10. 代码规范

10.1 命名惯例

- 一个文件最多一个 module 模块。
- 分离模拟、数字以及混和信号的 verilog 文件。
- 文字数字以及下划线是允许的字符集，文件名的第一个字符必须是字母。
- 要避免使用 escaped names。
- 信号名字的拼写和风格前后、层次上下要一致。
- 所有的名字不管大小写必须唯一。
- 全局文本宏要包括 module 名字。
- 信号名字长度不得超过 32 字符。
- 要避免不寻常的缩写。
- Instance 名字要和其 module 名字一样。
- 文件名字一般命名为如下格式：<设计单元名字>[_tb].<v, f>。
- 名字有很多分开的名字组成的要有下划线分隔。
- Module 名字必须与文件名相同。
- 参数和宏的名字必须是用大写字母，管脚、变量、结构（比如函数，任务，module）以及实例（比如门级描述等）则必须采用小写字母。
- 最好采用有意义的名字，而且不能与 verilog 的关键字冲突。
- 后缀的使用一般要遵循以下的规则。
    - 低电平有效信号一般命名以_b 结尾
    - 时钟信号命名一般以_clk 结尾
    - 状态机状态命名以_ns 结尾
    - 测试模式信号一般以_test 结尾
    - 模拟信号以_ana
    - 寄存器输出命名以_reg 结尾
    - 系统信号以_sys 结尾
    - 没有连接的输出信号以_nc 结尾
    - 高阻抗信号以_end 结尾

10.2 注释说明

- 所有的文件开头必须有一个文件头，提供一些相关的设计信息，比如版本号之类的，以//==========结尾。如下图所示

liupeng@zju.edu.cn

```
// +FHDR----------------------------------------------------------------------------
//                          Optional Copyright (c)
//      This confidential  and  proprietary  software may be used only as
//      authorized by a  licensing  agreement  from  Zhejiang University,
//      Institute of Information and Communication Electronic Engineering.
// ---------------------------------------------------------------------------------
// FILE NAME :
// DEPARTMENT :
// AUTHOR :
// AUTHOR'S EMAIL :
// ---------------------------------------------------------------------------------
// RELEASE HISTORY
// VERSION DATE AUTHOR DESCRIPTION
// 1.0 YYYY-MM-DD name
// ---------------------------------------------------------------------------------
// KEYWORDS : General file searching keywords, leave blank if none.
// ---------------------------------------------------------------------------------
// PURPOSE : Short description of functionality
// ---------------------------------------------------------------------------------
// ABSTRACT: Port description
// ---------------------------------------------------------------------------------
// PARAMETERS
// PARAM NAME RANGE : DESCRIPTION : DEFAULT : UNITS
// e.g.DATA_WIDTH [32,16] : width of the data : 32 :
// ---------------------------------------------------------------------------------
// REUSE ISSUES
// Reset Strategy :
// Clock Domains :
// Critical Timing :
// Test Features :
// Asynchronous I/F :
// Scan Methodology :
// Instantiations :
// Synthesizable (y/n) :
// Other :
// -FHDR----------------------------------------------------------------------------
```

图 13　文件头示例

- 在代码的每个功能块前面必须附上代码的意图和功能。
- 一般采用一行注释（//），不采用多行注释（/*…*/）。
- 没有用的代码以及将要删除的代码必须注上修改日期和原因。
- 每个端口定义都因该有个描述性的注释，最好在同一行。如果不在用一行，就因该在前一行。
- 条件编译指示，如：“ifdef”，“endif”，和“else”都因该在使用的地方有个注释。

10.3 代码风格

- 相同类型的代码一般都排列在一起。
- 一般用两个空格缩进，而不是用 tab 键。
- 在同一行一般只写一句 verilog 语句。
- 每行只定义一个端口。
- 端口定义必须与在 module 定义的时候的排列顺序一样。
- 定义输入，输出以及内部信号必须在不用的部分，由注释来分隔。
- 行的长度一般不超过 80 个字符。

10.4 编码技巧

- 不赋值给信号 x 值。
- 不在两个 always 模块里给同一个 reg 型变量赋值。

liupeng@zju.edu.cn

- 参数必须使用“define”。
- 采用参数来定义信号的宽度。
- Modules 必须采用端口名引用而不是位置引用。
- 对于系统内信号一般都采用单向信号定义，而不采用双向来实现。
- 不用表达式端口连接。
- 参数和宏都不能定义后就不能改变。
- 如果一个常数的值依赖与另外一个常数的值，那么依赖性要在定义中表示出来。
- 操作数位数要匹配。
- 对于矢量类型端口定义和连线类型定义要宽度匹配。
- 所有的矢量因该在端口连接的时候都表示出位宽。
- 尽量避免使用 inout 信号。
- 在复杂的等式表达时要采用括号。

10.5 可综合设计

- 数据类型的可综合性
  a. 标量类型
     - 要根据数据的范围确定数据的位宽，这样可以节约逻辑资源，减少不必要的浪费。Reg 型的负数，RTL 级语言描述的程序综合后变为负数的补码表示，不支持浮点数。
  b. 矢量类型
     - 当一维数组的下标是变量表达式或者另一个数组中的元素时，综合后的时间延迟是很大的。通过对 reg 数据类型建立二维数组，可以描述 ROM，RAM，但一般综合前 ROM 和 RAM 最好还是替换成厂家提供的库单元，以减小面积。
- 运算符的可综合性
  - 在/、%运算中，除数通常被限制为常数，且必须是 2 的指数倍，而全等==，!==通常不被支持。
- 赋值语句的可综合性
  - 连续赋值语句 assign 是可综合的，综合工具把它综合成组合逻辑，赋值语句的左边是 wire 型，右边是 reg，integer 型。
  - 过程赋值语句一般是在 always 块内，有非阻塞和阻塞两种赋值。两种赋值语句不能同时使用，建议使用非阻塞语句。
  - 被赋值的变量只能在一个 always 块内对其赋值，但是可以在多个 always 块内被引用。
  - 非阻塞赋值必须用在边沿敏感连续代码块。而阻塞赋值则不被允许
- 组合逻辑和时序逻辑的可综合性
  - 用 assign 语句对 wire 型变量进行赋值，综合后的结果是组合逻辑电路。
  - 用 always@综合后的结果是组合逻辑电路或电平明感的锁存器。块中要避免组合反馈回路。每次执行 always 块时，在生成组合逻辑的 always 块中被赋值的所有信号必须都在敏感电平列表中列出。

liupeng@zju.edu.cn

- 在每个 always 模块只有一个时钟信号可以用于同步过程。
- 用 always@（posedge clk）或（negdge clk）块描述的电路综合为同步时序逻辑电路。
- 计同步时序逻辑电路的关键时建立描述该电路的状态转移的可综合的有限状态机模型。
- 不允许有其他的@event 语句，大多数综合工具不能综合异步状态机。

➢ 行为语句的可综合性

- 对于 for，repeat，while 其循环次数，步长和范围必须固定，若在循环体内不固定，综合工具不知道需要执行的次数，无法决定需要多少电路来完成。循环体内不能存在事件驱动语句@event。
- 条件逻辑所罗列的状态要全部表示。（比如要在所有的情况下都要赋一个值给一个变量或者信号）。若 case 语句所罗列的状态并没有表示出 case 表达式的所有状态时，此时，应加入 default 语句。同样对于 if－else 语句，如果只有 if 而没有配套的 else 也会出现生成锁存器的错误。
- 不允许采用 initial 语句，在循环语句中也不用 disable 语句
- 等待语句和延迟语句是不允许的

➢ 函数和任务的可综合性

- 对于函数和任务，考虑到它们对于综合带来的复杂性，建议能不使用尽量不要使用。

➢ 仿真语句的可综合性

- 延迟语句是典型的面向仿真的，综合工具不支持。
- 初始化语句 initial 也是面向仿真的。综合工具不支持。
- 若需要初始化参数，应该在有限状态机中加入判别有无 reset 信号的状态来实现。

➢ 端口连接的可综合性

- 在端口连接的时候不允许有表达式。
- 所有的模块的输出都应该连上，不管有没有用到。

➢ 综合前后仿真结果不一样的原因

- 组合逻辑的敏感表要全
- 避免“竞真”状况的产生

解决方法：a. 将两个事件放在同一 block 中

b. 使用非阻塞赋值语句

- 建议不要使用“casex”语句
- 时序逻辑中对同一个变量的赋值应该在同一个 always 块中
- 在 task 中不要使用非阻塞赋值语句
- 避免时序不匹配

10.6 综合结果最优化性设计

liupeng@zju.edu.cn

- 连续/过程赋值语句，阻塞/非阻塞

  原则：

  - 在同一个过程赋值语句里面，不要混用阻塞赋值和非阻塞语句。
  - 对简单的组合逻辑使用连续赋值语句，复杂的组合逻辑使用过程赋值语句。
  - 使用过程赋值语句和非阻塞赋值描述时序逻辑，并且首先描述异步逻辑，后面描述同步逻辑。
  - 对锁存器（latch）描述时，使用阻塞赋值。
  - 使用过程赋值语句描述组合逻辑时，使用阻塞赋值。
  - 建议不要在同一个 always 块中同时描述时序逻辑和组合逻辑。
- 避免在组合逻辑中产生不必要的锁存器
  - 一个简单而且是良好的设计方法是对变量定义缺省值，这样综合将产生一个纯组合逻辑电路。
- 避免产生不必要的优先级逻辑
  - 有优先级逻辑的判断，这种逻辑电路无论从面积还是时延上来讲都远远不如并行逻辑，所谓并行判断指的是在一个 case 语句中所有的判断条件都不重合。书写 case 语句的要点：通过直接书写或者利用缺省情况覆盖所有的输入模式，如果可能，设定输出的值为“x”，给以综合工具更大的优化空间。
- 利用资源共享减少资源的占用
  - 采用硬件代价小的单元（通常为多项选择器）省下代价大的单元，就可以减少资源的占用了。
- 使用摩尔状态机而不是米里状态机
- 单独书写状态机

  写状态机时，有两种好的编程风格可以参考：
  - 两个 always 块型，一个时序的 always 块用来同步更新状态寄存器，另一个组合的 always 块计算下一个状态和输出逻辑。
  - 三个 always 块型，一个时序的 always 块用来同步更新状态寄存器，一个组合的 always 块计算下一个状态，最后一个用来计算输出逻辑。
- 分开书写随机逻辑与机构化逻辑模块
- 所有的顶层模块都是寄存器输出
- 保持顶层模块的连线简洁
  - 最好不能有孤立的逻辑单元。
- 对于常量数字写明位宽和进制
- 逻辑层模块划分
  - 逻辑层模块划分和物理层划分结构一致，建议不要超过 5－10 层，最小的模块不小于 1000－5000 门，尽量保持一个模块 4－5 万门左右。
- 清楚的知道所描述的电路的结构

liupeng@zju.edu.cn