2019 年 SEU-Xilinx 国际暑期学校团队项目设计文档

(Project Paper Submission Template)

作品名称	基于 Ultra96 和 DPU 的人脸检测应用
组员姓名	刘泽世 中国科学院大学 计算机系统结构 侯兴中 北京大学 计算机科学与技术 石正源 山东大学 信息科学与工程学院
房间号及桌号	717 - 19

第一部分

小组成员分工

	姓名	任务分配
组长	刘泽世	Application 开发、性能测试
组员 1	侯兴中	深度学习模型压缩、定点，Vivado 硬件平台搭建
组员 2	石正源	PetaLinux 生成镜像和 Sysroot 环境
组员 3		

第二部分

设计概述 /Design Introduction

随着 FPGA 的不断发展，FPGA 的应用领域也越来越广泛，在目前最热门的人工智能领域，已经出现了较为成熟的 FPGA 开发方法，如 DPU。而人工智能作为新一轮科技革命和产业变革的核心力量，正在推动传统产业升级换代，驱动“无人经济”快速发展，在智能交通、智能家居、智能医疗等领域产生重要的影响，极大程度地提高了人们的生活水平，推动了科技的发展。基于人工智能的高效和 FPGA 的灵活性与便携性，在 FPGA 上开发人工智能的软硬件结合的开发方式成为未来人工智能发展的方向之一。

基于此，我们学习了 DPU Integration Lab，在 Ultra96 上实现人脸检测功能，熟悉借助 Vivado 和 PetaLinux 生成 SD Card 镜像的设计流程，掌握使用 DPU 对 caffemodel 进行压缩定点的方法，并最终通过 SDK 生成能在 FPGA 上运行的人脸检测程序。我们希望通过这次实验，能够熟练地掌握 DPU 的开发流程，并且能够将不同的深度学习模型布置在 FPGA 上进行预测。

我们在 Ultra96 上实现的人脸检测功能，可以很方便地嵌入在各种系统中，如打卡系统、人脸支付系统等等，并且经过 DPU 压缩后的模型可以达到很高的实时性。同时也可以通过变换深度学习的模型来实现不同的功能，比如说我们可以在 Ultra96 上布置一个压缩后的车辆识别模型，这样可以将 Ultra96 布置在各个路段来达到监控的目的，或者说布置在无人汽车上协助驾驶。总之，在 FPGA 上布置深度学习模型可以应用在所有深度学习可以涉及到的领域，范围十分广泛。

所用设备：显示器、Ultra96、摄像头、SD 卡

第三部分

详细设计 /Detailed Design

实现功能：读取 USB 摄像头回传图像，基于 DPU 进行实时人脸检测，并通过 mini DP 输出到屏幕

开发流程图：

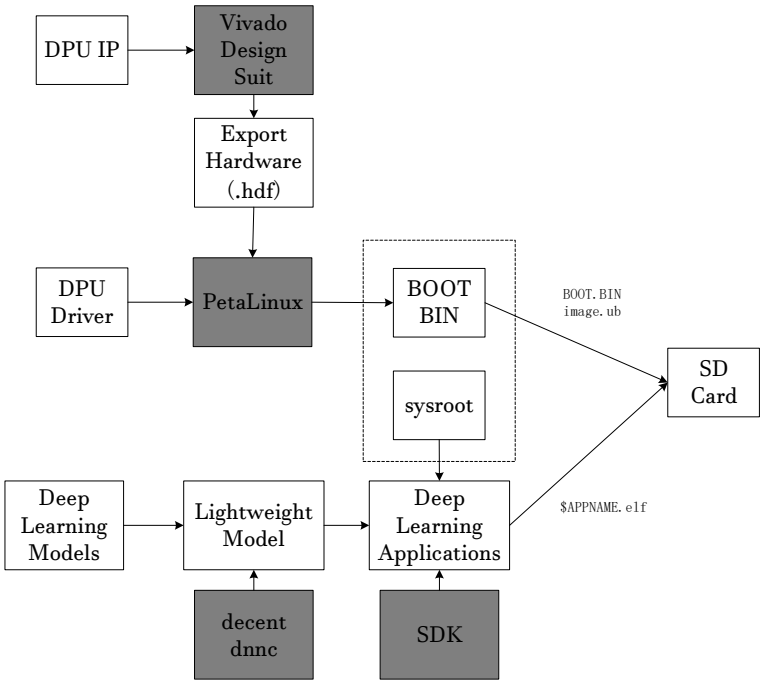


图.1

在项目开发中将整体分为四个部分，DNNDK 训练网络模型、PetaLinux 生成 Linux 镜像与 SDK 编译环境、Vivado HLx 设计 FPGA 硬件结构和 Vivado SDK 编译生成应用。

神经网络的训练在服务器上进行，服务器环境为 Ubuntu 16.04、Opencv3.4、

在 Windows 10 环境上运行 Vivado，利用给定的 u96_dpu_bd.tcl 简化开发步骤，最终生成.hdf 文件。PetaLinux 开发工具本质是自定义 Linux 内核并编译，为避免交叉编译出现不可预知的错误，我们首先考虑在 Ultra96 板卡上烧写 Linux 镜像，配置环境完成编译，但由于算力有限，我们调用了学校提供的服务器完成工作，服务器操作系统为 Ubuntu 16.04，内存 64GB，共两个物理 CPU，型号为 Intel® Xeon E5-2620 v4 (2.10GHz)。

```
zhengyuan@sdu-dpai-turing-2:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 79
Model name:            Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
Stepping:              1
CPU MHz:               1200.125
CPU max MHz:           3000.0000
CPU min MHz:           1200.0000
BogoMIPS:              4201.78
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              20480K
NUMA node0 CPU(s):     0-7,16-23
NUMA node1 CPU(s):     8-15,24-31
```

图.2

在调用 petalinux 命令前需要导入路径，输入命令 source $YOUR_PETALINUX_PATH/settings.sh，

然后运行命令 petalinux-create 新建工程，其中包含参数-t （类型：project） -s （source） -n（项目名） --template（项目模板）。进入工程文件夹加入内核文件和 dnndk 文件，修改 petalinux-image.bbappend 文件，加入下列命令调用 DPU：

IMAGE_INSTALL_append = " dnndk"

IMAGE_INSTALL_append = " autostart"

IMAGE_INSTALL_append = " dpu"

运行命令 petalinux-config --get-hw-description=../hsi 导入硬件设置（具体见 UG1141 附录 B 和 C），运行命令 petalinux-config -c rootfs 配置根文件系统，最后运行命令 petalinux-build --sdk 和 petalinux-package --sysroot 命令编译 Linux 内核源码并生成 SDK 编译环境，将 petalinux/images/linux 下的 aarch64-xilinx-linux.tar.gz 文件下载到本地，解压得到目录，该目录作为 SYSROOT 参数导入 SDK，配置 Vivado SDK 交叉编译环境。

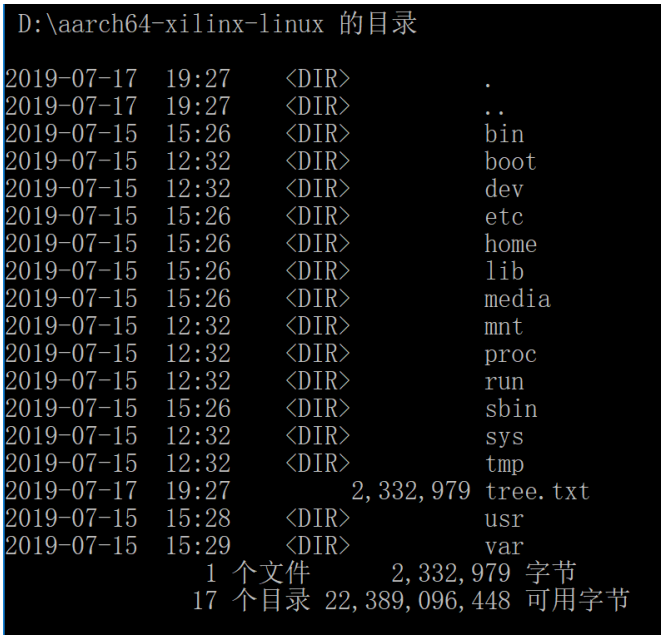


图.3

回到本地 Windows 10 环境，建立一个 Vivado SDK 工程，导入 face_detection.cc 和 dpu_densebox.elf，配置环境变量 SYSROOT=$YOUR_SYSROOT_PATH，并添加 g++ linker 参数--sysroot=${SYSROOT}和 g++ compiler 参数--sysroot=${SYSROOT}，加入编译所需要的库，指定 dnndk.h 路径。这里建议修改#include<dnndk/dnndk.h>为#include<dnndk>，同样修改 dnndk.h 文件内调用的 dputils.h 和 n2cube.h。最终编译生成.elf 文件，为 Ultra96 上 Linux 的可执行文件。

烧写 xilinx-ultra96-prod-dpu1.4-desktop-buster 镜像（建议为 2018-12-10 版本），删去 SD 卡目录下的 BOOT.BIN 文件，拷贝入 PetaLinux 生成的 BOOT.BIN、image.ub 和 SDK 编译生成的.elf 文件，上电运行。

第四部分

完成情况及性能参数 /Final Design & Performance Parameters

在本次实验中，我们利用 Xilinx 提供的解决方案，在 Ultra96 上实现了基于 DPU 的人脸检测应用。

按照预定计划，我们首先使用 Vivado 软件设计了硬件平台的结构，在设计中我们参考了 Xilinx 提供的解决方案中的设计，将一个 DPU 加入硬件平台中，最终的硬件结构如图 4，同时参考 Vivado 的工程报告，我们得到该平台的性能参数如图 5~7 所示。

然后我们搭建了 PetaLinux 环境，并基于 Vivado 导出的.hdf 文件，在 PetaLinux 中制作了对应于

硬件平台的 Linux 镜像，同时导出了 Sysroot 目录作为接下来进行交叉编译的环境（见第六部分 - github 链接）。

另一方面，我们基于 Xilinx 提供的压缩定量工具，对一个已有的深度学习模型进行了压缩和定点，成功生成了该模型压缩定点侯的.elf 文件。

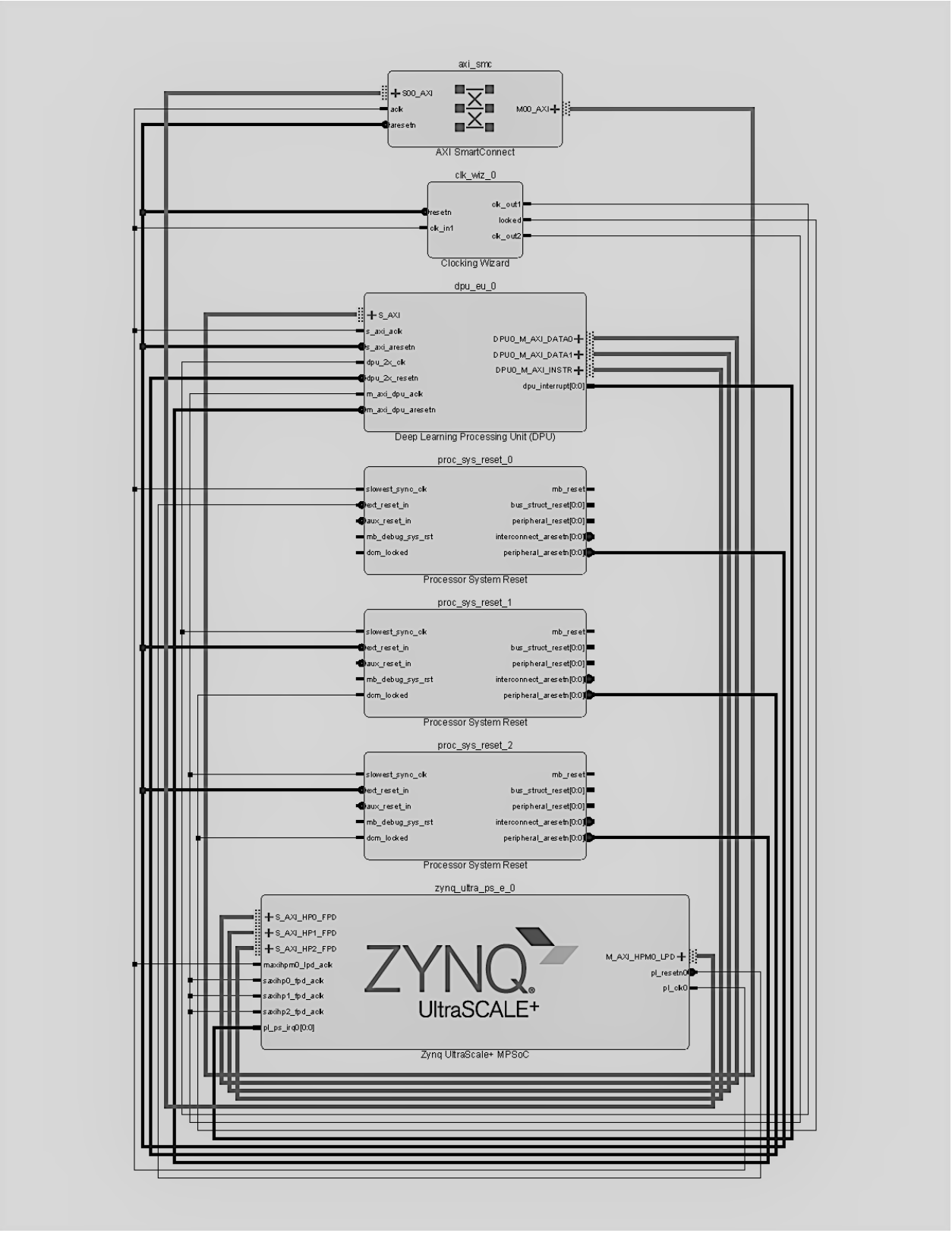


图. 4

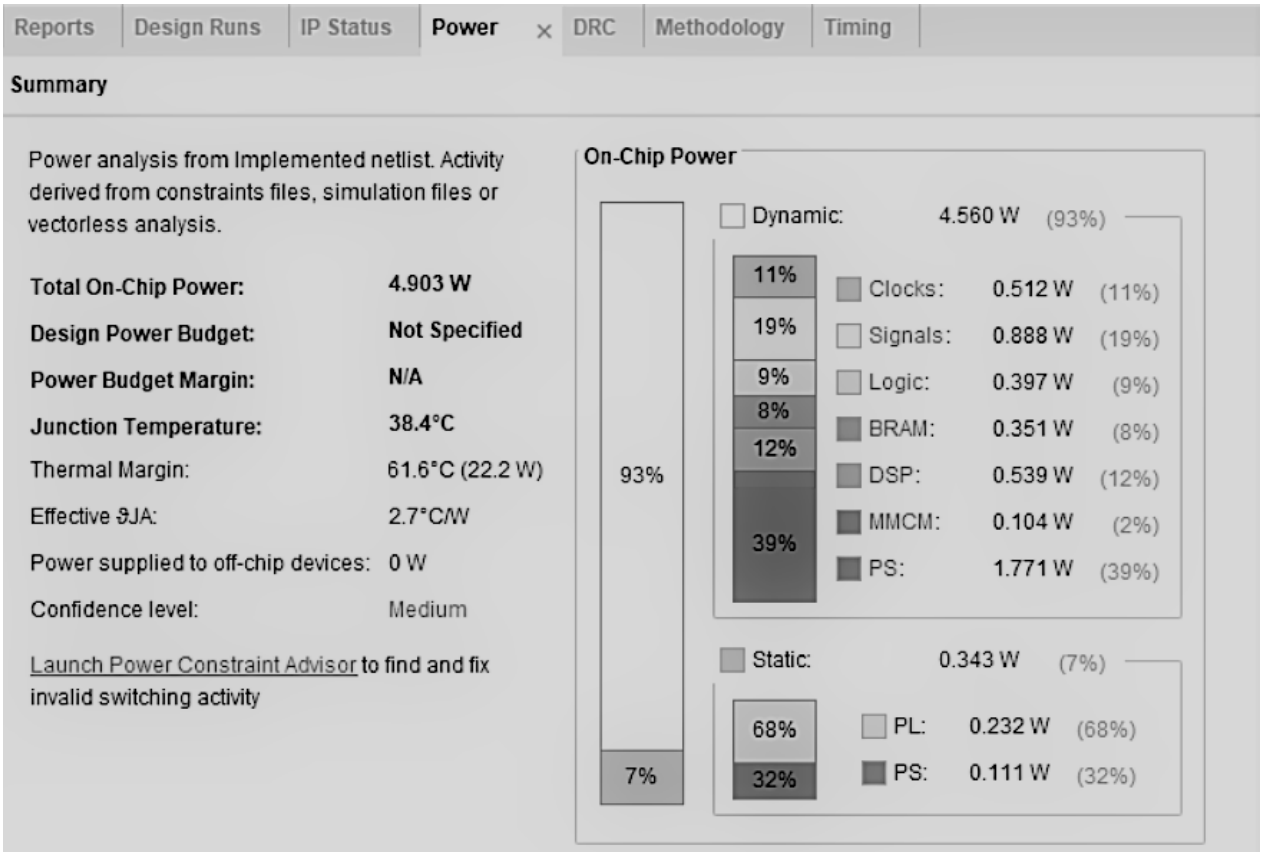

Reports
Design Runs
IP Status
Power
DRC
Methodology
Timing
Summary
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.
Total On-Chip Power: 4.903 W
Design Power Budget: Not Specified
Power Budget Margin: N/A
Junction Temperature: 38.4°C
Thermal Margin: 61.6°C (22.2 W)
Effective ϑJA: 2.7°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
Launch Power Constraint Advisor to find and fix invalid switching activity
On-Chip Power
93%
7%
Dynamic: 4.560 W (93%)
11% Clocks: 0.512 W (11%)
19% Signals: 0.888 W (19%)
9% Logic: 0.397 W (9%)
8% BRAM: 0.351 W (8%)
12% DSP: 0.539 W (12%)
39% PS: 1.771 W (39%)
Static: 0.343 W (7%)
68% PL: 0.232 W (68%)
32% PS: 0.111 W (32%)

图. 5

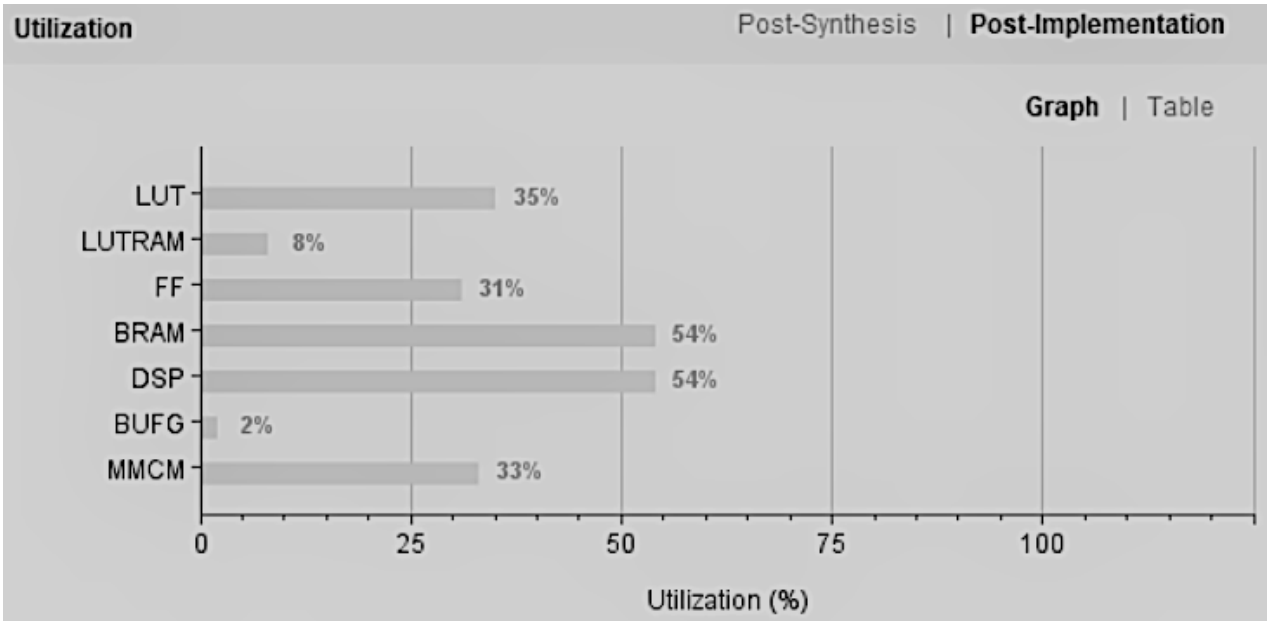
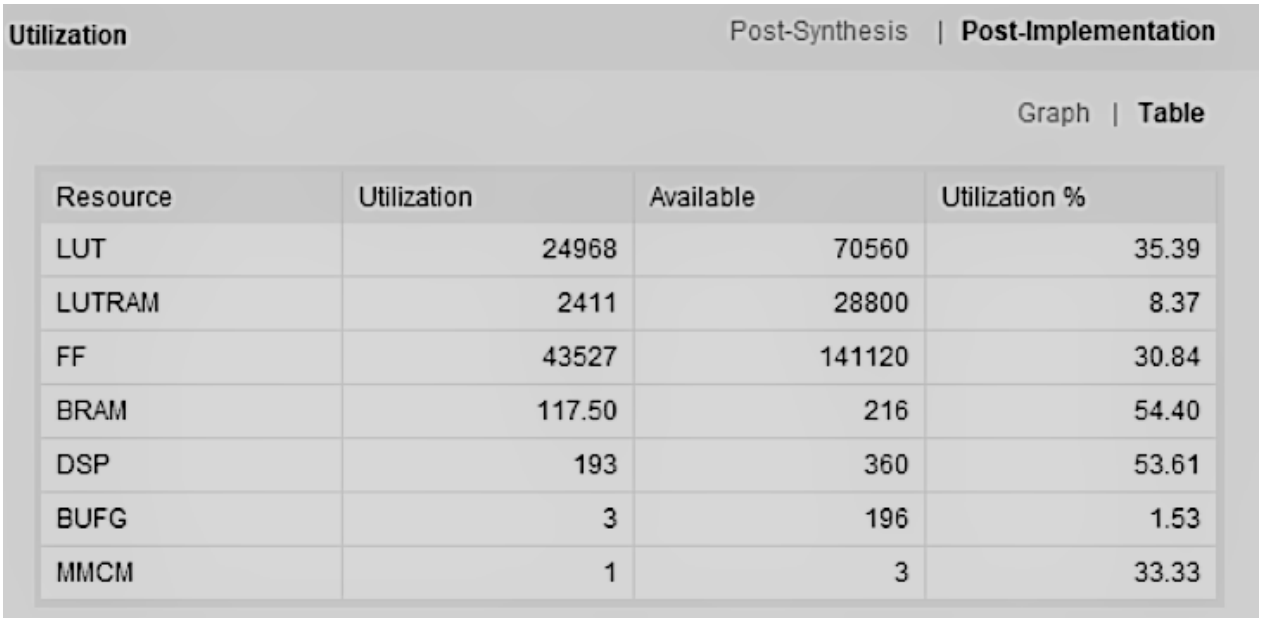

Utilization
Post-Synthesis | Post-Implementation
Graph | Table
LUT 35%
LUTRAM 8%
FF 31%
BRAM 54%
DSP 54%
BUFG 2%
MMCM 33%
0
25
50
75
100
Utilization (%)

图. 6



Utilization | Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	24968	70560	35.39
LUTRAM	2411	28800	8.37
FF	43527	141120	30.84
BRAM	117.50	216	54.40
DSP	193	360	53.61
BUFG	3	196	1.53
MMCM	1	3	33.33

图.7

最后在 application 开发过程中，我们使用 Xilinx SDK 工具，利用前几步中制作的配置文件，进行 ARM 平台上的应用开发。由于在前一步中制作的.elf 文件在这里调试不成功，最后我们使用了 Xilinx 提供的.elf 文件，然后在 SDK 中配置了交叉编译环境，基于 Xilinx 的 face_detection 应用，我们增加了人脸计数和实时性能监测，包括帧率的监测和 CPU 和内存使用率的监测。另外我们尝试了视频的多路显示进行性能对比。

在 SDK 中编译出程序后，我们在 Ultra96 平台上进行了测试。测试结果表明，通过 DPU 对深度学习模型进行加速后，其应用能够在 ARM 处理器上流畅运行，我们的测试结果显示：1）在 Ultra96 上部署一个 DPU 核（B1152-1.3.0），大约占用了 FPGA 上一半的资源，所以在进一步优化的情况下可以考虑部署两个 DPU 核；2）ARM 上的人脸检测的应用大约占用了 90%的 CPU 时间和 20%的内存（总 512MB），在这个情况下能够稳定在 30FPS 左右，画面非常流畅；3）增加一路视频显示会导致帧率下降到 13FPS 左右，出现延迟、跳帧的情况，我们估计这主要是 ARM 的性能瓶颈导致的。

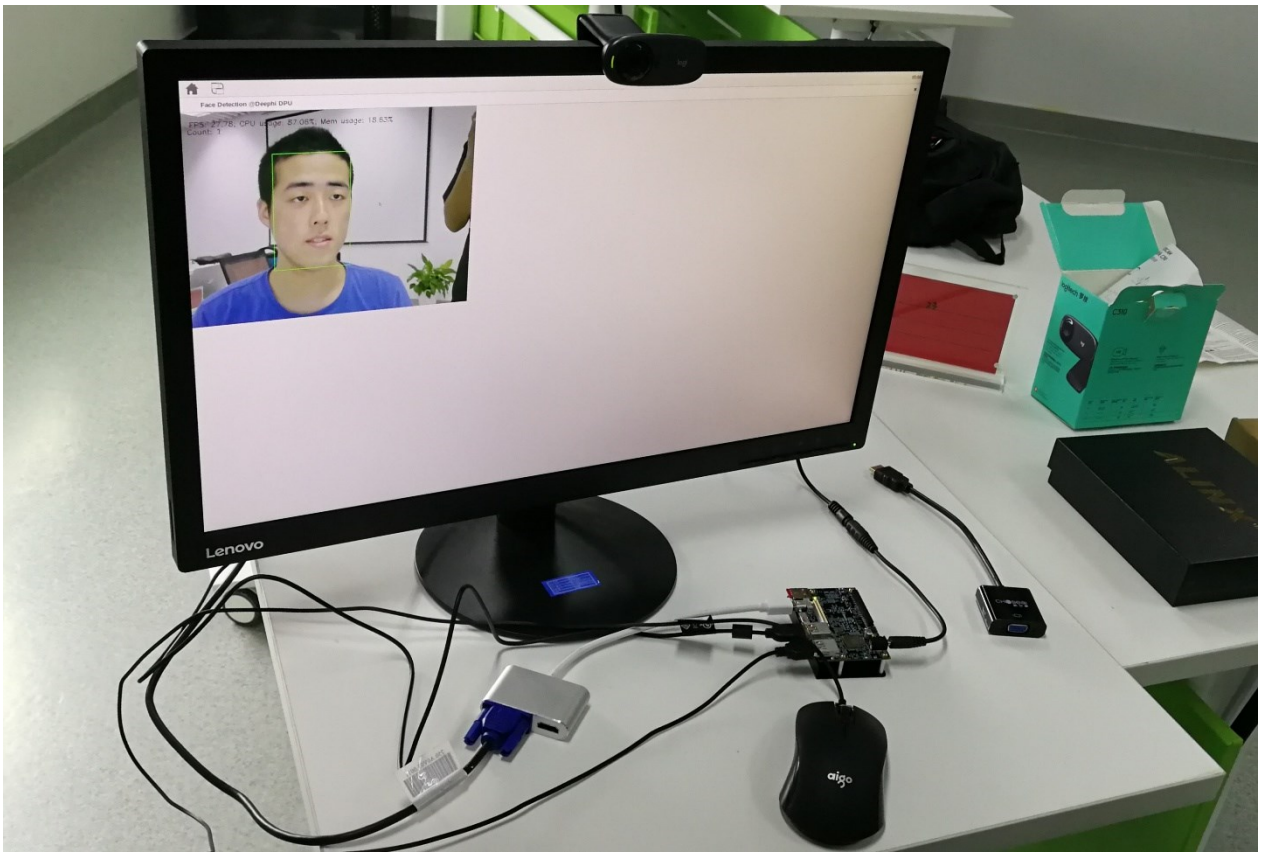


图. 8

图. 9

图. 10

第五部分

项目总结 /Conclusions

本小组在 Ultra96 板卡上成功实现人脸检测，算法基于 Resnet50 神经网络，板卡搭载 Xilinx ZYNQ UltraScale+ MPSoC，在 FPGA 上实现 DPU 加速神经网络计算，从而达到实时人脸检测，帧率稳定在 30fps 左右。由于简化的神经网络和 DPU 硬件结构，基于 ZYNQ 的人脸检测系统功率较低，配合精简的 Linux 内核，大部分 CPU 消耗和内存占用都集中在人脸检测程序上，不会造成过多的资源浪费。

在本次项目开发中，我们学习了 Xilinx 旗下的多种开发工具，了解到了 Linux 内核的配置编译过程，学习如何进行神经网络的剪枝，最终将训练好的神经网络部署在 Ultra96 板卡上。这种基于项目驱动型的教学方法，极大的提升了同学们的学习热情和学习效率，学生们能在短时间内掌握一种新的开发工具，积累了在实际工程开发中的经验。

第六部分

源代码/Source code

https://github.com/Ace-Pegasus/DPU

第七部分（技术总结/心得/笔记等分享）

我们基于 Vivado 调用 IP 核实现 DPU，在 Ubuntu 上完成 Linux 内核编译，生成 Linux 开发环境，并应用在 Vivado SDK 上，最终 FPGA 加速深度学习算法，减少功耗，提高硬件利用率。

在本次项目中，我们在实现人脸检测的同时还进行了初步的性能测试，从测试结果中我们认识到了基于 zynq 的 DPU 应用主要的瓶颈在于 ARM 核的性能，而 ARM 上的算力主要可能消耗在视频解码，视频输出上，由于 FPGA 上的资源比较充足，我们预计可以通过在 FPGA 上再添加一个视频解码核来解决，但是由于项目时间有限我们没有进行尝试。

同时小组成员学习了如何使用 Xilinx 的主流开发工具，对 Xilinx ZYNQ 有了较为深入的了解。从编译环境的搭建到 SDK 代码的编写，我们积累了工程经验，从零基础到迅速掌握一种开发工具，更能进一步提升我们的开发能力和快速学习的能力。