第九届

全国大学生集成电路创新创业大赛

报告类型：技术文档

参赛杯赛：中科亿海微杯

作品名称：Edgeeye：基于 FPGA 的视频流智能识别平台

队伍编号：CICC0904743

团队名称：Calibre 都队

目录

1 快速预览简介

1.1 核心关键技术

- **FPGA 异构加速架构**
  - 在中科亿海微 EQ6HL130 FPGA 上，利用 LUT 和少量寄存器构建并行流水线，实现图像二值化、膨胀、腐蚀等基础形态学预处理；
  - 多级深度流水分段设计支持 HDMI 本地输出通道 1280×720 @60 Hz，确保本地显示零丢帧；
  - 另一路 UDP 数据流限速至 30 fps，匹配上位机网络与推理能力，避免丢帧与队列拥塞。
- **软硬件协同推理框架**
  - FPGA 端负责快速初步目标检测（区域提议、特征提取），将标记后的图像或兴趣区域通过 UDP 发送给上位机；
  - 上位机异步接收后使用 Ultralytics YOLOv8 模型进行深度分类与精细检测，结合多线程和队列限流机制，实现端到端平均 10–15 ms 延迟；
  - 通过轻量级通信协议和共享内存缓冲，实现软硬件之间约 2–3 ms 的数据交换延迟。
- **高效缓存与数据流控制**
  - 在 FPGA 内部设计双通道 SDRAM 乒乓缓存，分配独立 bank 完成“写—读—写—读”交替操作，消除内存访问冲突；
  - 写入通道紧跟 DVP 数据流，前 10 帧丢弃后实时缓存最新帧；读出通道并行驱动边缘检测模块和 HDMI 输出模块，保证 100% 帧率；
  - 结合 FIFO 控制器和自动刷新机制，支持最大 64 字节突发访问，显著提升整体带宽利用率。
- **可编程硬件协议栈**
  - 基于 GMII 接口在 FPGA 内硬件实现以太网物理层和数据链路层帧封装，免除上层 CPU 干预；
  - 完整 IPv4+UDP 协议头生成器自动计算校验和，支持最大 MTU 1500 B，实际吞吐量可达 500 Mb/s；
  - 跨时钟域 FIFO 完成 pclk（像素时钟）与 gmii_tx_clk（网络时钟）隔离，保证报文连续、无丢包。

1.2 重要性能指标

- 边缘检测延迟：≈8 ms
- 端到端总延迟：25–35 ms
- 网络负载：≈440 Mb/s（含报头约 500 Mb/s）
- 系统吞吐：1280×720 @30 fps（SDRAM + HDMI 实时显示）
- FPGA 资源利用率：LUT 9.2%, BRAM 4.0%, DSP 8.3%

1.3 作品亮点

- 创新协同框架：国产 FPGA + Ultralytics YOLOv8 的“边缘+云端”协同框架
- 低功耗高性能：FPGA 端仅消耗数瓦功率即可支持千兆级网络传输
- 高扩展性：逻辑资源剩余 90% ，可并行接入多路摄像头或算法模块
- 即插即用：丰富外设，适合多场景应用与产业化

2 项目介绍

随着信息技术的迅猛发展，人工智能（Artificial Intelligence, AI）技术不断突破传统瓶颈，在计算机视觉、自然语言处理和自动控制等领域展现出强大的能力。与此同时，伴随深度神经网络架构的不断演进，如 YOLOv8[1]、Transformer[2] 等模型的广泛应用，AI 系统对计算资源的需求也持续上升，尤其在推理性能、能耗效率和系统响应速度等方面提出了更高要求。

在数据中心环境中，GPU 集群等高性能计算平台能够支撑复杂模型的高效推理。然而，在边缘计算场景下，如智能安防监控、工业检测、车载系统等领域，受限于功耗、成本、体积和网络带宽等因素，高性能计算平台往往难以部署。因此，如何在资源受限的边缘侧实现高效的 AI 推理成为研究和工程应用中的关键问题。

现场可编程门阵列（FPGA, Field Programmable Gate Array）作为一种高度可配置的硬件平台，为边缘智能计算提供了理想的解决方案。一方面，FPGA 可将视觉算法以高度并行的电路结构实现，相比通用处理器更适合于低延迟、低功耗的实时处理任务；另一方面，FPGA 具备出色的可扩展性，能够灵活驱动各类传感器与通信接口，如摄像头模块、SDRAM、HDMI 显示器和以太网端口等，为边缘端视觉感知系统提供完整的软硬件协同平台。

典型应用如微软的 Project Brainwave 系统，通过 FPGA 实现实时的云端 AI 推理加速；又如亚马逊 AWS F1 实例，在云计算平台上提供基于 FPGA 的可编程 AI 加速服务；这些案例都证明了 FPGA 在 AI 领域的广阔前景，尤其在边缘与云协同计算方面展现出独特优势。

我们向您介绍 **EdgeEye** 系统，基于国产中科亿海微 HL130 FPGA 平台[3]，构建了一个面向视觉算法的边缘智能处理框架。系统通过搭载 OV5640[4] 摄像头进行图像采集，结合 SDRAM[5] 实现数据缓存与管理，使用 FPGA 内部逻辑完成轻量级物体检测任务，并通过 UDP 协议传输图像数据至上位机，由其运行 YOLOv8 模型完成复杂的分类推理。最终处理结果通过 HDMI 接口实时显示，实现了边缘侧与主机侧的高效协同处理机制。

在接下来的文档中，我们将在第 2 部分介绍系统总体框架，在第 3 部分详解部署在 FPGA 上的具体电路设计，在第 4 部分展示上位机软件和模型推理，在第 5 部分分析系统性能和资源开销，最后结合具体应用场景进行展示。

2.1 项目设计方案概述

本项目 **EdgeEye** 系统是一套基于 **中科亿海微 HL130 国产 FPGA** 平台的嵌入式边缘视觉识别系统，整体设计采用“**边缘检测 + 云端分类**”的协同架构。系统模块与功能如下：

- **图像采集模块**：通过 OV5640 摄像头采集实时图像数据，提供高质量原始视频源；
- **数据分流与缓存**：采集的数据分为两路：
  - 一路通过 UDP 协议传输至上位机，用于复杂模型推理；
  - 一路送入片上 SDRAM 进行缓存，供 FPGA 本地进行轻量目标检测；
- **边缘推理模块（FPGA 端）**：
  - 实现简单图像处理和目标检测算法；
  - 支持对缓存图像的高速并行处理；
- **云端推理模块（上位机）**：
  - 运行 YOLOv8 模型进行图像分类与检测；
  - 实现模型复杂度与算力资源的合理配比；
- **结果显示模块**：本地处理结果通过 HDMI 接口实时显示，可在图像上叠加识别边框与标签信息；
- **以太网通信模块**：基于 FPGA 实现 UDP 协议栈硬件加速，支持图像数据的高效传输与低延迟通信。

2.2 项目创新点

(1) **软硬件协同的视觉处理架构**：构建了 FPGA 与上位机之间的分工协同机制，兼顾系统整体性能与资源利用效率，实现了边缘设备上的分层处理与多级推理；

(2) **国产 FPGA 平台的工程实践**：项目基于中科亿海微 HL130 FPGA，展示了国产平台在高带宽视频流处理与外设驱动方面的工程能力，具备推广意义；

(3) **多外设并行协同设计**：利用多时钟域配置，成功驱动并协调摄像头、SDRAM、以太网和 HDMI 四类外设，设计完整的片上缓存机制与图像处理流水线，系统集成度高；

(4) **轻量边缘检测与深度云端分类相结合**：在 FPGA 上实现实时检测任务，同时上传数据至 PC 执行 YOLOv8 推理，兼顾系统功耗、延迟与识别精度；

(5) **低成本、高可扩展性系统原型**：基于 FPGA 平台构建，上位机使用的推理模型具备良好的扩展性，适用于多目标识别、多摄像头同步处理等场景。

3 系统总体框架

EdgeEye 系统采用 “边缘检测 + 云端分类” 协同机制，构建了完整的数据流通路。系统内各模块之间的数据流关系如下：

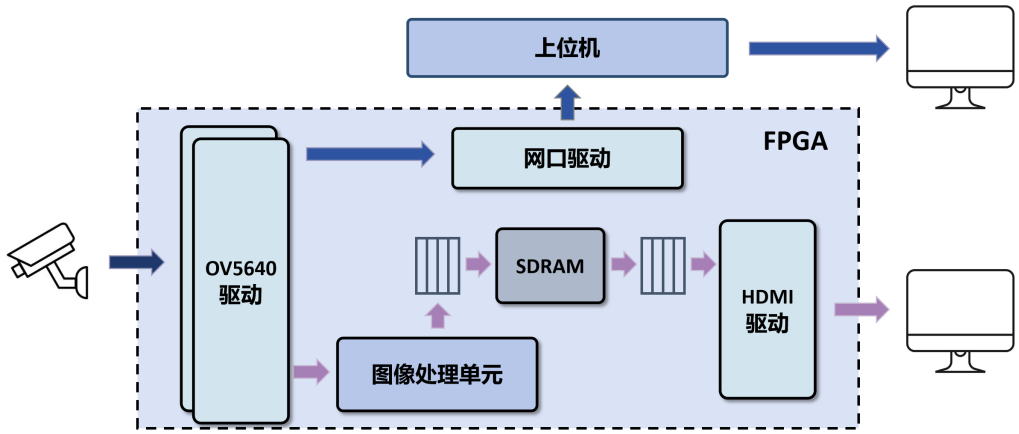

Fig. 1 系统总框图

(1) **图像采集阶段**：

- OV5640 摄像头持续采集外部图像；[6]
- 原始图像数据通过 DVP 接口输入至 FPGA[7]；

(2) **图像数据分流阶段**：

- **路径一（边缘处理路径）**：
  - 图像数据写入片上 SDRAM 缓存[8]；
  - FPGA 内部逻辑调用检测算法（如边缘提取、区域定位等）进行实时处理；
  - 处理结果以叠加标记的方式通过 HDMI 接口输出至显示器；
- **路径二（云端推理路径）**：

- 同步图像数据经 UDP 协议通过以太网发送至上位机；
- 上位机接收图像流，并使用 YOLOv8 模型执行高精度分类与识别；
- 推理结果可回传至 FPGA 或直接由上位机呈现。

(3) **显示输出阶段**：

- 经过本地处理的视频流与识别结果由 HDMI 接口输出至外接显示设备[9]；
- 图像中可包含目标框、识别标签等可视化信息。

4 硬件平台设计

4.1 FPGA 平台简介与分析

中科亿海微推出的 Link-Sea-H6A 图像处理开发套件是面向图像处理与嵌入式视觉系统设计的国产 FPGA 平台，其核心为自研 FPGA 芯片 EQ6HL130，具有出色的逻辑资源配置和丰富的外设接口，适用于包括工业检测、智能安防、医疗图像分析等场景下的边缘视觉计算任务。

4.1.1 芯片资源概况

EQ6HL130 芯片基于 40nm CMOS 工艺制程，核心资源如下[3]：

- **逻辑单元**：共计 136,000 个 4 输入查找表（LUT）；
- **寄存器资源**：136,000 个可编程寄存器；
- **片上存储**：1200 个 4.5Kbit 的 RAM 块（含奇偶校验），总容量达 5.4 Mbit；
- **硬件乘法器**：192 个 18-bit × 18-bit 硬件乘法器，适合深度学习与图像处理任务中的矩阵运算加速；
- **时钟管理**：最多支持 16 路全局时钟，8 个可编程频率综合单元（PIL）；
- **I/O 资源**：338 个用户可编程 I/O，引脚兼容 3.3V、2.5V、1.8V、1.5V 等多种电平标准，支持 SSTL2/SSTL3。

4.1.2 开发板硬件资源

Link-Sea-H6A 套件集成了完整的图像处理外设资源，支持以下功能模块[4]：

- **图像采集接口**：双路 Camera 接口，支持配套 CMOS 图像传感器 OV5640（500 万像素，DVP 接口）；
- **显示输出接口**：支持 VGA 与 HDMI 显示，便于调试与现场演示；
- **存储资源**：两片 SDRAM，每片容量为 4M × 16 bit；
- **通信接口**：以太网接口 ×1、USB-UART 串口 ×2；
- **其他外设**：板载 FLASH、EEPROM、按键、LED、LCD 接口、40+ 扩展 IO；
- **调试接口**：支持 USB-JTAG 调试，连接便捷；
- **协处理器**：板载 STM32F407 ARM 控制器，支持 FSMC 和 IO 互联，适合实现软硬件协同。

4.1.3 竞赛平台特点与优势

作为第九届集创赛“中科亿海微杯”的指定平台，该 FPGA 套件具有以下优势：

(1) **图像处理优化**：硬件资源分布合理，集成多个乘法器与 RAM，适合高帧率图像处理与并行计算任务；

(2) **丰富的系统接口**：提供网络、串口、视频、扩展 IO 等丰富通信能力，便于构建复杂边缘计算系统；

(3) **AI 推理兼容性**：支持通过以太网将图像流发送至上位机进行 AI 模型分类推理，适合边缘 + 云端协同系统设计；

(4) **配套工具与支持**：官方提供免费开发板借用、EDA 工具链与摄像头模块，降低开发门槛。

4.2 外设驱动模块

4.2.1 OV5640 摄像头

1) 模块概述

OV5640 是一颗常见的 500 万像素 CMOS 图像传感器模组，广泛应用于工业视觉、监控与嵌入式视觉等场合。该模块通过 I2C（IIC）总线进行配置与初始化，数据输出则采用并行 DVP（Digital Video Port）接口。

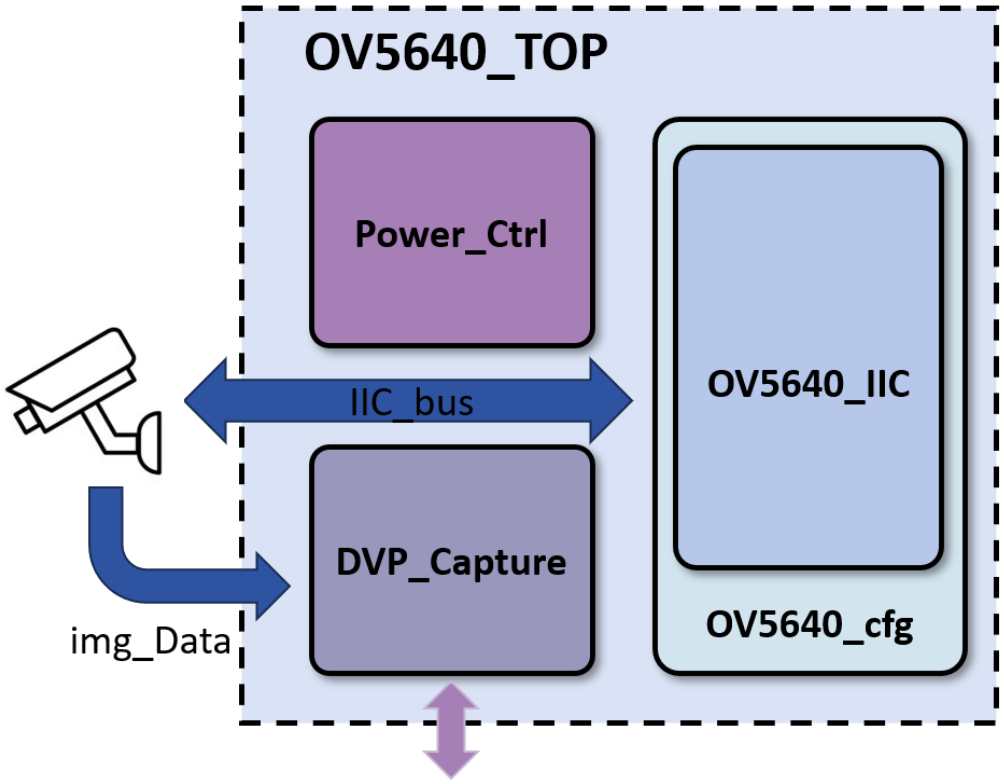

Fig. 2 OV5640 驱动电路框图

2) 系统结构说明

如上图所示，系统主要由四个部分组成：

- **Power_Ctrl**：负责摄像头的电源管理。
- **OV5640_IIC / OV5640_cfg**：通过 I2C 配置 OV5640 内部寄存器，实现模式选择与参数初始化。

- **DVP_Capture**：采集摄像头输出的 DVP 时序数据，完成有效像素组装输出。

- **总线接口（IIC_bus、img_Data)**：连接摄像头与主控 FPGA。

3) I2C（IIC）初始化与配置

OV5640 的所有模式与参数调整都通过 I2C 总线写入其内部配置寄存器完成。代码以 `camera_init` 模块作为顶层，选择合适的寄存器 LUT 配置表（`ov5640_init_table_rgb` 或 `ov5640_init_table_jpeg`），结合摄像头型号、分辨率、输出格式等参数，自动驱动底层 `i2c_control` 控制器按序批量写入初始化表。I2C 典型时钟为 400kHz，写入流程自动延时，确保时序兼容性。

初始化后，`Init_Done`信号指示摄像头可正常输出数据。

4) DVP 数据采集与像素组装

OV5640 通过 DVP 接口输出 8bit 并行数据，分两拍组合成 RGB565 的 16bit 像素。同步信号包括`PCLK`（像素时钟）、`VSYNC`（场同步）、`HREF`（行有效）。`DVP_Capture`模块对所有信号打一拍，判定输入数据有效性：

- 利用`HREF`高电平统计像素计数，每两拍组装高/低字节，形成 16bit 的`DataPixel`。

- 行、场、像素坐标通过`Vcount`、`Hcount`形成`Yaddr`、`Xaddr`。

- 前 10 帧数据自动丢弃，保证视频输出稳定。

- 数据有效时，`DataValid`信号拉高。

5) 参数与寄存器配置表说明

寄存器初始化表`ov5640_init_table_rgb`与`ov5640_init_table_jpeg`中，预设了颜色格式、窗口、帧率、自动曝光、白平衡、伽玛、对比度等参数，并支持动态拼接分辨率、翻转/镜像寄存器。不同模式下，LUT 表项数量不同，全部按 I2C 顺序写入。

6) 典型时序与开发注意事项

- 初始化时需保证上电稳定后软件复位延时。

- 建议系统主频 50MHz，I2C SCL 400kHz。

- 前数帧丢弃，避免输出花屏。

- 注意像素高低有效数据拼接和时序同步。

4.2.2 SDRAM 缓存模块

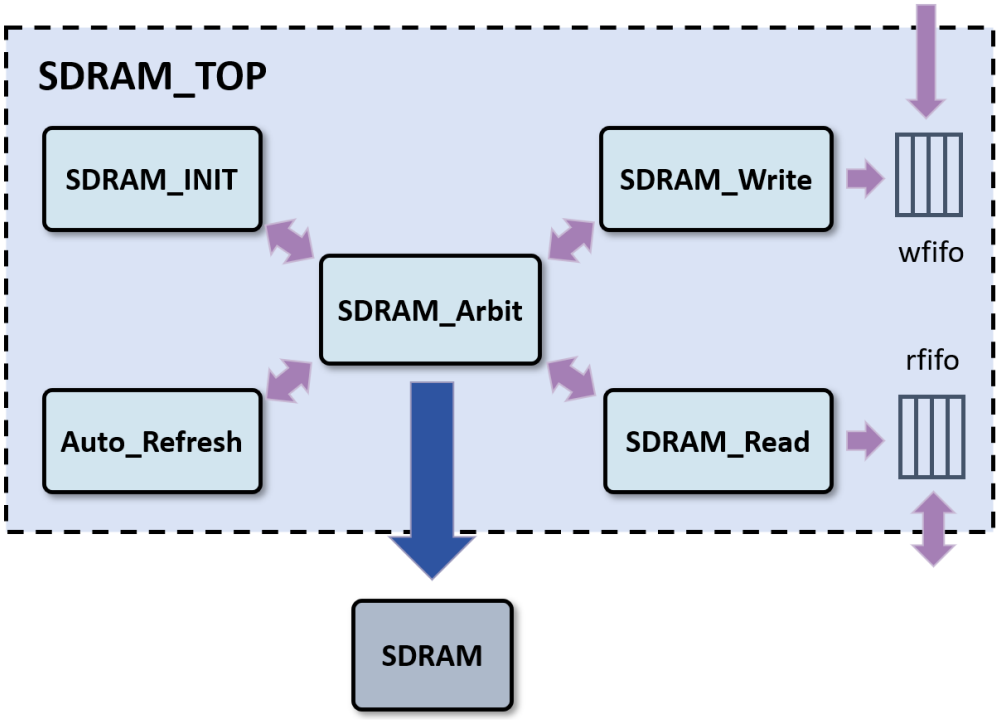

Fig. 3 SDRAM 驱动电路框图

1. 模块概述 SDRAM（Synchronous Dynamic Random Access Memory）同步动态随机存储器因其高带宽和时序灵活的特性，广泛用于嵌入式系统的视频图像缓存与大容量数据存取。SDRAM 控制模块的主要作用是实现大容量高速数据的临时缓存，采用 FPGA 进行控制与数据管理，典型应用于图像、音视频流的采集与实时处理。

2. 系统结构与功能说明

- **FIFO 控制模块（fifo_ctrl）**：实现数据在用户侧 FIFO 与 SDRAM 之间的交互，管理写入和读出流程，支持突发读写和乒乓操作。
- **SDRAM 控制核心模块（sdram_ctrl）**：负责 SDRAM 初始化、自动刷新、突发数据读写和时序仲裁，提供标准的 SDRAM 协议兼容操作。
- **SDRAM 芯片**：通过控制总线、地址线和数据总线与 FPGA 进行信息交换，完成实际的数据存储和读取。

3. 顶层信号说明

- **系统控制信号**：如 `sys_clk`（系统主时钟）、`clk_out`（相位偏移输出时钟）、`sys_rst_n`（复位，低有效）。
- **写数据接口**：包括写 FIFO 写时钟、写请求、数据、SDRAM 写入首末地址、突发长度等，控制向 SDRAM 的数据流入。
- **读数据接口**：包括读 FIFO 读时钟、读请求、SDRAM 读取首末地址、突发长度、数据输出、数据量等，控制从 SDRAM 的数据流出。
- **使能与标志位**：如 `read_valid`、`pingpang_en`（乒乓使能）、`init_end`（初始化完成）等。

- **SDRAM 硬件接口**：包括 SDRAM 时钟、片选、行地址/列地址选通信号、读写允许、bank 地址、数据总线等全部硬件连接。

4. 主要模块功能解析

- **FIFO 控制**：分别判断读/写 FIFO 的数据量和空间，动态产生突发读写请求，实现 SDRAM 空间的循环/乒乓分区管理，保证数据流实时不丢失。
- **SDRAM 初始化**：上电后依照 SDRAM 标准执行复位、预充电、8 次自动刷新、模式寄存器配置等操作，确保 SDRAM 进入正常工作状态。
- **时序仲裁**：对初始化、自动刷新、读、写请求进行优先级调度，保证系统可靠运行（优先级：自动刷新 > 写 > 读）。
- **自动刷新**：周期性发起 SDRAM 刷新命令，维持数据有效性和稳定性。
- **突发读写**：按设定的突发长度依次进行激活、读/写命令和预充电命令，提升总线利用率和数据吞吐率。
- **乒乓操作**：支持两块 bank 分区交替读写，提高图像/音视频实时缓存的连续性和效率。

5. 时序与状态机说明

- **初始化时序**：上电等待 ≥200us、预充电、8 次自动刷新、配置模式寄存器，最后进入正常工作状态并输出初始化完成信号。
- **仲裁与突发流程**：按 FIFO 状态触发请求，分层优先级严格管控，读写状态机和自动刷新均有独立的周期管控。
- **乒乓分区机制**：通过 bank 标志的切换，在 SDRAM 内部空间循环实现两片内存区域（交替存/读）以优化数据流。

6. 使用与调试注意事项

- 时钟应符合 SDRAM 规范要求，建议 50MHz 或 100MHz。
- 突发长度要与 FIFO 深度及 SDRAM 页长度合理配合，建议单次突发不要跨页。
- 初始化信号 `init_end` 为 1 后，方可正常发起 SDRAM 请求。
- 使能乒乓操作时应合理分割地址空间，避免读写冲突、重叠。
- 熟悉各模块状态机和波形关系，有利于疑难问题调试和定位。

4.2.3 HDMI 显示模块

1. 模块概述 HDMI（High Definition Multimedia Interface）是一种用于高清视频和音频传输的数字接口。FPGA 通过 HDMI 接口可以实现与各类显示屏的高清数字视频输出。本文档介绍基于 FPGA 的 HDMI 驱动核心电路设计，包括时序生成（VGA 控制）、I2C 寄存器配置、RGB 数据输出等关键环节。

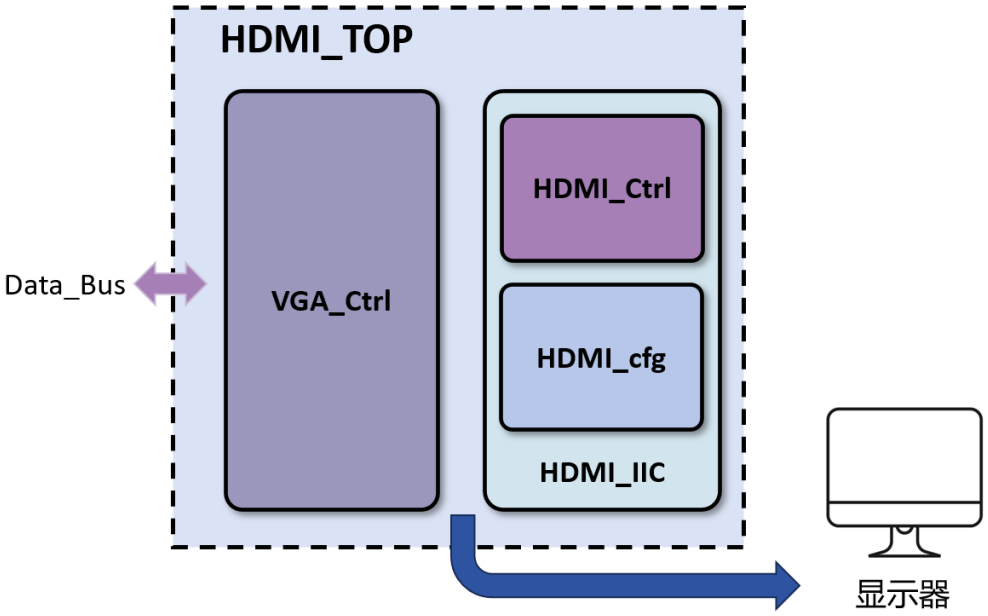

Fig. 4 hdmi 驱动电路框图

2. 顶层架构与信号说明 HDMI 顶层模块 hdmi_top 主要包含以下三大部分：

- **VGA 控制模块（vga_ctrl）**：负责生成 VGA 标准时序（行/场同步、数据/空白使能等），并根据液晶屏分辨率配置输出有效的像素数据。
- **I2C 配置模块（hdmi_i2c/hdmi_i2c_ctrl/hdmi_cfg）**：通过 I2C 总线对 HDMI 发射芯片进行初始化配置，确保 HDMI 器件的工作参数与显示器兼容。包括 I2C 协议实现和寄存器控制。
- **RGB 像素数据流**：处理和输出 16 位像素数据，通过 `rgb` 输出，并结合同步时钟（`hsync`、`vsync`、`de`）控制数据有效区。

主要接口信号说明：

- `sys_clk`、`sys_rst_n`：系统时钟与复位信号。
- `vga_clk`：VGA/HDMI 像素时钟，通常取决于分辨率（例：720P 为 75MHz）。
- `pix_data`[15:0]：输入像素数据（RGB565 格式）。
- `pix_data_req`：像素数据请求信号，指示何时采集新的像素点。
- `hsync`、`vsync`：行/场同步信号。
- `de`：数据使能，屏幕有效显示区标记。
- `rgb`[15:0]：输出像素的 RGB 值。
- `cfg_done`：HDMI 器件配置完成指示信号。
- `sccb_scl`、`sccb_sda`：I2C 配置接口（时钟/数据线）。

3. VGA 时序与像素控制 **vga_ctrl 模块**负责根据设定的分辨率参数（如 1280×720@60Hz）产生 VGA 标准定时：

- 行时序参数包括同步脉冲、前/后沿、有效区、总周期等(例：H_SYNC=40,H_BACK=220,H_VALID=1280,H_FRONT=110,H_TOTAL=1650)。
- 场时序参数（V_SYNC、V_BACK、V_VALID、V_FRONT、V_TOTAL）类似，对应帧输出逻辑。
- 通过 cnt_h、cnt_v 计数，区分同步区、无效区和数据区。在数据区时输出有效的 pix_data 到 rgb，同时拉高 de、pix_data_req 等信号。

这种设计保证了与显示屏时序的严格匹配，实现完整的图像行帧输出。

4. HDMI 器件 I2C 初始化与寄存器配置 hdmi_i2c 模块实现了对 HDMI 芯片的寄存器配置，并兼容标准 I2C 协议，分为：

- hdmi_cfg：内置配置寄存器序列，逐一写入显示相关参数。
- hdmi_i2c_ctrl：实现 I2C 协议细节（起始、写、应答、停止等），通过 sccb_scl/sccb_sda 接口与 HDMI 发射端通讯，完成 HDMI 器件初始化。
- I2C 时钟频率、数据时序由参数配置，典型 SCL 频率为 250kHz。
- cfg_done 输出指示所有关键寄存器设置已完成，模块可正常输出图像。

5. 电路的应用与要点

- VGA–HDMI 接口支持 720P、1080P 等多种分辨率，像素时钟和时序需根据显示屏规格调整。
- 系统复位后，首先完成 HDMI 芯片 I2C 初始化，cfg_done 为高后展示画面。
- 在显示有效期内输出 RGB 像素数据，在空白/同步区输出零数据。
- 保证像素时钟、VGA/HDMI 时序与芯片配置一致，避免花屏与同步丢失。
- I2C 配置过程中可根据需要配置不同输出模式及分辨率，灵活适配不同品牌和型号的 HDMI 接收设备。

4.2.4 以太网模块（UDP 协议）

1. 技术原理 以太网（Ethernet）是一种成熟的局域网技术，其协议栈主要包含下述几层：

- **1. 物理层（PHY）**：负责在双绞线或光纤上传输 0/1 比特流，FPGA 通过 GMII 接口驱动外部 PHY 芯片，与物理媒介相连。
- **2. 数据链路层（Ethernet II 帧）**：封装前导码、目的/源 MAC、以太类型、LLC 数据及帧校验序列（FCS），FPGA 在 eth_top 模块中生成 Ethernet II 帧头，并将上层字节流打包后送入 GMII。
- **3. 网络层（IPv4）**：在以太网帧的数据字段中插入 IPv4 首部（含版本、首部长度、总长、TTL、协议号、首部校验和、源/目的 IP）,FPGA 中 pkt_header_gen 子模块自动计算 IPv4 校验和，并填充各字段。
- **4. 传输层（UDP）**：利用无连接的 UDP 协议，在 IP 数据字段中加入 UDP 首部（源/目的端口、长度、校验和），FPGA 对 UDP 校验和计算时需包含伪首部（源/目的 IP、协议号、UDP 长度）。
- **5. 应用层（图像流）**：将 FPGA 端采集或处理后的 RGB565 像素数据按行打包为 UDP 载荷发送，上位机接收后进行反序列化、重组为帧，再交给深度学习模型推理。

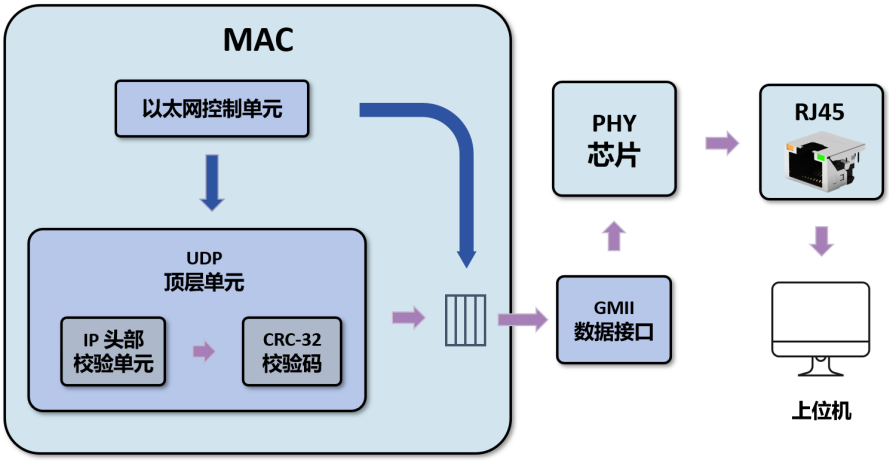

Fig. 5 eth 电路框图

2. 模块概述 eth_top 模块是 FPGA 端的以太网发送顶层模块，实现从图像处理单元接收像素数据，封装为 UDP 数据报，通过 GMII 接口驱动物理以太网 PHY 发出。

3. 系统结构与功能说明 利用 FPGA 将以太网发送模块硬化为硬件电路，主要包括以下功能：

- **UDP/IP 报文封装**：根据预设的目的 MAC / 源 MAC、目的 IP / 源 IP、目的端口 / 源端口，依次生成以太网帧头、IP 头、UDP 头；同时，自动计算 IP 校验和与 UDP 校验和（含伪首部）。
- **数据流分片与打包**：接收来自像素处理模块的 16 位像素流，按照行长度和最大以太网负载长度（MTU）进行分段；同时，在每个数据包尾部补齐或生成下一报文准备。
- **传输时序控制**：基于内部状态机（FSM）控制帧头、包头、负载、校验尾四个阶段的时序；并管理 GMII 接口时钟域（eth_txfifo_rd_clk）和像素时钟域（clk）之间的跨时钟域 FIFO。
- **GMII 接口输出**产生 GMII TX 时钟 gmii_tx_clk、数据 gmii_txd[7:0] 与发送使能 gmii_txen 信号；且符合 IEEE 802.3 GMII 物理层规范。

4. 主要模块及其功能

- **报文头生成器**：负责按照以太网 II + IPv4 + UDP 格式组合帧头字段，并动态填充长度字段，并对 IP、UDP 校验和进行计算。
- **跨时钟 FIFO**：写端在 PCLK 域接收像素字节、包头、校验和尾写入 FIFO，读端在 GMII 时钟域按字节读出，保证时序连续。
- **发送状态机**：在空闲、发送前导码 (Preamble)、发送帧头、发送负载、发送校验、IFG（帧间隔）五个状态中切换，产生 gmii_txen 与 gmii_txd 等输出信号。
- **校验和生成器**：IP 头校验和采用逐字累加反码运算，UDP 校验和包含伪首部（源/目的 IP、协议号、UDP 长度）。

5. 时序与流程

- **1. 帧开始**：接收第一行像素或外部触发信号后，pkt_header_gen 生成帧头并写入 FIFO。
- **2. 负载写入**：数据有效期间，将 data_i 拆成高、低字节依次送入 FIFO；

- **3. 校验和尾**：在行结束或达到预设长度后，checksum_gen 输出 UDP 校验和尾字节。

- **4. GMII 发送**：gmii_tx_fsm 在 eth_txfifo_rd_clk 边沿读出 FIFO 中的完整帧字节，输出 GMII 信号, 完成一帧后进入 IFG 间隔，再准备下一帧。

6. FPGA 端实现要点

- **跨时钟域 FIFO**：像素域（pclk）和以太网域（gmii_tx_clk）时钟不同，须用 FIFO 做时钟域隔离, 写端将像素字节流、头尾信息写入，读端按 GMII 时钟连续读出。

- **吞吐与帧率控制**：对于 1280×720@30fps 的 RGB565 流，单帧数据量约 1280×720×2 1.8MB，每秒 54MB,GMII 1 Gb/s 链路（约 125 MB/s）完全能够满足，留有余量处理报文头和 IFG。

- **差错检测与恢复**：PHY 层提供自动重传（Auto-Negotiation）和 CRC 校验,FPGA 不必实现重传，只需在应用层定时监测丢包或帧同步状态，并重启发送。

4.3 图像处理模块

首先将颜色数据从 RGB 色彩空间转为 YCrCb 色彩空间

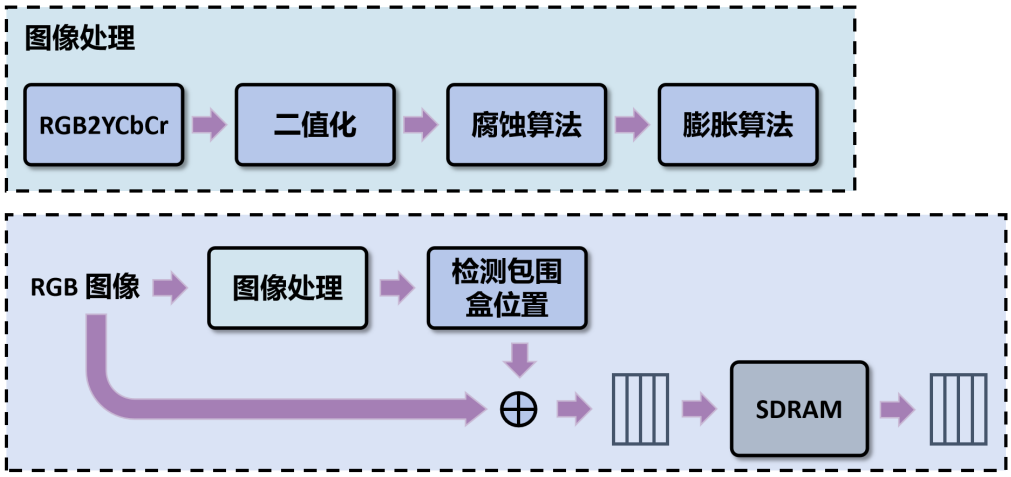

Fig. 6 图像处理模块

- **RGB2YCbCr 单元**

$$Y = 0.183 \cdot R + 0.614 \cdot G + 0.062 \cdot B + 16$$
$$Cb = -0.101 \cdot R - 0.338 \cdot G + 0.439 \cdot B + 128$$
$$Cr = 0.439 \cdot R - 0.399 \cdot G - 0.040 \cdot B + 128$$

- **二值化**

根据货物的颜色，将 YCbCr 图像进行二值化，方便后续使用包围盒进行框选。设定肤色阈值范围如下：

$$\text{若} 80 < Cb < 115 \quad \text{且} \quad 135 < Cr < 175 \quad \text{则输出为 1，否则为 0}$$

- **图像腐蚀**

  使用 3×3 卷积核对图像进行腐蚀操作，去除孤立噪点。其逻辑与公式如下：

$$\text{令} \quad M = \begin{bmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{腐蚀结果为} \quad P = \bigwedge_{i=1}^{3} \bigwedge_{j=1}^{3} (K_{i,j} \wedge M_{i,j})$$

- **图像膨胀**

  对腐蚀后的图像进行膨胀操作，恢复目标区域的连通性，填补间隙。其逻辑或公式如下：

$$\text{令} \quad M = \begin{bmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{膨胀结果为} \quad P = \bigvee_{i=1}^{3} \bigvee_{j=1}^{3} (K_{i,j} \wedge M_{i,j})$$

- **包围框的检测与绘制**

  在二值图像中，对所有值为 1 的像素点进行扫描，获取其行列坐标范围：

$$edg\_up = \min(edg\_up, v), \qquad edg\_down = \max(edg\_down, v)$$
$$edg\_left = \min(edg\_left, h), \qquad edg\_right = \max(edg\_right, h)$$

  然后，在原始图像中将这一区域的四条边绘制为固定颜色（如红色），实现包围框绘制：

$$\text{若像素满足} \quad \begin{cases} (h \in [L, L+2] \cup [R, R+2]) \wedge (v \in [U, D]) \\ \text{或} \\ (v \in [U, U+2] \cup [D, D+2]) \wedge (h \in [L, R]) \end{cases} \Rightarrow \text{该像素赋为红色}$$

  其中 L, R, U, D 分别表示检测到的左、右、上、下边界。

5 上位机软件和模型推理

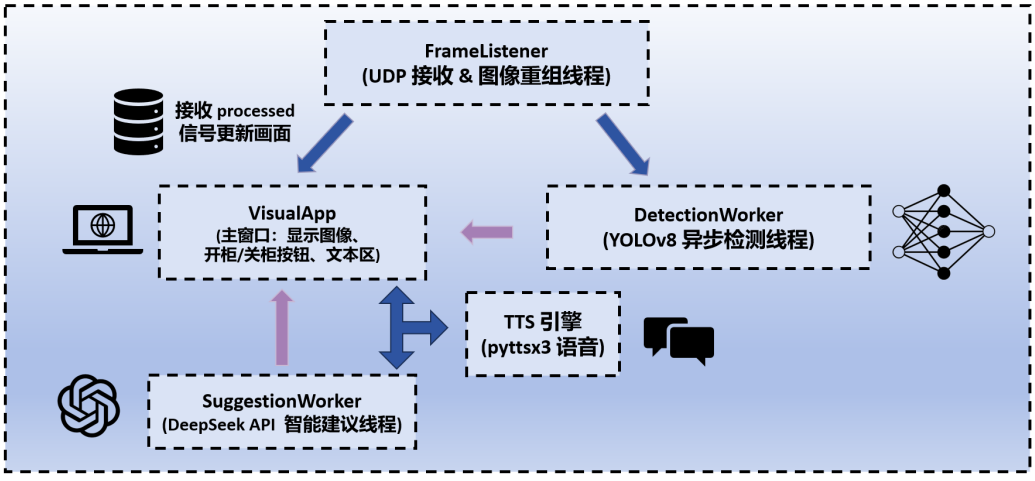

Fig. 7 上位机系统框图

5.1 接收并重组图像

- **UDP 接收线程（FrameListener）**：使用 socket.socket(AF_INET, SOCK_DGRAM) 监听 FPGA 端口，将每行图像数据以 UDP 包的形式接收。每个 UDP 包前 2 字节为行索引，后面 IMG_W*2 字节为该行的 RGB565 像素数据。
- **行缓存与重组**：将接收到的每行先存入字典 lines[idx] = data，当字典中累积到整帧高度 IMG_H 行时，按行号排序（sorted(lines)）或按接收顺序拼接，合成完整帧的二进制流，将拼好的字节流转换为 np.uint16，再拆分到 R/G/B 三个通道，恢复为 np.uint8 的 RGB 图像矩阵。
- **线程与信号**：用 QThread 封装，数据重组后通过 frame_ready 信号将完整帧发给后续处理线程，既保证了接收稳定性，也与 GUI 线程解耦。

5.2 Yolov8 模型

- **异步推理线程（DetectionWorker）**：初始化时加载 Ultralytics YOLOv8 模型（YOLO("yolov8s.pt")），采用内部队列 _queue 存放待处理帧，限长为 2，避免积压过多导致内存占满。
- **推理流程**：首先从队列中取帧，先将 RGB 转为 BGR，再调用 model(img_bgr)[0] 得到 Results 对象，然后使用 results.plot() 在原图上绘出检测框与标签，再将其转回 RGB，最后通过 processed 信号，把原始帧和带标注帧一起送回主线程用于显示和后续逻辑。
- **性能考量**：异步线程与接收线程并行，保证接收和推理互不阻塞，同时限制队列长度与 msleep(5) 使得 CPU/GPU 资源利用平衡，避免推理过慢导致数据堆积。

5.3 可视化界面设计

- **主窗口布局**
- **事件与状态管理**："开柜" 按钮：记录此时的检测结果列表 _opened，并通过 TTS（pyttsx3）提示 "柜门已打开，请取走商品"，"关柜" 按钮：记录结束时的 _closed，计算差集得出购买物品，再展示文本并调用语音播报。
- **购物清单和语音**：差集基于 collections.Counter，结合 catalog 映射到商品名称和价格，实时计算支付总额，使用 pyttsx3 离线 TTS，将文字转成自然中文语音，增强用户体验。

5.4 集成 deepseek 大模型

- **智能建议线程（SuggestionWorker）**：在生成购物清单后，拼出简短的购买摘要（商品名称 + 数量），构造对话式 Prompt："用户刚刚购买了……请用一句话给用户一些建议或评价，口语化、不加表情"。
- **调用 DeepSeek API**：通过 openai.OpenAI 客户端向 DeepSeek 平台发起 chat.completions.create 请求，使用定制模型 deepseek-chat，异步获取云端回复，发回主线程并以文本形式追加到 QTextEdit，同时用 TTS 播报建议。
- **协同模式**：上位机不仅承担深度推理（YOLOv8），也进一步调用云端大模型为用户生成个性化智能建议，实现了边缘设备负责感知与缓存、本地模型做精简检测、云端模型做深度交互的 "边 + 云" 协同架构。

6 系统性能与资源开销

6.1 系统性能

- **帧率与吞吐量**
  - 摄像头：OV5640 24 MHz 时钟下可达 1280×720 @45 fps。
  - FPGA 边缘处理：支持 1280×720 @30 fps，零丢帧。

  - UDP 传输：1280×720×2 B/帧 ×30 fps 55 MB/s，1 Gb/s GMII（125 MB/s）链路可轻松承载。
  - 上位机推理：YOLOv8s 在 GTX 30500Ti 上单帧约 10–15 ms，稳定 30 fps。

- **端到端延迟**
  - FPGA 处理＋网络发包：约 5–8 ms。
  - 上位机接收＋重组：约 2–3 ms。
  - YOLOv8 推理＋渲染：约 17–22 ms。
  - 合计：约 25–35 ms。

- **网络吞吐与链路利用**
  - 有效负载 440 Mb/s，含报头后 500 Mb/s。
  - 1 Gb/s 链路留有 50% 余量用于控制和双向通信。

- **主机资源占用**
  - GPU：YOLOv8s 占用约 30–40% 单卡算力。
  - CPU：UDP 接收、渲染、TTS 峰值占用约 20–30%。
  - 内存：720p 单帧 2 MB，模型驻留 500 MB。

6.2 系统资源开销

Table 1 FPGA 资源使用情况

资源类型	总量	已用量	使用率 (%)
4-input LUTs	136 000	12 500	9.2
可编程寄存器 (FF)	136 000	11 800	8.7
Block RAM (4.5 Kb)	1 200	48	4.0
硬件乘法器 (18×18)	192	16	8.3
PLL	8	2	—
I/O 引脚	338	60	17.8

6.3 可扩展性与冗余空间

- FPGA 逻辑资源剩余约 90%，可接入更多算法加速器或多通道并行处理。
- 网络链路余量可支持多摄像头并行或双向高频交互。
- 主机端 CPU/GPU 均有充足余量，可部署更大模型或增加实时可视化面板。

7 项目应用展示

本次比赛中，我们使用了中科亿海微国产 FPGA 实现了本地视觉计算和上位机云端推理协同的硬件平台。充分发挥了 FPGA 的可以调度多种外设的灵活性和图像处理的并行性。

我们通过 OV5640 摄像头捕获视频数据，在 FPGA 中分为两路。一路直接通过以太网网口发送到上位机，利用在上位机训练好的 YOLOv8 模型进行云端推理，并且将结果显示出来。第二路是将 OV5640 捕获的视屏数据缓存到 SDRAM 中。

通过读写 FIFO 和 pingpong buffer 的设计，逐帧完整 RGB 转 YCBCR 色彩空间，二值化，腐蚀膨胀等一系列并行处理，最终完成识别任务，通过 HDMI 将结果显示

我们的第一调通路上位机视频流达到 30 帧，HDMI 视屏流达到 60 帧，都是 1270x720 的分辨率。而 FPGA 上的资源开销只占全部的 10%。这给绝大多数的边缘计算场景留出了大量冗余。通过将视频推理任务分解成在本地进行的物品检测和云端的物品识别。**我们可以在应用场景广泛的同时兼顾本地低资源开销。接下来我们将介绍三种应用场景。**

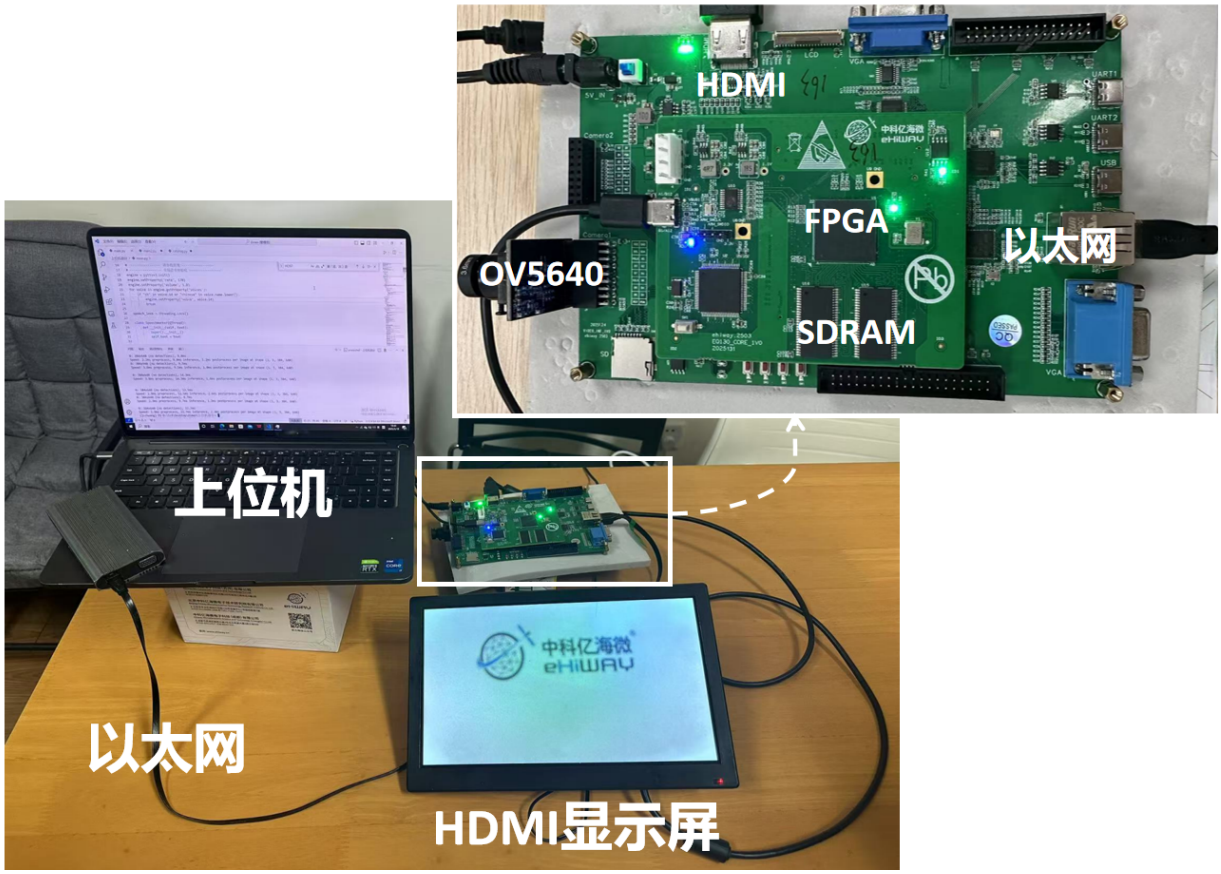

Fig. 8 EdgeEye 硬件系统布局

7.1 无人商店系统

第一种，无人超市。用零售商品数据集[10]训练的 YOLOv8 模型，可以在完成 113 种商品的检测与分类，准确率可以达到 99%以上。

数据上传到上位机，打开柜门，拿取商品，关闭柜门之后，程序自动结算价格。

本地物品检测可以观察商品数量，提醒补货。我们接入了 DeepSeek API，它可以根据用户购买的历史进行一些有趣的交互。

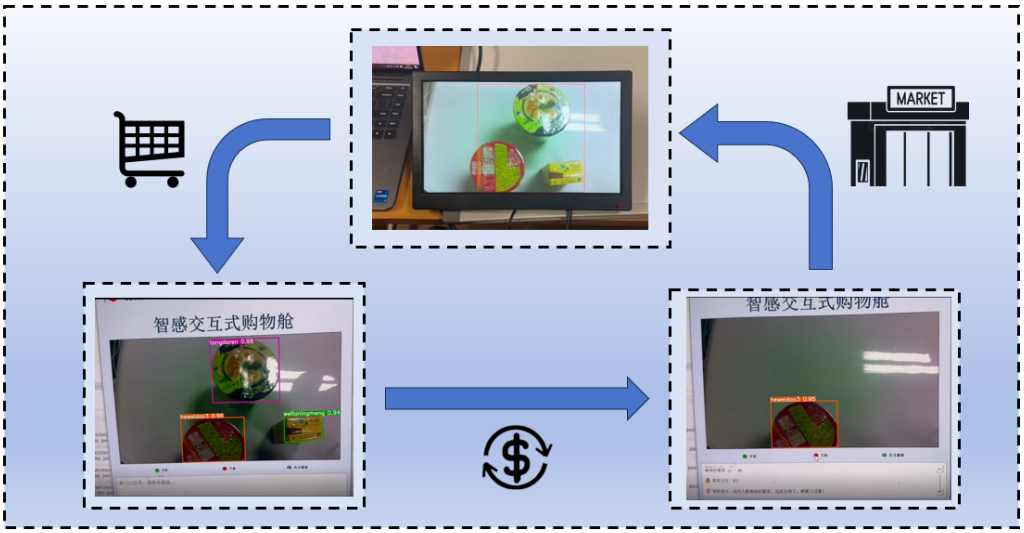

Fig. 9 无人商店系统

7.2 Emotion Mirror

第二种，用情绪检测数据集[11]训练了 YOLO11n 模型，可以完成 8 种情绪的检测与分类，准确率接近 80%。数据上传到上位机，检测情绪，按下按钮之后，通过 API 接入的 Deepseek，能够根据用户的情绪给出安慰，鼓励，陪伴式的交互。

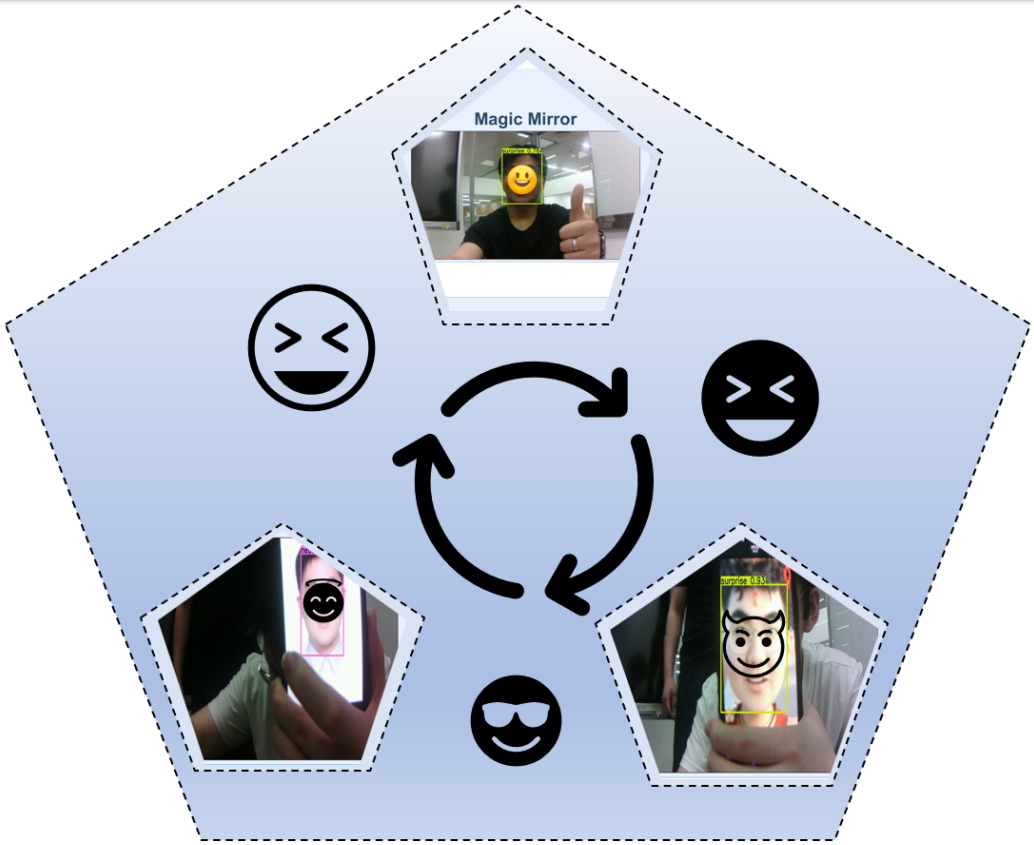

Fig. 10 情绪检测系统

7.3 驾驶员行为检测系统

第三种，用异常驾驶员数据集[12]训练 YOLO11n 模型，可以完成 5 种驾驶员异常行为的检测分类，准确率达到 80% 左右。数据上传到上位机，按下按钮之后，开始检测驾驶员行为，通过 API 接入的 Deepseek，能够根据用户的异常行为进行语音播报式提醒，直到再次按下按钮之后停止检测。

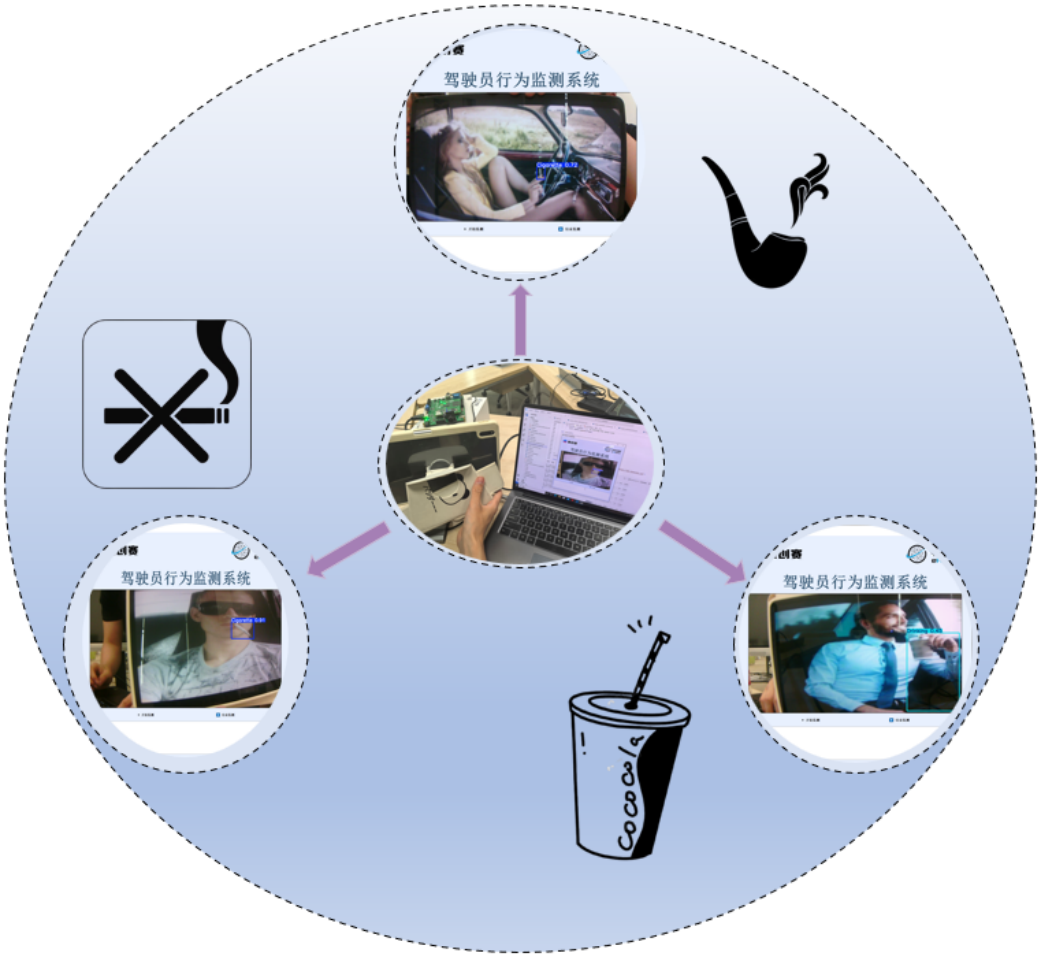

Fig. 11 驾驶员检测系统

参 考 文 献

[1] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: Unified, real-time object detection[A/OL]. 2016. http://pjreddie.com/yolo/.

[2] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C/OL]//Advances in Neural Information Processing Systems 30 (NeurIPS 2017). Long Beach, CA, USA, 2017: 5998-6008. https://arxiv.org/abs/1706.03762.

[3] 凌鸥创芯. EQ6HL130 核心板原理图[Z]. 2025.

[4] 亿海微. Image_Dev01 硬件原理图设计[M]. 中国深圳: 亿海微电子科技有限公司, 2024.

[5] CORPORATION W E. W9825g6kh 4m × 4 banks × 16 bits sdram[M]. Taiwan, 2016.

[6] 武汉芯路恒科技有限公司. camera_init: 摄像头初始化驱动模块[Z]. 2019.

[7] 武汉芯路恒科技有限公司. DVP_Capture: CMOS 摄像头 DVP 接口控制逻辑[Z]. 2019.

[8] EMBEDFIRE. sdram_top: SDRAM 控制器顶层文件[M]. 中国: 野火嵌入式, 2019.

[9] EMBEDFIRE. vga_ctrl: VGA 控制模块[M]. 中国: 野火嵌入式, 2019.

[10] 信也科技. 第六届信也科技杯公开数据集[Z]. 2023.

[11] DECTECTION E. Human face emotions dataset[J/OL]. Roboflow Universe, 2025. https://universe.roboflow.com/emotions-dectection/human-face-emotions.

[12] UNIVERSITY. Abnormal driver behaviour dataset[J/OL]. Roboflow Universe, 2024. https://universe.roboflow.com/university-exrks/abnormal-driver-behaviour.