2021《FPGA 应用实验》实验报告

实验编号：实验三　　实验时间：2021.3.30

实验名称：LCD 显示字符控制模块设计

班级：F1803603　学号：518021910534　姓名：尹俊同

1、实验平台

采用 Xilinx 公司的 FPGA 集成开发环境 Xilinx ISE Design Suite 10.1 sp3，实验开发板为 Xilinx Spartan-3E FPGA Starter Kit。

2、实验设计要求：

设计 LCD 显示字符控制电路模块，使用在 Spartan-3E FPGA Starter Kit Board 上的 2X16 字符型 LCD 显示指定的字符串。

控制电路的功能和工作状态：

（0）LCD 显示两行字符串，其中：

第 1 行显示：**Spartan-3E□FPGA**

第 2 行显示：**FPAG□Starter**

这里，□表示空格。

（1）使用 BTN_WEST 做为复位键，当按下 BTN_WEST 时，LCD 复位并刷新重新显示两行字符串。

3、模块设计框图

4、实验原理：

Spartan@-3E FPGA 入门套件板突出的特点是 2 行 16 字符液晶显示器(LCD)。EPGA 通过如图 5-1 所示的 4 位数据接口控制 LCD。虽然 LCD 支持 8 位数据接口，但 Starter Kit 板使用 1 位 4 位数据接口来保持与其他 Xilinx 开发板的兼容，并减少总引脚数。

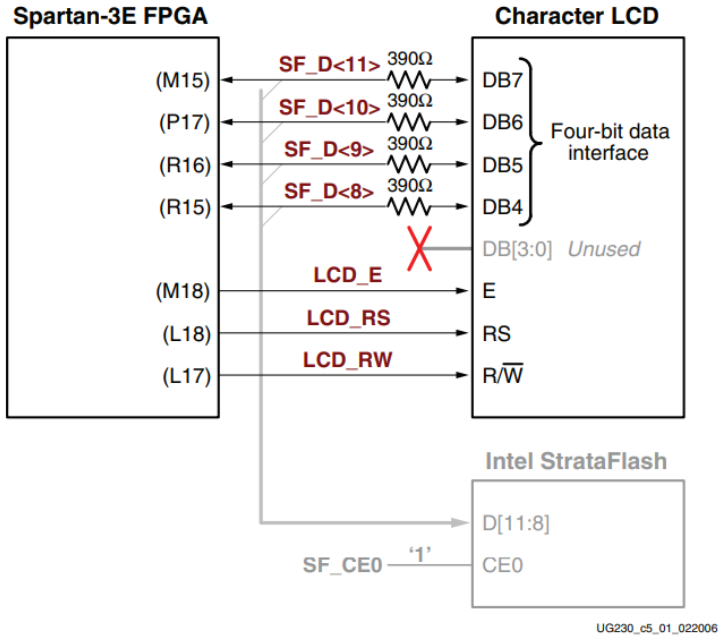


Figure 5-1: **Character LCD Interface**

一旦掌握，LCD 是一个实用的方式显示各种信息使用标准 ASCII 和自定义字符。然而，这些显示并不快。以半秒的间隔滚动显示屏，可以测试清晰度的实际极限。相比主板上有 50mhz 时钟可用，显示很慢。PicoBlaze 处理器有效地控制显示时间和显示的实际内容。

接口字符 LCD 接口信号如表 5-1 所示。

Table 5-1: **Character LCD Interface**

Signal Name	FPGA Pin	Function	
SF_D<11>	M15	Data bit DB7	Shared with StrataFlash pins SF_D<11:8>
SF_D<10>	P17	Data bit DB6	
SF_D<9>	R16	Data bit DB5	
SF_D<8>	R15	Data bit DB4	
LCD_E	M18	Read/Write Enable Pulse 0: Disabled 1: Read/Write operation enabled	
LCD_RS	L18	Register Select 0: Instruction register during write operations. Busy Flash during read operations 1: Data for read or write operations	
LCD_RW	L17	Read/Write Control 0: WRITE, LCD accepts data 1: READ, LCD presents data	

图 5-2 提供了字符液晶的 UCF 约束，包括 1/O 引脚分配和使用的 I/O 标准。

```
NET "LCD_E"   LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RS"  LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW"  LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

# The LCD four-bit data interface is shared with the StrataFlash.
NET "SF_D<8>"  LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<9>"  LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

Figure 5-2: **UCF Location Constraints for the Character LCD**

字符生成 ROM (CG ROM)包含 LCD 屏幕可以显示的每个预定义字符的字体位图，如图 5-4 所示。每个字符的字符代码存储在 RAM DD 位置随后引用 CG ROM 的位置。如图 5-4 所示。字符“S”出现在屏幕上。英文/罗马字符存储在 CG ROM 中，其等效 ASCII 代码地址。

Figure 5-4: **LCD Character Set**

5、Verilog 模块设计

LCD_Display.v 文件：

```
module LCD_Display( output SF_CE0,
    output LCD_RW,
    // 0: 写
    // 1: 读
    output LCD_RS,
    // 0: 指令寄存器
    // 1: 数据寄存器

    output [3:0] SF_D,
    output LCD_E,
    // 0: Disabled, 1: Read/Write operation enabled
```

```
input clock,
input reset
);

parameter    INIT_IDLE       = 4'h1,
          WAITING_READY  = 4'h2,

          WR_ENABLE_1        = 4'h3,
          WAITING_1      = 4'h4,
          WR_ENABLE_2        = 4'h5,
          WAITING_2      = 4'h6,

          WR_ENABLE_3        = 4'h7,
          WAITING_3      = 4'h8,
          WR_ENABLE_4        = 4'h9,
          WAITING_4      = 4'hA,

          INIT_DONE      = 4'hB;


 reg [3:0] init_state;

 reg [19:0] cnt_init;
 reg init_done;


 parameter   DISPLAY_INIT    = 4'h1,

             FUNCTION_SET    = 4'h2,
             ENTRY_MODE_SET  = 4'h3,
             DISPLAY_ON_OFF  = 4'h4,
             DISPLAY_CLEAR   = 4'h5,
             CLEAR_EXECUTION = 4'h6,
             IDLE_2SEC       = 4'h7,

             SET_DD_RAM_ADDR = 4'h8,
             LCD_LINE_1      = 4'h9,
             SET_NEWLINE     = 4'hA,
             LCD_LINE_2      = 4'hB,
             DISPLAY_DONE    = 4'hC;
```

```
reg [3:0] ctrl_state;

reg [16:0] cnt_delay;

reg init_exec;
reg [26:0] cnt_2sec;
reg tx_ctrl;
parameter   TX_IDLE     = 8'H01,
            UPPER_SETUP = 8'H02,
            UPPER_HOLD  = 8'H04,
            ONE_US      = 8'H08,
            LOWER_SETUP = 8'H10,
            LOWER_HOLD  = 8'H20,
            FORTY_US    = 8'H40;
reg [6:0] tx_state;
reg [10:0] cnt_tx;
reg select;
reg [3:0] nibble;
reg [3:0] DB_init;

reg enable;
reg en_init;

reg mux;
reg [7:0] tx_byte;
reg [7:0] tx_Line1;
reg [7:0] tx_Line2;
reg [3:0] cnt_1 = 4'b0; // For Line 1
reg [3:0] cnt_2 = 4'b0; // For Line 2
assign SF_CE0   = 1'b1; // Disable intel strataflash
assign LCD_RW   = 1'b0;
assign LCD_RS   = select;
assign SF_D = ( mux ) ? nibble : DB_init;
assign LCD_E    = ( mux ) ? enable : en_init;
always @(*)
begin
    case ( ctrl_state )
        DISPLAY_INIT:       mux = 1'b0;
```

```
        FUNCTION_SET,
        ENTRY_MODE_SET,
        DISPLAY_ON_OFF,
        DISPLAY_CLEAR,
        IDLE_2SEC,
        CLEAR_EXECUTION,
        SET_DD_RAM_ADDR,
        LCD_LINE_1,
        SET_NEWLINE,
        LCD_LINE_2:         mux = 1'b1;
        default:            mux = 1'b0;
    endcase
end
always @( * ) begin
    case ( ctrl_state )
        FUNCTION_SET:       begin
                                tx_byte = 8'b0010_1000;
                                select = 1'b0;
                            end
        ENTRY_MODE_SET:     begin
                                tx_byte = 8'b0000_0110;
                                select = 1'b0;
                            end
        DISPLAY_ON_OFF:     begin
                                tx_byte = 8'b0000_1100;
                                select = 1'b0;
                            end
        DISPLAY_CLEAR:      begin
                                tx_byte = 8'b0000_0001;
                                select = 1'b0;
                            end
        SET_DD_RAM_ADDR:    begin
                                tx_byte = 8'b1000_0000;
                                select = 1'b0;
                            end
        LCD_LINE_1:         begin
                                tx_byte = tx_Line1;
                                select = 1'b1;
                            end
        SET_NEWLINE:        begin
```

```
                            tx_byte = 8'b1100_0000;
                            select = 1'b0;
                        end
        LCD_LINE_2:     begin
                            tx_byte = tx_Line2;
                            select = 1'b1;
                        end

        default:        begin
                            tx_byte = 8'b0;
                            select = 1'b0;
                        end
    endcase
end

always @(*)
begin
    case ( cnt_1 )
        0:      tx_Line1 = 8'b0101_0011;        // CHAR_S
        1:      tx_Line1    = 8'b0111_0000;     // CHAR_p
        2:      tx_Line1 = 8'b0110_0001;        // CHAR_a
        3:      tx_Line1    = 8'b0111_0010;     // CHAR_r
        4:      tx_Line1    = 8'b0111_0100;     // CHAR_t
        5:      tx_Line1    = 8'b0110_0001;     // CHAR_a
        6:      tx_Line1    = 8'b0110_1110;     // CHAR_n
        7:      tx_Line1    = 8'b0010_1101;     // CHAR_-
        8:      tx_Line1    = 8'b0011_0011;     // CHAR_3
        9:      tx_Line1    = 8'b0100_0101;     // CHAR_E
        10:   tx_Line1   = 8'b0010_0000;     // CHAR_space
        11: tx_Line1    = 8'b0100_0110;     // CHAR_F
        12:   tx_Line1 = 8'b0101_0000;       // CHAR_P
        13:   tx_Line1 = 8'b0100_0111;       // CHAR_G
        14: tx_Line1 = 8'b0100_0001;         // CHAR_A
        default:tx_Line1    = 8'b0;              // NONE

    endcase
end

always @(*)
begin
```

```
    case ( cnt_2 )
        0:      tx_Line2 = 8'b0100_0110;        // CHAR_F
        1:      tx_Line2 = 8'b0101_0000;        // CHAR_P
        2:      tx_Line2 = 8'b0100_0111;        // CHAR_G
        3:      tx_Line2 = 8'b0100_0001;        // CHAR_A
        4:      tx_Line2 = 8'b0010_0000;        // CHAR_space
        5:      tx_Line2 = 8'b0101_0011;        // CHAR_S
        6:      tx_Line2 = 8'b0111_0100;        // CHAR_t
        7:      tx_Line2 = 8'b0110_0001;        // CHAR_a
        8:      tx_Line2 = 8'b0111_0010;        // CHAR_r
        9:      tx_Line2 = 8'b0111_0100;        // CHAR_t
        10: tx_Line2 = 8'b0110_0101;        // CHAR_e
        11: tx_Line2 = 8'b0111_0010;        // CHAR_r
        default:tx_Line2    = 8'b0;             // NONE
    endcase
end

always @( posedge clock )
begin
    if( reset ) begin
        init_state <= INIT_IDLE;

        DB_init <= 4'b0;
        en_init <= 0;

        cnt_init <= 0;

        init_done <= 0;
    end

    else begin
        case ( init_state )
            INIT_IDLE:          begin
                                    en_init <= 0;

                                    if ( init_exec  )
                                        init_state <= WAITING_READY;
                                    else
                                        init_state <= INIT_IDLE;
                                end
```

```
                WAITING_READY:      begin
                                        en_init <= 0;

                                        if ( cnt_init <= 750000 ) begin
                                            DB_init <= 4'h0;

                                            cnt_init <= cnt_init + 1;

                                            init_state <= WAITING_READY;
                                        end
                                        else begin
                                            cnt_init <= 0;

                                            init_state <= WR_ENABLE_1;
                                        end
                                    end

                WR_ENABLE_1:        begin
                                        DB_init <= 4'h3;
// Write SF_D<11:8> = 0x3

                                        en_init <= 1'b1;
// Pulse LCD_E High for 12 clock cycles.

                                        if ( cnt_init < 12 ) begin
                                            cnt_init <= cnt_init + 1;

                                            init_state <= WR_ENABLE_1;
                                        end
                                        else begin
                                            cnt_init <= 0;

                                            init_state <= WAITING_1;
                                        end
                                    end

                WAITING_1:          begin
// Wait 4.1 ms or longer, which is 205,000 clock cycles at 50 MHz.
                                        en_init <= 1'b0;
```

```
                                    if ( cnt_init <= 205000 ) begin

                                        cnt_init <= cnt_init + 1;

                                        init_state <= WAITING_1;
                                end
                                else begin
                                        cnt_init <= 0;

                                        init_state <= WR_ENABLE_2;
                                    end
                                end
                WR_ENABLE_2:        begin
                                    DB_init <= 4'h3;
// Write SF_D<11:8> = 0x3
                                    en_init <= 1'b1;
// Pulse LCD_E High for 12 clock cycles.

                                    if ( cnt_init < 12 ) begin

                                        cnt_init <= cnt_init + 1;

                                        init_state <= WR_ENABLE_2;
                                end
                                else begin
                                        cnt_init <= 0;

                                        init_state <= WAITING_2;
                                    end
                                end

// Wait 100 µs or longer, which is 5,000 clock cycles at 50 MHz.
                WAITING_2:          begin
                                    en_init <= 1'b0;

                                    if ( cnt_init <= 5000 ) begin

                                        cnt_init <= cnt_init + 1;
```

```
                        init_state <= WAITING_2;
                    end
                    else begin
                        cnt_init <= 0;

                        init_state <= WR_ENABLE_3;
                    end
                end

WR_ENABLE_3:        begin
//  Write SF_D<11:8> = 0x3, pulse LCD_E High for 12 clock cycles.
                    DB_init <= 4'h3;
                    // Write SF_D<11:8> = 0x3
                    en_init <= 1'b1;
                    // Pulse LCD_E High for 12 clock cycles.

                    if ( cnt_init < 12 ) begin

                        cnt_init <= cnt_init + 1;

                        init_state <= WR_ENABLE_3;
                end
                else begin
                        cnt_init <= 0;

                        init_state <= WAITING_3;
                    end
                end

WAITING_3:          begin
//  Wait 40 us or longer, which is 2,000 clock cycles at 50 MHz.
                    en_init <= 1'b0;

                    if ( cnt_init <= 2000 ) begin

                        cnt_init <= cnt_init + 1;

                        init_state <= WAITING_3;
                end
                else begin
```

```
                                    cnt_init <= 0;

                                    init_state <= WR_ENABLE_4;
                                end
                            end

                WR_ENABLE_4:        begin
                //  Write SF_D<11:8> = 0x2, pulse LCD_E High for 12 clock cycles.
                                    DB_init <= 4'h2;
                                    // Write SF_D<11:8> = 0x3
                                    en_init <= 1'b1;
                                    // Pulse LCD_E High for 12 clock cycles.

                                    if ( cnt_init < 12 ) begin

                                        cnt_init <= cnt_init + 1;

                                        init_state <= WR_ENABLE_4;
                                    end
                                    else begin
                                        cnt_init <= 0;

                                        init_state <= WAITING_4;
                                    end
                                end

                WAITING_4:          begin
                //  Wait 40 us or longer, which is 2,000 clock cycles at 50 MHz.
                                    en_init <= 1'b0;


                                    if ( cnt_init <= 2000 ) begin
                                        cnt_init <= cnt_init + 1;

                                        init_state <= WAITING_4;
                                    end
                                    else begin
                                        DB_init <= 4'h0;
// Write SF_D<11:8> = 0x0
                                        cnt_init <= 0;
```

```
                                    cnt_init <= 0;

                                    init_done <= 1'b1;
                                    init_state <= INIT_DONE;
                                end
                            end

            INIT_DONE:      begin
                                init_state <= INIT_DONE;

                                DB_init <= 4'h0;
                                en_init <= 1'b0;

                                cnt_init <= 0;

                                init_done <= 1'b1;
                            end
            default:        begin
                                init_state <= INIT_IDLE;

                                DB_init <= 4'b0;
                                en_init <= 0;

                                cnt_init <= 0;

                                init_done <= 0;
                            end
        endcase
    end
end


always @( * )
begin
    case ( ctrl_state )
        DISPLAY_INIT:       tx_ctrl = 1'b0;
        FUNCTION_SET,
        ENTRY_MODE_SET,
        DISPLAY_ON_OFF,
```

```
        DISPLAY_CLEAR:      tx_ctrl = 1'b1;
        CLEAR_EXECUTION:    tx_ctrl = 1'b0;
        SET_DD_RAM_ADDR,
        LCD_LINE_1,
        SET_NEWLINE,
        LCD_LINE_2:         tx_ctrl = 1'b1;
        DISPLAY_DONE:       tx_ctrl = 1'b0;
        default:            tx_ctrl = 1'b0;
    endcase
end

always @( posedge clock )
begin
    if( reset ) begin
        ctrl_state <= DISPLAY_INIT;

        cnt_delay <= 0;
        cnt_1 <= 0;
        cnt_2 <= 0;

        cnt_2sec <= 0;
    end

    else begin
        case ( ctrl_state )
            // power on initialization sequence
            DISPLAY_INIT:       begin
                                    init_exec <= 1;

                                    if ( init_done ) begin
                                        ctrl_state <= FUNCTION_SET;
                                        cnt_1 <= 0;
                                        cnt_2 <= 0;
                                    end
                                    else begin
                                        ctrl_state <= DISPLAY_INIT;
                                    end
                                end

            FUNCTION_SET:       begin
```

```
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= FUNCTION_SET;
                        end
                        else begin
                            ctrl_state <= ENTRY_MODE_SET;
                        end
                    end

ENTRY_MODE_SET:     begin
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= ENTRY_MODE_SET;
                        end
                        else begin
                            ctrl_state <= DISPLAY_ON_OFF;
                        end
                    end

DISPLAY_ON_OFF:     begin
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= DISPLAY_ON_OFF;
                        end
                        else begin
                            ctrl_state <= DISPLAY_CLEAR;
                        end
                    end

DISPLAY_CLEAR:      begin
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= DISPLAY_CLEAR;
                        end
                        else begin
                            ctrl_state <= CLEAR_EXECUTION;

                            cnt_delay <= 0;
                        end
                    end
```

```
CLEAR_EXECUTION:    begin
                    // The delay after a Clear Display command is 1.64ms,
                    // which corresponds to 82000 clock cycles.
                        if ( cnt_delay <= 82000 ) begin
                            ctrl_state <= CLEAR_EXECUTION;

                            cnt_delay <= cnt_delay + 1;
                        end
                        else begin
                            ctrl_state <= IDLE_2SEC;
                            cnt_delay <= 0;

                            cnt_2sec <= 0;
                        end
                    end

IDLE_2SEC:          begin
                        if ( cnt_2sec < 27'd100000000 ) begin
                            ctrl_state <= IDLE_2SEC;
                            cnt_2sec <= cnt_2sec + 1;
                        end
                        else begin
                            ctrl_state <= SET_DD_RAM_ADDR;

                            cnt_delay <= 0;
                        end
                    end

SET_DD_RAM_ADDR:    begin
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= SET_DD_RAM_ADDR;
                        end
                        else begin
                            ctrl_state <= LCD_LINE_1;
                            cnt_1 <= 0;
                        end
                    end
```

```
LCD_LINE_1:         begin
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= LCD_LINE_1;
                        end
                        else if ( cnt_1 < 14 ) begin
                                ctrl_state <= LCD_LINE_1;

                                cnt_1 <= cnt_1 + 1;
                            end
                            else begin
                                ctrl_state <= SET_NEWLINE;

                                cnt_1 <= 0;
                            end
                    end

SET_NEWLINE:        begin
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= SET_NEWLINE;
                        end
                        else begin
                            ctrl_state <= LCD_LINE_2;

                            cnt_2 <= 0;
                        end
                    end

LCD_LINE_2:         begin
                        // Wait 40 us or longer
                        if ( cnt_tx <= 2000 ) begin
                            ctrl_state <= LCD_LINE_2;
                        end
                        else if ( cnt_2 < 11 ) begin
                                ctrl_state <= LCD_LINE_2;

                                cnt_2 <= cnt_2 + 1;
                            end
                            else begin
```

```
                                    ctrl_state <= DISPLAY_DONE;

                                    cnt_2 <= 0;
                                end
                            end

            DISPLAY_DONE:       begin
                                    ctrl_state <= DISPLAY_DONE;
                                end
            default:            begin
                                    ctrl_state <= DISPLAY_INIT;

                                    cnt_delay <= 0;
                                    cnt_1 <= 0;
                                    cnt_2 <= 0;

                                    cnt_2sec <= 0;
                                end
        endcase
    end
end


always @( posedge clock )
begin
    if ( reset ) begin
        enable <= 1'b0;
        nibble <= 4'b0;

        tx_state <= TX_IDLE;
        cnt_tx <= 0;
    end
    else  begin
        case ( tx_state )
            TX_IDLE:            begin
                                    enable <= 1'b0;
                                    nibble <= 4'b0;
                                    cnt_tx <= 0;

                                    if ( tx_ctrl ) begin
```

```
                                    tx_state <= UPPER_SETUP;
                                end
                                else begin
                                    tx_state <= TX_IDLE;
                                end
                            end
            // Setup time ( time for the outputs to stabilize ) is 40ns, which is 2 clock
cycles
            UPPER_SETUP:        begin
                                    nibble <= tx_byte[7:4];

                                    if ( cnt_tx < 2 ) begin
                                        enable <= 1'b0;

                                        tx_state <= UPPER_SETUP;

                                        cnt_tx <= cnt_tx + 1;
                                    end
                                    else begin
                                        enable <= 1'b1;

                                        tx_state <= UPPER_HOLD;
                                        cnt_tx <= 0;
                                    end
                                end
            // Hold time ( time to assert the LCD_E pin ) is 230ns, which translates to
roughly 12 clock cycles
            UPPER_HOLD:         begin
                                    nibble <= tx_byte[7:4];

                                    if ( cnt_tx < 12 ) begin
                                        enable <= 1'b1;
                                        tx_state <= UPPER_HOLD;
                                        cnt_tx <= cnt_tx + 1;
                                    end
                                    else begin
                                        enable <= 1'b0;
                                        tx_state <= ONE_US;
                                        cnt_tx <= 0;
                                    end
```

```
                    end

ONE_US:             begin
                        enable <= 1'b0;

                        if ( cnt_tx <= 50 ) begin
                            tx_state <= ONE_US;
                            cnt_tx <= cnt_tx + 1;
                        end
                        else begin
                            tx_state <= LOWER_SETUP;
                            cnt_tx <= 0;
                        end
                    end

LOWER_SETUP:        begin
                        nibble <= tx_byte[3:0];

                        if ( cnt_tx < 2 ) begin
                            enable <= 1'b0;

                            tx_state <= LOWER_SETUP;
                            cnt_tx <= cnt_tx + 1;
                        end
                        else begin
                            enable <= 1'b1;

                            tx_state <= LOWER_HOLD;
                            cnt_tx <= 0;
                        end
                    end


LOWER_HOLD:         begin
                        nibble <= tx_byte[3:0];

                        if ( cnt_tx < 12 ) begin
                            enable <= 1'b1;
                            tx_state <= LOWER_HOLD;
                            cnt_tx <= cnt_tx + 1;
```

```
                                    end
                                    else begin
                                        enable <= 1'b0;
                                        tx_state <= FORTY_US;
                                        cnt_tx <= 0;
                                    end
                                end


                FORTY_US:           begin
                                    enable <= 1'b0;

                                    if ( cnt_tx <= 2000 ) begin
                                        tx_state <= FORTY_US;
                                        cnt_tx <= cnt_tx + 1;
                                    end
                                    else begin
                                        tx_state <= TX_IDLE;
                                        cnt_tx <= 0;
                                    end
                                end
                default:            begin
                                        enable <= 1'b0;
                                        nibble <= 4'b0;

                                        tx_state <= TX_IDLE;
                                        cnt_tx <= 0;
                                end
            endcase
        end
    end
endmodule
```

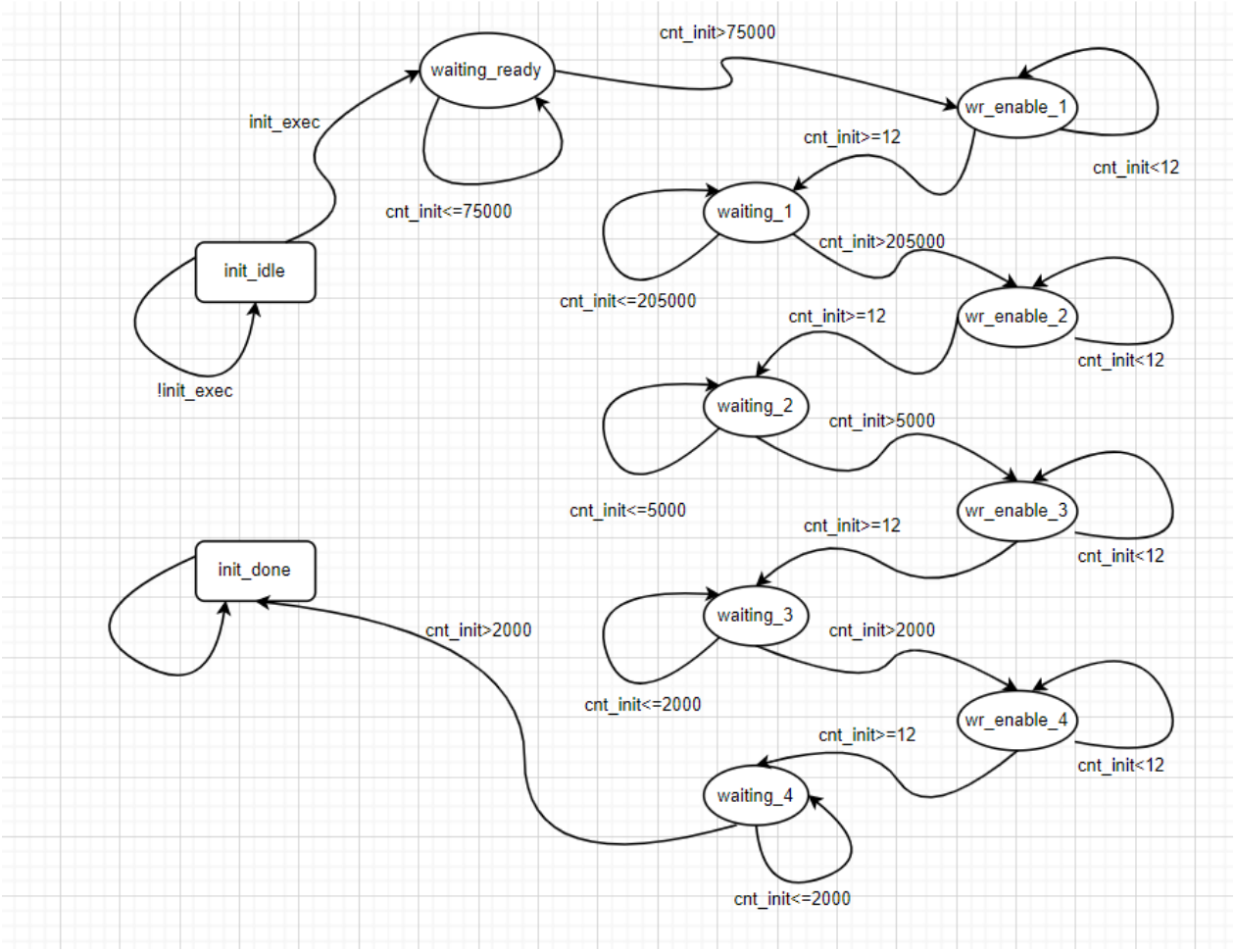
UCF 文件：

```
NET "SF_CE0"  LOC = "D16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW"  LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RS"  LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_E"   LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<0>" LOC  = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<1>" LOC  = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

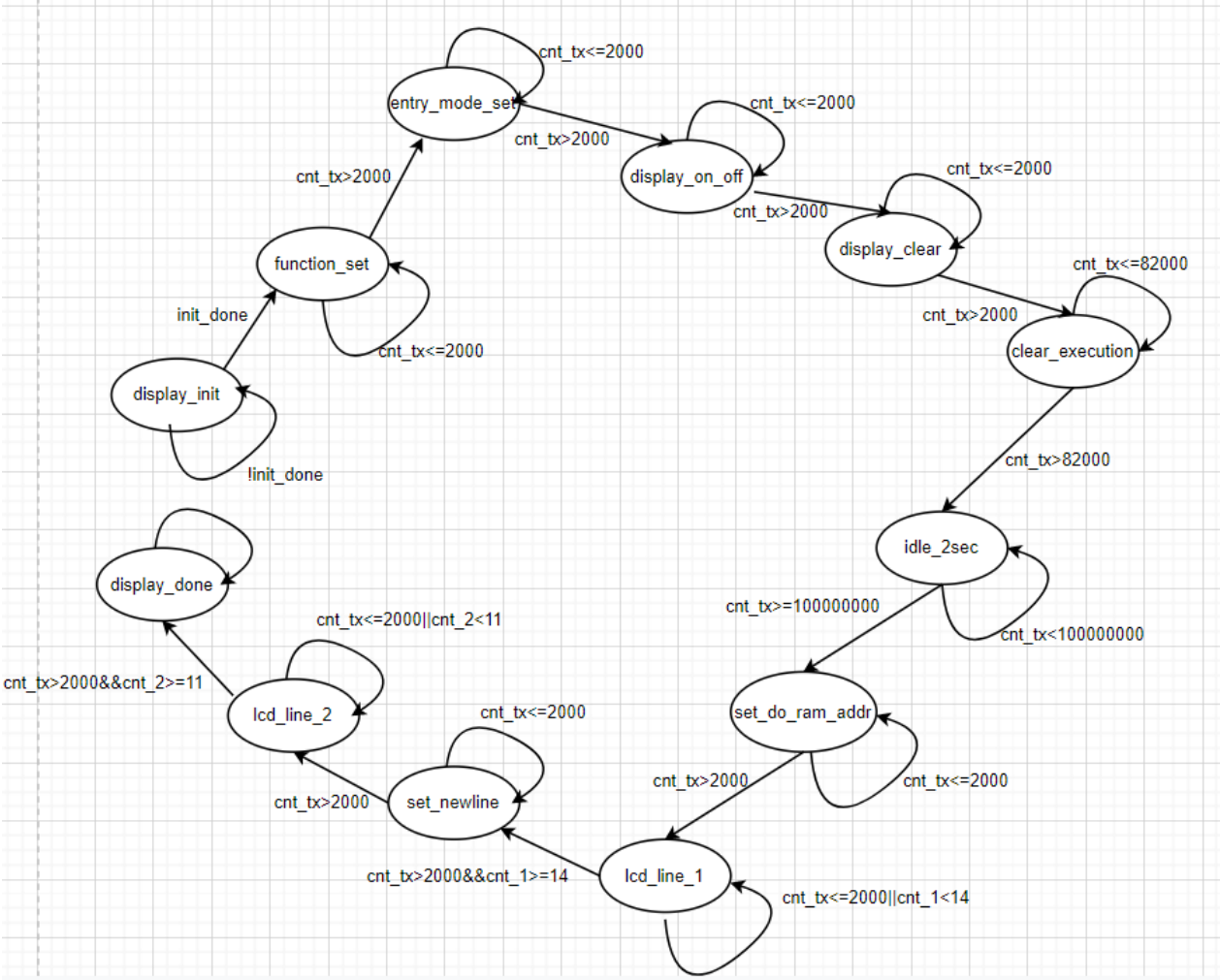
```
NET "SF_D<2>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<3>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "clock"  LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "reset"  LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
```

5.2、状态机的状态转移图

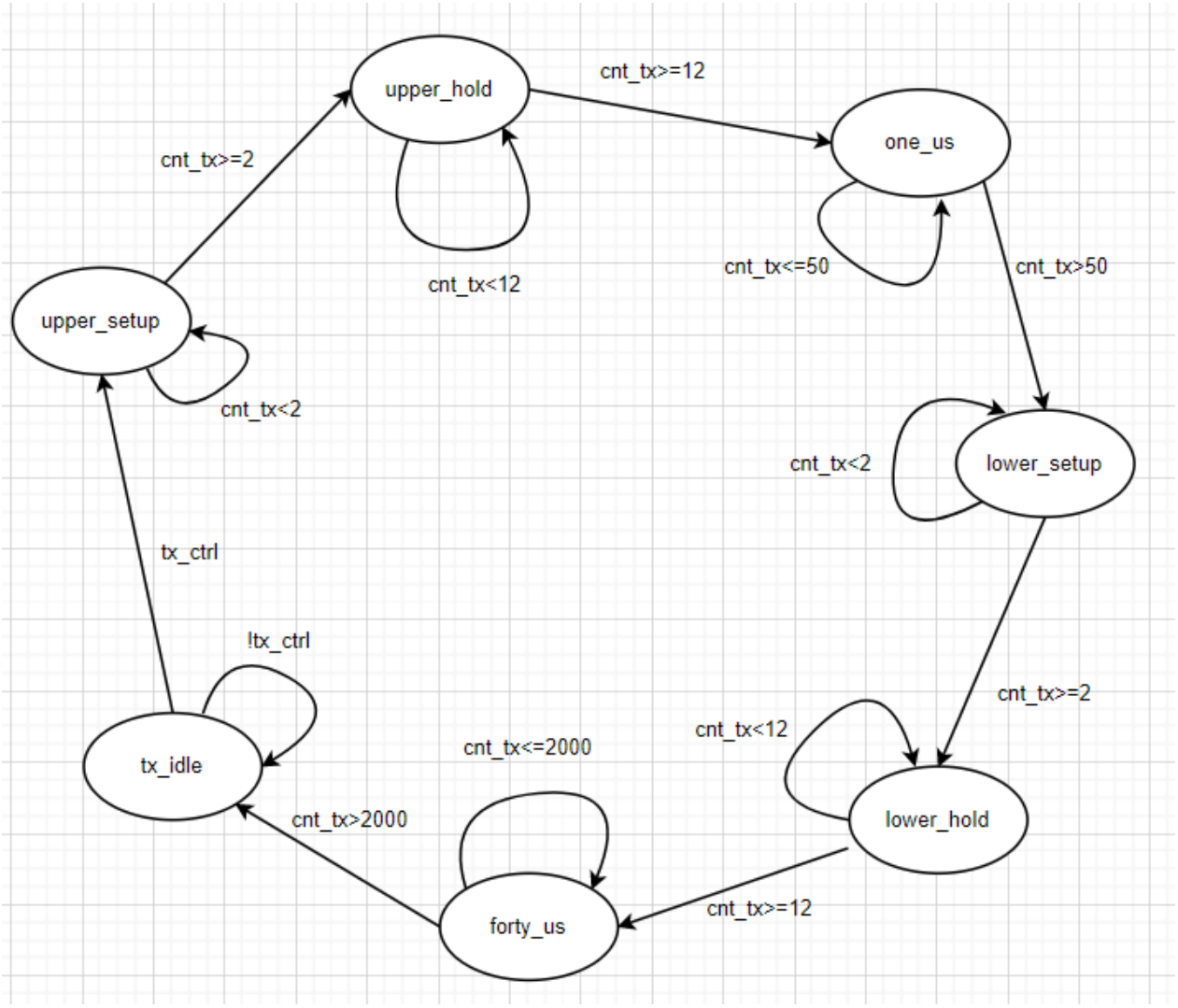
(1)上电初始化状态转移图



(2)屏幕显示状态转移图


cnt_tx<=2000
entry_mode_set
cnt_tx>2000
cnt_tx<=2000
display_on_off
cnt_tx>2000
cnt_tx<=2000
display_clear
cnt_tx>2000
cnt_tx<=82000
clear_execution
cnt_tx>82000
idle_2sec
cnt_tx>=100000000
cnt_tx<100000000
set_do_ram_addr
cnt_tx>2000
cnt_tx<=2000
lcd_line_1
cnt_tx>2000&&cnt_1>=14
cnt_tx<=2000||cnt_1<14
set_newline
cnt_tx>2000
cnt_tx<=2000
lcd_line_2
cnt_tx>2000&&cnt_2>=11
cnt_tx<=2000||cnt_2<11
display_done
display_init
!init_done
init_done
function_set
cnt_tx<=2000
cnt_tx>2000

(3)字符传输状态转移图


upper_hold
cnt_tx>=12
one_us
cnt_tx>=2
cnt_tx<12
cnt_tx<=50
cnt_tx>50
upper_setup
cnt_tx<2
cnt_tx<2
lower_setup
tx_ctrl
!tx_ctrl
cnt_tx>=2
cnt_tx<12
cnt_tx<=2000
tx_idle
lower_hold
cnt_tx>2000
cnt_tx>=12
forty_us