Verilog 编码规范 From：项目管理部
文档编号：AM-PMD038

本规范规定了 Verilog 编码规范，

即采用 Verilog 设计时的代码书写规范，

本规范适用于逻辑芯片开发中使用 Verilog 语言作为

RTL 级设计语言

电路描述规则（注释部分）

关 键 词：Verilog HDL、注释

摘　　要：本文档规定了在用 Verilog HDL 描述电路时注释要求。

一、　在使用 Verilog HDL 描述电路时，为了增加电路的可读性，必须在电路中加入注释。为了统一和规范设计，制定该规则。

二、　Verilog HDL 模块结构：

在用 Verilog HDL 描述电路时，基本结构如下：

```
{注释 1}
module [模块名（端口名列表）]
[参数定义] //{注释 2}
[端口类型说明]//{注释 3}
[数据类型说明]//{注释 4}
{注释 5}
[描述体部]//{注释 6}
endmodule
```

三、　各部分注释具体要求：

1. 注释 1：
   - 在 module 语句以前；
   - 建立时间和设计人；

- 修改时间和修改人列表；
- 描述模块的功能；
- 仿真文件名；

2. 注释 2：
   - 参数定义以后。每行只能定义一个参数；
   - 参数的含义；
   - 正常情况的取值；

3. 注释 3：
   - 端口类型说明以后。每行只能说明一个端口；
   - 端口的信号含义；

4. 注释 4：
   - 数据类型说明以后。每行只能说明一个数据；
   - 数据的具体含义；
   - 修改时间和修改人列表；

5. 注释 5：
   - 在每个 always 前；
   - 描述该块语句完成的功能；

6. 注释 6：
   - 关键的判断语句后；
   - 简单描述语句的功能；

四、 说明：

1. 不得使用中文注释；
2. 需要时使用参数化设计；
3. 模块、端口和变量命名尽可能统一，且意义明确；
4. 各描述体功能尽可能明确和单一；
5. 对任何需存档的修改必须记录在设计文挡中。

电路描述规则（代码部分）

大类	编号	规则要素
设计风格	1	低电平有效的信号，信号名后缀“ _n ”
	2	模块名小写 // 对 AM0202 不作要求。
	3	Module 例化名用 U_xx_x 标示（多次例化用次序号 0、1、2…）
	4	使用降序排列定义向量有效位顺序，最低位是 0
	5	采用小写字母定义 wire、reg 和 input/inout/output// 对 AM0202 不作要求。
	6	采用小写字母定义参数，参数名小于 20 个字母// 对 AM0202 不作要求。
	7	时钟信号应前缀“ clk ”，复位信号应前缀“ rst ” // 对 AM0202 不作要求。
	8	三态输出的寄存器信号应后缀“ _z ”
	9	代码中不能使用 VHDL 保留字，更不能使用 verilog 保留字
	10	输出信号尽可能被寄存（只对顶层模块）
	11	三态逻辑可以在顶层模块使用，子模块中避免使用三态
	12	到其它模块的接口信号，按如下顺序定义端口信号：输入、（双向）、输出
	13	建议 Coregen 生成的乘法电路
	14	采用基于名字（name_based）的调用而非基于顺序（order_based）的调用
	15	时钟事件的表达式要用“ negedge<clk_name> ”或“ posedge<clock_name> ”的形式// 对 AM0202 不作要求。
	16	异步复位，高电平有效用“ if(<as_reset>==1’b1) ”，低电平有效用 if(<as_reset>==1’b0) ”
	17	If 语句嵌套不能太多
	18	建议不使用 include 语句// 对 AM0202 不作要求。
	19	建议每个模块加 timescale
	20	代码中给出必要的注释
	21	每个文件有一个文件头
	22	每个文件只饮食一个模块//?
	23	模块名和文件名保持一致
设计可靠性	24	同步时序逻辑的 always block 中有且只有一个时钟信号，并且在同一个沿动作（如上升沿）
	25	同步时序逻辑的 module 中，在时钟信号的同一个沿动作
	26	尽量避免使用异步逻辑（全局异步复位除外）
	27	一般不要将时钟信号作为数据信号输入
	28	不要在时钟路径上添加任何 buffer
	29	尽量不使用门控时钟
	30	在顶层模块中，时钟信号必须可见
	31	不要采用向量的方式定义一组时钟信号
	32	时钟产生电路，单独成一模块
	33	建议使用单一的全局同步复位电路或者单一的全部异步复位电路
	34	不要在复位路径上添加任何 buffer，也不要使用任何门控复位信号
	35	不使用 PLI 函数

	36	建议不使用任务
	37	不使用事件变量
	38	不使用系统函数
	39	不使用用户自定义单元（UDP）
	40	不使用 disable 语句
	41	不使用==、！===等不可综合的运算符//应该是!= =
	42	建议不使用 forever、repeat、while 等循环语句//通常不要使用这些函数
	43	避免产生 Latch(除 CPU 接口外)//有时候不自觉得就产生了锁存，但是要注意！
	44	组合逻辑语句块敏感表中的敏感变量必须和该块中使用的相一致，不能多也不能少
	45	在一个 always 语句中有且只能有一个事件列表
	46	在 always 块的敏感事件列表中必须都是沿触发事件，不允许出现电平触发事件//最好这样，不让有时候会出问题。
	47	数据位宽要匹配
	48	不使用 real、time、realtime 类型
	49	建议不使用 integer 类型
	50	移位变量必须是一个常数
	51	避免使用异步反馈环路
	52	时序逻辑语句块中统一使用非阻塞型赋值
	53	组合逻辑语句块中使用阻塞型赋值
	54	非阻塞型赋值应不加单位延时，尤其是对于寄存器类型的变量赋值时
不常用规则	55	整型常量基数格式中不能有“ ？ ”
	56	字符串中不能含有控制字符（如 CTRL 键等）
	57	禁止使用空的时序电路块及非法的 always 结构
	58	不要在边疆赋值语句中引入驱动强度和延时
	59	不要为 net、n_input、n_output、enable_gate 型变量定义驱动强度、电荷保持强度以及延时
	60	禁止使用 trireg（具有电荷保持特性连接）NET 型定义
	61	禁止使用 tri1、tri0、triand 和 trior 型的连接（net）
	62	在 RTL 级代码中不能含有 initial 结构，也不可以对任何信号进行初始化赋值，而应采用复位的方式进行初始化
	63	不要在过程语句中使用 assign、deassign、force、release 等语句
	64	不要使用 wait 语句
	65	不要使用 fork-join 语句块
	66	不要为驱动类型为 supply0 和 supply1 型的连线（net）赋值
	67	设计中不使用 macro_module
	68	不要在 RTL 代码中实例门级单元，尤其是下列单元： CMOS 开关、RCMOS 开关、NMOS 开关、PMOS 开关、RNMOS 开关、RPMOS 开关、trans 双向开关、rtrans 双向开关、tranif0、tranif1、rtranif0、rtranif1、pull_gate
	69	不要使用 specify 模块