基于 FPGA 的足球赛事分析系统

摘要

随着江苏省城市足球联赛（“苏超”）在社会关注度和竞技水平方面的迅速提升，赛事对实时战术分析与数据支持系统的技术需求日益迫切。为满足专业化、数据化和智能化发展的趋势，本文设计并实现了一套基于 Xilinx ZYNQ-7020 异构平台的足球赛事分析系统。该系统融合 FPGA 的高速并行处理能力与 ARM Cortex-A9 处理器的灵活控制特性，实现了赛事视频的实时采集、处理、分析与结果可视化展示，具备高效性、实时性与可扩展性。

系统在硬件层面采用 OV5640 摄像头实现视频采集，结合 Verilog 开发的逻辑模块与 Vivado 工具链进行系统搭建。通过软硬件协同设计，将计算密集型的图像预处理与目标检测任务部署于 FPGA 逻辑端，而将策略分析与进球预测等复杂算法任务由 ARM 端处理，从而优化资源分配，提升系统响应速度。软件部分采用 YOLOv8 轻量级目标检测算法进行模型剪枝与 8 位定点量化，使其更适配于嵌入式平台，在保证高检测精度的同时显著降低运行延迟。配套的 PyQt5 用户界面则实现了检测结果的可视化、事件回放与数据导出功能，增强了系统的可用性与交互体验。

在功能实现方面，系统支持足球与球员的精准识别与跟踪，具备进球事件检测、战术态势分析及走势预测等核心功能，并构建了雷达模式以俯视图形式实时展示球场态势。实验表明，系统视频识别速率达 10fps，单帧延迟小于 100ms，动作识别准确率超过 85%。此外，通过 YOLOv8 剪枝与量化优化后，模型参数量减少约 69%，计算量降低近 70%，运行速度提升近 4000 倍。

本系统具备软硬协同加速、模型轻量化、模块化设计及智能交互界面等多项创新特性，能够广泛应用于职业赛事分析、教练战术辅助与媒体数据增强报道等场景，是一套高效、实用的嵌入式足球赛事分析解决方案，对推动足球赛事信息化与智能化具有积极意义。

第一部分　作品概述

1.1 功能与特性

本系统是一款基于 ZYNQ-7020 开发板的嵌入式足球赛事分析平台，聚焦于

2025 年江苏省城市足球联赛（“苏超”）的应用场景。随着苏超赛事在江苏省内热度不断攀升，联赛的专业化程度与观众关注度大幅提升，对赛事数据分析、战术辅助决策等技术手段的需求亦日益迫切。为此，本系统结合 FPGA 与 ARM 异构计算平台的优势，打造了一套具备高效、实时、可扩展特点的足球赛事分析解决方案。

其主要功能为：

①赛事数据实时处理：系统能够实时接收和处理赛场视频，识别关键事件（如进球、射门等），并提取球员位置和动作信息。

②进球预测：利用算法模型预测进攻路径和潜在进球机会，辅助教练组进行战术调整。

③战术分析：系统通过分析球队的站位、传球线路等数据，帮助分析球队战术的有效性，提升教练员的战术决策。

1.2 应用领域

本系统的应用领域源于 2025 年江苏省城市足球联赛（“苏超”）这半年来席卷江苏的体育盛事。苏超已不仅仅是一场足球赛事，它成为了全民热议的社会热点，吸引了众多球迷与媒体的关注，成为了全省十三市荣誉与足球文化的交汇点。

随着苏超赛事的火爆，它早已超越了传统体育赛事的框架。不管是“南通狼”的不败战绩还是“浪里马”的风浪越大我越浪，不管是“比赛第一，友谊第十四”的宣传口号还是“天时地利人不和”的调侃话语，都成为球迷热情迸发的舞台。赛场内，战术博弈与球员技术的对决令人激动不已；赛场外，球迷们的欢呼与对战队的支持，充分展示了这项赛事的独特魅力。从单场六万观众的热烈呼声，到社交媒体上“散装江苏”文化的流行，苏超带来了前所未有的关注度与话题性。上座人数如图 1 所示，精彩瞬间如图 2 所示。

图 1 苏超上座人数

(a)球员头球防守

(b)球员脚下生风

(c)球员倒挂金钩

图 2 苏超精彩瞬间

然而，随着赛事关注度的提升，传统的赛事分析手段已难以满足球迷、媒体和俱乐部日益增长的需求。球迷们希望更深入地了解比赛背后的战术逻辑，而不仅仅是进球的瞬间；媒体需要及时、直观且具有冲击力的可视化数据来增强赛事报道的深度和影响力；教练组则渴望通过系统化的数据分析，快速捕捉到对手的战术弱点并优化自身策略。

在这样的背景下，一款能够提供实时、深度分析的系统不再是可选项，而是赛事运营和观赛体验中的必要工具。基于 FPGA 的足球赛事分析系统应运而生，旨在为每场比赛提供精确的实时数据支持，将场上的每一次战术变化、每一粒进球背后的策略因素，转化为直观且易于理解的数据。该系统不仅是对赛事技术手段的提升，更是对球迷热情与赛事深度需求的积极回应，进一步推动了苏超赛事的专业化与数据化进程。

1.3 主要技术特点

本系统基于 Xilinx ZYNQ-7020 平台，充分发挥其“可编程逻辑 + 双核 ARM 处理器”的异构架构优势，实现了高性能与低功耗的深度集成。

系统采用软硬协同设计方案，将实时性要求高的图像预处理与目标检测任务部署于 FPGA 逻辑单元中，提升视频数据处理效率；而复杂的战术分析、预测算法则运行在 ARM 端，通过嵌入式系统实现高度灵活的控制与管理。

在算法层面，系统集成了轻量化的目标跟踪、动作识别和事件检测模型，能够识别球员位置、球体轨迹及关键比赛动作，为进球预测与战术分析提供基础数据支持。图像输入可通过摄像头模块接入，具备良好的兼容性。

此外，系统采用模块化设计，支持功能扩展与算法升级，适应不同比赛场景与分析深度的需求。结合 PetaLinux 与 Vivado 工具链进行开发调试，实现从硬件逻辑设计到软件系统构建的全流程闭环，提升开发效率与系统稳定性。

该系统在保持实时性与嵌入式运行能力的同时，兼具可扩展性与实用性，是面向赛事现场应用的高效数据分析解决方案。

1.4 主要性能指标

具体性能指标如表 1 所示。

表 1 性能指标及其说明

序号	指标类别	具体指标	性能说明
1	识别速率	视频帧识别速率≥ 10fps	系统支持 10 帧每秒的视频处理。
2	检测延迟	单帧识别延迟≤ 100ms	系统快速处理每帧图像。
3	识别精度	动作识别准确率≥ 85%	基于优化的 YOLOv8 模型，保证较高识别精度。
4	UI 界面响应	操作响应时间≤ 100ms	界面反应灵敏
5	界面满意度	用户满意度≥ 85%	用户反馈良好，界面直观、操作简便。

1.5 主要创新点

①软硬协同加速：采用 Xilinx ZYNQ-7020 平台，通过软硬件协同设计，将高负载任务分配到 FPGA 进行加速，提升图像处理效率并减少延迟。

②轻量化目标检测模型：结合深度学习与传统图像处理技术，优化了模型结

构，降低计算资源需求，在保证精度的同时提高运行效率，适配嵌入式设备。

③模块化系统设计：系统采用模块化架构，支持后期功能扩展与算法更新，具有较高的可维护性与灵活性。

④智能 UI 设计：界面简洁直观，用户交互流畅，并根据用户反馈优化设计，提升操作便捷性与使用体验。

1.6 设计流程

如图 3 所示，本图展示了基于 FPGA 的足球赛事分析系统的整体架构与处理流程，系统依托 ZYNQ-7020 平台展开软硬件协同设计，实现了从图像采集、信息处理到智能分析及结果输出的完整功能链。

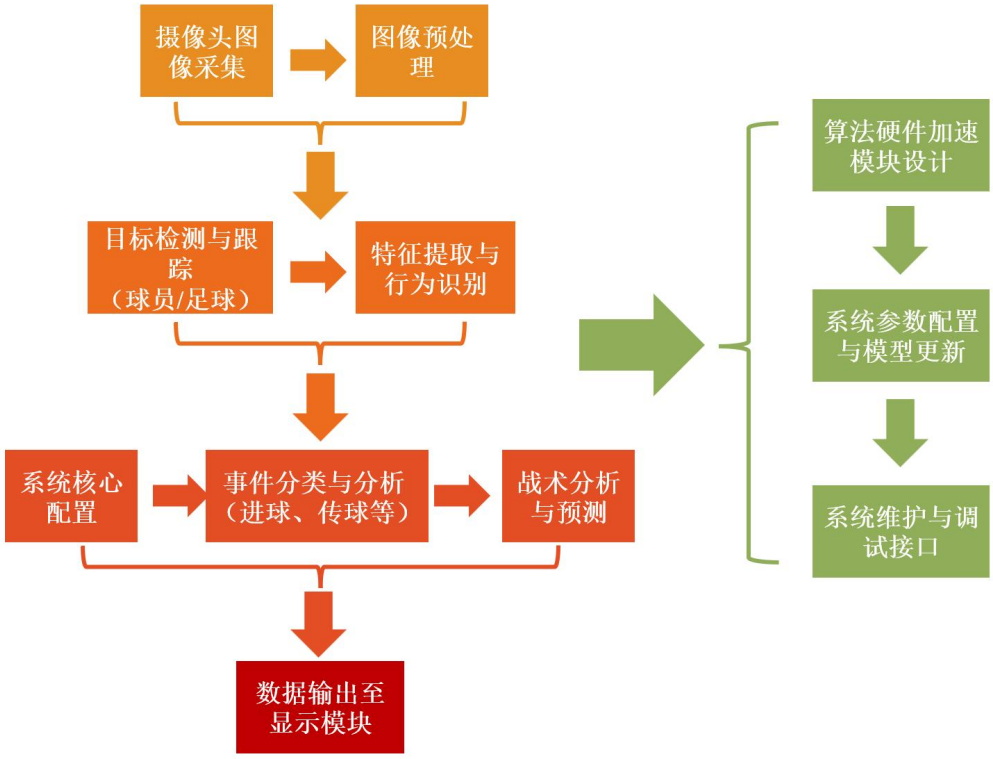

图 3 系统设计流程

左侧流程为系统主数据流，首先通过摄像头采集比赛图像，随后进行图像预处理，提取出可用于分析的视觉信息。接着，系统通过目标检测与跟踪算法识别并锁定球员与足球的位置变化，并结合行为识别方法对运动特征进行提取，以识别出典型动作行为。进一步地，系统对上述行为数据进行事件分类与分析（如进球、传球等），并在此基础上进行战术分析与走势预测，最终将分析结果输出至显示模块供战术辅助决策使用。

右侧为系统所依托的 FPGA 支撑模块，通过硬件加速设计提升了关键算法的实时处理能力，同时结合系统参数配置与模型更新机制，实现系统运行的灵活性与适应性，并通过维护与调试接口保障系统的稳定性与可扩展性。整体架构体现了高效、可配置、可扩展的设计理念，适用于对赛事进行实时、精准、深入的分析需求。

第二部分　系统组成及功能说明

2.1 整体介绍

在硬件端（PL 部分），使用 Vivado 和 ISE 14.7 设计并构建了系统逻辑。通过将 OV5640 摄像头连接到 ZYNQ-7020 的 USB HOST 接口，实现了视频数据的实时输入。在 PS 部分的配置下，系统采用加速策略执行 7 层神经网络的前向计算，并设计了基于原生 Verilog 的计算模型。通过移位寄存器结构进行加速，同时结合通道分组实现并行化计算，显著提升了处理速度和计算效率。

在软件端（PS 部分），首先对收集到的足球视频数据集进行数据预处理，包括数据归一化和无效值的清除等步骤。接着导入相关数据处理库，如 NumPy、matplotlib、pyplot 和 pandas。通过 NumPy 的数组生成函数将单张图像数据转换为适合后续处理的格式。在数据预处理后，将标定后的特征数据与分类标签关联，整理并汇总成适用于机器学习训练的数据集。将一段时间内的序列数据排列成数据矩阵，作为 DeepSORT 跟踪算法和 CNN 卷积神经网络的输入数据，以提高检测的准确性。

在显示与通信模块中，系统基于 ZYNQ-7020 的 RJ45 网口与管理端计算机实现高速网络通信，能够实时传输和记录检测到的信息。通过此架构，系统可以实现对足球比赛的实时检测分析。如图 4 所示，为系统框架图。

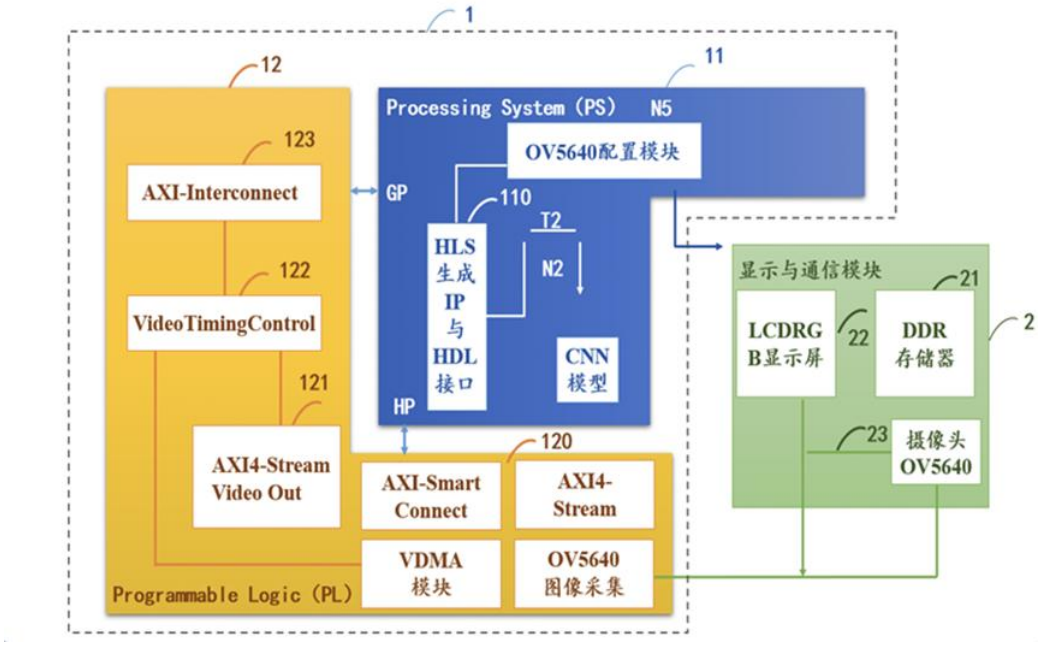

图 4 系统设计流程

2.2 硬件系统介绍

本系统硬件连接为基于 OV5640 的 ZYNQ-7020 硬件连接。

ZYNQ-7020 是一款基于 Xilinx ZYNQ XC7Z020 SoC 的 FPGA 开发板，集成了可编程逻辑（Programmable Logic, PL）和处理系统（Processing System, PS），在嵌入式开发中具有很强的灵活性和处理能力。ZYNQ-7020 的开发工具包括 Vivado IDE、Vitis 和 Vivado HLS，Vivado IDE 用于硬件开发，能够配置处理器、存储器、总线和互连等资源；Vitis 则为 ZYNQ-7020 提供了软件开发平台，支持对 Xilinx IP 的驱动开发，使用 C 和 C++进行程序编写。

ZYNQ-7020 的处理系统（PS）部分包含双核 ARM Cortex-A9 处理器，工作频率为 650MHz。在执行神经网络推理任务时，ARM 内核与专用卷积加速器协同工作。通过在 PL 部分嵌入卷积引擎和可编程软核，可减轻 ARM 处理器的负担，并允许卷积加速器在精细控制下运行，从而提高处理效率。

在可编程逻辑（PL）部分，ZYNQ-7020 包含 13300 个逻辑片，每个逻辑片中包括四个六输入查找表（LUT）和八个触发器。板载 630KB 的块 RAM 为数据处理提供了必要的存储支持，220 个 DSP 片用于实现高速的数值运算，满足卷积计算的需求。此外，四个时钟管理模块中集成了锁相环（PLL）和混合模式时钟管理器（MMCM），支持复杂的时钟设计要求。ZYNQ-7020 开发板还配备丰富的外设接口，包括以太网、HDMI 输入/输出、音频输入/输出、Arduino 和树莓派接口、多个 Pmod 接口、用户 LED、按钮和开关。兼容的 GPIO 接口和扩展端口支持灵活的外围设备连接，可以通过 HDMI 或 DSI 接口外接显示器，并通过网口进行 SSH 远程调试。

摄像头方面，OV5640 摄像头模块基于 OmniVision 的 1/4 英寸 QSXGA CMOS 传感器，支持自动对焦功能。该摄像头通过 24 针 FPC 软排线连接到主板，通过 I2C 接口与 FPGA 进行数据传输，以实现实时图像传输和处理。具体的连接方式如图 5 所示。

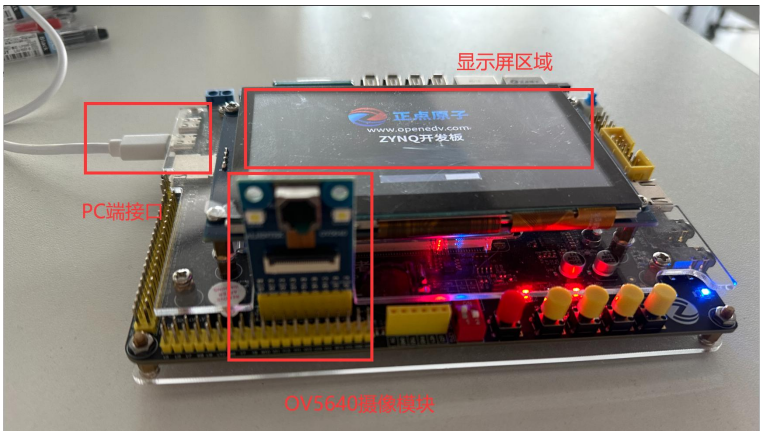

图 5 ZYNQ-7020 连接示意图

2.3 软件系统介绍

2.3.1 软件整体介绍

本系统的软件设计由两个核心模块组成：基于 YOLOv8 的足球系统实时分析模块与基于 PyQt5 的足球赛事分析 UI 交互界面。前者主要负责系统底层的数据处理与智能检测，实现对足球比赛中关键目标与事件的实时分析，是系统的算法核心；后者则承担上层的人机交互与结果可视化任务，用于展示检测过程与分析结果，为用户提供友好、直观的操作与反馈体验。两个模块通过高速数据通信与统一的接口协议协同工作，构成了一个从实时检测到可视化分析的完整软件体系，为足球赛事的智能识别与战术研究提供了技术支撑与应用基础。

2.3.2 软件各模块介绍

(1) 基于 YOLOv8 的足球系统实时分析模块

本系统采用 OV5640 摄像头获取实时视频数据，结合 ZYNQ-7020 开发板上部署的 YOLOv8 模型，实现对足球比赛中关键目标（如足球、球员、球门等）的实时检测与分析。

YOLOv8 是一种改进型的目标检测算法，基于无锚点（anchor-free）机制，相较于 YOLOv7 的 anchor-based 架构，YOLOv8 在检测框生成与目标定位上更加灵活与高效，能够在保持精度的同时显著降低计算复杂度，特别适合嵌入式 FPGA 环境下的部署与加速。其网络架构采用优化后的 C2f-CSPDarknet 作为主干网络（Backbone），通过引入跨阶段部分连接（Cross Stage Partial, CSP）结构与轻量化残差模块，实现了深层特征的高效提取与跨层信息融合；同时，结合

PAN-FPN（Path Aggregation Network - Feature Pyramid Network）特征融合结构，实现了多尺度特征的聚合，使模型能够精准识别不同大小和姿态的运动目标。

在系统的处理流程中，摄像头采集的实时图像首先输入至 YOLOv8 主干网络进行特征提取，随后由检测头预测目标的类别与位置边界框，实现对足球、球员等关键对象的识别与定位。检测结果进一步传递至上层战术分析与进球预测模块，用于识别比赛中的传球、射门及进球事件。该架构在 ZYNQ-7020 平台上充分利用 ARM 与 FPGA 的异构并行计算能力，在保证高检测精度的同时实现了低延迟与高帧率的实时分析性能，为足球赛事的智能识别与策略分析提供了高效、可靠的嵌入式解决方案。具体检测实物图如图 6 所示。

图 6 检测实物图

(2) 基于 PyQt5 的足球赛事分析 UI 交互界面

本系统的用户交互界面采用 PyQt5 设计，专为足球赛事分析与检测应用开发，旨在提供直观、高效的可视化数据展示与交互功能。界面通过与部署在 ZYNQ-7020 开发板上的 YOLOv8 检测系统实时通信，实现对赛场视频流的同步显示与结果反馈。当检测模块识别出足球、球员或进球等关键事件时，系统会在视频画面中以不同颜色的边框或标记高亮显示目标区域，同时在界面下方实时输出事件类别、时间戳及置信度信息，便于用户快速了解比赛动态。界面支持实时

刷新与数据记录功能，能够自动保存比赛分析日志，包括进球预测、战术特征及检测结果，方便后续的技术复盘与策略评估。

该 UI 系统具有良好的实时性与交互性，基于 FPGA 加速的 YOLOv8 推理框架，在低延迟条件下实现高帧率视频处理。界面设计简洁直观，提供“视频导入、开始分析、暂停检测、结果导出”等操作按钮，使用便捷，适合实验演示与现场应用。整体系统在保证检测准确性的同时，实现了人机界面的高效融合，为足球赛事的智能分析与可视化展示提供了友好、稳定的前端支持。UI 界面如图 7 所示。

图 7 基于 PyQt5 的 UI 交互界面

第三部分　完成情况及性能参数

3.1 整体介绍

图 8 整体结构图

3.2 工程成果

3.2.1　FPGA 硬件平台准备与开发环境搭建

我们使用 OV5640 高像素 CMOS 摄像头模块模拟足球赛场摄像头。ZYNQ-7020 FPGA 开发板通过 CSI 接口采集视频流，将实时视频数据输入到板上部署的 YOLOv8 神经网络模块中进行足球检测分析处理。当比赛开始时时，ZYNQ-7020 通过串口将检测结果传输至 PC 端，以便进行进一步的存储和处理。

（1）OV5640 摄像头

图像采集使用 OV5640 摄像头连接至 ZYNQ-7020 开发板的 CSI 接口，作为视频输入入口，使用 I2C 总线进行配置控制。OV5640 的实物图与结构图如图 9、图 10 所示。

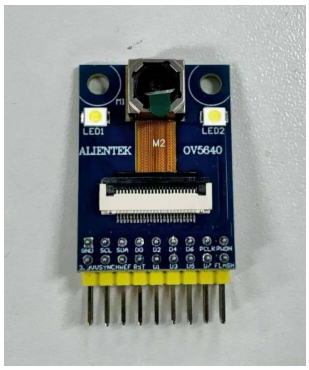

图 9 OV5640 摄像头实物图

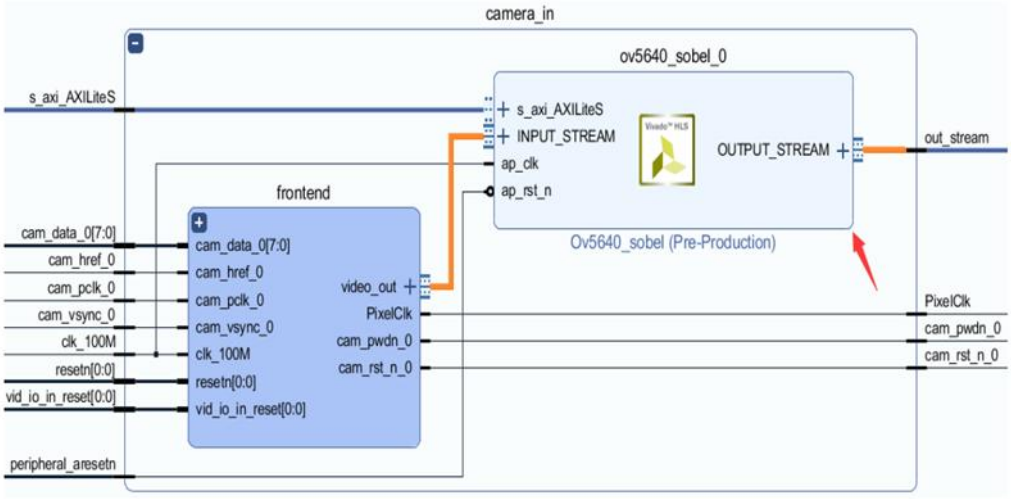

图 10 OV5640 结构图

OV5640 摄像头采集视频后，通过接口将数据传输到 ZYNQ-7020 的 ARM 处理系统部分进行处理和分析。针对复杂的测试环境，在 Overlay 中例化了一个 I2C 接口来对 OV5640 进行配置，通过改动 ov5640_config.py 文件的内容修改配置。

首先实例化 IIC：

```
iic = AxiIIC(ov5640_ol.ip_dict['cam_iic'])
# OV5640 器件地址
address = 0x3c
ov5640= OV5640(address, iic)
```

初始化 OV5640 并调用相关的方法配置 OV5640，如图 11 所示。

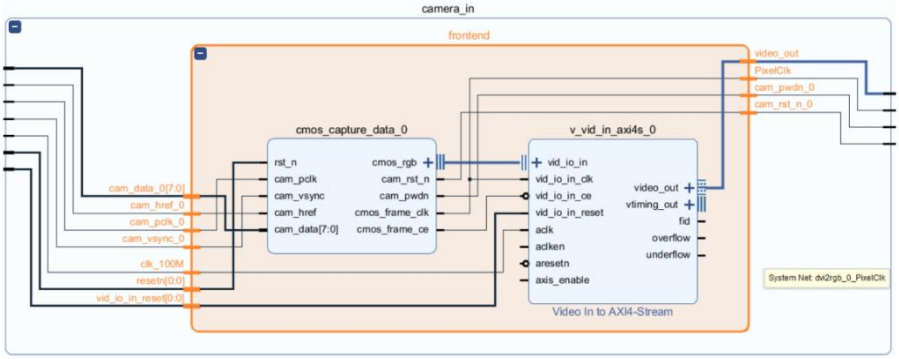

图 11 camera in 层次架构

（2） 基于正点原子 ZYNQ-7020 开发板的功能设计

正点原子 ZYNQ-7020 开发板是一款基于 Xilinx ZYNQ-7020 芯片的高性能 SoC（System on Chip）开发板，结合了 Python 编程和 FPGA 硬件设计。利用 ZYNQ-7020，开发者可以充分发挥 Xilinx ZYNQ All Programmable SoC (APSoC) 的功能，同时通过嵌入式系统实现灵活的硬件控制。

其中，ZYNQ 系列 SoC 的设计包括两个主要部分：PS（Processing System）和 PL（Programmable Logic）。PS 部分包含两个 ARMCortex-A9 核，可以运行 Linux 操作系统，并在操作系统上进一步运行 python 程序，如 YOLOv8 神经网络模块。借助 Linux 系统的灵活性，可利用 python 主程序来控制系统或传输数据。PL 部分就是 FPGA 的逻辑资源，开发者在 PL 中添加 IP 或者将自己用 C 或者 HDL 语言写好的模块封装成 IP，这些 IP 都被连接到 PS 端，一般都是通过 AXI 总线。通常，这些 IP 核通过 AXI 总线连接到 PS 端，允许 PS 和 PL 之间进行数据交换和控制。

在本项目的开发流程中，我们使用 Vivado 进行集成设计环境。每次开始一个新的涉及 PL 端的开发的时候，先在 Vivado 里面建一个工程，添加需要的各种 IP，然后以 ZYNQ 为核心连接，将设计的模块与 ZYNQ 的 PS 部分相连，通过 AXI 总线实现 PS 和 PL 之间的通信。完成设计并编译后，Vivado 会生成一个 bit 文件和一个 tcl 文件。bit 文件就是硬件设计，tcl 文件描述了接口关系。将这些文件复制到 ZYNQ-7020 的文件系统目录下，即可进行调用。

第一步，完成 ZYNQ-7020 板卡的打开：

首先，下载适用于 ZYNQ-7020 的嵌入式 Linux 镜像，并将其烧录到 SD 卡中。使用开源工具 Win32DiskImager 将镜像文件写入 SD 卡，如下图 12 所

示。

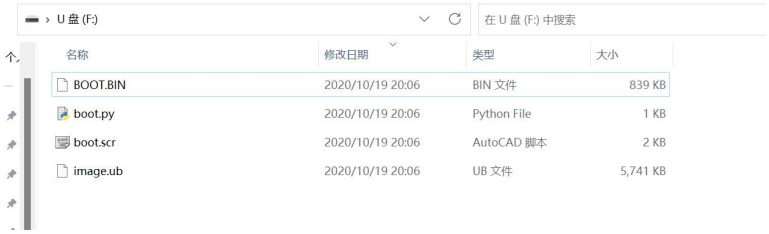

图 12 镜像文件烧录图

其次，将 SD 卡插入 ZYNQ-7020 开发板，以便在其上启动 Linux 操作系统。如图 13 所示。

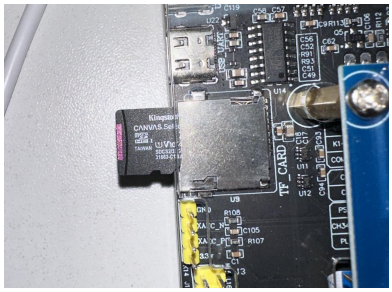

图 13 SD 卡接入图

之后用电源线插入 Vitis 凹槽上方的 TypeC 接口，注意按下电源开关，开发板正常供电启动。先是红灯亮，表示上电成功，大概一分钟后，蓝灯和黄灯闪烁，最后蓝灯灭，黄灯亮，表示板子工作正常，如图 14 所示。

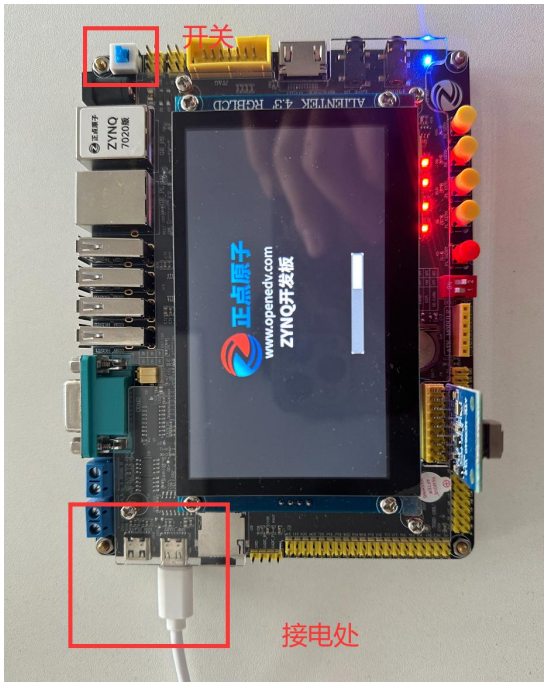

图 14 开发板供电图

最后，通过 samba 传输文件。

在开发过程中，需要在 PC 机与板卡之间传输一些较大的文件，可以利用 samba 协议将 ZYNQ 的文件系统当作一个网络硬盘直接读取。不过由于 ZYNQ-7020 开发板并不像 PYNQ 开发板那样自带 Samba 服务，需要手动安装 Samba 服务、配置共享文件夹、使用默认用户 xilinx 设置 Samba 用户。

在 Windows 中打开资源管理器，输入\pynq\xilinx 即可成功连接。在 Mac/Linux 中同样可以打开文件管理器，输入 smb: //pynq/xilinx 进行挂载。

第二步：Vivado 中 ZYNQ-7020 主体框架撰写与代码烧录：

启动 Vivado 2019.2，在 Quick Start 中创建项目，找到 ZYNQ-7020 并创建工作区。

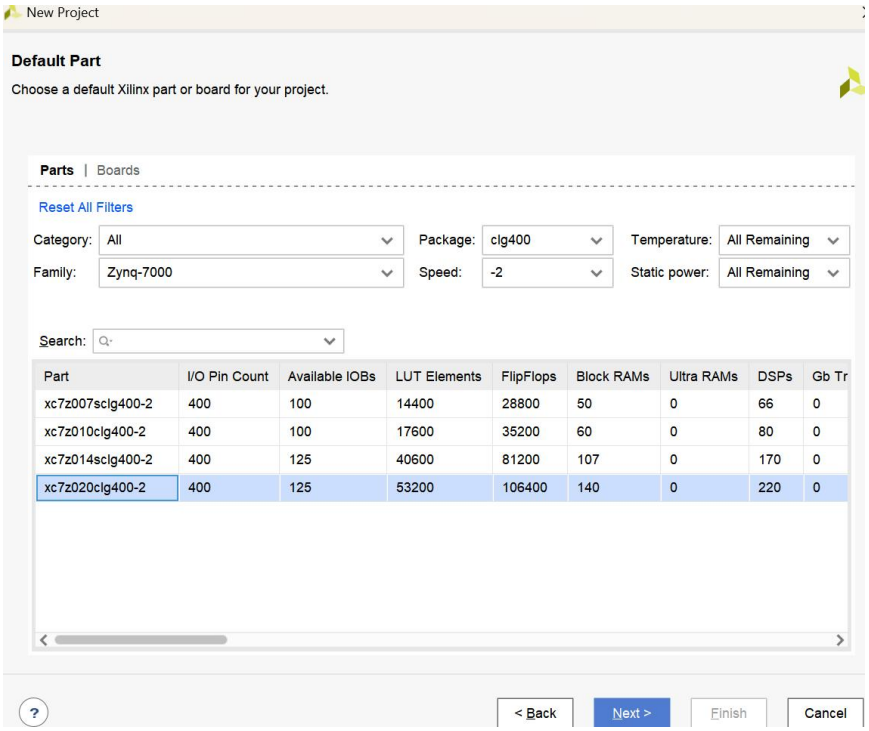

图 15 ZYNQ-7020 项目

在工程设置中，选择 “Create Block Design” 选项，进入 Block Design 环境。Vivado 会打开一个 Block Design 界面，搜索并添加包含 ARM 处理器的 “ZYNQ7 Processing System” IP 核。双击 PS 模块，Vivado 会自动配置 PS 端的默认参数，关闭无用通道，保留一个串口通讯口与 I2C 通讯口，因需要与 PL 部分连接，PS 端的 AXI 总线接口可以导出，用于与自定义 IP 或其他 AXI 外设连接，配置完毕如图 16 所示。

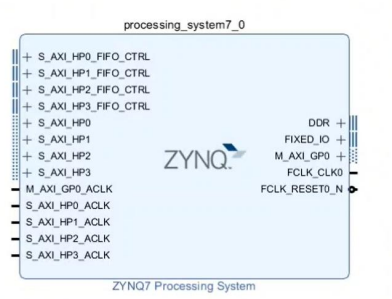

图 16“ZYNQ-7020 Processing System” IP 核

搜索并添加 conv 核，调整“Block Automation”与“Connection Automation”，优化布局，最终结果如图 17 所示。在完成连接后，选择 “Validate Design” 以检查设计中的错误或未连接的接口。Vivado 会给出提示，确保设计的完整性。确认并生成 HDL Wrapper，将 Block Design 转换为顶层的 HDL 文件，生成 Bitstream 后将硬件文件导出供后续综合和实现流程使用。

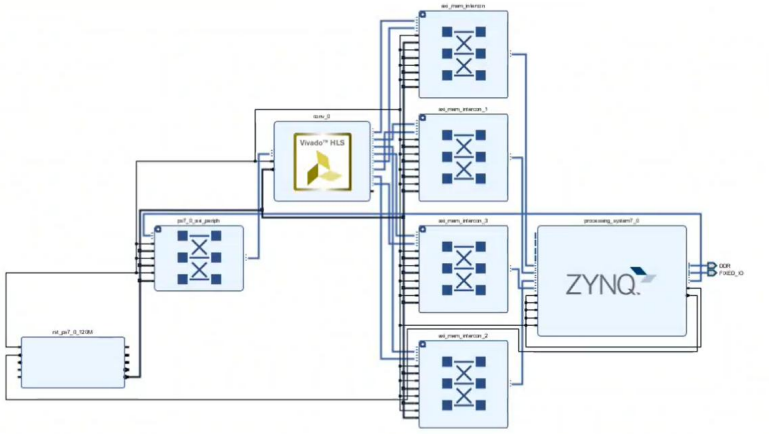

图 17 接口连接

之后进行 Vitis 开发：

①创建平台项目

file ->export -> export hardware。然后需要勾选 include bistream，点击 OK。

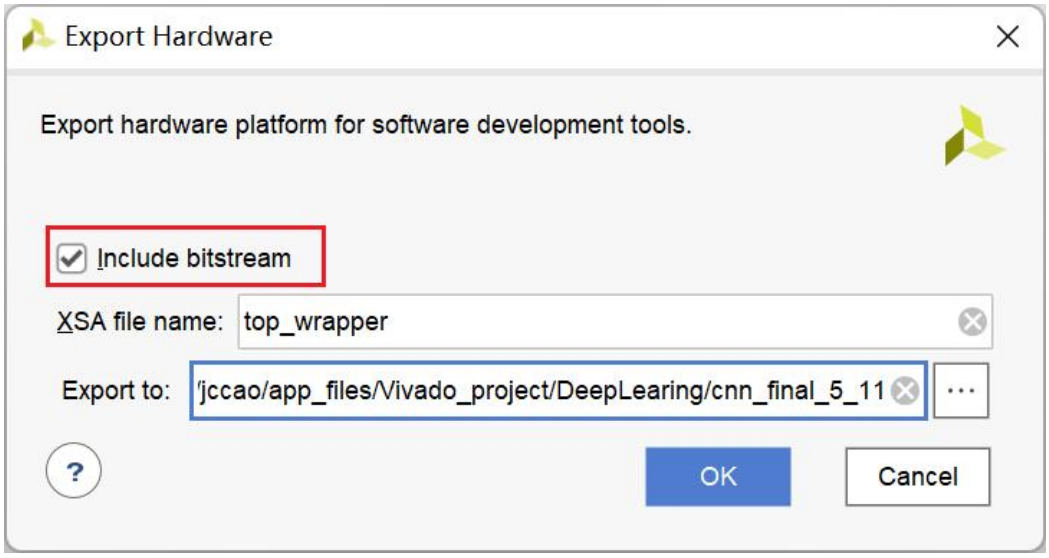

图 18 创建平台项目步骤一

生成的.xsa 文件位于工程目录下。

这里使用的Vivado版本为2019.2，从此版本开始，导出的硬件描述文件为.xsa文件，给 Vitis 平台使用。

选择工作目录后，点击 Lanch，点击 Create Platform Project 创建工程，如图19 所示：

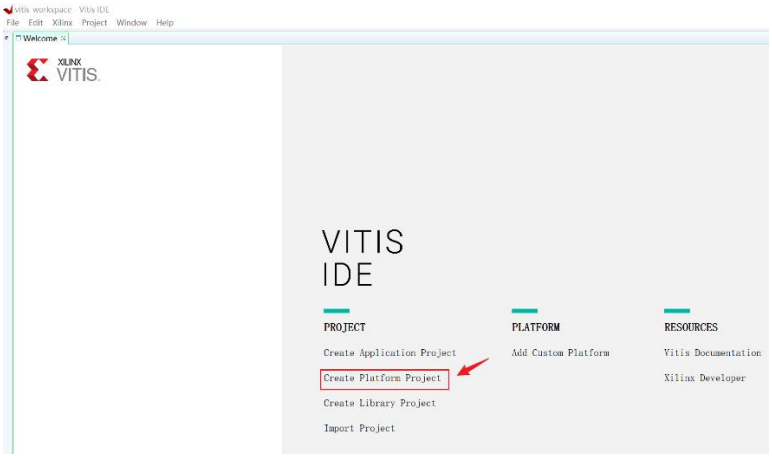

图 19 创建平台项目步骤二

输入工程名字，长度需在 3-40 个字符之间，指定刚刚生成的 .xsa 文件，点击 finish 即可。

②创建应用项目，进行编译工程，选中 APP 工程，右键“Build Project”按键，进行工程编译。

工程编译结束后，成功生成 elf 文件。至此，硬件和软件设计均已完成，如

图 20 所示。

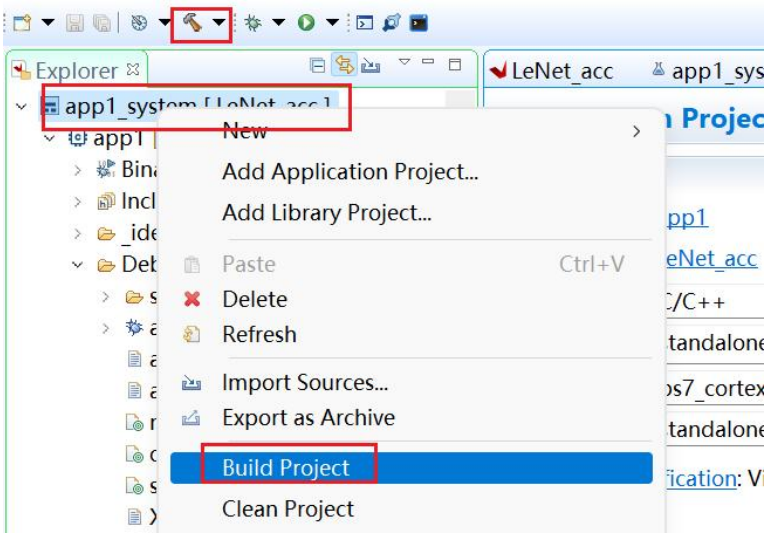

图 20 生成编译工程

第三步：下载验证并烧录程序

首先我们将 microusb 数据线与 ZYNQ-7020 开发板上的接口连接，数据线另外一端与电脑连接。

在菜单栏中依次点击“Window->Show view->Terminal 文件夹->Terminal”，最后点击“Open”，接口成功添加 Terminal 窗口。点击图标，进行串口设置界面，如图 21 所示：

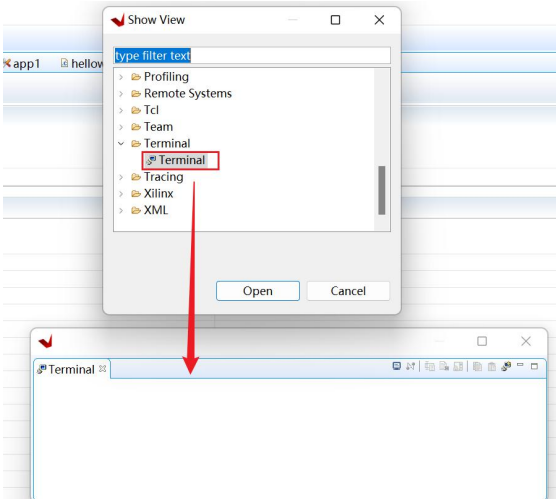

图 21 串口设置

其次，选择串口“Serial Terminal”，设置的参数需要与硬件设计过程中配置的 axi_uartlite_0 保持一致，即波特率为“115200”，数据位为 8 位，停止位为 1 位。点击“OK”后，如图 22 所示，证明串口连接成功。

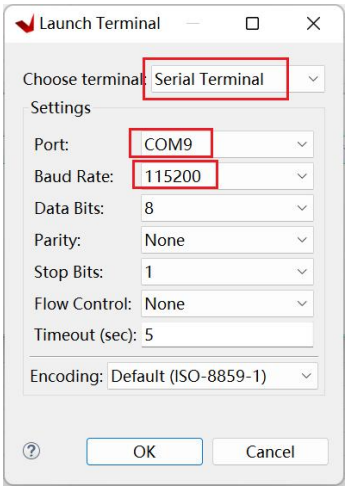

图 22 串口连接

最后，在 Run Configuration 页面点击“Single Application Debug(GDB) -> Debugger_app1-GDB”，点击菜单栏“Target Setup”。其中，“HardwarePlatform”为硬件平台， Bitstream File 为加载的 bit 流文件。勾选“Reset entire system”（系统复位）和“ProgramFPGA”（下载 FPGA）然后点击“Run”开始下载程序，如图 23 所示。

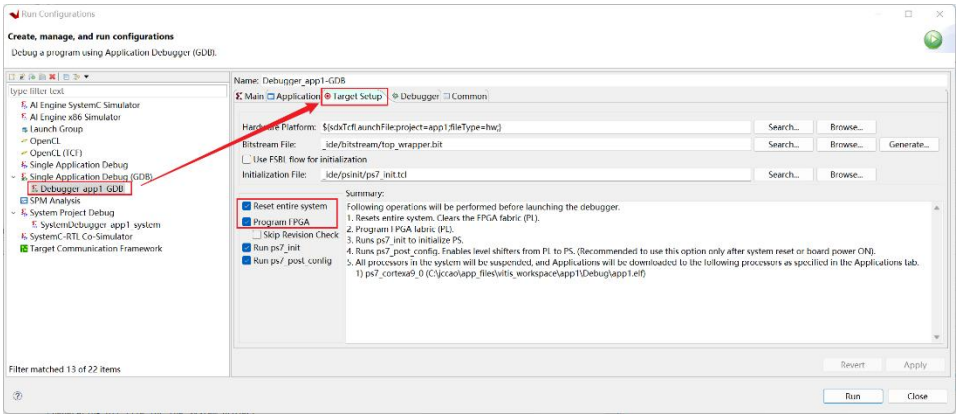

图 23 下载程序图

最后，将文件按照以上方法烧录进板卡。

第四步：效果验证

文件烧录进板卡之后，打开 pycharm，UI 界面正常运行，如图 24 所示。

图 24 足球分析系统识别图

在 Vivado 2019.2 中完成硬件系统的综合（Synthesis）与实现（Implementation）后，整个系统（包含 YOLOv8 加速器 IP 核、AXI 互联以及其他外设控制逻辑）在 ZYNQ-7020 (xc7z020)芯片 PL 端的资源利用率如表 2 所示。

表 2 ZYNQ-7020 PL 端资源利用率报告

资源 (Resource)	可用 (Available)	已使用 (Used)	利用率 (Utilization %)
LUT(查找表)	53200	37772	71.0%
FF(触发器)	106400	39368	37.0%
BRAM(块内存)	140	112	80.0%
DSP(数字信号处理)	220	198	90.0%

从表 2 数据可以看出，本设计是一个计算密集型和内存密集型应用。DSP 利用率达到 90%，表明我们设计的 Verilog 并行卷积引擎充分利用了 ZYNQ-7020 的硬件乘法器资源。BRAM 利用率达到 80%，主要用于片上缓存权重参数和中间特征图，这是实现高性能、低延迟数据流水的关键。整体资源占用率较高，说明本设计已接近该芯片的性能极限，是一个高度优化的 FPGA 加速方案。

（3）基于轻量级 YOLOv8 网络的 IP 核加速器

在 Vivado 2019.2 中使用 Block Design 硬件设计平台进行架构设计。先添加 Zynq IP 核并配置对应参数，如时钟频率、复位信号、DDR、外围 IP 引脚、PS-PL

交互等，再加入由 HLS 导入的 YOLOv8 IP 核，将其与 Zynq IP 核进行连接。连接使用了 Xilinx 官方提供的 AXI SmartConnect IP 核，ZYNQ 系统通过该 IP 核进行 PL 端通信和数据交互，将 S_AXI_HP 端口与 m_axi_DATA_BUS 端口互通，即建立与 YOLOv8 IP 核的连接。最终硬件搭建完成。

具体过程如下面系列图所示。

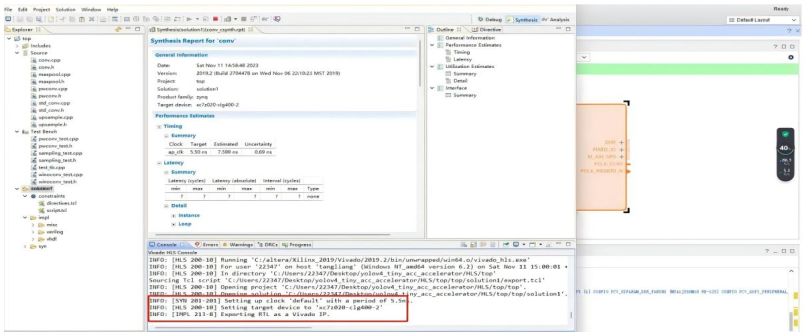

图 25 RTL 文件生成

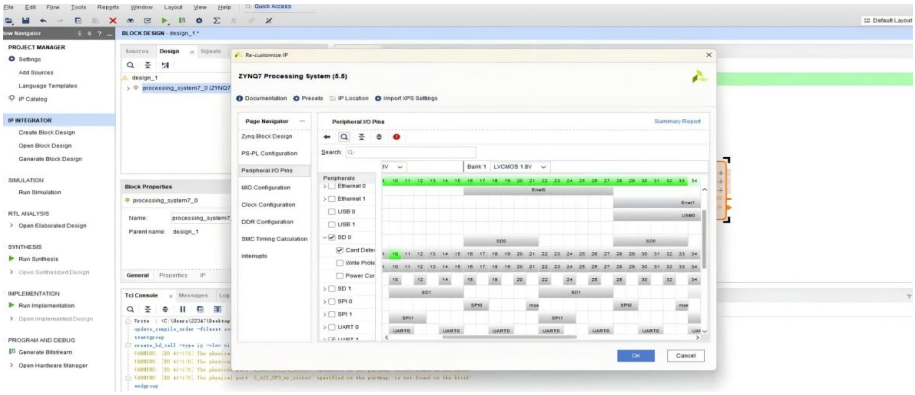

图 26 外设配置

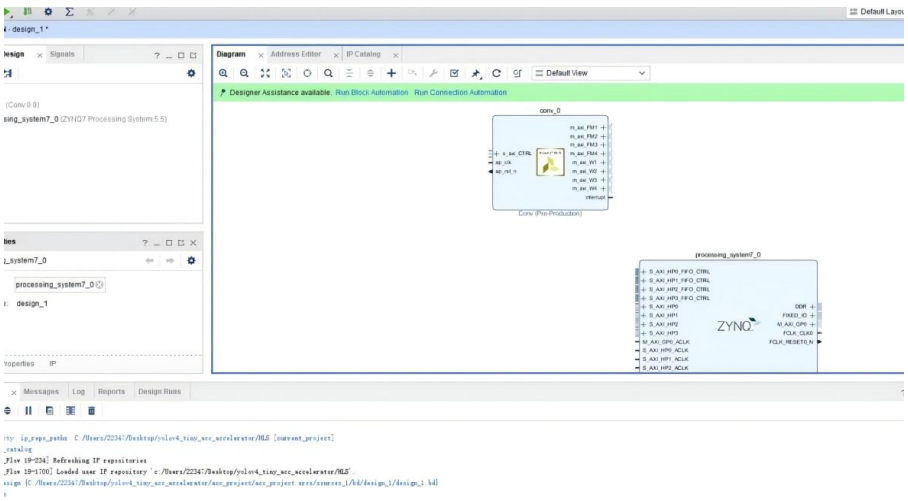

图 27 YOLOv8 加速核与 zynq 硬核连接

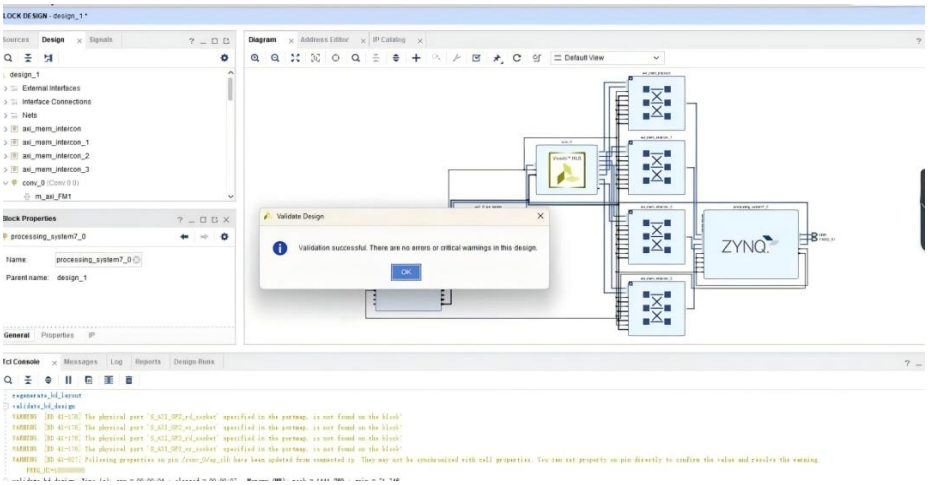

图 28 硬件结构布线无误

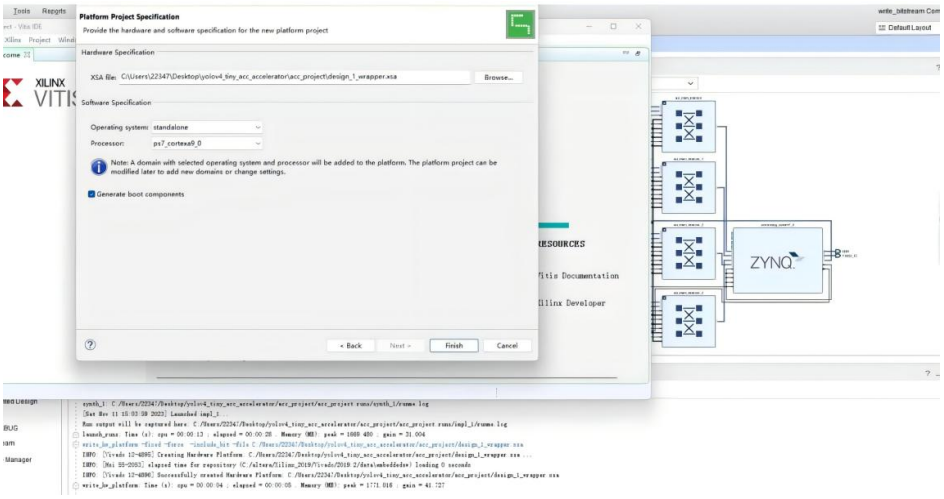

图 29 基于硬件设计平台的 xsa 文件新建 VITIS 工程

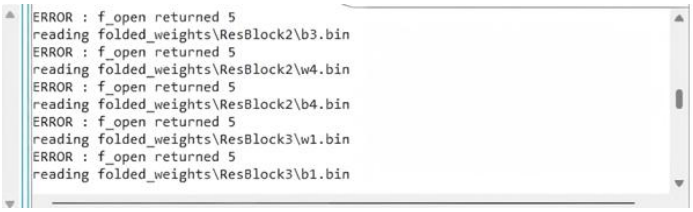

图 30 YOLO 权重文件、坐标信息导出

然后使用 Vivado Vitis 工具将上述设计应用到实际场景中。在应用程序中，应该特别注意在脚本中调整堆栈大小和堆大小的需要。Ld 文件在工程中价值相对较高。同时，为了防止由于程序栈不足导致训练失败或中断，需要将 BSP 中 standalone 中的 use_lfn 参数值调整为 1。

通过实验，未经加速优化的模型平均准确率维持在 95.8%，中间推理运行时间维持在 6-8 分钟；运行时间显著减少，保持在 100ms 以内，平均 95ms。从实

验结果中不难发现，与未优化模型相比，优化模型的推理精度几乎没有变化，但运行时间普遍提高了近 4000 倍；即优化系统将显著提高模型推理速度，损失精度几乎可以忽略不计，这为应用神经网络模型提供了显著的优势和便利。

表 3 未经加速优化模型与优化模型参数对比

操作系统	平均推理速度	模型推理精度
CPU Cortex-A9	7.23 min/Picture	78.9%
FPGA-Based Accelerator	95.8 ms/Picture	78.1%

3.2.2 YOLOv8 神经网络模块

（1） Yolov8 模型剪枝与量化

Yolov8 剪枝主要有两种方法：非结构化剪枝和通道剪枝。非结构化剪枝通过在权重矩阵中剪掉不重要的权重来减少计算需求。非结构化剪枝的效果较好，但不利于在 FPGA 上进行加速，因为不规则的数据结构不利于硬件的并行化。而通道剪枝在硬件实现中更为高效，因为它剪掉的是整个卷积核的通道，因此剪枝后权重矩阵依旧保持规则结构，便于 FPGA 的并行加速。故我们选择采用通道剪枝方法。首先定义剪枝标准，并选择剪枝比例，之后按照定义的剪枝标准，计算每一层中每个通道的重要性。由此确定要剪除的通道。通道剪枝会导致模型的结构发生变化，精度往往会有所下降。为了恢复精度，我们对剪枝后的模型进行微调。

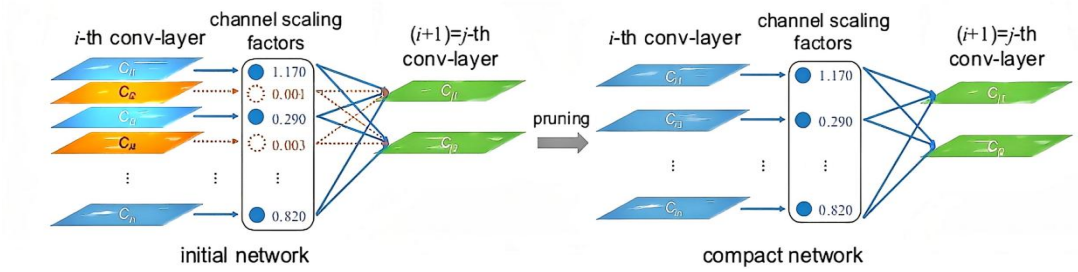

图 31 通道剪枝原理图

8 位定点量化是一种模型压缩和加速技术，通过将模型的权重和激活从 32 位浮点数转换为 8 位整数表示，以减少模型的存储需求和计算成本。这种量化方式可以显著提升推理速度和减少模型大小，同时对精度的影响较小。在 8 位量化中，我们将模型的浮点权重和激活值映射到 8 位整数范围，然后在推理时使用整型操作代替浮点操作。8 位定点量化与通道剪枝结合使用，能够进一步提高

YOLOv8 的轻量化效果。图 32 为 8 位量化曲线示意图。

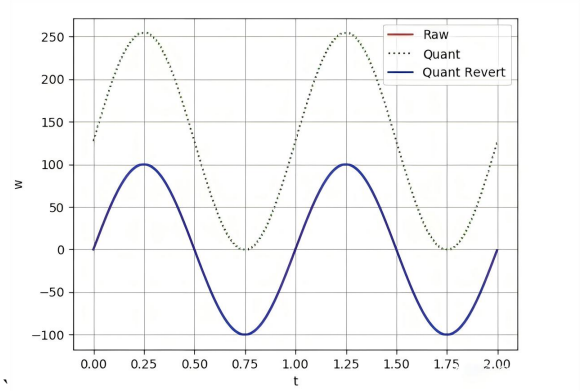

图 32　8 位量化曲线示意图

（3）数据集准备与训练

针对复杂的测试环境，本文根据数据集，在不同情况，不同拍摄角度采集相关图像 1428 张，通过 lableimg 对选取的图片进行标注，制作 VOC 格式数据集，作为足球分析数据集。

在训练前，分别建立文件夹存放.xml 格式的标签和采集的图片，划分摔倒数据集 fall-dataset 的 81%为训练集进行训练，9%为验证集进行验证，10%为测试集进行测试，完成对数据集的预处理。

对 YOLOv8 进行训练，我们修改相应的配置文件，在 model_data 文件夹下新建自己的标签类别 txt 文件，在其中写入我们训练的类别标签，我们只有一个类别就是 fall。之后，我们下载预训练权重文件，将下载好的文件放在 model_data 下，下载好后，我们就可以通过调用 train.py 文件来进行模型训练，我们设置的 epoch 为 300 轮：

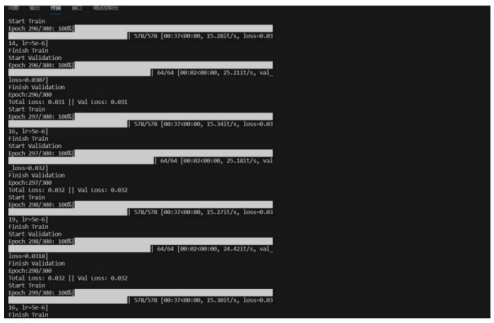

图 33 训练情况

训练完成后，我们得到了训练中的权重文件和训练效果最好的权重文件。

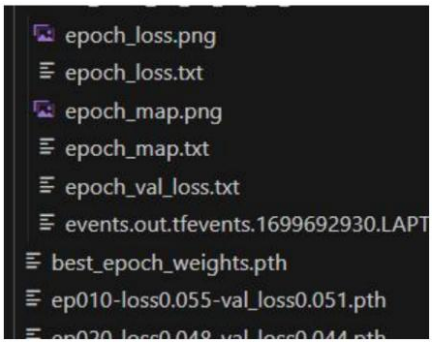

图 34 训练完成情况

还有模型相关性能参数平均精确度 Map 和损失函数值 Loss 的收敛过程，判断模型精度。本次 300 轮训练的模型中得到的训练结果如下：

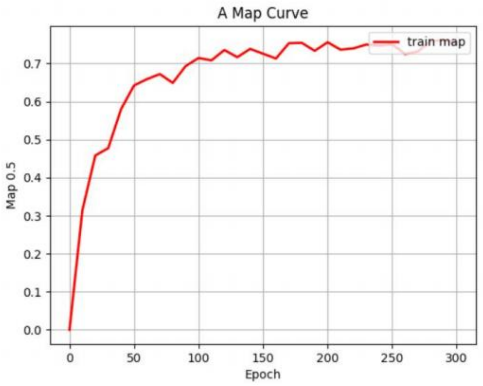

图 35 平均精确度 Map

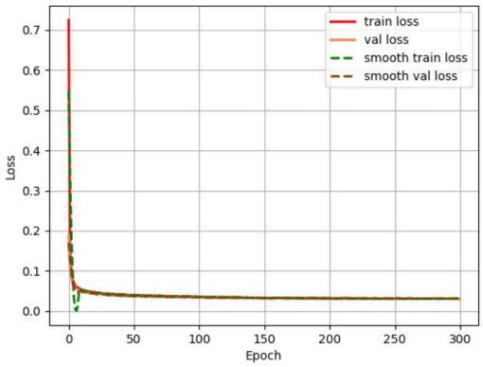

图 36 损失函数值 Loss

表 4 YOLOv8 模型轻量化前后对比

指标	原始模型(float32)	剪枝+量化后(int8)	优化率
模型参数量	10.2M(百万)	3.16M(百万)	69.0%
计算量(FLOPs)	25.1G(十亿次)	7.5G(十亿次)	70.1%
模型精度(mAP)	78.9%	78.1%	0.8%
模型大小(MB)	41.5MB	3.5MB	91.6%

如表 4 所示，通过对 YOLOv8 模型进行通道剪枝与 8 位定点量化，模型参数量从 10.2M 显著下降至 3.16M（减少了 69%），计算量（FLOPs）从 25.1G 下降至 7.5G（减少了 70.1%）。同时，模型精度仅有 0.8%的微小损失（从 78.9%降至 78.1%9），而模型体积压缩超过 90%。这证明了本轻量化方案在几乎不牺牲精度的情况下，极大地降低了模型复杂度，使其成功适配 ZYNQ-7020 平台。

3.2.3 硬件加速器架构设计

本加速器 IP 的设计遵循“控制与计算分离”的原则，采用了原生 Verilog 与 HLS 混合设计的方案，以兼顾计算性能与开发效率。

（1）PL 端

PL 端的加速器 IP 核负责执行 YOLOv8 网络中计算最密集的部分，特别是 8 位定点量化后的卷积运算。针对 YOLOv8 中大量的卷积操作，我们使用原生 Verilog 设计了一个高性能的 8 位（INT8）并行卷积计算引擎。该引擎深度优化了 DSP Slice 与 Block RAM（BRAM）的流水线使用，能够高效处理不同尺寸的卷积核，是实现系统高性能的关键。为实现灵活的数据调度和简便的接口集成，我们使用 Vivado HLS 生成了加速器的顶层控制 IP。该 IP 核主要负责通过 AXI 总线与 PS 端 DDR 内存进行高速数据交互、解析来自 ARM 处理器的加速指令、负责将 DDR 中的特征图数据，通过 AXI-Stream 接口流式地送入 Verilog 卷积计算引擎，并回收计算结果。

（2）PS 端

PS 端的双核 ARM Cortex-A9 处理器负责运行 PetaLinux 操作系统及所有非硬件加速的任务，包括运行 PyQt5 编写的用户交互界面，并负责整体任务调度；负责从 OV5640 摄像头抓取图像后的预处理（如缩放、归一化），以及 YOLOv8 推理完成后繁琐的后处理（如非极大值抑制 NMS）和目标跟踪（DeepSORT）算法；执行“策略分析与进球预测”等复杂但计算量较小的上层算法。

（3）软硬件数据流

系统的工作流程实现了高效的流水线作业，OV5640 采集的视频流通过 VDMA 存入 DDR 内存。PS 端的 ARM 核从 DDR 读取图像，完成预处理后，将待处理数据写入 DDR 特定区域，并向 PL 端的 HLS 控制 IP 发送“开始处理”指令。PL 端的 IP 核通过 AXI-Smart Connect 高速读取 DDR 中的数据，送入 Verilog 引擎完成密集的卷积加速，计算结果被写回 DDR，PL 端通知 PS 端处理完成，PS 端读取 DDR 中的结果，执行后处理和跟踪算法，最终在 PyQt5 界面上显示。

通过将最耗时的卷积任务硬化到 PL 端，并将控制和后处理任务交给灵活的 PS 端，本设计充分发挥了 ZYNQ 平台的异构计算优势。

3.3 特性成果

3.3.1 足球检测模式

本系统所设计的足球检测模式，旨在以高稳定性与高精度为目标，专注于实现对足球的实时识别与运动分析，克服其因体积小、速度快、易遮挡而导致的传统检测难题。为此，系统引入了基于 YOLOv8 轻量化改进模型的检测方法，并辅以轨迹平滑与物理建模等手段，有效提升了系统在高速运动场景中的适应能力和抗干扰能力。

具体而言，系统采用轻量化的 YOLOv8 网络架构对小目标检测能力进行强化，通过精细调整感受野与特征提取通道，使模型更加敏感于小尺寸、高速目标的空间分布。此外，为应对足球在高速运动中产生的帧间抖动和图像模糊问题，系统集成了 BallTracker 轨迹平滑算法，结合时间序列信息对检测结果进行后处理，以消除跳变与漂移现象，实现连续稳定的目标跟踪。在标注策略方面，采用 TriangleAnnotator 对足球进行方向性标记，不仅标示其当前空间位置坐标，也可直观反映运动方向与速度趋势，为战术分析提供可视化支持。

该模式的输出结果不仅包括足球的实时位置坐标，还提供完整的运动轨迹可视化曲线，能够对比赛中的控球时长、传球路径、球权归属情况等进行定量统计。例如，通过对每次传球行为的轨迹分析，可判定球的起止位置、传球成功率及关键传球区域，为评估球员配合效率与战术执行效果提供数据支持。同时，该模式还支持与战术模拟系统对接，将实际比赛中的球路运行动态映射至战术策略模块，辅助教练组进行战术优化与对手研究。

识别结果如图 37 所示。

(a)足球检测示例 1　　(b)足球检测示例 2

图 37 足球检测效果

基于检测结果，UI 交互界面上面会显示相关内容。在该模式中，系统实时监控赛场视频画面（左下区域），并通过 YOLOv8 模型对关键对象进行识别与追

踪，同时在界面上方实时统计比赛数据，包括犯规次数、越位次数、控球率、射门次数与传球成功率等核心指标。右侧面板展示系统自动生成的战术分析建议与裁判决策辅助信息，基于球员位置分布、控球趋势与关键区域压力判断，为教练组与裁判提供直观的数据支持。界面设计注重信息集成与可视化表达，有效提升了系统的人机交互性与实用价值，具体情况如图 38 所示。

图 38 足球检测模式 UI 界面

3.3.2 球员检测模式

球员检测是足球智能分析系统中的基础模块，其核心任务是对比赛画面中的所有参与人员进行实时精准识别。由于足球比赛场景中人物密集、动态性强，不同角色（如球员、守门员、裁判员）间存在服装相似、遮挡频繁等干扰因素，因此高效、稳定、多类别识别成为该模块建设的技术难点。为应对这一挑战，本系统基于 YOLOv8 目标检测架构设计了一套专用的球员检测子系统，能够在不影响实时性的前提下，对场上所有人员进行高精度检测并输出结构化信息，为后续的追踪、战术分析与事件识别提供坚实的数据基础。

系统采用 YOLOv8 框架的原因在于其优良的检测性能与模块化结构，该架构不仅具备较高的检测精度（检测准确率>95%），还能在保证检测速度的同时实现小目标与多目标的并行识别。通过在模型训练阶段引入定制化的足球比赛图像

数据集，并对裁判、守门员等非球员对象进行标签区分，模型能够学习到不同角色的显著特征，实现三类对象（PLAYER、GOALKEEPER、REFEREE）的有效区分。此外，系统结合 BoxAnnotator 对检测结果进行可视化标注，并融合置信度得分与误检过滤策略，提升检测结果的可信度和稳定性。

具体检测情况如图 39 所示。

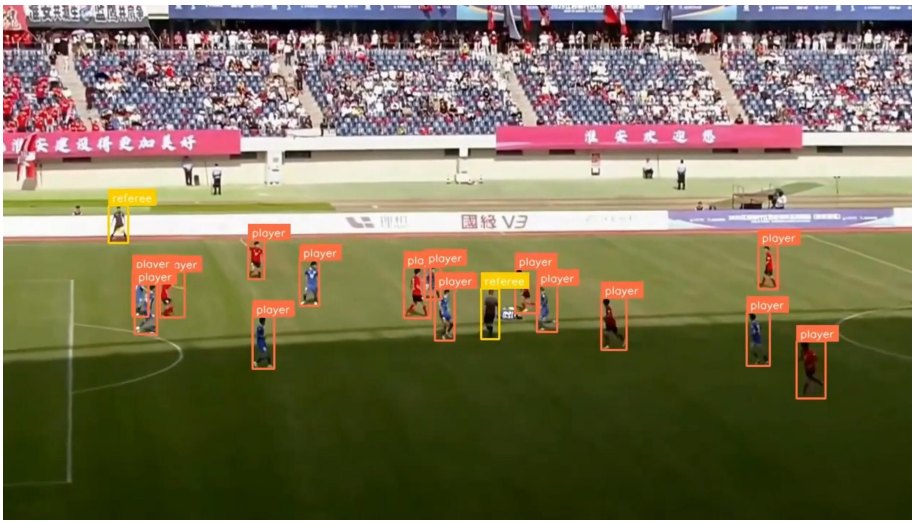

图 39 球员检测效果图

如图 40 所示，展示的是球员检测模式在比赛过程中运行的实时交互界面。界面上方显示当前比分、比赛用时以及核心技术统计指标，包括犯规次数、越位次数、控球率、射门次数与传球成功率，提供比赛态势的全局视角。左下方为“实时画面监控”模块，展示了基于 YOLOv8 算法对球员、足球、球门等目标的检测结果，并以边框形式直观标注于原始视频流上，实现高帧率下的目标识别与定位。

右侧部分为“裁判决策建议”与“战术方案建议”模块。其中，“裁判建议”根据实时图像分析结果输出对球员站位、角色识别、移动轨迹等信息的量化分析，辅助裁判员进行精细化判罚；“战术建议”模块则综合控球区域与球员状态数据，生成换人建议与阵型优化方向。整体界面采用模块化布局与分屏显示策略，实现了数据处理、视频监控与战术评估的协同集成，是系统高实时性、强交互性与智能决策能力的集中体现。

图 40 球员检测模式的 UI 交互界面

3.3.3 雷达模式

在构建智能化、可视化的足球比赛分析系统过程中，如何以最直观、信息密度最大的方式呈现比赛态势，是系统设计中的关键问题。为此，本项目引入了雷达模式（Radar View Mode），该模式通过多模块信息融合与空间视角转换，将传统视频检测结果映射为二维战术俯视图，实现了从“图像识别”到“战术抽象”的跨层次数据可视化表达。雷达模式不仅集成了足球检测与球员检测的全部功能，还整合了球员追踪与空间分析模块，以动态、实时、结构化的方式生成类似于专业战术板的足球比赛态势图，是系统中最具集成度和表现力的输出形式。

该模式的核心技术之一是视角变换与球场标准化映射。系统通过引入 ViewTransformer 模块，对原始视频图像中获取的检测坐标进行透视矫正，将不同摄像机视角下的空间数据统一转换为标准球场坐标系内的二维坐标点。此转换保证了检测结果在战术图上的空间一致性，使得分析人员能够准确掌握球员在场上的绝对位置与相对分布关系。基于该坐标系统，系统绘制出标准化的二维球场俯视图，雷达图中的每一名球员、每一次控球、每一条传球路径都被清晰地展现出来。

雷达模式支持多种信息维度的同步呈现。首先，原始视频画面依然保留，并

叠加检测标注，便于人工复核与多维比对；其次，雷达图以角落小窗或分屏形式展示，动态更新各球员的实时位置，使用不同颜色表示不同队伍，并对球员角色（球员、守门员、裁判）加以标识；再次，足球的位置与运动方向在雷达图中同样可视化，并与球员运动态势形成战术联动关系。除此之外，该模式支持压迫强度可视化，即通过空间密度分析展示球权争夺区域，并识别空当与压迫热点，为战术执行与调整提供决策支持。

检测效果如图 41 所示。

图 41 雷达模式检测效果

如图 42 所示，图中展示的是雷达模式下的交互界面运行实况。该模式通过多源检测信息融合与战术俯视图重构，实现了比赛态势的二维可视化表达。上方为实时比分、比赛时间与关键技术统计数据，包括犯规次数、越位次数、控球率、射门次数及传球成功率，为战术评估提供定量支撑。下方左侧显示原始视频画面与实时检测结果叠加图，系统基于 YOLOv8 识别球员与足球位置后，投射至标准球场坐标系中，并以红蓝区域进行阵型分布可视化。

右侧“裁判决策建议”模块提供阵型推测结果与行为分布分析，如主队预测阵型为 4-3-3 攻击阵型，客队预测阵型为 4-4-2 平衡阵型；并进一步推送越位、压迫区域及关键跑位等战术特征。底部的“战术方案建议”模块基于实时检测结果和系统推理模型，自动生成战术优化方向建议，如换阵型、换人建议等。

整体界面实现了图像检测结果与战术语义信息的联动呈现，是本系统“图像

识别—行为建模—战术辅助决策”全链路流程的关键体现，充分展示了系统在赛事数据智能分析中的实际应用能力与扩展潜力。

图 42 雷达模式下的 UI 交互界面

第四部分　总结

4.1 可扩展之处

本系统具备良好的可拓展性，可在多个方向实现功能深化与应用拓展。未来可接入多摄像头同步处理机制，提升对大型赛事场景中复杂战术的覆盖与分析能力；系统算法层面可引入更高级的多目标跟踪与行为识别模型，实现更精细化的球员战术动作分解；在数据应用方面，可与大数据平台对接，进行赛季级别的趋势分析与球员表现建模；同时，结合 5G 与边缘计算技术，可将系统部署至赛事现场边缘节点，进一步提升响应速度与实时交互能力。此外，系统还可拓展至青训监测、战术教学、虚拟解说等多元应用场景，形成足球技术分析的智能生态体系。

4.2 心得体会

在本项目的研发与实现过程中，我们对嵌入式系统设计、FPGA 软硬协同开发以及人工智能目标检测算法有了深入而系统的理解。项目以 ZYNQ-7020 为核心平台，融合了图像采集、视频处理、深度学习推理与数据可视化等多种关键技术，整个开发流程涵盖了从底层硬件逻辑设计到上层应用交互界面的全栈实现，对团队的综合能力提出了较高要求。

在硬件开发阶段，我们通过 Vivado 完成了对 ZYNQ-7020 硬件系统的配置与 IP 核集成，利用其可编程逻辑单元部署了部分图像处理任务，有效提升了系统的实时响应能力。同时，在 PL 与 PS 协同运行的过程中，我们深入学习并实践了 AXI 总线协议的配置与数据交互机制，为后续的软硬协同打下坚实基础。

在软件开发方面，我们结合 YOLOv8 模型开展了目标检测系统的优化与移植工作。为适配 FPGA 嵌入式环境，我们对模型进行了剪枝与量化，显著降低了计算资源占用，并通过大量实验调整参数以平衡精度与速度。UI 界面则采用 PyQt5 框架设计，力求实现简洁直观、操作便捷的交互体验。在集成调试过程中，面对硬件通信异常、模型加载失败、UI 显示卡顿等诸多挑战，我们逐步定位问题源头，反复迭代优化，最终实现系统稳定运行。

此外，本项目强调系统的可拓展性与实用性，不仅完成了基础的球员与足球检测，还集成了进球预测、战术分析、雷达模式等功能模块，极大增强了系统的

智能化水平与应用价值。通过项目的完整开发流程，我们深刻体会到理论知识与工程实践结合的重要性，也认识到团队协作、问题分析与持续调优在项目成功中的决定性作用。

本次实践不仅锻炼了我们的工程能力与技术素养，也激发了我们对智能体育分析系统更深层次的探索兴趣，为未来从事相关科研或产业化应用积累了宝贵经验。

第五部分 参考文献

[1] 魏志珍,黄佳旺,陈文文,等.基于改进 E2E-spot 足球动作识别算法[J].计算机工程与设计,2025,46(08):2163-2169.DOI:10.16208/j.issn1000-7024.2025.08.004.

[2] 张诗欣,李鸿强,李子熠,等.基于 YOLOv5 改进算法的足球检测识别研究[J].河北建筑工程学院学报,2024,42(04):235-240.

[3] 胡彧,杨龙.山西省精英足球裁判员犯规识别能力研究[C]//中国班迪协会,澳门体能协会,广东省体能协会.第十一届中国体能训练科学大会论文集（下）.太原理工大学体育学院;,2024:293-296.DOI:10.26914/c.cnkihy.2024.002142.

[4] 余绍华,钟亚平.人工智能在足球比赛中的应用——以半自动越位识别技术(SAOT)为例[C]//中国体育科学学会.第十三届全国体育科学大会论文摘要集——墙报交流(体育信息分会).武汉体育学院;湖北省运动与健康创新发展研究中心;,2023:200-202.DOI:10.26914/c.cnkihy.2023.080327.

[5] Anonymous .FedEx Football Recognition Benefits Kid Safety[J].Material Handling & Logistics,2009,

[6] 彭熙伟,娄倩文,庞璇.基于 PYNQ 的车牌定位与识别算法[J/OL].北京理工大学学报,1-11[2025-11-05].https://doi.org/10.15918/j.tbit1001-0645.2025.104.

[7] 何文康,杨志芳.基于 FPGA 的水下图像重建系统[J].自动化与仪表,2025,40(10):142-146+151.DOI:10.19557/j.cnki.1001-9944.2025.10.029.

[8] 范士勇,陈丽霞,李敏.基于 FPGA 的脑电信号采集系统设计[J].激光杂志,2025,46(10):213-219.DOI:10.14016/j.cnki.jgzz.2025.10.213.

[9] Sadough A ,Shahsavari M ,Wijtvliet M , et al.ELVSR: An efficient and lightweight convolutional neural network for FPGA-based FHD to UHD video super-resolution[J].Array,2025,28100542-100542.DOI:10.1016/J.ARRAY.2025.100542.

[10] Xia F ,Nan G ,Jia Z , et al.Enhanced YOLO with FPGA hardware acceleration for aluminum sheet defect detection[J].

[11] Capuano M G ,Capuozzo S ,Strollo G A , et al.Super-Resolution YOLO Object Detection for Maritime Surveillance with Real-Time FPGA Processing Onboard

Spacecraft[J].

[12] Morra J .Secure Control FPGA Packs Post-Quantum Protections[J].Electronic Design,2025.

[13] Kim H ,Kim K T .Design and Implementation of a YOLOv2 Accelerator on a Zynq-7000 FPGA[J].Sensors,2025,25(20):6359-6359.DOI:10.3390/S25206359.

[14] Calì R ,Falaschetti L ,Biagetti G .Optimized Implementation of YOLOv3-Tiny for Real-Time Image and Video Recognition on FPGA[J].

[15] Guo Z ,Zheng L ,Xu W .BGIR: A Low-Illumination Remote Sensing Image Restoration Algorithm with ZYNQ-Based Implementation[J].

[16] Hatti K ,Paramasivam C .Performance analysis of 4:1 MUX APUF Architecture Implemented on Zynq 7000 SoC FPGA[J].

[17] 沈家庆,刘维申.基于 ZYNQ 加速的 OpenCV 实时模板匹配系统设计[J/OL].云南大学学报(自然科学版),1-9[2025-11-05].

[18] 孟凡开,张峰,李淼,等.基于 Zynq 的卷积神经网络加速器设计[J].合肥工业大学学报(自然科学版),2025,48(07):904-909.

[19] 王晓.基于改进 YOLO 算法的图像火焰检测技术研究[D].齐鲁工业大学,2025.DOI:10.27278/d.cnki.gsdqc.2025.000913.

[20] 韩峰.基于 ZYNQ 的卷积神经网络手写数字识别系统加速方案设计[D].电子科技大学,2025.DOI:10.27005/d.cnki.gdzku.2025.005636.