FPGA的LCD显示

把这次毕设的一些东西整理一下，今天答辩是老师说东西简单，但是没看到在简简单单的LCD显示上，就花了大功夫呢，不吐槽不愉快啊！（我做的主体在FPGA里面啊，多少的代码，外围电路当然简单了…）
进入正题：
用的是LCD1602，资料之类的网上一大推，就不再详细说了。

表3-1 LCD1602的指令集

序号	指令	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标返回	0	0	0	0	0	0	0	0	1	*
3	置输入模式	0	0	0	0	0	0	0	1	I/D	S
4	显示开/关控制	0	0	0	0	0	0	1	D	C	B
5	光标或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	置功能	0	0	0	0	1	DL	N	F	*	*
7	置字符发生存贮器地址	0	0	0	1	字符发生存贮器地址					
8	置数据存贮器地址	0	0	1	显示数据存贮器地址						
9	读忙标志或地址	0	1	BF	计数器地址						
10	写数到CGRAM或DDRAM）	1	0	要写的数据内容							
11	从CGRAM或DDRAM读数	1	1	读出的数据内容							

LCD1602液晶模块内部的控制器共有11条控制指令，如表3-1所示。一个完整的LCD初始化过程为

写指令38H：功能设置(序号7)

延时5ms

写指令08H：显示开关控制(序号8)

延时5ms

写指令01H：显示清屏(序号1)

延时5ms

写指令06H：输入模式设置(序号3)

延时5ms

写指令0CH：显示开及光标设置(序号4)

初始化完成之后就可以利用命令显示自己的字符了。

本次设计采用的是一段式状态机，共分为10个状态，状态图如图3-8所示。

图3-8控制LCD的状态图

其中各个状态描述如下：

IDLE：空闲状态

MODE_SET：模式设置，设定每次定入1位数据后光标的移位方向，并且设定每次写入的一个字符是否移动。

SWITCH_SET：显示控制，设定显示和光标的开关，以及开关是否闪烁。

SHIFT_SET：光标功能设定，设定光标移位或使整个显示屏幕移位。

FUN_SET：功能设定，设定数据总线位数、显示的行数及字型。

CLEAR：清屏状态，调用清屏指令。

ROW1_SET：第一行地址设置。

ROW1_SET：第一行写入。

ROW2_SET：第二行地址设置。

ROW2_SET：第二行写入。

```
//分频模块，得到5ms的时钟信号，2^18/50,000,000=5ms
reg [17:0] cnt ;
reg [31:0] cnt2 ;
always @(posedge clk or negedge rst)
begin
    if(!rst)
        cnt<=0;
    else
        cnt<=cnt+1 ;
end
wire clk_5ms = cnt[17] ;
assign EN = clk_5ms ;//把5ms的时钟信号赋值给使能信号，这样在你写入数据或指令的时候，使能信号是拉高的
//具体可看时序图

reg clk_1s ;//一秒的时钟
always @(posedge clk or negedge rst)
begin
    if(!rst)    begin
        cnt2 <= 0 ;
        clk_1s <= 0 ; end
    else if (cnt2==25000000-1)
        begin
            cnt2 <= 0 ;
            clk_1s <= ~clk_1s ;
        end
    else
        cnt2 <= cnt2+1 ;
end
```

```
//----------------------------------------------------------------------
reg [7:0] data1 [5:0] ;//定义了一个存储器，相当于C语言里面的数组，包括六个8位的变量
reg flag ;
//1小时简易时钟显示
always @(posedge clk_1s or negedge rst)
    begin
        if(!rst)begin
            data1[4]<="0";
            data1[3]<="0";end
        else if(data1[4]<8'h39)
            data1[4] <= data1[4]+1'b1 ;
        else
            begin
                data1[4]<=8'h30 ;
                if(data1[3]<8'h35)
                begin
                    data1[3] <= data1[3]+1 ;
                    flag <= 0;
                end
                else
                    begin
                        data1[3] <= 8'h30 ;
                        flag <= 1 ;
                    end
            end
    end
always @(posedge flag or negedge rst)
    begin
        if(!rst)
            begin
                data1[1]<="0";
                data1[0]<="0";
            end
        else if(data1[1]<8'h39)
            data1[1] <= data1[1]+1'b1 ;
```

```
        else
            begin
                data1[1] <= 8'h30 ;
                if(data1[0]<8'h35)
                    data1[0] <= data1[0]+1 ;
                else
                    data1[0] <= 8'h30 ;
            end
    end

// 状态机控制顺序进行初始化和数据写入操作
//由于状态数较少，使用一段式状态机
reg [4:0] state ;
parameter
    IDLE          =4'b0000 ,//空闲
    CLEAR         =4'b0001 ,//清除
    MODE_SET   =4'b0011 ,//模式设置指令
    //功能：设定每次定入1位数据后光标的移位方向，并且设定每次写入的一个字符是否移动。
    //0000_01I/DS
    //I/D    0=写入新数据后光标左移     1=写入新数据后光标右移
    // S     0=写入新数据后显示屏不移动 1=写入新数据后显示屏整体右移1个字符
    SWITCH_SET      =4'b0010 ,//光标/显示开关控制指令
    //0000_1DCB
    //D:0=显示功能关  1=显示功能开,C:0=无光标    1=有光标,B:0=光标闪烁   1=光标不闪烁
    SHIFT_SHOW      =4'b0110 ,//光标和显示屏移动方向设置
    //0001_SRXX
    // 0    0      0100
    //    0    1      光标右移1格，且AC值加1
   //   1    0      显示器上字符全部左移一格，但光标不动
    //    1    1      显示器上字符全部右移一格，但光标不动
    FUN_SET          =4'd0111 ,//功能设定
    //001D_NFXX
    //DL    0=数据总线为4位     1=数据总线为8位
    //N 0=显示1行          1=显示2行
    //F      0=5×7点阵/每字符    1=5×10点阵/每字符
```

```
    ROW1_SET  =4'b0101 ,//第一行地址设置
    ROW1_WRITE    =4'b0100 ,//第一行写入
    ROW2_SET  =4'b1100 ,//第二行地址设置
    ROW2_WRITE    =4'b1101 ,//第二行写入
    STOP          =4'b1111;//


//用状态机进行初始化和数据写入
reg [7:0] address ;
always @(posedge clk_5ms or negedge rst)
begin
    if(!rst)
        begin
            state <= IDLE ;
            address <= 8'd0 ;
            LCD_DATA <= 8'd0 ;
            RS <= 0 ;
            RW <=0 ;
        end
    else
        begin
            case(state)
                IDLE          :
                    begin
                        LCD_DATA <= 8'bzzzz_zzzz ;
                        state <= FUN_SET ;
                    end
                FUN_SET       :   ///八位数据位宽，两行，5x10点阵
                    begin
                        RS <= 0 ;//
                        RW <= 0 ;//写入指令
                        LCD_DATA <= 8'b0011_1100 ;//指令和数据都通过LCD_DATA送入到FPGA
                        state <= MODE_SET ;
                    end
                MODE_SET  :   //模式设置，每个字符光标右移
```

```
        begin     RS <= 0 ;
            RW <= 0 ;
            LCD_DATA <= 8'b0000_0110 ;
            state <= SWITCH_SET ;
        end
    SWITCH_SET     :    //无光标
        begin     RS <= 0 ;
                RW <= 0 ;
                LCD_DATA <= 8'b0000_1100 ;
                state <= SHIFT_SHOW ;
        end
    SHIFT_SHOW     :    //光标移动，显示屏不移动
        begin
            RS <= 0 ;
            RW <= 0 ;
            LCD_DATA <= 8'b0001_0100 ;
            state <= CLEAR ;
        end
    CLEAR       :
        begin
            RS <= 0 ;
            RW <= 0 ;
            LCD_DATA <= 8'b0000_0001 ;
            state <= ROW1_SET ;
        end


    ROW1_SET   :
        begin
            RS <= 0 ;
            RW <= 0 ;
            LCD_DATA <= 8'h80+8'h0 ;
```

//从指令8中可以看到，写入数据的第一位数D7是1，也就是第一行第一格的写入的数据应为1000_0000，就是8'h80，但是第二行第一格写入的地址是8'hC0，虽然一行仅有16格，但是地址数却又8'h40个呢，所以就不要指望它写完第一行后就开始写第二行了…

```
            state <= ROW1_WRITE ;
            address <= 8'd0 ;
```