浙江大学 2015－2016 学年 春夏 学期

《HDL 语言 》课程期末考试试卷

课程号：_____，开课学院：_____生仪学院_____

考试形式：闭卷 √、开卷（请在选定项上打 √），允许带_____入场

考试日期：_2016_____年_____月_____日，考试时间：__120_____分钟

诚信考试，沉着应考，杜绝违纪。

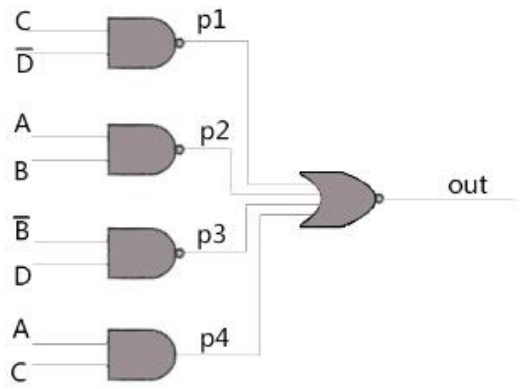
考生姓名：_____学号：_____所属院系：_____

题序	一	二	三	四	五	六	七	八	总 分
得分									
评卷人									

考生注意：本试卷共 11 张，3 道大题，分 19 道小题，请检查一下有无缺页。

一、 填空题（每空 2 分，共 30 分）

1、请补充如下图所示的门级电路的**结构描述**的 HDL 语言：




```
module DUT
    wire    p1,p2,p3,p4,out;
    reg     A,B,C,D;
    nand
    g1(p1,C,～D);
    g2(p2,A,B);
    g3(p3,～B,D);
    and g4(p4,A,C);
    nor g5(out,p1,p2,p3,p4);

end module
```

2、Verilog 有四种基本循环语句，包括 for、 forever、_____while_____repeat_____。

verilog 的本质是___事件（event）____触发的 HDL 语言。

3、状态机按照输出逻辑可以分为两种，一种是____Moore____，另一种是___Mealy_____，

二者的区别是：_Moore 型输出只与当前状态有关而 Mealy 型与输入和状态都有关_____。

4、状态机的状态变量一般采用的编码方式有____独热码、格雷码(or binary)___，

其各自的优点是_独热码可以减少组合逻辑使用量，少毛刺；格雷码状态变量位宽短____。

5、以下代码用于生成斐波纳挈数列，其语法上有两处错误，请指出。

①没有声明 event ready　　②temp 应该是 reg 变量

```
module top;
wire    [15:0]    num,num_out;
number_gen    ng      (num);
figNumCalc    fnc      (num,num_out);
endmodule

module number_gen
(output reg    [15:0]    num = 0);

always
      begin
            #50 num = num + 1;
            #50 -> ready;
      end
endmodule

module figNumCalc
(input        [15:0]    startingValue,
output reg [15:0]    fibNum);

reg [15:0] count,oldNum;
wire [15:0] temp;
always
      begin
            @ng.ready
            count = startingValue;
            oldNum = 1;
            for(fibNum = 0;count != 0; count = count - 1)
                  begin
                        temp = fibNum;
                        fibNum = fibNum + oldNum;
                        oldNum = temp;
                  end
      end
endmodule
```

6、以下代码是否有问题，如何优化？ 会生成锁存器，需要补全 case-default

```
module test(
input      a,
input      data_in,
output    data_out);
      always @(a or data_in)   begin
            case(a)
            0:    data_out <= data_in;
            endcase
      end
endmodule
```

7、以下两段代码的行为有什么不一样？ 前者上升沿敏感、后者电平敏感

```
...
@(posedge exp)
      #1 statement1;
...
```

```
...
wait(exp)
      #1 statement1;
...
```

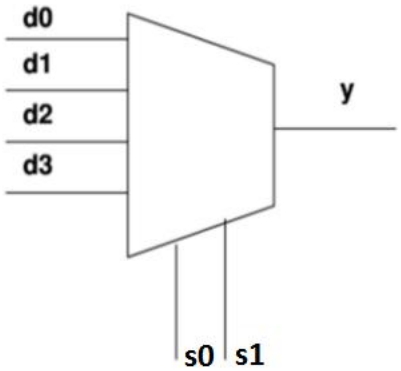
8、用 verilog 设计一个带异步复位、同步置位，时钟上升沿触发的 D 触发器。

```
module DFF(
input clk,
input clk_enable, //为高电平时 clk 输入有效
input d,
input set_n,
input rst_n,
output q);

    always@(posedge clk or negedge rst_n)
    begin
    if(!rst_n)
    q<= 1'b0;
    else if(!set_n and clk_enable)
    q<= 1'b1;
    else if(clk_enable)
    q<=d;
    end

endmodule
```

9、用行为描述设计以下四选一数据选择器，要求四路输入有相同优先级：




```
module mux_4_to_1(
module mux_4_to_1(
input d0,
input d1,
input d2,
input d3,
input s0,
input s1,
output y);

 bengin
 always @(s1 or s0)
       case({s1,s0})//synopsys parallel_case
       2'b00: y=d0;//上面综合选项可以不写
       2'b01: y=d1;
       2'b10: y=d2;
       2'b11: y=d3;
       default:y=0;
       endcase
 end

endmodule
```

二、 简述题（每题 5 分，共 25 分）

10、试简述阻塞赋值（=）和非阻塞赋值（<=）的区别。并解析下面两段代码在时序上有什么不同。

```
begin
      a = #50 1; b = #100 1; c= #150 1;
end
```

```
begin
      a <= #50 1; b <= #100 1; #100 c = 1;
end
```

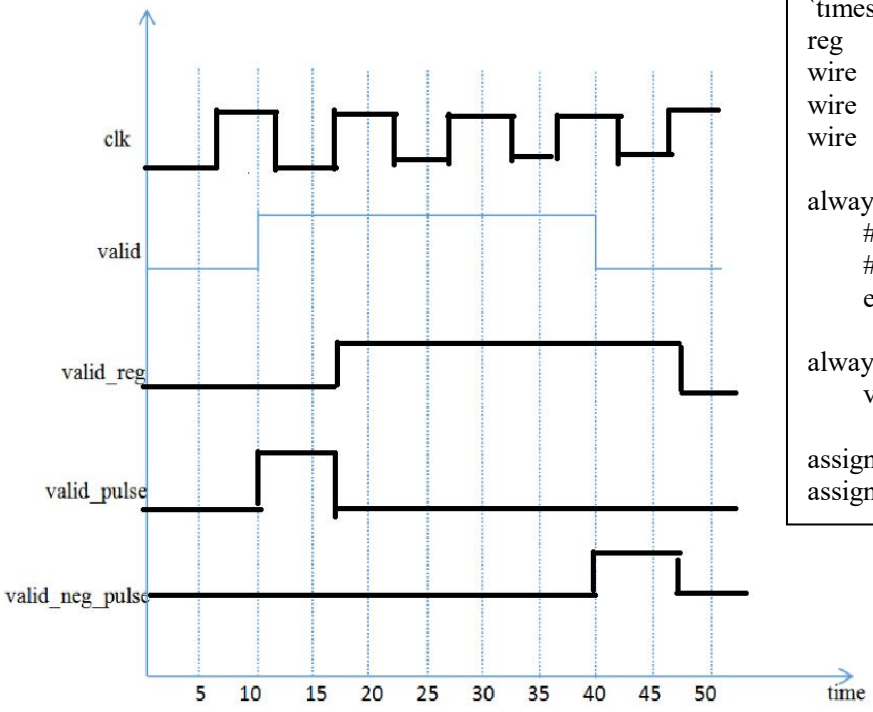
阻塞赋值是赋值结束后才允许其它语句的执行；

非阻塞赋值先计算等号右边的值，允许其它语句同时执行，最后同时更新等号左面的值。

前者 a 在 0ns 采样，50ns 赋值；b 在 50ns 采样，150ns 赋值；c 在 150ns 采样，300ns 赋值；

后者 a 在 0ns 采样，50ns 赋值；b 在 0ns 采样，100ns 赋值；c 在 100ns 采样并赋值。

11、根据已知的 valid 波形，画出下面程序中变量 clk、valid_reg、valid_pulse、valid_neg_pulse 的仿真波形(从 0ns 到 50ns)。




```
`timescale 1ns/1ns
reg     valid_reg;
wire    clk;
wire    valid_pulse;
wire    valid_neg_pulse;

always    begin
      #3    clk = 0;
      #7    clk = 1;
      end

always @(posedge clk)
      valid_reg <= valid;

assign valid_pulse = valid &(~valid_reg);
assign valid_neg_pulse = (~valid )&valid_reg;
```

*图中 clk 占空比为 30%，不好画就不画了

12、系统任务$monitor 与$display 有什么区别？$stop 与$finish 有什么区别？

$display 语句每次执行的时候打印输出；$monitor 的监测变量改变时打印输出；
$stop 命令表示暂停仿真，可以用 run 命令继续仿真，$finish 表示停止仿真。

13、always 语句和 initial 语句的关键区别是什么？能否相互嵌套？

always 可循环执行，initial 只执行 1 次，他们都不可相互嵌套。

14、用 function 实现计算一个 8 位变量的奇校验位（如果 1 的个数为偶数，奇校验位为 1），并简述 function 与 task 的区别(至少三点）。

```
function calc_parity;
input [7:0] value;
begin
      calc_parity = ~(^value);
end
endfunction
```

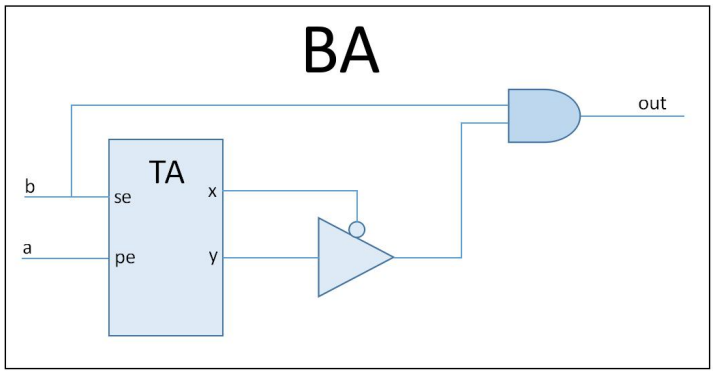
- task 可以自定仿真时间单位。
- 任务可以启动其它任务或函数。
- 任务可以没有输入变量。
- 任务可以不返回值。

三、 设计题（共 45 分）

15、TA 模块程序如下，写出引用 TA 模块实现 BA 模块的行为级描述 verilog 代码。（5 分）

```
module TA(
input se,
input pe,
output x,
output y);

assign {x,y} = se + pe;
endmodule
```




```
module BA(
input a,
input b,
output out);
wire x,y,e;
TA TA(.se(b),.pe(a),.x(x),.y(y));
assign e = x?y:z;
assign out = b&e;
endmodule
```

16、已知输出表达式如下：（10 分）

$$F（A,B,C,D）=\sum m(0,4,6,8,10,11)+\sum d(2,13,15)$$

要求：

- 画出相应的卡诺图；
- 写出输出的最简与或表达式；
- 用 verilog 实现输出表达式。

	AB				
CD		00	01	11	10
	00	1	1	0	1
	01	0	0	x	0
	11	0	0	x	1
	10	x	1	0	1

$$F(A,B,C,D)=\overline{A}\overline{D}+\overline{B}\overline{D}+A\overline{B}C$$

```
module out(
        input A,
        input B,
        input C,
        input D,
        output F);

        assign F = (~A)&(~D)+(~B)&(~D)+A&(~B)&C;

endmodule
```

17、设计一个同步系统实现的秒杀仲裁电路，2 个用户参与秒杀，模块定义如下：（10 分）

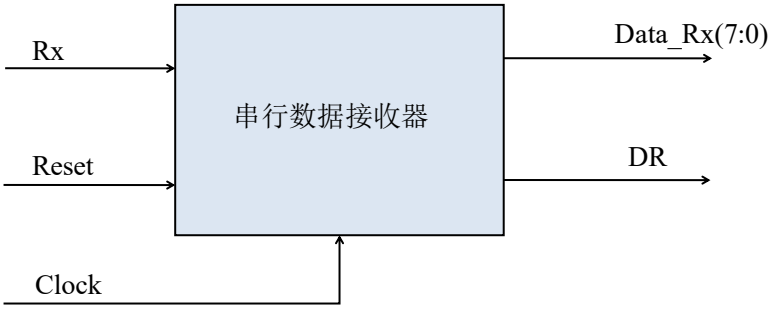
要求：

- 如果用户在开始秒杀前请求秒杀，则失去秒杀资格；
- 开始秒杀后，第一个输入秒杀请求的用户秒杀成功，其余失败；
- 如果两个秒杀请求同时到达，则用户 1 秒杀成功；
- 可能没有用户秒杀成功。

```
module SecKill(
input clock,
input req1,
input req2,
input start,
input clear,
output out1,
output out2);

reg flag1,flag2;
always @(posedge clock) begin
    if(clear == 1) begin
        flag1 = 0;
        flag2 = 0;
        out1 = 0;
        out2 = 0;
    end
    else;
    if(start == 0) begin
        if(req1 == 1) flag1 = 1;
        else flag1 = flag1;
        if(req2 == 1) flag2 = 1;
        else flag2 = flag2;
        end
    else begin
        if (req1 == 1 && flag1 == 0) begin
            out1 = 1;
            flag2 = 1;
        end
        else  if(req2 == 1&& flag2 == 0) begin
            out2 = 1;
            flag1 = 1;
        end
    else;
    end
end
endmodule
```

18、请用 verilog 语言实现一个串行数据接收电路，接收串行数据并输出并行数据，其框图如下，不要求测试文件。（10 分）



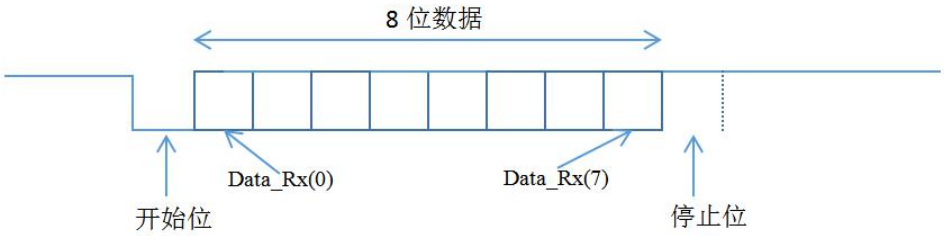
该电路时钟频率为串行数据传输波特率的 16 倍。产生一个 8 位并行输出，和一个状态输出(DR)，当 1 字节的数据接收完毕时，该状态输出一个时钟周期的逻辑 1。Rx 输入端的数据格式如下：

① 1 位起始位（逻辑 0）；

② 8 位传输数据位（低位在前、高位在后）；

③ 1 为停止位（逻辑 1）；

④ 无奇偶校验。



当接收电路等待数据时，Rx 为逻辑 1。当 Rx 为逻辑 0 时代表起始位的指示（由外部电路产生）。然后传输 8 位数据位，最后用 1 位停止位（逻辑 1）代表传输结束。

```
module Uart_Receiver(
input Rx,
input clock,
input Reset,
output reg [7:0] Data_Rx,
output reg DR);

reg [7:0] count;
reg [7:0] Data_Init;

always @(posedge clock) begin
      if(Reset == 0) count <= 8'b00000000;
      else if (Rx == 1 && (count == 8'b00000000 or count == 8'b 10101011)
            count == 8'b00000000;
      else  count <= count +1;
```

```
always @(posedge clock) begin
	if(Reset == 0) begin
		Data_Init <= 8'b00000000;
		Data_Rx <= 8'b00000000;
		DR <= 0;
	end

	else begin
		if(count == 8'b00000000) DR <= 0;
		else;
		if(count == 8'b00000001) DR <= 0;
		else;
		if(count == 8'b00011000) Data_Init[0] <= Rx;
		else;
		if(count == 8'b00101000) Data_Init[1] <= Rx;
		else;
		if(count == 8'b00111000) Data_Init[2] <= Rx;
		else;
		if(count == 8'b01001000) Data_Init[3] <= Rx;
		else;
		if(count == 8'b01011000) Data_Init[4] <= Rx;
		else;
		if(count == 8'b01101000) Data_Init[5] <= Rx;
		else;
		if(count == 8'b01111000) Data_Init[6] <= Rx;
		else;
		if(count == 8'b10001000) Data_Init[7] <= Rx;
		else;
		if(count == 8'b10011000) Data_Init[7] <= Rx;
		else;
		if(count == 8'b10101000) Data_Rx <= Data_Init;
		else;
		if(count == 8'b10101010) DR <= 0;
		else;
	end
end

endmodule
```

19、设计 verilog 代码实现一个 4 位十进制数组合锁（假设密码为 0420），如果用户成功输入顺序正确的 4 位数，输出 Right 信号，如果任何一位输入错误，则在 4 位数输入完毕后输出 Wrong 信号，并且输入 3 次错误后组合锁锁死，不能再解锁，Lock 信号保持逻辑 1，除非系统通过 reset 信号复位。（10 分）

要求：

- 画出状态转移图；
- 用 verilog 实现状态机，模块定义如下；
- 不要求测试文件。

```
module NumLock(
input clock,
input reset,
input [3:0] Num,
output Right,
output Wrong,
output Lock);

clock:时钟输入；
reset:系统复位输入，高电平有效；
Num:单个数位输入；
Right:解锁成功输出；
Wrong:解锁失败输出；
Lock:锁死输出。
```

代码略

要点：

状态机 2 段式/3 段式写法均可；

输入 3 次错误后组合锁锁死，所以状态转移图要画 3 个密码错误的状态；

不要忘记 Lock 信号；

复位时 Lock 信号置零，并且回到 IDLE 状态；

Mealy 型与 Moore 型实现均可，Moore 型状态数目多 2 个。