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	DNN 加速器硬件实现与优化
组员姓名	**谭祖雄** **崔傲** **张伟枫** **赵天翼**
房间号及桌号	717 第 18 桌

第一部分

小组成员分工

（各成员任务分配）

	姓名	任务分配
组长	谭祖雄	制订项目计划，完成项目中用到的有关 Vivado SDK 和 HLS 部分的编译工作，撰写设计文档
组员 1	崔傲	学习并优化算法，并成功将算法在 Ultra96 上运行，撰写设计文档
组员 2	张伟枫	协助优化算法，并完成所需环境的配置工作
组员 3	赵天翼	协助优化算法，并完成项目汇报 PPT 制作

第二部分

设计概述 /Design Introduction

（请简要描述一下你的设计：1. 功能说明；2. 所用到的设备清单）

自从 20 世纪 70 年代至今，人脸识别的研究工作一直是图像分析领域最热门的研究内容之一。人脸识别从最开始的时候，需要人为的手工干预，识别方法也比较单一，精度不够高，逐步发展到目前能够进行机器自动化识别、识别方法丰富、精度比较高的阶段。研究人脸识别的目的和意义，一是提高人类视觉系统对事物的认识使；二是满足人工智能的应用。使用人脸识别技术来设计自动人脸识别系统，使用计算机实现对人脸的自动识别，有着很好的发展前景和广阔的发展空间。目前人脸识别的应用领域主要有：

1. 企业、住宅安全和管理；
2. 电子护照及身份证、公安、司法和刑侦、自助服务、信息安全。

此次实验用到的算法是基于 DNN（深度神经网络）实现的。现如今，DNN 应用已被广泛部署于云端和终端设备中，如人脸识别、语音识别（翻译）、产品推荐、物体检测等。这些应用需要大量计算与存储资源，以满足其高吞吐率、低能耗和低延时要求。可见，不论是云端还是终端计算，DNN 的推理过程都需要作加速处理才能适应日常使用需求。而早期的基于 DNN 模型的算法是在中央处理单元（Central Processing Unit，CPU）的计算平台上实现的。但是由于“冯·诺依曼瓶颈”，指令与数据存放在同一内存带来的 CPU 利用率降低，因此很难在 CPU 上实现并行化，这大大制约了计算速率。FPGA（可编程逻辑门阵列）作为可编程硬件，可提供比基于 CPU 或 GPU 解决方案更低的延时和能耗，也能提供比专用集成电路（ASIC）更高的灵活度和更短的产品上市周期，在加速 DNN 中表现非常出色。因此，FPGA 是非常理想的 DNN 加速平台。

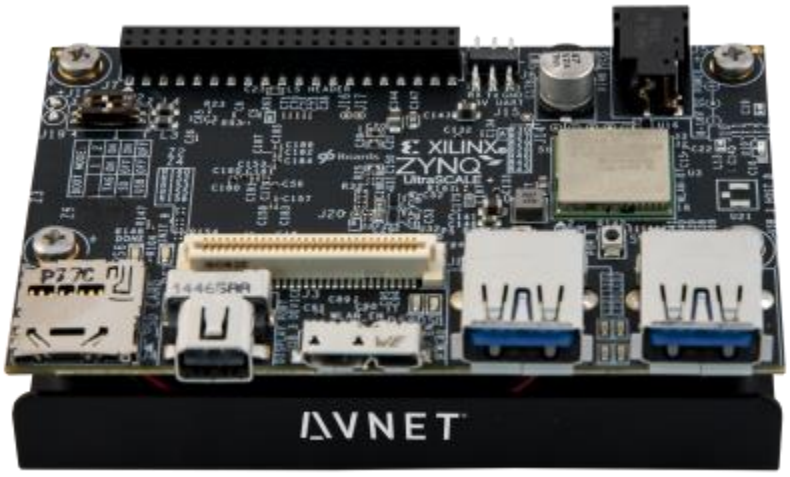

图 2-1　Ultra96 开发板

本次实验，我们主要在 Xilinx Ultra96 开发板上部署 DPU，用 DNNDK 修剪、量化、编译深度神经网络模型，并编写相关函数进行验证。相比于 CPU，GPU 等平台，在识别速度上有着显著的提升，同时极大的降低了功耗，大大拓宽了其实用性。

本实验所需要求如下

一、硬件要求：

1. Ultra 96 开发板
2. 12 直流电源
3. 显示器（hdmi 接口）
4. Mini DP 转 hdmi 接口转接线
5. 16G SD 卡
6. 摄像头

二、软件要求：

Vivado®设计套件 2018.2

Ultra96 v1 版本镜像

Xilinx SDK 2018.2

Petalinux 2018.2

第三部分

详细设计 /Detailed Design

（请详细说明你作品要实现的所有功能以及如何组建系统以实现该功能，还包括为实现该功能需要用到的所有参数和所有操作的详细说明，必要的地方多用图表形式表述）

如因文档篇幅限制无法详述，可提供附件。

我们的设计目的是在 Xilinx Ultra96 开发板上部署 DPU，用 DNNDK 修剪、量化、编译深度神经网络模型，并编写相关函数进行验证。

本作品的开发流程如下图：

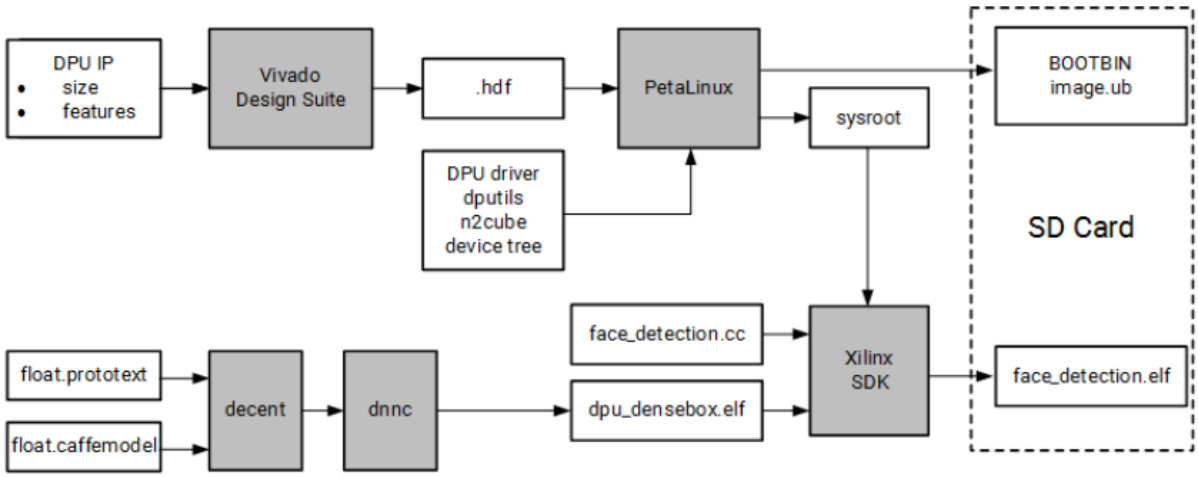

图 3-1 实验开发流程

1. 在 Vivado 设计套件中搭建硬件平台
2. 在 Petalinux 中搭建 Linux 操作系统
3. 使用 DNNDK 量化、编译神经网络
4. 在 Xilinx SDK 中生成机器学习应用程序
5. 在 Ultra96 开发板上运行并验证应用程序

DPU：

Xilinx®深度学习处理器单元（Deep Learning Processor Unit, DPU）是一个专用于卷积神经网络的可配置引擎。能够基于选择的硬件设备及应用配置其计算并行性。DPU 包含了一系列高效的优化指令，能够支持大部分卷积神经网络，如 VGG, ResNet, GoogleNet, YOLO, SSD, MobileNet, FPN 等等。

根据所选硬件资源的不同，可以选择不同规格的 DPU 核心，如 B1152, B2304, B4096 等，这关系到卷积单元的并行性。我们的作品运行在 Ultra96 开发板上，所以选用了一个 B1152 单元，具有 4*12*12 的并行性（4 Pixel Parallelism，8 Input Channel Parallelism (ICP)，8 Output Channel Parallelism (OCP)）。

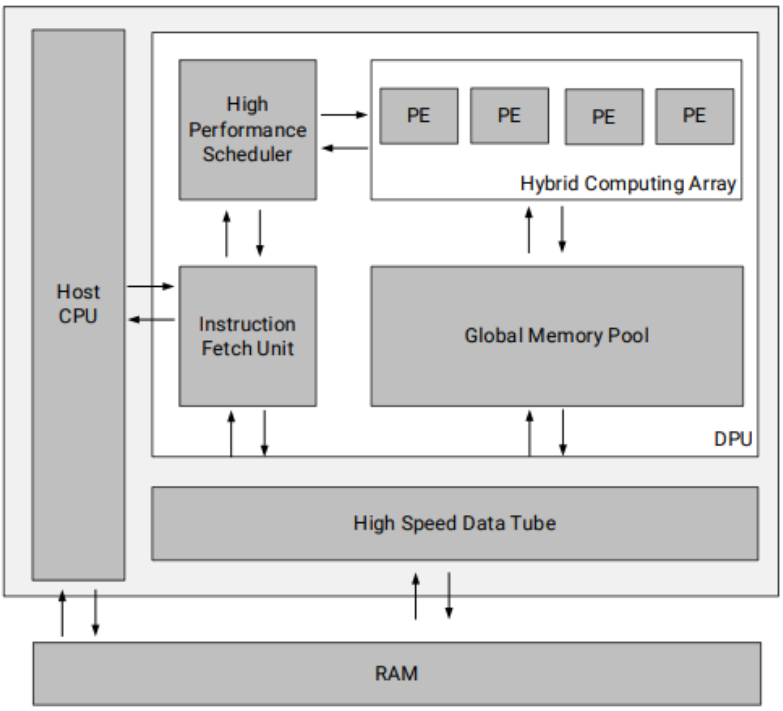

图 3-2　DPU 基本结构

Vivado 设计套件：

我们使用 Vivado 设计套件 2018.2 版本和 DPU IP 搭建硬件平台：

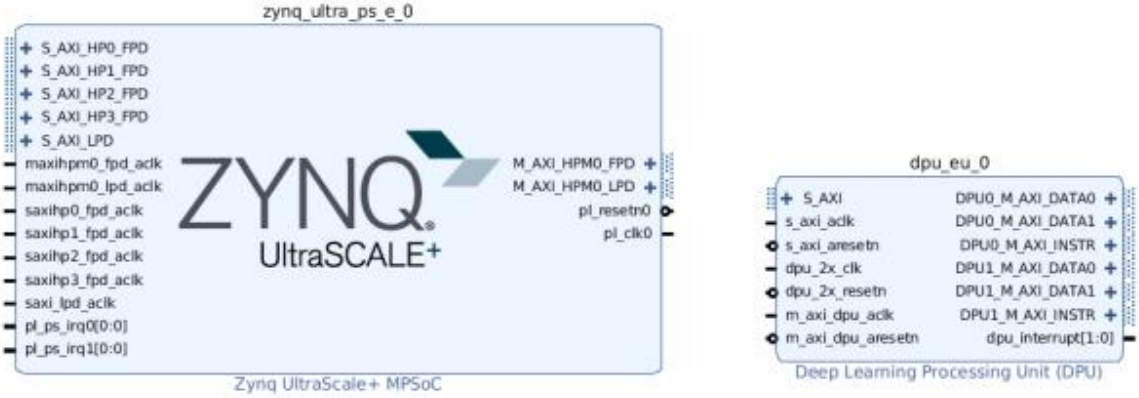

图 3-3　Vivado Block Design

1. 创建一个 Ultra96 的新工程。
2. 添加 DPU IP 核到工程中。
3. 使用 .tcl 脚本来恢复工程文件。
4. 设置 DPU 及其连接的配置。

5. 生产比特流文件
6. 导出.hdf 文件

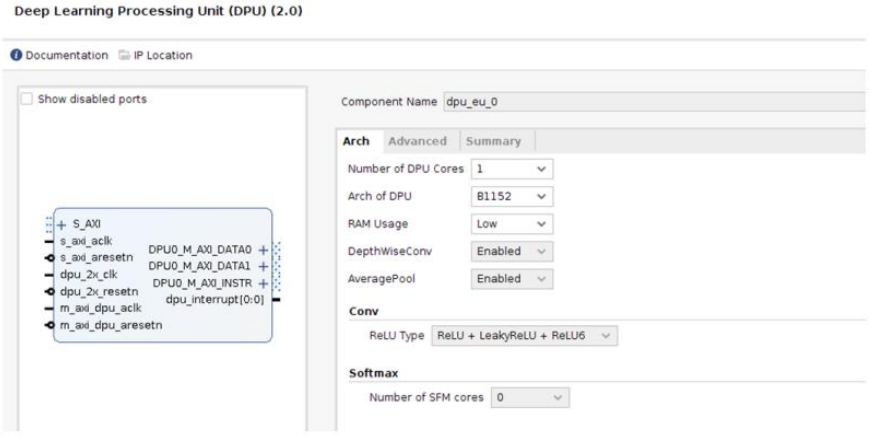

图 3-4　Vivado IP 核设计中 DPU 参数设定

Ultra96 开发板：

Ultra96™是基于 ARM 的 Xilinx Zynq UltraScale +™MPSoC 开发板，基于 Linaro 96Boards 规范，并且兼容 Zynq 7000 系列的软件生态。主要参数如下：

- SoC: Xilinx Zynq UltraScale+ MPSoC ZU3EG A484
- RAM: Micron 2 GB (512M x32) LPDDR4 内存
- Storage: 16 GB microSD card + adapter
- USB: 1x USB 3.0 Type Micro-B upstream port, 2x USB 3.0, 1x USB 2.0 Type A downstream ports
- Display: Mini DisplayPort

此外 Ultra96 还支持无线上网功能，支持 Linux 系统。

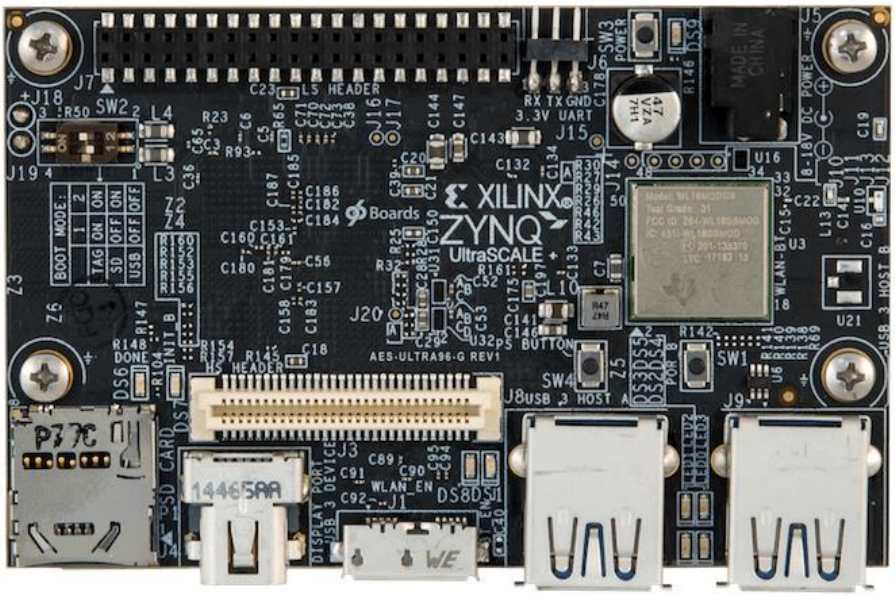

图 3-5　Ultra96 开发板基本结构

DNNDK：

深度神经网络开发套件(Deep Neural Network Development Kit, DNNDK) 提供了一套对神经网络进行剪枝、量化、编译、优化等功能的工具链，包括 DECENT (Deep Compression Tool), DNNC (Deep Neural Network Compiler), DNNAS (Deep Neural Network Assembler), N2Cube (Neural Network Runtime), DPU

Simulator 等部分组成。本作品主要使用了 DECENT 和 DNNC 两个模块对网络进行处理。

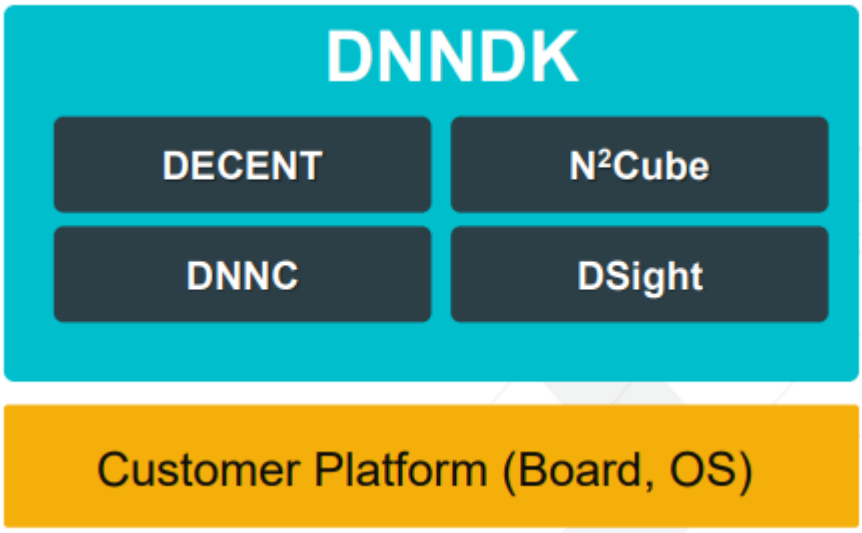

图 3-6　DNNDK 基本组成结构

DECENT 过程主要采用粗粒度修剪、训练量化和权重共享来减小网络的规模，在保证精度降低很小的前提下，实现高性能和高能效。

DNNC 通过平衡计算工作量和内存访问，将神经网络算法编译为 DPU 指令来最大化 DPU 资源的利用率。

基于 ResNet 的人脸识别：

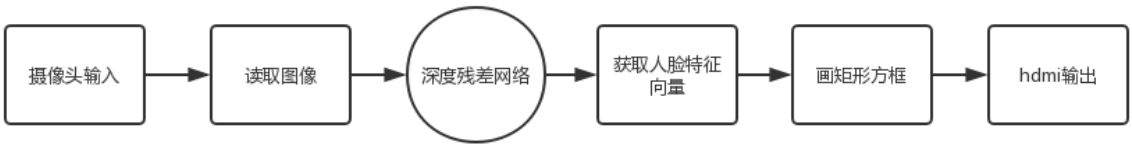

图 3-7　人脸识别主要运行流程

本实验中人脸识别算法用到的网络是深度残差网络（Deep residual network, ResNet），其参考了 VGG19 网络，并在此基础上进行了修改，并通过短路机制加入了残差单元。ResNet 通过残差学习解决了深度网络的退化问题，让我们可以训练出更深的网络。我们利用已经训练好的 ResNet 模型，经过 DNNDK 的优化处理之后，实现实时人脸识别功能。人脸检测算法需要用大小位置不同的窗口在图像中进行滑动，然后判断窗口中是否存在人脸。

第四部分

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

（作品完成情况，附照片/视频链接）

完成情况：

1、在 Vivado 中配置了 DPU IP，导出.hdf 文件。

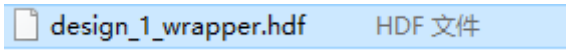

图 4-1　.hdf 文件

2、在 Petalinux 中生成 Linux 系统的启动文件 BOOT.BIN 和 image.ub，以及 SYSROOT 文件。

图 4-2　.bin 以及.ub 文件

3、使用 DNNDK 对 ResNet 模型进行量化和编译，生成.elf 文件，编写相关函数验证其功能，并在 SDK 中生成最终的可执行.elf 文件。

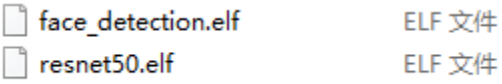

图 4-3　可执行.elf 文件

4、将上述文件拷贝至 SD 卡，启动系统，连接上摄像头和显示器，观察其功能并记录。实验结果如下图所示。其中方框代表识别出的人脸，左上角为实时帧率值显示，识别帧率大概在 70~80 帧左右。

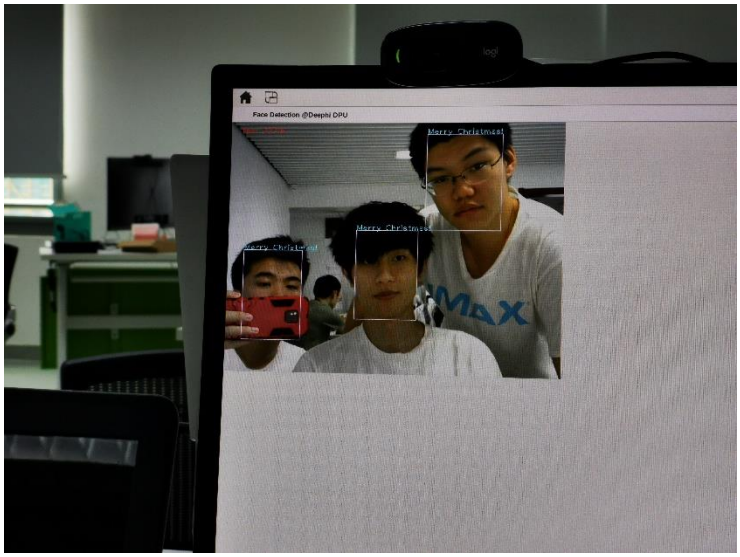

图 4-4　人脸识别最终结果

第五部分

项目总结 /Conclusions

(项目中所用到的知识点，项目收获及心得)

本次项目用到的是基于 Xilinx 的 edge AI solution 对边缘的神经网络进行处理与加速。主要工作是在 Ultra96 片上通过 CPU+DPU 的方式实现了人脸识别功能，并添加了帧率显示，同时对比了仅 CPU 时和 CPU+DPU 时工作效率的差异。相比与仅用 Ultra96 片上的 ARM A53 运行该算法，DPU+CPU 处理时运行速率提升了 30~50 倍。本次实验用到的网络模型是深度残差网络，其特点是容易优化，并且能够通过增加相当的深度来提高准确率。其内部的残差块使用了跳跃连接，缓解了在深度神经网络中增加深度带来的梯度消失问题。

目前，虽然本实验虽然成功运行了算法，并比较准确的识别出来了人脸，但是此算法还存在的很大

的提升的空间。在人脸检测上，无法支持多视角的检测，稍微转动头的角度就会造成无法识别。同时，在识别速率上，未来也可以做进一步的提升，以增强实用性。

在本实验基础之上，我们还有新的改进方案，还需要更多的时间来验证。在人脸检测方面，多视角人脸检测可以作为未来的研究重点，增强识别的容宽度。因此，在接下来的模型修改中，需要加入更多视角的人脸图像进行识别。在识别速率方面，可以通过对深度残差网络进行进一步优化、剪枝、压缩，或者在 DPU 中加入乒乓 RAM 加速数据读写等方式来提升。

第六部分

源代码/Source code

（作品源码 Github 链接，github 操作步骤见 *Github 项目上传简单教程*）

第七部分（技术总结/心得/笔记等分享）

（可分享的技术文档附件或已发表的文章链接）

本次项目的原计划是在实现基于 DPU 的深度神经网络加速器的基础上，对算法可以优化的模块进行分析，找到可以优化的地方进行优化加速。在项目动手的过程中，由于没有接触过相关的项目开发经历，在根据实验教程进行 DPU 部署的过程中遇到许多实验环境的配置问题，在这些问题上耗费了大量的时间。不过在 Xilinx 技术人员的帮助和指导下，以及和其他小组的互相帮助和讨论下，我们成功的解决了项目环境配置的问题，并在 ultra96 上成功的实现了人脸检测的识别。目前实验中主要存在的问题是实验环境的配置，在短短的 6 天实验时间里，配置环境的时间就多达 4 天。主要出现这个问题的原因还是由于经验不足导致的，4 个小组成员都没有类似项目的开发经历。但是，在这个期间，我们也学习到了很多，积累了一些经验，希望能对以后的项目开发有一定的帮助。

随后我们对实现的人脸检测进行了优化分析，由于时间关系，我们对于算法的了解还不够深刻，因此没能对算法做出显著性的调整与优化，仅仅开发了一些简单的功能。如果后续有时间和精力可以和小组的其他人员一起再进行深入的优化分析，达到算法加速的目的。

本次项目最大的感触，就是累。我们小组 4 名成员，1 名研究生，2 名学生本科刚刚毕业，还有一个即将升入大三的本科生，两个学集成电路，两个学计算机。作为组长，我本来以为我们的搭配十分合理，结果在真正上手做项目的时候才发现，我把问题想的过于简单了。由于缺乏神经网络训练开发相关的经验，以及 linux 系统的使用经验，在做项目的前期遇到了极大的阻力。有时候可能白天忙了一整天才解决一个 Petalinux 的环境配置，晚上经常需要加班到深夜才能完成当日的任务。这几天，我们小组 4 人睡眠都极其不足，而且不断出现的 bug 也把我们的心情推入谷底。不过很庆幸，我们最终挺过来了，虽然成果和其他博士带队的小组没法比，但是我们自己在此之中收获的，是之前从来没有过的。

本次项目是小组 4 个成员一起合作完成的，大家在这几天的时间里一起合作一起解决难题，对我们来说是一段宝贵的经历。同时，十分感谢 Xilinx 提供这次免费的培训机会与资源，同时也十分感谢技术人员和其他小组成员的帮助与指导。如果未来有机会，还希望继续参与到此活动中来。