米级，因而在设计中往往需要首先考虑延迟因素，但究竟选择哪种策略，还是要根据具体的设计要求而决定。

常见公司数字IC设计招聘一题目

1,什么是Setup 和Holdup时间？（汉王）

2,setup和holdup时间,区别.（南山之桥）

3,解释setup time和hold time的定义和在时钟信号延迟时的变化。

4,解释setup和hold time violation，画图说明，并说明解决办法。（威盛VIA ）

常见公司数字IC设计招聘一题目

:**Setup/hold time** 是测试芯片对输入信号和时钟信号之间的时间要求。

建立时间是指触发器的时钟信号上升沿到来以前，数据稳定不变的时间。输入信号应提前时钟上升沿（如上升沿有效）**T**时间到达芯片，这个**T**就是建立时间**Setup time.**如不满足**setup time,**这个数据就不能被这一时钟打入触发器，只有在下一个时钟上升沿，数据才能被打入触发器。

保持时间是指触发器的时钟信号上升沿到来以后，数据稳定不变的时间。如果**hold time** 不够，数据同样不能被打入触发器。

常见公司数字IC设计招聘—题目

1,给了reg的setup,hold时间，求中间组合逻辑的delay范围。（飞利浦—大唐）

:$T_{Delay} < T_{period} - T_{setup} – T_{hold}$

($T_{Delay}+T_{setup}+T_{hold} < T_{period}$ MAX和MIN的情况都必须满足这个条件)

2,时钟周期为T,触发器D1的建立时间最大为T1max，最小为T1min。组合逻辑电路最大延迟为T2max,最小为T2min。问，触发器D2的建立时间T3和保持时间应满足什么条件。（华为）

常见公司数字IC设计招聘—题目

3,说说静态、动态时序模拟的优缺点。（威盛VIA ）

4,一个四级的Mux,其中第二级信号为关键信号, 如何改善timing。（威盛VIA ）

5,给出一个门级的图，又给了各个门的传输延时，问关键路径是什么，还问给出输入， 使得输出依赖于关键路径。