至简设计系列_上位机控制 LCD 显示放大和缩小图片

--作者：肖肖肖

本文为明德扬原创及录用文章，转载请注明出处！

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。

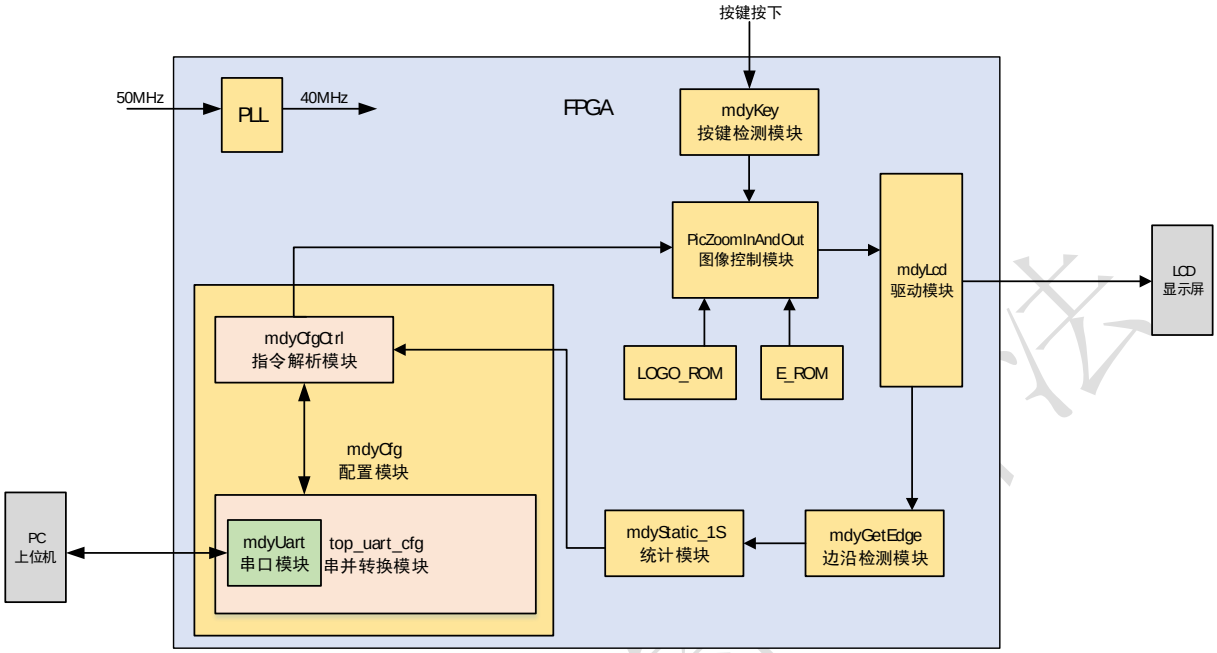
1.1.2 设计目标

1. 在 7 寸 LCD 显示屏上实现明德扬 LOGO 图标的左上角显示和大写字母“E”图片的居中显示；
2. 可以选择使用上位机还是按键来控制“E”图片的放大和缩小；
3. 并且统计 1 秒时间图像的帧率。

1.1.3 系统结构框图

系统结构框图如下所示：

明德扬FPGA至简设计法



图一

1.1.4 模块功能

以下本工程所用到的所有模块中，除了 PicZoomInAndOut 模块都是明德扬提供的可用的 IP 模块，这些模块不允许修改避免调用 IP 模块使用时出现问题。

➢ mdyPllAltera 模块实现功能

1. 将输入的 50MHz 时钟分频输出 40MHz 的时钟。

➢ mdyRom 模块实现功能

1. 存储明德扬 LOGO 图标数据；
2. 存储大小字母“E”图像数据。

➢ mdyCfg 模块实现功能

top_uart_cfg

1. 对接收的数据进行串并转换；
2. 对发送的数据进行并串转换。

mdyCfgCtrl

1. 对传输的指令进行解析。

- **mdyKey 模块实现功能**

1. 检测按下的按键并输出对应按键有效的数值。

- **PicZoomInAndOut 模块实现功能**

1. 规定了明德扬 LOGO 图标和大写字母“E”显示的区域；
2. 控制大写字母“E”显示的放大和缩小。

- **mdyLcd 驱动模块实现功能**

1、 产生驱动 LCD 屏显示的时序。

- **mdyGetEdge 模块实现功能**

1. 对信号的边沿进行检测。

- **mdyStatic_1S 模块实现功能**

1. 统计一段时间内的某些信号的数据。

1.1.5 顶层信号

信号名	I/O	位宽	定义
clk	I	1	系统工作时钟 50M
rst_n	I	1	系统复位信号，低电平有效
key	I	4	独立按键输入信号
uart_rxd	I	1	串口接收接口信号
uart_txd	O	1	串口发送接口信号
lcd_hs	O	1	LCD 行时序信号
lcd_vs	O	1	LCD 场时序信号

明德扬FPGA至简设计法

lcd_de	O	1	LCD 数据输入使能信号
lcd_dat	O	24	LCD RGB 信号，RGB 格式为使用 24 位来表示一个像素，RGB 分量都用 8 位表示，取值范围为 0-255。
lcd_clk	O	1	LCD 数据采样时钟

1.1.6 参考代码

```
1. module top_mdyLcdPicZoomInAndOut(
2.     clk          ,
3.     rst_n        ,
4.     key          ,
5.     uart_rxd     ,
6.
7.     uart_txd     ,
8.     lcd_hs       ,
9.     lcd_vs       ,
10.      lcd_de       ,
11.      lcd_dat      ,
12.      lcd_clk
13.      );
14.
15.      parameter    PICTURE_W    = 24   ;
16.      parameter    KEY_W        =  4   ;
17.
18.
19.      input                     clk          ;//50MHz
20.      input                     rst_n        ;
21.      input    [KEY_W-1 :0]     key          ;
22.      input                     uart_rxd     ;
23.
24.      output                    uart_txd     ;
25.      output                    lcd_hs       ;
26.      output                    lcd_vs       ;
27.      output                    lcd_de       ;
28.      output   [PICTURE_W-1:0] lcd_dat      ;
29.      output                    lcd_clk      ;
```

```
30.
31.       wire                          uart_txd      ;
32.       wire                          lcd_hs        ;
33.       wire                          lcd_vs        ;
34.       wire                          lcd_de        ;
35.       wire    [PICTURE_W-1:0]     lcd_dat       ;
36.       wire                          lcd_clk       ;
37.
38.       wire                          clk_0         ;
39.       wire      [23:0]              pic_data      ;
40.       wire      [10:0]              req_x         ;
41.       wire      [ 9:0]              req_y         ;
42.       wire                          frame_start ;
43.       wire      [KEY_W-1:0]          key_vld       ;
44.       wire                          frame_pos     ;
45.       wire      [63:0]              uart_dout             ;
46.       wire                          uart_dout_vld       ;
47.       wire      [63:0]              cfgCtrl_dout          ;
48.       wire                          cfgCtrl_dout_vld    ;
49.
50.       `include "mdyCfg_wire.v"
51.
52. /******************     PLL 模块    *********************************/
53.
54. mdyPllAltera#(.C0_M(4),.C0_D(5)) u1_pll_40m(
55.       .areset      (~rst_n     ), //高电平有效
56.       .inclk0      (clk        ),
57.       .c0           (clk_0     ),
58.       .c1           (       ),
59.       .c2           (       ),
60.       .c3           (       ),
61.       .c4           (       ),
62.       .locked      (       )
63.
64. );
65.
66. /******************     LCD 驱动模块    *********************************/
67.
68. mdyLcd#(.D_DLY(1))    u2_lcd_driver(
69.      .clk              (clk_0          ),//40MHz
70.      .rst_n            (rst_n          ),
71.      .ack_data         (pic_data       ),
72.
```

```
73.       .req_x              (req_x            ),
74.       .req_y              (req_y            ),
75.       .req_en             (       ),
76.       .req_addr           (       ),
77.
78.       .hys                (lcd_hs             ),
79.       .vys                (lcd_vs             ),
80.       .lcd_de             (lcd_de           ),
81.       .lcd_rgb            (lcd_dat          ),
82.       .lcd_dclk           (lcd_clk        ),
83.
84.       .frame_start    (frame_start )
85.        );
86.
87.  /******************    功能模块    *******************************/
88.
89.  PicZoomInAnDout u3_PicZoomInAnDout(
90.       .clk                    (clk_0      ),
91.       .rst_n                  (rst_n      ),
92.       .mode                     (MODE_SELECT_en),
93.       .key_en                   (key_vld ),
94.       .cpu_ZoomIn               (SOFTWARE_CTRL_in),
95.       .cpu_ZoomOut              (SOFTWARE_CTRL_out),
96.
97.       .req_h                    (req_x      ),
98.       .req_v                    (req_y      ),
99.       .pic_data                 (pic_data           )
100.
101.       );
102.
103.
104./*********************    按键模块  ***************************/
105.
106.  mdyKey#(.DATA_W(24),.TIME_20MS(8_00_000)) u4_mdykey(
107.       .clk              (clk_0      ),
108.       .rst_n            (rst_n      ),
109.       .key_in           (key    ),
110.       .key_vld          (key_vld )
111.);
112./*********************    边沿检测模块    ***************************/
113.
114.mdyGetEdge    u5_GetEdge(
115.       .clk                    (clk_0      ),
```

```
116.      .rst_n                (rst_n     ),
117.      .cfg_init_value     (0     ),
118.      .din                  (frame_start     ),
119.      .dout_pos             (frame_pos),
120.      .dout_neg             (     ),
121.      .dout_pos_reg         (     ),
122.      .dout_neg_reg         (     )
123.      );
124.
125./*********************   1s 统计模块   ****************************/
126.
127.mdyStatic_1S       u6_Static_1S(
128.      .clk              (clk_0     ),
129.      .rst_n            (rst_n     ),
130.      .cfg_num_1s     (50_000_000     ),
131.      .cfg_add_1s     (1                 ),
132.      .din_vld          (frame_pos       ),
133.      .sta_1s           (DATA_FRAME_data     ),//32bit
134.      .sta_rt           (     )
135.);
136.
137./*********************   指令模块   ****************************/
138.
139.top_uart_cfg#(.BPS(4167)) u7_top_uart_cfg(
140.      .clk          (clk_0                 ),
141.      .rst_n        (rst_n                 ),
142.      .cfg_head (16'h55d5              ),
143.      .rx           (uart_rxd              ),
144.      .tx           (uart_txd              ),
145.      .din          (cfgCtrl_dout          ),//s2p
146.      .din_vld    (cfgCtrl_dout_vld    ),
147.      .din_rdy    (                      ),
148.      .dout         (uart_dout             ),//p2s
149.      .dout_vld (uart_dout_vld         )
150.
151.);
152.
153.mdyCfgCtrl u8_mdyCfgCtrl(
154.      `include "mdyCfg_inst.v"
155.      .clk              (clk_0                 ),
156.      .rst_n            (rst_n                 ),
157.      .din              (uart_dout             ),
158.      .din_vld          (uart_dout_vld         ),
```

```
159.      .dout            (cfgCtrl_dout          ),
160.      .dout_vld        (cfgCtrl_dout_vld      )
161.
162.);
163.
164./*************************************************/
165.endmodule
```

1.2 mdyPllAltera 模块接口说明

1.2.1 接口信号

信号名	I/O	位宽	定义
areset	I	1	PLL 复位信号，高电平有效
inclk0	I	1	PLL 输入时钟 50MHz
c0	O	1	PLL 输出时钟 40MHz

1.2.2 使用说明

本模块主要用于产生 LCD 驱动时序所需要的时钟，关于 mdyPllAltera 模块的使用详细介绍请看下方链接：

http://www.fpgabbs.cn/forum.php?mod=viewthread&tid=1172&fromuid=100105

1.3 mdyRom 模块设计

1.3.1 接口信号

信号名	I/O	位宽	定义
address	I	16	ROM 数据存放地址

明德扬FPGA全能设计法

clock	I	1	ROM 工作时钟 40MHz
q	O	8	ROM 输出数据

1.3.2 设计思路

本模块主要用于存储明德扬 LOGO 图标和大小字母“E”图像的数据，关于 mdyRom 模块的使用详细介绍请看下方链接：

http://www.fpgabbs.cn/thread-1192-1-1.html

1.4 mdyCfg 模块接口说明

1.4.1 接口信号

top_uart_cfg 模块的接口信号：

信号名	I/O	位宽	定义
clk	I	1	输入时钟信号，40MHz
rst_n	I	1	输入复位信号，低电平有效
cfg_head	I	16	输入指令头数据，指令头为 16'h55d5
rx	I	1	串口接收数据信号
tx	I	1	串口发送数据信号
din	I	64	输入 FPGA 指令数据
din_vld	I	1	输入 FPGA 指令数据有效指示信号
dout	O	64	输出 PC 指令数据
dout_vld	O	1	输出 PC 指令数据有效指示信号

明德扬FPGA至简设计法

mdyCfgCtrl 模块的接口信号：

信号名	I/O	位宽	定义
clk	I	1	输入时钟信号，40MHz
rst_n	I	1	输入复位信号，低电平有效
din	I	64	输出 PC 指令数据
din_vld	I	1	输出 PC 指令数据有效指示信号
dout	O	64	输入 FPGA 指令数据
dout_vld	O	1	输入 FPGA 指令数据有效指示信号
DATA_FRAME_data	I	32	输出帧率数据
MODE_SELECT_en	O	1	对图片的放大和缩小控制模式选择，复位值为 0，表示按键控制；1，表示上位机控制。
SOFTWARE_CTRL_in	O	1	放大控制
SOFTWARE_CTRL_out	O	1	缩小控制

1.4.2 使用说明

本模块主要用于对传输的数据进行串并转换和解析传输的指令，关于 mdyCfg 模块的使用详细介绍请看下方链接：

http://www.fpgabbs.cn/forum.php?mod=viewthread&tid=1173&fromuid=100105

关于具体的指令信息可以查看工程里的 XML 表格 regTable_mdyLcdPicZoomInAndOut.xml：

上位机指令全长 64bit，其中，

指令段	位宽	定义
[63:48]	16	指令头，固定为 16'h55d5
[47]	1	读/写控制位，0 表示写寄存器，1 表示读寄存器

明德扬FPGA全简设计法

[46:32]	15	寄存器地址
[31:0]	32	数据

name	名字	范围	说明	属性	复位值	保持
reg_name	MODE_SELECT	ADDR	16'h0000			
field_name	en	0	用于指示是使用按键控制还是上位机控制图像 0：使用按键控制 1：使用上位机控制	W	0	Y
reg_name	SOFTWARE_CTRL	ADDR	16'h0001			
field_name	in	1	放大控制	W	0	N
field_name	out	0	缩小控制	W	0	N
reg_name	DATA_FRAME	ADDR	16'h0002			
field_name	data	31:0	帧率数值	R	0	Y

1.5 mdyKey 模块接口说明

1.5.1 接口信号

下面为使用独立按键时的接口信号：

信号名	I/O	位宽	定义
clk	I	1	输入时钟信号，40MHz
rst_n	I	1	输入复位信号，低电平有效
key_in	I	4	按键输入信号
key_vld	O	4	按键有效指示信号，4'b0001 表示按键 S1 按下，控制放大，4'b0010 表示按键 S2 按下，控制缩小。

1.5.2 使用说明

本模块主要检测按下的按键并输出对应按键的有效指示信号，关于 mdyKey 模块的使用详细介绍请看下方链接：

http://www.fpgabbs.cn/forum.php?mod=viewthread&tid=1181&fromuid=100105

1.6 PicZoomInAndOut 模块接口说明

1.6.1 接口信号

信号名	I/O	位宽	定义
clk	I	1	输入时钟信号，40MHz
rst_n	I	1	输入复位信号，低电平有效
mode	I	1	对图片的放大和缩小控制模式选择，复位值为 0，表示按键控制；1，表示上位机控制。
key_en	I	4	按键有效指示信号，4'b0001 表示按键 S1 按下，控制放大，4'b0010 表示按键 S2 按下，控制缩小。
cpu_ZoomIn	I	1	上位机控制住图片放大指示信号
cpu_ZoomOut	I	1	上位机控制住图片缩小指示信号
req_h	I	11	LCD 显示屏有效显示区域每一行的某一像素指示信号
req_v	I	10	LCD 显示屏有效显示区域某一行指示信号
pic_data	O	24	输出显示图像数据，格式为 RGB888

1.6.2 参考代码

```
1.  module PicZoomInAnDout(
2.          clk                 ,
3.          rst_n               ,
4.          mode                ,
5.          key_en              ,
6.          cpu_ZoomIn          ,
7.          cpu_ZoomOut         ,
8.
9.          req_h               ,//h_cnt - THB
10.         req_v               ,//v_cnt - TVB
```

```
11.     pic_data
12.     );
13.
14.     //LCD 显示屏居中
15.     parameter       HDE_CENTRE   = 400       ;//800/2
16.     parameter       VDE_CENTRE   = 240       ;//480/2
17.
18.
19.     parameter       DATA_W       =           8;
20.     parameter       KEY_W        =           4;
21.
22.     input                        clk          ;
23.     input                        rst_n        ;
24.     input                        mode         ;
25.     input                        cpu_ZoomIn   ;
26.     input                        cpu_ZoomOut  ;
27.     input   [KEY_W-1 :0]         key_en       ;
28.
29.     input   [10:0]               req_h        ;
30.     input   [ 9:0]               req_v        ;
31.     output  [23:0]               pic_data     ;
32.
33.     reg     [23:0]               pic_data     ;
34.
35.     reg     [15:0]               logo_rom_addr   ;
36.     reg                          logo_rom_area   ;
37.     wire    [7:0]                logo_rom_data   ;
38.     reg     [17:0]               e_rom_addr      ;
39.     reg                          e_rom_area      ;
40.     reg      [2:0]               e_rom_addr_low  ;
41.     reg                          e_dataout       ;
42.     wire    [7:0]                e_rom_data      ;
43.
44.     reg     [ 2:0]               size            ;
45.     reg     [ 2:0]               size_ff0        ;
46.
47.     wire    [9:0]                len_size        ;
48.     wire    [9:0]                wid_size        ;
49.     wire    [9:0]                e_x0            ;
50.     wire    [9:0]                e_x1            ;
51.     wire    [9:0]                e_y0            ;
52.     wire    [9:0]                e_y1            ;
53.     wire    [9:0]                x               ;
```

```
54.       wire      [9:0]                    y                    ;
55.
56.  /*****************************************************/
57.
58.  always   @(posedge clk or negedge rst_n)begin
59.       if(rst_n==1'b0)begin
60.            size <= 0;
61.       end
62.       else if(mode==1)begin
63.            if(cpu_ZoomOut)begin
64.                 if(size!=3)
65.                      size <= size + 1;
66.            end
67.            else if(cpu_ZoomIn)begin
68.                 if(size!=0)
69.                      size <= size - 1;
70.            end
71.       end
72.       else if(mode==0)begin
73.            if(key_en==4'b0010)begin
74.                 if(size!=3)
75.                      size <= size + 1;
76.            end
77.            else if(key_en==4'b0001)begin
78.                 if(size!=0)
79.                      size <= size - 1;
80.            end
81.       end
82.  end
83.
84.  always   @(posedge clk or negedge rst_n)begin
85.       if(rst_n==1'b0)begin
86.            size_ff0 <= 0;
87.       end
88.       else if(req_h==(HDE_CENTRE-200) && req_v==(VDE_CENTRE-150)) begin
89.            size_ff0 <= size;
90.       end
91.  end
92.
93.  assign len_size = 400 >> size_ff0;//缩小多少倍
94.  assign wid_size = 300 >> size_ff0;
95.
96.
```

```
97.  assign     e_x0 = (HDE_CENTRE-len_size[9:1]);
98.  assign     e_x1 = (HDE_CENTRE+len_size[9:1]);
99.  assign     e_y0 = (VDE_CENTRE-wid_size[9:1]);
100.assign     e_y1 = (VDE_CENTRE+wid_size[9:1]);
101.
102.assign x = (req_h-e_x0)<<size;
103.assign y = (req_v-e_y0)<<size;
104.
105.
106.
107.always   @(*)begin
108.       e_rom_area = req_h >=(e_x0+5) && req_h < e_x1 && req_v >= e_y0 && req_v < (e_y1+5);
109.end
110.
111.always   @(*)begin
112.       e_rom_addr = x + 400*y;
113.end
114.
115.
116./*******************************************************/
117.
118.//120*55
119.always   @(*)begin
120.       logo_rom_area = req_h >=0 && req_h < 119 && req_v >= 0 && req_v < 54;
121.end
122.
123.always   @(*)begin
124.       logo_rom_addr = req_h + 120*req_v;
125.end
126.
127./*******************************************************/
128.
129.always   @(posedge clk or negedge rst_n)begin
130.       if(rst_n==1'b0)begin
131.            e_rom_addr_low <= 0;
132.       end
133.       else begin
134.            e_rom_addr_low <= e_rom_addr[2:0];
135.       end
136.end
137.
138.always   @(*)begin
139.       if(e_rom_area)
```

```
140.            e_dataout = ~e_rom_data[7-e_rom_addr_low];
141.      else
142.            e_dataout = 1;
143.end
144.
145.always   @(posedge clk or negedge rst_n)begin
146.      if(rst_n==1'b0)begin
147.            pic_data <= 0;
148.      end
149.      else if(e_rom_area)begin
150.            pic_data <= {24{e_dataout}};
151.      end
152.      else if(logo_rom_area)begin
153.            pic_data <=
    {logo_rom_data[7:5],5'b11111,logo_rom_data[4:2],5'b11111,logo_rom_data[1:0],6'b111111} ;
154.      end
155.end
156./*****************************************************/
157.mdyRom#(.MIF("../src/data/logo.mif"),.DEP(8192),.D_W(8))   u_fpga_rom(
158.      .address    (logo_rom_addr     ),
159.      .clock      (clk     ),
160.      .q          (logo_rom_data     )
161.);
162.
163.mdyRom#(.MIF("../src/data/e.mif"),.DEP(16384),.D_W(8))   u_e_rom(
164.      .address    (e_rom_addr[16:3]     ),
165.      .clock      (clk     ),
166.      .q          (e_rom_data     )
167.);
168.
169.      endmodule
```

1.7 mdyLcd 驱动模块设计

1.7.1 接口信号

信号名	I/O	位宽	定义
clk	I	1	模块工作时钟 40MHz

rst_n	I	1	系统复位信号，低电平有效
ack_data	I	24	输入的图像有效显示数据
req_x	O	11	输出的 LCD 显示屏有效显示区域每一行某一像素点指示信号
req_y	O	10	输出的 LCD 显示屏有效显示区域某一行指示信号
frame_start	O	1	每帧图像开始指示信号
hys	O	1	LCD 行时序信号
vys	O	1	LCD 场时序信号
lcd_de	O	1	LCD 数据输出使能信号
lcd_rgb	O	24	LCD RGB 信号，RGB 格式为使用 24 位来表示一个像素，RGB 分量都用 8 位表示，取值范围为 0-255。
lcd_dclk	O	1	LCD 数据采样时钟

1.7.1 使用说明

本模块主要用于产生LCD显示屏的驱动时序，关于mdyLcd模块的使用详细介绍请看下方链接：
http://www.fpgabbs.cn/forum.php?mod=viewthread&tid=1177&fromuid=100105

1.8 mdyGetEdge 模块接口说明

1.8.1 接口信号

信号名	I/O	位宽	定义
clk	I	1	输入时钟信号，40MHz

明德扬FPGA实战设计法

rst_n	I	1	输入复位信号，低电平有效
cfg_init_value	I	1	检测边沿的初始值，初始值为 0
din	I	1	输入的检测信号，图像帧开始指示信号
dout_pos	O	1	输出的上升沿检测有效指示信号

1.8.2 使用说明

本模块主要用于检测特定信号的边沿，关于 mdyGetEdge 模块的使用详细介绍请看下方链接：
http://www.fpgabbs.cn/forum.php?mod=viewthread&tid=1190&fromuid=100105

1.9 mdyStatic_1S 模块接口说明

1.9.1 接口信号

信号名	I/O	位宽	定义
clk	I	1	输入时钟信号，40MHz
rst_n	I	1	输入复位信号，低电平有效
cfg_num_1s	I	32	多少时间统计一次对应的时钟个数
cfg_add_1s	I	8	统计一次增加多少数值
din_vld	I	1	输入的上升沿检测有效指示信号
sta_1s	O	32	输出的规定时间内统计的数据

1.9.2 使用说明

本模块主要用于统计数据，关于 mdyStatic_1S 模块的使用详细介绍请看下方链接：

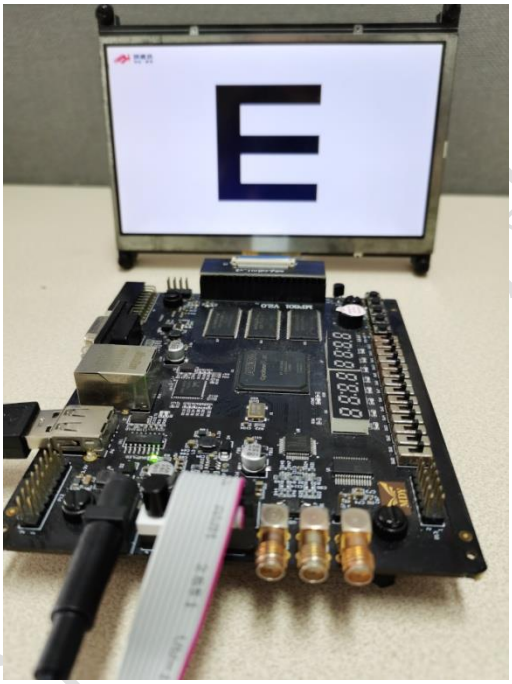
明德扬FPGA至简设计法

http://www.fpgabbs.cn/forum.php?mod=viewthread&tid=1188&fromuid=100105

1.10 效果和总结

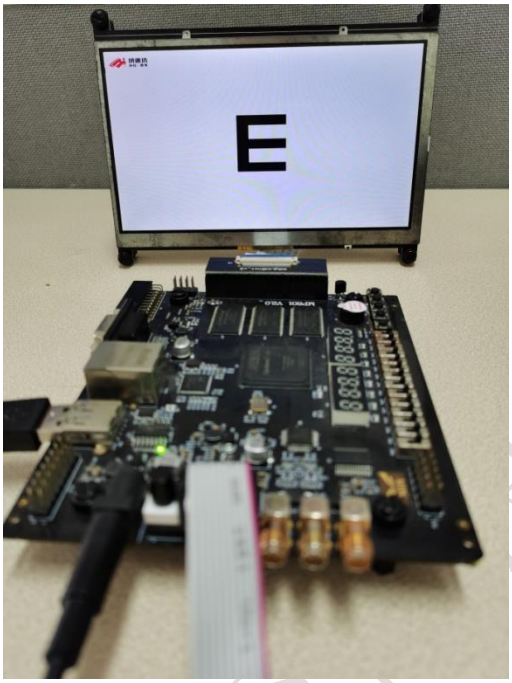
以下为工程上板后的现象效果图：

- **mp801 开发板——缩小 0 倍**

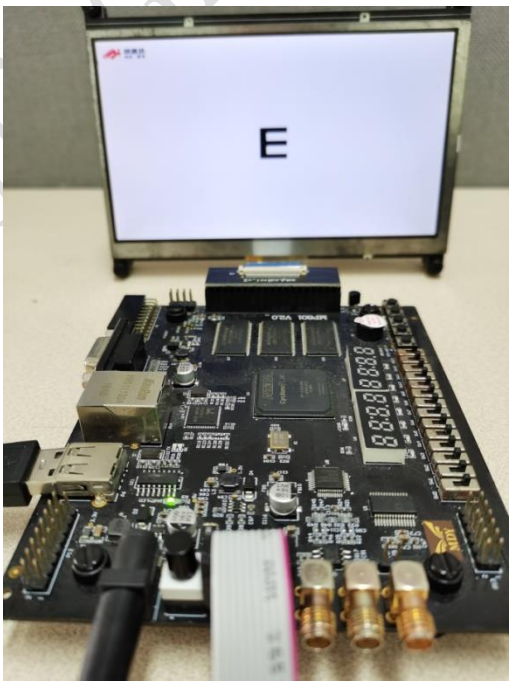


明德扬FPGA设计法

- mp801 开发板——缩小 1 倍



- mp801 开发板——缩小 2 倍



明德扬FPGA至简设计法

- **mp801 开发板——缩小 3 倍**

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：

《至简设计系列_LCD 入门案例_边框显示》
《至简设计系列_BCD 译码实现》
《至简设计系列_简易计算器》
《至简设计系列_基于 FPGA 的超声波测距系统设计》
《至简设计系列_串口回环工程》
《至简设计系列_LCD 入门案例-动态矩形》
《至简设计系列_闹钟》
《至简设计系列_7 段数码管显示》
《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》

1.11 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。

明德扬FPGA至简设计法