



南京大学

《数字电路与数字系统实验》实验报告

实验五： 计数器和时钟

姓名： 毛彦杰

学号： 191220081

班级： 数字电路与数字系统实验 2 班

院系： 计算机科学与技术系

邮箱： 1363818182@qq.com

实验时间： 2020/9/28

预习报告：

一、计数器

5.1 加法计数器

利用触发器可以构成简单的计数器。图 5-1 是由 3 个上升沿触发的 D 触发器组成的 3 位二进制异步加法计数器，即在每个 Clock 的上升沿，计数器输出 $Q_2Q_1Q_0$ 加 1。

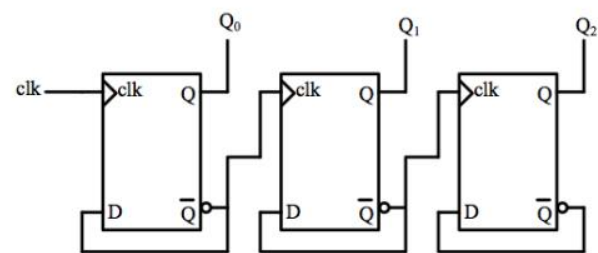


图 5-1: 3 位二进制加法计数器

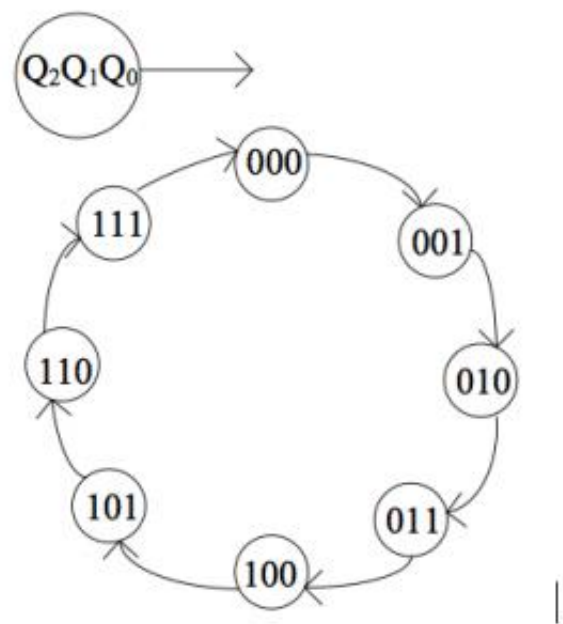


图 5-2: 3 位二进制加法计数器状态图

5.2 减法计数器

利用 D 触发器同样可以构成减法计数器，图 5-3 是由 3 个上升沿触发的 D 触发器组成的 3 位二进制异步减法计数器

图 5-4 是此 3 位二进制异步减法计数器的状态转移图。

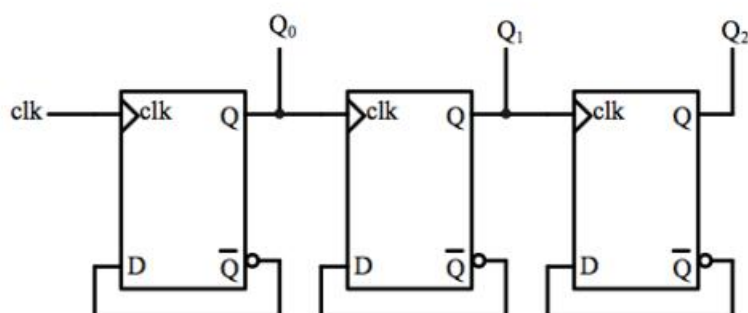


图 5-3: 3 位二进制异步减法计数器

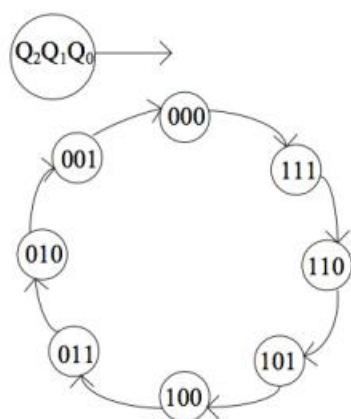


图 5-4: 3 位二进制减法计数器状态图

三位二进制减法计数器功能代码：

```
1 module vminus3(clk,en,out_q);
2     input  clk;
3     input  en;
4     output reg [2:0] out_q;
5
6     always @ (posedge clk)
7         if (en)    out_q <= out_q -1;
8         else      out_q <= 0;
9 endmodule
```

*注意:



在仿真计数器的时候时常会出现输出值一直是未定义，即 XXXX 的情况。这主要是由于计数器的值或输出在仿真开始时是未定义的。在累加或翻转输出时对未定义的值进行操作结果还是 XXXX。这时需要在计数器代码内部增加 `initial` 语句，对计数值和输出结果进行初始化。

二、定时器

5.3 定时器

如果在计数器的时钟输入端输入一个固定周期的时钟，那么计数器就变成了定时器。

时钟信号作为计数器的时钟信号，即可构成一个定时器：

$$\text{计时时间} = \text{脉冲个数} \times \text{脉冲周期}$$

产生时钟周期为 1s 的时钟信号示例代码

```
1 always @(posedge clk)
2     if(count_clk==25000000)
3     begin
4         count_clk <=0;
5         clk_1s <= ~clk_1s;
6     end
7     else
8         count_clk <= count_clk+1;
```

思考题：

请阅读、理解此段代码，并请思考为了能满足计数到 25000000 的要求，变量“count_clk”的宽度如何设定？

解答：由于使用 reg 类型的 count_clk 变量，通过计算可以发现 $\log_2 25000000$ 的值在 24.5 左右，故对于二进制数 count_clk 来说至少需要 25 位的宽度才能表示到 25000000。



对数	反对数
Log	25000000
底数>	<input type="radio"/> e <input type="radio"/> 10 <input checked="" type="radio"/> 2 <input type="radio"/> 其它: <input type="text"/>
计算	24.575424759

5.4.1 基础实验：在 DE10-Standard 开发板上实现一个计时器，在七段数码管上直接以十进制显示。

一、实验目的

复习计数器的工作原理，通过介绍几种简单计数器的工作过程和设计方法、以及开发板系统时钟的使用，学习计数器的设计和定时器的工作原理。学习 FPGA 开发平台上时钟源的使用，在开发板上实现一个计时器，在七段数码管上直接以十进制显示。

利用开发板上的频率为 50MHz 的时钟，请先设计一个分频器，其输入为 50MHz 的时钟，输出为一个频率为 1Hz，周期为 1 秒的时钟信号。再用这个新的频率为 1Hz 的时钟信号作为你设计的时钟信号，进行计数。要求此计时器有开始、暂停和清零功能，要求从 00 计数

到 99，计数值到 99 后重新从零开始计数。在数码管上用两位数字显示。可以在计时结束的时候让某一个发光二极管闪烁，提示计时结束，类似数电教材 8.4.3 节 74x163 计数器中的 RCO 信号高电平一个周期。

二、实验原理

利用开发板自带的 50MHZ 的时钟信号生成所需的时钟信号：

表 5-2: 1 秒时钟生成代码

```
1 always @(posedge clk)
2     if(count_clk==25000000)
3     begin
4         count_clk <=0;
5         clk_1s <= ~clk_1s;
6     end
7     else
8         count_clk <= count_clk+1;
```

计数器：计数器是数字系统中用的较多的基本逻辑器件，它的基本功能是统计时钟脉冲的个数，即实现计数操作，它也可用于分频、定时、产生节拍脉冲和脉冲序列等。本次实验是利用 FPGA 开发板上提供的外部输入时钟，形成一个自己的周期一定的时钟，再利用这个时钟来进行接下来的实验。

三、实验设备环境

硬件器材：FPGA 开发板

软件平台：Quartus 开发平台

四、实验步骤

设计思路：

clk 表示开发板自带的时钟信号

通过 count_clk 的计数来将 clk 转变成周期为 1s 的时钟信号 clk_1s

en 代表使能端

stop 为暂停信号

clear 为清零信号

endone 为一轮结束的标志（一轮计数结束时它为 1，持续 1s）

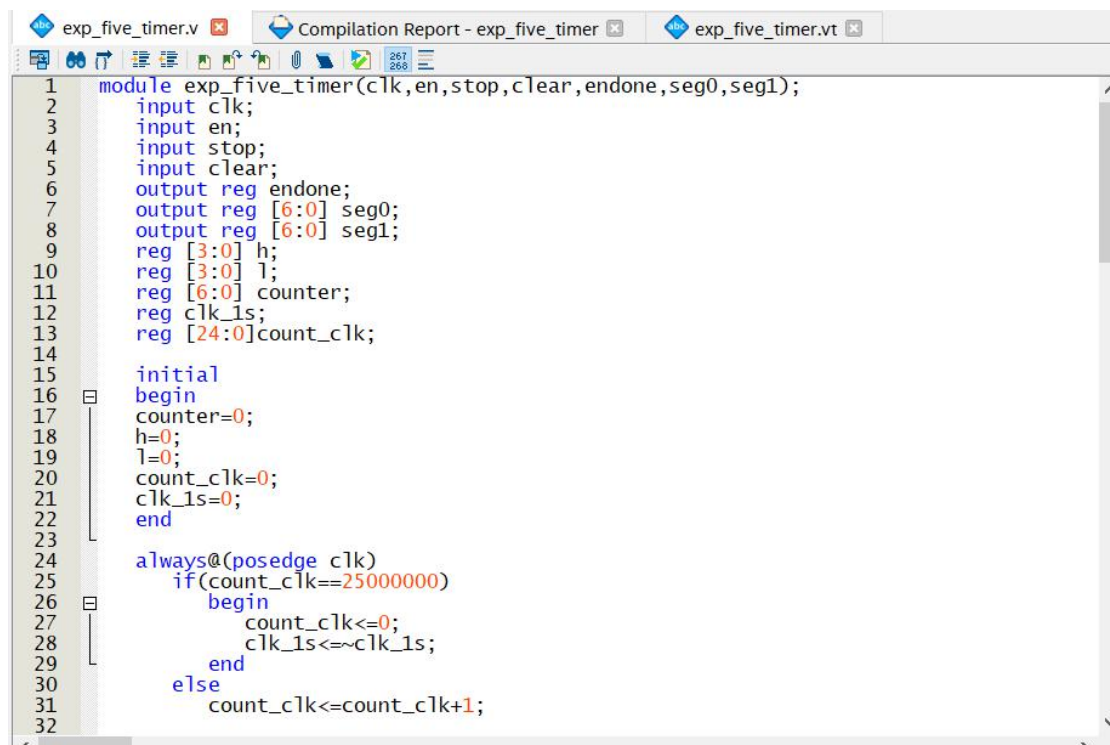
h 表示其十位，用 seg1 来将其输出到数码管上

l 表示其个位，用 seg0 将其输出到数码管上

用 counter 表示计数的真值

0. 先将 counter, h, l, count_clk, clk_1s 初始化为 0

1. 产生一个 1s 的时钟信号：



```
1 module exp_five_timer(clk,en,stop,clear,endone,seg0,seg1);
2     input clk;
3     input en;
4     input stop;
5     input clear;
6     output reg endone;
7     output reg [6:0] seg0;
8     output reg [6:0] seg1;
9     reg [3:0] h;
10    reg [3:0] l;
11    reg [6:0] counter;
12    reg clk_1s;
13    reg [24:0] count_clk;
14
15    initial
16    begin
17        counter=0;
18        h=0;
19        l=0;
20        count_clk=0;
21        clk_1s=0;
22    end
23
24    always@(posedge clk)
25        if(count_clk==25000000)
26            begin
27                count_clk<=0;
28                clk_1s<=~clk_1s;
29            end
30        else
31            count_clk<=count_clk+1;
32    end
```


2. 对每次 clk 上升沿时进行判断:

1) 没有使能时, 将输出数据变为 0, 数码管不显示数据。

2) 使能时, 将 endone 置为 0 **endone=0;**

先看是否是清零状态, 再看是否是暂停状态, 最后再处理正常状态。采用模 100 运算来计算 counter 以达到在 100 内循环计数的效果。在 counter=99 时将 endone 置为 1, 这样在下一次 clk 上升沿时 (也就是一轮计时结束时) 表示 endone 的 led 会亮起。

```
if(clear)
begin
h=0;
l=0;
counter=0;
end

else if(stop)
begin
h=h;
l=l;
counter=counter;
end

else
begin
if(counter==99)
begin
counter=(counter+1)%100;
endone=1;
end
else
begin
counter=(counter+1)%100;
endone=0;
end
end
```


然后处理 h, l 以及对应的 seg1, seg0:

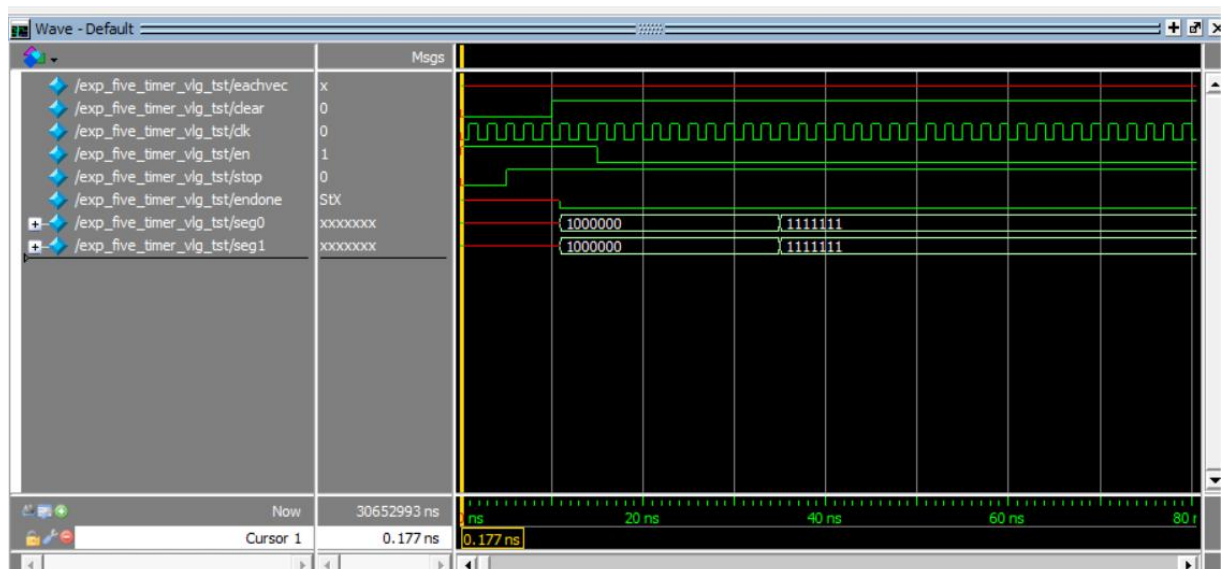
```
if(counter<100)
    begin
        l=counter%10;
        h=(counter-(counter%10))/10;
    end
else
    begin
        l=0;
        h=0;
    end

case(h)
0: seg1 = 7'b1000000;
1: seg1 = 7'b1111001;
2: seg1 = 7'b0100100;
3: seg1 = 7'b0110000;
4: seg1 = 7'b0011001;
5: seg1 = 7'b0010010;
6: seg1 = 7'b0000010;
7: seg1 = 7'b1111000;
8: seg1 = 7'b0000000;
9: seg1 = 7'b0010000;
endcase
case(l)
0: seg0 = 7'b1000000;
1: seg0 = 7'b1111001;
2: seg0 = 7'b0100100;
3: seg0 = 7'b0110000;
4: seg0 = 7'b0011001;
5: seg0 = 7'b0010010;
6: seg0 = 7'b0000010;
7: seg0 = 7'b1111000;
8: seg0 = 7'b0000000;
9: seg0 = 7'b0010000;
endcase
```

激励代码：




















```
exp_five_timer.v  Compilation Report - exp_five_timer  exp_five_timer.vt
37 // wires
38 wire endone;
39 wire [6:0] seg0;
40 wire [6:0] seg1;
41
42 // assign statements (if any)
43 exp_five_timer i1 (
44 // port map - connection between master ports and signals/registers
45 .clear(clear),
46 .clk(clk),
47 .en(en),
48 .endone(endone),
49 .seg0(seg0),
50 .seg1(seg1),
51 .stop(stop)
52 );
53 initial
54 begin
55 // code that executes only once
56 // insert code here --> begin
57 clk=0;
58 en=1;clear=0;stop=0;#5;
59 en=1;clear=0;stop=1;#5;
60 en=1;clear=1;stop=1;#5;
61 en=0;clear=1;stop=1;#5;
62 // --> end
63 $display("Running testbench");
64 end
65 always
66 // optional sensitivity list
67 // @(event1 or event2 or .... eventn)
68 begin
69 // code executes for every event on sensitivity list
70 // insert code here --> begin
71 #1 clk=~clk;
72 // --> end
73 end
74 endmodule
75
```

ModelSim 仿真波形：



将 clk_1s 加快 5000000 倍来仿真，从而能够较为清晰地观察仿真波形，仿真结果和预期一样。

引脚分配:

Named: *		Edit: <input type="checkbox"/>					
Node Name	Direction	Location	I/O Bank	VREF Group	Pin Location	I/O Stan	
 clear	Input	PIN_AB30	5B	B5B_N0	PIN_AB30	2.5 V	
 clk	Input	PIN_AF14	3B	B3B_N0	PIN_AF14	2.5 V	
 en	Input	PIN_AA30	5B	B5B_N0	PIN_AA30	2.5 V	
 endone	Output	PIN_AA24	5A	B5A_N0	PIN_AA24	2.5 V	
 seg0[6]	Output	PIN_AH18	4A	B4A_N0	PIN_AH18	2.5 V	
 seg0[5]	Output	PIN_AG18	4A	B4A_N0	PIN_AG18	2.5 V	
 seg0[4]	Output	PIN_AH17	4A	B4A_N0	PIN_AH17	2.5 V	
 seg0[3]	Output	PIN_AG16	4A	B4A_N0	PIN_AG16	2.5 V	
 seg0[2]	Output	PIN_AG17	4A	B4A_N0	PIN_AG17	2.5 V	
 seg0[1]	Output	PIN_V18	4A	B4A_N0	PIN_V18	2.5 V	
 seg0[0]	Output	PIN_W17	4A	B4A_N0	PIN_W17	2.5 V	
 seg1[6]	Output	PIN_V17	4A	B4A_N0	PIN_V17	2.5 V	
 seg1[5]	Output	PIN_AE17	4A	B4A_N0	PIN_AE17	2.5 V	
 seg1[4]	Output	PIN_AE18	4A	B4A_N0	PIN_AE18	2.5 V	
 seg1[3]	Output	PIN_AD17	4A	B4A_N0	PIN_AD17	2.5 V	
 seg1[2]	Output	PIN_AE16	4A	B4A_N0	PIN_AE16	2.5 V	
 seg1[1]	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V	
 seg1[0]	Output	PIN_AF16	4A	B4A_N0	PIN_AF16	2.5 V	
 stop	Input	PIN_Y27	5B	B5B_N0	PIN_Y27	2.5 V	
<<new node>>							

SW[9]:en

SW[1]:stop

SW[0]:clear

LED[0]:endone

数码管 0:计数的个位

数码管 1: 计数的十位

五：实验过程中遇到的问题及解决

第一次实现时数码管会在 99 到 00 时在 00 处停留 2 秒，前 1 秒发光二极管闪烁，后 1 秒发光二极管熄灭，后来发现是自己在 `count=100` 的情况以 `count>=100` 为条件单独处理了一次，而下一个时钟到来时 `count=0`，`endone=1` 生效，再下一个时钟到来时再执行以 `count<100` 为条件的操作，而这一次 `count` 还是 0，`endone=0`。因而多了 1 秒。

后来将 `count=99` 的情况单独处理，`count=99` 时 `endone=1`，在下一个时钟周期到来时 `endone` 使二极管发光，而其他情况用模 100 计算 `count` 的形式完成了合并，从而每一轮 00 只会出现 1 秒。

在 .vt 文件中没去掉 `@eachvec` 导致波形图无法输出。经多次检查才发现是这个问题。以后在写 testbench 的时候一定要先将多余的部分注释掉再开始写。

5.4.2 拓展实验：在 DE-10 Standard 开发板上实现一个电子时钟

一、实验目的

在 DE-10 Standard 开发板上实现一个电子时钟，时钟要求能够显示时、分、秒；还可以有以下功能：调整时间；闹铃（在特定时间 LED 闪烁）；秒表；等。

二、实验设备环境

硬件器材：FPGA 开发板

软件平台：Quartus 开发平台

三、实验步骤

设计思路：

利用总秒数 total 进行模 86400 加法运算，使其在 1 到 86400 间滚动（一天总秒数 86400），并利用 total 值计算出当下的时分秒（hour, minute, second）。

```
begin
total=(total+3600)%86400;
hour=total/3600;
minute=(total%3600)/60;
second=total%60;
end
```

闹铃功能的实现核心在于设一个闹铃的总秒数 alarmtotal 来表示闹铃预设时间, 在开启闹铃 (alarm_on==1) 且闹铃时间与时钟时间相同时 (alarmtotal==total) LED 亮。

```
//检查闹铃时间到没到
if(alarm_on && (total==alarmtotal))
alarmlight=1;
else
alarmlight=0;
```

秒表的实现即单独设置一个 stoptotal，不与 total 冲突，当开启秒表功能时（stopwatch==1）stoptotal 开始计数，其余原理与时钟一样，只是多出了清零(clearstopwatch)功能，方便置零。

需要注意的是在秒表与闹钟功能进行的时候时钟还需要正常保持计时（时间正常流淌），故在这两种功能下 total 还是正常进行模 86400 的自加运算。以下为秒表与闹钟功能代码中对 total 的处理：

```
else if(stopwatch)//如果stopwatch功能打开了
begin//*****以下为秒表功能
    if(clearstopwatch)
    begin
        total=(total+1)%86400;//时间正常计算，不影响时钟的正常运作，只是不显示
    else
    begin
        total=(total+1)%86400;//时间正常计算，不影响时钟的正常运作，只是不显示
    else//stopwatch没打开，那就是set_alarm打开了
    begin//*****以下为设置闹钟的功能
        flag=~flag;//便于闪烁
        total=(total+1)%86400;//时间正常计算，不影响时钟的正常运作，只是不显示
```

另外在设置时钟与闹钟的时候我做了一个闪烁功能，如果调整小时（adjh_n==1）则显示小时的数码管每两秒闪烁一次，如果调整分钟（adjm_n==1）则分钟对应的数码管每两秒闪烁一次，调整秒钟同上。

闪烁的原理比较简单，在功能开始时设定一个 flag，使其每秒取反一次；在下面处理数码管显示的时候在 flag 为 1 时显示小时的数码管熄灭，分钟和秒钟的数码管正常显示；flag 为 0 时时分秒的数码管都正常显示。下图以调整秒钟为例：


```

else if(adjs)//调整秒钟
begin
    if(keys_n)
    begin
        hour=hour;
        minute=minute;
        second=second;
        total=total;
    end
    else
    begin
        total=(total+1)%86400;
        hour=total/3600;
        minute=(total%3600)/60;
        second=total%60;
    end

    if(flag)//秒钟数码管的闪烁处理
    begin
        hh=hour/10;
        lh=hour%10;
        hm=minute/10;
        lm=minute%10;

        hs=10;
        ls=10;
    end
    else
    begin
        hh=hour/10;
        lh=hour%10;
        hm=minute/10;
        lm=minute%10;
        hs=second/10;
        ls=second%10;
    end
end

```

例子中 hh 为小时的十位，lh 为小时的个位

hm 为分钟的十位，lm 为分钟的个位

hs 为秒钟的十位，ls 为秒钟的个位

在需要秒钟的数码管闪烁的时候把 hs 和 ls 置为 10，并在数码管的 case 语句内加入 10 的情况：




























```

case(hs)
0: seg1s = 7'b1000000;
1: seg1s = 7'b1111001;
2: seg1s = 7'b0100100;
3: seg1s = 7'b0110000;
4: seg1s = 7'b0011001;
5: seg1s = 7'b0010010;
6: seg1s = 7'b0000010;
7: seg1s = 7'b1111000;
8: seg1s = 7'b0000000;
9: seg1s = 7'b0010000;
10: seg1s = 7'b1111111;
endcase

case(ls)
0: seg0s = 7'b1000000;
1: seg0s = 7'b1111001;
2: seg0s = 7'b0100100;
3: seg0s = 7'b0110000;
4: seg0s = 7'b0011001;
5: seg0s = 7'b0010010;
6: seg0s = 7'b0000010;
7: seg0s = 7'b1111000;
8: seg0s = 7'b0000000;
9: seg0s = 7'b0010000;
10: seg0s = 7'b1111111;
endcase

```

引脚分配:

Node Name	Direction	Location
 adjh	Input	PIN_AB28
 adjm	Input	PIN_Y27
 adjs	Input	PIN_AB30
 alarm_on	Input	PIN_V25
 alarmlight	Output	PIN_AA24
 clearstopwatch	Input	PIN_AD30
 clk	Input	PIN_AF14
 keyh_n	Input	PIN_AA15
 keym_n	Input	PIN_AA14
 keys_n	Input	PIN_AK4
 seg0h[6]	Output	PIN_AH22
 seg0h[5]	Output	PIN_AF23
 seg0h[4]	Output	PIN_AG23
 seg0h[3]	Output	PIN_AE23
 seg0h[2]	Output	PIN_AE22
 seg0h[1]	Output	PIN_AG22
 seg0h[0]	Output	PIN_AD21
 seg0m[6]	Output	PIN_W16
 seg0m[5]	Output	PIN_AF18
 seg0m[4]	Output	PIN_Y18
 seg0m[3]	Output	PIN_Y17
 seg0m[2]	Output	PIN_AA18
 seg0m[1]	Output	PIN_AB17
 seg0m[0]	Output	PIN_AA21
 seg0s[6]	Output	PIN_AH18
 seg0s[5]	Output	PIN_AG18

out	seg0s[4]	Output	PIN_AH17
out	seg0s[3]	Output	PIN_AG16
out	seg0s[2]	Output	PIN_AG17
out	seg0s[1]	Output	PIN_V18
out	seg0s[0]	Output	PIN_W17
out	seg1h[6]	Output	PIN_AB21
out	seg1h[5]	Output	PIN_AF19
out	seg1h[4]	Output	PIN_AE19
out	seg1h[3]	Output	PIN_AG20
out	seg1h[2]	Output	PIN_AF20
out	seg1h[1]	Output	PIN_AG21
out	seg1h[0]	Output	PIN_AF21
out	seg1m[6]	Output	PIN_AD20
out	seg1m[5]	Output	PIN_AA19
out	seg1m[4]	Output	PIN_AC20
out	seg1m[3]	Output	PIN_AA20
out	seg1m[2]	Output	PIN_AD19
out	seg1m[1]	Output	PIN_W19
out	seg1m[0]	Output	PIN_Y19
out	seg1s[6]	Output	PIN_V17
out	seg1s[5]	Output	PIN_AE17
out	seg1s[4]	Output	PIN_AE18
out	seg1s[3]	Output	PIN_AD17
out	seg1s[2]	Output	PIN_AE16
out	seg1s[1]	Output	PIN_V16
out	seg1s[0]	Output	PIN_AF16
in	set_alarm	Input	PIN_AC28
in	stop	Input	PIN_AA30
in	stopwatch	Input	PIN_AC29

KEY[3]keyh_n, 调整时钟的“小时”

KEY[2]keym_n, 调整时钟的“分钟”

KEY[1]keys_n, 调整时钟的“秒钟”

SW[9]stop, 时钟与秒表的暂停键, 高电平暂停, 时钟设置暂停是为了更好的调整时间的功能

SW[8]stopwatch, 秒表的开关, 高电平为开

SW[7]clearstopwatch, 秒表的清零键, 高电平清零秒表

SW[6]set_alarm, 设置闹钟键, 高电平开始设置闹钟

SW[5]alarm_on, 开启闹钟键, 高电平开启闹钟

LEDR[0], 闹钟显示, 闹钟到时间则亮

六、实验得到的启示

先规划好各个模块的大框架，再专门实现每一个小功能。自顶向下，逐步求精。并且在实现具体的小功能的时候考虑好使用的变量类型/宽度/个数，语句类型/函数/任务等等问题，用恰当的语句来实现适合的功能，构建数字电路建模的 first principle 思考体系。

七、意见与建议

实验手册中的实例对学习 verilog 语句使用与 quartus 相关操作有巨大帮助，希望能在学习新知识的同时接触更多相关例子来加深对新知的理解与应用能力。

希望自己能加强对网上知识的搜索自学能力，这样可以迅速掌握新的知识。