代码风格与规则

1.变量与参数命名

1) 不用语言的关键词命名变量与参数

这是废话，就是我同意您老这么变态，编译也不会教您好受的。

2) 命名尽量体现信号的特点

很多人不喜欢长的名字，不是到是键盘输入不好还是懒。贫僧建议还是长点的名字更利于以后的阅读。例如，一个同步信号名字叫“syn”看起来简单呢，还是“synchronization_1sec”看起来容易理解？

3) 常数，尤其是状态机的状态面，用宏表示

代码里面的常数，在后期阅读中绝对是埋下的地雷，需要专门的注释或者处理。在做状态机的时候的状态名称是自己规定的，这时候宏就是一个很好的机制。“if state == STATE_IDLE”远比“if state= = 4’b0000”容易理解。

4) 所有定义必须有说明

这些说明内容需要尽可能详细，尤其是位宽、速度这些信息一定要有，以方便以后重用。我们前面各讲都说了，系统内部参数的位宽，会影响结构的设计。很多程序员为了以后的重用，又喜欢吧位宽定义为参数。这样问题来了，原来 8 比特的代码，到了 128 比特能重用的可能性有多少呢？所以，一定要好好说明，防止被误用。

5) 宏、参数和变量等通过名称里面的大小写区别，并且严格执行

这是防止例如给宏赋值之类的错误，也方便代码的阅读。

6) 用 n 结尾的下横线表示低电平有效。例如： reset_memory_n, reset_n

7) 每一个端口声明必须有一个描述性的注释，最好是位于同一行。如果注释不是与被注释的代码位于同一行，那么注释应位于被注释代码的前面

2.代码与注释

1) 模块开头必须加详细说明，介绍输入、输出、功能以及参考文件

2) 注释必须有一定的比例

很多人不重视写注释，认为不是有效代码，敲起来还老长。实际上，很大程度上，注释就是你为以后自己阅读代码留下的指路标记，用来防止后面你自己都不认识自己的代码了。

3) 注释应尽量靠近被注释的语句

4) 首格缩进一定要对齐

这也是为了方便阅读，首格缩进不一致很容易造成阅读困难

5) 每一行最多只能有一个 Verilog 语句

6) 一个 always 模块里面只操作一个 reg 变量或者一组相关的 reg 变量

以前强调过，我们设计是“看图说话”，所以这个要求不难达到。那些说“臣妾做不到”的，肯定是不自信、心急的，不画图就写代码了。这个风格实在是问题多多。

7) “if”语句层次不能太多

“if”就是比较器，多层的“if”就是串联的比较器，这或造成组合逻辑在时间上的灾难。

8) 分层设计，每个模块的功能必须单一

自顶向下，分层设计是被实践证明最有效的工程流程，需要遵守。

9) 分区时钟生成电路应位于单独的模块

10) reg 型不能在两个分离的 always 块中赋值

11) 复杂的等式要使用圆括号

12) 每个 always 敏感信号列表对应一个时钟

13) 可综合代码中循环的次数必须是有限的

14) 端口连接时不允许用表达式

3.调试与纪录

1) 在修改程序时，不要直接修改原来部分，最好注释掉原来语句，在重写修改部分

一般规律，改错比改正容易。所以保留原来的部分，实在不成还能恢复，否则可能是欲哭无泪。

2) 认真纪录模块注释的“历史”部分

列宁说：“忘记历史就意味着背叛”，贫僧曰：“纪录历史，防止二过”。

3) 如果有很多函数要修正，请一个一个地作，修正一个函数检查一个函数

一次修改多处，出了问题都不知道到底是哪里的问题了。

4.设计风格

1) 严格采用同步设计，异步时钟部分必须隔离。同一时钟域内，所有模块中触发一致

2) 所有寄存器的输出信号都能被复位/置位

3) 遵守先写后读，否则要清醒地剔除掉无效数据

4) 每个模块寄存器入寄存器出

5) 相信综合工具，bug 绝对是自己搞的，别瞎怀疑

6) 嵌入式存储器中使用 BIST

7) 如果时间充裕，通过时钟做一个多触发器来取代用 MUX

8) 芯片内部无需“三态”

9) 一定要做时序仿真

10) 一个计算式中，排列每个信号的位数

11) 不要使用语言里面的复杂逻辑。例如：除法器和求模运算

12) 任何变量不能赋初始值 X，对任何寄存器所赋的初始值必须是确定的

13) 代码语句中不能加时间延迟，不允许出现如下语句： #4 out = cin;

14) 不允许使用门控时钟和门控复位

15) 不允许使用锁存器。（商用综合器可帮助执行这项规范，如使用了锁存器，综合结果会出现 Latch inferred）

16) 不允许在可综合的设计代码中使用`Define 来定义参数, 应该使用参数 Parameter 来定义。`Define 只用于编写不可综合的仿真测试模块

17) 不允许在可综合代码中使用 wait、fork-join、while

18) 不允许使用 UDP（用户定义的原语元件）

19) 不允许在可综合代码中出现逻辑反馈环路，否则会生成不可预知的逻辑电路

20) 不建议在可综合模块中使用 casex, casez 语句，只允许使用 case 和 if-else 语句作条件分支语句

21) 在时序逻辑的设计中不允许用阻塞赋值 = ，只允许用非阻塞赋值 <=

22) 在组合逻辑的设计中而不允许使用非阻塞赋值 <= ，只允许用阻塞赋值 =

23) 只允许用 Verilog2001 中的 always (*) 来生成组合逻辑，其他 Verilog 2001 语法在可综合模块中目前都不建议采用