2017《FPGA 应用实验》实验报告

实验编号：<u>Lab03</u>　　　　　实验时间：<u>2018. 4. 3</u>

实验名称：<u>**LCD 显示字符控制模块设计**</u>

班级：<u>F1503006</u>　学号：<u>515030910141</u>　姓名：<u>翟拙存</u>

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：

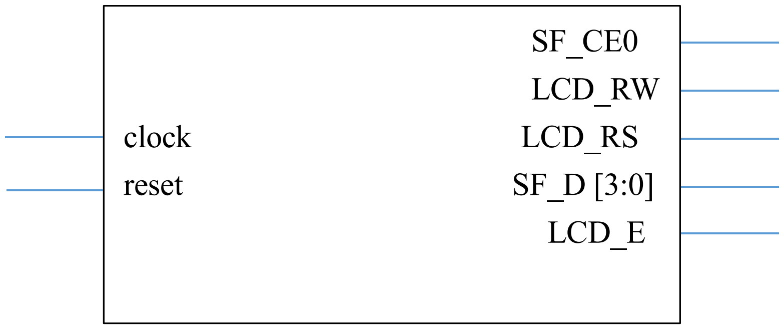
第 1 行显示：**Spartan-3E□FPGA**

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。

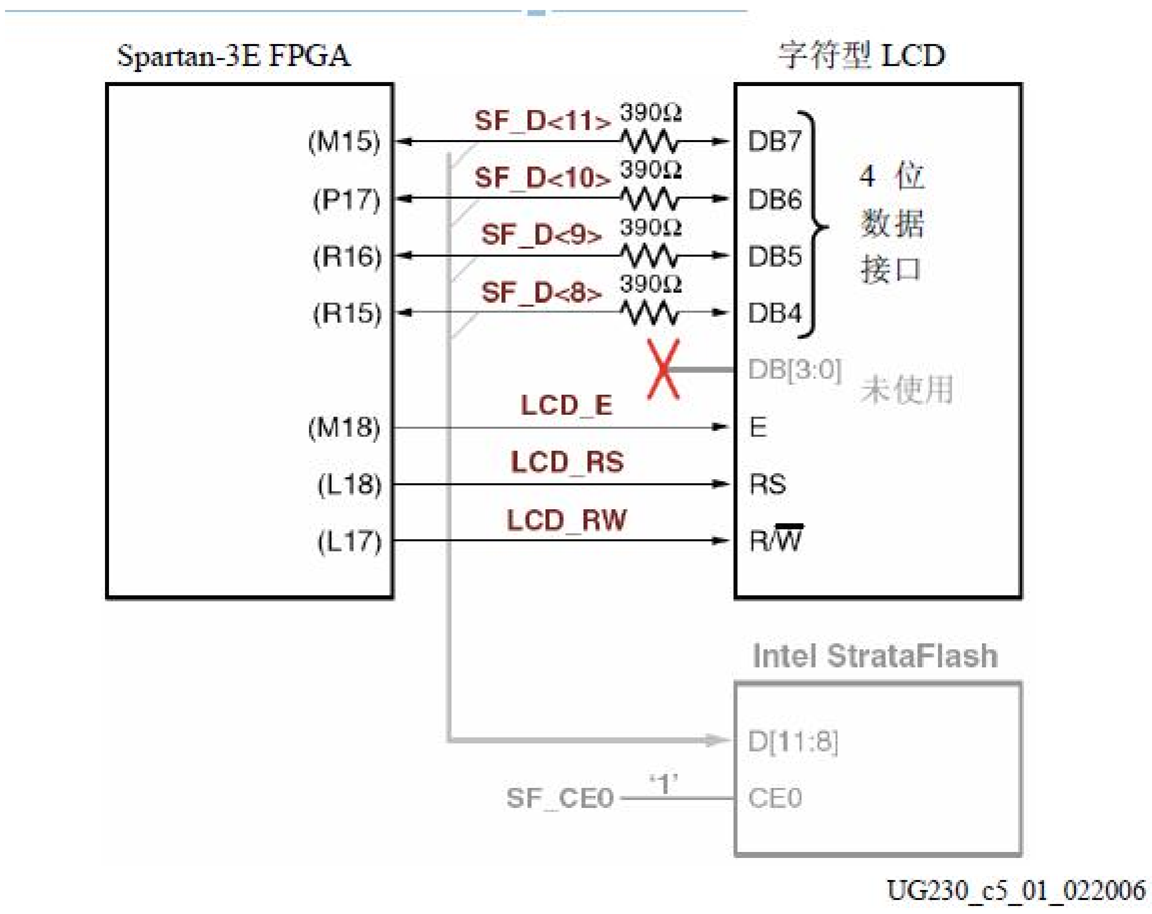
3、模块设计框图



4、实验原理：

1. LCD 控制器:

FPGA 通过下图 所示的4 位数据接口控制LCD。尽管此LCD 支持8 位数据接口，为了保持与Xilinx 的其他开发板兼容并减少使用管脚数，此开发板使用了4 位数据接口。



2. 字符型LCD接口信号:

信号名称	FPFA 管脚	功能	
SF_D<11>	M15	数据位 DB7	与 StrataFlash 管脚 SF_D<11:8>共用
SF_D<10>	P17	数据位 DB6	
SF_D<9>	R16	数据位 DB5	
SF_D<8>	R15	数据位 DB4	
LCD_E	M18	读/写使能脉冲 0：禁用 1：允许读/写	
LCD_RS	L18	寄存器选择 0：写指令寄存器，读忙标志（译者：原文有误） 1：读写数据寄存器	
LCD_RW	L17	读/写控制 0：写，LCD 接收数据 1：读，LCD 送出数据	

字符生成器 ROM（CG ROM）存储了LCD 显示屏所能显示的所有预定义好的字符的点阵图，如下图所示。存储在DD RAM 中的字符代码指向了CG ROM 中的一个位置。例如，存储在DD RAM 中的十六进制字符代码0x53 应显示为字符“S”。0x53的前半个字节DB[7:4]=“0101”而后半个字节DB[3:0]=“0101”。如图5—4，对应字符“S”显示在屏幕上。

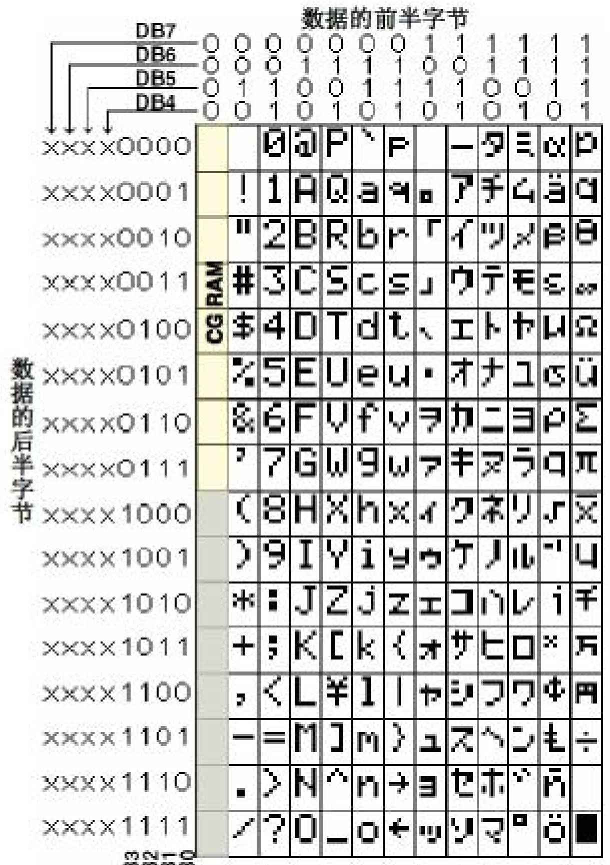


图 5—4 LCD 字符集 UG230_c5_02_030306

由于 CGRAM 在本次实验中并不需要重点涉及,因此相关信息不给出。

LCD显示屏还是一系列的命令集，显示屏被设置成4 位模式，所以每个8 位命令分为两个4 位的半字节发送。先发送前半字节，然后是后半字节。例如：

表 5—3 LCD 字符型显示屏的命令集

功能	LCD_RS	LCD_RW	前半字节				后半字节			
			DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear Display	0	0	0	0	0	0	0	0	0	1
Return Cursor Home	0	0	0	0	0	0	0	0	1	—
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S
Display On/Off	0	0	0	0	0	0	1	D	C	B
Cursor and Display Shift	0	0	0	0	0	1	S/C	R/L	—	—

3. LCD 显示屏的操作：

通过 4 位借口传送 8 位数据：

显示屏被初始化并建立通讯后，所有命令和数据都是通过连续两个4 位数据传送操作所组成的8 位数据传送。每个8 位数据的传送必须分解成两次4 位传送，之间至少间隔1us，如图5—6 所示。前半字节先发送，然后是后半个字节。8 字节写操作与下一次通讯间至少间隔40us。如果上一个命令是Clear Display（清屏），则间隔时应增加到1.64ms。

4. 初始化显示屏：

上电以后，显示屏必须被初始化以建立所需的通讯协议。初始化过程很简单，非常适用于高效8 位PicoBlaze 嵌入控制器。初始化以后，除了简单的驱动此显示屏，PicoBlaze控制器还可以进行更复杂的控制和计算工作

5. 上电初始化：

初始化过程首先按 FPGA 应用程序希望的4 位数据接口方式建立所需通讯：

等待 15ms，尽管当FPGA 完成配置时显示屏通常已经准备好。15ms 相当于50MHz时钟的750，000 个周期。

写 SF_D<11:8> = 0x3，LCD_E 发12 个时钟周期高脉冲信号

等待4.1ms 或更长，相当于50MHz 时钟的205，000 个周期

写SF_D<11:8> = 0x3，LCD_E 发12 个时钟周期高脉冲信号

等待100us 或更长，相当于50MHz 时钟的5，000 个周期

写SF_D<11:8> = 0x3，LCD_E 发12 个时钟周期高脉冲信号

等待40us 或更长，相当于50MHz 时钟的2，000 个周期

写SF_D<11:8> = 0x2，LCD_E 发12 个时钟周期高脉冲信号

等待40us 或更长，相当于50MHz 时钟的2，000 个周期

6. 显示屏配置：

上电初始化完成，现在可以使用 4 位数据接口了。下面的过程配置显示屏：
发 Function Set 命令，0x28，以配置显示屏在Spartan－3E 开发板上工作。
发 Entry Mode Set 命令，0x06，设置显示屏地址指针自动增加
发 Display On/Off 命令，0x0C，打开显示屏，关闭光标和闪烁
最后，发出Clear Display 命令。此命令后等待至少1.64ms（82，000 时钟周期）。

7. 向显示屏写数据：

要向显示屏写数据，先指定开始地址，然后是一个或更多的数据值。
在写任何数据前，先发出Set DD RAM Address 命令以指定DD RAM 的7 位起始地址。

使用 Write Data to CG RAM or DD RAM 命令向显示屏写数据。8 位的数据值代表了指向CG ROM 或CG RAM 的查找表地址，如图5－4 所示。CG ROM 或CG RAM 中存储的点阵图驱动了5x8 点的点阵，来显示相应的字符。

如果地址计数器被配置为自动增加，如前文所述，可以顺序的向显示屏写入多个字符代码，每个字符代码将自动的存储和显示在下一个可用位置。

但是，连续写入字符，最终会超出显示区域的第一行，额外的字符不会自动出现在第二行，因为DD RAM 的地址不是从第一行连续映射到第二行。

5、Verilog 模块设计

1. LCD_Driver.v:

```
module LCD_Driver( output SF_CE0,            // 4 位 LCD 数据信号与 StrataFlash 存储器共享数据线
SF_D<11:8>.
                                              // 当 SF_CE0 = High 时, 禁用 StrataFlash 存储器,
                                              // 此时 FPGA 完全 read/write 访问 LCD.
                   output LCD_RW,             // Read/Write Control
                                              // 0: WRITE, LCD accepts SF_D
                                              // 1: READ, LCD presents SF_D

                   output LCD_RS,             // Register Select
                                              //  0:  Instruction  register  during  write
operations.
                                              //    Busy Flash during read operations
                                              // 1: Data for read or write operations

                   output [3:0] SF_D,         // Four-bit SF_D interface, Data bit DB7 ~ DB4,
                                              // Shared with StrataFlash pins SF_D<11:8>

                   output LCD_E,          // Read/Write Enable Pulse,
                                              // 0: Disabled, 1: Read/Write operation enabled
```

```
              input clock,              // 连接 On-Board 50 MHz Oscillator CLK_50MHz (C9)
              input reset               // 使用按键 BTN EAST(H13)做为复位键

             );

//////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////
// 定义 LCD 初始化和显示配置状态变量

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;

// 保存 LCD 初始化状态变量: 位宽 3 bits
reg [3:0] init_state;

// 时序控制计数器
// The 15 ms interval is 750,000 clock cycles at 50 MHz.
// 750,000 (dec) = 1011_0111_0001_1011_0000(bin) 需要 20 bits
reg [19:0] cnt_init;

// 初始化状态标志
// 0: 初始化未完成
// 1: 初始化已完成
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;



// 保存 LCD 显示配置状态变量: 位宽 3 bits
reg [3:0] ctrl_state;

// 时序控制计数器
// Clear the display and return the cursor to the home position, the top-left corner.
// Execution Time at least 1.64 ms (82,000 clock cycles)
// 82,000 (dec) = 1_0100_0000_0101_0000 (bin) 需要 17 bits
reg [16:0] cnt_delay;

// 控制初始化标志
// 1: 启动传输过程
// 0: 停止传输过程
reg init_exec;

// 复位后, 等待 2 sec, 运行在 50 MHz 时钟频率
// 等待 100,000,000(dec) = 101_1111_0101_1110_0001_0000_0000 (bin) (27 bits) 时钟周期
reg [26:0] cnt_2sec;

// 控制传输标志
// 1: 启动  涔  ? // 0: 停止传输过程
reg tx_ctrl;
```

```
// 传输序列状态
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;



// 保存传输序列状态：位宽 7 bits
reg [6:0] tx_state;

// 传输控制时序计数器
// The time between successive commands is 40us, which corresponds to 2000 clock cycles
// 2000 (dec ) = 111_1101_0000 (bin) 需要 11 bits
reg [10:0] cnt_tx;

// Register Select
// 0: Instruction register during write operations. Busy Flash during read operations
// 1: Data for read or write operations
reg select;

// The upper nibble is transferred first, followed by the lower nibble.
reg [3:0] nibble;
reg [3:0] DB_init;  // 用于初始化

// Read/Write Enable Pulse, 0: Disabled, 1: Read/Write operation enabled
reg enable;
reg en_init;      // 用于初始化

reg mux;            // 标志初始化过程，传输命令/数据
                    // 0: 初始化
                    // 1: 传输命令/数据

// 向 LCD 传输的数据字节：位宽 8 bits
reg [7:0] tx_byte;

// 保存第 1 行显示输出的字符数据
reg [7:0] tx_Line1;
```

```
    // 保存第 2 行显示输出的字符数据
    reg [7:0] tx_Line2;

    // 显示字符计数器
    reg [3:0] cnt_1 = 4'b0; // For Line 1
    reg [3:0] cnt_2 = 4'b0; // For Line 2

    /////////////////////////////////////////////////////////////////////////////////////
//////
    // 禁用 Intel strataflash 存储器，将 Read/Write 控制设置为 Write，即：LCD 接收数据
    assign SF_CE0   = 1'b1; // Disable intel strataflash

    assign LCD_RW   = 1'b0;     // Write only

    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; // power on initialization sequence
            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
    /////////////////////////////////////////////////////////////////////////////////////
//////
```

```
// The following "always" statements simplify the process of adding and removing states.

//
// refer to datasheet for an explanation of these values

// 向 LCD 传输的命令字节: 位宽 8 bits
// In Verilog-2001, you can initialize registers when you declare them.
// Now Xilinx XST has supported to initialize registers
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

/*      上电后 LCD 初始化过程
        Power-On Initialization

        The initialization sequence first establishes that the FPGA application
        wishes to use the four-bit SF_D interface to the LCD as follows:

        (0) Wait 15 ms or longer, although the display is generally ready when the FPGA finishes
configuration.
            The 15 ms interval is 750,000 clock cycles at 50 MHz.

        (1) Write SF_D<11:8> = 0x3, pulse LCD_E High for 12 clock cycles.

        (2) Wait 4.1 ms or longer, which is 205,000 clock cycles at 50 MHz.

        (3) Write SF_D<11:8> = 0x3, pulse LCD_E High for 12 clock cycles.

        (4) Wait 100 μs or longer, which is 5,000 clock cycles at 50 MHz.

        (5) Write SF_D<11:8> = 0x3, pulse LCD_E High for 12 clock cycles.

        (6) Wait 40 μs or longer, which is 2,000 clock cycles at 50 MHz.

        (7) Write SF_D<11:8> = 0x2, pulse LCD_E High for 12 clock cycles.

        (8) Wait 40 μs or longer, which is 2,000 clock cycles at 50 MHz.
```

```
*/

    // Initializing the Display
    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 )
                // power on initialization sequence
                INIT_IDLE:          begin
                                        en_init <= 0;

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

                WAITING_READY:      begin   // (0 )等待 15 ms 或更长, LCD 准备显示
                                        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

    // Main state machine
    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   // (0 )等待 15 ms 或更长, LCD 准备显示
                                    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;
```

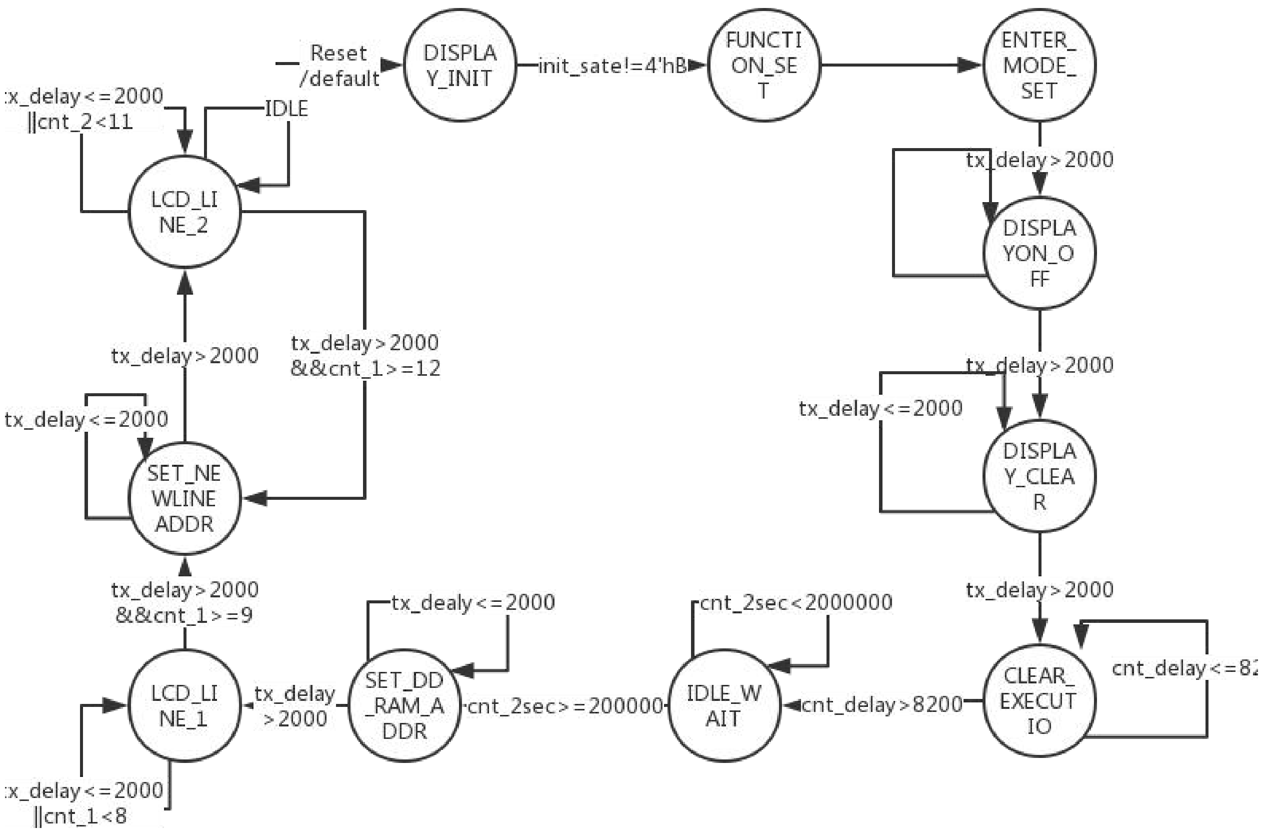
```
                            cnt_2sec <= 0;
                        end
                    end

IDLE_2SEC:          begin // 清屏后, 等待 2 sec, 观察复位
                        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
/*
    Four-Bit Data Interface

    The board uses a 4-bit SF_D interface to the character LCD.
    The SF_D values on SF_D<11:8>, and the register select (LCD_RS) and the read/write (LCD_RW)
    control signals must be set up and stable at least 40 ns before the enable LCD_E goes High.

    The enable signal must remain High for 230 ns or longer-the equivalent of 12 or more clock
cycles at 50 MHz.

    In many applications, the LCD_RW signal can be tied Low permanently
    because the FPGA generally has no reason to read information from the display.

    Transferring 8-Bit Data over the 4-Bit Interface

    After initializing the display and establishing communication,
    all commands and SF_D transfers to the character display are via 8 bits,
    transferred using two sequential 4-bit operations.
    Each 8-bit transfer must be decomposed into two 4-bit transfers, spaced apart by at least
1 μs.

    The upper nibble is transferred first, followed by the lower nibble.
    An 8-bit write operation must be spaced least 40 μs before the next communication.
    This delay must be increased to 1.64 ms following a Clear Display command.

    Note that the period of the 50MHz onboard clock is 20ns.

    The time between corresponding nibbles is 1us, which is equivalent to 50 clock cycles.

    The time between successive commands is 40us, which corresponds to 2000 clock cycles.
    The delay after a Clear Display command is 1.64ms, which corresponds to 82000 clock cycles.

    Setup time ( time for the outputs to stabilize ) is 40ns, which is 2 clock cycles,
    the hold time ( time to assert the LCD_E pin ) is 230ns, which translates to roughly 12
clock cycles,
    and the fall time ( time to allow the outputs to stabilize) is 10ns, which translates to
roughly 1 clock cycle.
*/
    // specified by datasheet, transmit process
        // specified by datasheet, transmit process
    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
                // Each 8-bit transfer must be decomposed into two 4-bit transfers, spaced
apart by at least 1 μs.
                // The upper nibble is transferred first, followed by the lower nibble.
                // The time between corresponding nibbles is 1us, which is equivalent to 50
clock cycles.
                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
                // Setup time ( time for the outputs to stabilize ) is 40ns, which is 2 clock
cycles
                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

                // Hold time ( time to assert the LCD_E pin ) is 230ns, which translates to
roughly 12 clock cycles
                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
                // The time between successive commands is 40us, which corresponds to 2000
clock cycles.
                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
```

2. LCD_Driver.ucf:

```
# Disabled StrataFlash Parallel Flash PROM
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 ;
# The LCD four-bit data interface is shared with the StrataFlash.
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 ;
```

6、状态转移图


Reset
/default
DISPLAY_INIT
init_sate!=4'hB
FUNCTION_SET
ENTER_MODE_SET
tx_delay>2000
DISPLAYON_OFF
tx_delay>2000
tx_delay<=2000
DISPLAY_CLEAR
tx_delay>2000
CLEAR_EXECUTIO
cnt_delay<=82
cnt_delay>8200
IDLE_WAIT
cnt_2sec<2000000
cnt_2sec>=200000
SET_DD_RAM_ADDR
tx_dealy<=2000
tx_delay
>2000
LCD_LINE_1
tx_delay<=2000
||cnt_1<8
tx_delay>2000
&&cnt_1>=9
SET_NEWLINE ADDR
tx_delay<=2000
tx_delay>2000
tx_delay>2000
&&cnt_1>=12
LCD_LINE_2
tx_delay<=2000
||cnt_2<11
IDLE