**Verilog 必考点总结**

1. **设计题**

**程序查错：**位宽，reg\wire类型，组合电路的Latch产生（敏感列表完备，初始化， 选项完整）等。

**组合逻辑优化**：将晚到的信号放到靠近输出的位置。包括数据信号、控制信号。

**有限状态机**：注意分段以及初始化。

**LFSR**：参数化描述，主要是特征向量、起始值、终止值等。

**参数化程序编写**：由底层模块构成顶层模块。

**断言的编写**：

Property p\_name;

@(posedge clock) a ##1 b;

Endproperty

A\_name : assert property p\_name;

1. **简答题以及概念题**
2. **验证相关**

**验证vs测试**：

验证检查设计的正确性，主要方法为模拟方法、硬件仿真和形式化验证，一个设计仅需要一次完整正确验证，为设计的质量负责。

测试是检查生产结果的正确性，分两个步骤：设计期间产生测试逻辑、生产后应用测试，对生产出来的每个产品均完成一次测试，对产品的质量负责。

**主要验证方法**：模拟验证、硬件仿真、形式化验证。

**模拟验证和形式化验证区别及优缺点**：

基于模拟的验证是通过输入测试激励来判断设计是否满足要求，平等对待每一种可能的输入；形式化验证用数学方法证明设计的正确性，理论上最佳的验证方法，受限于问题规模，不需要激励而需设计的各种特性来判断设计是否满足要求；基于模拟的验证只能模拟到小部分的状态空间；基于模拟的验证很难判断验证是否完备；基于形式化的验证很难判断提取的设计特性是否完备。

**模拟验证的主要流程**：

根据设计得到验证计划，构造激励，预测输出结果->构建测试平台testbench-> 模拟运行->回归测试。

**模拟验证四要素**：设计、测试激励、参考输出、比较机制。

**形式化验证分类**：等价性检查，模型检验，定理证明。

**另一种分类方法**：黑盒、白盒、灰盒。

**验证周期**：功能规范；建立验证计划；开发环境；测试硬件设计和环境；回归测试；硬件制造；调试流片后的硬件；逃逸错误分析；常见的中断验证周期的情况。

**基本验证环境**：设计，测试激励，监视器模块，检查模块，计分板模块。

**模拟器分类**：基于RTL的（事件驱动，基于周期）；基于仿真器；基于电路。

**两种主要的模拟引擎**：事件驱动的模拟方法，基于周期的模拟方法。

**事件驱动的模拟**：输入值的改变称为事件，驱动模拟过程；仅当模型有事件发生时才评估模型的行为；输入恒定时不执行模型；输入的变化不一定导致输出的变化。

**基于周期的模拟**：现代数字设计多为同步设计；同步电路，状态单元的值在时钟沿改变；在每个时钟周期对边界节点求值。有更好的性能：不需要调度模块、算法简单；对HDL风格有严格的限制，不支持延迟控制、限制时序结构，不支持HDL测试平台相关的特性。

**流行的模拟引擎实现**：混合模拟。以通用事件驱动模拟引擎为基础，结合加速模拟的引擎（基于周期的模拟）；

1. **综合相关**

**概念**：综合是从基于HDL语言的RTL描述构造出电路门级网表的过程。

**步骤**：

翻译：将RTL语言转化为工艺无关的功能部件组成的逻辑电路的过程；

优化：根据约束对翻译结果作逻辑重组和优化；

映射：从目标库构成实际电路，得到门级网表。

**综合的两大特点**：约束驱动，基于路径。

**路径的分类**：

输入到FF的D端；

FF的clk到下一FF的D端；

FF的clk到输出；

输入到输出；

**什么是划分**：将复杂设计分割成更小部分的过程。

**方法**：编写HDL代码时，做初步划分；综合时对划分进行修改。

**好的划分标准**：

组合逻辑尽量不跨模块；

模块尽量以寄存器的输出为边界；

模块大小均匀。

**可综合的三大编码原则**：

想到RTL

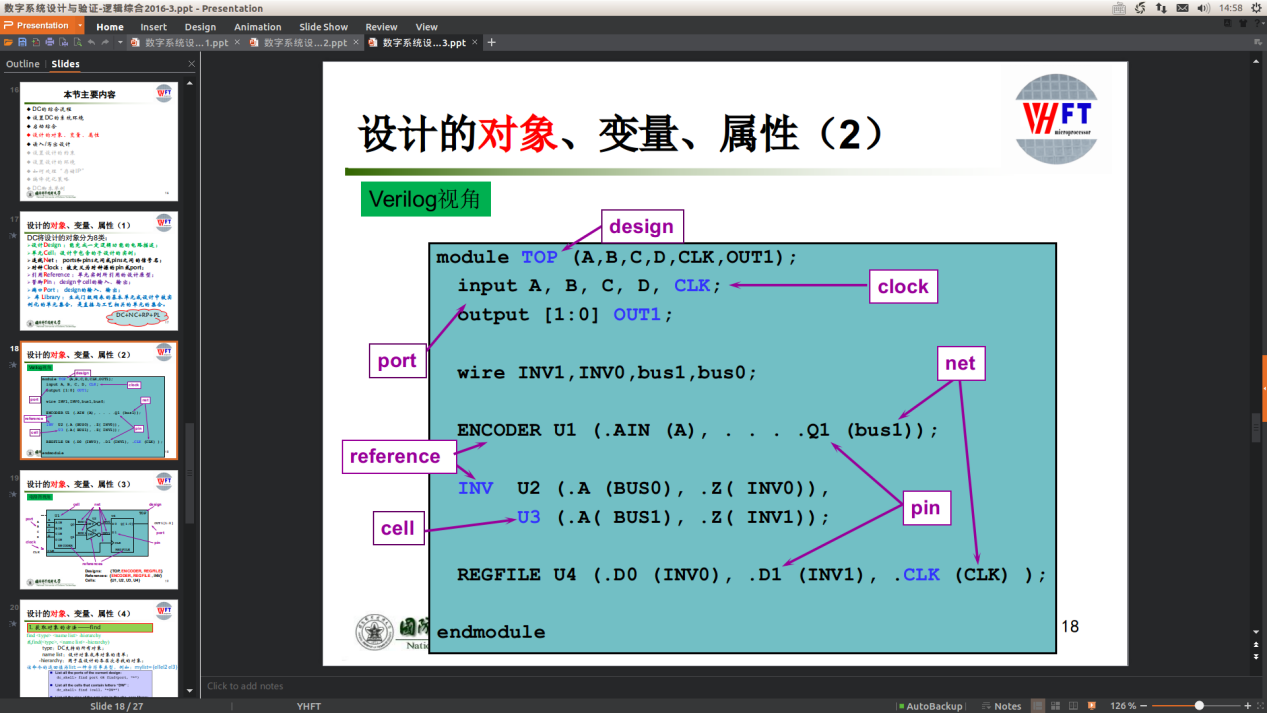
想到硬件

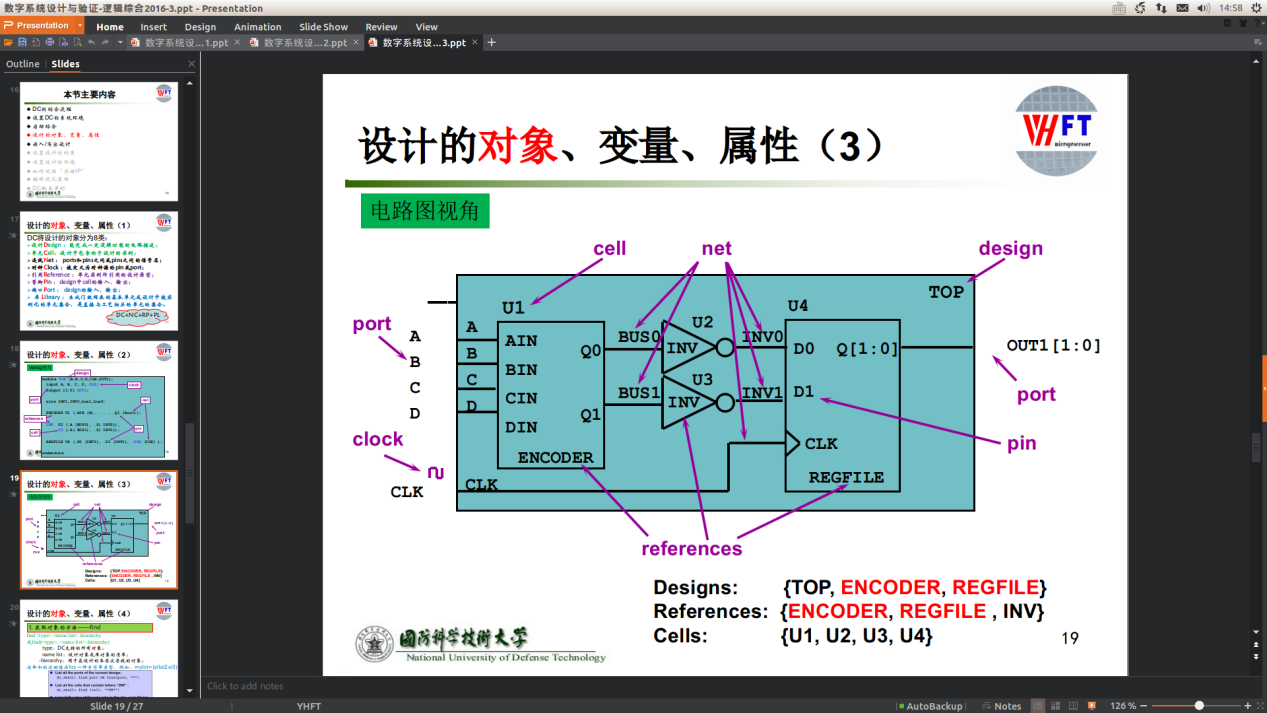
想到同步

**DC**：

.synopsys\_dc.setup:库搜索路径+综合库、目标库、链接库、符号库。

设计的八类对象：Design, ref, cell, ports, pin, nets, clock, library.





**获取对象方法**：

Find (<type>, <name list>, -hierarchy); find (ports, port\_name);

Get\_types : get\_ports (port\_name);

**变量**：set， unset, echo

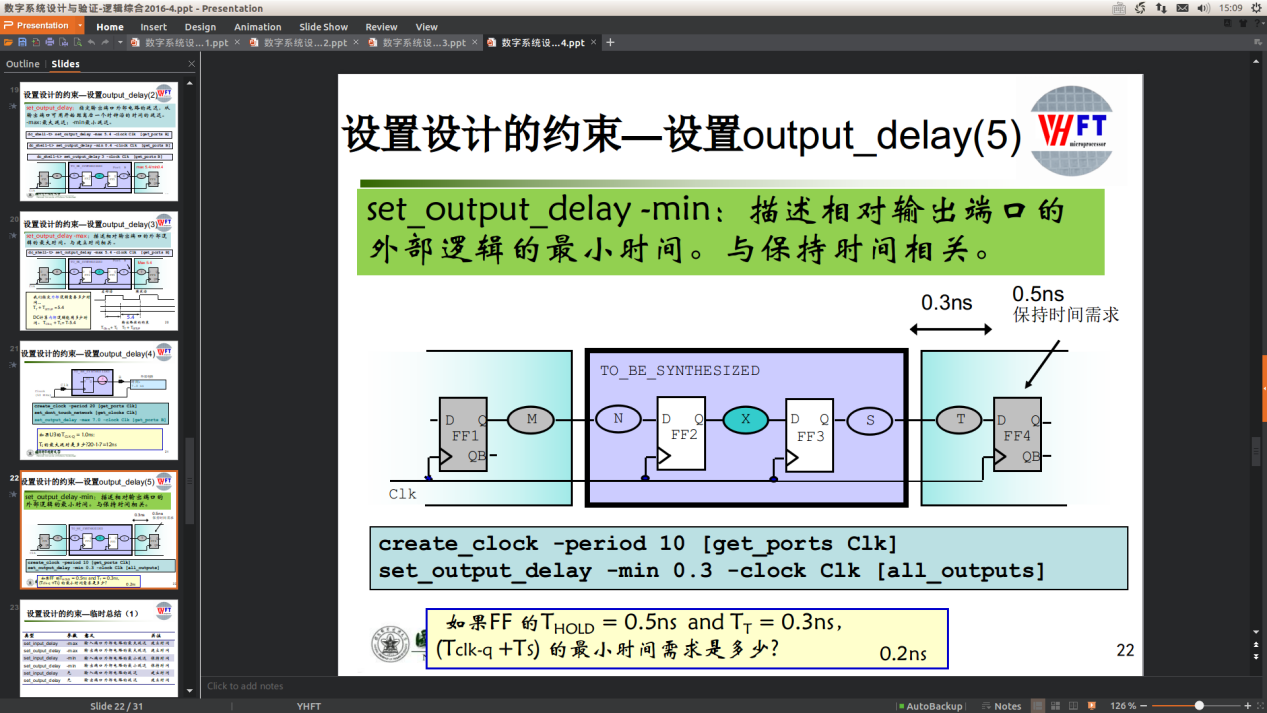
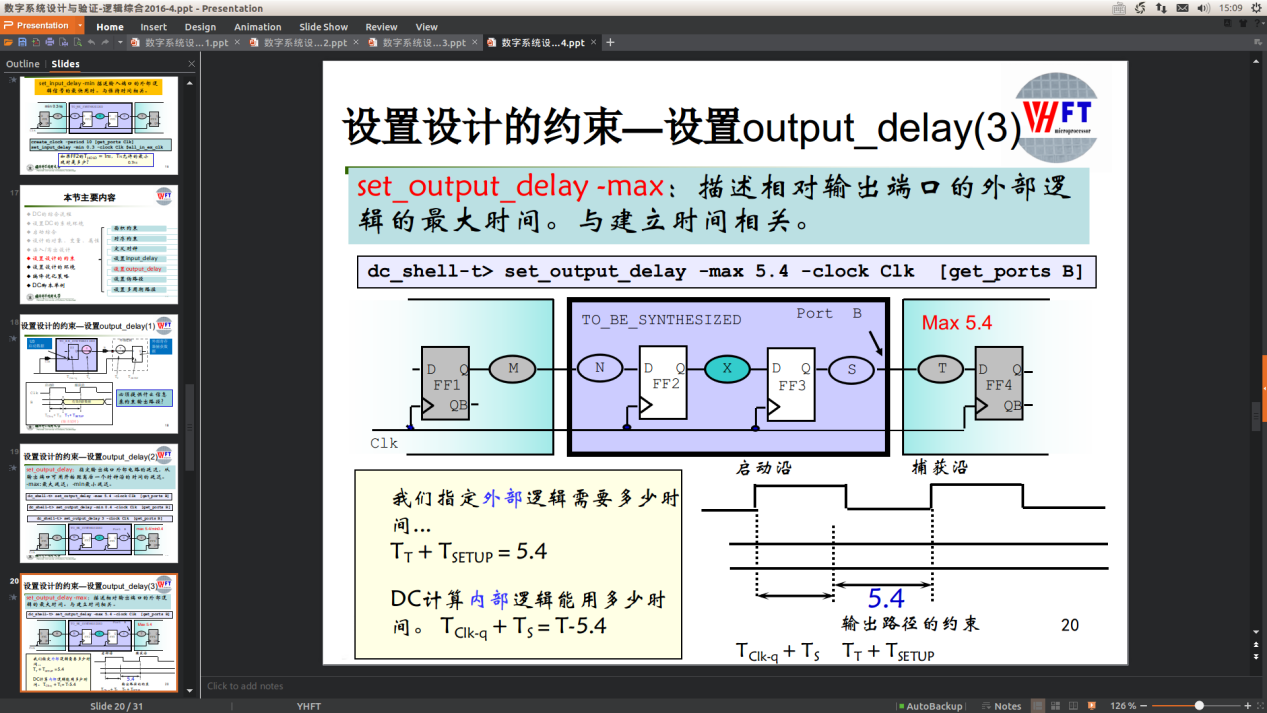
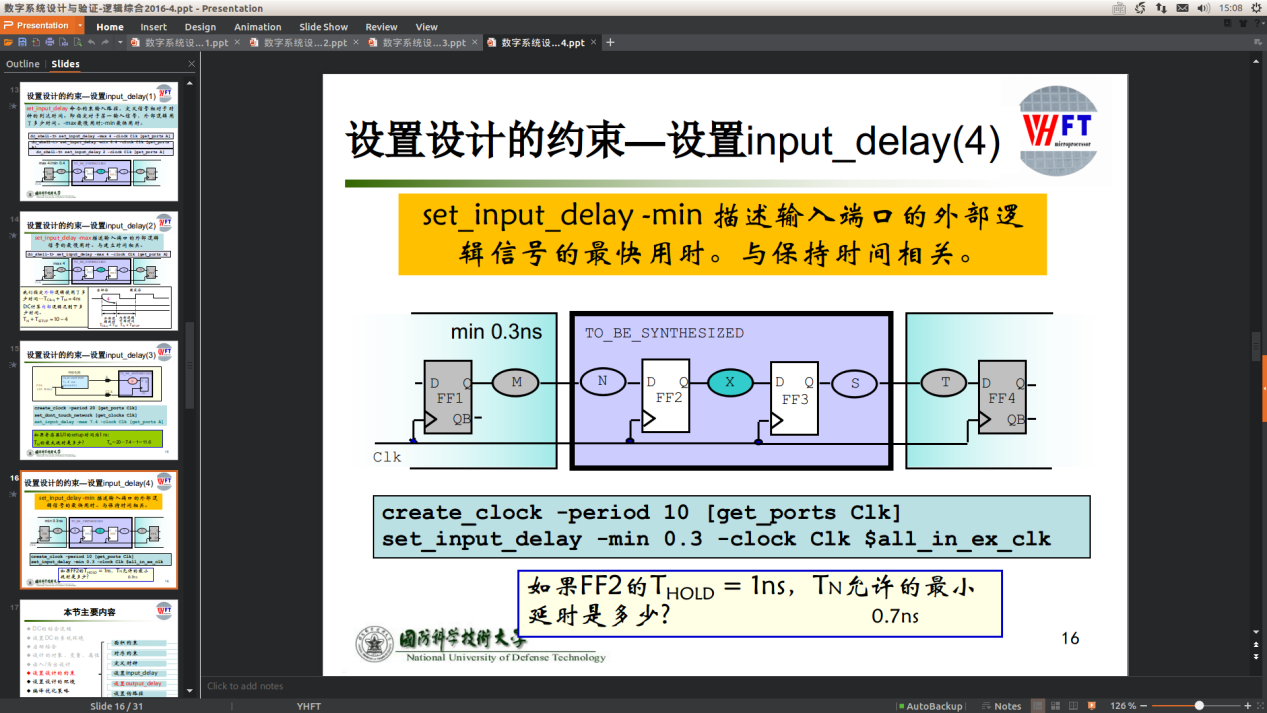
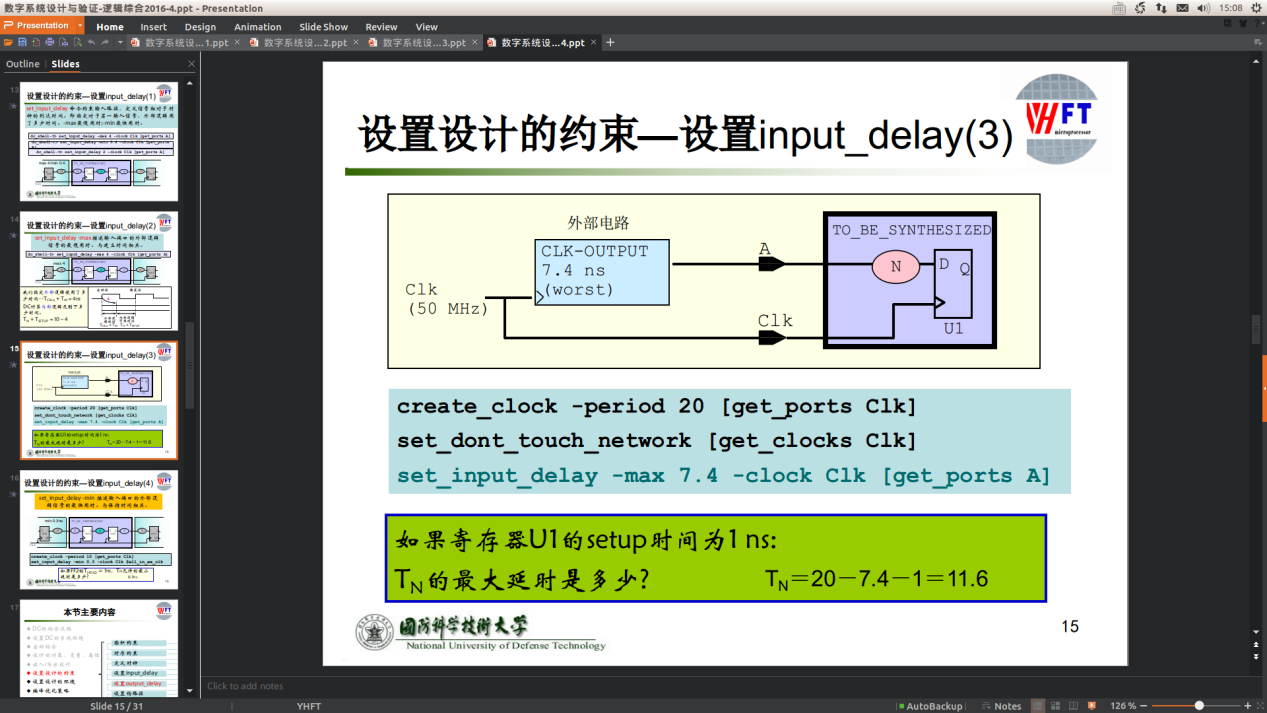
**属性**：set/get/remove\_attribute

**设置综合的约束：**！

**时序约束**：

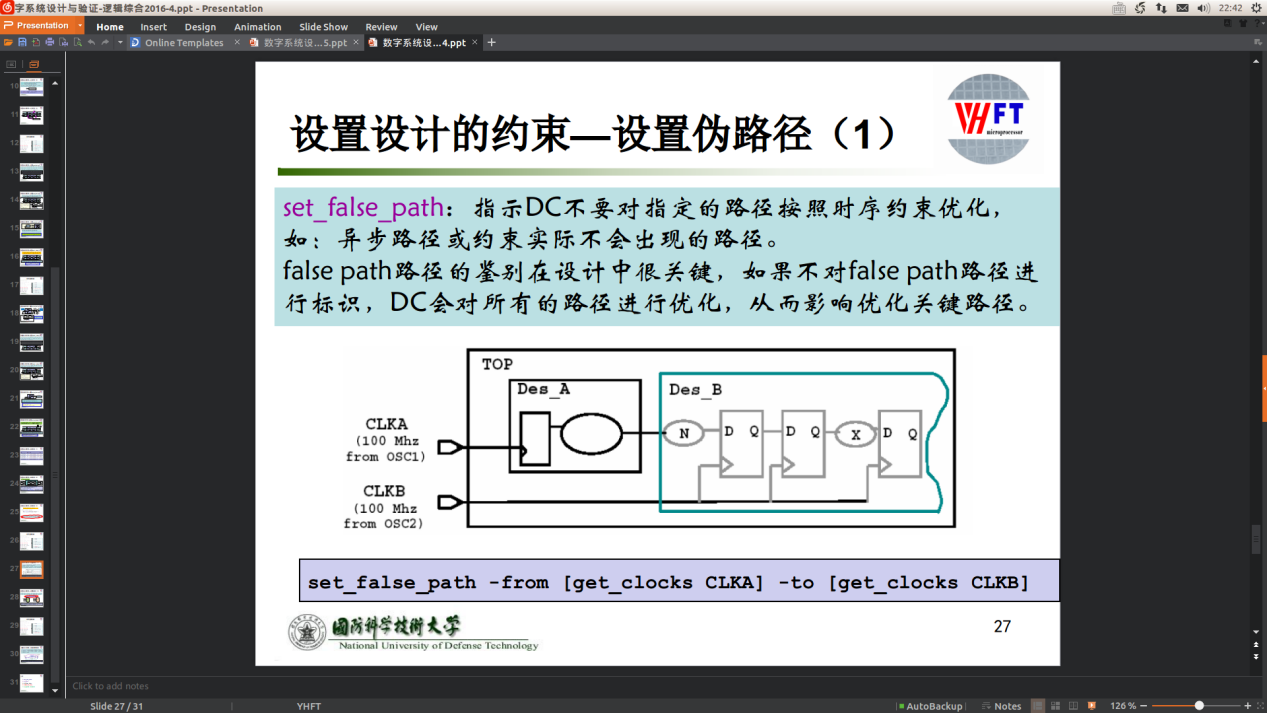
建立时间：触发器的时钟上升沿到来以前，数据稳定不变的时间。

保持时间：触发器的时钟上升沿到来以后，数据稳定不变的时间。



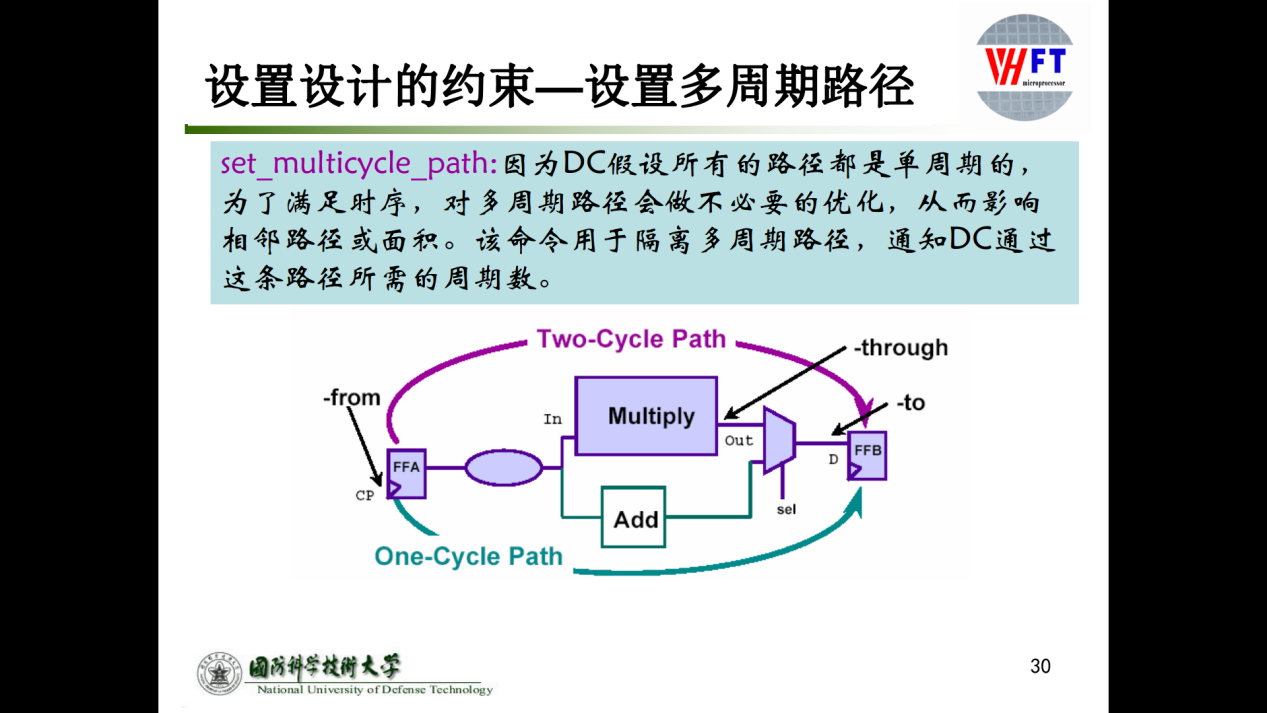
**伪路径**：指示DC不要对指定的路径按照时序约束优化，如异步路径或约束实际不存在的路径，另一种情况：输入被固定的时候。

**为什么设置伪路径**：伪路径的鉴别在实际设计中很关键，如果不对伪路径进行标识，DC会对所有路径进行优化，从而影响优化关键路径。



**多周期路径**：因为DC假设所有路径都是单周期的，为了满足时序的约束，对多周期路径会做不必要的优化，从而影响相邻路径和面积。该命令用于隔离多周期路径，通知DC通过这条路径所需的周期数。

Set\_mulitcycle\_path -from -to

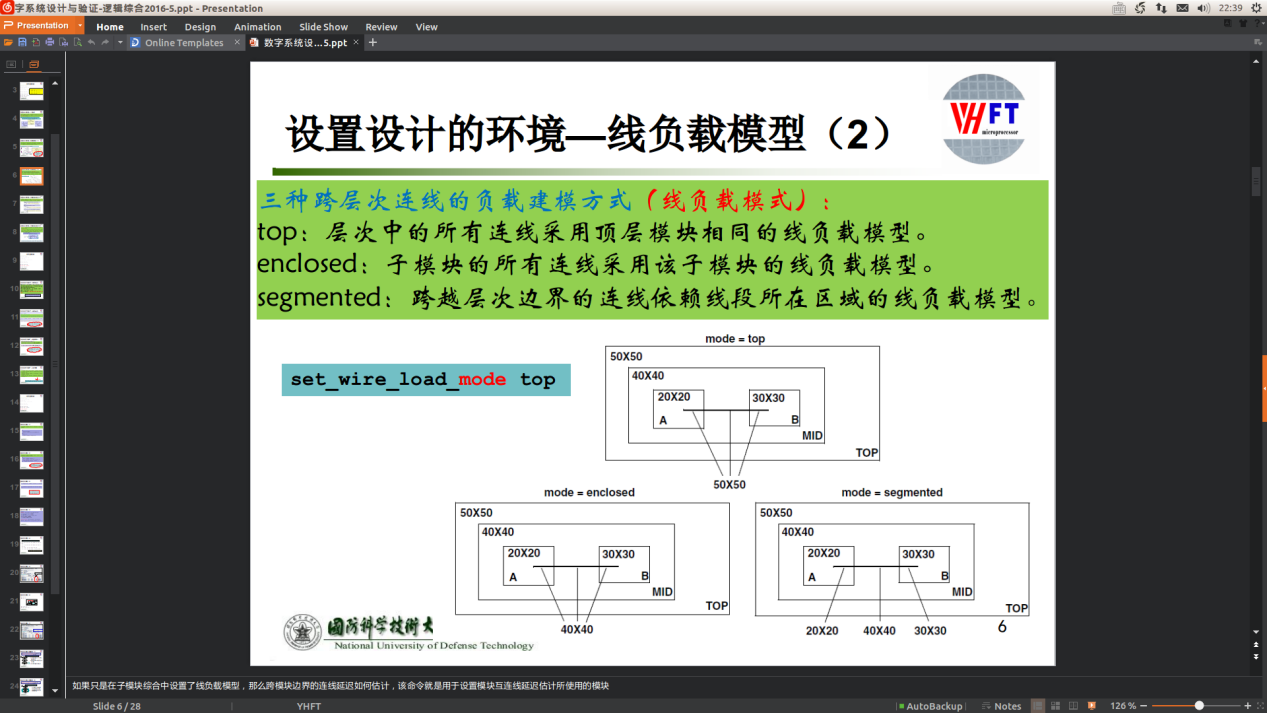


**三种跨层次连线的负载建模方式**：

Top:层次中所有连线采用顶层模块相同的线负载模型。

Enclosed:子模块的所有连线采用子模块的线负载模型。

Segmented:跨越层次边界的连线依赖线段所在区域的线负载模型。



**Lib转db**:

Read\_lib name.lib

List\_libs

Write\_lib -format db name -output name.db

**延时缩放**：

由于后端布局布线、CTS插入等因素，一般来说综合时的约束，往往更严格于目标频率的要求。

存储IP不需要考虑后端布局布线，CTS插入等因素，其周期可以更大点。

如果直接综合，综合器会只报出存储IP的相关路径违反且不会去优化其它路径。

**编译优化策略**：

**自顶向下**：将整个设计作为一个模块进行编译，仅需Top\_level约束。

优点：仅需Top\_level的约束；将整个设计作为整体优化，可获得较好的结果。

缺点：编译时间长；子模块的改变需要整个设计重新综合。

**自底向上**：子模块可以独立编译，适合于划分合理，各自模块的时序约束能够被明确定义的设计。

优点：减少多次被引用模块的综合时间；每一个子模块都有自己的script文件，便于管理；子模块的改变不需要整个设计全部重新编译。

缺点：需要维护多个scripts.

**编译何时停止**：

满足约束；

综合工具无法找到对关键路径更优的优化方案。

1. **物理设计相关**

**流程**：综合->设置（逻辑数据+物理数据设置）—》设计规划—》布局—》时钟树综合—》布线—》芯片完成。

**FloorPlan+PowerPlan**: 放好IO Pads, 确定chip/core边界，创立了单元行和走线通道，完成硬宏的布局。

**布局以后的步骤**：标准单元位置确定，时钟树建立，时钟及信号布线完成。

**逻辑库**：为所有标准单元提供延时和功能信息，为硬宏提供延时信息，定义驱动/负载等设计规则。

**物理库**：包括标准单元和宏单元的物理信息用于布局，定义布局单位面积。

**布局**：

**Pre-CTS做什么**：时序分析假设当前是理想的时钟树，产生较乐观的timing结果。

**时序驱动的布局**：试图将关键路径上的单元靠近放置以减少电容来满足setup要求；线电容是基于虚拟布线算法来估计的。

**拥塞**：当小区域里面的走线条数超过限制后，这个区域被称为拥塞。

**拥塞驱动的布局**：

使高拥塞区域的单元散开放置。

**时序vs拥塞驱动**：关键路径的单元可能会散开来减少拥塞，这些路径可能出现时序违反。

**拥塞的解决**：

重新运行拥塞驱动的布局；

修改物理约束，调整拥塞区域的单元密度。修改floorplan（移动硬宏，修改设计的形状和尺寸，移动引脚位置）。

**Power优化技术**：

门控时钟减少了功耗和面积。

**漏流功耗**：关键路径上使用高Vth单元来提高timing, 其它路径使用低Vth来降低功耗。

**动态功耗**：低功耗布局（寄存器靠近摆放）；门级功耗优化。

**CTS**：

**CTS做什么**：插入Buffer满足skew, latency, transition目标；

**CTS目标**：

满足时钟树的设计规则（最大信号切换时间，最大负载，最大扇出）；否则会报错。

满足CTS目标（最大skew， 最大/最小插入延时）；否则会影响性能。

**布线**：

**布线目标**：布线为所有的时钟和信号通过金属层创建物理连接。

**串扰**：是通过耦合电容从一条信号切换线到另一条静止或信号切换中的线的电压传递。

**串扰导致的后果**：

在另一条静止的信号线上产生噪音。

在另一条信号切换中的线导致信号加快或延时。

**串扰预防**：

在布局阶段：设置最大transition, 控制拥塞；

CTS时：用自定义布线规则--如双倍间距。

布线阶段：采取预防措施。

**ECO**：

**硅冻结ECO**：

只能改变金属层，使用之前插入的Spare单元。

单元布局无变化；

删除的单元可成为Spare单元；

ECO单元只能被映射到靠近所需位置的Spare单元上。

**非硅冻结ECO**：

对布局布线无限制；

对已有版图影响小；

单元可以放到理想的位置上。

**Spare单元**：

是在设计阶段被布局但未被使用且无功能单元链接的额外逻辑。

如在设计后期需要改变功能,可以链接这些Spare单元，只改变金属层的版图。

Spare单元分布在core中；

不会被当做无用单元删除。

1. **断言**

SVA，

**两种断言**：即时断言，并发断言。

**即时断言**：基于事件的变化，表达式的计算像Verilog中组合逻辑一样，是立即求值的，与是时序无关。

**并发断言**：基于时钟周期的，在时钟边沿计算表达式，以关键字property定义。

断言是对设计应当如何执行功能的描述，是嵌入设计的检查。

**为什么使用断言**：

让设计师把设计意图传递给验证过程，改善了设计师与验证师的沟通；

让验证师确认自己对设计的理解正确；

可对设计内部进行检测，增加了设计的可观察性；

实现代价低，对模拟性能不影响；

基于断言发现大量的BUG；

模拟验证和形式化验证都可以使用；

可用于覆盖率统计。

**断言类型**：实现型（设计师负责），规范型（验证师负责）

**实际使用的断言类型**：OVL，PSL，SVA，0in CheckWare, IAL等。

**使用示例**：

Property p\_name;

@(posedge clock) a ##1 b;

Endproperty

A\_name :

Assert property p\_name;

1. **等价性检查**

**什么是等价性检查**：

假设参考设计是正确的

确定转换后设计的功能是相同的

是穷尽对比，无细节遗漏

不需要验证激励

**设计包括多个逻辑锥和比较点**。

**比较点**：

设计的输出

设计中的寄存器

设计中黑盒输入

**逻辑锥**：

驱动每个比较点的组合逻辑

**等价性检查**：

输入：寄存器的输出；设计的输入端口；黑盒的输出；

输出：寄存器的输入；设计的输出端口；黑盒的输入；

**比较点匹配**：

对齐比较点称为比较点匹配。

逻辑锥的终点

每一对匹配的比较点，对所有可能的输入组合产生同样的结果，则标记为pass, 否则标记为failed。

**参考设计**：是golden设计。

**转换后设计**：修改后的设计，要和参考设计进行比对检查。

**流程**：读入参考设计-》读入变换后的设计-》设置-》匹配->验证-》完成。

**什么会导致比较点不匹配**：

设计对象没有相似的名字，

缺少设置；

模拟/综合的区别；

冗余寄存器在综合时被优化掉；

参考设计和转换后设计的黑盒不一致。

**怎样算验证通过**：

如果读取了参考设计的一个比较点：

转换后设计必须有一个匹配的点；

转换设计比较点的值必须与参考设计中的一样。

**内部扫描是什么**：

由DFT实现，

扫描链使设计者在做制造后测试时设置和观察内部寄存器状态时更加方便。

**门控时钟是什么**：

是寄存器时钟路径上的附加逻辑，它能在寄存器输出不改变的情况的时候关闭clock。

使用门控时钟的目的是通过减少不必要的单元上的clock切换来节省功耗。

**验证失败的主要原因**：

模拟综合的区别；

设置不全；

HDL代码风格不好；

两个设计的黑盒不一致；

手工错。

1. **静态时序分析相关**

**概念**：

采用适当的时序计算模型,对逻辑电路进行分析，判断是否满足时序约束的要求；

和功能是否正确无关；

穷尽式分析；使用形式化和数学的方法。

速度快；

**时序路径**：

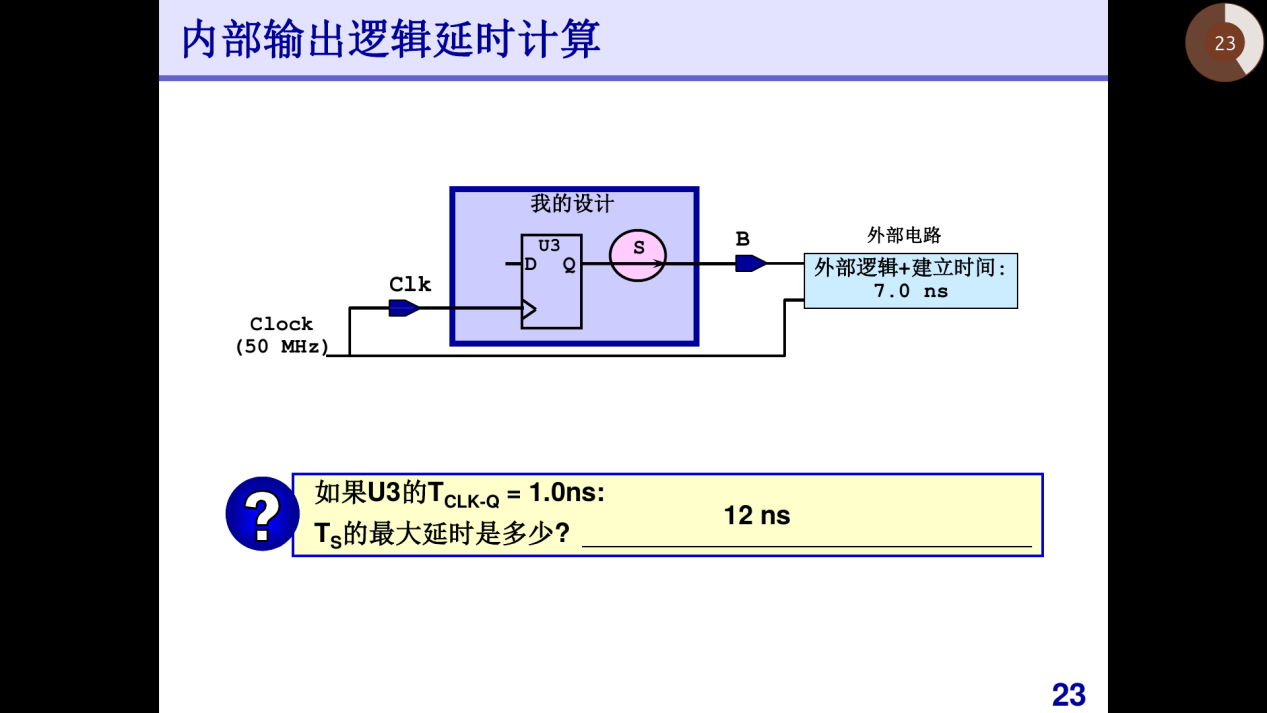
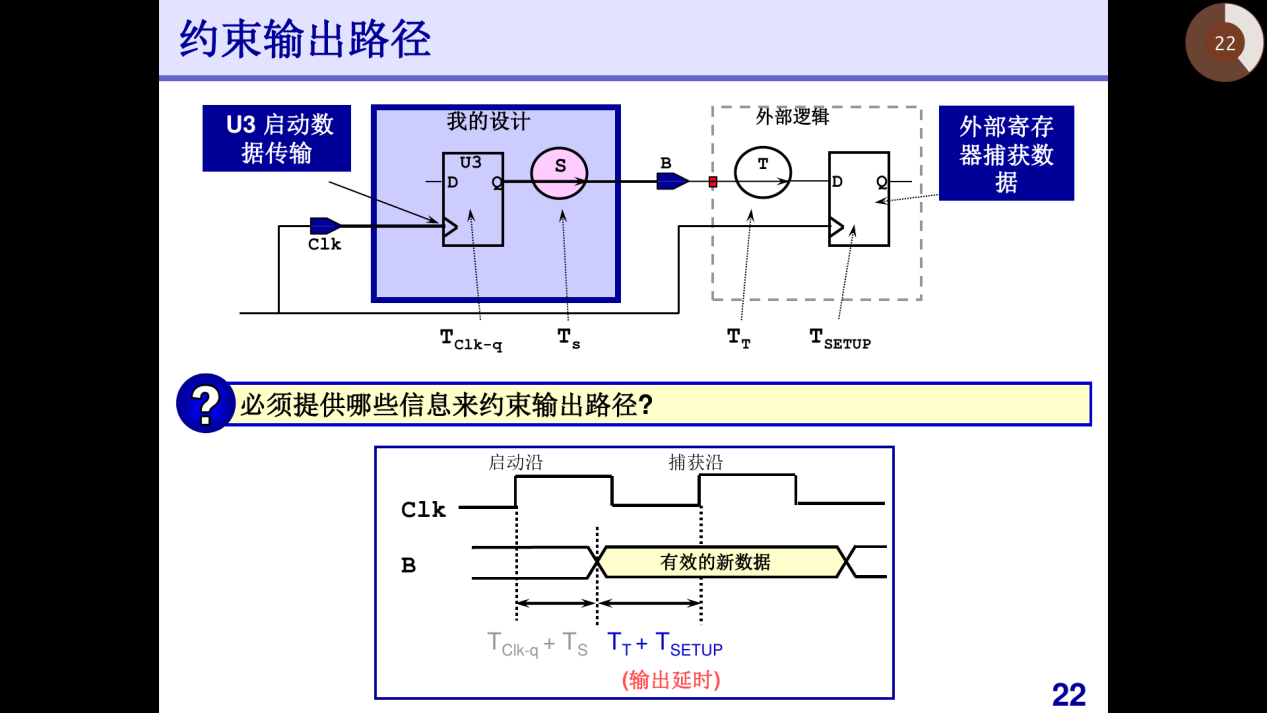
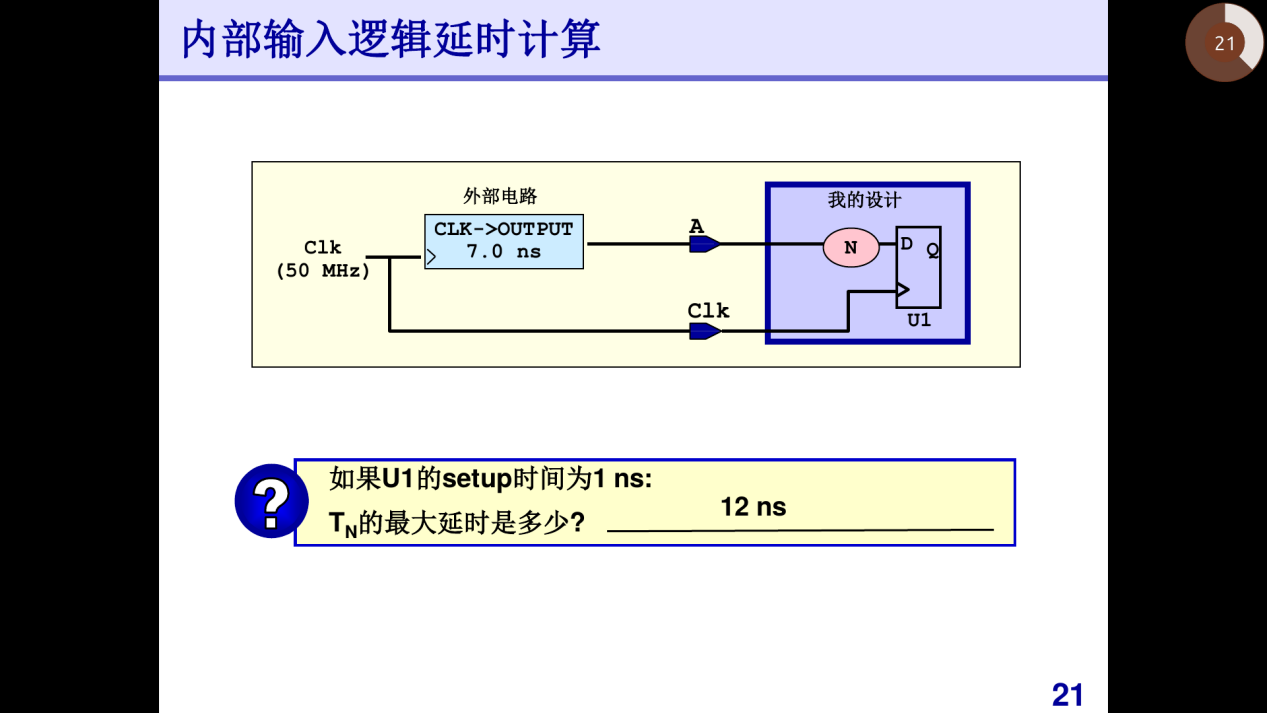
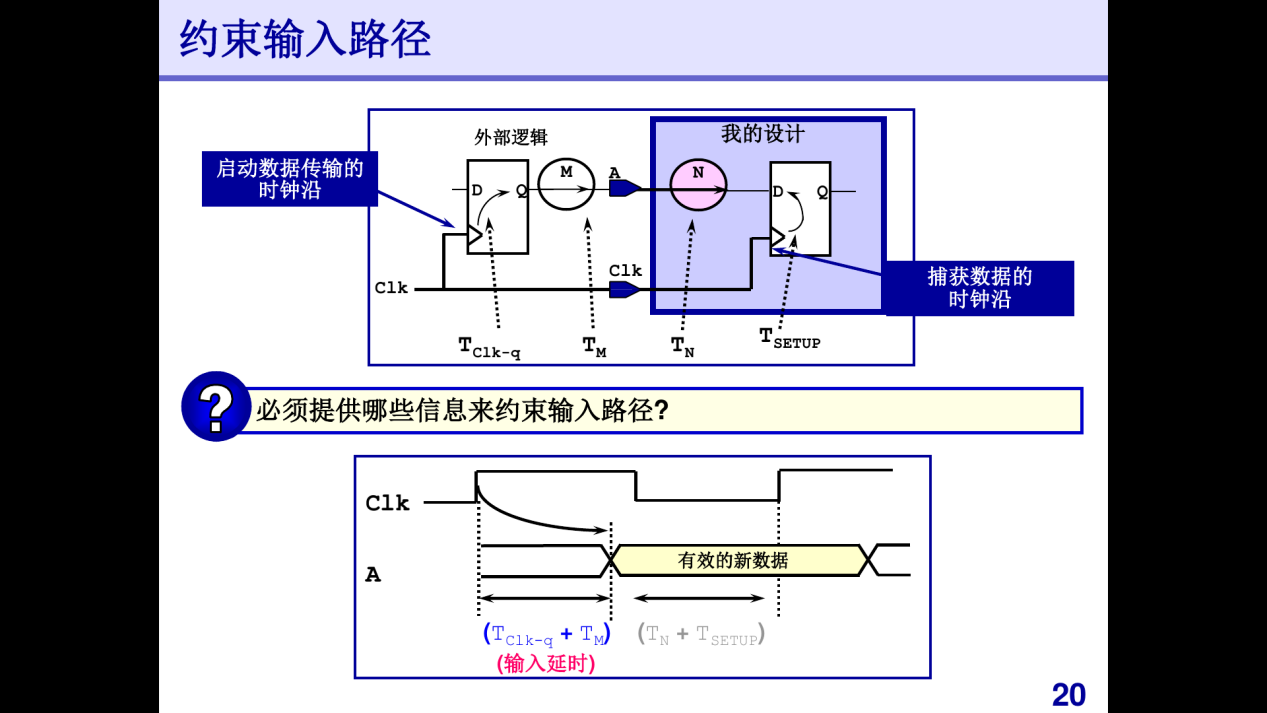
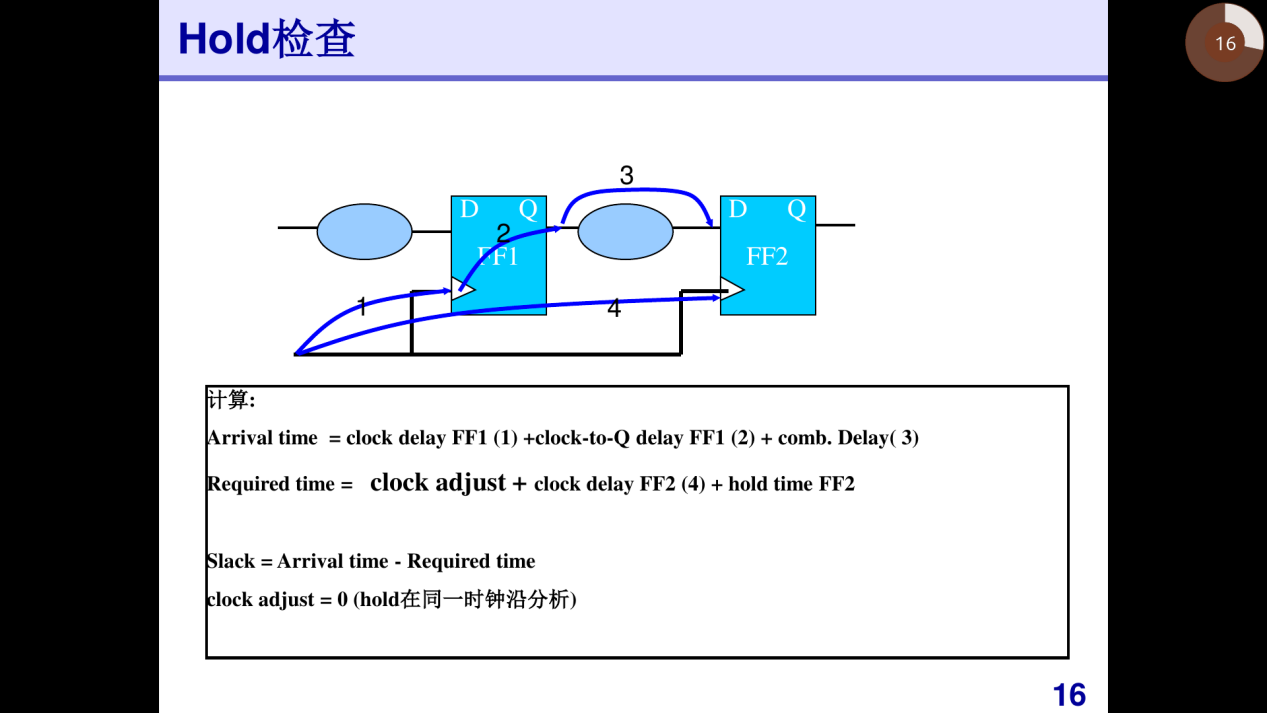
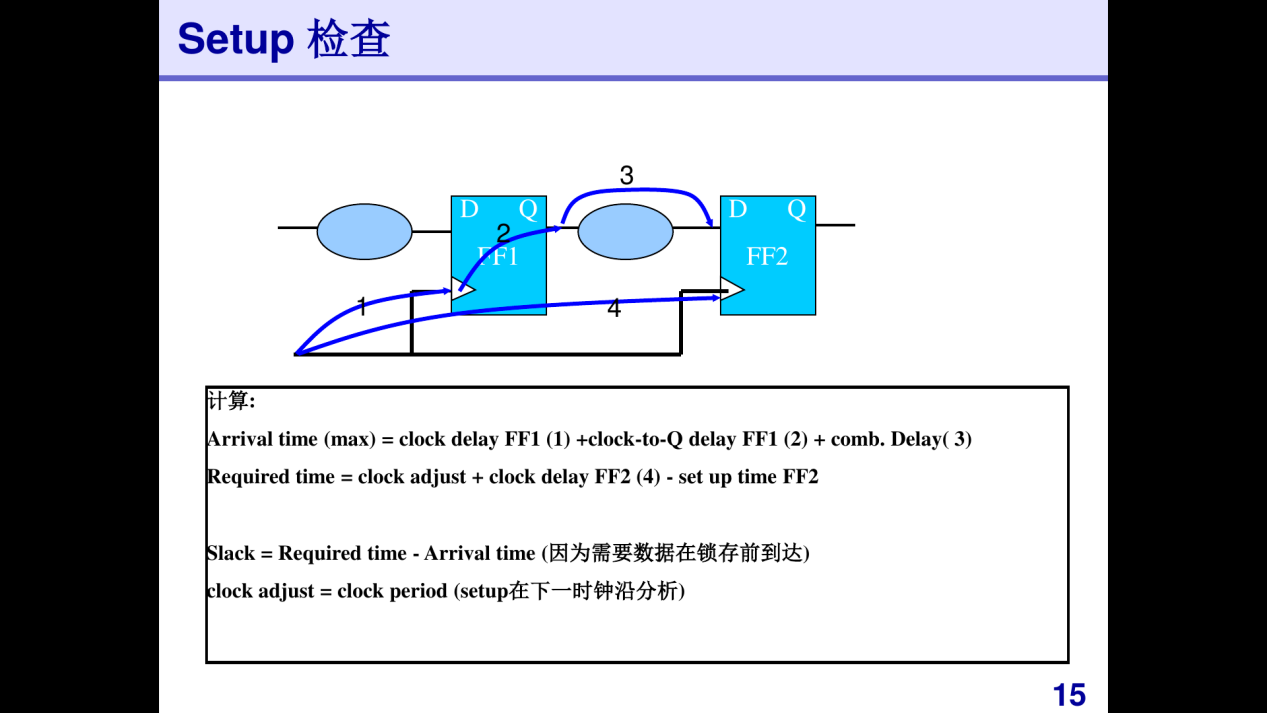
起点：输入数据端口；寄存器的时钟引脚；

重点：输出数据端口；时序器件除了时钟引脚以外的所有其它输入引脚。

**怎样是满足时序约束**：

所有寄存器都可以在制定时钟沿可靠的拿到数据。

**延时计算**：



**时钟属性**：

Insertion delay: 从时钟源到时钟叶节点的延时；

Skew ：从时钟源到各时钟叶节点时间的差值；

Jitter ： 时钟源不稳定的变换；

Transition：时钟信号从低到高或从高到低所用时间。

STA假设所有路径为一个周期的约束。

从而要设置伪路径，多时钟路径。

**时序计算步骤**：

所有时序路径被分为组；

计算每一条路径的延时；

查看所有路径的延时是否满足延时约束。

**时序计算**：

单元延时计算：

非线性延时模型；

线延时计算：

根据真实的RC或线负载模型来计算。

**单元延时计算**：

在模型中，输出负载和输入切换时间决定了输出延时和输出切换时间；

输出切换时间又成为下一单元的输入切换时间；

每个单元都有两个表：单元延时和输出切换时间。

**时序报告解读**：

分四个部分：路径信息，路径延时，路径需求，总结。

1. **计算题**

**建立时间、保持时间是否满足**：

建立时间：slack = RT - AT; 也就是说，数据到达时间要比需要的时间小。

保持时间：slack = AT - RT; 也就是说，新数据到达的时间要晚于需要保持的时间。

**计算输入、输出最大延时**：周期减去已知的时间，最大延时与建立时间有关，最小延时与保持时间有关。

关键是理解保持时间中，clock adjust为0，且新信号到达时间要晚于保持时间；在建立时间中，clock adjust为一个时钟周期，且新信号到达的时间要早于建立时间。