同济大学

本科实验报告

FPGA 秒表功能实现

课程名称:	电子技术实验（下）
姓名:	余政希/惠仪仁
学院:	国豪书院
专业:	人工智能/通信工程
学号:	**2452633/2453574**
指导老师:	周伟

2025 年 12 月 28 日

同濟大學实验报告

专业：	人工智能/通信工程
姓名：	余政希/惠仪仁
学号：	2452633/2453574
日期：	2025 年 12 月 28 日
地点：	阜新路 281-203

课程名称：	电子技术实验（下）	指导老师：	周伟	成绩：	
实验名称：	FPGA 秒表功能实现	实验类型：	综合实验	同组学生姓名：	无

一、 实验目的和要求

基于 FPGA 实现多功能秒表系统，满足以下要求与扩展功能：

(1) 通过按键 S2 启动秒表计时，S0 停止秒表计时，通过数码管显示计时时间，计时范围：00.00.00-59.59.99，秒表分辨率为 10ms，可以通过 RESET 按键复位计时器；（1 分）

(2) 通过 LED0-LED7 不同的花样灯显示计时、停止状态；（1 分）

(3) 秒表可以实现定时功能，从预定时间倒计时到 0，并通过 LED0-LED7 进行状态指示；（1 分）

(4) 能够实现秒表跑圈功能，每按一次 S4 按键，记录下当前秒表时间, 但是不停止计数；计时结束后（S0 按钮）可以查阅每圈秒表读数，需要至少保存四组数据；（1 分）；

二、 需求分析与功能分解

1. 功能需求

功能点	需求描述	设计实现方式
1	S2 启动、S0 停止、RESET 复位；显示范围 00.00.00–59.59.99，10ms 分辨率	100Hz 时钟 + BCD 计数
2	LED0–LED7 花样灯区分计时/停止状态	`led_pattern` 根据 running/done/-mode 输出不同灯效
3	定时功能：预设时间倒计时到 0，LED 指示	SW 预置（BCD MMSS）+ 倒计时逻辑 + done 标志
4	跑圈：S4 记录不停表，停止后浏览；至少 4 组	4 组循环存储 + 写指针/浏览指针

2. 硬件特性与关键约束

实验板数码管采用“两组段选 + 两组位选”结构（`seg1` 与 `seg2`，`an[3:0]` 与 `an[7:4]`）。因此显示驱动参考示例，采用“双 4 位并行扫描”。

三、　总体设计方案

1.　系统结构与信号流

```
             +----------------+
clk/reset ->| Reset Sync     |-> rst (sync)
             +--------+--------+
                      |
keys -> key_pulse ->| p_start/p_stop/p_load/p_mode/p_lap
                      |
                      v
             +----------------------------+
             | Core Control + Time Engine|
             | mode/running/done          |
             | BCD Up/Down counter (10ms)|
             | Lap RAM (4 entries)        |
             +-----------+----------------+
                          |
                          v
             +----------------------------+
             | Display Driver (Dual 4)    |
             | seg1/sel1 + seg2/sel2      |
             | dp_mask + digit mapping    |
             +-----------+----------------+
                          |
                          v
                  8-digit display

running/done/mode (different) -> led_pattern -> LED[7:0]
```

2.　数据表示与显示格式

系统内部采用 BCD 计数：分钟/秒/百分秒分别用 m_tens，m_ones，s_tens，s_ones，c_tens，c_ones 保存。显示格式为 MM.SS.CC，小数点位由 dp_mask=8'b0010_1000 控制，对应“分钟个位后”和“秒个位后”点亮。

四、　关键模块设计与实现

1.　复位同步（Reset Sync）

```
reg [1:0] rst_sync;
always @(posedge clk) rst_sync <= {rst_sync[0], ~reset_n};
wire rst = rst_sync[1]; // rst=1 同步复位
```

reset_n 为异步输入，通过两级触发器同步到时钟域，降低亚稳态风险。

2. 按键消抖与单脉冲（key_pulse）

```
localparam integer CNT_MAX = (CLK_HZ/1000)*DEBOUNCE_MS;
...
if (key_sync2 == stable) cnt <= 0;
else if (cnt == CNT_MAX-1) begin
  stable <= key_sync2;
  cnt    <= 0;
end else cnt <= cnt + 1;
assign pulse = debounced & ~debounced_d;
```

按键先同步，再以 20ms 计数稳定判定，最终输出单周期脉冲，保证一次按键只触发一次控制事件。

3. 10ms 时基生成（tick_gen）

```
localparam integer DIV = CLK_HZ / TICK_HZ;
if (cnt == DIV-1) begin
  cnt  <= 0;
  tick <= 1'b1;
end else begin
  cnt  <= cnt + 1'b1;
  tick <= 1'b0;
end
```

参数化分频，默认 `CLK_HZ=100_000_000`（100MHz）时得到 100Hz 时基（10ms）。

4. 核心控制与 BCD 计时链

状态变量：`mode_timer`（0= 正计时，1= 倒计时）、`running`、`done`。模式切换策略：仅在停止状态切换模式，避免运行中跳变。

```
if (p_mode && !running) begin
  mode_timer <= ~mode_timer;
  done <= 1'b0;
end
```

正计时进位链（节选）：

```
if (tick_10ms && running && !mode_timer) begin
  if (c_ones != 9) c_ones <= c_ones + 1;
  else begin
    c_ones <= 0;
    if (c_tens != 9) c_tens <= c_tens + 1;
    else begin
      c_tens <= 0;
      if (s_ones != 9) s_ones <= s_ones + 1;
      else begin
        s_ones <= 0;
        if (s_tens != 5) s_tens <= s_tens + 1;
        else begin
          s_tens <= 0;
          if (m_ones != 9) m_ones <= m_ones + 1;
          else begin
            m_ones <= 0;
            if (m_tens != 5) m_tens <= m_tens + 1;
            else m_tens <= 0;
          end
        end
      end
    end
  end
end
```

倒计时借位链与 **done**：

```
if (tick_10ms && running && mode_timer) begin
  if ({m_tens,m_ones,s_tens,s_ones,c_tens,c_ones} == 24'h000000) begin
    running <= 1'b0;
    done    <= 1'b1;
  end else begin
    if (c_ones != 0) c_ones <= c_ones - 1;
    else begin
      c_ones <= 9;
      ... // 逐级借位
    end
  end
end
```

5. 预置时间与合法性裁剪

```
wire [3:0] sw_mt = (SW[15:12] > 4'd5) ? 4'd5 : SW[15:12];
wire [3:0] sw_mo = (SW[11:8]  > 4'd9) ? 4'd9 : SW[11:8];
wire [3:0] sw_st = (SW[7:4]   > 4'd5) ? 4'd5 : SW[7:4];
wire [3:0] sw_so = (SW[3:0]   > 4'd9) ? 4'd9 : SW[3:0];

if (p_load && mode_timer && !running) begin
  m_tens <= sw_mt; m_ones <= sw_mo;
  s_tens <= sw_st; s_ones <= sw_so;
  c_tens <= 0;     c_ones <= 0;
  done   <= 1'b0;
end
```

仅在倒计时模式且停止时装载，保证输入合法且可控。

6. 跑圈记录与回放

```
if (p_lap) begin
  if (running) begin
    lap_m_tens[lap_wr_idx] <= m_tens;
    ...
    lap_wr_idx <= lap_wr_idx + 1'b1;
    if (lap_count < 3'd4) lap_count <= lap_count + 3'd1;
  end else if (lap_count != 0) begin
    if (!viewing_lap) begin
      viewing_lap  <= 1'b1;
      lap_view_idx <= 2'd0;
    end else begin
      if (lap_view_idx == (lap_count - 1)) lap_view_idx <= 2'd0;
      else lap_view_idx <= lap_view_idx + 2'd1;
    end
  end
end

wire [1:0] lap_rd_idx = lap_wr_idx - 2'd1 - lap_view_idx;
```

运行中写入、停止后回放；显示时以“最近一次记录”为起点循环浏览。

7. 显示驱动（双 4 位并行扫描）

顶层将 6 位时间映射到 8 位显示：

```
wire [3:0] dig7 = dc_ones;
wire [3:0] dig6 = dc_tens;
wire [3:0] dig5 = ds_ones;
wire [3:0] dig4 = ds_tens;
wire [3:0] dig3 = dm_ones;
wire [3:0] dig2 = dm_tens;
wire [3:0] dig1 = 4'hF;
wire [3:0] dig0 = 4'hF;
```

disp8_dual4 内部产生扫描时钟，分别驱动低 4 位与高 4 位段线，并用 seg_decoder_dp 完成译码；4'hF 被译码为空白显示，保证两位空屏。

8. LED 花样灯（状态可视化）

```
if (done) begin
  led = slow ? 8'hFF : 8'h00;   // DONE: 全闪
end else if (running) begin
  led = (8'b0000_0001 << pos); // RUN: 流水
end else if (!mode) begin
  led = slow ? 8'hAA : 8'h55;   // STOP + 正计时: 交替慢闪
end else begin
  led = stop_pat;               // STOP + 倒计时: 顺序亮灭
end
```

cnt[26] 生成慢闪节拍，cnt[22] 驱动流水/顺序亮灭效果，实现可视化状态提示。

五、　管脚约束

根据 stopwatch_top.xdc 完成管脚约束：clk、reset_n、五个按键、16 位拨码 SW[15:0]、数码管位选 an[7:0]、双段选 seg1/seg2 以及 led[7:0]。

六、　总结与可扩展方向

本设计实现了秒表、倒计时与跑圈功能，计时精度与显示稳定性满足实验要求。后续可扩展方向包括：增加更多跑圈存储、倒计时完成警报提示（蜂鸣器/PWM）与串口数据输出等。

A　附录：完整源代码

1.　src/stopwatch_top.v

```
`timescale 1ns/1ps

module stopwatch_top #(
    parameter integer CLK_HZ = 100_000_000
)(
    input  wire        clk,
    input  wire        reset_n,
    input  wire        S0_stop,
    input  wire        S1_load,
    input  wire        S2_start,
    input  wire        S3_mode, // 0 正计时，1 倒计时
    input  wire        S4_lap,
    input  wire [15:0] SW,  // 预置时间输入（BCD: MM,SS）
    output wire [7:0]  led,
    output wire [7:0]  seg1,  // 段选 a b c d e f g dp
    output wire [7:0]  seg2,
    output wire [7:0]  an  // 位选
);

    reg [1:0] rst_sync;
    always @(posedge clk) rst_sync <= {rst_sync[0], ~reset_n};
    wire rst = rst_sync[1]; // rst=1 同步复位

    wire p_stop, p_start, p_load, p_mode, p_lap;
    key_pulse #(.CLK_HZ(CLK_HZ)) u_k0 (.clk(clk), .rst(rst), .key_in(S0_stop),
        .pulse(p_stop));
    key_pulse #(.CLK_HZ(CLK_HZ)) u_k1 (.clk(clk), .rst(rst), .key_in(S1_load),
        .pulse(p_load));
    key_pulse #(.CLK_HZ(CLK_HZ)) u_k2 (.clk(clk), .rst(rst), .key_in(S2_start),
        .pulse(p_start));
    key_pulse #(.CLK_HZ(CLK_HZ)) u_k3 (.clk(clk), .rst(rst), .key_in(S3_mode),
        .pulse(p_mode));
    key_pulse #(.CLK_HZ(CLK_HZ)) u_k4 (.clk(clk), .rst(rst), .key_in(S4_lap),
        .pulse(p_lap));
```

```
wire tick_10ms;
tick_gen #(.CLK_HZ(CLK_HZ), .TICK_HZ(100)) u_tick (
    .clk(clk), .rst(rst), .tick(tick_10ms)
);

reg mode_timer;  // 0=stopwatch up, 1=timer down
reg running;
reg done;    // timer 到0标志

// 当前计时值（BCD）
reg [3:0] m_tens, m_ones, s_tens, s_ones, c_tens, c_ones;

// lap 存储（4组循环）
reg [3:0] lap_m_tens [0:3];
reg [3:0] lap_m_ones [0:3];
reg [3:0] lap_s_tens [0:3];
reg [3:0] lap_s_ones [0:3];
reg [3:0] lap_c_tens [0:3];
reg [3:0] lap_c_ones [0:3];

reg [1:0] lap_wr_idx;   // 写入lap索引 0..3
reg [2:0] lap_count;
reg [1:0] lap_view_idx;
reg       viewing_lap;   // 0显示当前计时; 1显示lap

// 预置时间合法性裁剪（分钟十位/秒十位<=5）
wire [3:0] sw_mt = (SW[15:12] > 4'd5) ? 4'd5 : SW[15:12];
wire [3:0] sw_mo = (SW[11:8]  > 4'd9) ? 4'd9 : SW[11:8];
wire [3:0] sw_st = (SW[7:4]   > 4'd5) ? 4'd5 : SW[7:4];
wire [3:0] sw_so = (SW[3:0]   > 4'd9) ? 4'd9 : SW[3:0];

integer i;
always @(posedge clk) begin
    if (rst) begin
        mode_timer  <= 1'b0;
        running     <= 1'b0;
        done        <= 1'b0;

        m_tens <= 0; m_ones <= 0; s_tens <= 0; s_ones <= 0; c_tens <= 0; c_ones
            <= 0;

        lap_wr_idx <= 0; lap_count <= 0; lap_view_idx <= 0;
        viewing_lap <= 1'b0;

        for (i=0; i<4; i=i+1) begin
            lap_m_tens[i] <= 0; lap_m_ones[i] <= 0; lap_s_tens[i] <= 0;
                lap_s_ones[i] <= 0; lap_c_tens[i] <= 0; lap_c_ones[i] <= 0;
        end
    end else begin
```

```
// 仅在停止时模式切换
if (p_mode && !running) begin
    mode_timer <= ~mode_timer;
    done <= 1'b0;
end

if (p_start) begin
    if (!done) begin
        running     <= 1'b1;
        viewing_lap<= 1'b0;
    end
end
if (p_stop) begin
    running      <= 1'b0;
    viewing_lap  <= 1'b0;
end

if (p_load && mode_timer && !running) begin
    m_tens <= sw_mt; m_ones <= sw_mo;  s_tens <= sw_st; s_ones <= sw_so;
        c_tens <= 0; c_ones <= 0;
    done <= 1'b0;
end

// Lap: 运行中记录; 停止后查阅
if (p_lap) begin
    if (running) begin
        lap_m_tens[lap_wr_idx] <= m_tens;
        lap_m_ones[lap_wr_idx] <= m_ones;
        lap_s_tens[lap_wr_idx] <= s_tens;
        lap_s_ones[lap_wr_idx] <= s_ones;
        lap_c_tens[lap_wr_idx] <= c_tens;
        lap_c_ones[lap_wr_idx] <= c_ones;

        lap_wr_idx <= lap_wr_idx + 1'b1;
        if (lap_count < 3'd4) lap_count <= lap_count + 3'd1;
    end else begin
        if (lap_count != 0) begin
            if (!viewing_lap) begin
                viewing_lap  <= 1'b1;
                lap_view_idx <= 2'd0;
            end else begin
                if (lap_view_idx == (lap_count - 1)) lap_view_idx <= 2'd0;
                else lap_view_idx <= lap_view_idx + 2'd1;
            end
        end
    end
end

if (tick_10ms && running) begin
    if (!mode_timer) begin
```

```
            if (c_ones != 9) c_ones <= c_ones + 1;
            else begin
                c_ones <= 0;
                if (c_tens != 9) c_tens <= c_tens + 1;
                else begin
                    c_tens <= 0;
                    if (s_ones != 9) s_ones <= s_ones + 1;
                    else begin
                        s_ones <= 0;
                        if (s_tens != 5) s_tens <= s_tens + 1;
                        else begin
                            s_tens <= 0;
                            if (m_ones != 9) m_ones <= m_ones + 1;
                            else begin
                                m_ones <= 0;
                                if (m_tens != 5) m_tens <= m_tens + 1;
                                else begin
                                    m_tens <= 0;
                                end
                            end
                        end
                    end
                end
            end
        end else begin
            if ({m_tens,m_ones,s_tens,s_ones,c_tens,c_ones} == 24'h000000)
               begin
                running <= 1'b0;
                done    <= 1'b1;
            end else begin
                if (c_ones != 0) c_ones <= c_ones - 1;
                else begin
                    c_ones <= 9;
                    if (c_tens != 0) c_tens <= c_tens - 1;
                    else begin
                        c_tens <= 9;
                        if (s_ones != 0) s_ones <= s_ones - 1;
                        else begin
                            s_ones <= 9;
                            if (s_tens != 0) s_tens <= s_tens - 1;
                            else begin
                                s_tens <= 5;
                                if (m_ones != 0) m_ones <= m_ones - 1;
                                else begin
                                    m_ones <= 9;
                                    if (m_tens != 0) m_tens <= m_tens - 1;
                                    else m_tens <= 0;
                                end
                            end
                        end
                        end
                    end
                end
            end
        end

        if (mode_timer && ({m_tens,m_ones,s_tens,s_ones,c_tens,c_ones} ==
            24'h000000)) begin
            if (running) begin
                running <= 1'b0;
                done    <= 1'b1;
            end
        end
    end
end

wire [1:0] lap_rd_idx = lap_wr_idx - 2'd1 - lap_view_idx;
reg [3:0] dm_tens, dm_ones, ds_tens, ds_ones, dc_tens, dc_ones;
always @(*) begin
    if (viewing_lap) begin
        dm_tens = lap_m_tens[lap_rd_idx];
        dm_ones = lap_m_ones[lap_rd_idx];
        ds_tens = lap_s_tens[lap_rd_idx];
        ds_ones = lap_s_ones[lap_rd_idx];
        dc_tens = lap_c_tens[lap_rd_idx];
        dc_ones = lap_c_ones[lap_rd_idx];
    end else begin
        dm_tens = m_tens; dm_ones = m_ones;
        ds_tens = s_tens; ds_ones = s_ones;
        dc_tens = c_tens; dc_ones = c_ones;
    end
end


wire [7:0] dp_mask = 8'b0010_1000; // 第2位、第4位显示小数（倒置）
wire [3:0] dig7 = dc_ones;
wire [3:0] dig6 = dc_tens;
wire [3:0] dig5 = ds_ones;
wire [3:0] dig4 = ds_tens;
wire [3:0] dig3 = dm_ones;
wire [3:0] dig2 = dm_tens;
wire [3:0] dig1 = 4'hF;
wire [3:0] dig0 = 4'hF;

disp8_dual4 #(
    .CLK_HZ(CLK_HZ),
    .SCAN_DIV(250000)
) u_seg (
    .clk(clk),
    .rst_n(reset_n),
        .d0(dig0), .d1(dig1), .d2(dig2), .d3(dig3),
        .d4(dig4), .d5(dig5), .d6(dig6), .d7(dig7),
        .dp_mask(dp_mask),
        .seg1(seg1),
        .seg2(seg2),
        .an(an)
    );

    led_pattern #(.CLK_HZ(CLK_HZ)) u_led (
        .clk(clk), .rst(rst),
        .running(running),
        .done(done),
        .mode(mode_timer),
        .led(led)
    );

endmodule
```

2. src/seg_scan_8.v

```
module disp8_dual4 #(
    parameter integer CLK_HZ = 100_000_000,
    parameter integer SCAN_DIV = 250000
)(
    input  wire        clk,
    input  wire        rst_n,
    input  wire [3:0]  d0,d1,d2,d3,d4,d5,d6,d7,
    input  wire [7:0]  dp_mask, // dp_mask[i]=1 点亮第i位小数点

    output wire [7:0]  seg1,
    output wire [7:0]  seg2,
    output wire [7:0]  an
);

    reg [19:0] CNT;
    reg scan_clk;
    always @(posedge clk or negedge rst_n) begin
        if(!rst_n) begin
            CNT      <= 0;
            scan_clk <= 0;
        end else if(CNT == SCAN_DIV) begin
            CNT      <= 0;
            scan_clk <= ~scan_clk;
        end else begin
            CNT <= CNT + 1'b1;
        end
    end

    // 低4位（d0~d3）-> seg + an[3:0]
    wire [7:0] seg_lo;
    wire [3:0] sel1;
    seg_decoder_dp u_lo (
        .rst_n(rst_n),
        .clk(scan_clk),
        .dat({d3,d2,d1,d0}),
        .dp_mask(dp_mask[3:0]),
        .seg(seg_lo),
        .sel(sel1)
    );

    // 高4位 (d4~d7) -> seg_data_1_pin + an[7:4]
    wire [7:0] seg_hi;
    wire [3:0] sel2;
    seg_decoder_dp u_hi (
        .rst_n(rst_n),
        .clk(scan_clk),
        .dat({d7,d6,d5,d4}),
        .dp_mask(dp_mask[7:4]),
        .seg(seg_hi),
        .sel(sel2)
    );

    assign seg1 = seg_lo;
    assign seg2 = seg_hi;
    assign an[3:0] = sel1;
    assign an[7:4] = sel2;

endmodule


module seg_decoder_dp (
    input  wire        rst_n,
    input  wire        clk,
    input  wire [15:0] dat, // 4 BCD: {d3,d2,d1,d0}
    input  wire [3:0]  dp_mask,
    output reg  [7:0]  seg,  // [6:0]=a~g, [7]=dp
    output reg  [3:0]  sel
);

    reg [3:0] display_dat;
    reg dp_on;

    always @(posedge clk or negedge rst_n) begin
        if(!rst_n) begin
            sel <= 4'b0001;
        end else begin
            case(sel)
                4'b0001: sel <= 4'b0010;
                4'b0010: sel <= 4'b0100;
                4'b0100: sel <= 4'b1000;
                4'b1000: sel <= 4'b0001;
                default: sel <= 4'b0001;
            endcase
        end
    end

    always @(*) begin
        case(sel)
            4'b0001: begin display_dat = dat[3:0];   dp_on = dp_mask[0]; end
            4'b0010: begin display_dat = dat[7:4];   dp_on = dp_mask[1]; end
            4'b0100: begin display_dat = dat[11:8];  dp_on = dp_mask[2]; end
            4'b1000: begin display_dat = dat[15:12]; dp_on = dp_mask[3]; end
            default: begin display_dat = dat[3:0];   dp_on = 1'b0;       end
        endcase
    end

    always @(*) begin
        case (display_dat)
            4'h0: seg[6:0] = 7'b0111111;
            4'h1: seg[6:0] = 7'b0000110;
            4'h2: seg[6:0] = 7'b1011011;
            4'h3: seg[6:0] = 7'b1001111;
            4'h4: seg[6:0] = 7'b1100110;
            4'h5: seg[6:0] = 7'b1101101;
            4'h6: seg[6:0] = 7'b1111101;
            4'h7: seg[6:0] = 7'b0000111;
            4'h8: seg[6:0] = 7'b1111111;
            4'h9: seg[6:0] = 7'b1101111;
            4'hA: seg[6:0] = 7'b1110111;
            4'hB: seg[6:0] = 7'b1111100;
            4'hC: seg[6:0] = 7'b0111001;
            4'hD: seg[6:0] = 7'b1011110;
            4'hE: seg[6:0] = 7'b1111001;
            4'hF: seg[6:0] = 7'b0000000;
            default: seg[6:0] = 7'b0000000;
        endcase
        seg[7] = dp_on;
    end

endmodule
```

3. src/tick_gen.v

```
module tick_gen #(
    parameter integer CLK_HZ  = 100_000_000,
    parameter integer TICK_HZ = 100
)(
    input  wire clk,
    input  wire rst,
    output reg  tick
);
    localparam integer DIV = CLK_HZ / TICK_HZ;
    reg [$clog2(DIV)-1:0] cnt;

    always @(posedge clk) begin
        if (rst) begin
            cnt  <= 0;
            tick <= 1'b0;
        end else begin
            if (cnt == DIV-1) begin
                cnt  <= 0;
                tick <= 1'b1;
            end else begin
                cnt  <= cnt + 1'b1;
                tick <= 1'b0;
            end
        end
    end
endmodule
```

4. src/key_pulse.v

```
module key_pulse #(
    parameter integer CLK_HZ = 100_000_000,
    parameter integer DEBOUNCE_MS = 20
)(
    input  wire clk,
    input  wire rst,
    input  wire key_in,
    output wire pulse
);
    localparam integer CNT_MAX = (CLK_HZ/1000)*DEBOUNCE_MS;

    reg key_sync1, key_sync2;
    always @(posedge clk) begin
        key_sync1 <= key_in;
        key_sync2 <= key_sync1;
    end

    reg stable;
    reg [$clog2(CNT_MAX+1)-1:0] cnt;
    reg debounced;

    always @(posedge clk) begin
        if (rst) begin
            stable    <= 0;
            cnt       <= 0;
            debounced <= 0;
        end else begin
            if (key_sync2 == stable) begin
                cnt <= 0;
            end else begin
                if (cnt == CNT_MAX-1) begin
                    stable <= key_sync2;
                    cnt    <= 0;
                end else begin
                    cnt <= cnt + 1;
                end
            end
            debounced <= stable;
        end
    end

    reg debounced_d;
    always @(posedge clk) begin
        if (rst) debounced_d <= 0;
        else     debounced_d <= debounced;
    end

    assign pulse = debounced & ~debounced_d;

endmodule
```

5. src/led_pattern.v

```
module led_pattern #(
    parameter integer CLK_HZ = 100_000_000
)(
    input  wire clk,
    input  wire rst,
    input  wire running,
    input  wire done,
    input  wire mode,
    output reg  [7:0] led
);
    reg [31:0] cnt;
    always @(posedge clk) begin
        if (rst) cnt <= 32'd0;
        else     cnt <= cnt + 32'd1;
    end

    wire slow = cnt[26];
    wire mid  = cnt[22];

    reg mid_d;
    always @(posedge clk) begin
```

```
        if (rst) begin
            mid_d  <= 1'b0;
        end else begin
            mid_d  <= mid;
        end
    end

    wire mid_pulse  = mid  & ~mid_d;

    reg [2:0] pos;
    always @(posedge clk) begin
        if (rst) begin
            pos <= 3'd0;
        end else if (running && !done && mid_pulse) begin
            pos <= pos + 3'd1;
        end
    end

    reg [3:0] step;
    reg [7:0] stop_pat;
    always @(posedge clk) begin
        if (rst) begin
            step     <= 4'd0;
            stop_pat <= 8'h00;
        end else if (!running && !done && mode && mid_pulse) begin
            if (step < 4'd8) begin
                stop_pat <= (8'h01 << step) | stop_pat;
            end else begin
                stop_pat <= stop_pat & ~(8'h01 << (step - 4'd8));
            end
            step <= step + 4'd1;
        end else if (running || done || !mode) begin
            step     <= 4'd0;
            stop_pat <= 8'h00;
        end
    end

    always @(*) begin
        if (done) begin
            led = slow ? 8'hFF : 8'h00;  // DONE: 全闪
        end else if (running) begin
            led = (8'b0000_0001 << pos); // RUN: 流水
        end else if (!mode) begin
            led = slow ? 8'hAA : 8'h55;  // STOP + mode=0: 交替慢闪
        end else begin
            led = stop_pat;     // STOP + mode=1: 顺序亮灭
        end
    end

endmodule
```

6. src/stopwatch_top.xdc

```
############################ top_exp6.xdc ############################

############################ ?????±???????? ############################
set_property -dict {PACKAGE_PIN P17 IOSTANDARD LVCMOS33} [get_ports clk]
set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports reset_n]

############################ 5??°??ü btn_pin[0..4] ############################
set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVCMOS33} [get_ports S0_stop]
set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports S1_load]
set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports S2_start]
set_property -dict {PACKAGE_PIN V1  IOSTANDARD LVCMOS33} [get_ports S3_mode]
set_property -dict {PACKAGE_PIN U4  IOSTANDARD LVCMOS33} [get_ports S4_lap]

############################ ???????? SW[0..7] ############################
set_property -dict {PACKAGE_PIN P5 IOSTANDARD LVCMOS33} [get_ports {SW[0]}]
set_property -dict {PACKAGE_PIN P4 IOSTANDARD LVCMOS33} [get_ports {SW[1]}]
set_property -dict {PACKAGE_PIN P3 IOSTANDARD LVCMOS33} [get_ports {SW[2]}]
set_property -dict {PACKAGE_PIN P2 IOSTANDARD LVCMOS33} [get_ports {SW[3]}]
set_property -dict {PACKAGE_PIN R2 IOSTANDARD LVCMOS33} [get_ports {SW[4]}]
set_property -dict {PACKAGE_PIN M4 IOSTANDARD LVCMOS33} [get_ports {SW[5]}]
set_property -dict {PACKAGE_PIN N4 IOSTANDARD LVCMOS33} [get_ports {SW[6]}]
set_property -dict {PACKAGE_PIN R1 IOSTANDARD LVCMOS33} [get_ports {SW[7]}]
set_property -dict {PACKAGE_PIN U3 IOSTANDARD LVCMOS33} [get_ports {SW[8]}]
set_property -dict {PACKAGE_PIN U2 IOSTANDARD LVCMOS33} [get_ports {SW[9]}]
set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {SW[10]}]
set_property -dict {PACKAGE_PIN V5 IOSTANDARD LVCMOS33} [get_ports {SW[11]}]
set_property -dict {PACKAGE_PIN V4 IOSTANDARD LVCMOS33} [get_ports {SW[12]}]
set_property -dict {PACKAGE_PIN R3 IOSTANDARD LVCMOS33} [get_ports {SW[13]}]
set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVCMOS33} [get_ports {SW[14]}]
set_property -dict {PACKAGE_PIN T5 IOSTANDARD LVCMOS33} [get_ports {SW[15]}]

//////////////////////////////8个数码管位选信号////////////////////////////////
set_property -dict {PACKAGE_PIN G2 IOSTANDARD LVCMOS33} [get_ports {an[0]}]
set_property -dict {PACKAGE_PIN C2 IOSTANDARD LVCMOS33} [get_ports {an[1]}]
set_property -dict {PACKAGE_PIN C1 IOSTANDARD LVCMOS33} [get_ports {an[2]}]
set_property -dict {PACKAGE_PIN H1 IOSTANDARD LVCMOS33} [get_ports {an[3]}]
set_property -dict {PACKAGE_PIN G1 IOSTANDARD LVCMOS33} [get_ports {an[4]}]
set_property -dict {PACKAGE_PIN F1 IOSTANDARD LVCMOS33} [get_ports {an[5]}]
set_property -dict {PACKAGE_PIN E1 IOSTANDARD LVCMOS33} [get_ports {an[6]}]
set_property -dict {PACKAGE_PIN G6 IOSTANDARD LVCMOS33} [get_ports {an[7]}]

//////////////////////////////////////数码管段选信号//////////////////////////////////
set_property -dict {PACKAGE_PIN B4 IOSTANDARD LVCMOS33} [get_ports {seg1[0]}]
set_property -dict {PACKAGE_PIN A4 IOSTANDARD LVCMOS33} [get_ports {seg1[1]}]
set_property -dict {PACKAGE_PIN A3 IOSTANDARD LVCMOS33} [get_ports {seg1[2]}]
set_property -dict {PACKAGE_PIN B1 IOSTANDARD LVCMOS33} [get_ports {seg1[3]}]
set_property -dict {PACKAGE_PIN A1 IOSTANDARD LVCMOS33} [get_ports {seg1[4]}]
set_property -dict {PACKAGE_PIN B3 IOSTANDARD LVCMOS33} [get_ports {seg1[5]}]
set_property -dict {PACKAGE_PIN B2 IOSTANDARD LVCMOS33} [get_ports {seg1[6]}]
set_property -dict {PACKAGE_PIN D5 IOSTANDARD LVCMOS33} [get_ports {seg1[7]}]

set_property -dict {PACKAGE_PIN D4 IOSTANDARD LVCMOS33} [get_ports {seg2[0]}]
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {seg2[1]}]
set_property -dict {PACKAGE_PIN D3 IOSTANDARD LVCMOS33} [get_ports {seg2[2]}]
set_property -dict {PACKAGE_PIN F4 IOSTANDARD LVCMOS33} [get_ports {seg2[3]}]
set_property -dict {PACKAGE_PIN F3 IOSTANDARD LVCMOS33} [get_ports {seg2[4]}]
set_property -dict {PACKAGE_PIN E2 IOSTANDARD LVCMOS33} [get_ports {seg2[5]}]
set_property -dict {PACKAGE_PIN D2 IOSTANDARD LVCMOS33} [get_ports {seg2[6]}]
set_property -dict {PACKAGE_PIN H2 IOSTANDARD LVCMOS33} [get_ports {seg2[7]}]

############################# LED0~LED15 led[0..15] #############################
set_property -dict {PACKAGE_PIN F6 IOSTANDARD LVCMOS33} [get_ports {led[0]}]
set_property -dict {PACKAGE_PIN G4 IOSTANDARD LVCMOS33} [get_ports {led[1]}]
set_property -dict {PACKAGE_PIN G3 IOSTANDARD LVCMOS33} [get_ports {led[2]}]
set_property -dict {PACKAGE_PIN J4 IOSTANDARD LVCMOS33} [get_ports {led[3]}]
set_property -dict {PACKAGE_PIN H4 IOSTANDARD LVCMOS33} [get_ports {led[4]}]
set_property -dict {PACKAGE_PIN J3 IOSTANDARD LVCMOS33} [get_ports {led[5]}]
set_property -dict {PACKAGE_PIN J2 IOSTANDARD LVCMOS33} [get_ports {led[6]}]
set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVCMOS33} [get_ports {led[7]}]
set_property -dict {PACKAGE_PIN K1 IOSTANDARD LVCMOS33} [get_ports {led[8]}]
set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {led[9]}]
set_property -dict {PACKAGE_PIN H5 IOSTANDARD LVCMOS33} [get_ports {led[10]}]
set_property -dict {PACKAGE_PIN J5 IOSTANDARD LVCMOS33} [get_ports {led[11]}]
set_property -dict {PACKAGE_PIN K6 IOSTANDARD LVCMOS33} [get_ports {led[12]}]
set_property -dict {PACKAGE_PIN L1 IOSTANDARD LVCMOS33} [get_ports {led[13]}]
set_property -dict {PACKAGE_PIN M1 IOSTANDARD LVCMOS33} [get_ports {led[14]}]
set_property -dict {PACKAGE_PIN K3 IOSTANDARD LVCMOS33} [get_ports {led[15]}]

############################# 32?? PMOD/?????? exp_io[0..31]
    #############################
set_property -dict {PACKAGE_PIN B16 IOSTANDARD LVCMOS33} [get_ports {exp_io[0]} ]
set_property -dict {PACKAGE_PIN A15 IOSTANDARD LVCMOS33} [get_ports {exp_io[1]} ]
set_property -dict {PACKAGE_PIN A13 IOSTANDARD LVCMOS33} [get_ports {exp_io[2]} ]
set_property -dict {PACKAGE_PIN B18 IOSTANDARD LVCMOS33} [get_ports {exp_io[3]} ]
set_property -dict {PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports {exp_io[4]} ]
set_property -dict {PACKAGE_PIN B13 IOSTANDARD LVCMOS33} [get_ports {exp_io[5]} ]
set_property -dict {PACKAGE_PIN D14 IOSTANDARD LVCMOS33} [get_ports {exp_io[6]} ]
set_property -dict {PACKAGE_PIN B11 IOSTANDARD LVCMOS33} [get_ports {exp_io[7]} ]
set_property -dict {PACKAGE_PIN E15 IOSTANDARD LVCMOS33} [get_ports {exp_io[8]} ]
set_property -dict {PACKAGE_PIN D15 IOSTANDARD LVCMOS33} [get_ports {exp_io[9]} ]
set_property -dict {PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports {exp_io[10]}]
set_property -dict {PACKAGE_PIN F15 IOSTANDARD LVCMOS33} [get_ports {exp_io[11]}]
set_property -dict {PACKAGE_PIN H14 IOSTANDARD LVCMOS33} [get_ports {exp_io[12]}]
set_property -dict {PACKAGE_PIN E17 IOSTANDARD LVCMOS33} [get_ports {exp_io[13]}]
set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33} [get_ports {exp_io[14]}]
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {exp_io[15]}]
set_property -dict {PACKAGE_PIN B17 IOSTANDARD LVCMOS33} [get_ports {exp_io[16]}]
set_property -dict {PACKAGE_PIN A16 IOSTANDARD LVCMOS33} [get_ports {exp_io[17]}]
set_property -dict {PACKAGE_PIN A14 IOSTANDARD LVCMOS33} [get_ports {exp_io[18]}]
set_property -dict {PACKAGE_PIN A18 IOSTANDARD LVCMOS33} [get_ports {exp_io[19]}]
set_property -dict {PACKAGE_PIN F14 IOSTANDARD LVCMOS33} [get_ports {exp_io[20]}]
set_property -dict {PACKAGE_PIN B14 IOSTANDARD LVCMOS33} [get_ports {exp_io[21]}]
set_property -dict {PACKAGE_PIN C14 IOSTANDARD LVCMOS33} [get_ports {exp_io[22]}]
set_property -dict {PACKAGE_PIN A11 IOSTANDARD LVCMOS33} [get_ports {exp_io[23]}]
set_property -dict {PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports {exp_io[24]}]
set_property -dict {PACKAGE_PIN C15 IOSTANDARD LVCMOS33} [get_ports {exp_io[25]}]
set_property -dict {PACKAGE_PIN G16 IOSTANDARD LVCMOS33} [get_ports {exp_io[26]}]
set_property -dict {PACKAGE_PIN F16 IOSTANDARD LVCMOS33} [get_ports {exp_io[27]}]
set_property -dict {PACKAGE_PIN G14 IOSTANDARD LVCMOS33} [get_ports {exp_io[28]}]
set_property -dict {PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports {exp_io[29]}]
set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {exp_io[30]}]
set_property -dict {PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports {exp_io[31]}]
```