LCD1602显示之FPGA

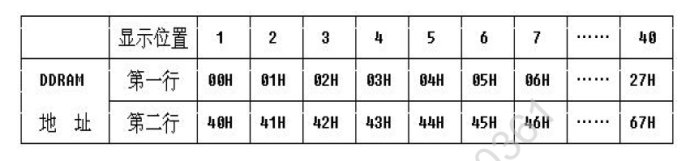
作者:kb129   来源:kb129   点击数:x  更新时间：2014年06月08日   【字体：[大](javascript:ContentSize(16)) [中](javascript:ContentSize(14)) [小](javascript:ContentSize(13))】

    本科准备电赛的时候，学习单片机，由于自己没有LCD，没有碰。刚开始学FPGA时候，刚准备学LCD，自己又出了些事，后来一直忙到现在。如今再来看看LCD，感触颇多，废话不多说，代码加文档走起。

      lcd1602应该算是一个难点，不管是对于单片机的学习还是FPGA的学习。因为里面涉及到时序分析，地址度写，数据读写，指令读写，建立时间和保持时间，还有操作流程会变的复杂，这些都给刚学习的人一些困惑或是难以理解。在我的博文中，一直贯彻的理念是，希望能够起到抛砖引玉的作用，尽量将思考的过程描述清楚，而不是简单的写出相关知识和代码。

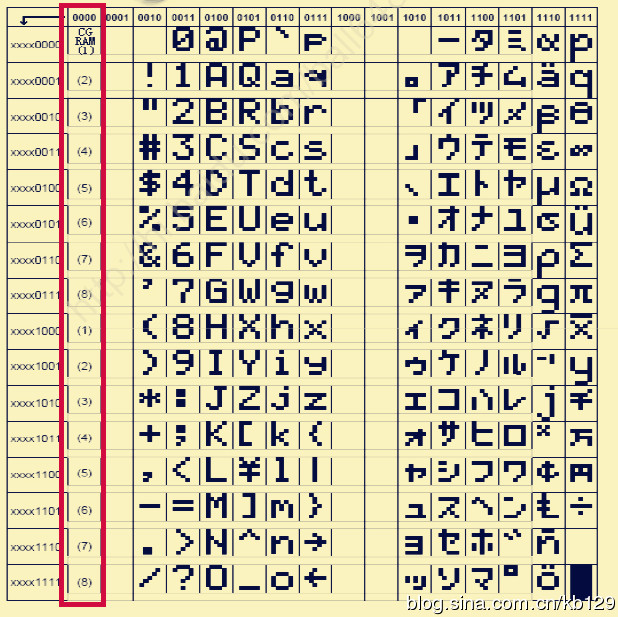
1.文档分析：

     关于lcd1602的文档有很多，自己搜索一下网。市面上绝大部分都是基于HD44780，所以这里的控制显示等都是一样的。HD44780内置了DDRAM、CGROM、CGRAM.其中要让lcd显示就需要将DDRAM的相应地址写入数据。

[](http://www.51hei.com/UpFiles/up/0/4681158113103.jpg)  
  
[](http://www.51hei.com/UpFiles/up/0/46811581116216.jpg)  
上图可知DDRAM一共有40个地址，但是对应于1602显示，只能有32个地址有效。这是因为1602可以显示上下两行，每一行显示16符号，一共显示32个符号，每个显示对应于DDRAM一个地址。例如，我需要在1602的第一行最左边显示一个字母A。首先找到第一行最左边对应DDRAM的地址是什么，查看上图可知是：00H，然后大写字母A对应于ASCII中为41H，此时我们只需要给DDRAM的00H地址写个数据41H即可显示了。

问题2：为何写个41H，就可以显示为"A"呢？

       对于这个问题，就需要理解CGROM和CGRAM的作用。在芯片HD44780中内置了192个常用字符的字模，存于CGROM（character generate ROM）中，还有8个允许用户自定义字符（也就是可以显示八个中文字）的RAM，也就是CGRAM。具体描述为下图：

[](http://www.51hei.com/UpFiles/up/0/46811591195714.jpg)  
可以从上图分析A在字模中代码：高4位为0100，低4位为0001.所以组成8位就变成了41H，这就说明了为何写入41H就可以显示“A”。

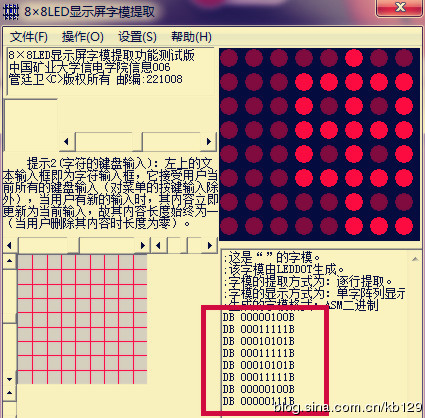
上图红框里面表示为CGRAM，字模代码为：00H-0FH；ASCII的字模代码为：20H-7FH；日文和希腊字符的字模代码为：A0H-FFH；10H-1FH和80H-9FH没有使用。

问题3：我要任意显示一个字母，数字怎么办？

      这个问题是接着上面一个问题而言，具体就是：在1602中我要在某一行某个位置显示我想要的数字或是字母，我应该对应DDRAM地址写个什么样的八位数据？例如，我想显示“1”，那不是就写个01H呢？此时就需要一个思维转换，我们要显示的“1”不再是一个数据，而是需要转换为一个图案，可以看到上图有1的图案，该图案对应了31H，所以需要显示一个“1”，我们就需要给1602的数据总线（DB7--DB0）输入31H。以此类推，例如我们需要输入kb129 is a good man，于是就需要给1602顺序输入：6BH，62H，31H，32H，39H ，20H（空格），69H，73H，20H，61H，20H，67H，6FH，6FH，64H，20H，6DH，61H，6EH。

问题4：那显示汉字怎么办呢？

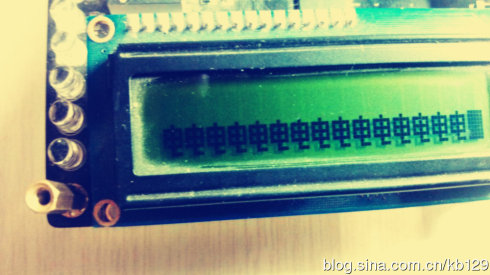
       问题2中解决了显示任意一个字母和数字，但是汉字在图中找不到汉字，怎么办？这时候需要使用CGRAM了，先用字模软件，将对应汉字的变为二进制数。

[](http://www.51hei.com/UpFiles/up/0/4681111259900.jpg)  
例如我想要显示一个“电”字，由于1602中显示的图案为5\*7或是5\*10,所以在8\*8中左边三列不能使用。得到8列八位数据：04, 1F, 15，1F, 15，1F, 04，07.

然后就需要将这8个8bit数据写入CGRAM中，写CGRAM需要使用指令：

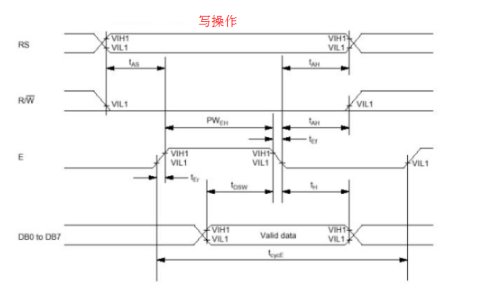
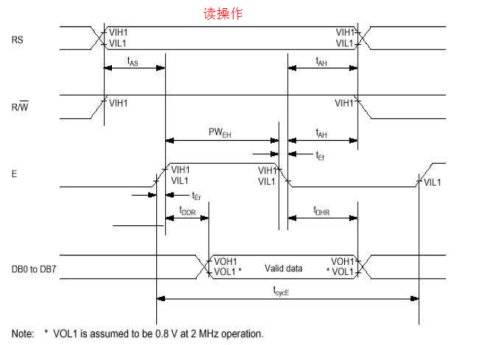
[](http://www.51hei.com/UpFiles/up/0/4681121229673.jpg)  
可以设置地址指针自加一模式，所以如果我们想把“电”这个字方在第1个CGRAM中，也就是对应DDRAM中的00H，就需要将地址写为DB7--DB0：0100\_0000.然后将数据04, 1F, 15，1F, 15，1F, 04，07依次写进CGRAM中。这样在CGROM字符的字模中00H就代表了“电”。

最后就是显示，也就是如果需要将“电”显示在1602中，就讲地址指针指向DDRAM，然后写数据为00H。

[](http://www.51hei.com/UpFiles/up/0/4681121218842.jpg)

2.关注时序

      上面已经了解到要让1602显示就需要将特地的地址输入特地的值，如果你刚接触1602，此时一定特别想了解怎么将上面的思路转换为verilog代码。但是一定需要注意时序问题，1602的读写时序为：

[](http://www.51hei.com/UpFiles/up/0/4681131285542.jpg)  
[](http://www.51hei.com/UpFiles/up/0/4681141229276.jpg)  
  
这里特别注意了setup和 hold time，但是实际使用中由于1602显示不需要很高的时序，所以我们只需降低1602工作的时钟就可以很容易满足1602的时序要求。

    在上篇博文中已经说了1602显示原理，在这篇主要是讲讲怎么将那些原理转换为verilog代码。1602的显示有很多种选择，一共有11条指令，可以根据不同设置，达到不同的显示，下面给了verilog显示代码，可以以此为基础，修改为具体应用。由于明天是五一劳动节，所以祝大家五一快乐。

代码显示结果：

[](http://www.51hei.com/UpFiles/up/0/468117136270.jpg)  
verilog代码：

1.代码顶层

//data: 2014-04-28  
//addr: kb129  
//info: this is the top of the LCD  
module lcd\_1602(  
         clk\_50M,  
         rst,  
         en,  
         RS,  
         RW,  
        data  
 );  
input      clk\_50M;  
input      rst;  
output    en;  
output    RS;  
output    RW;  
output    [7:0] data;  
wire        clk\_500;  
clk50M\_500   u\_clk50M\_500  
(  
    .clk\_50M(clk\_50M),  
    .rst(rst),  
    .clk\_500(clk\_500)  
 );   
lcd\_show u\_lcd\_show  
(  
     .clk\_LCD(clk\_500),  
     .rst(rst),  
     .en(en),  
     .RS(RS),  
     .RW(RW),  
     .data(data)  
 );  
endmodule  
2.时钟模块

//data: 2014-04-28  
//addr: kb129  
//info: change the clk 50Mhz to 500Hz  
module clk50M\_500(  
      clk\_50M,  
      rst,  
      clk\_500  
 );  
input         clk\_50M;  
input         rst;  
output      clk\_500;

reg    [8:0] cnt\_1;  
reg    [7:0] cnt\_2;  
reg           clk\_500hz;  
always@(posedge clk\_50M)  
begin  
      if(!rst)  
           begin  
                 clk\_500hz <= 0;  
                     cnt\_1   <= 0;  
                     cnt\_2   <= 0;  
            end  
 else if(cnt\_2==8'd199)  
            begin  
                    cnt\_2 <= 0;  
                    if(cnt\_1==9'd499)  
                          begin  
                                cnt\_1   <= 0;  
                                clk\_500hz <= ~clk\_500hz;  
                           end  
                  else  
                         cnt\_1   <= cnt\_1+1;  
             end  
        else    cnt\_2   <= cnt\_2+1;  
end   
assign clk\_500 = clk\_500hz;  
endmodule

3.显示模块

//module: lcd\_show.v  
//data:2014-04-30  
//addr: kb129           
//info: this is all the lcd module ,can show 8 zhongwen .  
module lcd\_show(  
        clk\_LCD,  
        rst,  
        en,  
        RS,  
        RW,  
        data  
);  
input        clk\_LCD;  // 500Hz  
input        rst;        
output     en,RS,RW;  
output   reg  [7:0]  data;  
reg                 RS,en\_sel;  
reg      [4:0]    disp\_count;  
reg      [4:0]   wrtie\_count;  
reg      [2:0]   num;  
reg      [3:0]   state;  
parameter   clear\_lcd           = 4'b0000,                    //清屏并光标复位  
                  set\_disp\_mode   = 4'b0001,                    //设置显示模式：8位2行5x7点阵     
                  disp\_on             = 4'b0010,                   //显示器开、光标不显示、光标不允许闪烁  
                      shift\_down    = 4'b0011,                    //文字不动，光标自动右移  
                    write\_cgram    = 4'b0100,                    //写中文进入CGRAM，以显示中文    
                  write\_data\_first  = 4'b0101,                    //写入第一行显示的数据  
             write\_data\_second  = 4'b0110,                    //写入第二行显示的数据  
                                 idel     = 4'b0111;                    //空闲状态     
assign  RW = 1'b0;                            //RW=0时对LCD模块执行写操作  
assign  en = en\_sel ? clk\_LCD : 1'b0;  
     
reg [7:0] data\_character  [7:0];    //this is 五   00H  
reg [7:0] data\_character2 [7:0];    //节           01H  
reg [7:0] data\_character3 [7:0];   //日           02H  
always @(posedge clk\_LCD )  
begin  
    data\_character[0] <= 8'h00;  
    data\_character[1] <= 8'h1e;  
    data\_character[2] <= 8'h08;  
    data\_character[3] <= 8'h1e;  
    data\_character[4] <= 8'h0a;  
    data\_character[5] <= 8'h0a;  
    data\_character[6] <= 8'h1F;  
    data\_character[7] <= 8'h00;  
    data\_character2[0] <= 8'h0A;  
    data\_character2[1] <= 8'h1f;  
    data\_character2[2] <= 8'h0A;  
    data\_character2[3] <= 8'h1f;  
    data\_character2[4] <= 8'h05;  
    data\_character2[5] <= 8'h05;  
    data\_character2[6] <= 8'h05;  
    data\_character2[7] <= 8'h04;  
    data\_character3[0] <= 8'h00;  
    data\_character3[1] <= 8'h1F;  
    data\_character3[2] <= 8'h11;  
    data\_character3[3] <= 8'h11;  
    data\_character3[4] <= 8'h1f;  
    data\_character3[5] <= 8'h11;  
    data\_character3[6] <= 8'h11;  
    data\_character3[7] <= 8'h1f;  
end  
  
reg [7:0]     data\_first\_line      [15:0];  //first line show data  
reg [7:0]     data\_second\_line [15:0];  //second line show data  
always @(posedge clk\_LCD )  
begin  
   data\_first\_line[0] <= 8'h54;  
   data\_first\_line[1] <= 8'h6F;  
   data\_first\_line[2] <= 8'h20;  
   data\_first\_line[3] <= 8'h6d;  
   data\_first\_line[4] <= 8'h79;  
   data\_first\_line[5] <= 8'h20;  
   data\_first\_line[6] <= 8'h66;  
   data\_first\_line[7] <= 8'h72;  
   data\_first\_line[8] <= 8'h69;  
   data\_first\_line[9] <= 8'h65;  
   data\_first\_line[10] <= 8'h6e;  
   data\_first\_line[11] <= 8'h64;  
   data\_first\_line[12] <= 8'h73;  
   data\_first\_line[13] <= 8'h8a;  
 data\_second\_line[1] <= 8'h00;  
 data\_second\_line[2] <= 8'h2d;  
 data\_second\_line[3] <= 8'h01;  
 data\_second\_line[4] <= 8'h02;  
 data\_second\_line[5] <= 8'h68;  
 data\_second\_line[6] <= 8'h61;  
 data\_second\_line[7] <= 8'h70;  
 data\_second\_line[8] <= 8'h70;  
 data\_second\_line[9] <= 8'h79;  
end  
  
always @(posedge clk\_LCD or negedge rst)  
begin  
   if(!rst)  
      begin  
          state         <= clear\_lcd;             //复位：清屏并光标复位    
          RS             <= 1'b0;                  //复位：RS=0时为写指令；                        
          data          <= 8'b0;                  //复位：使DB8总线输出全0  
          en\_sel        <= 1'b1;                  //复位：开启夜晶使能信号  
          disp\_count <= 5'b0;  
                 num   <= 3'b0;  
        wrtie\_count <= 5'b0;  
      end  
   else  
      case(state)  
      clear\_lcd:                               //初始化LCD模块  
             begin          //清屏并光标复位  
                state  <= set\_disp\_mode;  
                data  <= 8'h01;                 
             end  
      set\_disp\_mode:        //设置显示模式：8位2行5x8点阵   
             begin  
                state  <= disp\_on;  
                data  <= 8'h38;                                
             end  
      disp\_on:            //显示器开、光标不显示、光标不允许闪烁  
             begin  
                state  <= shift\_down;  
                data  <= 8'h0c;                             
             end  
      shift\_down:        //文字不动，光标自动右移   
            begin  
                state  <= write\_cgram;  
                data  <= 8'h06;                           
            end  
      write\_cgram:       //写CGRAM  
            begin  
    case(num)  
    0:begin  
             data  <= 8'h40;        //the first character addr  
             num   <= num+1;  
             state <= write\_cgram;  
      end  
    1:begin  
             if(wrtie\_count==8)  
                  begin  
                        data <= 8'h48;  //the second character addr  
                        RS   <= 1'b0;  
                        num  <= num+1;  
                        state<= write\_cgram;  
                        wrtie\_count <= 0;  
                 end  
           else  
                 begin  
                        data <= data\_character[wrtie\_count];  
                        RS   <= 1'b1;  
                        wrtie\_count <= wrtie\_count + 1'b1;  
                        state     <= write\_cgram;  
                   end            
      end  
    2:begin  
            if(wrtie\_count==8)  
                  begin  
                          data <= 8'h50;  //the second character addr  
                          RS   <= 1'b0;  
                          num  <= num+1;  
                          state<= write\_cgram;  
                          wrtie\_count <= 0;  
                   end  
             else  
                   begin  
                        data <= data\_character2[wrtie\_count];  
                        RS   <= 1'b1;  
                        wrtie\_count <= wrtie\_count + 1'b1;  
                        state     <= write\_cgram;  
                 end        
      end  
    3:begin  
            if(wrtie\_count==8)  
                   begin  
                           data <= 8'h80;  //the DDROM first line start addr  
                           RS   <= 1'b0;

                           state<= write\_data\_first;  
                          wrtie\_count <= 0;  
                   end  
           else  
                   begin  
                         data <= data\_character3[wrtie\_count];  
                         RS   <= 1'b1;  
                         wrtie\_count <= wrtie\_count + 1'b1;  
                        state     <= write\_cgram;  
                   end        
      end  
       endcase  
   end  
      write\_data\_first:              //显示第一行                           
            begin  
                if(disp\_count == 14)                        
                    begin  
                        data    <= 8'hc2;                 
                        RS     <= 1'b0;  
                        disp\_count   <= 4'b0;  
                        state    <= write\_data\_second;          
                    end  
                else  
                    begin  
                        data    <= data\_first\_line[disp\_count];  
                        RS     <= 1'b1;                    
                        disp\_count   <= disp\_count + 1'b1;  
                        state    <= write\_data\_first;  
                    end  
            end  
      write\_data\_second:                      //显示第二行  
            begin  
                if(disp\_count == 9)  
                    begin  
                        en\_sel   <= 1'b0;  
                        RS    <= 1'b0;  
                        disp\_count  <= 4'b0;  
                        state   <= idel;                       
                    end  
                else  
                    begin  
                        data    <= data\_second\_line[disp\_count+1];  
                        RS     <= 1'b1;  
                        disp\_count   <= disp\_count + 1'b1;  
                        state    <= write\_data\_second;  
                    end               
            end  
      idel:            //写完进入空闲状态  
            begin  
                state <=  idel;             //在Idel状态循环   
            end  
      default:  state <= clear\_lcd;         //若state为其他值，则将state置为Clear\_Lcd  
      endcase  
end  
endmodule

4.11条指令详细说明

NO1：

[](http://www.51hei.com/UpFiles/up/0/46811181373856.jpg)  
NO.2

[](http://www.51hei.com/UpFiles/up/0/46811191327635.jpg)

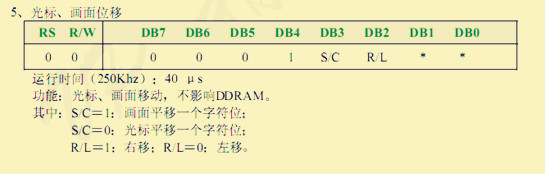
NO.3

[](http://www.51hei.com/UpFiles/up/0/4681119135062.jpg)

NO.4

[](http://www.51hei.com/UpFiles/up/0/46811201364839.jpg)

NO.5

[](http://www.51hei.com/UpFiles/up/0/46811201315749.jpg)

NO.6

[](http://www.51hei.com/UpFiles/up/0/46811211318615.jpg)

NO.7

[](http://www.51hei.com/UpFiles/up/0/46811221394379.jpg)

NO.8 设定DDRAM地址

[](http://www.51hei.com/UpFiles/up/0/46811231321212.jpg)

NO.9

[](http://www.51hei.com/UpFiles/up/0/46811231391977.jpg)

NO.10

[](http://www.51hei.com/UpFiles/up/0/46811241334530.jpg)

NO.11 从CGRAM和DDRAM中读取数据

[](http://www.51hei.com/UpFiles/up/0/46811251378725.jpg)