至简设计系列_LCD 入门案例_边框显示

--作者：喝喝

1.1 总体设计

1.1.1 概述

液晶显示器是一-种通过液晶和色彩过滤器过滤光源，在平面面板上产生图像的数字显示器。LCD 的构造是在两片平行的玻璃基板当中放置液晶盒，下基板玻璃上设置薄膜晶体管,.上基板玻璃上设置彩色滤光片，通过薄膜晶体管上的信号与电压改变来控制液晶分子的转动方向，从而达到控制每个像素点偏振光出射与否而达到显示目的。与传统的阴极射线管相比，LCD 具有占用空间小，低功耗，低辐射，无闪烁，降低视觉疲劳等优点。现在 LCD 已渐替代 CRT 成为主流，价格也已经下降了很多，并已充分的普及。

本设计的主要任务是基于 FPGA 的 LCD 显示控制器设计，兼顾程序的易用性，方便此后模块的移植和应用。采用 VHDL 硬件描述语言在 QUARTUS II 软件平台上实现 FPGA 对 LCD 的控制，在 LCD 模块上实现任意彩色图片的显示，与此同时还须实现实时刷新数据的功能。这将有助于采用 FPGA 的系列产品的开发，特别是需要用到 LCD 而采用 FPGA 的产品的开发。不但缩短了 FPGA 的开发周期，也使更多采用 FPGA 设计的产品上出现 LCD，增加了人机之间的交互性。

1.1.2 设计目标

此设计通过 fpga 给 lcd 发送图片信息，然后直接在 LCD 显示出图片

1.1.3 信号列表

信号名	接口方向	定义
clk_50m	输入	系统时钟
rst_n	输入	低电平复位信号
lcd_hsync	输出	行同步信号

lcd_vsync	输出	场同步信号
lcd_de	输出	行和场同时显示时序段 有效显示数据段信号
lcd_rgb	输出	显示颜色 RGB [23:16]:表示的是 R[7:0] [15:8]:表示的是 G[7:0] [7:0]:表示的是 B[7:0]
lcd_dclk	输出	像素时钟信号

1.1.4 设计思路

设计行显示时序段和场显示时序段，来确定矩形边框的宽度，根据各种颜色的数值来确定 lcd 显示屏显示出的边框颜色

行时钟计数器 cnt_hys：用来计算行同步信号的帧长，加一条件是 1，结束条件为数到 1056 个像素就结束

场时钟计数器 cnt_vys：用来计算场同步信号的帧长，加一条件是场信号每数到 1056 个像素（即为一行结束的时刻），结束条件为数到 525 行就结束

1.1.5 参考代码

```
1.  module mdyLcdDispRect(
2.      clk_50m     ,
3.      rst_n       ,
4.
5.      lcd_hsync   ,
6.      lcd_vsync   ,
7.      lcd_de      ,
8.
9.
10.     lcd_rgb     ,
11.     lcd_dclk
12.
13. );
14.
15.     input             clk_50m     ;
16.     input             rst_n       ;
```

```
17.     output              lcd_hsync   ;
18.     output              lcd_vsync   ;
19.     output              lcd_de      ;
20.
21.     output  [23:0]      lcd_rgb     ;
22.     output              lcd_dclk    ;
23.
24.
25.
26.     reg                 lcd_hsync   ;
27.     reg                 lcd_vsync   ;
28.
29.     reg     [23:0]      lcd_rgb     ;
30.
31.
32.     parameter     LINE_PR   =  1056 ;
33.     parameter     FRAME_PER =   525 ;
34.
35.
36.     parameter     H_SYNC    =    20 ;
37.     parameter     V_SYNC    =    10 ;
38.
39.     parameter     HDE_START =    46 ;
40.     parameter     HDE_END   =   846 ;
41.     parameter     VDE_START =    23 ;
42.     parameter     VDE_END   =   503 ;
43.
44.
45.
46.     reg    [12:0]     cnt_hsy       ;
47.     reg    [12:0]     cnt_vsy       ;
48.     reg               hsync_de      ;
49.     reg               vsync_de      ;
50.
51.     wire              display_area  ;
52.     wire              e_area        ;
53.     wire              add_cnt_hsy   ;
54.     wire              end_cnt_hsy   ;
55.     wire              add_cnt_vsy   ;
56.     wire              end_cnt_vsy   ;
57.     reg [ 7:0]        cnt0          ;
58.     wire              add_cnt0      ;
59.     wire              end_cnt0      ;
```

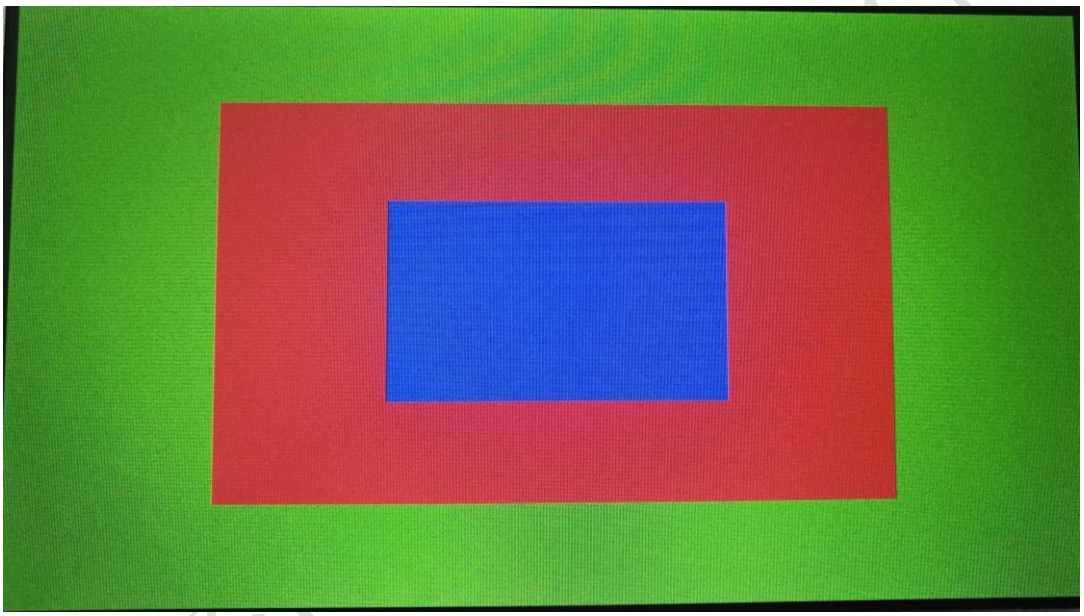
```
60.     reg  [15:0]       cnt1         ;
61.     wire              add_cnt1     ;
62.     wire              end_cnt1     ;
63.
64.
65.     assign clk       = clk_50m             ;
66.     assign lcd_dclk  = ~ clk_50m           ;
67.     assign lcd_de    = hsync_de & vsync_de  ;
68.
69.
70.
71.
72.     always @ (posedge clk or negedge rst_n)begin
73.             if(!rst_n)begin
74.                 cnt_hsy <= 0;
75.             end
76.             else if(add_cnt_hsy)begin
77.                 if(end_cnt_hsy)
78.                     cnt_hsy <= 0;
79.                 else
80.                     cnt_hsy <= cnt_hsy + 1;
81.             end
82.         end
83.
84.     assign add_cnt_hsy = 1;
85.     assign end_cnt_hsy = add_cnt_hsy && cnt_hsy == LINE_PR -1;
86.
87.
88.     always @ (posedge clk or negedge rst_n)begin
89.         if(!rst_n)begin
90.             cnt_vsy <= 0;
91.         end
92.         else if(add_cnt_vsy)begin
93.             if(end_cnt_vsy)
94.                 cnt_vsy <= 0;
95.             else
96.                 cnt_vsy <= cnt_vsy + 1;
97.         end
98.     end
99.
100.    assign add_cnt_vsy = end_cnt_hsy;
101.    assign end_cnt_vsy = add_cnt_vsy && cnt_vsy == FRAME_PER - 1;
102.
```

```
103.
104.    always @ (posedge clk or negedge rst_n)begin
105.        if(!rst_n)begin
106.            lcd_hsync <= 1'b0 ;
107.        end
108.        else if(end_cnt_hsy)begin
109.            lcd_hsync <= 1'b0;
110.        end
111.        else if(add_cnt_hsy && cnt_hsy == H_SYNC-1 )begin
112.            lcd_hsync <= 1'b1;
113.        end
114.    end
115.
116.    always @ (posedge clk or negedge rst_n)begin
117.        if(!rst_n)begin
118.            hsync_de <= 1'b0;
119.        end
120.        else if(add_cnt_hsy && cnt_hsy == HDE_START-1)begin
121.            hsync_de <= 1'b1;
122.        end
123.        else if(add_cnt_hsy && cnt_hsy == HDE_END-1)begin
124.            hsync_de <= 1'b0;
125.        end
126.    end
127.
128.
129.    always @ (posedge clk or negedge rst_n)begin
130.        if(!rst_n)begin
131.            lcd_vsync <= 1'b0 ;
132.        end
133.        else if(add_cnt_vsy && cnt_vsy == V_SYNC-1 )begin
134.            lcd_vsync <= 1'b1;
135.        end
136.        else if(end_cnt_vsy)begin
137.            lcd_vsync <= 1'b0;
138.        end
139.
140.    end
141.
142.
143.    always @ (posedge clk or negedge rst_n)begin
144.        if(!rst_n)begin
145.            vsync_de <= 1'b0;
```

```
146.        end
147.        else if(add_cnt_vsy && cnt_vsy == VDE_START-1)begin
148.            vsync_de <= 1'b1;
149.        end
150.        else if(add_cnt_vsy && cnt_vsy ==VDE_END-1)begin
151.            vsync_de <= 1'b0;
152.        end
153.    end
154.
155.
156.    assign   display_area = hsync_de && vsync_de;
157.
158.
159.    assign   blue_area = (cnt_hsy >= HDE_START + 400-125) &&
    (cnt_hsy<HDE_START+400+125) &&
160.                         (cnt_vsy >= VDE_START + 240-80) &&
    (cnt_vsy<VDE_START+240+80) ;
161.
162.
163.    assign   reb_area  = (cnt_hsy >= HDE_START + 400-250) &&
    (cnt_hsy<HDE_START+400+250) &&
164.                         (cnt_vsy >= VDE_START + 240-160) &&
    (cnt_vsy<VDE_START+240+160) ;
165.
166.
167.    always @ (posedge clk or negedge rst_n)begin
168.        if(!rst_n)begin
169.             lcd_rgb <= 0;
170.        end
171.        else if(display_area)begin
172.            if(blue_area)begin
173.                lcd_rgb <= 24'h00_00_ff ;
174.            end
175.            else if(reb_area)begin
176.                lcd_rgb <= 24'hff_00_00 ;
177.            end
178.            else begin
179.                lcd_rgb <= 24'h00_ff_00 ;
180.            end
181.        end
182.        else begin
183.            lcd_rgb <= 0;
184.        end
```

```
185.    end
186.
187.
188.
189.endmodule
190.
191.
192.
```

1.2 效果和总结



本案例我们设计了蓝色、红色和绿色的矩形框，蓝色和红色的矩形框的场信号是 160 行、行信号是 250 个像素；绿色的矩形框的场信号是 160 行、行信号是 300 个像素，所以我们后面就得到一个 160*250 的蓝色矩形框、500*320-250*160 的红色矩形边框和一个 800*480-500*320 的绿色矩形边框

在这个设计案例中，至简设计法和明德扬计数器模板发挥了至关重要的作用，使我能够快速准确完成设计。希望有兴趣的同学可以运用至简设计法和明德扬模板尝试一下拓展设计哦。

感兴趣的朋友也可以访问明德扬论坛（http://www.fpgabbs.cn/）进行 FPGA 相关工程设计学习，也可以看一下我们往期的文章：

《基于 FPGA 的密码锁设计》

《波形相位频率可调 DDS 信号发生器》

《基于 FPGA 的曼彻斯特编码解码设计》

《基于 FPGA 的出租车计费系统》

《数电基础与 Verilog 设计》

《基于 FPGA 的频率、电压测量》

《基于 FPGA 的汉明码编码解码设计》
《关于锁存器问题的讨论》
《阻塞赋值与非阻塞赋值》
《参数例化时自动计算位宽的解决办法》

1.3 公司简介

明德扬是一家专注于 FPGA 领域的专业性公司，公司主要业务包括开发板、教育培训、项目承接、人才服务等多个方向。

点拨开发板——学习 FPGA 的入门之选。

MP801 开发板——千兆网、ADDA、大容量 SDRAM 等，学习和项目需求一步到位。

网络培训班——不管时间和空间，明德扬随时在你身边，助你快速学习 FPGA。

周末培训班——明天的你会感激现在的努力进取，升职加薪明德扬来助你。

就业培训班——七大企业级项目实训，获得丰富的项目经验，高薪就业。

专题课程——高手修炼课：提升设计能力；实用调试技巧课：提升定位和解决问题能力；FIFO 架构设计课：助你快速成为架构设计师；时序约束、数字信号处理、PCIE、综合项目实践课等你来选。

项目承接——承接企业 FPGA 研发项目。

人才服务——提供人才推荐、人才代培、人才派遣等服务。