基于 FPGA 的数字识别的实现二

作者：OpenS_Lee

1 背景知识

1.1 基于 FPGA 的数字识别的方法

通常，针对印刷体数字识别使用的算法有：基于模版匹配的识别方法、基于 BP 神经网络的识别方法、基于数字特征的识别方法等。下文将对这几种算法进行讨论以及比较。

1>模版匹配法

模版匹配法是一种被较早应用的数字识别算法，该算法的关键是对所要识别的所有数字进行模版构建，之后将图像中的数字与所有的数字模版一一进行比较，计算出图像中数字与每个模版的相似度，根据所计算出的相似度结果进行识别。其中相似度最高的模版即为我们所要识别的结果。模版匹配法的对数字的大小、结构形状的规范化程度要求很高，数字的规范化程度对识别的准确率有着直接的影响。该算法原理较为简单，但计算复杂度过大，同时不利于 FPGA 的实现。

2>神经网络识别算法

神经网络识别的方法是模仿动物神经网络的特征，对信息进行分布式并行处理的一种算法。神经网络识别算法具有一定的抗干扰能力，但为了保证识别的准确率，该算法需要负责并且大量的计算，来对神经网络进行训练，而过于复杂的计算不利于 FPGA 对该算法的实现。

3>数字特征识别算法

基于数字特征的识别算法其核心是通过对数字的形状以及结构等几何特征进行分析与统计，通过对数字特征的识别从而达到对图像中数字的识别。

1.2 基于数字特征算法实现数字识别

我们采用基于数字特征的算法进行数字的识别，通过图像采集模块采集到图像，进行灰度化，二值化，然后进行数字特征的提取和统计来完成对数字的识别，最终显示到数码管上，完成图像信息到数字信息的转化。

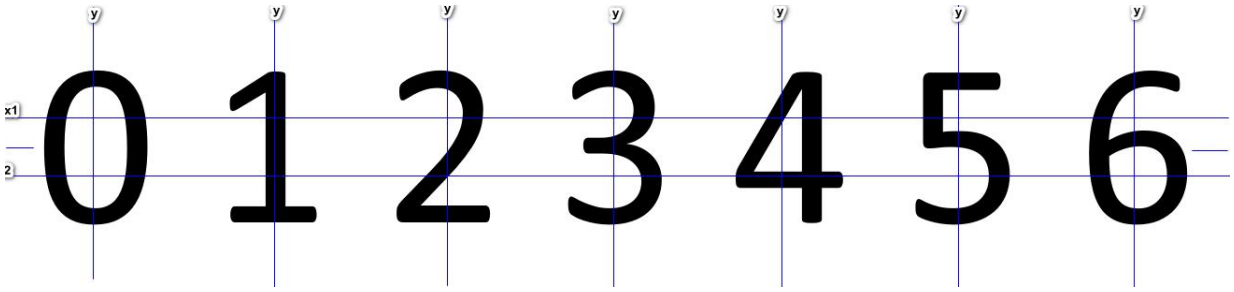

图 1 0-6 数字特征标线

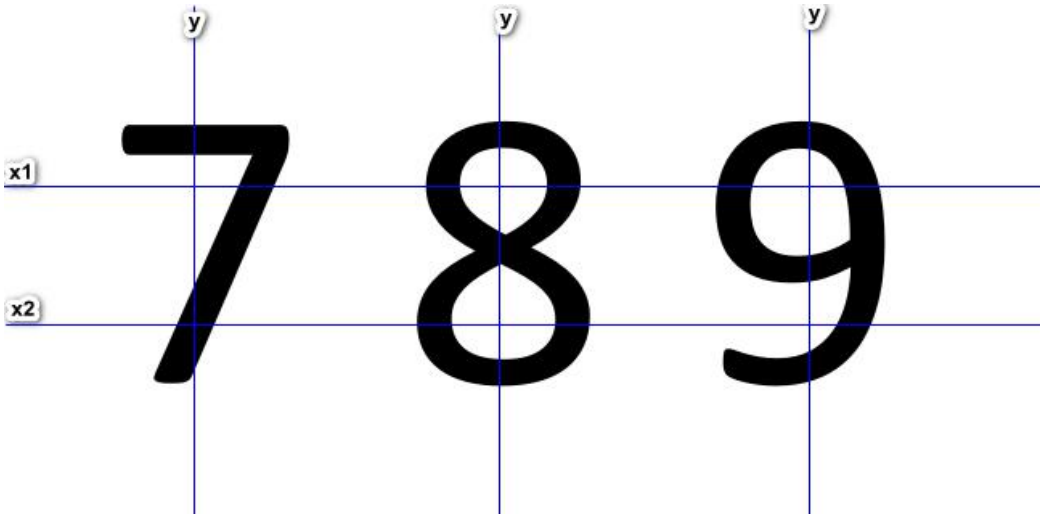

图 2　7-8 数字特征标线

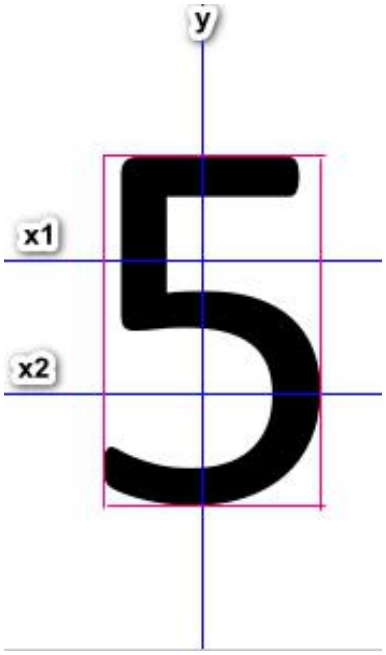

图 3　5 数字特征

数字特征信息的提取基于打印体，如上图 1,图 2,图 3 所示，以图 3 数字 5 举例，红框是数字 5 的水平和竖直的上下左右边界。X1 在竖直方向的 2/5 处的水平线，x2 在竖直方向的 2/3 处的水平线，y 在水平方的 1/2 处的水直线。我们以此特征来统计 x1,x2,y 与数字 5 的交叉点。

以交叉统计法来区分 0-9 数字的特征如下表 1：

表 1　0-9 数字特征统计表

数字	与 y 交叉点个数	与 x1 交叉点个数	与 x2 交叉点个数	分类
0	2	2	2	A
1	1	1	1	B
2	3	1	1	C
3	3	1	1	C
4	2	2	1	D
5	3	1	1	C
6	3	1	2	E
7	2	1	1	F
8	3	2	2	G
9	3	2	1	H

由于 2,3,5 的数字特征统计表一样，无法区分所以我们继续增加数字特征以区分 2,3,5。

如表 2：

表 2　2,3,5 数字特征统计

数字	与 x1 的交叉点位置（左，右）	与 x2 的交叉点位置（左，右）	分类
2	右	左	I(C)
3	右	右	J(C)
5	左	右	K(C)

这样通过数字统计完全区分开数字 0-9。然后利用 FPGA 系统搭建实时数字识别系统。

如图 1 所示，交通摄像头对公路上移动的汽车进行实时的定位，随着小汽车的移动，

红色框也跟随小汽车移动，实时将小汽车框起来。

1.3 目标跟踪方法

基于实时物体移动的静态图像背景中移动目标检测是计算机视觉领域的研究热点，在安防、监控、智能交通、机器智慧、以及军事领域等社会生活和军事防御等诸多领域都有较大的实用价值。移动目标检测的实质是从实时图像序列中将图像的变化区域从整体图像中分割提取出来。由于图像的后期处理，比如移动目标的分类、跟踪、测距、判断大小以及行为动作分析等，主要考虑的是移动目标区域的像素信息，所以对移动目标的准确检测和有效分割是整个检测跟踪系统的重要基础。

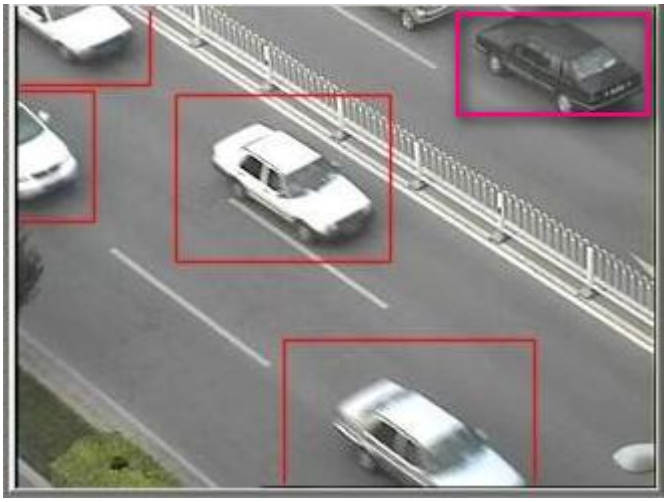

图 4 公路上的运动汽车

目前逐渐形成三种运动目标的检测算法：

1）帧间差分法 是采用视频序列中的相邻两帧图像做差的方法，来检测视频序列中的移动目标。但是受运动目标和背景变化的影响，检测过程中有可能出现伪目标或者目标中出现“空洞”，在目标运动不是太快时可以有效的检测到目标。

2）背景减除法首先在没有目标的场景中获取背景图像，然后利用实时视频序列和背景图像做差，来实现地移动目标的检测。如何获得背景是背景减除法的关键。

3）光流法是通过给图像中每个像素点赋予一个速度矢量的方法建立光流场，利用光流场中矢量运动的连续性来检测移动目标。该方法的计算量通常很大，难以实现实时性的检测。

本节将基于 FPGA 的目标跟踪以及统计学的特征统计来实现对数字的位置实时定位以及数字识别，不在局限于数字在屏幕中的位置，也不局限数字的大小。

2 基于 FPGA 的数字识别的实现

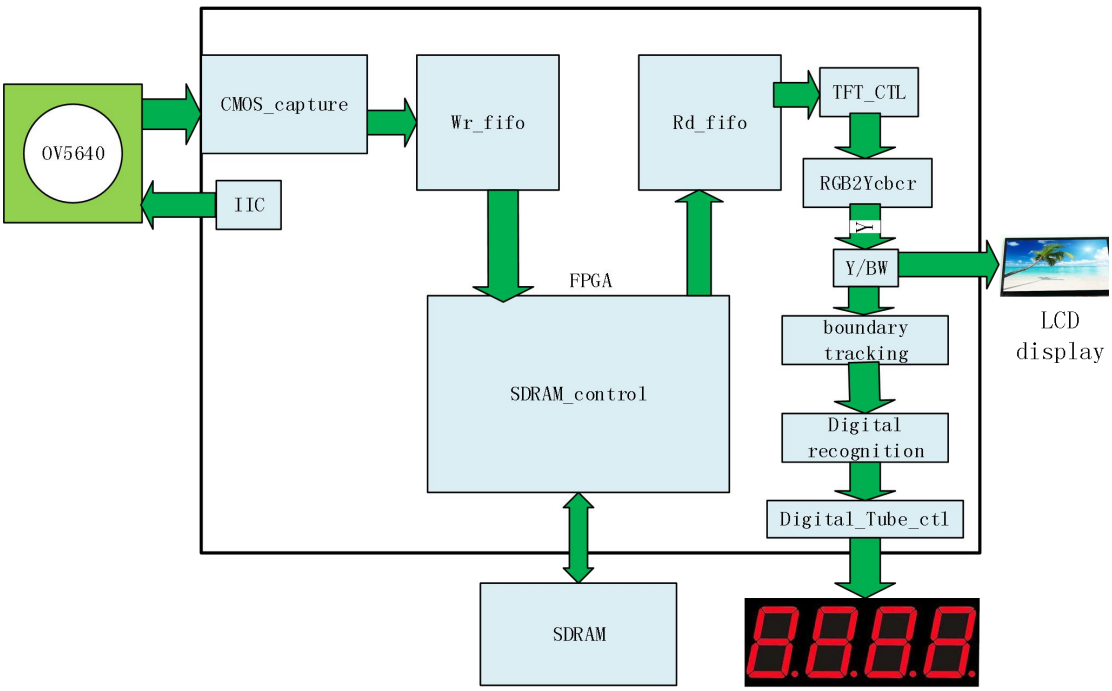

图 5 基于 ov5640 的 FPGA 实时数字识别系统

如图 5 所示，我们图像采集使用 ov5640 cmos 500W 像素摄像头，将采集到的彩色 RGB 图像首先存入 SDRAM 中，然后由 TFT 显示控制端读出图像数据，读出 RGB 图像数据后，我们首先进行 RGB 转 Ycbcr 算法操作，然后对灰度图像进行阈值分割，形成二值图像，对二值数字图像进行边界追踪的基础上进行数字识别，最终将边界显示在 TFT5 寸屏幕上，将识别的数字信息显示在数码管上。

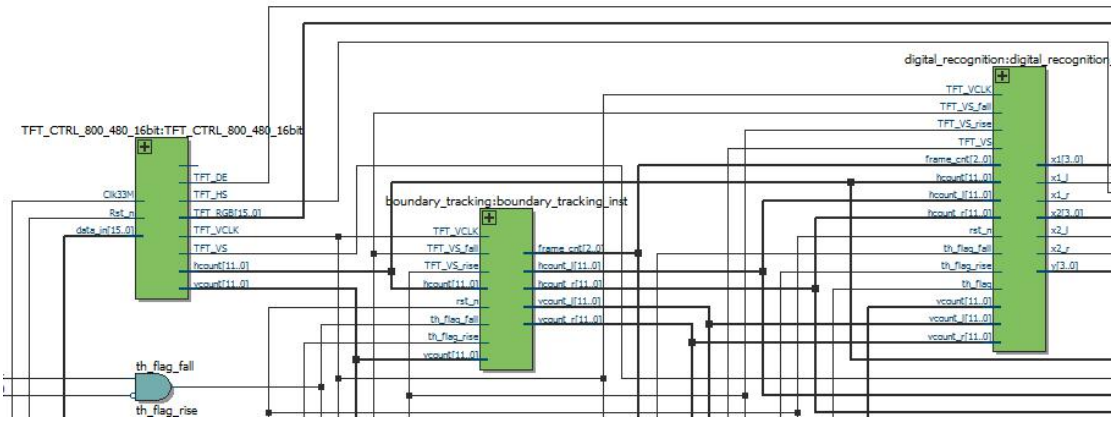

图 6 边界追踪数字识别的三大主要核心模块

如图 6 所示，以 TFT 屏的显示时序为基准，首先进行边界追踪，识别数字边界后，我们在边界的基础上进行统计特征的数字识别。

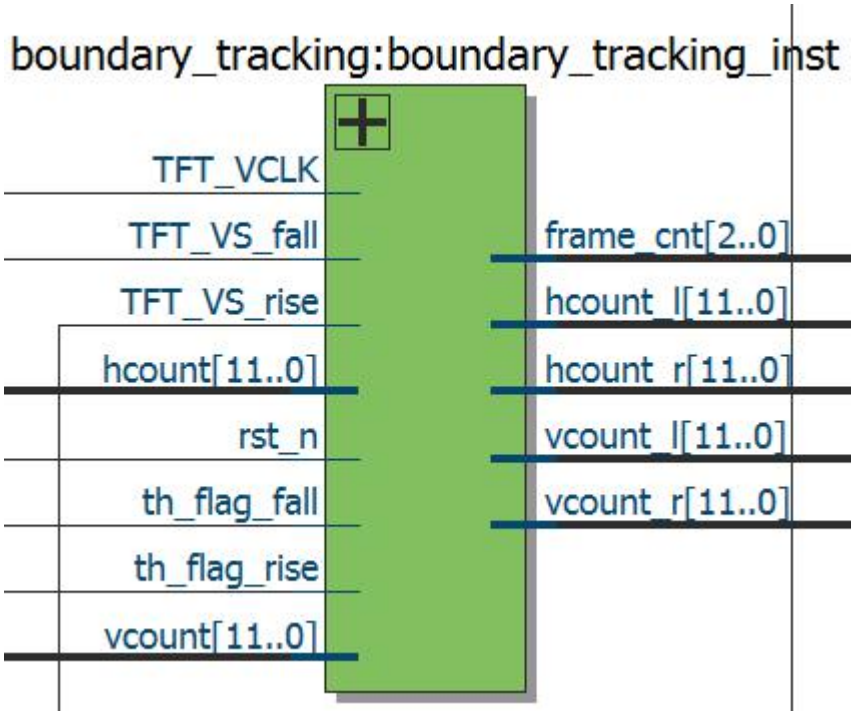

图 7 边界追踪模块

如图 7 所示，hcount 为列计数器，vcount 为行计数器，TFT_VS_fall 和 TFT_VS_rise 分别是帧下降沿标志和帧上升沿标志，frame_cnt 为帧计数器，hcount_l 和 hcount_r 分别是识别后数字的左右边界，vcount_l 和 vcount_r 分别是数字的上下边界。Th_flag_fall 和 th_flag_rise 分别是灰度图像阈值后的下降沿和上升沿标志。

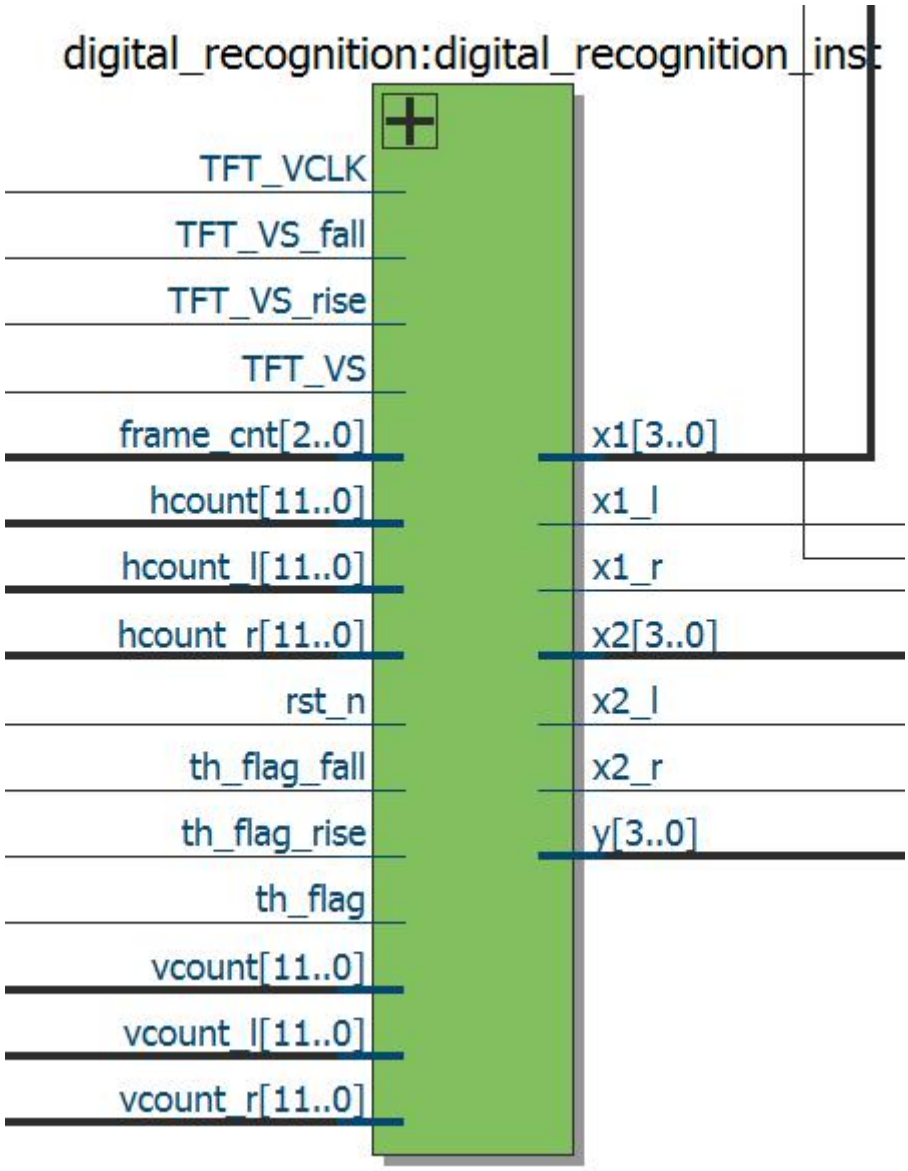

图 8 数字识别模块

如图 8 所示基本的边界信息均来自数字边界识别模块，数字识别模块主要的到数字统计学的两横一竖（x1,x2,y）与数字的交点信息，以及其他补充信息。

主要代码解释：

Top 层：

```
357   //-------------------------------------
358   //pipeline
359   //-------------------------------------
360   always @(posedge TFT_VCLK ) begin
361     TFT_VS_r0 <= TFT_VS;
362     TFT_VS_r1 <= TFT_VS_r0;
363   end
364
365   always @(posedge TFT_VCLK ) begin
366     th_flag_r0 <= th_flag;
367     th_flag_r1 <= th_flag_r0;
368   end
369   //-------------------------------------------------------------------
370   // rise or fall flag
371   //-------------------------------------------------------------------
372   assign th_flag_rise = (th_flag_r0 && (!th_flag_r1)) ? 1'b1:1'b0;  //
373   assign th_flag_fall = ((!th_flag_r0) && th_flag_r1) ? 1'b1:1'b0;  //
374   assign TFT_VS_rise  = (TFT_VS_r0 && (!TFT_VS_r1)) ? 1'b1 :1'b0;   //VS rise edge
375   assign TFT_VS_fall  = ((!TFT_VS_r0) && TFT_VS_r1) ? 1'b1 :1'b0;   //VS fall edge
376
```

```
//-------------------------------------------------------
// lcd display
//-------------------------------------------------------
always @(posedge TFT_VCLK or negedge rst_n) begin
  if(!rst_n)
    TFT_RGB <= 16'h0000;
  else if (vcount > vcount_l && vcount < vcount_r && ( hcount == hcount_l
            || hcount == hcount_r || hcount == (hcount_l -1) || hcount == (hcount_r+1))) //形成左右竖直边界线
    TFT_RGB <= 16'hf0f0;
  else if (hcount > hcount_l && hcount < hcount_r && (vcount== vcount_l
            || vcount== vcount_r || vcount==( vcount_l - 1) || vcount== (vcount_r + 1))) //形成上下水平边界线
    TFT_RGB <= 16'hf0ff;
  else if(o_y_8b >64)
    TFT_RGB <= 16'hffff; //white
  else
    TFT_RGB <= 16'h0000;  //dark
end
```

```
394   //-------------------------------------------------------------------
395   // threshold value
396   //-------------------------------------------------------------------
397   always @(posedge TFT_VCLK or negedge rst_n) begin
398     if(!rst_n)
399       th_flag <= 1'b0;
400     else if(o_y_8b >64) //阈值
401       th_flag <= 1'b1;
402     else
403       th_flag <= 1'b0;
404   end
```

```
405   //------------------------------------------------------------------
406   // tube display
407   //------------------------------------------------------------------
408
409   always @(posedge TFT_VCLK or negedge rst_n) begin
410     if(!rst_n)
411       disp_data <= 32'h0;
412     else if((frame_cnt == 3'd2) && TFT_VS_rise)
413       case({x1_l,x1_r,x2_l,x2_r,y,x1,x2})              //特征统计数据对应列表
414         16'b1111_0010_0010_0010: disp_data <= {28'b0,4'h0}; //0
415         16'b1010_0001_0001_0001: disp_data <= {28'b0,4'h1}; //1
416         16'b0110_0011_0001_0001: disp_data <= {28'b0,4'h2}; //2
417         16'b0101_0011_0001_0001: disp_data <= {28'b0,4'h3}; //3
418         16'b1101_0010_0010_0001: disp_data <= {28'b0,4'h4}; //4
419         16'b1001_0011_0001_0001: disp_data <= {28'b0,4'h5}; //5
420         16'b1011_0011_0001_0010: disp_data <= {28'b0,4'h6}; //6
421         16'b0110_0010_0001_0001: disp_data <= {28'b0,4'h7}; //7
422         16'b1111_0011_0010_0010: disp_data <= {28'b0,4'h8}; //8
423         16'b1101_0011_0010_0001: disp_data <= {28'b0,4'h9}; //9
424         default: disp_data <= 32'b0;
425       endcase
426     else
427       disp_data <= disp_data;
428   end
```

边界追踪模块：

```
1   /*
2   Module name:  boundary_tracking.v
3   Description:  boundary tracking
4
5   Data:         2018/04/17
6   Engineer:     lipu
7   e-mail:       137194782@qq.com
8   微信公众号:     FPGA开源工作室
9   */
10  `timescale 1ns/1ps
11  module boundary_tracking(
12          input TFT_VCLK, //lcd clock
13          input rst_n,
14          input th_flag_rise,
15          input th_flag_fall,
16          input TFT_VS_rise,
17          input TFT_VS_fall,
18          output reg [2:0]   frame_cnt,//frame counter
19          input  [11:0]      hcount,
20          input  [11:0]      vcount,
21          output reg [11:0] hcount_l,
22          output reg [11:0] hcount_r,
23          output reg [11:0] vcount_l,
24          output reg [11:0] vcount_r
25          );
26
```

```
//------------------------------------------------
//得到数字的左右边界
//------------------------------------------------
always @(posedge TFT_VCLK or negedge rst_n) begin
  if(!rst_n) begin
    hcount_l <= 12'd0;
    hcount_r <= 12'd0;
  end
  else if((cstate == F3) && TFT_VS_rise) begin
    hcount_l <= x_data_min-5; //左边界调整
    hcount_r <= x_data_max+5; //右边界调整
  end
  else
   ;
end
```

```
//------------------------------------------------
// 得到数字的上下边界
//------------------------------------------------
always @(posedge TFT_VCLK or negedge rst_n) begin
  if(!rst_n) begin
    vcount_l <= 12'd0;
    vcount_r <= 12'd0;
  end
  else if((cstate == F3) && TFT_VS_rise) begin
    vcount_l <= y_data_min-5;  //上边界调整
    vcount_r <= y_data_max+5;  //下边界调整
  end
  else
   ;
end
```

数字识别部分：

```
/*
Module name:  digital_recognition.v
Description:  digital recognition

Data:         2018/04/17
Engineer:     lipu
e-mail:       137194782@qq.com
微信公众号:     FPGA开源工作室
*/
`timescale 1ns/1ps

module digital_recognition(
        input               TFT_VCLK,
        input               TFT_VS,
        input               rst_n,
        input               th_flag,  //threshold value
        input     [11:0]    hcount,
        input     [11:0]    vcount,
        input     [11:0]    hcount_l,//左边界
        input     [11:0]    hcount_r,//右边界
        input     [11:0]    vcount_l,//上边界
        input     [11:0]    vcount_r,//下边界
        input               th_flag_rise,
        input               th_flag_fall,
        input               TFT_VS_rise,
        input               TFT_VS_fall,
        input     [2:0]     frame_cnt,
        output reg          x1_l, //x1左
        output reg          x1_r, //x1右
        output reg          x2_l,
        output reg          x2_r,
        output reg [3:0]    y,
        output reg [3:0]    x1,
        output reg [3:0]    x2
        );

```

```
75    //-------------------------------------------------
76    // 1/2 x            2/5 y             2/3 y
77    //-------------------------------------------------
78  always @(posedge TFT_VCLK or negedge rst_n) begin
79    if(!rst_n) begin
80      h_2 <= 12'd0;
81      v_5 <= 12'd0;
82      v_3 <= 12'd0;
83      h_2_r <= 18'd0;
84      v_5_r <= 23'd0;
85      v_3_r <= 23'd0;
86      hcount_l_r <= 18'b0;
87      hcount_r_r <= 18'b0;
88      vcount_l_r <= 18'b0;
89      vcount_r_r <= 18'b0;
90    end
91    else if(frame_cnt == 3'd3) begin
92      if(TFT_VS_rise) begin
93        hcount_l_r <= {hcount_l,6'b0};//位扩展
94        hcount_r_r <= {hcount_r,6'b0};
95        vcount_l_r <= {vcount_l,6'b0};
96        vcount_r_r <= {vcount_r,6'b0};
97      end
98      if(TFT_VS_rise_r0) begin
99        h_2_r <=  (hcount_r_r + hcount_l_r)>>1;       // y 线
100       v_5_r <=  vcount_r_r*P_2_5 + vcount_l_r*P_3_5;//x1 线
101       v_3_r <=  vcount_r_r*P_2_3 + vcount_l_r*P_1_3;//x2 线
102     end
103     if(TFT_VS_rise_r1) begin
104         h_2 <= h_2_r[17:6]; //bit位还原
105         v_5 <= v_5_r[23:12];
106         v_3 <= v_3_r[23:12];
107     end
108   end
109   else
110     ;
111 end
```

结果展示：

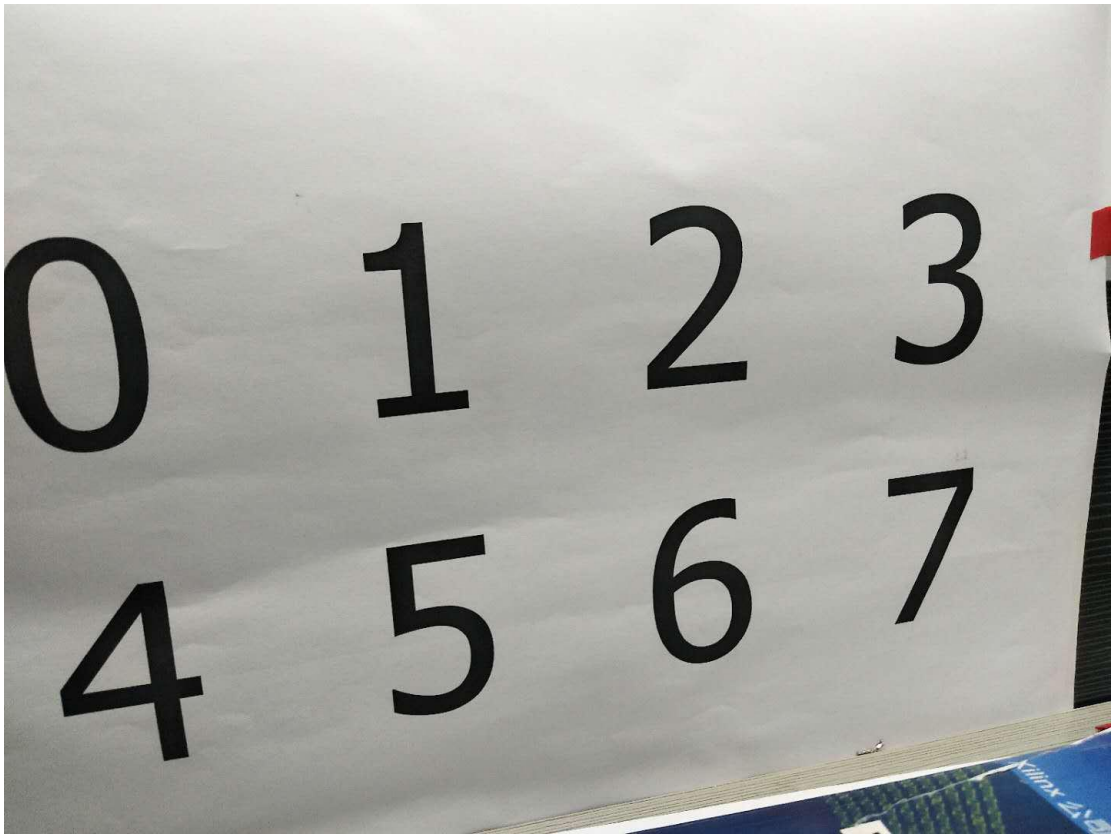

图 9 实验原图

图 10 边界跟踪数字识别 6

图 11 边界跟踪数字识别 7

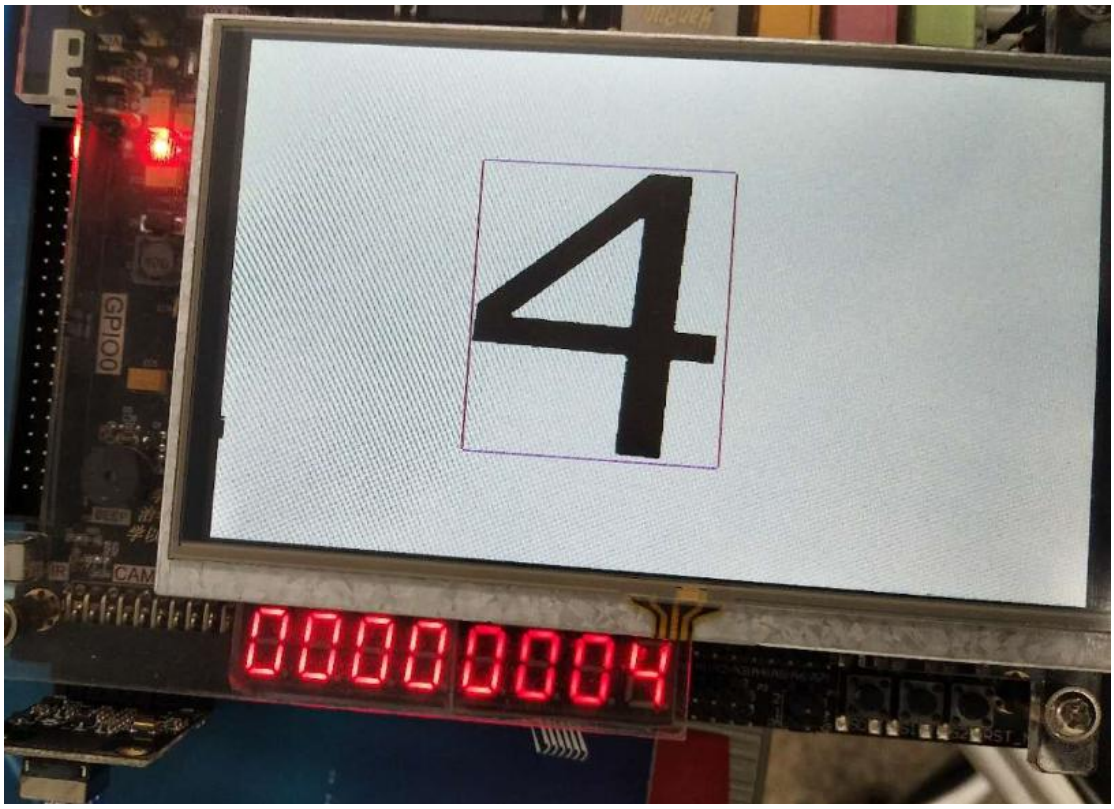

图 12 边界跟踪数字识别 4

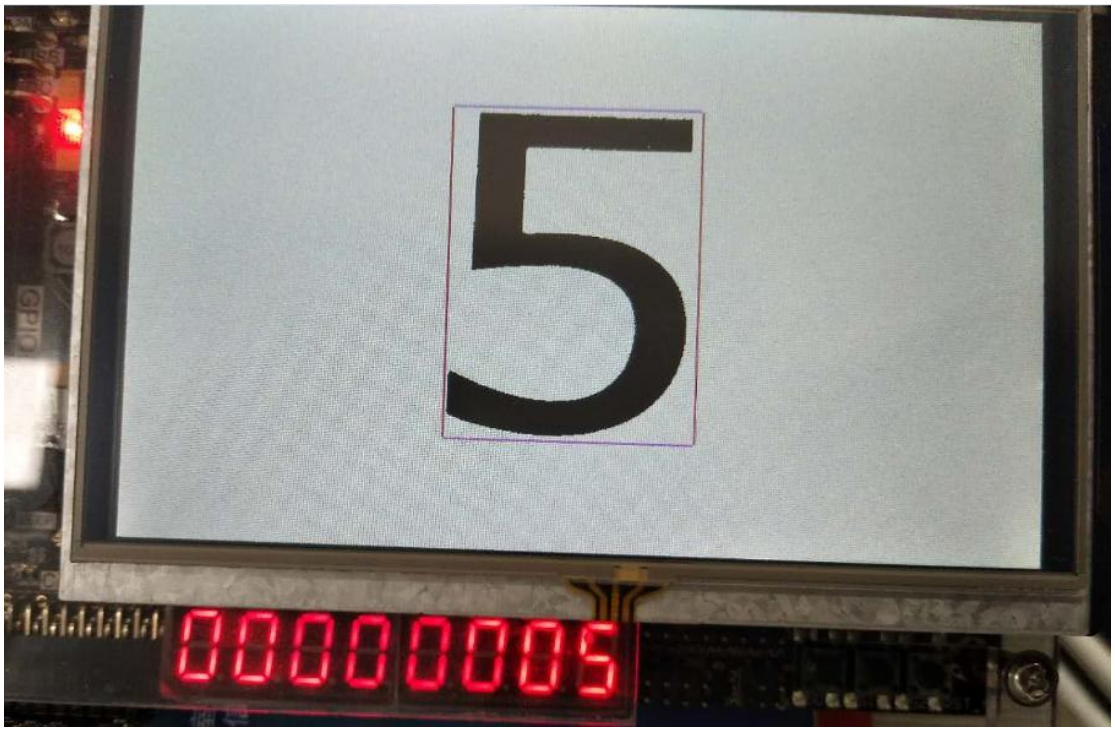

图 13 边界跟踪数字识别 5

展望：

基于机器视觉的识别是走向人工智能的必然之路，字符的识别就是这条路的敲门砖。本次实验的结果完成了无论数字大小，数字在屏幕中的位置均可正确识别。基于此，可以开发人脸位置识别，人脸模板匹配识别，车牌识别等现如今比较火的机器视觉，人工智能等。

欢迎大家关注我的微信公众号：FPGA 开源工作室