各位老师好，由于我们DIY项目的硬件设计是在lab10的基础上完成的，我们这里先进行一下lab10的演示，等下对硬件进行讲解。

（打开PPT）

接下来介绍我们的项目。我们DIY的项目是戴口罩的人脸识别。在疫情施虐的今天，在一些人脸识别的场合，我们常常需要摘下口罩，如车站检票处，还有我们宿舍楼下的门禁，这样既麻烦又不安全。因此，我们提出了戴口罩人脸识别的项目，即可以识别带着口罩的人脸，当然也可以识别不戴口罩的人脸。

（翻页）

接下来我们将从环境准备、硬件、软件、软硬交互等几个方面讲述我们的项目。

（翻页）

为了充分利用zynq芯片的特性及其FPGA资源、ARM核资源，提高程序的可扩展性，以及利用opencv库函数，我们在开发板上移植了Linux系统，其主要需要BOOT.BIN、image.ub两个启动文件以及根文件系统。为了成功移植，我们使用PetaLinux对Xilinx官方提供的uboot和Linux内核进行了交叉编译，同时根据我们的硬件生成了设备树，最后打包成了启动文件。根文件系统是我们在ubuntu最小文件系统的基础上制作而成。

（翻页）

在硬件上，除了lab10中视频显示所必须的软核外，我们利用HLS生成了一个IP，用于对摄像头的输入进行直方图均衡与高斯滤波。在设备树文件中，可以查询到vdma的物理地址为43000000，在后续vdma启动时，我们需要将这个物理地址映射到进程虚拟地址空间，从而像vdma的寄存器写值。除此之外，我们还添加了按键、LED灯以及摄像头数据位、时钟位的gpio接口，用于软件中的用户交互以及摄像头的驱动。

（翻页）

在软件上，我们采用了两款开源的神经网络，用于人脸检测和计算人脸的特征向量，相比而言，它们都比较轻量化，运行速度更快。为了更好的提高识别算法的性能，我们首先在PC上利用python实现了整体算法，完成了神经网络的验证以及算法的改进。与此同时，我们也利用python对普通图像进行了预处理，以满足神经网络输入的要求，包括直方图均衡，高斯滤波，中值滤波，人脸检测与切割等。只需将包含人脸的照片输入此程序，即可得到预处理后的3乘160乘160的.bmp图片，同时也可以直接计算预处理后的图片的特征向量。

（翻页）

接下来这部分由黄嘉欣同学为大家讲述。

各位老师好，我是黄嘉欣，接下来由我为大家讲述剩下的内容。

整个项目的人脸识别程序，可以大致分为人脸检测、口罩检测、训练、人脸识别四个部分。其中人脸检测采用了libfacedetection网络，可以在图像中找到戴口罩或不戴口罩人脸的位置。口罩检测对裁剪出的人脸图像进行二值化。由于一般口罩的口罩是黑色、白色、蓝色三种，我们将人脸图片中处于这三种颜色范围内的部分统一为白色，剩余部分为黑色，通过计算最大块白色的面积占整张人脸的比例，判断其是否戴有口罩。如果在程序启动时已经加载了训练好的模型，我们可以跳过训练部分直接进行人脸识别。训练实际上是利用Mobile FaceNet网络计算前两步保存的人脸的特征向量，并将其与人脸的标签作为键值对保存起来，作为模型。当检测到模型存在并切换到识别模式后，算法会计算当前检测到人脸的特征向量与已保存的特征向量之间的欧式距离，取其中距离最小且小于阈值的特征向量对应的标签作为识别结果。

（翻页）

项目的软硬件交互主要涉及到GPIO和VDMA两个部分。这里先讲一下GPIO。当设备树中生成GPIO节点以后，linux系统下会对应的生成目录，如图中所示，其中gpio1014 - gpio1021对应的是LED指示灯及按键，主要用于算法模式的检测，gpio1022和gpio1023对应摄像头的时钟位和数据位，用于摄像头的驱动。在目录下的value文件中保存有gpio对应管脚的电平值，利用这个特点我们就可以在程序中读取或者更改该管脚的状态。根据ov7725摄像头的官方文档，其启动实际上是往摄像头的寄存器中写值，每次写值开始和结束时都有固定的时钟、数据电平变化，（点击）这里的DATA_HIGH就是往数据位写入高电平的意思，如右边这一部分所示。（点击）在具体写某个寄存器时，我们需要将并行的值转换成串行输出，从高位到低位写入到value文件中。

（翻页）

在VDMA这一部分，需要讲一下我们程序读取到摄像头输入图像的方法。由于VDMA指定了视频流保存和读取的物理地址，在三帧缓存的模式下，VDMA会将摄像头输入的视频流写到写通道所指向的某一帧缓存，这里假设是帧缓存1，与此同时，VDMA会将读通道所指向的帧缓存中的数据传输给VGA显示，这里假设为帧缓存2。我们的程序会从VDMA的寄存器中获取到读通道指向的帧缓存，并从中读取图像数据。经过大约1-2秒的处理后，再将指示有人脸位置的图像写到写通道指向的帧缓存。（点击）假设这里写通道指向了帧缓存3。这样看起来可行，但由于图像处理的速度太慢、而视频刷新的速度很快，往往我们刚写回一帧，它就会被视频流冲掉，VGA显示上就会出现闪烁、不连贯的问题。（点击）因此这种方法是不可行的。

（翻页）

为了解决这个问题，我们将VDMA改为了两帧缓存，并将它的读写地址进行了分离，这里VIDEO_BASEADDR0和VIDEO_BASEADDR2都对应帧缓存1，VIDEO_BASEADDR1和VIDEO_BASEADDR3都对应帧缓存2。（点击）与此同时，我们还将人脸识别的程序和图像显示的程序分离了开来。当检测到读通道指向帧缓存1时，我们的两个程序都会从帧缓存1对应的写地址中读取图像，这里需要注意的是，Application即人脸识别的程序仍然需要1-2s的时间进行图像的处理，而showFrame只对图形进行人脸的标识，因此showFrame的刷新速度很快，基本上都是实时的。（点击）Application找到人脸位置后，会将其写入一个txt文件，showFrame会读取txt文件并在它现在处理的这一帧的对应位置画框。这样的处理使得VGA的显示连贯，且不会出现残影、闪烁等问题，但会因此而引入人脸框的一定延迟。

（翻页）

最后是我们测试时的一些效果展示图。蓝翔同学预设的标签是2，我预设的标签是3，可以看到这里识别出来他的identity都是2，然后是否戴口罩都检测正确。（点击）这下面两张图片是我测试时的输出，这里的identity都是3。

最后，我们现场演示一下我们的项目。