

第四届

全国大学生集成电路创新创业大赛

CICIEC

项目设计报告

参赛题目： RISC-V挑战杯 子题目2

队伍编号： AGD113923

团队名称： UGV

目录

[简介 1](#_Toc6307)

[基础功能 2](#_Toc9534)

[GPIO控制LED闪烁 2](#_Toc15709)

[逻辑分析仪显示“RISC-V”字样 2](#_Toc14171)

[LCD显示屏显示图片 4](#_Toc19901)

[进阶功能 4](#_Toc23350)

[操作系统 4](#_Toc26595)

[模型训练 5](#_Toc28098)

[人脸识别程序 5](#_Toc14054)

[身份证识别程序 7](#_Toc28645)

[程序优化 8](#_Toc22933)

[命令入口 9](#_Toc10414)

[参考文献 10](#_Toc30807)

[附录 12](#_Toc28208)

[命令清单 12](#_Toc19219)

[演示视频 12](#_Toc29270)

[程序代码 13](#_Toc5949)

1. 简介

本次比赛我们小组参加的是RISC-V杯的子赛题2。我们采用Nexys Video开发板，在开发板内搭载了一个开源的RISC-V项目，该项目使用Rocket Chip来实现RISC-V SoC，并嵌入Linux系统。此RISC-V SoC包括DDR、UART、SD和以太网控制器。DDR和UART由Vivado提供，SD和Ethernet则来自其他开源项目。在此基础上，我们又加入了自己设计的LED-按钮控制模块，实现了以不同形式闪烁板上的LED灯。我们还实现了输出8路模拟信号在逻辑分析仪上画出RISC-V字样的功能。此外，我们还开发了LCD屏幕显示图片的功能，该显示屏由一块荔枝派控制，开发板可以通过串口传输图片，并在屏幕上显示。

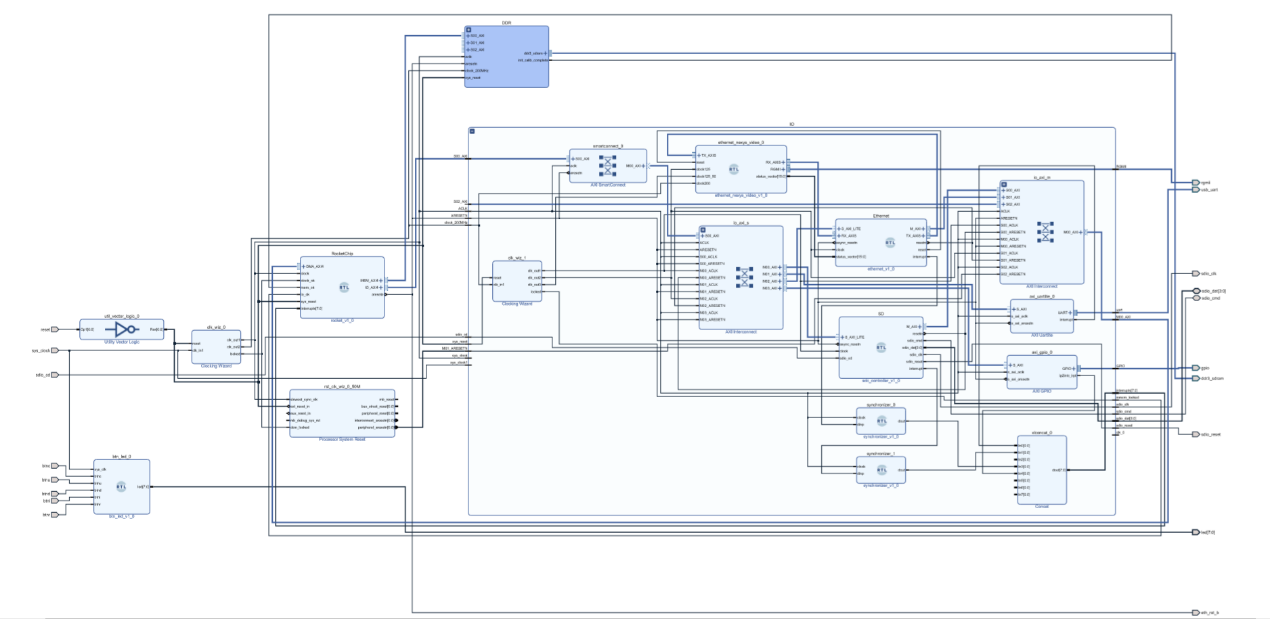


图1 系统原理图

在此基础上，我们完成了基础的GPIO设计以及在逻辑分析仪上显示“RISC-V”字样。开发板打开时，板上的每一路Pmod JA可以输出特定的信号，与逻辑分析仪输入端口连接，便可在逻辑分析仪上显示“RISC-V”字样。此外，还可以通过按开发板上的按钮来改变8个LED灯开关以及闪烁频率。

我们编写了人脸识别和身份证识别两个程序，它们都用C++写成，可以运行在板载的Linux系统中，基于OpenCV进行图像处理。其中，人脸识别程序使用了一个开源的深度学习模型，用于人脸特征抽取。在提取特征后，程序会自动训练一个支持向量机（SVM）进行分类识别。身份证识别程序使用OpenCV的轮廓提取功能提取图片中的数字，并将其与参考字符进行模板匹配以确定是哪个数字或“X”。此外，程序实现了从ftp的下载和上传功能，还可以通过串口与上位机通信，传输转码后的图片，在上位机的LCD显示屏中进行显示。

文末附有演示视频的链接。

1. 基础功能

我们小组在Vivado上使用Verilog HDL语言对开发板进行编程，成功实现了比赛所要求的所有基础功能。

1. GPIO控制LED闪烁

在本项目中，我们采用按键改变开发板上LED灯的闪烁位置以及频率。具体每个按钮对应LED灯的改变如下：

BTNC（B22）按钮使得8个LED灯全亮。

当LED全亮时，按下按钮BTNL（C22）或者BTNR（D14），8个LED灯中相连的3个LED会闪烁。之后使用BTNL（C22）或者BTNR（D14）按钮时则会改变闪烁的LED灯组的位置（分别对应向左移动和向右移动）。

BTNU（F15）按钮使LED组闪烁频率变快。

BTND（D22）按钮使LED组频率闪烁变慢。

此GPIO控制程序由Verilog硬件编程语言编写，代码可以在项目工程中找到。

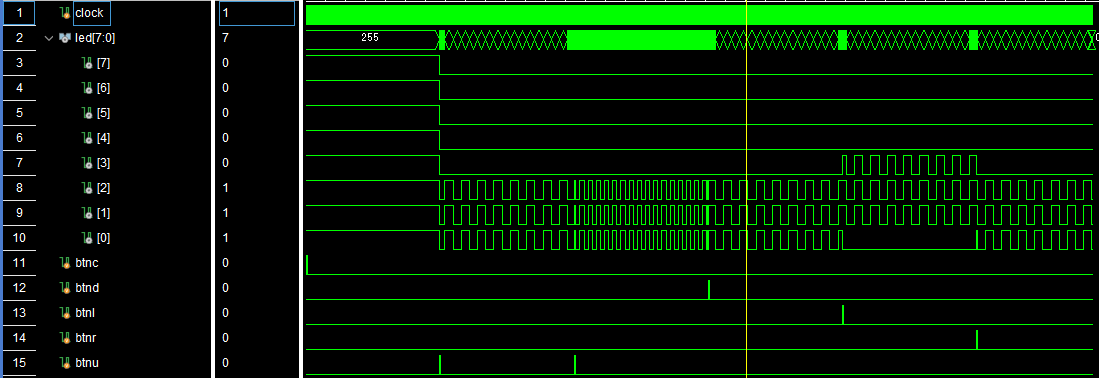


图2 GPIO的仿真效果图，可以看到，按钮可以改变LED灯的开关以及闪烁方式

1. 逻辑分析仪显示“RISC-V”字样

在本项目中，我们比较了使用逻辑分析仪与使用示波器两种方法，最终决定采用八路逻辑分析仪来显示“RISC-V”字样。

在逻辑分析仪上显示字样的原理如下：首先，我们需要产生频率足够高的信号，使得其显示在逻辑分析仪上时可以形成方块。



图3 当信号的频率足够高时，其显示在逻辑分析仪器显示为一个填充的方块

然后，只需要让Nexys Video的8路Pmod JA中每一路端口在特定的时间段内输出频率足够高的信号，剩余时间段保持低电平，并将其连接到逻辑分析仪上，便可以显示特定的字样。

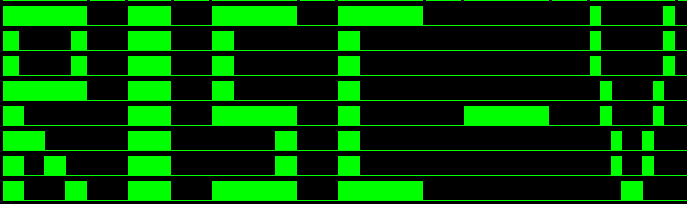


图4 在Vivado2019.2中仿真得到的“RISC-V”字样

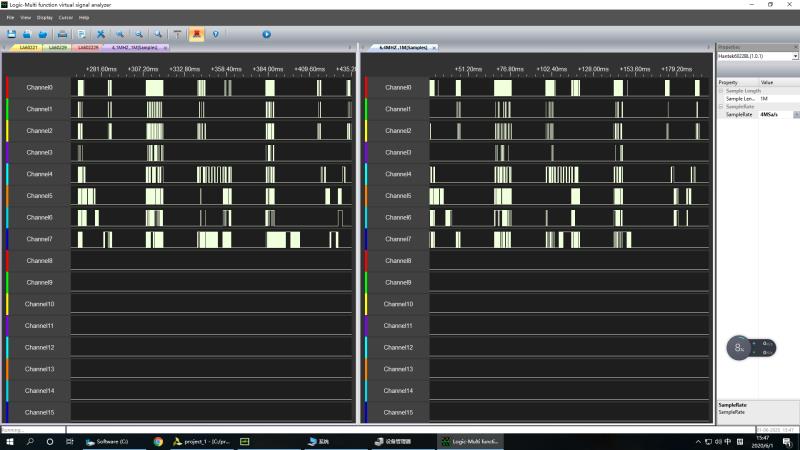


图5 开发板连接逻辑分析仪中画出的“RISC-V”字样。由于逻辑分析仪采样频率较低，显示的字样与仿真中相比质量较差

1. LCD显示屏显示图片

显示屏由一块荔枝派驱动，荔枝派里运行我们编写的Python串口监视程序，与开发板的串口相连，可以向开发板发送命令并回显执行结果。当需要显示图片时，荔枝派通过串口向开发板发送传输数据指令，随后开发板把要显示的图片使用base64编码后通过串口传输到荔枝派，然后再由荔枝派解码出.jpg文件并控制LCD显示器显示。

1. 进阶功能
2. 操作系统

我们在开发板上搭载了Linux操作系统，具体发行版本为Linux Debian 5.6.14。FTP服务器访问，人脸识别程序，身份证识程序别均在板载Linux系统内运行。Linux内核和boot启动程序均已提前配置好，储存在SD卡中，当开发板上电时，系统会自动从SD卡上被加载并启动。板载Linux系统支持串口终端，可以将板上的串口与另外一台电脑连接，通过串口终端操作板载系统。此外，该系统还带有与开发板硬件对应的网口驱动，可以访问互联网，也支持通过ssh远程控制。此板载Linux系统还带有apt包管理器，可以方便的从网络下载安装应用程序，为后续开发提供方便。我们在系统内安装了必要的软件工具，包括GCC，CMake，Python3，OpenCV等。

我们在系统中实现了ftp的上传和下载功能，并且下载的图片能在上位机的LCD显示屏中显示。

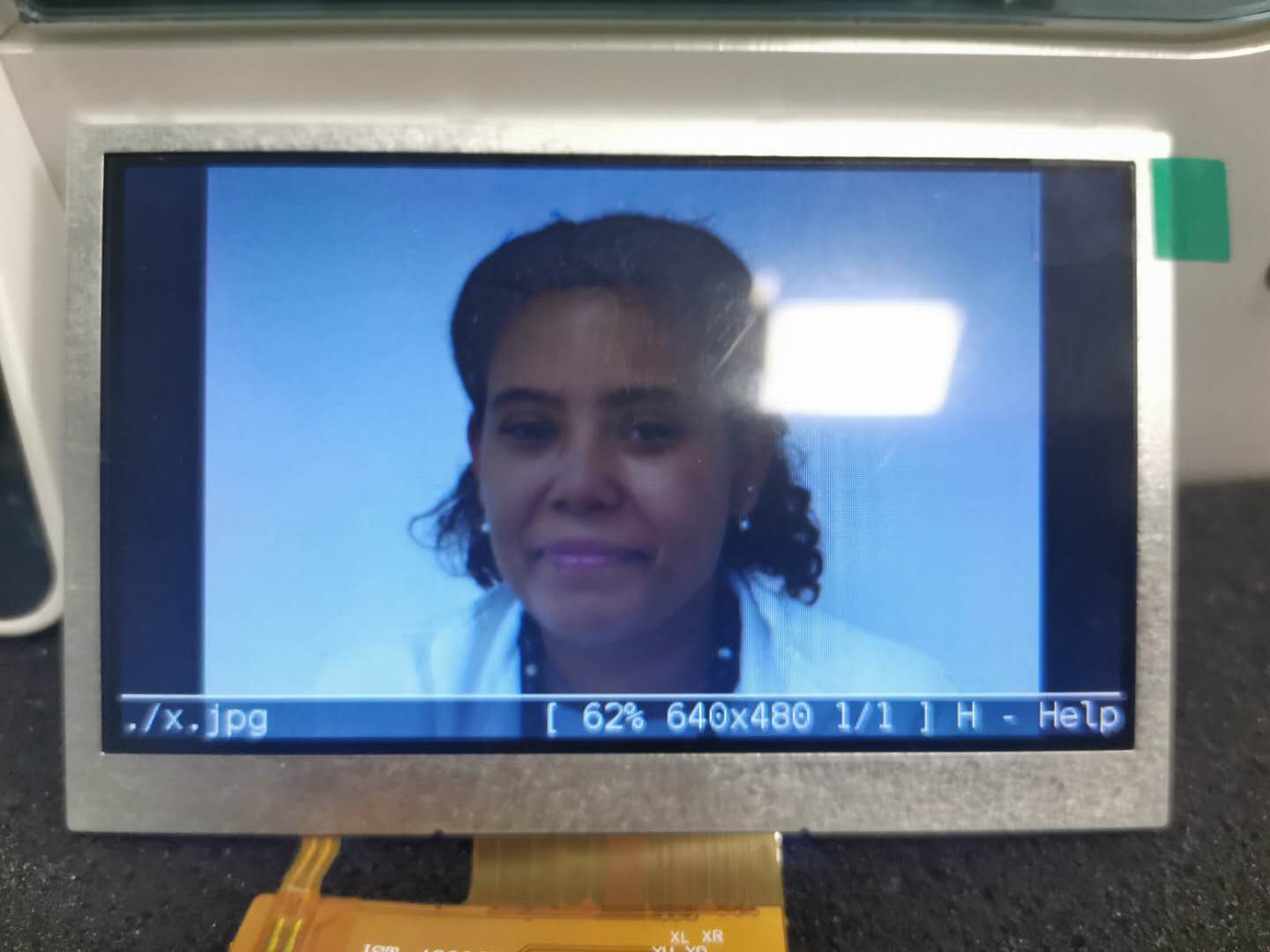


图6 从ftp获取图片并在LCD屏幕上显示

1. 模型训练

我们开始使用的是OpenFace的模型，但该模型在有侧脸存在时识别效果较差。因此我们重新选择了更复杂的神经网络。由于很多复杂神经网络的某些层在OpenCV中不受支持，我们最终选取了FaceNet的简化版本facenet-light，该模型基于facenet进行修改，完全采用opencv支持的层和模块，在LFW验证集上的准确率可达到94%左右。我们下载了VGGFace2人脸数据集，截取并校正得到182 px \*182 px的人脸图片，采用学习率（learning rate）lr=0.001，使用部分的人脸数据训练了100个周期，最终得到的神经网络准确率约为88%±0.5%。由于时间和训练集大小的限制，我们训练的模型在准确率上与原作者有所差距，但相比于原有的模型速度更快，且侧脸识别效果略好。

1. 人脸识别程序

人脸识别程序用C++写成，运行在板载的Linux系统上。该程序使用OpenCV进行图像处理。它使用了一个是FaceNet模型来提取人脸特征。该深度学习网络由OpenCV的dnn模块加载使用。此外，程序还加载了OpenCV库的svm模块，训练了一个支持向量机对提取的人脸特征进行分类识别。

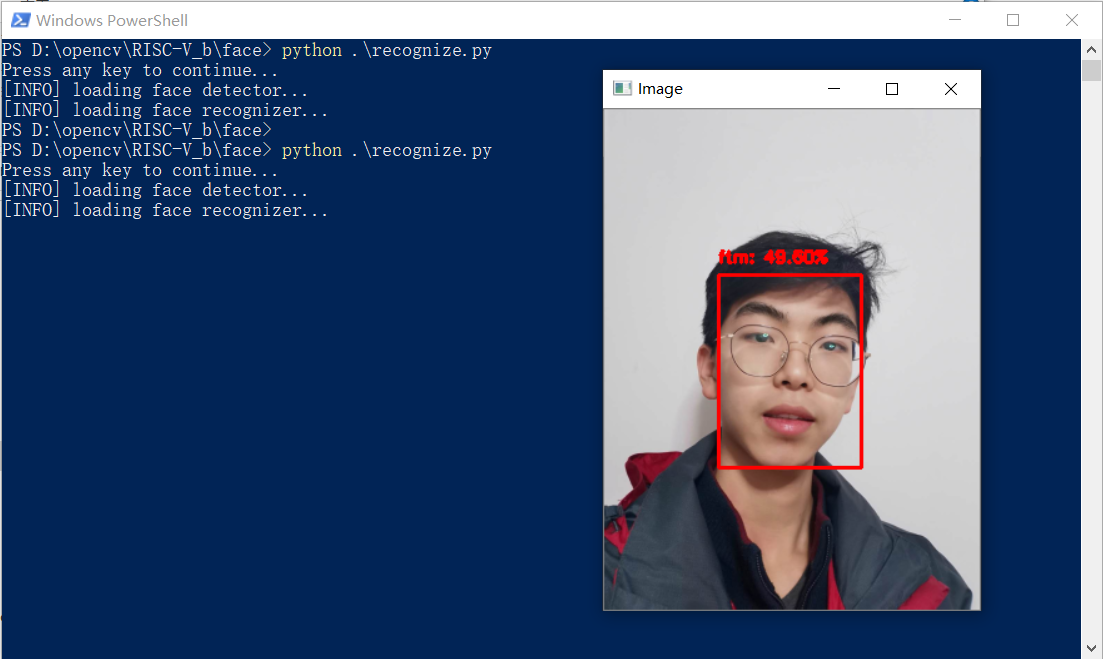
具体来说，程序分为3个部分。第一部分在extract()中，主要是加载参考图片，定位其中的人脸，并提取特征，第二部分在train()中，主要是用第一步提取的特征训练SVM，实现分类。第三部分在recognize()中，主要是加载待识别的图片，定位人脸，提取特征，并使用SVM进行分类，输出人脸对应的名字。程序运行过程中控制台会有指示运行进度的输出。识别结果会写入.txt文件，也会在控制台输出。

图7 仿真运行人脸识别（Windows系统）

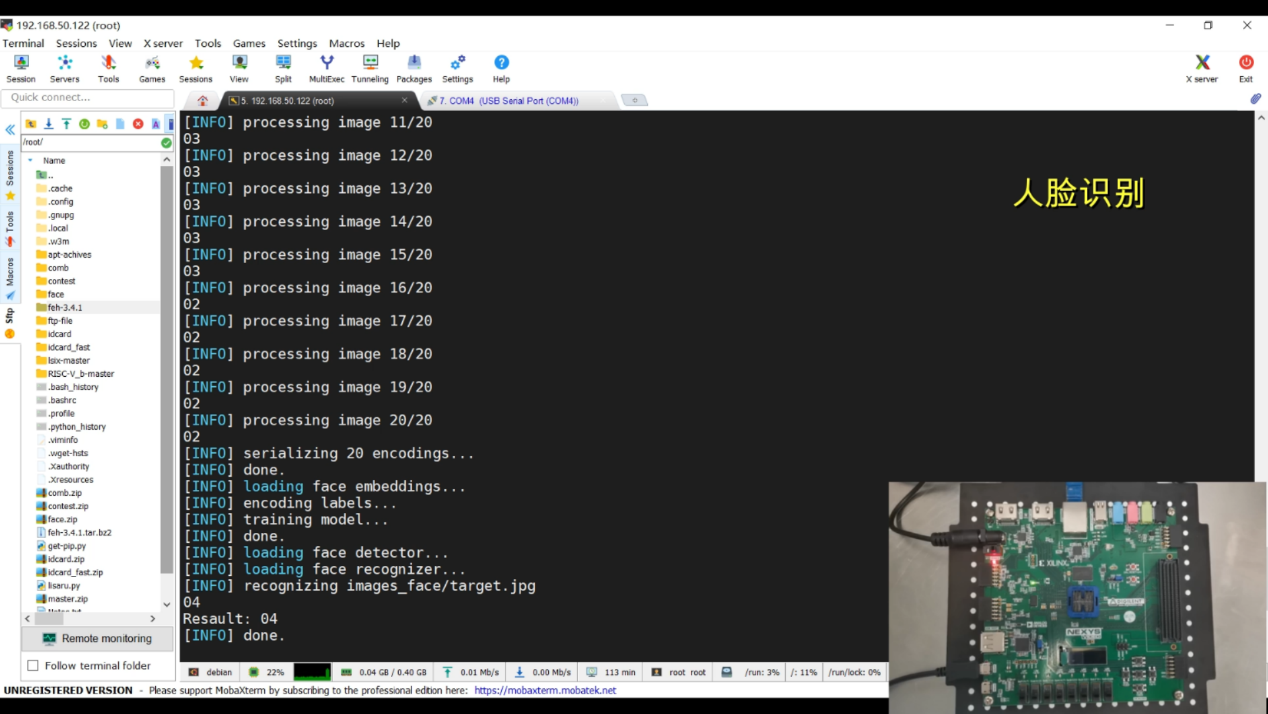


图8 开发板运行人脸识别

1. 身份证识别程序

身份证识别程序用C++写成，并使用OpenCV进行图像处理。首先，程序加载参考图像，参考图像依次含有“0123456789X”的字符，并且字体与待识别图片一致。程序使用OpenCV的轮廓提取功能从参考图像中依次提取每个字符，然后用同样的方法提取待识别图像中的每一个字符，将其与参考图像进行模板匹配，以确定是哪一个数字或“X”。识别结果会写入txt文件，也会在控制台输出。

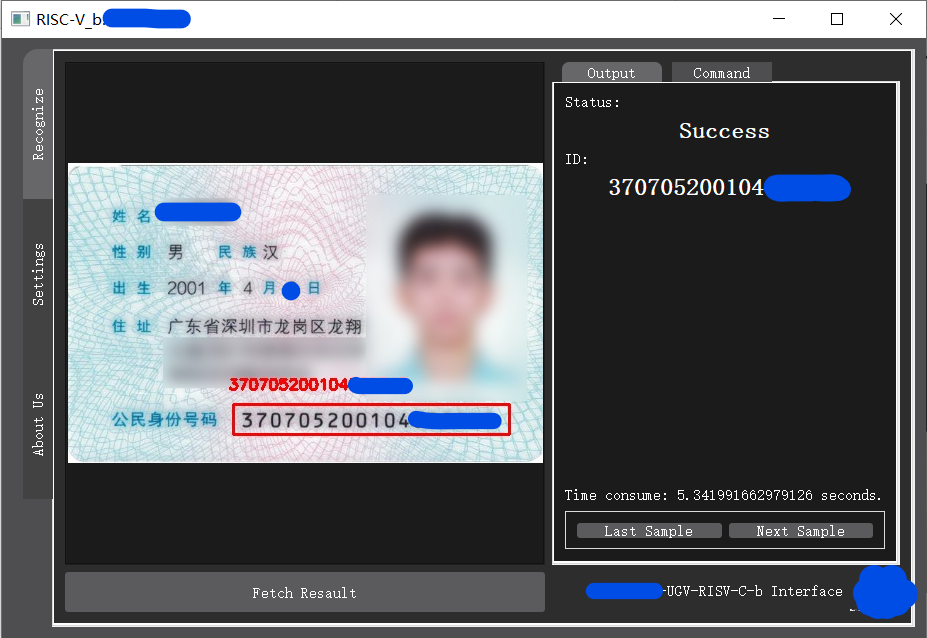


图9 仿真运行身份证识别（在比赛的身份证示例图片发出之前，我们识别的目标是身份证照片，现在已经改为识别比赛要求的格式。此图形界面是我们自己使用PyQt编写的）

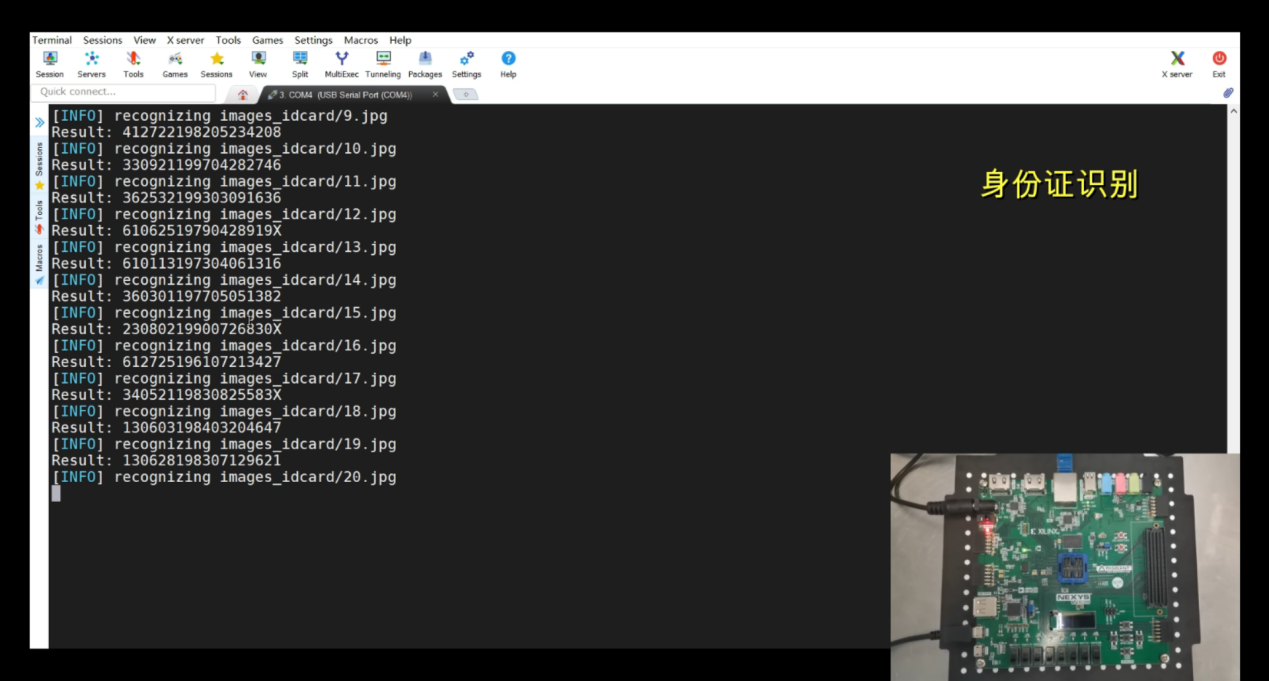


图10 开发板运行身份证识别

1. 程序优化

随着比赛的进行，我们小组也在不断改进程序，优化人脸以及身份证号识别的性能。在身份证识别上，我们改进了算法，对模板和提取的字符先进行降采样，压缩至一个较小的分辨率，以减少后续模板匹配的运算量，提高识别速率。我们进行了多次不同压缩参数的实验，在保证准确度的情况下确定了一个最佳压缩比。在人脸识别方面，我们采用了一个自己训练的新模型，相较于最开始的模型，有效地提高了侧脸识别的准确率。

为了方便使用深度学习模型，我们一开始使用Python编写程序。但考虑到Python中import模块的用时较长，我们又重写了C++版本，节省了大量的时间。

另外，我们观察到运行身份证识别程序时，开发板CPU的使用率较低，于是我们又实现了多线程同时处理来充分利用CPU算力。在开发板从FTP服务器提取图片后，同时使用多个线程识别身份证图片。

最终，我们使用自己的ftp服务器测试，在板上运行身份证识别的用时达到了19秒以内，人脸识别用时5分10秒左右。

1. 命令入口

为了方便程序的使用，也为了适应比赛竞速的需求，避免计时过程中手动输入命令，我们将两个识别程序和FTP服务程序结合起来，开发了一个统一的命令入口。我们把几个比赛要求的功能设计成一个命令，比如只要输入“face”指令，就可以自动完成FTP图片获取、人脸识别、结果回传等。具体命令详见附录。

1. 参考文献

**人脸识别教程：**https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

**身份证识别教程：**https://www.pyimagesearch.com/2017/07/17/credit-card-ocr-with-opencv-and-Python/

**人脸检测神经网络：**https://github.com/thegopieffect/computer\_vision/blob/master/CAFFE\_DNN/res10\_300x300\_ssd\_iter\_140000.caffemodel

**人脸特征提取神经网络模型：**

Github:

notecola：facenet-light. https://github.com/notecola/facenet-light

tbmoon: facenet. https://github.com/tbmoon/facenet

davidsandberg: facenet. https://github.com/davidsandberg/facenet

Paper:

F. Schroff, D. Kalenichenko, & J. Philbin. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering.IEEE *Computer Society Conference on Computer Vision and Pattern Recognition 2015.* Retrieved from: https://arxiv.org/abs/1503.03832

Dataset:

Q. Cao, L. Shen, W. Xie, O. M. Parkhi, A. Zisserman. (2018). VGGFace2: A dataset for recognising face across pose and age *International Conference on Automatic Face and Gesture Recognition.*

Retrieved from: https://github.com/ox-vgg/vgg\_face2

**FPGA RISC-V SoC：**

Xilinx Vivado block designs for FPGA RISC-V SoC running Debian Linux distro. https://github.com/eugene-tarassov/vivado-risc-v

- aswaterman. RISC-V Proxy Kernel. https://github.com/riscv/riscv-pk/tree/6fa3555cc501ab1dfb034061e991e065e2e54253

- ucbjrl. Rocket Chip Generator. https://github.com/chipsalliance/rocket-chip/tree/18c91b2e32fea96d9693c3c5d1ce488fe7ff0664

- trini. "Das U-Boot" Source Tree. https://github.com/u-boot/u-boot/tree/36fec02b1f90b92cf51ec531564f9284eae27ab4

1. 附录
2. 命令清单

|  |  |  |
| --- | --- | --- |
| **命令** | **简写** | **操作** |
| extract | ex | 提取数据集中每张人脸的特征 |
| train | tr | 根据人脸特征训练SVM进行分类 |
| recognize | re | 识别target图片中的人脸 |
| ocr |  | 识别身份证数字 |
| connect |  | 连接ftp服务器 |
| show |  | 从ftp服务器下载指定图片并显示（需要输入ftp上图片的路径） |
| face |  | 人脸识别。程序会自动从ftp拉需要的数据，并把结果传回ftp |
| idcard | id | 身份证识别。程序会自动从ftp拉需要的数据，并把结果传回ftp |

1. 演示视频

https://www.bilibili.com/video/BV1XV411U7pc/



图11 演示视频二维码

1. 程序代码

**身份证识别和人脸识别C++代码：**

#include <cmath>

#include <vector>

#include "ftp.cpp"

#include <fstream>

#include <stdlib.h>

#include <unistd.h>

#include <iostream>

#include <pthread.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <opencv2/ml/ml.hpp>

#include <opencv2/opencv.hpp>

#include <opencv2/dnn/dnn.hpp>

#include <opencv2/core/types.hpp>

using namespace cv::ml;

using namespace std;

bool enable\_ftp = true;

void face(const cv::Ptr<SVM>& svm, cv::dnn::Net& embedder);

void extract(cv::dnn::Net& embedder);

void train(const cv::Ptr<SVM>& svm);

void recognize(const cv::Ptr<SVM>& svm, cv::dnn::Net& embedder);

void idcard();

void ocr\_load(vector<cv::Mat>& refs);

void ocr(vector<cv::Mat>& refs);

// void pull\_ocr(int a, int b, vector<cv::Mat>& refs);

void \*pull\_ocr(void\* args);

void ftp\_pull(const string& remote, const string& local);

void ftp\_push(const string& local, const string& remote);

vector<int> known\_names = vector<int>(); // Names of people

vector<cv::Mat> known\_embeddings = vector<cv::Mat>();

// Collections of image features from dnn net

cv::Mat trainingDataMat;

cv::Mat labels;

int num\_of\_people = 5;

string photolist[20] = {

"1\_a.jpg", "1\_b.jpg", "1\_c.jpg", "1\_d.jpg",

"2\_a.jpg", "2\_b.jpg", "2\_c.jpg", "2\_d.jpg",

"3\_a.jpg", "3\_b.jpg", "3\_c.jpg", "3\_d.jpg",

"4\_a.jpg", "4\_b.jpg", "4\_c.jpg", "4\_d.jpg",

"5\_a.jpg", "5\_b.jpg", "5\_c.jpg", "5\_d.jpg"

};

vector<cv::Mat> refs = vector<cv::Mat>();

pthread\_t tid[4];

string ftpip = "10.30.11.68";

string userpass = "ugv:ugv";

string user = "ugv";

string pass = "ugv";

int main(){

ocr\_load(refs);

cout << "[INFO] loading face recognizer..." << endl;

cv::dnn::Net embedder = cv::dnn::readNet("epoch40.onnx","","ONNX");

// Facenet-light pretrained with VGGFace2 dataset

cv::Ptr<SVM> svm = SVM::create();

string op;

while (true){

cout << "command: ";

cin >> op;

if (op == "id"){

//idcard();

for (int i = 1; i <= 20; i++){

cout<<i<<endl;

ftp\_pull("id\_database/" + to\_string(i) + ".jpg", "images\_idcard/" + to\_string(i) + ".jpg");

}

}

else if (op == "face"){

face(svm, embedder);

}

else if (op == "quit"){

cout << "[Done] Goodbye!"<< endl;

exit(0);

}

else {

cout << "[ERROR] Invalid command!"<< endl;

}

}

return 0;

}

void face(const cv::Ptr<SVM>& svm, cv::dnn::Net& embedder){

cout << "[INFO] pulling photos from ftp..." << endl;

for (int i = 0; i < 20; i++){

ftp\_pull("face\_database/" + photolist[i], "database/" + photolist[i]);

}

extract(embedder);

train(svm);

recognize(svm, embedder);

}

/\* Extract image feature vectors by DL model induction\*/

void extract(cv::dnn::Net& embedder){

cout << "[INFO] Quantifying faces..." << endl;

for (int i = 0; i < 20; i++){

cout << "[INFO] processing image: " << i+1 << "/20" << endl; // i -> i+1

string img\_path = "dataset/" + photolist[i];

cout<<img\_path<<endl;

int intName = i / (20 / num\_of\_people) + 1;

cv::Mat image = cv::imread(img\_path);

cv::Rect rect(170, 70, 300, 300);

cv::Mat face = image(rect);

// cv::imshow("face", face);

// cv::waitKey(0);

cv::Size face\_size = cv::Size(96,96); // Input size of the model should be 96x96

cv::Scalar mean = cv::Scalar(0,0,0);

cv::Mat face\_blob = cv::dnn::blobFromImage(face, 1.0, face\_size, mean, true, false, 0);

embedder.setInput(face\_blob);

cv::Mat vec = embedder.forward(); // Shallow copy cause vec changes, remains to fixed

known\_names.push\_back(intName);

known\_embeddings.push\_back(vec.reshape(1, 1).clone());

// cout << "name: " << known\_names[i] << endl;

// cout << "vec: " << known\_embeddings[i] << endl;

}

}

/\* Train the SVM using face features extracted \*/

void train(const cv::Ptr<SVM>& svm){

cout << "[INFO] Encoding labels..." << endl;

int cols = known\_embeddings[0].cols;

trainingDataMat = cv::Mat(known\_embeddings.size(), cols, CV\_32FC1);

for (int i = 0; i < known\_embeddings.size(); i++){

known\_embeddings[i].row(0).copyTo(trainingDataMat.row(i));

}

labels = cv::Mat(known\_names.size(), 1, CV\_32SC1, &known\_names[0]); // must be integer

// cout << "svm data: " << trainingDataMat << endl;

// cout << "svm lable: " << labels << endl;

svm->setType(SVM::C\_SVC);

svm->setKernel(SVM::LINEAR);

svm->setTermCriteria(cv::TermCriteria(cv::TermCriteria::MAX\_ITER, 100, 1e-6));

cout << "[INFO] Training SVM model..." << endl;

svm->trainAuto(trainingDataMat, ROW\_SAMPLE, labels); // Segfault. guess: if global var, text section not enough

cout << "[INFO] Done." << endl;

}

/\* Classify people through svm\*/

void recognize(const cv::Ptr<SVM>& svm, cv::dnn::Net& embedder){

cout << "[INFO] Recognizing..." << endl;

string path = "image\_face/target.jpg";

cv::Mat image = cv::imread(path);

cv::Rect rect(170, 70, 300, 300);

cv::Mat face = image(rect);

// cv::imshow("face", face);

// cv::waitKey(0);

cv::Size face\_size = cv::Size(96,96);

cv::Scalar mean = cv::Scalar(0,0,0);

cv::Mat face\_blob = cv::dnn::blobFromImage(face, 1.0, face\_size, mean, true, false, 0);

embedder.setInput(face\_blob);

cv::Mat vec = embedder.forward();

cv::Mat result = cv::Mat();

svm->predict(vec, result);

cout << "result: " << result << endl; // The result is already the name (1-5)

}

void idcard(){

cout << "[INFO] Creating threads by pthread\_create..." << endl;

int status;

int num[] = { 1, 6, 11, 16 };

for (int i=0; i<4; i++){

status = pthread\_create(&tid[i], NULL, &pull\_ocr, num + i); // doSomeThing is a test function

if (status){

cout << "[ERROR] Failed to create thread " << i+1 << ". Return status: " << status << endl;

}

else {

cout << "[INFO] Successfully create thread " << i+1 << endl;

}

}

pthread\_join(tid[0], NULL);

pthread\_join(tid[1], NULL);

pthread\_join(tid[2], NULL);

pthread\_join(tid[3], NULL);

ofstream outfile;

outfile.open("result.txt");

for (int i = 0; i < 4; i++){

string fname = to\_string(num[i]) + ".txt";

ifstream infile;

infile.open(fname);

string id;

for (int j = 0; j < 5; j++){

infile >> id;

outfile << id << endl;

}

}

outfile.close();

ftp\_push("result.txt", "result.txt");

cout << "[INFO] ID done!" << endl;

}

void ocr\_load(vector<cv::Mat>& refs){

cout << "[INFO] loading reference..." << endl;

string img\_path = "ocr\_ref.png";

cv::Mat digits = cv::imread(img\_path);

cv::Mat tmp;

cv::cvtColor(digits, tmp, cv::COLOR\_BGR2GRAY);

cv::threshold(tmp, digits, 127, 255, cv::THRESH\_BINARY\_INV);

vector< vector<cv::Point> > contours;

vector< cv::Vec4i > hierarchy;

cv::findContours(digits, contours, hierarchy, cv::RETR\_EXTERNAL, cv::CHAIN\_APPROX\_SIMPLE);

for (int i = 0; i < contours.size(); i++){

cv::Rect bounding = cv::boundingRect(contours[i]);

cv::Mat tmp = digits(bounding);

// cv::imshow("roi", tmp);

// cv::waitKey(0);

cv::Mat roi;

cv::resize(tmp, roi, cv::Size(10, 16));

refs.push\_back(roi.clone());

}

}

string ocr(string fpath, vector<cv::Mat>& refs){

cv::Mat image = cv::imread(fpath);

cv::Rect rect(45, 150, 400, 40);

cv::Mat digits = image(rect);

// cv::imshow("digits", digits);

// cv::waitKey(0);

cv::Mat tmp;

cv::cvtColor(digits, tmp, cv::COLOR\_BGR2GRAY);

cv::threshold(tmp, digits, 127, 255, cv::THRESH\_BINARY\_INV);

vector< vector<cv::Point> > contours;

vector< cv::Vec4i > hierarchy;

cv::findContours(digits, contours, hierarchy, cv::RETR\_EXTERNAL, cv::CHAIN\_APPROX\_SIMPLE);

// cout << "contours: " << endl;

// for (int i = 0; i < contours.size(); i++){

// cout << contours[i] << endl;

// }

// cout << "hierarchy: " << endl;

// for (int i = 0; i < hierarchy.size(); i++){

// cout << hierarchy[i] << endl;

// }

vector<char> output;

for (int i = 0; i < contours.size(); i++){

cv::Rect bounding = cv::boundingRect(contours[i]);

cv::Mat tmp = digits(bounding);

// cv::imshow("roi", tmp);

// cv::waitKey(0);

cv::Mat roi;

cv::resize(tmp, roi, cv::Size(10, 16));

int num = -1;

int mxscore = -1;

for (int j = 0; j < refs.size(); j++){

int res\_rows = roi.rows - refs[j].rows + 1;

int res\_cols = roi.cols - refs[j].cols + 1;

cv::Mat res(res\_rows, res\_cols, CV\_32FC1);

cv::matchTemplate(roi, refs[j], res, cv::TM\_CCOEFF);

double min, score;

cv::minMaxLoc(res, &min, &score);

if (score > mxscore){

mxscore = score;

num = 10 - j;

}

}

if (num == 10){

output.push\_back('X');

}

else{

output.push\_back('0' + num);

}

}

string res = "";

for (int i = output.size() - 1; i >= 0; i--){

res += output[i];

}

return res;

}

/\* void pull\_ocr(int a, int b, vector<cv::Mat>& refs){

ofstream outfile;

outfile.open(to\_string(a) + ".txt");

for (int i = a; i <= b; i++){

string fname = to\_string(i) + ".jpg";

ftp\_pull("id\_database/" + fname, "images\_idcard/" + fname);

string res = ocr("images\_idcard/" + fname, refs);

outfile << res << endl;

}

outfile.close();

} \*/

void \*pull\_ocr(void\* args){

int a = \*(int\*)args;

cout<<a<<endl;

ofstream outfile;

outfile.open(to\_string(a) + ".txt");

for (int i = a; i < a+5; i++){

string fname = to\_string(i) + ".jpg";

ftp\_pull("id\_database/" + fname, "images\_idcard/" + fname);

string res = ocr("images\_idcard/" + fname, refs);

outfile << res << endl;

}

outfile.close();

return NULL;

}

void ftp\_pull(const string& remote, const string& local){

if (!enable\_ftp) return;

//system(("curl -s ftp://" + ftpip + "/" + remote + " -u " + userpass + " -o " + local).c\_str());

// curl ftp://malu.me/size.zip –u name:passwd -o size.zip

FtpDownload("ftp://" + ftpip + "/" + remote, local, user, pass);

}

void ftp\_push(const string& local, const string& remote){

if (!enable\_ftp) return;

//system(("curl -s -u " + userpass + " -T " + local + " ftp://" + ftpip + "/" + remote).c\_str());

// curl –u name:passwd -T size.mp3 ftp://malu.me/mp3/

FtpUpload("ftp://" + ftpip + "/" + remote, local, user, pass);

}

**LED灯控制代码：**

`timescale 1ps / 1ps

module btn\_led(

input sys\_clk,

input btnc,

input btnu,

input btnd,

input btnl,

input btnr,

output [7:0]led,

output [7:0]ja

);

reg [25:0] count = 0;

reg [7:0] a = 0;

reg [7:0] b = 0;

reg [4:0] btn = 0;

always@(posedge sys\_clk) begin

if (btnu == 1) begin

btn = 5'b10000;

end else if (btnd == 1) begin

btn = 5'b01000;

end else if (btnl == 1) begin

btn = 5'b00100;

end else if (btnr == 1) begin

btn = 5'b00010;

end else if (btnc == 1) begin

btn = 5'b00001;

end

count <= count + 1;

if (btn == 5'b00001) begin

if (count < 25'h020\_0000 || ( count > 25'h040\_0000 && count < 25'h050\_0000)

|| ( count > 25'h070\_0000 && count < 25'h090\_0000) || ( count > 25'h0b0\_0000 && count < 25'h0d0\_0000)

|| ( count > 25'h0e0\_0000 && count < 25'h0e4\_0000)

|| ( count > 25'h0fc\_0000 && count < 25'h100\_0000)) begin // \*4

a[0] = count[0];

end else begin

a[0] = 1'b0;

end

if (count < 25'h004\_0000 || ( count > 25'h01c\_0000 && count < 25'h020\_0000)

|| ( count > 25'h040\_0000 && count < 25'h050\_0000) || ( count > 25'h070\_0000 && count < 25'h078\_0000)

|| ( count > 25'h0b0\_0000 && count < 25'h0b8\_0000) || ( count > 25'h0e0\_0000 && count < 25'h0e4\_0000)

|| ( count > 25'h0fc\_0000 && count < 25'h100\_0000)) begin // \*4

a[1] = count[1];

a[2] = count[2];

end else begin

a[1] = 1'b0;

a[2] = 1'b0;

end

if (count < 25'h020\_0000

|| ( count > 25'h040\_0000 && count < 25'h050\_0000) || ( count > 25'h070\_0000 && count < 25'h078\_0000)

|| ( count > 25'h0b0\_0000 && count < 25'h0b8\_0000) || ( count > 25'h0e4\_0000 && count < 25'h0e8\_0000)

|| ( count > 25'h0f8\_0000 && count < 25'h0fc\_0000)) begin // \*4

a[3] = count[3];

end else begin

a[3] = 1'b0;

end

if ( count < 25'h008\_0000

|| ( count > 25'h040\_0000 && count < 25'h050\_0000) || ( count > 25'h070\_0000 && count < 25'h090\_0000)

|| ( count > 25'h0b0\_0000 && count < 25'h0b8\_0000) || ( count > 25'h0e4\_0000 && count < 25'h0e8\_0000)

|| ( count > 25'h0f8\_0000 && count < 25'h0fc\_0000)) begin // \*4

a[4] = count[4];

end else begin

a[4] = 1'b0;

end

if ( count < 25'h010\_0000

|| ( count > 25'h040\_0000 && count < 25'h050\_0000) || ( count > 25'h088\_0000 && count < 25'h090\_0000)

|| ( count > 25'h0b0\_0000 && count < 25'h0b8\_0000) || ( count > 25'h0e8\_0000 && count < 25'h0ec\_0000)

|| ( count > 25'h0f4\_0000 && count < 25'h0f8\_0000)) begin // \*4

a[5] = count[5];

end else begin

a[5] = 1'b0;

end

if ( count < 25'h008\_0000 || ( count > 25'h010\_0000 && count < 25'h018\_0000)

|| ( count > 25'h040\_0000 && count < 25'h050\_0000) || ( count > 25'h088\_0000 && count < 25'h090\_0000)

|| ( count > 25'h0b0\_0000 && count < 25'h0b8\_0000) || ( count > 25'h0e8\_0000 && count < 25'h0ec\_0000)

|| ( count > 25'h0f4\_0000 && count < 25'h0f8\_0000)) begin // \*4

a[6] = count[6];

end else begin

a[6] = 1'b0;

end

if ( count < 25'h008\_0000 || ( count > 25'h018\_0000 && count < 25'h020\_0000)

|| ( count > 25'h040\_0000 && count < 25'h050\_0000) || ( count > 25'h070\_0000 && count < 25'h090\_0000)

|| ( count > 25'h0b0\_0000 && count < 25'h0d0\_0000)

|| ( count > 25'h0ec\_0000 && count < 25'h0f4\_0000)) begin // \*4

a[7] = count[7];

end else begin

a[7] = 1'b0;

end

b = a;

end

else if (btn == 5'b10000) begin

b = 8'b11111111;

end

else if (btn == 5'b01000) begin

b[7] = count[24];

b[6] = count[24];

b[5] = count[24];

b[4] = count[24];

b[3] = count[24];

b[2] = count[24];

b[1] = count[24];

b[0] = count[24];

end

else if (btn == 5'b00100) begin

b[7] = count[25];

b[6] = count[25];

b[5] = count[25];

b[4] = count[25];

b[3] = count[25];

b[2] = count[25];

b[1] = count[25];

b[0] = count[25];

end

else if (btn == 5'b00010) begin

b[7] = count[23];

b[6] = count[23];

b[5] = count[23];

b[4] = count[23];

b[3] = count[23];

b[2] = count[23];

b[1] = count[23];

b[0] = count[23];

end

end

assign led[7:0] = b;

assign ja[7:0] = b;

endmodule