1. Verilog语法

Verilog文件基本结构

模块声明 模块名(端口列表)

a.

Module mux2(a,b,sel,out,io);

//端口属性定义

inout [7:0] a;

inout [7:0] b;

inout sel; //sel =0,out输出a; sel =1,out输出b

output [7:0] out;

inout io;

wire oe;//定义内部信号：时序逻辑用寄存器类型reg型，组合逻辑用线网类型wire型;但是 always块内赋值变量必须是reg

//连续赋值语句

//assign out = (sel==0)?a:b;

//assign out = !sel?a:b;

assign out = sel?b:a;

//三态门控制

assign oe = sel;

assign io = oe?out[0]:1’bz;// oe 为1输出为out[0]，oe 为0为高组态z

Endmodule

b.另一种定义方法

Module mux2(

//端口属性定义

inout [7:0] a,

inout [7:0] b,

inout sel,

output [7:0] out,

inout io

};

endmodule

1. 组合逻辑，Verilog位操作

1、取某一位直接用作数据源

wire [2:0] m;

assign m = out[5:3];

2、移位循环

reg [7:0] shift\_a;

always@(posedge clk)

shift\_a <= { shift\_a [0],shift\_a[7:1]};//循环右移

reg [7:0] shift\_a;

wire data;

always@(posedge clk)

shift\_a <= { shift\_a[6:0],data};//左移位

3、位拼接

wire [3:0]x;

wire [3:0]y;

wire [3:0]z;

wire [31:0]n;

assign z={x,y};//用花括号进行位拼接

assign n={y,7{x}};//一个y和7个x拼接

assign n={y,7{1‘b1}};//一个y和7个1拼接

4、位宽表示

assign x = 4’b1001//四位二进制

assign x = 4’d9//四位十进制

assign x = 4’hc//四位十六进制

assign n = 32’h1234\_abcd//32位十六进制用下划线分开

assign x = 8’b1001\_1101//8位二进制用下划线分开

5、位运算，加(+)减(-)乘(\*)除(/)模(%)，除法考虑效率用IP核或自己写，如移位

6、逻辑运算，与(&&)，或(||)，非(!)，按位与(&)，按位或(|)，按位取反(~)，按位异或(^)，按位同或(^~)

a=4’b1011;b=4’b0000;wire c;c=a&&b;//c=0

a=4’b1011;b=4’b0110;wire [3:0]c;c=a&b;//c=4’b0010

1. 时序逻辑，需要有时钟信号，复位信号，使能信号

移位寄存器（D触发器），计数器等

module counter(

clk,

en,

clr,

cnt\_value,

c

);

input clk;//时钟信号

input en;//使能信号

input clr;//清零信号

output [3:0]cnt\_value;

output reg c;

reg [3:0] cnt;

always@(posedge clk , posedge clr)//在clr上升沿触发，**异步清零**要写在always中

if (clr)//**同步清零**要受时钟信号控制才能清零，**异步清零**由自己的边沿控制

cnt <= 4'b0;//<=是非阻塞赋值//复位逻辑，初始化变量

else if(en) begin

cnt<= cnt +1'b1;

end

assign cnt\_value = cnt;

always@(posedge clk)//

if (cnt <= 2)//

c<=0;//c在always中赋值，类型要是reg型

else

c<=1;

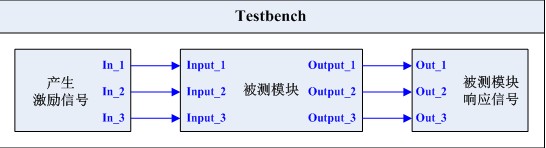
endmodule

1. 在always语句块中，verilog语言支持两种类型的赋值：阻塞赋值和非阻塞赋值。阻塞赋值使用“=”语句；非阻塞赋值使用“<=”语句。注意，千万不要将这两种赋值方法与assign赋值语句混淆起来，assign赋值语句根本不允许出现在always语句块中。
2. 不要在不同的always块内为同一个变量赋值。即某个信号出现在<=或=左边时，只能在一个always块内。所以注意，在产生一个信号时，所有产生该信号的条件都应放在一个always块内考虑。
3. 任何在always块内被赋值的变量都必须是寄存器型（reg）。即<=或=左边的信号，必须是reg型，<=或=右边的信号可以是reg型也可以是wire型。
4. 使用always块描述组合逻辑时使用阻塞赋值（=），在使用always块描述时序逻辑时使用非阻塞赋值（<=）。简单理解可以是，在电平敏感的always块内使用阻塞赋值，在边沿敏感的always块内使用非阻塞赋值。非阻塞赋值（<=）在always结束后才把值给左边的寄存器。
5. 模块名和工程名默认是相同的，如果不同，可以在”Assignments”->”Setting…”窗口General项目中Top –level entity输入顶层的模块名。
6. FPGA程序下载称为配置，配置方法：a.使用JTAG直接配置.sof文件下载到FPGA的RAM中，数据掉电丢失；b.使用外部的配置器件（FLASH）进行配置；c.使用外部的控制器进行配置；
7. testbench编写

`timescale 单位时间/时间精确度

module testbench; //定义一个没有输入输出的module

与待测模块接口:与输入端口相连接的变量定义为**reg**；与输出端口相连的定义为**wire**；如果是双向信号(inout)，在编写testbench时要注意。需要一个reg变量来表示其输入，还需要一个wire变量表示其输出。



被测模块名 例化进Testbench后的模块名//可以例化多个

(

．被测模块输入口(Testbench产生的激励信号),

．被测模块输出口(Testbench里用来显示输出的信号)

);

必须用initial初始化

用$stop或$finish暂停或结束仿真

wait(z==1’b1);//等待变量值改变，变量可以是待测试模块的输出或者内部变量

时钟产生： initial clock = 0;

always # 10 clk =~clk;//产生时钟

initial repeat(13) #5 clk =~clk; //控制只产生13个时钟。

initial begin

clk = 0; //初始值clk=0

forever #20 clk = ~clk; //每经过20个单位时间，clk取反

end// forever只在initial中使用

a = $random`; //产生-59~59之间随机数

a = {$random}`; //产生0~59之间随机数

输出显示：$monitor 输出打印显示;

Task和module区别：

1，task只能定义在module内部，不能单独在一个文件中，不能定义在module外面。

2，在task调用的是必须在过程性语句内部使用，initial，begin ... end

3，task可以没有参数，直接使用全局变量来实现功能。

4，可以使用延时控制，可以调用其他task和函数

在testbench中，必须指定不同信号有效和无效或等待某事件或条件的时间。有三种时序控制结构：

时延控制：#[delay\_time] [statement\_to\_get\_data];

事件控制：@([event], [event], …] [statement\_to\_get\_data];

等待语句：wait([boolean\_expression]) [statement\_to\_get\_data];

#10 a=1'b0;//a于t+10时刻得到0值；

#5 y=a|b;//又过了5个时间单位后a|b表达式被计算，其结果被赋给y。

#10 a=1'b0;//a于t+10时刻得到0值；

y=#5 a|b;//a|b表达式被立即运算（即在t+10时刻），但其结果却在t+15时刻才赋给y。

1. 例化待仿真文件使用的调用方式如果采用显式例化，这种方式要求例化时信号顺  
   序需要与编写的文件顺序一致，且不能在一个激励文件中例化两次。这种方式容易  
   出错，且具有局限性，因此不推荐使用。
2. 掌握可综合风格的 Verilog 模块编程的六个原则， 可以为解决在综合后仿真中出现绝大多数的冒险竞争问题。  
   1) 时序电路建模时，用非阻塞赋值；  
   2) 锁存器电路建模时，用非阻塞赋值；  
   3) 用 always 块建立组合逻辑模型时，用阻塞赋值；  
   4) 在同一个 always 块中建立时序和组合逻辑电路时，用非阻塞赋值；  
   5) 在同一个 always 块中不要既用非阻塞赋值又用阻塞赋值；  
   6) 不要在一个以上的 always 块中为同一个变量赋值。
3. 编码原则很多，就阻塞非阻塞赋值而言，新手最需要牢记的是其中三条：

1、时序逻辑一定用非阻塞赋值“<=”,一旦看到敏感列表有 posedge 就用“<=”。

2、组合逻辑一定用“=” ，一旦敏感列表没有 posedge 就用“=”，一旦看到 assign 就用“=”。

3、时序逻辑和组合逻辑分成不同的模块，即一个 always 模块里面只能出现非阻塞赋值“<=”或者“=”。如果发现两种赋值并存，一个字“改”，心存侥幸可能会给后续工作带来更多麻烦。

1. 有限状态机分为摩尔（Moore）型有限状态机与米利（Mealy）型有限状态机。摩尔状态机输出是只由输入确定的有限状态机（不直接依赖于当前状态）。米利有限状态机的输出不止与其输入有关还与它的当前状态相关。
2. 状态机描述方式，可分为一段式、两段式以及三段式。  
   一段式，整个状态机写到一个 always 模块里面。 在该模块中既描述状态转移，又描述状态的输入和输出。  
   两段式，用两个 always 模块来描述状态机。 其中一个 always 模块采用同步时序描述状态转移， 另一个模块采用组合逻辑判断状态转移条件，描述状态转移规律及其输出。  
   三段式，在两个 always 模块描述方法基础上，使用三个 always 模块。 一个 always 模块采用同步时序描述状态转移，一个 always 采用组合逻辑判断状态转移条件，描述状态转移规律，另一个 always 模块描述状态输出(可以用组合电路输出，也可以时序电路输出)。
3. FPGA内部的逻辑延迟，需要进行后仿真。后仿真，时序仿真，带有布局布线信息的仿真。多路信号传输不可能同时到达，会出现信号翻转的不稳定状态，为了避免多位翻转的问题可以采用格雷码，每次只有一位发生翻转，可以提高时钟频率。
4. 独热码译码逻辑简单，传输延时小可以提高传输频率
5. 按键信号变化正好是时钟上升沿，触发器输出状态不定可能产生震荡，处于亚稳态，会影响后面的信号。需要将异步信号，用时钟进行同步，加2级或更多D触发器进行同步。

按键程序需要学习pedge\_key可靠吗，led闪烁程序需要学习，key\_react工程需要修改

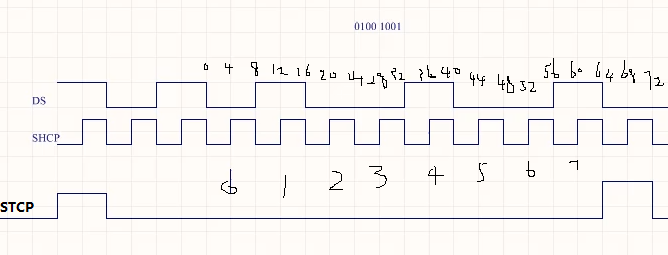
1. 调用模块内部的变量，module1.变量
2. 参数化设计

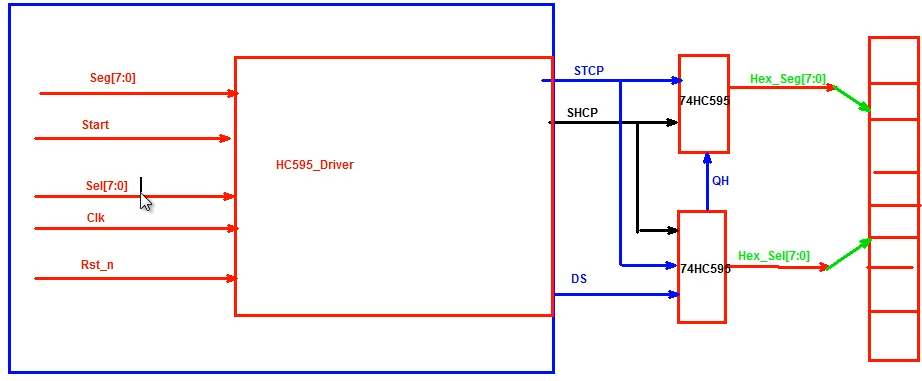
在模块中定义了参数parameter VAL = 32’d10000

在调用模块时可以修改参数:

1.例化时修改参数 #( .VAL(32’d1000))

2.程序中修改 defparam module1. VAL = 32’d1000

1. 



1. 自定义的时钟（门控时钟）无法保证到达每个逻辑单元时间的一致性，用门控时钟去驱动很多寄存器时钟质量无法保证。全局时钟可以保证到逻辑单元的时间相同。可以把自定义的信号作为使能信号。
2. 线性序列机（LSM）的设计思想就是使用一个计数器不断计数，由于每个计数值都会对应一个时间，那么当该时间符合需要操作信号的时刻时，就对该信号进行操作。这样，就能够轻松的设计出各种时序接口了。
3. In-system sources and probes editor

Sources：在PC端通过软件设置某个信号的值，该数值会经由USB Blaster传递到正在运行的FPGA芯片中相应的位置，FPGA在使用该信号作为激励

Probes：在FPGA内部使用Probes将待检测的某些信号的值读取并经由USB Blaster传送到PC端，在PC端的软件上显示这个值。

1. 真
2. 真
3. 真
4. 真
5. 真