第一章思考题

1、什么是硬件描述语言？它的主要作用是什么？
2、目前世界上符合IEEE标准的硬件描述语言有哪两种？它们各有什么特点？
3、什么情况下需要采用硬件描述语言的设计方法？
4、采用硬件描述语言设计方法的优点是什么？有什么缺点？
5、简单叙述一下利用EDA工具并采用硬件描述语言（HDL）的设计方法和流程。
6、硬件描述语言可以用哪两种方式参与复杂数字电路的设计？
7、用硬件描述语言设计的数字系统需要经过哪些步骤才能与具体的电路相对应？
8、为什么说用硬件描述语言设计的数字逻辑系统具有最大的灵活性并可以映射到任何工艺的电路上？
9、软核是什么？虚拟器件是什么？它们的作用是什么？
10、集成电路行业中IP的含义是什么？固核是什么？硬核是什么？与软核相比它们各有什么特点？各适用于什么场合？
11、简述Top-Down设计方法和硬件描述语言的关系。
12、System Verilog与Verilog有什么关系？适合于何种设计？

第二章思考题

1、Verilog语言有什么作用？
2、构成模块的关键词是什么？
3、为什么说可以用Verilog构成非常复杂的电路结构？
4、为什么可以用比较抽象的描述来设计具体的电路结构？
5、是否任意抽象的符合语法的Verilog模块都可以通过综合工具转变为电路结构？
6、什么叫综合？
7、综合是用什么工具来完成的？
8、通过综合产生的是什么？产生的结果有什么用处？
9、仿真是什么？为什么要进行仿真？
10、仿真可以在几个层面上进行？每个层面的仿真有什么意义？
11、模块的端口是如何描述的？
12、在引用实例模块的时候，如何在主模块中连接信号线？
13、如何产生连续的周期性测试时钟？
14、如果不用initial块，能否产生测试时钟？
15、从本讲的简单例子，是否能明白always块与initial块有什么不同？
16、为什么说Verilog可以用来设计数字逻辑电路和系统？

第三章思考题

1、模块由几个部分组成？
2、端口分为几种？
3、为什么端口要说明信号的位宽？
4、能否说模块相当于电路图中的功能模块，端口相当于功能模块的引脚？
5、模块中的功能描述可以由哪几类语句或语句块组成？它们出现的顺序会不会影响功能的描述？
6、这几类描述中哪一种直接与电路结构有关？
7、最基本的Verilog变量有几种类型？
8、reg型和wire型变量的差别是什么？
9、由连续赋值语句（assign）赋值的变量能否是reg类型的？
10、在always块中被赋值的变量能否是wire类型的？如果不能是wire类型，那么必须是什么类型的？它们表示的一定是实际的寄存器吗？
11、参数类型的变量有什么用处？
12、Verilog语法规定的参数传递和重新定义功能有什么直接的应用价值？
13、逻辑比较运算符小于等于“<=”和非阻塞赋值“<=”的表示是完全一样的，为什么Verilog在语句解释和编译时不会搞错？
14、是否可以说实例引用的描述实际上就是严格意义上的电路结构描述？

第四章思考题

1、逻辑运算符与按位逻辑运算符有什么不同，它们各在什么场合使用？
2、指出两种逻辑等式运算符的不同点，解释书上的真值表。
3、拼接符的作用是什么？为什么说合理地使用拼接符可以提高程序的可读性和可维护性？拼接符表示的操作其物理意义是什么？
4、如果都不带时间延迟，阻塞和非阻塞赋值有什么不同？举例说明它们的不同点。
5、举例说明顺序块和并行块的不同？
6、如果在顺序块中，前面有一条语句是无限循环，下面的语句能否进行？
7、如果在并行块中，发生上述情况，会如何呢？

第五章思考题

1、为什么建议在编写Verilog模块程序时，如果用到if语句，建议大家把配套的else情况也考虑在内？
2、用if（条件1）语句；else if（条件2）语句；elseif（条件3）语句；…else 语句和用case endcase表示不同条件下的多个分支是完全相同的，还是有什么不同？
3、如果case语句的分支条件没有覆盖所有可能的组合条件，定义了default项和没有定义default项有什么不同？
4、仔细阐述case、casex和casez之间的不同。
5、forever语句如果运行了，在它下面的语句能否运行？它位于begin－end块和位于fork－join块有什么不同？
6、forever语句、repeat语句能否独立于过程块而存在，即能否不在initial或always块中使用？
7、用for循环为存储器许多单元赋值时是否需要时间？为什么如果不定义时间延迟，它可以不需要时间就把不管多大的存储器赋值完毕？
8、for循环是否可以表示可以综合的组合逻辑？请举例说明。

第五章思考题

1、为什么建议在编写Verilog模块程序时，如果用到if语句，建议大家把配套的else情况也考虑在内？

因为如果没有配套的 else 语句，在不满足 if 条件语句时，将会保持原来的状态不变，从而在综合时会产生一个锁存器，而这是设计不想要的结果。

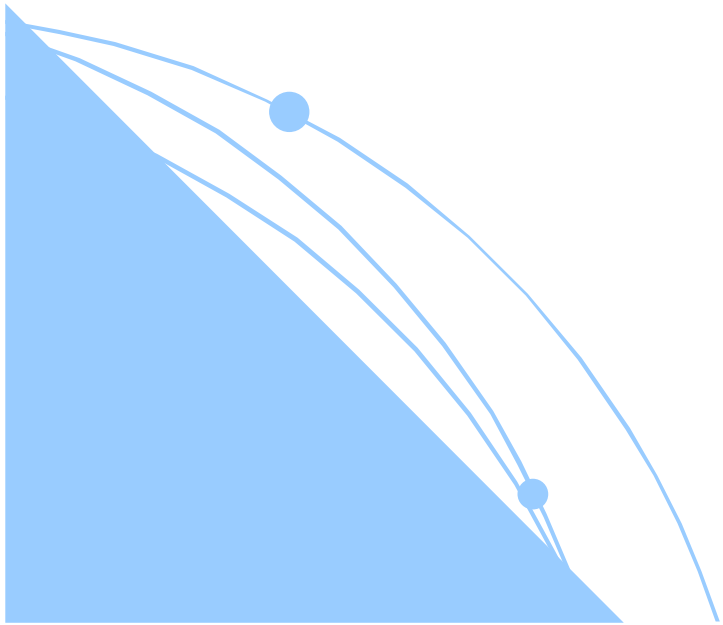
2.用if（条件1）语句；else if（条件2）语句；elseif（条件3）语句；…else 语句和用case endcase表示不同条件下的多个分支是完全相同的，还是有什么不同？

不是完全相同。(1) 与 case 语句中的控制表达式和多分支表达式这种比较相比，if_else_if 结构中条件表达式更为直观些。(2) 对于那些分支表达式中存在不定值 x 和高阻值 z 的位时 case 语句提供了处理这种情况的手段。

第五章思考题

3、如果case语句的分支条件没有覆盖所有可能的组合条件，定义了default项和没有定义default项有什么不同？

定义了 default 项则会使电路描述的更加的清楚，综合的时候不会产生不想要的结果，没用定义 default 则会使在综合是产生一个锁存器。



第五章思考题

4. 仔细阐述case、casex和casez之间的不同?

case	0	1	x	z
0	1	0	0	0
1	0	1	0	0
x	0	0	1	0
z	0	0	0	1

casez	0	1	x	z
0	1	0	0	1
1	0	1	0	1
x	0	0	1	1
z	1	1	1	1

casex	0	1	x	z
0	1	0	1	1
1	0	1	1	1
x	1	1	1	1
z	1	1	1	1

case、casex、casez 对应的真值表如上，可以看出 case 无论是 0,1,还是 x 高阻都能够比较，而 casez 不将高阻进行比较，在其它情况都进行比较；而 casex 不将高阻和 x 进行比较，在其它情况进行比较。

第五章思考题

5、forever语句如果运行了，在它下面的语句能否运行？它位于begin－end块和位于fork－join块有什么不同？

不能运行。位于 begin end，由于 begin and 是顺序块，所以只要执行到 forever 则将不能运行下面的程序；而位于 fork join，它是并行块，执行了 forever 还是能够执行 forever 下面的语句。

6. forever语句、repeat语句能否独立于过程块而存在，即能否不在initial或always块中使用？

forever 不能独立于过程块中，而 repeat 能够独立于过程块中。

第五章思考题

7、用for循环为存储器许多单元赋值时是否需要时间？为什么如果不定义时间延迟，它可以不需要时间就把不管多大的存储器赋值完毕？

如果定义了时间延迟则需要时间，否则不需要时间。因为循环的边界是确定的那么在综合时该循环语句被认为是重复的硬件结构。

8. for循环是否可以表示可以综合的组合逻辑？请举例说明。

可以表示综合的组合逻辑。例如用 for 循环实现的乘法器

第五章思考题

9、在编写测试模块时用什么方法可以使for 循环按照时钟的节拍运行？请比较图5.3 所示程序段。？

```
always @(posedge clk)
  begin
    for ( i=0; i<=1024; i=i+1)
        mem[i] = i;
  end
这样写能不能按照时钟节拍来对
mem[i]赋值？右边框内的程序呢？
```

```
initial
  begin
    for ( i=0; i<=1024; i=i+1)
        begin
          mem[i] = i;
          @(posedge clk)
    end
  end
```

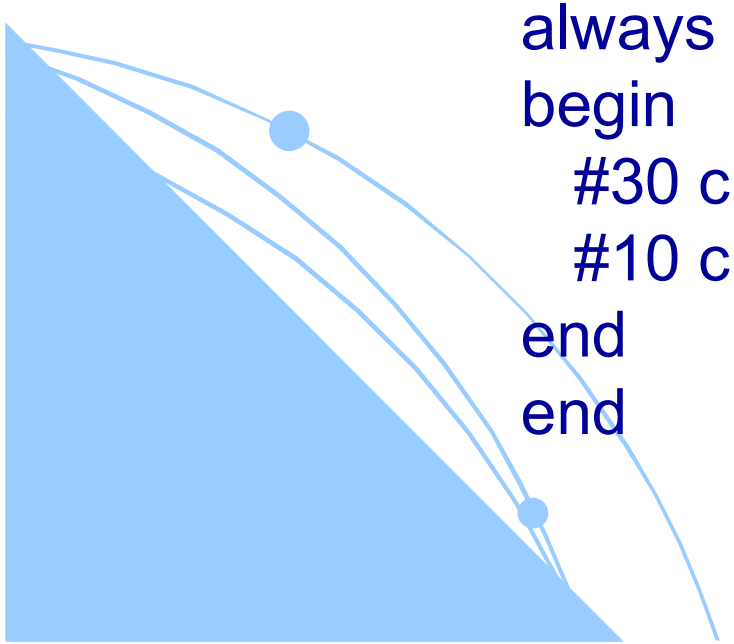
图 5.3　两种程序段

可以在for 循环的最后嵌套时钟节拍运行的信号。第一种程序不能按照时钟节拍来对mem[i]赋值，而第二种程序可以。

第五章思考题

10、设计一个周期为40 个时钟单位的时钟循环，其占空比为25%，使用always和initial 块进行设计，将其在仿真0 时刻的值初始化为0。

```
initial
begin
clock=0;
always
begin
   #30 clock=0;
   #10 clock=1;
end
end
```



第五章思考题

12、给定下面含有阻塞过程赋值语句的initial 块，每个语句在什么仿真时刻开始执行？a，b，c 和d 在仿真过程中的中间值和仿真结束时的值是什么?。

```
initial
begin
  a=1'b0;
  b=#10 1'b0;
  c=#5 1'b0;
  d=#20{a,b,c};
end
```

第一条语句在仿真开始时就执行，第二句在仿真10 个时钟单元后执行，第三句在仿真15 个时钟信号单元后执行，第四句在仿真35 个时钟单元后执行。

在中间仿真过程中a=0,b,c,d 为不确定值

结束时abcd 的值是a=1'b0,b=1'0,c=1'0,d=3'b000。

第五章思考题

14.下面例子中d 的最终值是什么？

```
initial
begin
  b=1'b1;
  c=1'b0;
  #10 b=1'b0;
end
initial
begin
  d=#25(b|c);
end
```

D 的最终值0

第五章思考题

15使用case 语句设计八功能的算术运算单元(ALU),其输入信号a 和b 均为4 位，还有功能选择信号select 为3 位，输出信号为out(5 位)，算术运算单元ALU所执行的操作与select 信号有关，具体关系如5.1 所列(忽略输出结果中的上溢和下溢的位)。

表 5.1　select 信号的功能

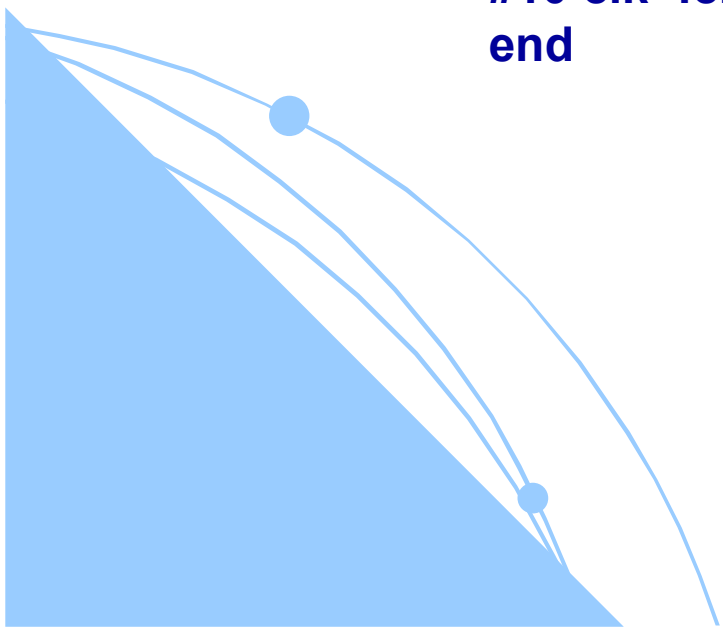
select 信号	功　能
3 'b 000	out = a
3 'b 001	out = a+b
3 'b 010	out = a−b
3 'b 011	out = a/b
3 'b 100	out=a%b（余数）
3 'b 101	out = a << 1
3 'b 110	out = a >> 1
3 'b 111	out = a > b（大小比较）

```
module ALU(a,b,select,out);
input[3:0] a,b;
input[2:0] select;
output[4:0] out;
reg[4:0] out;
always @(select)
begin
case(select)
3'b000: out=a;
3'b001: out=a+b;
3'b010: out=a-b;
3'b011: out=a/b;
3'b100: out=a%b;
3'b101: out=a<<1;
3'b110: out=a>>1;
3'b111: out=a>b;
default:out=5'bx;
endcase
end
endmodule
```

第五章思考题

16.使用while 循环设计一个时钟信号发生器。其时钟信号的初值为0，周期为10个时间单元。

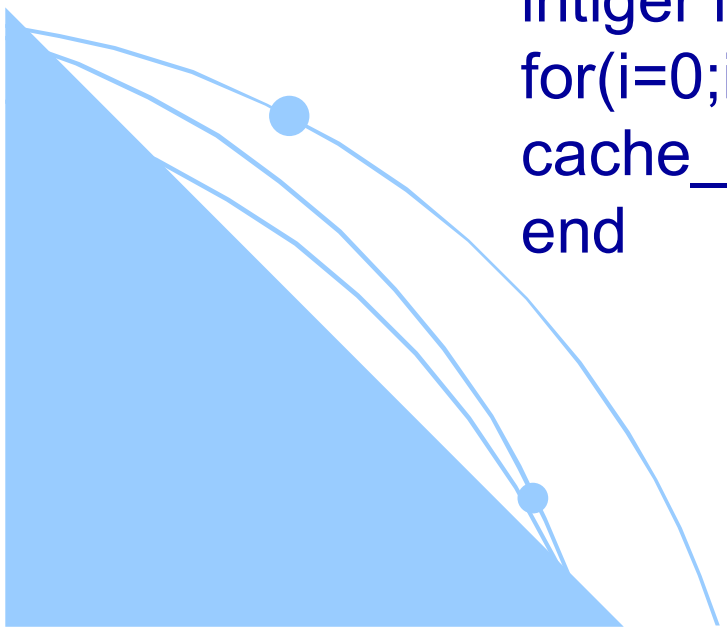
```
initial
begin
clk=0;
while(1)
#10 clk=!clk;
end
```



第五章思考题

17.使用for 循环对一个长度为1024(地址从0~1023)、位宽为4 的寄存器类型数组cache_var 进行初始化，把所有单元都设置为0.

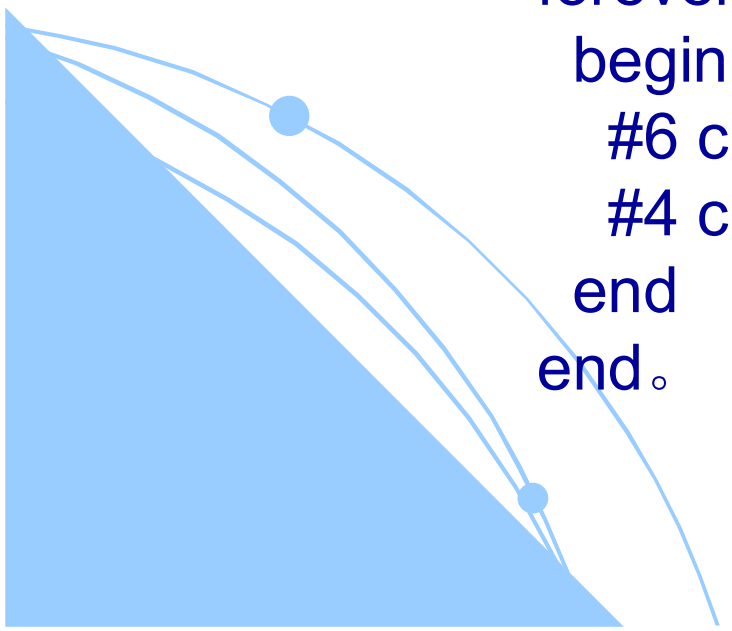
```
begin
reg[3:0] cache_var[1023:0];
intiger i;
for(i=0;i<1024;i++)
cache_var[i]=0;
end
```



第五章思考题

18.使用forever 循环设计一个时钟信号，周期为10，占空比为40%，初值为0.

```
initial
begin
 clk=0;
forever
  begin
    #6 clk=0;
    #4 clk=1;
  end
end。
```



第五章思考题

19.下面是一个内嵌顺序块和并行块的块语句。该块的执行结束时间是多少？事件的顺序是怎样的？每条语句的仿真结束时间是多少？.

```
initial
begin
  x=1'b0;
  #5 y=1'b1;
    fork #20 a=x;
         #15   b=y;
     join
 #40 x=1'b1;
    fork //并行执行
       #10 p=x;
          begin
             #10 a=y; #30 b=x;
          end
      #5 m=y;
     join
end
```

```
initial
 begin // 顺序执行
 x=1'b0;
 #5 y=1'b1; ......5
 fork //并行执行
   #20 a=x; ......25
   #15 b=y; ......20
 join
 #40 x=1'b1; // 顺序执行......65
 fork //并行执行
   #10 p=x; ......75
   begin // 顺序执行
     #10 a=y; ......75
     #30 b=x; ......105
   end
   #5 m=y; ......70
 join
 end
```

第五章思考题

19.用forever 循环语句，命名块(named block)和禁用(disabling of) 命名块来设计 一个八位计数器。这个计数器从count=5 开始计数，到count=67 结束计数。每个时钟正跳变计数器加一，时钟的周期为10，计数器的设计只用到了一次循环，然后就被禁用了(提示：使用disable 语句)

```
reg[7:0] count;
initial
  begin
    count =5;
  end
 begin:mame block
  forever
    begin: disabling of
    always @(posedge clk)
     begin
        if(count<67)
        #10 count=count+1;
        disable: disabling of;
      end
    end
end
```

第六章思考题

1、怎样理解initial语句只执行一次的概念？
2、在initial语句引导的过程块中是否可以有循环语句？如果可以，是否与思考题1相矛盾？
3、怎样理解由always语句引导的过程块是不断活动的？
4、不断活动与不断执行有什么不同？
5、怎样理解沿触发和电平触发的不同？
6、是不是可以说沿触发是有间隔的，在一定的时间区间里只需要注意有限的点，而电平触发却需要注意无穷多个点？
7、沿触发的always块和电平触发的always块各表示什么类型的逻辑电路的行为？为什么？
8、简单叙述任务与函数的不同点。
9、简单叙述$display、$write和$strobe的不同点。
10、简单叙述Verilog1364－2001版语法规定的电平敏感列表的简化写法。
11、如何在Verilog测试模块中，利用文件的读写产生预定格式的信号，并记录有测试价值的信号？

思考题

1、为什么在多模块调试的情况下$monitor需要配合$monitoron和$monitoroff来工作？

2、请用$random配合求模运算编写：

（1）用于测试的跳变沿抖动为周期1/10的时钟波形。

（2）随机出现的脉宽随机的窄脉冲。

3、Verilog的编译预处理与C语言的编译预处理有什么不同？

4、请仔细阐述`timescale编译预处理的作用？

5、不同`timescale定义的多模块仿真测试时需要注意什么？

6、为什么说系统任务$readmem可以用来产生用于算法验证的及其复杂的测试用数据流？

7、为什么说熟练的使用条件编译命令可以使源代码有更大的灵活性，可以适用于不同的实现对象，如不同工艺的ASIC或速度规模不同的FPGA或CPLD，从而为软核的商品化创造条件？