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

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~~

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）