ALINX VD100+Simulink 快速实现 FPGA 图像处理 Sobel 边缘检测

ALINX 2026年6月10日 13:48 上海

基于ALINX VD100的Simulink图像处理

> 原文作者：Adam Taylor
> 原文标题：MicroZed Chronicles: Creating Image Processing with Simulink
> 原文链接：https://www.adiuvoengineering.com/post/microzed-chronicles-creating-image-processing-with-simulink
> Github 链接：https://github.com/ATaylorCEngFIET/mz629

FPGA 大神、Ardiuvo & Hackster.IO 知名博主 Adam Taylor 此前 基于 ALINX VD100（AMD Versal AI Edge VE2302）开发平台实现 LCD 测试图案显示。

最近，他对这个项目进行了升级：**增加 ALINX 摄像头模块**，并构建完整的图像处理流水线，**实现实时视频采集与显示。**

由于 LCD 显示输出链路已经搭建完成，本次升级的重点主要集中在视频采集和图像处理部分。

项目硬件清单

本项目基于 ALINX VD100 Versal AI Edge 开发平台搭建，实现了从摄像头采集、图像处理到显示输出的完整视频处理链路。

类别	硬件型号	作用
主开发平台	ALINX VD100	项目开发与验证平台
核心器件	AMD Versal AI Edge VE2302	图像采集、处理及系统控制
摄像头	ALINX MIPI Camera Module	图像采集输入
显示设备	1280×720 LVDS LCD Panel	图像显示输出
存储器	板载 4GB DDR4	视频帧缓存
非易失存储	8GB eMMC、512Mb QSPI Flash	系统启动与程序存储

(AMD Versal自适应计算加速平台开发板VD100)

系统架构

```
ALINX MIPI Camera
        ↓
MIPI CSI-2 RX
        ↓
Sensor Demosaic
       ↓
VDMA Write
        ↓
DDR4 Frame Buffer
        ↓
VDMA Read
        ↓
Sobel Edge Detection IP
        ↓
LVDS LCD Display
```

构建视频采集链路

要实现摄像头视频采集，需要在设计中加入以下 IP 核：

MIPI CSI-2 RX Subsystem

配置参数如下：

- 4 Lane 输入
- RAW10 图像格式
- 每个时钟周期输出 4 个像素

该模块负责接收来自 MIPI 摄像头的数据流。

Sensor Demosaic

配置参数：

- 4 Pixels Per Clock
- RAW10 输入

该模块用于完成 Bayer 图像的去马赛克处理，将图像传感器输出转换为标准 RGB 图像。

Video DMA（VDMA）

通过 NoC（Network on Chip）连接 DDRMC，实现：

- 视频帧写入 DDR
- 视频帧读取 DDR

从而完成帧缓存管理。

CIPS I2C（EMIO）

启用 EMIO 方式的 I2C 接口，用于：

- 摄像头初始化
- 摄像头参数配置

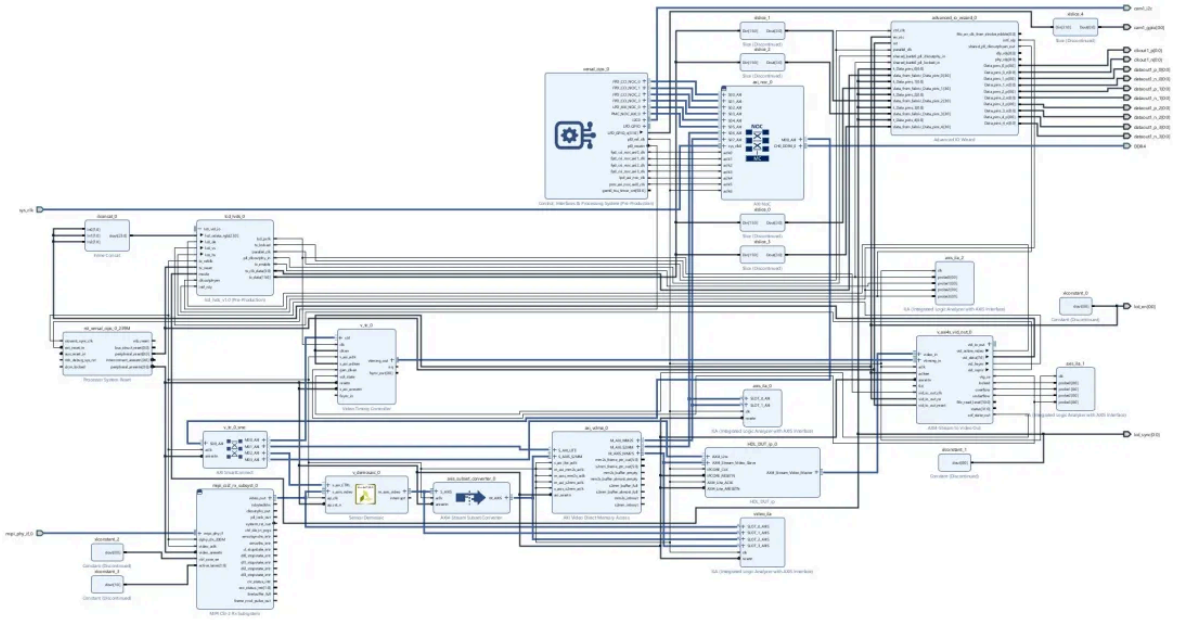
例如曝光、增益、分辨率等参数设置。

完成上述模块连接后，就建立起了一条完整的视频链路：

MIPI Camera → Demosaic → DDR Buffer → LCD Display

与此同时，还需要 Vitis 软件完成：

- 摄像头配置
- 图像处理 IP 配置
- VDMA 控制



整个系统设计已发布在 Adam Taylor 的 GitHub 中，点击文末“阅读原文”即可跳转。

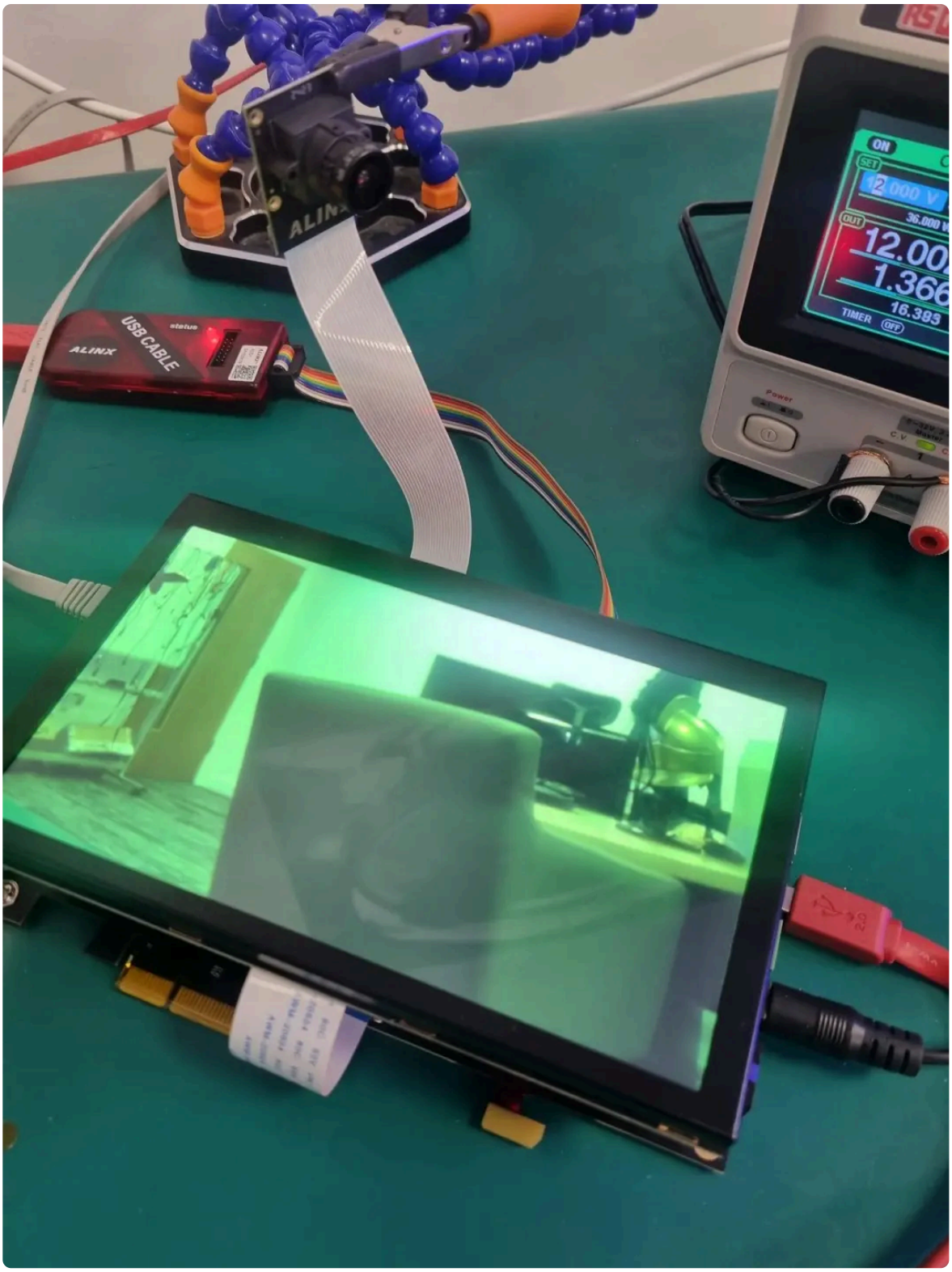
原始图像存在的问题

系统运行后，摄像头采集的画面已经能够正常显示到 LCD 上。

不过从显示效果来看，仍然存在两个比较明显的问题：

- Gamma 校正不足
- 白平衡未优化

因此图像质量还有进一步优化空间。



使用 Simulink 实现 Sobel 边缘检测

在完成视频采集与显示后，希望快速验证一个能够插入现有视频流水线的图像处理算法。

最终选择的是：

Sobel Edge Detection（Sobel 边缘检测）

并通过 MathWorks Simulink HDL Coder 自动生成 FPGA IP。

创建 HDL 子系统

首先在 Simulink 中建立一个新的 Subsystem，后续通过 HDL Coder 转换为 Vivado IP。

由于 Sobel 算法需要灰度图像作为输入，因此第一步需要实现：**RGB 转灰度**

```
RGB Image
    ↓
Grayscale Conversion
    ↓
Sobel Processing
```

即把彩色图像转换为灰度图像后，再进行边缘检测处理。

AXI4-Stream 视频接口设计

整个模块采用标准 AXI4-Stream Video 接口。

输入数据包括

tData

24bit 像素数据：

- R：8bit
- G：8bit
- B：8bit

组成一个 RGB888 像素。

控制信号

- tUser
- tLast

用于表示**帧起始**和**行结束**等视频时序信息。

Sobel 算法处理流程

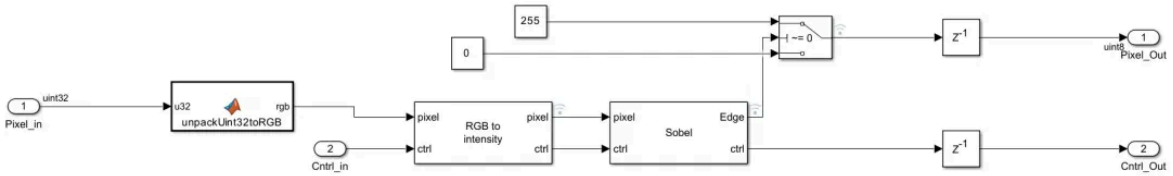
完成灰度转换后，图像进入 Sobel 运算模块。

Sobel 算法会输出边缘检测结果。

通过 Switch 模块完成二值化：

- 检测到边缘 → 输出 255
- 非边缘区域 → 输出 0

最终生成黑白边缘图像。



Simulink 仿真验证

为了在 FPGA 实现之前验证算法效果，可以先在 Simulink 进行视频仿真。

输入端采用 **From Multimedia File** 读取视频文件。

随后利用 **Frame To Pixels** 将视频帧转换为：

- 像素流
- 视频同步信号

模拟真实视频接口的数据格式。

输出端通过 **Pixels To Frame** 重新组合图像帧。

重新组装视频帧。

并利用 **Video Viewer** 查看处理结果。

AXI4-Stream 数据格式转换

这里遇到一个比较典型的问题。

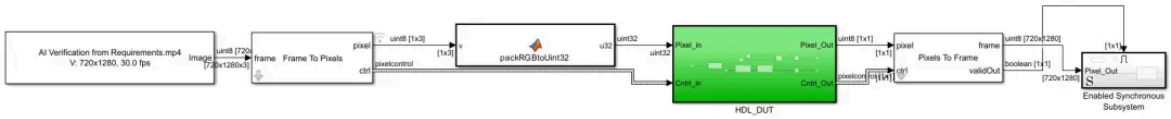
Frame To Pixels 输出格式为 `uint8[1×3]` 即 `[R G B]` 三个独立的 8bit 数据。

而 AXI4-Stream 接口要求输入格式为 `uint32` 单一数据输入。

因此需要增加一个 MATLAB Function：将 RGB 数据打包为单个 32 位数据（`uint8[1x3]→uint32`）。由于 MATLAB 不支持 24bit 数据类型，因此只能采用 uint32 作

为替代方案。

在 HDL 模块入口处，再执行相反操作： `uint32 → uint8[1x3]` 恢复 RGB 数据。



Source	Port Type	Data Type	Interface	Interface Mapping
Pixel_in	Inport	uint32	AXI4-Stream Video Slave	Pixel Data
Cntrl_in	Inport	bus	AXI4-Stream Video Slave	Pixel Control Bus
Pixel_Out	Outport	uint8	AXI4-Stream Video Master	Pixel Data
Cntrl_Out	Outport	bus	AXI4-Stream Video Master	Pixel Control Bus

---

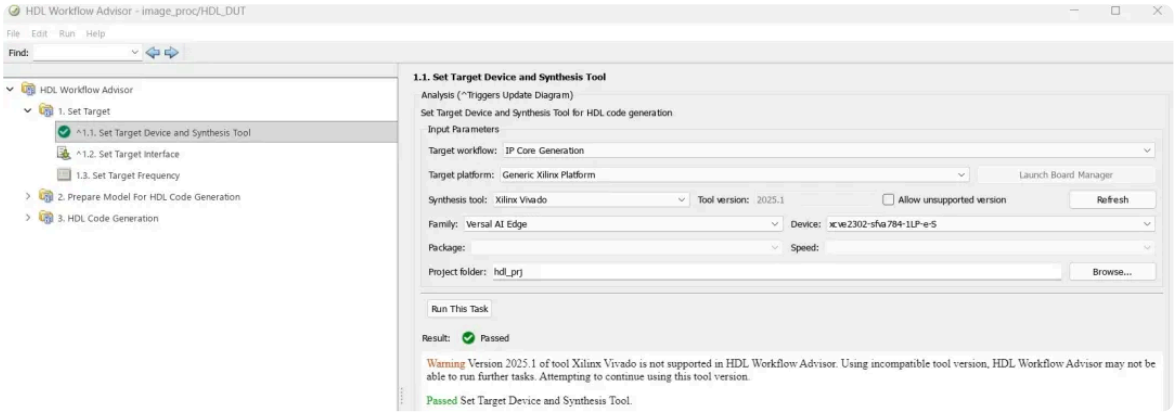
自动生成 Vivado IP

完成仿真验证后，通过 HDL Coder 直接生成 Vivado IP。

生成内容包括：

- RTL HDL
- AXI4-Stream 接口
- Vivado IP Packager 文件
- 工程支持文件

开发者可以直接导入 Vivado 使用。



集成到 VD100 视频流水线

生成的 Sobel IP 被加入到 Vivado 工程中。

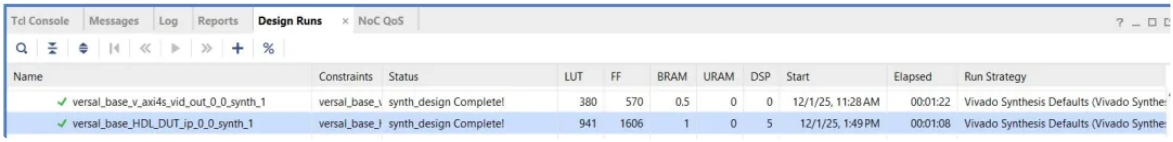
由于该模块采用**单像素处理（1 Pixel Per Clock）**，因此放置在 **VDMA 输出端之后。**

资源占用情况

资源类型	使用量
DSP	5
LUT	941
Flip-Flop	1606

工作频率：**200 MHz**

在 Versal AI Edge VE2302 上实现时序收敛没有任何问题。



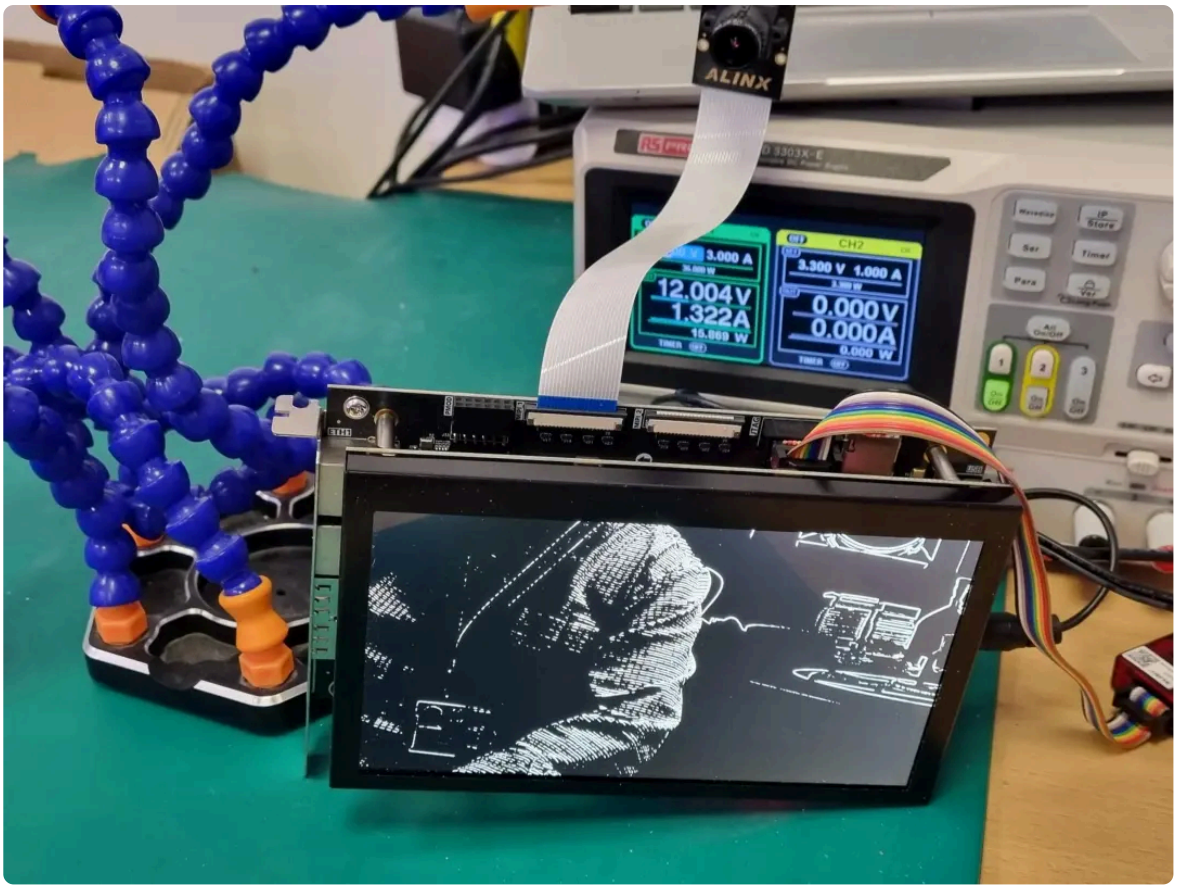
实际运行效果

修改 Vitis 软件，使其能够配置 Sobel 模块后，系统成功运行。

实时摄像头画面经过 FPGA 处理后实现了完整的实时边缘检测演示系统：

- 实时摄像头采集
- FPGA 边缘检测处理
- LCD 实时显示

显示刷新率达到 **60Hz**



项目总结

本项目展示了一条典型的 Versal AI Edge 图像处理开发流程：**MIPI 摄像头采集 → 去马赛克处理 → DDR缓存 → Sobel边缘检测 → LCD显示**

> 实践表明，利用 **Simulink HDL Coder** 可以快速完成**算法验证**，并**直接生成可用于 Vivado 的 IP 核**，从而缩短图像处理算法从原型设计到 FPGA 实现的开发周期。

对于机器视觉、边缘 AI、智能相机以及实时图像处理应用而言，这种开发模式具有较高的工程实践价值。

整个系统设计已发布在 Adam Taylor 的 GitHub 中，点击文末“阅读原文”即可跳转。