篮球比赛计分器 FPGA 实验报告

一．FPGA 报告组成

本报告基于篮球比赛计分器实验完成。报告主要分为**六个部分**，分别为：**设计背景、设计思路、实现过程、特色拓展功能、心得体会及源程序附录**。其中核心内容为设计思路、特色拓展功能及心得体会。**仿真文件**附在实现过程这一板块中。在本实验中，作者共开发出了**四项拓展功能**，较为创新的对本实验进行了完善，希望老师们在本报告中对此部分进行重点阅读，并予以批评指正。

二．设计背景

篮球是一项普及度较高的全民运动，篮球比赛亦是随处可见。因此，联想本实验：用小脚丫开发板完成篮球比赛计分器，作者思考到不局限于题给的设计要求，而加入更多日常比赛中所需要的功能，更加贴合实际的篮球比赛，将 fpga 开发板真正的与日常生活融入起来，通过对 fpga 开发板的掌握，意在今后能够通过开发板编程实现解决实际问题、生产实际产品的能力。

三．设计思路

题目要求如图：

5、篮球比赛计分器（组号尾数 4/9）↵

设计要求：分别用 2 个 LED 灯表示 A 和 B 两个球队，组织一场进攻要 24 秒，用数码管分别显示 A 队和 B 队的进攻倒计时。用数码管能够分别显示 A 队、B 队的累计分值，进球可以加 2 分或者加 3 分，在罚球时，可以加 1 分。用 4 个 led 灯分别表示篮球比赛的 4 个小节，能够用数码管显示 1 小节比赛的总时间（自行设定），打完 1 小节对应的 led 灯灭。(自行设计)↵

如上图所示，由于本实验中显示要素过多，同时需要多个计时器分别对 A、B 两队的进攻时长、比赛的总时长进行计时，还需要通过按键切换显示不同队伍的累计分值等，作者考虑到模块化编程，不仅可以增加代码的可读性，还可以简化编程思路，使流程更加清晰。

1. 首先，作者将本实验代码分为四个例化程序，分别为 segment（16 进制数码管显示模块）、key_debounce（按键消抖模块）、Display（A、B 两队计分、进攻倒计时等主要功能集成模块）及 BallCompetition（Top 文件，对以上三个例化文件进行调用）。这样进行分板块编程，让编程思路更加清晰，更有条理。
2. 其次，作者将已有的成品代码：segment（16 进制数码管显示模块）、key_debounce（按键消抖模块）分别进行了程序例化，便于后续进行调用。

```
`timescale 1ns / 1ps
//基本的数码管显示模块
module segment(
    input [3:0] seg_data_1,
    input [3:0] seg_data_2,
    output [8:0] seg_led_1,
    output [8:0] seg_led_2
);

reg [8:0] seg [16:0];

initial

   begin
       seg[0] = 9'h3f;
       seg[1] = 9'h06;
       seg[2] = 9'h5b;
       seg[3] = 9'h4f;
       seg[4] = 9'h66;
       seg[5] = 9'h6d;
       seg[6] = 9'h7d;
       seg[7] = 9'h07;
       seg[8] = 9'h7f;
       seg[9] = 9'h6f;
       seg[10] = 9'h77;   //A
       seg[11] = 9'h7c;   //B
       seg[12] = 9'h39;
       seg[13] = 9'h5e;
       seg[14] = 9'h79;
       seg[15] = 9'h71;
       seg[16] = 9'h00;  //OFF
     end

   assign seg_led_1 = seg[seg_data_1];
   assign seg_led_2 = seg[seg_data_2];

endmodule
```

（segment 16 进制数码管显示模块）

```v
`timescale 1ns / 1ps
//按键消抖模块
module key_debounce #(
    parameter   DELAY_TIME = 250_000
)(
        input wire  clk,
        input wire  rstn,
        input wire  key_in,
        output reg  press_on
);

  reg       key_in_r0;
  reg       key_in_r1;
  reg       key_in_nedge;
  reg   [19:0]delay_cnt;
  reg       delay_flag;    //延时标志位

  always @(posedge clk or negedge rstn)
begin
    if(rstn == 1'b0) begin
        key_in_r0 <= 1'b0  ;
        key_in_r1 <= 1'b0  ;
    end
    else begin
       key_in_r0 <= key_in;
      key_in_r1 <= key_in_r0;
    end
end

  always @(posedge clk or negedge rstn)
begin
    if(rstn == 1'b0) begin
        key_in_nedge <= 1'b0;
    end
    else begin
        key_in_nedge <= ~key_in_r0 & key_in_r1;
    end
end

  //delay_cnt 延时计数
  always @(posedge clk or negedge rstn)
begin
    if(rstn == 1'b0) begin
        delay_cnt  <= 'd0;
    end
    else begin
        if((delay_flag == 1) && (delay_cnt == DELAY_TIME - 1))
            delay_cnt <= 0;
        else if ((delay_flag == 1) && (delay_cnt < DELAY_TIME - 1))
            delay_cnt <= delay_cnt + 1;
        else
            delay_cnt <= 0;
    end
end

  //delay_flag
  always @(posedge clk or negedge rstn)
begin
    if(rstn == 1'b0) begin
    delay_flag <= 1'b0;    //若复位按键按下，则延时标志位置0
    end
    else begin
        if(key_in_nedge) begin   //按键下降沿到来
            delay_flag <= 1'b1;  //延时标志位置1
        end
        else if (delay_cnt == DELAY_TIME - 1) begin
            delay_flag <= 1'b0;
        end
        else
            delay_flag <= delay_flag;
    end
end
  //press_on
  always @(posedge clk or negedge rstn)
begin
    if(rstn == 1'b0) begin
        press_on <= 1'b0;
    end
    else begin
        if((delay_cnt == DELAY_TIME - 1) && (delay_flag == 1'b1)) begin    //判断按键是否按下
          press_on <= 1'b1;
        end
        else begin
            press_on <= 1'b0;
        end
    end
end

  endmodule

```

（key_debounce 按键消抖模块）

3. 再根据设计要求，分配好各个拨码开关、按键的功能，以及 led 灯、rgb 灯、数码管如何对信息进行显示，如下图所示。

   ① 拨码开关 sw_in[0]是计数暂停开关，控制计时暂停与否。

   ② 其余拨码开关 sw_in[3:1]分别显示：

      a) 011 时显示 A 队累计分值 scorer_A

b) 111 时显示 B 队累计分值 scorer_B

c) 000 时显示 A 队进攻秒数（倒计时）

d) 001 时显示 B 队进攻秒数（倒计时）

③ 三位按键 press_on[2:0]分别表示，按下后队伍总分分别加 1 分、2 分、3 分

④ press_on[3]是 rstn 复位按键，按下后所有显示、计数值回归初始化

⑤ reg_ab[1:0]是两位 RGB 灯，分别为绿色和蓝色，表示现在为 A 队 or B 队进攻或累计分值数

⑥ led_time[7:0]表示七位 led 灯，为了好看起见我们每两位代表一个小节的比赛，每经过一小节比赛，两位 led 灯熄灭，四节比赛结束后所有 led 灯均熄灭

⑦ seg_led_1 和 seg_led_2 分别为两位数码管，我们用十进制分别显示 A、B 两队的累计分值及进攻倒计时

```
module BallCompetition(
    input   wire clk,                //时钟
    input   wire rstn,               //复位
    input   wire  [3:0]  sw_in,      //4位拨码开关
    input   wire  [2:0]  key_in,     //3位投篮加分按键
    output  wire  [1:0]  reg_ab,     //两位rgb灯分别显示当前为A队orB队
    output  wire  [7:0]  led_time,   //8位led灯，表示当前为第几节，每结束一节，按顺序灭掉两位led灯
    output  [8:0]  seg_led_1,        //数码管1
    output  [8:0]  seg_led_2         //数码管2
);
```

4. 根据上述的思路，由于各队伍的 24s 进攻倒计时以及进球都是在时钟到来后完成，因此将以上功能分模块写在时钟触发函数中。如下图四小节比赛倒计时模块所示。

```
 /*每小节比赛过后 led灯熄灭两盏，且重新开始每小节倒计时*/
 always @(posedge clk or negedge rstn)
begin
    if(rstn == 1'b0) begin //若复位，则均归零
       led_time_reg <= 8'b0000_0000;
       Qcounter <= 'd0;//计数功能
       Quarter_Time_Down <= Quarter_Time;
       Quarter_Time_rest_Down <= Quarter_Time_rest;
       DownCNT <= 'd0;
    end
    else begin
        if(Qcounter == COUNT_1S - 1) begin
            Qcounter <= 'd0;
            if(Quarter_Time_Down == 0) begin
                  if(DownCNT == 2'd3) begin
                     DownCNT <= 2'd3;
                     Quarter_Time_Down <=  0;
                     Quarter_Time_rest_Down <= 0;
                     led_time_reg <= {led_time_reg[5:0],1'b1,1'b1};// 每次灭两盏灯
                  end
                  else begin
                     if(Quarter_Time_rest_Down == 0)begin
                        DownCNT <= DownCNT + 1'b1;
                        Quarter_Time_Down <=  Quarter_Time;
                        Quarter_Time_rest_Down <= Quarter_Time_rest;
                        led_time_reg <= {led_time_reg[5:0],1'b1,1'b1};// 每次灭两盏灯
                     end
                     else begin
                        Quarter_Time_rest_Down <= Quarter_Time_rest_Down -1'b1;
                     end
                  end

              end
              else begin
                 Quarter_Time_Down <= Quarter_Time_Down - 1'b1;
              end

        end
        else begin
           Qcounter <= Qcounter + 1;//计数器
        end
    end
end
```

5. 根据上述思路完成完整实验。

四．实现过程

每一个模块，按照一下思路进行实现：

1. Clk 时钟上升沿或 rstn 下降沿触发进入模块
2. 首先检验 rstn 复位按键是否按下，若按下，则所有显示、计数均归零
3. 其次，由 case endcase 语句，写入在不同拨码开关下（即切换不同模式下的按键、rgb、数码管显示内容）。如在 A、B 累计计分模块中，sw_in[3:1]拨码开关拨至 011 和 111 分别显示 A、B 队的累计分值，而在不同的拨码开关下，嵌套 case endcase 语句，写入按下不同的按键时，计分不同，如按下 key1，key2，key3 分别加分为 1 分、2 分和 3 分。

```
85   //A B Score
86   always @(posedge clk or negedge rstn)
87  begin
88      if(rstn == 1'b0) begin
89          scorer_A <= 'd0;
90          scorer_B <= 'd0;
91          countA_1 <= 'd0;countA_2 <= 'd0;countA_3 <= 'd0;
92          countB_1 <= 'd0;countB_2 <= 'd0;countB_3 <= 'd0;
93      end
94      else begin
95          case(sw_in[3:1])
96              3'b011:begin //A Score
97                case(press_on) //key
98                  3'b001: begin
99                      scorer_A <= scorer_A + 1;
100                     countA_1 <= countA_1 + 1;
101                 end
102                 3'b010: begin
103                     scorer_A <= scorer_A + 2;
104                     countA_2 <= countA_2 + 1;
105                 end
106                 3'b100: begin
107                     scorer_A <= scorer_A + 3;
108                     countA_3 <= countA_3 + 1;
109                 end
110                 default:begin
111                     scorer_A <= scorer_A;
112                 end
113               endcase
114             end
115             3'b111:begin //B Score
116                case(press_on)
117                 3'b001: begin
118                     scorer_B <= scorer_B + 1;
119                     countB_1 <= countB_1 + 1;
120                 end
121                 3'b010: begin
122                     scorer_B <= scorer_B + 2;
123                     countB_2 <= countB_2 + 1;
124                 end
125                 3'b100: begin
126                     scorer_B <= scorer_B + 3;
127                     countB_3 <= countB_3 + 1;
128                 end
129                 default:begin
130                     scorer_B <= scorer_B;
131                 end
132               endcase
133             end
134             default:begin
135                 scorer_A <= scorer_A;
136                 scorer_B <= scorer_B;
137             end
138         endcase
139     end
140 end
```

4. 对于 A、B 两队的进攻倒计时模块，拨码开关写入 000 为 A 队进攻倒计时模块，写入 001 为 B 队进攻倒计时模块。在各自的倒计时模块中，共用一个计时器，每个时钟上升沿到来时计数器加一，当计数器每加到 COUNT_1s 时，计数器清零同时 AttackT_A 减 1，当 AttackT_A 减至 0 时，进攻倒计时结束。进攻倒计时模块如下图所示。

```
142   // A B 进攻倒计时24s
143   always @(posedge clk or negedge rstn)
144  begin
145      if(rstn == 1'b0) begin
146          AttackT_A <= 5'd24;
147          AttackT_B <= 5'd24;
148          counter <=  'd0;
149      end
150      else begin
151          case(sw_in[3:1])
152        3'b000:begin   //A 24S
153                if(sw_in[0] == 1'b1) begin
154                      if(counter == COUNT_1S - 1) begin
155                              counter <= 'd0;
156                              if(AttackT_A == 0) begin
157                                  AttackT_A <= 0 ;
158                              end
159
160                              else begin
161                                 AttackT_A <= AttackT_A - 1'b1;
162                              end
163                      end
164                      else begin
165                         counter <= counter + 1;
166                      end
167              end
168              else begin
169                 counter <= counter;
170              end
171        end
172        3'b001:begin //B 24S
173                if(sw_in[0] == 1'b1) begin
174                      if(counter == COUNT_1S - 1) begin
175                              counter <= 'd0;
176                              if(AttackT_B == 0) begin
177                                  AttackT_B <= 0 ;
178                              end
179                              else begin
180                                 AttackT_B <= AttackT_B - 1'b1;
181                              end
182                      end
183                      else begin
184                         counter <= counter + 1;
185                      end
186              end
187              else begin
188                 counter <= counter;
189              end
190        end
191        3'b011:begin //A Score
192             counter <=  counter;
193             AttackT_A <= AttackT_A;
194             AttackT_B <= AttackT_B;
195        end
196        3'b111:begin //B Score
197             counter <=  counter;
198             AttackT_A <= AttackT_A;
199             AttackT_B <= AttackT_B;
200        end
201        default:begin
202              counter <=  'd0;
203              AttackT_A <= 5'd24;
204              AttackT_B <= 5'd24;
205         end
206        endcase
207      end
208  end
209
```

5. 集成 RGB+SEG 显示模块的思路较为简单，即为不断的 case endcase 语句进行嵌套，当拨码开关拨到不同的码数时，数码管或 rgb 灯给予不同的显示。此处便不再展示。
6. 在写完程序后，作者先使用 modelsim 软件进行了仿真，仿真结果说明了本程序能完整的、正确的完成设计要求并且有效完成扩展功能，简单编写 testbench 文件对本程序进行仿真，仿真图如下所示。

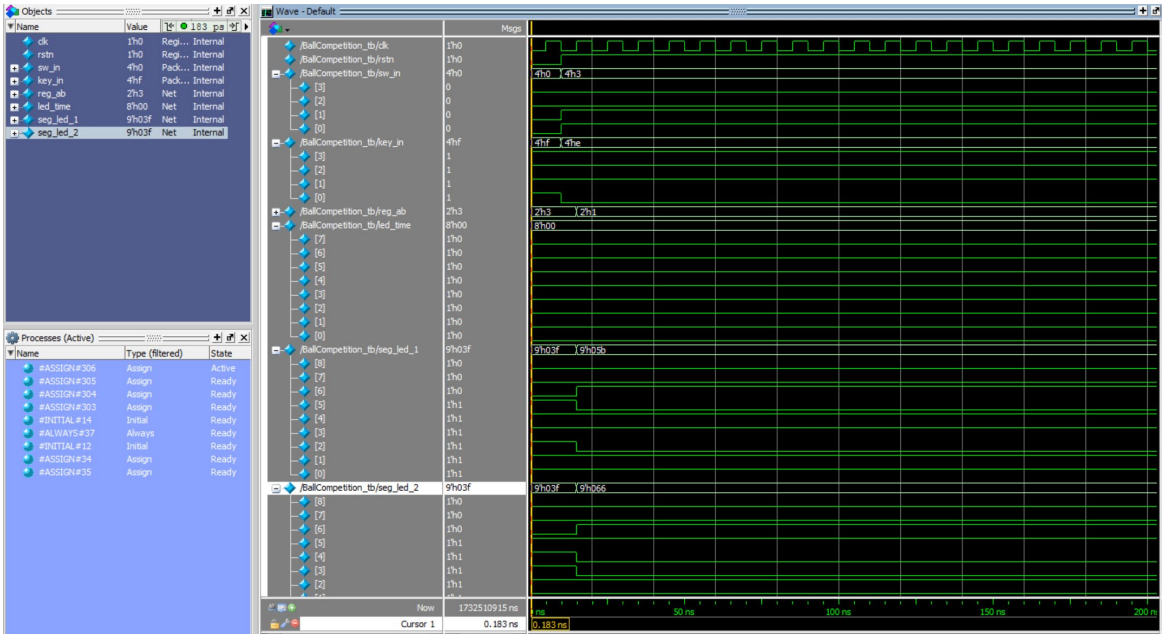
```
`timescale 1ns / 1ps

module BallScoreTOP_tb();
reg           clk;
reg         rstn ;
reg    [3:0]   sw_in   ;
reg  [3:0]   key_in  ;
wire [1:0]  reg_ab ;
wire [7:0]  led_time;
wire  [8:0]  seg_led_1;
wire  [8:0]  seg_led_2;


initial
begin
    clk <= 0;
    rstn <=0;
    sw_in <=  4'b0000;
    key_in <= 4'b1111;
    #10
    rstn <=1;
    sw_in <=  4'b0011;
    key_in <= 4'b1110;
    #300
    key_in <= 4'b1111;
    #20;
    key_in <= 4'b1101;
    #300
    key_in <= 4'b1111;
    #20;
    key_in <= 4'b1011;
    #300
    key_in <= 4'b1111;
    #20;
end

always #5 clk <= ~clk;

BallScoreTOP u_BallScoreTOP(
    .clk            (clk)   ,
    .rstn          (rstn       )  ,
    .sw_in        (sw_in)   ,
    .key_in          (key_in)   ,
    .reg_ab         (reg_ab)  ,
    .led_time         (led_time) ,
    .seg_led_1       (seg_led_1),
   .seg_led_2      (seg_led_2)
    );
endmodule
```



五．特色扩展功能

1. WIN_Time：若某队获胜，则代表该队的 RGB 亮灯，同时数码管 seg_led_2 用十六进制数显示代表该队的字母。如 A 队获胜，则 RGB1 亮绿灯，同时 seg_led_2 显示 A 字样；若 A、B 两队分数相同，则数码管显示 0，RGB1、2 均不亮。

```
end
3'b100:begin  //拓展1：WIN Time 若某队获胜，则该队的RGB亮灯，且数码管显示为该队的字母，否则不亮
        if(scorer_A > scorer_B) begin
           seg_data_2_reg <= 4'd10;  //A
           seg_data_1_reg <= scorer_A-scorer_B;  //扩展3：显示A,B两队的比分差
           reg_ab_reg <= 2'b10;
        end
        else if(scorer_A < scorer_B) begin
           seg_data_2_reg <= 4'd11;  //B
           seg_data_1_reg <= scorer_B-scorer_A;  //比分差
           reg_ab_reg <= 2'b01;
        end
        else begin  //打成平局
           seg_data_2_reg <= 4'd00;
           seg_data_1_reg <= 4'd00;  //A,B分数清零
           reg_ab_reg <= 2'b00;
        end
end
```

2. 显示 A、B 两队比分差：比赛结束时，若某队获胜，则 seg_led_1 显示两队的比分差。代码如上图所示。
3. 可对四小节比赛进行连续计时，两节比赛之间进行休息时间倒计时：如要进行四小节比赛，设置每小节比赛时长 Quarter_Time=60s,设置两节比赛之间队员休息时长 Quarter_Time_rest=10s，则数码管可进行连续计时，且每小节比赛结束后，8 位 led 灯熄灭两位，直至四小节比赛结束后，led 灯全部熄灭。

```
);
localparam COUNT_1S = 12048193; //1S
localparam Quarter_Time = 60;     //扩展2：可连续进行四节60s+10s的比赛 60s为每一节的时长，10s为中场休息时长
localparam Quarter_Time_rest = 10;  //rest时间
//localparam COUNT_1S = 120; //simulation time
//localparam Quarter_Time = 60; //simulation time

wire clk1h;
reg [23:0] counter ;//计数器
reg [23:0] Qcounter ;
reg [6:0] scorer_A ;
reg [6:0] scorer_B ;
reg [6:0] Quarter_Time_Down ; //每小节比赛倒计时
reg [2:0] Quarter_Time_rest_Down ; //各小节比赛之间的休息时间倒计时
reg [4:0] AttackT_A ;
```

```
/*每小节比赛过后 led灯熄灭两盏，且重新开始每小节倒计时*/
always @(posedge clk or negedge rstn)
begin
    if(rstn == 1'b0) begin //若复位，则均归零
       led_time_reg <= 8'b0000_0000;
       Qcounter <= 'd0;//计数功能
       Quarter_Time_Down <= Quarter_Time;
       Quarter_Time_rest_Down <= Quarter_Time_rest;
       DownCNT <= 'd0;
    end
    else begin
        if(Qcounter == COUNT_1S - 1) begin
            Qcounter <= 'd0;
            if(Quarter_Time_Down == 0) begin
                  if(DownCNT == 2'd3) begin
                     DownCNT <= 2'd3;
                     Quarter_Time_Down <=  0;
                     Quarter_Time_rest_Down <= 0;
                     led_time_reg <= {led_time_reg[5:0],1'b1,1'b1};// 每次灭两盏灯
                  end
                  else begin
                     if(Quarter_Time_rest_Down == 0)begin
                        DownCNT <= DownCNT + 1'b1;
                        Quarter_Time_Down <=  Quarter_Time;
                        Quarter_Time_rest_Down <= Quarter_Time_rest;
                        led_time_reg <= {led_time_reg[5:0],1'b1,1'b1};// 每次灭两盏灯
                     end
                     else begin
                        Quarter_Time_rest_Down <= Quarter_Time_rest_Down -1'b1;
                     end
                  end

             end
             else begin
                Quarter_Time_Down <= Quarter_Time_Down - 1'b1;
             end

        end
        else begin
           Qcounter <= Qcounter + 1;//计数器
        end
    end
end
```

4. 可显示 A、B 队分别进了多少个 1 分球、2 分球、3 分球：将拨码开关 sw_in[2:0] 分别写入 010 和 101 可进入 A、B 队的比赛详情显示模式，在这种模式中，将按键 press_on[3:1]按下 001、010、100 可分别查看本次比赛中各队分别进了多少个 1 分球、2 分球和 3 分球。

```
32  reg [1:0] DownCNT;    //记录目前是第几节比
33  reg [6:0] countA_1;   //A投篮次数 1分球
34  reg [6:0] countA_2;   //A投篮次数 2分球
35  reg [6:0] countA_3;   //A投篮次数 3分球
36  reg [6:0] countB_1;   //B投篮次数 1分球
37  reg [6:0] countB_2;   //B投篮次数 2分球
38  reg [6:0] countB_3;   //B投篮次数 3分球
39
```

```
        scorer_A <= 'd0;
        scorer_B <= 'd0;
        countA_1 <= 'd0;countA_2 <= 'd0;countA_3 <= 'd0;
        countB_1 <= 'd0;countB_2 <= 'd0;countB_3 <= 'd0;
    end
    else begin
        case(sw_in[3:1])
            3'b011:begin //A Score
              case(press_on) //key
                3'b001: begin
                    scorer_A <= scorer_A + 1;
                    countA_1 <= countA_1 + 1;
                end
                3'b010: begin
                    scorer_A <= scorer_A + 2;
                    countA_2 <= countA_2 + 1;
                end
                3'b100: begin
                    scorer_A <= scorer_A + 3;
                    countA_3 <= countA_3 + 1;
                end
                default:begin
                    scorer_A <= scorer_A;
                end
              endcase
            end
            3'b111:begin //B Score
               case(press_on)
                3'b001: begin
                    scorer_B <= scorer_B + 1;
                    countB_1 <= countB_1 + 1;
                end
                3'b010: begin
                    scorer_B <= scorer_B + 2;
                    countB_2 <= countB_2 + 1;
                end
                3'b100: begin
                    scorer_B <= scorer_B + 3;
                    countB_3 <= countB_3 + 1;
                end
                default:begin
                    scorer_B <= scorer_B;
                end

    3'b010:begin
    case(press_on) //扩展4：显示队伍A,B分别进了多少了1分球，2分球，3分球
          3'b001:begin
             seg_data_2_reg <= countA_1%10;
             seg_data_1_reg <= countA_1/10;end
          3'b010:begin
             seg_data_2_reg <= countA_2%10;
             seg_data_1_reg <= countA_2/10;end
          3'b100:begin
             seg_data_2_reg <= countA_3%10;
             seg_data_1_reg <= countA_3/10;end
          endcase
    end
    3'b101:begin
          case(press_on)
          3'b001:begin
             seg_data_2_reg <= countB_1%10;
             seg_data_1_reg <= countB_1/10;end
          3'b010:begin
             seg_data_2_reg <= countB_2%10;
             seg_data_1_reg <= countB_2/10;end
          3'b100:begin
             seg_data_2_reg <= countB_3%10;
             seg_data_1_reg <= countB_3/10;end
          endcase
    end
```

六．心得体会

在本次实验中，作者遇到了重重困难。好在最后都一一解决，并且仍旧有效的完成了扩展功能。作者主要的心得体会如下：

1. 在开始着手写程序之前，应当先理清思路，将编程逻辑理出来。在本实验中，即应当将程序分成不同的模块进行思考，提前例化按键消抖模块、数码管显示模块，将主功能集成在 Display 文件中，使程序更加直观。
2. 本次编程的主要困难是对 Verilog 语言的不熟悉。如在对不同拨码开关和按键控制不同模式下的功能时，作者一开始并未想到使用 case endcase 语句，因而绕了大弯，使代码不简洁的同时也让自己的思路变得混乱。
3. 对于计数器的使用上，要明确计数器计数的机理。因为计数器当且仅当时钟上升沿触发时进行计数，而同一个计数器仅由一个时钟上升沿进行触发，因此面对计数器计数时需要设定中间变量来对每小节比赛倒计时至 0 时的计数进行更新，而不能直接更新原计数器。这也是使作者困惑了好久的一个重点。
4. 对于数码管的显示，使用两个数码管主要应用十进制进行显示，则在输入数据时则可采用取余和整除的方法，或者用某一段数据的前几位和后几位分别代表是十位和个位进行显示。在本实验中，由于数字均为两位数，则采用前者明显更为明智。

七．源程序附录（见附件）

1. BallCompetition.v
2. segment.v
3. key_debounce.v
4. Display.v