实验名称：实验五　计数器和时钟

姓名：张涵之

学号：191220154

班级：周一 5-6

邮箱：191220154@smail.nju.edu.cn

实验时间：2020/10/16

5.4.1 基础实验

请在 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。

利用开发板上的频率为 50MHz 的时钟，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。

实验目的：实现一个计数器。

实验原理：利用开发板上的频率为 50MHz 的时钟，参考课件中 1 秒时钟生成代码设计一个分频器，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。用这个新的频率为 1Hz 的时钟信号作为设计的时钟信号，进行计数。

计时器的开始和暂停用 SW0 控制，清零功能用 SW1 控制，两个时钟信号分别用两个四位二进制数表示，其中个位逢九清零，十位进一，个位和十位均为九时双双清零，LEDR0 亮，判断清零实现之后，LEDR0 灭。个位与十位在七段数码管上分别显示。

实验环境/器材：实验箱一个，笔记本电脑一台。

程序代码或流程图：

```
module clk_1s(clk,clk_1s);
   input clk;
   output reg clk_1s = 0;
   reg [24:0] count_clk = 1;

always @(posedge clk)
   if(count_clk == 25000000)
      begin
         count_clk <= 1;
         clk_1s <= ~clk_1s;
      end
   else
      count_clk <= count_clk + 1;
endmodule
```

```
module counter(clk,en,clr,out_q1,out_q2,finish);
   input clk;
   input en;
   input clr;
   output reg [3:0] out_q1 = 4'b0000;
   output reg [3:0] out_q2 = 4'b0000;
   output reg finish = 0;

always @ (posedge clk or posedge clr)
   if (clr) begin
      out_q1 = 0;
      out_q2 = 0;
   end
   else if (en) begin
      out_q1 <= out_q1 + 1;
      if (out_q1 == 9) begin
         out_q1 <= 0;
         out_q2 <= out_q2 + 1;
      end
      if (out_q1 == 9 && out_q2 == 9) begin
         out_q1 <= 0;
         out_q2 <= 0;
         finish = 1;
      end
      else if (out_q1 == 0 && out_q2 == 0)
         finish = 0;
   end
   else begin
      out_q1 <= out_q1;
      out_q2 <= out_q2;
   end
endmodule
```

```
module hex(in,out);
   input [3:0] in;
   output reg [6:0] out;

always @ (*)
   case(in)
      0: out = 7'b1000000;
      1: out = 7'b1111001;
      2: out = 7'b0100100;
      3: out = 7'b0110000;
      4: out = 7'b0011001;
      5: out = 7'b0010010;
      6: out = 7'b0000010;
      7: out = 7'b1111000;
      8: out = 7'b0000000;
      9: out = 7'b0010000;
      10: out = 7'b0001000;
      11: out = 7'b0000011;
      12: out = 7'b1000110;
      13: out = 7'b0100001;
      14: out = 7'b0000110;
      15: out = 7'b0001110;
      default: out = 7'b1111111;
   endcase
endmodule
```

```
module exp5_1(clk,en,clr,hex1,hex2,finish);
   input clk;
   input en;
   input clr;
   wire [3:0] out1;
   wire [3:0] out2;
   output [6:0] hex1;
   output [6:0] hex2;
   output finish;

   clk_1s c(clk,clk_1s);
   counter ct(clk_1s,en,clr,out1,out2,finish);
   hex h1(out1,hex1);
   hex h2(out2,hex2);
endmodule
```

输入开发板时钟信号 —> clk_1s 转化为 1s 时钟信号 —>
与开始/暂停和清零段一起接入计数器 counter —> 得到两个四位二进制数 —>
分别接入 hex 转化为七段数码管的对应输入序列 —> 接入两个七段数码管

实验步骤/过程：分模块写出 clk_1s，counter 和 hex 的代码，分别测试。
测试功能无误后综合到顶层模块，再对顶层模块进行测试。

测试方法：测试代码也就是时钟循环输入而已，不如直接烧板子吧。

实验结果：确实可以从 00 计数到 99 并自动清零，此时表示计时结束的二极管确实会发光并保持一个时钟周期，开始/暂停端和清零端确实可以正常工作并实现相应功能。

实验中遇到的问题及解决办法：实验非常顺利，没有碰到任何问题。

实验得到的启示：无。

意见和建议：无。

5.4.2 拓展实验

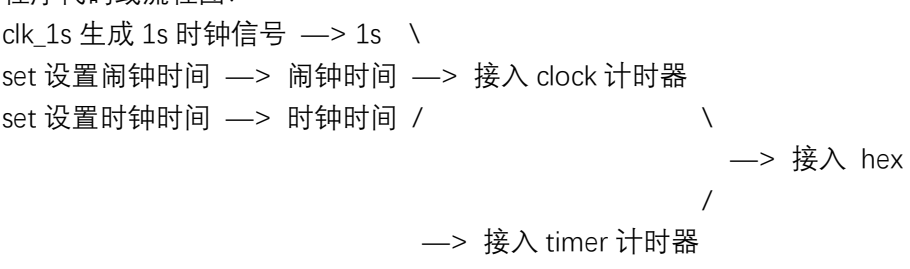
在 DE-10 Standard 开发板上实现一个电子时钟，时钟要求能够显示时、分、秒；
还可以有以下功能：调整时间；闹铃（在特定时间 LED 闪烁）；秒表；等。

实验目的：在 DE-10 Standard 开发板上实现一个电子时钟。

实验原理：分别生成 1 秒和 1 毫秒的时钟信号用于时钟/闹钟和秒表，用两个模块分别设定闹钟和时钟的时间，一个模块模拟时钟的正常走时和闹钟的功能，一个模块单独模拟秒表的功能，一个模块用于将数字显示在七段数码管上，一个顶层模块用于综合。
以开发板上频率为 50MHz 的时钟接入，开关 SW9 控制时钟工作模式/设置模式的切换，SW0 控制设置时钟时间/设置闹钟时间的切换，SW1 控制设定小时/设定分钟的切换，SW2 控制闹钟的开关，SW3 控制时钟/秒表模式的切换，SW4 控制秒表的开始/暂停，SW5 控制秒表的清零。按键 KEY0 在时间设置模式下使用，按下弹起一次相应的小时/分钟增加一次，七段数码管 HEX0~HEX6 用于显示时、分、秒，二极管 LEDR0 用于显示闹钟。
实现的电子时钟能显示时、分、秒，可以调整时间和设置闹钟，达到设定时间后二极管亮起并在一分钟后熄灭，秒表以用于 1 小时以内的计时，精度为毫秒，有暂停、清零功能。

实验环境/器材：实验箱一个，笔记本电脑一台。

程序代码或流程图：




```
module clk_1s(clk,clk_1s);
   input clk;
   output reg clk_1s = 0;
   reg [24:0] count_clk = 0;

always @(posedge clk)
   if(count_clk == 25000000)
      begin
         count_clk <= 0;
         clk_1s <= ~clk_1s;
      end
   else
      count_clk <= count_clk + 1;
endmodule
```

```
module clk_1ns(clk,clk_1s);
   input clk;
   output reg clk_1s = 0;
   reg [24:0] count_clk = 0;

always @(posedge clk)
   if(count_clk == 250000)
      begin
         count_clk <= 0;
         clk_1s <= ~clk_1s;
      end
   else
      count_clk <= count_clk + 1;
endmodule
```

```
module set(en,set_sig,h_m,h,m,s);

   input en;
   input set_sig;
   input h_m;
   output reg [7:0] h = 0;
   output reg [7:0] m = 0;
   output reg [7:0] s = 0;

   always @ (negedge set_sig) begin
      if (en) begin
         if (!h_m) begin
            h <= h + 1;
            if (h == 23)
               h <= 0;
         end
         else begin
            m <= m + 1;
            if (m == 59) begin
               m <= 0;
               h <= h + 1;
               if (h == 23)
                  h <= 0;
            end
         end
      end
   end

endmodule
```

```
module clock(clk,reset,en_alr,in_h,in_m,in_s,alr_h,alr_m,h,m,s,alarm);

   input clk;
   input reset;
   input en_alr;
   input [7:0] in_h;
   input [7:0] in_m;
   input [7:0] in_s;
   input [7:0] alr_h;
   input [7:0] alr_m;

   output reg [7:0] h;
   output reg [7:0] m;
   output reg [7:0] s;
   output reg alarm = 0;

   always @ (posedge clk) begin
      if (clk) begin
         if (reset)
            s <= in_s + 1;
         else s <= s + 1;
         if (s == 59) begin
            s <= 0;
            if (reset)
               m <= in_m + 1;
            else m <= m + 1;
            if (m == 59) begin
               m <= 0;
               if (reset)
                  h <= in_h + 1;
               else h <= h + 1;
               if (h == 23)
                  h <= 0;
            end
            else begin
               if (reset)
                  h <= in_h;
               else h <= h;
            end
         end
         else begin
            if (reset) begin
               m <= in_m;
               h <= in_h;
            end
            else begin
            m <= m;
            h <= h;
            end
         end
         if (en_alr && h == alr_h && m == alr_m)
            alarm = 1;
         else
            alarm = 0;
      end
      else begin
         if (reset)
            s <= in_s;
         else
            s <= s;
      end
   end

endmodule
```

```
module timer(clk,en,clr,min,s,ns);
   input clk;
   input en;
   input clr;
   output reg [7:0] min = 0;
   output reg [7:0] s = 0;
   output reg [7:0] ns = 0;

always @ (posedge clk or posedge clr)
   if (clr) begin
      min = 0;
      s = 0;
      ns = 0;
   end
   else if (en) begin
      ns <= ns + 1;
      if (ns == 99) begin
         ns <= 0;
         s <= s + 1;
         if (s == 59) begin
            s <= 0;
            min <= min + 1;
            if (min == 59)
               min <= 0;
         end
      end
   end
   else begin
      min <= min;
      s <= s;
      ns <= ns;
   end
endmodule
```

```
module hex(in,out);
   input [3:0] in;
   output reg [6:0] out;

always @ (*)
   case(in)
      0: out = 7'b1000000;
      1: out = 7'b1111001;
      2: out = 7'b0100100;
      3: out = 7'b0110000;
      4: out = 7'b0011001;
      5: out = 7'b0010010;
      6: out = 7'b0000010;
      7: out = 7'b1111000;
      8: out = 7'b0000000;
      9: out = 7'b0010000;
      10: out = 7'b0001000;
      11: out = 7'b0000011;
      12: out = 7'b1000110;
      13: out = 7'b0100001;
      14: out = 7'b0000110;
      15: out = 7'b0001110;
      default: out = 7'b1111111;
   endcase
endmodule
```

```
module exp5_2(clk,set,clk_alr,h_m,en_alr,sel_timer,en_timer,clr_timer,set_sig,
   hex0,hex1,hex2,hex3,hex4,hex5,alarm);

   input clk;
   input set,clk_alr,h_m,en_alr;
   input sel_timer,en_timer,clr_timer;
   input set_sig;

   wire cs,cns;
   wire [7:0] hour [3:0];
   wire [7:0] min [3:0];
   wire [7:0] sec [3:0];

   output [6:0] hex0;
   output [6:0] hex1;
   output [6:0] hex2;
   output [6:0] hex3;
   output [6:0] hex4;
   output [6:0] hex5;
   output alarm;

   clk_1s clk1(clk,cs);
   clk_1ns clk2(clk,cns);

   set s_alr(set && clk_alr,set_sig,h_m,hour[2],min[2],sec[2]);
   set s_clk(set && !clk_alr,set_sig,h_m,hour[1],min[1],sec[1]);
   clock c(cs,set && !clk_alr,en_alr,hour[1],min[1],sec[1],hour[2],min[2],
      hour[0],min[0],sec[0],alarm);
   timer tim(cns,en_timer,clr_timer,hour[3],min[3],sec[3]);

   hex h0(sec[set + set * clk_alr + sel_timer * 3] % 10,hex0);
   hex h1(sec[set + set * clk_alr + sel_timer * 3] / 10,hex1);
   hex h2(min[set + set * clk_alr + sel_timer * 3] % 10,hex2);
   hex h3(min[set + set * clk_alr + sel_timer * 3] / 10,hex3);
   hex h4(hour[set + set * clk_alr + sel_timer * 3] % 10,hex4);
   hex h5(hour[set + set * clk_alr + sel_timer * 3] / 10,hex5);

endmodule
```

*hour, min, sec 均以数组(？)形式定义，其中下标为 0 的是时钟走时，下标为 1 的是用户设定时间，下标为 2 的是闹钟时间，下标为 3 的是秒表的分、秒、毫秒，hex 模块中下标为构造的算式，set 即调整模式关闭（正常走时）时取 0，调整模式打开且 clk_alr 为 0 即选择调整时钟时间时取 1，clk_clr 为 1 即选择调整闹钟时间时取 2，上述的几种模式下 sel_timer 均关闭，sel_timer 仅在正常走时模式下打开有效，此时下标取 3，非法下标暂不考虑。

实验步骤/过程：
起初没有单独设置调整时钟时间的模式，用了两个按键分别控制时和分，企图在时钟正常运行模式下，用一个 always 模块同时检测时钟信号和两个按键信号，在时钟正常走时过程中修改时间，单独写出 N 个模块并把 N 个模块合并后，直接接入开发板，发现没有出现预期的效果，只有秒表能正常工作而时钟显示一直进行无规律跳动。
修改后拆分出了时钟正常工作模式和修改时间模式，仍然用两个按键分别控制时和分，继续企图用一个 always 模块检测两个按键信号，发现只有 else if 后的分钟能正常修改，且修改后稳定显示，else if 前的 if 语句不能正确修改小时，且小时仍然不停地出现无规律跳动，思考也许是同时检测两个按键下降沿而没有进行按键消抖，导致混乱情况的发生。
怎么办呢，决定再把时和分的修改拆成两种情况分别进行，只保留一个按键，用开关控制修改小时和修改分钟的情况，基本实现了用按键修改时间这一模块的功能。
顶层模块原先只设置一组时、分、秒，在时钟正常走时和修改时间模块中均有操作，编译时发生报错，提示不能在两个模块中分别修改同一组参数，于是将设定的静态时间与时钟的动态时间分开，修改时间模块输出静态时间，作为时钟模块的输入，在时钟正常走时模块中通过相应判断条件决定是在当前时间上递增还是导入新时间，基本解决问题。
在开发板上测试时钟、修改时间、闹钟、秒表功能，观察显示是否符合预期，进行微调。

测试方法：
时钟整体功能过于复杂很难进行仿真测试，按键抖动和同一 always 模块中多个上升、下降沿叠加问题也难以再模拟中暴露出来，因此之间在开发板上观察数码管输出。

实验结果：
通过观察，开发板上时钟、设置、闹钟、秒表的显示均符合预期。

实验中遇到的问题及解决办法：
1. 多个按键在同一个 always 模块中检测上升/下降沿造成混乱（输出无规律跳动）。
   对功能进行拆分，尽量多用开关少用按键，检测值而非上升/下降沿。
2. 同一组参数不能在一个以上的模块中修改，否则会无法编译。
   多设置几组参数，用其中一个模块的输出作为另一个的输入，避免重复修改。

实验得到的启示：
1. 模块并不是分得越多、越精细越好，反之也不是越少、越集中越好，要根据实际情况具体分析，遇到困难时，可以选择变换一种划分方式来解决问题。
2. 不要害怕“返工”或推倒重做，盲目地试图在某一特定模块中通过少量调整解决一些看似小问题，结果可能耗费大量时间而没有效果，必要时还是应该重新设计大框架。

意见和建议：无。