Synthesis

Synthesis Levels

- hight level synthesis
  - translation from **algorithmic behavior** description to **RTL** description
  - Input: HDL description code
  - Results: datapath(a network including registers, function units, multiplexer, bus for data transferring), controller(including hardwared logic or firmware for controlling the data transferring in datapath)
- logic synthesis
  - Translation from **RTL** description to **logic gate netlist** (automatic process).
  - ~~Translation from abstract description to detailed description.~~
- layout synthesis

Constrains

用于综合的约束可以用用户定义属性来表示。用户定义只是用于给综合工具传递约束信息，仿真时被忽略。

```
attribute  属性名 ： 数据类型 ；
attribute  属性名  of  对象名 ：对象类型  is  值；
```

约束种类

- 资源约束
  - 基本元件的种类个数
- 时间约束
  - 假定一个时序系统工作在20MHZ的时钟频率，那么通过状态寄存器和状态产生逻辑的最大延迟不能超过50ns（20MHZ的倒数）。
  - 状态机编码约束
    - 自然二进制，格雷码，onehot

VHDL Synthesis

Common features:

1) Supported subset

Supported types: `enumeration` , `integer`, `bit`, `std_logic`, `1darray`
Partially supported types: `float`, `record`,`2darray`

2) Unsupported subset

Types without corresponding hardware can't be synthesized, e.g. FILE. Type :Time (after/wait for statement)

3）Rules for process description

~~It's more free to write a process for simulation, but~~ **for synthesis, a process includes at most one clock signal**. ~~Some synthesis tools has further limitations on process description.~~

4）Synchronous design are preferred.

Synchronous description style is recommended by most of existing EDA tools, ~~the circuit states change according to the clock rhythm. For asynchronous design, synthesis results usually need to be optimized by designer.~~

Synthesizable data types

-**a) bit, boolean, bit_vector**

- **b) character, string**
- **c) integer**
- **d) std_ulogic, std_ulogic_vector, std_logic, std_logic_vector**
- **e) signed, unsigned**

Enumeration type

- For more abstract boolean and state_type, state encoding is necessary.(给enum里面的元素用二进制编号)

Integer

- synthesizable integer should **have a range specification**

Array

- 1darray is usually synthesized as bus
- 2darray can be synthesize into RAM

Record(not supported by all tools)

- Difference between array and record:
  - Array: all elements should be of the same type.
  - Record: elements can be of different types

Synthesizable data objects

- Constant
- Variable
- Signal
- ~~File~~

注意并非所有综合器都支持变量综合

Variable and signal has different behaviors in simulation, and they produce different results in synthesis.**(P194-195,注意生成寄存器的数目)**

~~1）Generally，**better synthesis results can be obtained by making good use of variable**. Because synthesis optimization can be more flexible by using variable .But not all synthesizer support variable synthesis~~

~~2）By using signal in process, **good consistency** can be achieved in signal I/O. It is usually necessary to use a signal if an intermediate data should be hold.~~

初值

三种赋初值的方式，见**P195**，建议使用第三种，即在process内部一开始就赋初值

Synthesizable operator

Logic operator

包括二元逻辑运算符如and``or等以及not操作，这些在综合的时候直接调用逻辑门单元实现，但不排除在优化时会被改动

Relational Operator

没有统一的综合方式，但可以综合

Unary arithmetic operator (一元算术运算符)

+ - 可以被综合，但是abs不可以

Binary arithmetic operator (二元算术运算符)

通常支持加减乘操作，部分支持除，取模运算一般不被支持

Synthesizable Statements

Sequential Statements

if statements

- if语句包含所有可能性，此时被综合为多路选择器
- 不完整if语句会产生寄存器。如果if语句判断条件为检测某一信号的跳变，则一般综合成FF。

case statements

- 不平衡的case语句可能也会产生寄存器，所谓不平衡是指不同case里面的操作次数不同（见下面"避免引入多于寄存器"的4）

loop statements

- In RTL synthesis, **the maximum number of iterations should be a finite, known and static value**.
- P200-201两个例子，注意第二个用了next使得循环能够有选择性地执行

Concurrent statement

Synthesizable concurrent statements include process, concurrent assignment, block statement, generate statement.

寄存器和锁存器的可综合描述

寄存器的引入方法

触发器的引入

- P205-207
- 重点注意
  - 方法三没有敏感参数表
  - 方法四结合敏感参数表和IF进行操作
  - 方法五使用的WHEN...ELSE语句中的ELSE要删掉

锁存器的引入

- P207-209
- 与引入寄存器的区别：敏感参数表多了一个a
- CASE语句的不完全覆盖也会引入寄存器（例8-20），注意WHEN OTHERS=>NULL并不意味着完全覆盖的操作
- 例8-21中有一个BUFFER类型变量的例子，要注意

具有时钟门控结构的触发器引入

例8-22这种IF语句里面有三个并列的方式不建议使用，应当改用8-23中的两个IF嵌套方式

~~同步复位、置位功能引入~~

~~其实就是上面的那种，课本还tm专门拎出来讲，sb~~

异步复位、置位

- P211例8-25
  - 就是让复位操作优先级最高，给它放在检测上升沿IF语句上面就行！！
- 置位复位都要有->和上面一样的，就是多一个复位分支

避免引入多余寄存器

1. 将不需要进入寄存器的语句单独提出来组成一个process，以避开wait until clk'event或者不完整if的包含
    1. 重点看8-27和8-28
2. 避免边沿触发的引入
3. IF CASE涵盖要完整
4. 如果信号或者变量在一个CASE里面有赋值操作，那么在其他CASE里面也要有赋值操作（操作要平均）；或者在CASE语句前面统一赋初值，这样没有覆盖到的都会用初值而不需要寄存器了（例8-32）

优化

优化的两个方面：

- 速率优化
  - 流水线优化：拆分为若干个过程，每个过程并行运行
  - 合理使用嵌入式阵列块EAB资源和LPM宏单元库来提高乘法器速度
  - 关键路径优化：关键路径指**输入到输出延时最长的逻辑通道**
- 面积优化
  - 资源共享：通过数据缓冲或多路选择的方法来共享数据通道中占用资源较多的模块（如乘法器、多位加法器等算术模块）（如两乘法器一选择，变为一选择一乘法）

针对传输延时的要求(要会计算！！)

- 一个时钟周期的最坏情况为$T = T_{clk} + \delta + t_{jitter}$
- 传输所需要的最短时间为$T_{delay} = t_{su} + t_{c-q} + t_{plogic}$
- **结合起来应有**$T_{clk} + \delta + t_{jitter} > t_{su} + t_{c-q} + t_{plogic}$

惯性延时（固有延时）

- 信号脉宽小于器件的惯性延时，器件对输入信号不做任何反应
- 对应语句`z <= x after 5ns`

传输延时

- 不考虑信号的脉宽，仅仅表现为普通的延时
- 对应语句`z <= TRANSPORT x after 5ns`

提高综合效率的策略

1. 尽量不使用WAIT FOR XX ns语句和AFTER XX ns语句。XX ns表明在执行下一操作之前需要等待的时间，但综合器不予支持，一般忽略该时间，而不会综合成某种元件，故对于包含此类语句的程序，仿真结果与综合结果往往不一致。
2. 声明信号和变量时尽量不赋初值，定义某确定数值时，使用常量而不用变量赋初值的形式。因为大多数综合工具将忽略赋值等初始化语句
3. 函数或过程调用时尽量使用名称关联。不要在同一个语句中同时使用两种关联。

```
clk_1:bufes port map(I=>clock_in，clock_out)；--(不正确的用法)
clk_1:bufes port map(I=>clock_in，O=>clock_out)；--(正确的用法)
```

4. 正确使用when_else语句、if_else语句和case语句，条件涵盖要完整，防止产生不必要的寄存器
5. 注意算术功能的设计优化。

```
Out<=A+B+C+D;
Out<=(A+B)+(C+D);
--第一条语句综合后将会连续叠放3个加法器（（（A+B）+C）+D）；第二条语句（A+B）和（C
```

- 在进程中, 对变量要先读后写。 因为变量值是立即获得的, 如果先写后读就会产生长的组合逻辑和锁存器。因此, 在编写代码时, 对变量要先读后写.