上海交通大学数字系统芯片实验室
Verilog HDL 代码规则

版本	V1
作者	蒋剑飞
邮箱	Jiangjianfei@sjtu.edu.cn
联系电话	13774201676
修改时间	2022-2-22

1 概述

1.1 前言

本文档属于上海交通大学数字系统及芯片研究设计中心内部文档。

1.2 目的

为规范 Verilog HDL 编码风格，提高 Verilog RTL 代码的可读性、可维护性和可综合性，减少因编码风格出现的仿真和综合风险，制定此编码规范。

1.3 代码规则分类

表 1-1 规则分类

规则类型	符号	说明
强制规则	【M】	该规则需要强制执行
推荐规则	【R】	该规则建议执行

1.4 其他

本文档包括部分示例代码，以提供必要的参考。

2 命名规则

2.1 规则列表

NM01【M】一个 Verilog 文件只包含一个模块。

NM02【M】文件名(不包含.v)与模块名称必须相同。

NM03【M】模块名称字母全部小写。

NM04【M】信号名称字母全部小写，且以字母开头。

NM05【M】低电平有效的信号以_n 结尾。

NM06【M】parameter、localparam 名称必须全部大写。

NM07【M】 define 定义的宏名称必须全部大写。同一设计中多次调用同一模块（define 不同），会产生编译顺序要求，容易引起混乱，所以应尽量避免使用 define。

NM08【M】模块名、信号名、参数名等不能使用 Verilog/System Verilog 的关键字。

NR01【R】宏定义集中存放，文件名以 xxxx_defines.v。

NR02【R】模块名、信号名、参数名等必须有一定的意义，且单词间以下划线隔开。例如，不允许随便定义 xxx_tmp1，xxx_tmp2 等命名。

NR03【R】对于不需要通过模块实例化来改变参数值的参数，建议采用 localparam 定义，例如，状态机状态参数。

NR04【R】模块实例化的名字建议采用 i_模块名+后缀的方式组成。

2.2 Examples

2.3 常用信号名缩写

名称	缩写	名称	缩写
clock	clk	reset	rst
address	addr	pointer	ptr
valid	vld	ready	rdy
request	req	acknowledge	ack
enable	en	delay	dly
counter	cnt	control ctrl	ctrl
select	sel	start	strt
response	rsp	packet	pkt
transmiter	tx	receiver	rx/rcv
read	rd	read_enable	rd_en
write	wr	write_enable	wr_en

3 头文件和注释

3.1 规则列表

HM01【M】采用//行注释，除了头部信息，禁止采用/* */注释方式。

HM02【M】使用宏`ifdef, `else, `endif 条件分支必须添加注释。

HM03【M】文件开头统一添加 timescale 设置。

HM04【M】文件头统一使用实验室模板(参考 examples)

HR01 【R】代码中应当添加有意义的注释，对实现的功能或算法等进行说明，以提高代码的可读性。

3.2 Examples

3.2.1 HM04

```
//=====================================================================
/// Description:
// The Core module to implement the core portion of the cpu
// Designer : A@sjtu.edu.cn, B@sjtu.edu.cn
//Revision History:
//V0 date:xxx Initial version，A@sjtu.edu.cn
//V1 data:xxx Modify xxx by    B@sjtu.edu.cn
// =====================================================================
```

4 编码风格

4.1 规则列表

CM01 【M】不使用 TAB 缩进，设置以 4 个空格来代替 TAB 键。
CM02 【M】同一个信号只允许在一个 always 块中进行赋值。
CM03 【M】时序逻辑采用 non-blocking 赋值。
CM04 【M】组合逻辑采用 blocking 赋值。
CM05 【M】敏感列表采用 always@(*) 的方式来指明。
CM06 【M】左侧、右侧表达式的比特位宽必须一致。常数必须指明位宽。
CM07 【M】禁止’ ；’分隔的多个表达式出现在代码的同一行。
CM08 【M】内部信号必须显式地声明，且所有声明必须位于模块的开头部分。
CM09 【M】条件表达式的结果必须为 1bit 的位宽。
CM10 【M】避免出现 Latch，always 块种组合逻辑所有条件都必须赋值（包括使用 default）。
CM11 【M】不允许内部的三态逻辑。
CM12 【M】 FSM 状态机组合逻辑与时序逻辑分开到两个 always 块，对组合逻辑的判断分支必须补全，且不得出现将变量自身赋给自身的情况。
CM13 【M】FSM 状态机必须有 default 状态(参考 examples)。
CM14 【M】 模块实例化必须以端口名称进行关联。
CM15 【M】 模块实例化端口与信号位宽必须匹配。
CM16 【M】 Clock、Reset 放到单独的模块进行统一处理。
CM17 【M】 单 bit 跨时钟域处理至少需要同步 2 拍以上，且 2 拍之后的信号才能使用。

CR01 【R】 复杂表达式采用()分隔，显式地表示优先级。
CR02 【R】 使用 Verilog-2001 的语法进
CR03 【R】在一个 always 块中只对一个信号进行赋值，且必须加上最外层的 begin…end，与 always 对齐排列。例外，当一组信号具有相同的时序时，可在一个 always 块中对这组信号同时赋值。
CR04 【R】顶层模块应仅包含子模块的实例化，没有其它逻辑。
CR05 【R】信号在不同层次间传递时名称应尽量保持一致。
CR06 【R】优先考虑同步设计。
CR07 【R】多 bit 跨时钟域的处理优先采用 IP 公共模块，如异步 FIFO。
CR08 【R】 简单组合逻辑建议采用数据流模型：以 assign 语句实现，
复杂组合逻辑建议采用行为级模型：以 always@(*) 块方式实现。
CR09 【R】对于重复使用的组合逻辑可提炼成一个 function，function 调用可提高代码的可读性
CR10 【R】模块的 reset 策略需要一致，选择使用异步 reset 或者选择同步 reset，建议不要混用
CR11 【R】 端口风格采用统一格式，在端口定义 input/output (参考 examples)
CR12 【R】 assign 语句不缩进，always 过程块缩进规则：begin…end 不缩进，其他信号缩进 4 字符(参考 example CM17)

4.2 Examples

4.2.1 CM12/13

```
//Sequential logic:
always @(posedge clk or negedge reset_b)
begin:COMB
    if(~reset_b)
        mealy_fsm_state <= ST0;
    else
        mealy_fsm_state <= mealy_fsm_state_nx;
end

//Combinational logic:
always @(mealy_fsm_state or a)
begin:SEQ
    case(mealy_fsm_state)
    ST0:
        begin
            z <= a ? 1'b1 : 1'b0;
            mealy_fsm_state_nx <= a ? ST2 : ST0;
        end
    ST1:
        begin
            z <= a ? 1'b1 : 1'b0;
            mealy_fsm_state_nx <= a ? ST0 : ST1;
        end
    ST2:
        begin
            z <= 1'b0;
            mealy_fsm_state_nx <= a ? ST1 : ST2;
        end
    default:
        begin
            z <= 1'b0;
            mealy_fsm_state_nx <= ST0;
        end
    endcase
end
```

4.2.2 CM17

```
always @(posedge clk or negedge rst_n)
begin:Block_name
    if(~rst_n)
    begin
        sig_d1<=1'b0;
        sig_d2<=1'b0;
    end
    else
    begin
        sig_d1<=sig_in;
        sig_d2<=sig_d1;
    end
end
```

4.2.3 CR11

```
module e203_lsu(
  input  commit_mret,
  input  commit_trap,
  input  excp_active,
  output  lsu_active,

  `ifdef E203_HAS_ITCM //{
  input [`E203_ADDR_SIZE-1:0] itcm_region_indic,
  `endif//}
  `ifdef E203_HAS_DTCM //{
  input [`E203_ADDR_SIZE-1:0] dtcm_region_indic,
  `endif//}
  ////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////
  // The LSU Write-Back Interface
  output lsu_o_valid, // Handshake valid
  input  lsu_o_ready, // Handshake ready
  output [`E203_XLEN-1:0] lsu_o_wbck_wdat,
  output [`E203_ITAG_WIDTH -1:0] lsu_o_wbck_itag,
  output lsu_o_wbck_err ,
  output lsu_o_cmt_ld,
  output lsu_o_cmt_st,
  output [`E203_ADDR_SIZE -1:0] lsu_o_cmt_badaddr,
  output lsu_o_cmt_buserr , // The bus-error exception generated


  ////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////
  // The AGU ICB Interface to LSU-ctrl
  //    * Bus cmd channel
  input                          agu_icb_cmd_valid, // Handshake valid
  output                         agu_icb_cmd_ready, // Handshake ready
  input  [`E203_ADDR_SIZE-1:0]   agu_icb_cmd_addr, // Bus transaction start addr
  input                          agu_icb_cmd_read,   // Read or write
  input  [`E203_XLEN-1:0]        agu_icb_cmd_wdata,
  input  [`E203_XLEN/8-1:0]      agu_icb_cmd_wmask,
  input                          agu_icb_cmd_lock,
  input                          agu_icb_cmd_excl,
  input  [1:0]                   agu_icb_cmd_size,
```

5 参考文献

[1] pango_Verilog HDL 编码规范.