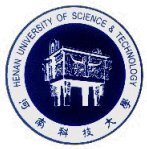
分类号 _____ UDC _____

密　级 _____ 编　号 _____



河南科技大學

专业硕士 学位论文

基于 Zynq 的运动目标检测技术研究

Research on Moving Target Detection Technology Based on Zynq

学位申请人： 张帅

指 导 教 师： 林青松　副教授

合 作 教 师： 潘晓东　研究员

专 业 领 域： 控制工程

学 位 类 别： 工程硕士

2022 年 05 月

独创性声明

本人声明，所呈交的论文是我个人在导师指导下完成的研究工作及取得的研究成果。据我所知，文中除了特别加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包含为获得河南科技大学或其它教育机构的其他学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：张帅

日　　期：2022年5月25日

关于论文使用授权的说明

本人完全了解河南科技大学有关保留、使用学位论文的规定，即：学校拥有对所有学位论文的复制权、传播权、汇编权及其它使用权（特殊情况需要保密的论文应提前说明，但在解密后应遵守此规定）。

需要保密的论文请填写：本学位论文在　　年____月至____年____月期间需要保密，解密后适用本授权书。（需要保密的学位论文无须向图书馆提供论文的电子文档）

研究生签名：张帅　　导师签名：

日　　期：2022年5月25日

摘　要

多运动目标的实时检测技术作为计算机视觉研究中重要的一部分，目前已经广泛应用于智能安防、智能交通、军事等诸多领域中，其多元化应用场景对系统的功耗、体积、实时性的指标要求非常高。但是，PC 平台存在功耗高、体积大等缺点，并且传统的嵌入式处理平台由于其串行处理的方式无法满足实时性的要求。此外，基于现场可编程门阵列（Field Programmable Gate Array，FPGA）的运动目标检测系统满足实时性的要求，但是其缺少处理器系统软件开发的灵活性，需要进一步配合处理器来完成。针对上述问题，本文构建了一种基于全可编程片上系统（Zynq-7000 All Programmable SoC，Zynq）的实时多运动目标检测系统。主要贡献如下：

首先，从运动目标检测算法和视频图像处理平台两个方面，阐述了国内外运动目标检测技术的研究现状。针对当前运动目标检测系统存在实时性差、功耗高、便携性差等诸多问题，本文对常用的运动目标检测算法及辅助的图像处理技术进行了系统的理论分析。

其次，本文设计了一种运动目标检测知识产权核（Intellectual Property Core，IP 核）完成图像处理。使用先入先出（First Input First Output，FIFO）存储器缓存的方式将两帧数据对齐；为加快系统的处理速度和提高检测算法的精度，对连续两帧图像数据使用了灰度化、中值滤波的预处理操作；对预处理后的两帧图像进行差分操作；选取了 4 组不同阈值进行对比实验，实现差分结果的二值化；对二值化的处理结果进一步使用形态学滤波操作；使用一种基于邻域的理论优化多运动目标标记方法，即通过间距来判断是否为同一个物体，并去除重叠边框，实现对多运动目标的标记；将检测结果与当前帧彩色图像进行叠加，获得良好的视觉效果。

最后，采用模块化的设计思想，设计了多运动目标检测系统总体结构。该系统包括图像采集、数据存储、图像处理、图像显示 4 个模块，并使用软硬件协同设计的方法将任务合理划分为可编程逻辑（Programmable Logic，PL）和处理系统（Processing System，PS）。在 PL 端利用其并行运算的方式实现运动目标检测的硬件算法加速、图像采集和高清多媒体接口（High Definition Multimedia Interface，HDMI）驱动，将各个 IP 集成完成硬件系统的搭建。使用视频直接存储器访问（Video Direct Memory Access，VDMA）IP 核将图像数据缓存到第三代双倍数据率同步动态随机存取存储器（Double-Data-Rate Three Synchronous Dynamic Random Access Memory，DDR3）中实现帧缓存，解决了 PL 端存储资

源受限的问题。此外，图像采集和 HDMI 驱动都采用可编程逻辑实现并封装成 IP 核，提高了系统的集成度；在 PS 端中完成摄像头的驱动、帧缓存的配置以及各个 IP 核的初始化。

综上所述，本文搭建的系统不仅具有并行运算速度快的优势，而且具有处理器系统软件开发的灵活性，为后续系统开发升级奠定基础。测试结果表明，该系统能够实时检测并标记多个运动目标，并且系统体积小、功耗和成本低，具有一定的实用价值。

关 键 词：Zynq；多运动目标；实时检测；软硬件协同设计

论文类型：技术类

选题来源：其他

ABSTRACT

As an important part of computer vision research, the real-time detection technology of multi-motion targets has been widely used in many fields such as intelligent security, intelligent transportation, military, etc. Its diversified application scenarios have very high requirements on the system's power consumption, size, and real-time index. However, the PC platform has the disadvantages of high power consumption and large size, and the traditional embedded processing platform can not meet the real-time requirements due to its serial processing method. In addition, the motion target detection system based on Field Programmable Gate Array (FPGA) meets the real-time requirements, but it lacks the flexibility of processor system software development and needs to further cooperate with the processor to complete. To address the above issues, a real-time multi-motion target detection system based on Zynq-7000 All Programmable SoC（Zynq）is constructed in this paper. The main contributions are as follows.

Firstly, the current research status of motion target detection technology at home and abroad is described from two aspects: motion target detection algorithm and video image processing platform. For the current motion target detection system has many problems such as poor real-time, high power consumption, and poor portability, this paper provides a systematic theoretical analysis of the commonly used motion target detection algorithms and auxiliary image processing techniques.

Secondly, this paper designs a motion target detection Intellectual Property Core (IP core) to complete the image processing. The two frames are aligned using First Input First Output (FIFO) memory cache; to speed up the processing speed of the system and improve the accuracy of the detection algorithm, a pre-processing operation of graying and median filtering is used for two consecutive frames of image data; a differencing operation is performed on the pre-processed two frames; four sets of different thresholds are selected for comparison experiments to realize the binarization of the differencing results; the binarized processing results are further processed using morphological filtering operation; using a neighbor-based theory to optimize the multi-motion target labeling method, i.e., determining whether it is the same object by spacing and removing overlapping borders to achieve the labeling of multi-motion targets; overlaying the detection results with the current frame of color

images to obtain good visual effects.

Finally, the overall structure of the multi-motion target detection system is designed using the modular design idea. The system includes four modules of image acquisition, data storage, image processing, and image display, and the task is reasonably divided into two parts, Programmable Logic (PL) and Processing System (PS), using a hardware-software co-design approach. The hardware algorithm acceleration for motion target detection, data acquisition and High Definition Multimedia Interface (HDMI) interface driver are realized by using parallel operation at the PL side, and each IP is integrated to complete the hardware system construction. The Video Direct Memory Access (VDMA) IP core is used to cache image data into Double-Data-Rate Three Synchronous Dynamic Random Access Memory (DDR3) for frame caching, solving the problem of limited storage resources on the PL side. In addition, the image acquisition and HDMI driver are implemented with programmable logic and encapsulated into IP cores to improve the system integration; the camera driver, frame cache configuration and initialization of each IP core are completed in the PS side.

To sum up, the system built in this paper not only has the advantage of fast parallel computing speed, but also has the flexibility of processor system software development, which lays the foundation for subsequent system development and upgrading. The test results show that the system can detect and mark multiple motion targets in real time, and the system has practical value because of its small size, low power consumption and cost.

KEY WORDS: Zynq; Multiple moving objects; Real-time detection; Hardware and software co-design

Dissertation type: Technology

Subject source: Other

目　录

第1章 绪论

1.1 课题的研究背景与意义

人类是视觉动物，主要通过视觉获得外界信息，从而感知世界、认识世界。在这个信息庞杂的时代，只靠人来处理信息是远远不够的。如何使用计算机代替人高效地处理信息就显得尤为重要，在这种情况下，计算机视觉应运而生。计算机视觉的目的就是使计算机具有目标识别、目标行为分析与场景感知等机器视觉的能力，从而代替人脑高效地处理信息[1]。

在人类接收到的视觉信息中，既包含静止不动的物体也包含运动的物体，人类往往更加关注运动的物体，所以运动目标检测备受各位专家学者的青睐。运动目标检测是将发生位移的物体从背景中准确地分离出来并进行标记的过程。运动目标检测是目标识别与跟踪、目标行为分析与场景感知等高级视频图像处理的前期基础，检测结果会对后期高级图像处理的进行产生直接影响[2]。运动目标检测的本质是提取感兴趣的区域以减小后续图像处理的运算量，从而提高效率。多运动目标的实时检测是计算机视觉中极其重要的一个任务。由此可见，研究多运动目标的实时检测有着重要的理论和现实意义。

运动目标检测技术广泛应用于民用领域、工业领域、军事科技等领域。在民用领域，运动目标检测技术主要用于智能安防监控系统。运动目标检测技术可以使人不再需要长期集中注意力查看监控视频，既节约了雇佣劳动人员的成本，又可以提高实时性和准确性。在智能交通领域，运动目标检测技术可以统计当前道路上的车流量、车密度、车速、来往车辆等信息，可以很好地检测交通事故或者车辆故障，一定程度上可以减少交通事故和交通拥堵的发生。运动目标检测对于工业检测也有着重要意义。在军事科技领域，运动目标检测也发挥了不可替代的作用，例如军事侦察、武器自动瞄准、导弹跟踪等。

目前，运动目标检测算法大都是使用 PC 上的 MATLAB 与开源计算机视觉库（Open Source Computer Vision Library，OpenCV）等工具开发的，但是 PC 存在便携性差、专用性不强、功耗高等缺点。相对于传统通过串行处理图像信息的硬件平台，选择能够并行处理的硬件才可以应对高速数据传输现状。FPGA 采用多级流水线技术，运算速度快，适合视频图像处理中像素级的运算。但是基于 FPGA 的运动目标检测系统的检测结果难以直接利用，需要额外的手段把 FPGA 处理完的数据传输给处理器。ARM 架构灵活性高、控制能力强，但是由于其串行处理的弊端，面对视频图像处理庞大的数据量时不能满足实时性的要求。

Zynq-7000 的可编程逻辑部分具有 FPGA 并行运算速度快的优势，处理系统部分具有 ARM 灵活度高的优势，采用 AXI（Advanced eXtensible Interface）标准总线实现两者的互联使得通讯便捷、集成度高，并且两者共用内存使得数据交互方便。与传统的嵌入式硬件相比，在性能、成本、功耗等方面具有明显的优势。

综上所述，把运动目标检测技术和 Zynq 技术结合在一起，提出基于 Zynq 的运动目标检测技术的嵌入式系统方案，从而实现低功耗的实时检测系统，在民用、工业、军事科技等领域有着广泛的商业价值和应用前景。

1.2 国内外研究现状

1.2.1 运动目标检测算法的研究现状

运动目标检测技术在民用领域、工业领域、军事科技等领域被广泛运用，引起了一大批国内外学者的关注。在过去的几十年里，人们通过大量的深入研究，已经提出了许多行之有效的算法。

1981 年，Horn 和 Schunck 建立了光流约束方程将速度场与图像的灰度联系了起来，奠定了光流法的基础，并在光流约束方程中添加光流平滑的约束条件，建立了 H-S 光流法[3]。Lucas 与 Kanade 利用相邻像素点具有相同运动的特点建立了 LK 光流法，解决了光流方程模糊性的问题，但是不能计算均匀区域的光流信息，而且计算量比较大，实时性较差[4]。此外还有金字塔光流法、区域光流法和特征光流法等[5]。Gomaa 等人提出了一种卷积神经网络和光流法相结合的车辆检测和计数算法，并取得了不错的检测和计数精度[6]。Sengar 通过计算连续三帧图像的光流有效地检测到运动物体，但是该方法时间复杂度较高[7]。光流法具有较高的检测准确率，但是计算复杂度高、实时性差、在硬化上实现较为困难，这些缺点限制了它在实际工程中的应用。

背景减除法通过当前帧与背景差分处理来检测运动目标。对于背景减除法来说背景的建模至关重要，为保证检测实时性并获得良好的检测效果，出现了多种背景建模的方法。Wren 提出了单高斯的背景建模方法，该方法实时性有所提高，但是在复杂背景下检测效果不好[8]。Stauffer 等人提出了基于统计样本像素信息的混合高斯模型背景建模法，在复杂背景下取得了不错的检测效果，但是该方法计算量大，实时性不好[9]。Barnich 等人提出了视觉背景提取算法（Visual Background Extractor，ViBe），使用单帧图像的数据来完成背景模型的初始化，并随机选择领域中的像素值作为该点像素的模型样本值，该方法减小了背景初始化的运算量，提高了实时性，但是当用来初始化的图像中存在运动物体时，检测结果中会出现拖影区域[10]。Mabrouk 等人提出了一种混合高斯模型背景并行化

的算法，大大提升了该算法的运行速度[11]。

1979 年，知名学者 Jain 在文献[12]中提出了帧间差分法的基础理论。随后，Lipton 等人正式提出了通过连续图像的像素差异化来检测运动目标的方法[13]，即选取相邻的两帧，通过对应像素点差分得到运动目标，该方法运算速度快，但是会出现空洞、重影等现象。Hsu.Y.Z 等人在 1984 年提出了三帧差分法[14]，即通过相邻两帧图像进行差分、二值化，然后将两个二值化的结果进行逻辑与运算，该方法解决了两帧差分法检测到的目标所在区域比运动目标大的问题。Abughalieh 和 Sababha 等人将帧间差分法与彩色投影法相结合，节省了大量计算资源，实现了低细节目标的检测[15]。Husein 和 Halim 等人使用背景减除法完善了帧间差分法，使得算法的有效性和检测精度得到了一定程度地提高[16]。Wang S 和 Kong P 利用帧间差分法提取最原始的运动特征，然后通过卷积神经网络进行行为分析，取得了良好的检测精度[17]。Nallasivam 和 Senniappan 等人将背景减除法和帧间差分法相结合识别感兴趣的区域，然后在感兴趣区域选择一个质心像素点进行目标跟踪，该算法降低了算法的时间复杂度，提高了检测精度[18]。

相对于国外，国内在运动目标实时检测的研究起步较晚，但也取得了一定的研究成果。Sun W 等人提出了光流法与免疫粒子滤波相结合的运动目标检测方法，该方法能较好地解决阴影的影响，然而该方法受光照条件的影响较大[19]。乐英等人在 MATLAB 上采用背景减除法实现了多个运动目标的检测与分割，得到了较为完整的运动目标[20]。Liu Ling 使用历史数据来完成背景模型的初始化，并结合领域的像素来更新二值化阈值，该算法解决了 ViBe 出现的“鬼影”现象。李明月使用暗通道图像对噪声进行过滤，然后采用保守更新与计数更新相结合的背景更新方法，检测到的运动目标相对完整，但是图像整体偏暗，对光线比较敏感[22]。何银飞融合了改进的帧间差分法和改进的背景减除法，提高了算法对复杂环境的适用性[23]。王梦菊等人提出了一种将帧间差分与背景减除法相融合的运动目标检测算法，提取到的运动目标比较完整，并且噪声小，对光线等环境因素有一定的适应性[24]。张丽平等人将金字塔光流法与帧间差分法相融合，实现了运动车辆的检测[25]。赵淑胜提出了一种将颜色信息与纹理信息以置信系数相融合的运动目标检测的算法，该算法可以很好地适应光照变化的影响[26]。Zhang Wei 提出了一种基于时间一致性的邻域帧搜索去噪算法，实现了较强背景噪声干扰下运动目标的检测[27]。

1.2.2 视频图像处理平台的发展现状

传统运动目标检测的算法主要使用 PC 上的 MATLAB、OpenCV 等软件工

具，在 PC 上的理论研究推动了运动目标检测技术的发展与进步，但是 PC 平台存在功耗高、便携性差、专用性不强等缺点，难以胜任一些对功耗、体积敏感的应用场景。运动目标检测算法需要搭载在小型化、功耗低的嵌入式平台上才能应用到实际中，所以在嵌入式平台上实现运动目标检测具有很大的研究价值。国内外对嵌入式视频处理平台做了大量的研究。

Cho J 等人在 FPGA 上融合光流法和高斯混合模型实现了动态背景下的目标检测，并达到了 30fps 的处理速度[28]。Kumar 等人将高斯平均算法搭载在 ZedBoard 开发板上，利用硬件资源实现了背景减除法，取得了良好的检测效果[29]。Yi Q 在 FPGA 平台上优化了混合高斯模型的学习率，提高了算法的实时性，能够对分辨率为 640*480，30fps 视频信号流进行实时处理[31]。Sridevi N 和 MMeenakhi 等人在 FPGA 平台上将背景减除法与 Harr 小波相融合，以较少的硬件消耗实现了分辨率为 1280×720 视频的实时检测[32]。Correia 和 Campilho 在 FPGA 平台实现了光流法，对分辨率为 252×316 的视频处理时间降低为 47.8ms，但仍不能满足实时性的要求[33]。

郑淋萍在数字信号处理（Digital Signal Processing，DSP）上结合颜色和分形特征和改进的 ViBe 算法实现了复杂背景下运动目标的检测，算法总耗时为 39.62ms[30]。Zhang S 等人在 DSP6678 上使用特征点配准的方法实现了红外弱小目标的检测，满足一定的实时性[34]。高文刚等人在 TMS320DM642 DSP 上使用 ViBe 算法提取运动目标信息，有效解决了背景建模慢的问题，该系统能够提取到较为完整的运动目标[35]。

Jianhui Gao 在嵌入式 ARM-Linux 系统上结合 ORB（Oriented FAST and Rotated BRIEF）特征检测算子对目标中心进行估计，该系统算法具有一定的抗光照和背景色干扰的能力，提高了算法的检测精度[36]。宋翰林在 ARM 平台上采用角点特征匹配来估算全局运动参数，然后使用帧间差分法检测运动目标，实现了运动目标的实时检测，但是该算法在复杂环境下检测效果不好[37]。何国锋在 S3C6410 ARM 处理器上使用改进的三帧差分法设计了运动目标检测系统，该系统可以有效地检测到缓慢运动的物体[38]。

邵鹏、杨晨等人在 Altera 的 Cyclone_IV 平台实现了利用颜色空间中的亮度值动态调整阈值，解决了帧间差分法阈值不能自动调节的问题[39]。张棋在 FPGA 上实现了先边缘检测后背景减除法的运动目标检测算法，帧率达到了 25fps[40]。林培杰、郑柏春等人在 FPGA 上将帧间差分法和背景减除法相结合，实现了 640*480，30fps 视频信号流的运动目标实时检测，并且提出了分区域的矩形框标记法，实现了多运动目标的检测，但是分区域的矩形框标记法具有很大局限性

[41]。邢凯在 Spartan6 FPGA 平台使用帧间差分法，提出了边缘标记法，实现了多运动目标的检测，但是边缘标记法效果不够明显[42]。郭铮等人在 FPGA 上融合了三帧差分法和边缘检测算法，在高清视频下提取了运动目标的轮廓，并且实时性较好，但是显示结果为二值化图像，显示效果不佳[43]。王俊龙在 FPGA 上实现了四帧差分法，解决了帧间差分法易于产生空洞、双影的问题，但是其标记方法只适用于单个运动目标，并且检测结果为灰度图，视觉效果不好[44]。

周佩在 Zedboard 开发板上采用帧间差分法，实现了低功耗、高性能的运动目标检测系统，但是只实现了单运动目标的检测，并且检测结果为二值化化图像，视觉效果不好[45]。李鹏在 Zynq 上使用改进的 ViBe 算法设计了运动目标检测系统，该系统具有较低的功耗，能够完整地检测出运动目标，但是检测结果为二值化图像，视觉效果不好[46]。陈科成等人在 Zynq 上使用帧间差分法对分辨率为 640*480 的视频图像能以 60fps 速度识别单个运动目标，功耗仅为 1.5W 左右，但是检测结果为灰度图，视觉效果不好[47]。张祖昊和王云光在 Zynq-7000 芯片上实现 1280×720 分辨率下 30fps 的实时目标的边缘检测，为后续运动目标的实时、准确跟踪打下了基础[48]。张俊杰提出了将 Census 变换信息、边缘信息、颜色信息相融合的立体匹配算法，并在 Zynq 上实现了该算法，提高了匹配精度，但是系统的处理帧率仅为 16fps，无法满足实时性的要求[49]。

从上述文献中可以看出，目前运动目标检测平台主要分为以下几种：

(1) 基于 PC 机平台。采用 PC 机上的图像处理软件（主要为 OpenCV 和 MATLAB）进行运动目标检测。PC 机具有强大的运算能力，数据处理速度快，效率高，但是 PC 机存在功耗高、价格昂贵以及体积大不易于携带等缺点。目前，PC 机主要用于图像算法的验证环节。

(2) 基于 DSP 平台。DSP 采用哈佛结构，可以同时获取指令和读取数据，实现多级流水线处理，运算速度较快。但是其本质上是串行执行系统，流水线的级数和结构固定，硬件无法实现重构，因此运算能力有限。

(3) 基于专业集成芯片（Application Specific Integrated Circuit，ASIC）平台。专业集成芯片处理速度非常快，体积小，功耗低，可靠性高，但是其开发周期比较长，设计成本高，可扩展性差，灵活性不好，只适用于大批量生产。

(4) 基于 ARM 平台。ARM 体积小，成本低，性能高，功耗低，具有丰富的外设，支持主流的嵌入式操作系统，指令执行速度快，可以使用高级语言编程，灵活度高。但是 ARM 本身的串行运算的结构特性限制了它在视频数据处理方面的应用。

(5) 基于 FPGA 平台。FPGA 内部逻辑资源丰富，并且并行运算速度快，可

以实现硬件算法加速，特别适合像素级别的运算。但是 FPGA 对开发者要求很高，有些复杂的算法在 FPGA 上实现难度较高，因此处理结果难以直接利用，需要处理器的配合。

(6) 基于 Zynq 平台。Zynq 中集成了 FPGA 与 ARM，兼具 FPGA 与 ARM 的优点，带来性能、成本、功耗等优势。Zynq 可以采用软硬件协同设计的方法发挥可编程逻辑与处理系统的独特优势，因此特别适合视频图像处理。

1.3 本文主要研究内容及组织结构

本文的主要研究内容是在 Zynq 硬件平台的研究基础上，完成多运动目标的实时检测与标记，并且系统满足低功耗的要求，对光照变化有一定的适应能力，在复杂环境中可以准确的检测到运动目标。系统主要由图像采集模块、算法处理模块、数据存储模块、图像显示模块组成。通过 OV5640 摄像头采集图像，传递到 Zynq 芯片中进行处理，最终通过 HDMI 接口实时显示检测结果。

论文的组织结构如下：

第 1 章：绪论。阐述本文的研究背景与意义，分析运动目标检测算法和视频图像处理平台的研究现状，最后总结本文的主要内容以及论文的组织结构。

第 2 章：硬件平台及关键技术介绍。首先对硬件平台 Zynq-7020 进行简介，其次为后续提到的技术做出一定的理论铺垫，最后介绍设计工具、设计方法。

第 3 章：介绍当前运动目标检测的几种算法，然后对几种算法进行对比，选取合适的运动目标检测算法；介绍数字图像处理相关的理论基础，并使用软件对其进行仿真；设计运动目标检测算法 IP 核，使用 Verilog 语言分别实现灰度化、中值滤波、差分处理、二值化、形态学滤波；优化运动目标的标记方法，实现对多个运动目标的标记。

第 4 章：系统的总体设计。首先分析系统的工作流程，阐述各个模块的作用。其次调用相关 IP 核完成硬件平台的搭建。接着分别实现图像采集模块、数据存储模块、图像显示模块，并集成到系统中完成硬件部分的设计。最后在处理系统中完成 OV5640 摄像头的初始化、VDMA IP 核、内存的分配。

第 5 章：系统调试与结果分析。首先对系统的功能进行验证，对比传统 FPGA 运动目标检测标记方法，并对实验结果进行分析。然后分析系统的资源利用率、功耗和实时性。最后制作启动镜像实现程序固化。

第 6 章：总结与展望。对本文涉及的工作内容进行总结，并提出了未来需要进一步完善的地方。

第2章 硬件平台及关键技术

2.1 硬件平台 Zynq-7020

本系统采用的硬件平台为 Zynq-7020 开发板。Zynq 是赛灵思公司（Xilinx）将软件可编程性（ARM）与硬件可编程性（FPGA）进行结合的产物。作为一种全可编程片上系统，Zynq 为高端嵌入式应用提供了强大的处理能力与计算性能，同时兼具灵活性、可扩展性、高性能、低功耗等优点[50]。Zynq 由 PS 和 PL 两部分组成。Zynq 芯片的内部结构如图 2-1 所示。

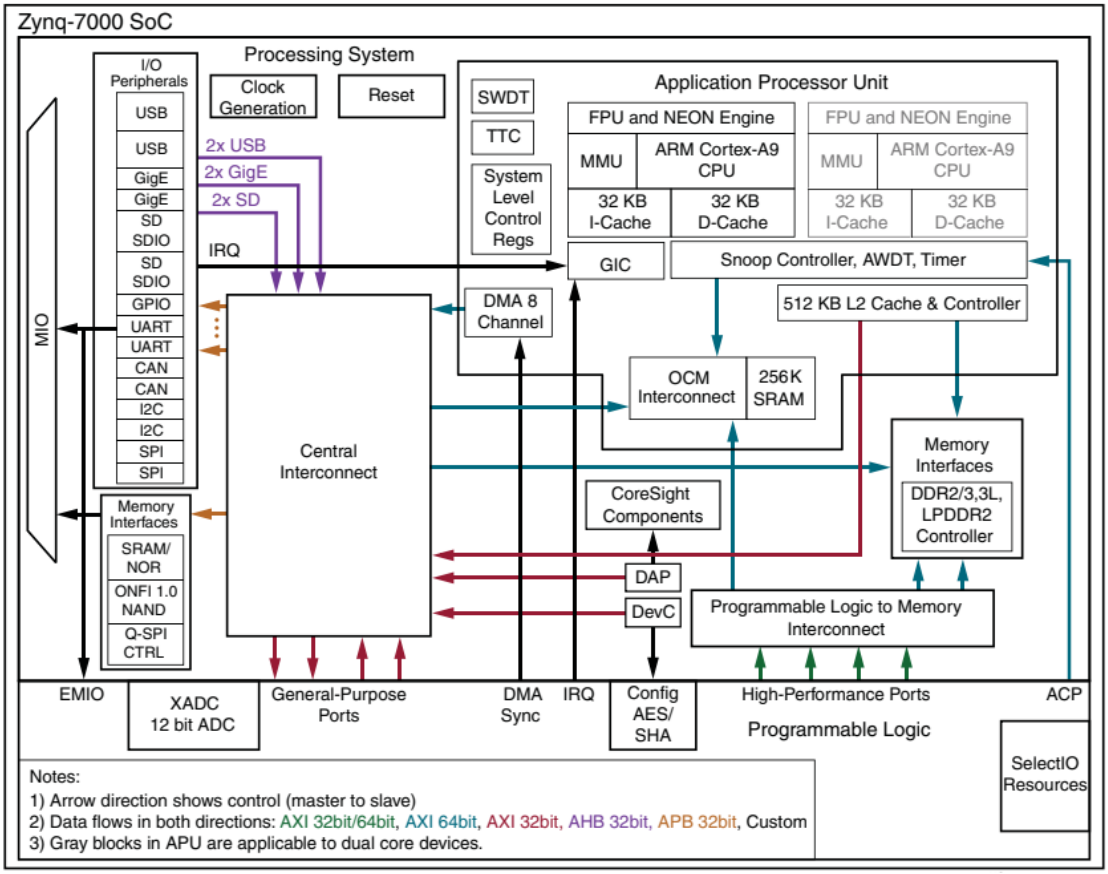


图 2-1 Zynq 芯片的内部结构

Fig. 2-1 Internal structure of Zynq chip

处理系统包含完整的 ARM 处理器系统，可以保证开发板的运行速度，可以在上面运行 Linux 等主流操作系统[51]，并且集成了内存控制器、丰富的可扩展外设接口。

可编程逻辑部分等价于 Xilinx 7 系列的 FPGA，不仅降低了开发板的功耗，而且降低了开发板的制作成本。FPGA 采用流水线逻辑体系结构，数据流处理低延时，高性能，适合并行运算的算法，从而实现硬件算法加速[52]。

Zynq 采用高性能、高带宽、低延迟的 AXI 协议将处理系统与可编程逻辑在单芯片上紧密结合，实现了两者的强耦合，这意味着可以采用软硬件协同设计的方法使软件和硬件同时发挥其最高性能[53]。与两者分离系统相比，这种模式在通信上的开销大量减少，并且可以减小体积、降低功耗和设计风险、增加设计灵活性。Zynq 具有 9 个 AXI 接口，每个接口由多个通道组成，这些接口形成了处理系统内部的互联以及处理系统与可编程逻辑的互联。

本文介绍的基于 Zynq 的运动目标检测系统所使用的 Zynq 开发套件是正点原子公司生产的 Zynq-7020 领航者开发板，其实物图如图 2-2 所示。

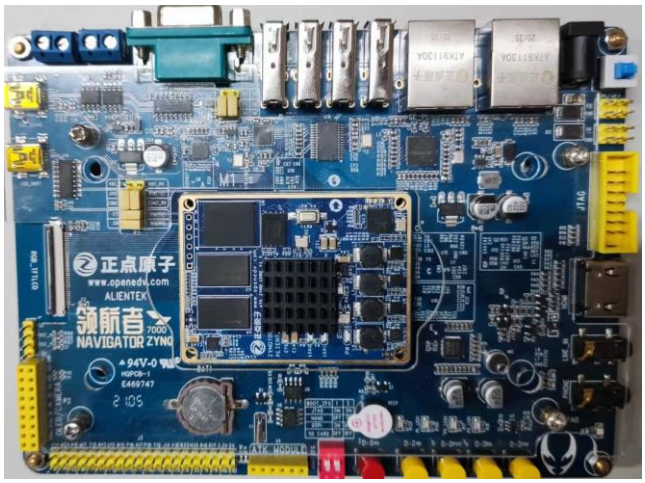


图 2-2 Zynq-7020 开发板

Fig. 2-2 Zynq-7020 development board

2.2 AXI Stream 协议及视频流格式

AXI Stream 协议中包括 tvalid、tready、tuser、tlast、tdata 五根信号线，其中最重要的信号线是 tvalid 和 tready。tvalid 由发送端控制的，当 tvalid 为高电平时表示 tdada 上的数据是有效的；tready 由接收端控制，当 tready 为高电平时，表示接收端已经准备就绪。只有当 tvalid 与 tready 信号同时为高电平时，数据才能有效传输。因为数据传输是连续的，并且不会出现锁死的状态，所以 AXI Stream 协议具有很高的效率。当 tlast 为高电平信号时表示一个数据包已经传输完成。tuser 是自定义信号，可以根据用户的需要自行定义其含义。

Xilinx 公司提供的所有视频图像处理的 IP 核都基于 AXI Stream 视频流协议[54]。不同于普通 AXI Stream 协议的是，tuser 与 tlast 有了新的定义。当 tuser 为高电平时表示一帧图像的第一个数据开始传输，当一帧图像的第一个数据传输完成后，tuser 立刻变为低电平。当 tlast 为高电平时表示一行数据的传输即将结束，当一行数据的最后一个像素传输完成后 tlast 立刻变为低电平。

2.3 DDR3

DDR 的全称为 Double Data Rate SDRAM，即双倍速率的同步动态随机存取内存。相对于传统的 SDRAM（Synchronous Dynamic Random-Access Memory，同步动态随机存取内存）只在信号的上升沿传输数据，DDR 可以在信号的上升沿与下降沿都传输数据，具有双倍的传输速率。DDR3 是第三代的 DDR，相较于前一代的 DDR2，具有更高的工作频率、更低的功耗，传输性能也更高。

Zynq 通过 DDR 控制器来控制外部 DDR 存储器。Zynq-7020 板载的两片 DDR3 内存总容量为 1GB，最高运行速度可达 533MHz，采样率可达 1066Mbps，理论上可以提供 30.7Gbit/s 的物理带宽。这样高的存储带宽使得 Zynq-7020 能够轻松应对各种大内存、高带宽场景需求[55]。此外，DDR3 内存也作为 PS 端处理器的运行内存。由于 DDR3 是 PS 部分的存储接口，因此 PL 逻辑需要通过 AXI 接口访问 DDR3。

2.4 VDMA

直接存储器访问（Direct Memory Access，DMA）是一种数据传输更快的内存访问技术。VDMA 可以看作是一种专门视频图像处理的 DMA，其内部集成了视频处理所用到的帧缓存与同步锁相等功能，在 Zynq 上使用 VDMA IP 核可以简化对系统存储器（DDR3）的访问[56]。在视频图像处理中，一般采用帧缓存来解决图像的输入源和图像显示的速率不匹配的问题，同时也有利于后续的图像处理。一帧图像的大小一般会大于 BRAM 的存储容量，因此选用 PS 端的 DDR3 来作为帧缓存。Xilinx 所有与视频图像处理的 IP 都遵循 AXI4-Stream 协议，因此在做视频图像处理时为保持 IP 的通用性，数据在 PL 端也遵循 AXI4-Stream 协议。而数据只有转化为 Memory Map 格式才能正常写入系统存储器中，并且从系统存储器读出的数据也是 Memory Map 格式，这个时候就需要一个模块来完成 AXI4-Stream 格式与 Memory Map 格式的相互转换，VDMA IP 就是可以完成上述的转换的模块。VDAM 数据接口分为读、写两个通道，通过写通道将 AXI-Stream 类型的数据流写入系统存储器中，通过读通道从系统存储器中读取写通

道写入的数据以 AXI-Stream 类型输出，从而实现视频处理 IP 核与系统存储器之间的高速数据交互。VDMA 的框图如图 2-3 所示。

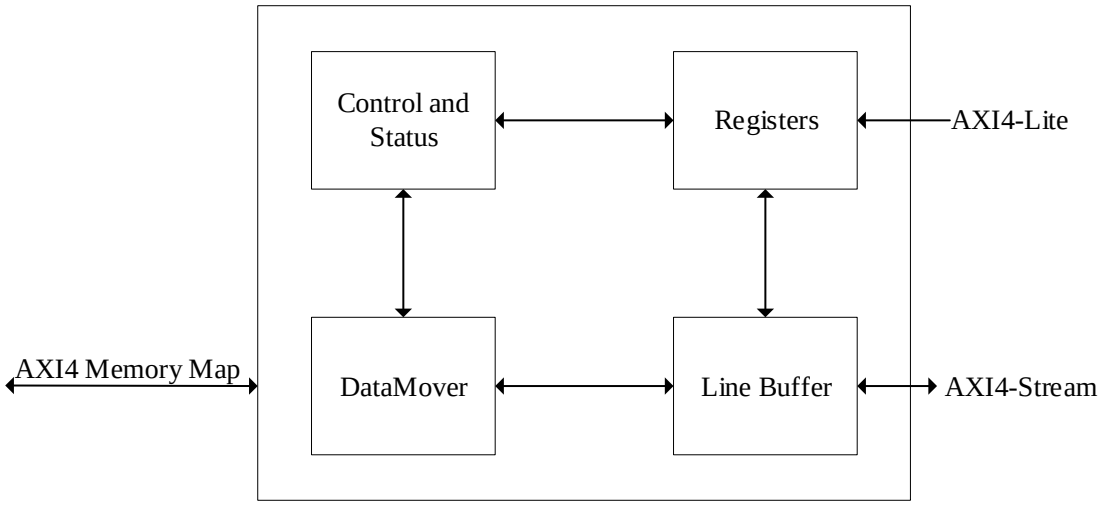


图 2-3 VDMA 的框图

Fig. 2-3 Block diagram of VDMA

VDMA 的接口主要有 AXI4-Lite、AXI4-Memory Map 和 AXI4-Stream 三种。处理系统的通用 AXI（General Purpose AXI，AXI_GP）接口通过 AXI 互联模块连接 VDMA 的 AXI4-Lite 接口，从而实现处理系统对 VDMA 的寄存器进行读写操作，最终达到对 VDMA 的动态配置以及状态获取的目的。AXI4-Memory Map 接口分为写通道存储器端映射（M_AXI_S2MM）、读通道存储器端映射（M_AXI_MM2S）两个接口，通过 AXI 智能互联模块与处理系统的高性能端口（High Performance Ports，AXI_HP）连接，从而实现对存储器（DDR3）的高效访问。AXI4-Stream 接口分为读通道 AXI-Stream 端映射的 AXI4 接口（M_AXIS_MM2S）、写通道 AXI-Stream 端映射的 AXI4 接口（S_AXIS_S2MM），分别用于输出到外设、从外设输出 AXI_Stream 格式的数据流。从 VDMA 的框图可以看出，VDMA 内部由寄存器、数据搬运模块、行缓冲模块三部分组成。AXI4-Stream 格式的数据流进出系统存储器都要经过行缓冲模块进行缓存，然后相关寄存器控制数据搬运模块写入或读出数据。

2.5 OV5640 摄像头

本系统中图像采集所使用的 OV5640 摄像头模组是由豪威科技公司生产的 CMOS 图像传感器，可靠性高，采样速率快，相对于 CCD 传感器功耗更低。

OV5640 摄像头模组实物图如图 2-4 所示。

图 2-4 OV5640 摄像头模组实物图

Fig. 2-4 OV5640 camera module physical picture

OV5640 摄像头具体特点如下：

(1) 支持 YUV、YCbCr422、RGB565 以及 JPEG 等多种输出格式；

(2) 工作功率在 150mW-200mW 之间；

(3) 最高可实现 500 万像素的分辨率（15fps）的采集；

(4) 最高可达到 90fps 的帧率（640*480）的采集；

(5) 支持多种分辨率的视频输出；

(6) 可对采集的图像进行补偿。

2.6 HDMI

HDMI 指的是高清多媒体接口，能够同时传输数字音频数据和高分辨率视频数据，简化了设备之间的接口和连线[57]。新一代 HDMI 理论带宽高达 48Gbps，支持更高的分辨率和刷新率，已经逐渐替代了传统的视频图形阵列（Video Graphics Array，VGA）、数字视频接口（Digital Visual Interface，DVI），成为了新一代的多媒体接口标准，在生活中的音频、视频的传输中经常可以看到它的身影。HDMI 接口协议在物理层使用差分信号高速传输串行的音频和视频数据。

2.7 设计工具

Xilinx 为 Zynq 设计提供了一系列的工具，即 Vivado 设计套件（Vivado Design Suite）。在对 Zynq-7020 开发板进行研究的过程中，主要用到了以下两个工具：

Vivado IDE：主要进行片上系统中硬件部分的设计。主要通过硬件描述性语

言进行开发，也可以调用官方 IP 核进行系统级的设计以缩短开发周期。软件内集成了封装 IP 的功能，可以封装自定义的 IP 核，添加到 Vivado 的 IP 库中，方便模块的重复利用，节约开发时间。创建工程、完成设计输入后，即可进行分析、综合、约束输入、实现、生成比特流.bit 文件等操作。将比特流文件下载到对应的开发板就完成了 FPGA 的设计，但是对于 Zynq 来说，只完成了硬件的部分，还需在 SDK（Software Development Kit）中完成处理系统部分的开发。Vivado 和 SDK 软件之间有交互，可以很方便地将比特流文件导出到 SDK，并一键启动 SDK。

SDK：软件开发环境，通过 SDK 对 Zynq 处理系统部分进行设计，以及整个工程的下载测试、程序烧录等。SDK 内部包括 JTAG（Joint Test Action Group，联合测试工作组）调试器、闪存编程器、官方 IP 的驱动和板级支持包等资源。

在本次设计中，使用的版本为 Vivado 2018.3，其性能较为稳定。图 2-5 是 Zynq 开发流程，大致可以分为 6 步。其中在 Vivado 软件中完成前 4 步的硬件部分，在 SDK 软件中完成软件部分和板级调试。

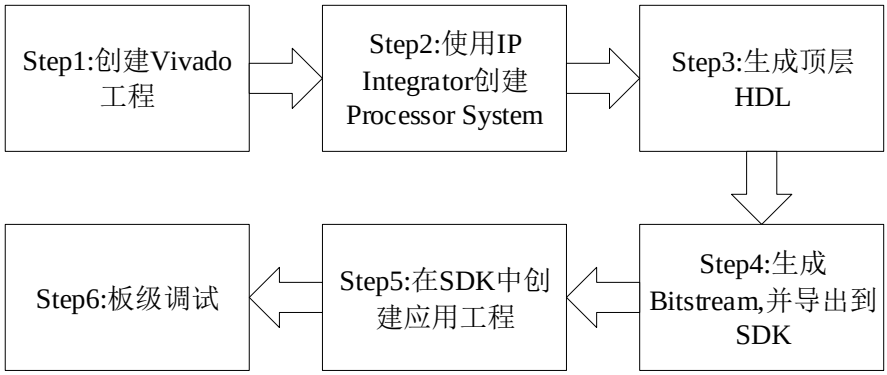


图 2-5 嵌入式系统开发流程

Fig. 2-5 Embedded system development process

2.8 设计方法

2.8.1 模块化设计

模块化设计是系统设计的一个重要技巧，在对一个复杂的数字系统进行设计时，一般使用从顶层向下设计的方式将整个系统合理地划分为相互独立的功能模块，每个功能模块又可以划分为更小的子模块。由于各个子模块之间相互独立，当更改某一子模块时不会影响其他子模块的功能[58]。模块化设计可以化繁为

简，并且有利于代码的测试与维护，同时也增加设计的通用性，即各个子模块可以方便地重复利用。在系统顶层只需分别调用各个功能模块、设置顶层参数即可。模块化设计的功能框图如图 2-6 所示，将系统划分成一个个的子模块。

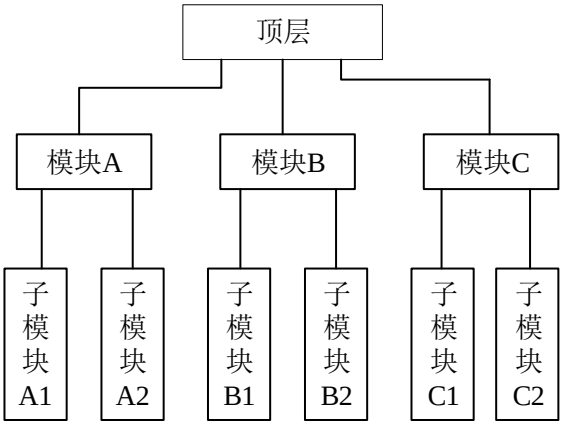


图 2-6 模块化设计的功能框图

Fig. 2-6 Functional block diagram of modular design

2.8.2 软硬件协同设计

Zynq-7020 分为处理系统和可编程逻辑两部分，既可以在处理系统上实现软件编程，又可以在可编程逻辑部分实现硬件编程，因此可以采用软硬件协同设计的方法对 Zynq 进行开发。采用模块化设计定义完功能单元后，根据软件与硬件各种擅长的领域将功能模块合理地分配到软件和硬件中。处理系统灵活度高、控制能力强，适合完成一些顺序执行的任务；可编程逻辑采用多级流水线并行处理速度快，更适合数据流计算等任务[59]。采用软硬件协同设计的方法，既可以保留处理器系统灵活性高的优点，又可以使用可编程逻辑进行硬件算法加速，充分利用软硬件资源，提高系统性能。

Zynq 片上系统设计流程如图 2-7 所示。在将功能模块划分到处理系统（软件）和可编程逻辑（硬件）后，先在 Vivado IDE 中完成硬件部分的，然后导出结果到 SDK 中再完成软件部分的设计，集成测试不通过则对软硬件部分进行修改，直至测试通过为止，最终完成系统集成并进行最终测试。

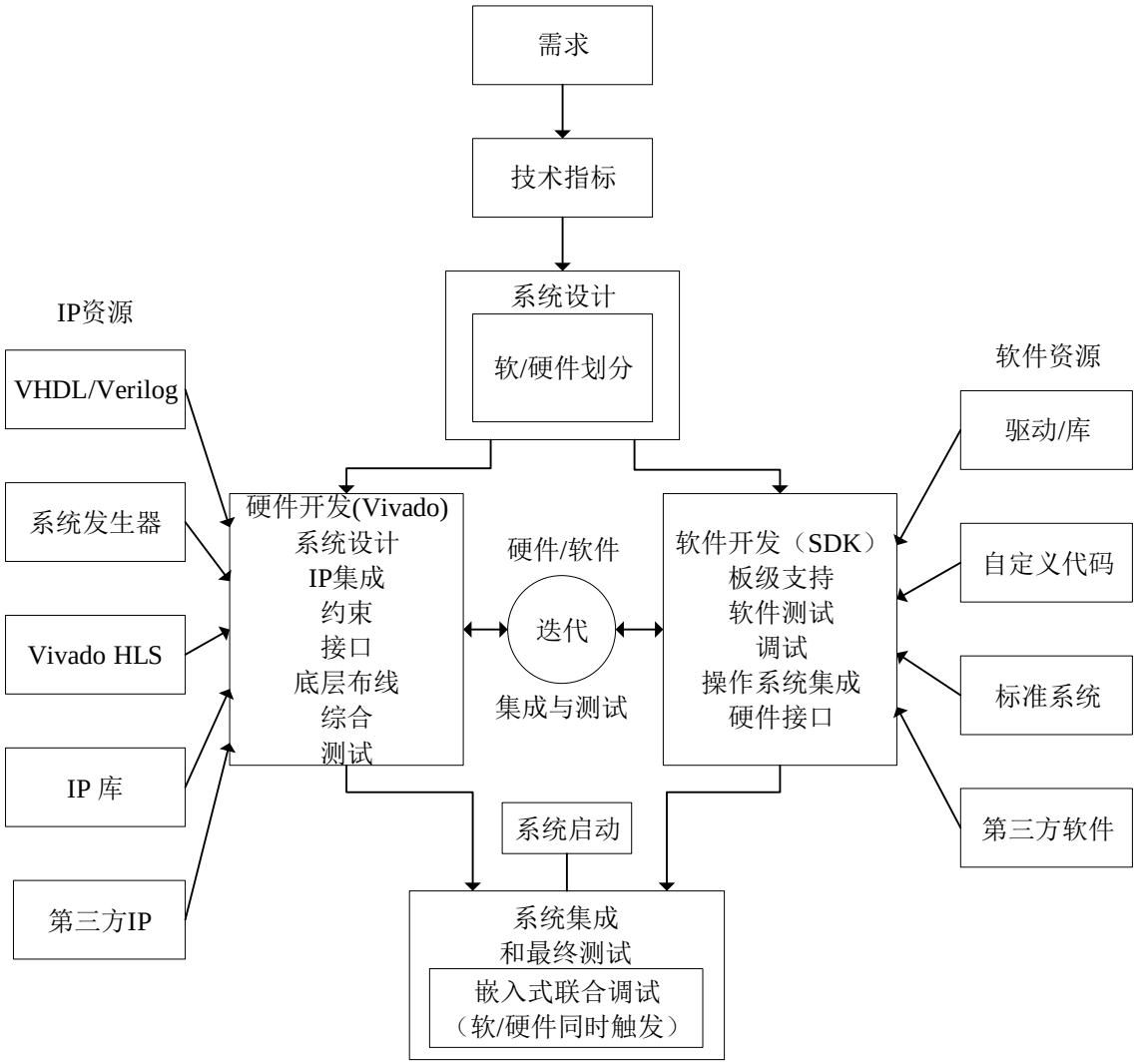


图 2-7 Zynq 片上系统设计流程

Fig. 2-7 Zynq system on chip design process

2.9 本章小结

本章主要介绍了硬件平台 Zynq-7020、V5640 摄像头以及用到的一些关键技术，例如 AXI Stream 协议及视频流格式、DDR3 及本系统将数据存储到系统内存（DDR3）中使用的 VDAM IP 核。此外还介绍了 Vivado IDE、SDK 等系统设计工具、模块化设计和软硬件协同的设计方法。

第3章 检测算法分析以及 IP 核的设计

3.1 运动目标检测算法分析

每一种运动目标检测算法都有各自的优势和需要改进的地方，都有其适用的领域，目前还没有哪一种算法能够适合所有情况。因此，在进行系统设计前选择一种检测效果好、使用硬件容易实现的算法至关重要。

3.1.1 光流法

物体发生位移时，它在成像平面上对应的像素点发生了相应的变化，这种由像素点表现的运动就是光流。光流包含了物体发生位移的信息，因此可以用来检测运动目标。光流法图像的灰度随时间的变化与速度场联系起来建立光流方程，加上梯度、特征、能量、相位等约束条件中的一个来求解光流矢量，进而分析物体运动的信息的方法。光流法可以很好地适应场景变化的影响，即使在动态背景下仍然可以检测到运动目标，具有较高的检测准确率。但是光流法要求亮度一直保持不变和运动距离小，适用条件苛刻，因此光流法对光线变化较为敏感，不适合运动目标运动过快的情况。此外，该算法需要计算每一个像素点的偏移量，算法复杂度高，计算量大，实时性差[60]，在硬件上实现较为困难，这限制了它在实际工程的应用。

3.1.2 背景减除法

背景减除法将当前帧与背景帧进行差分运算来检测运动目标，差别较大的区域被判定为运动目标所在区域。背景减除法的检测效果依赖于背景的建模，背景建模越好，检测效果就越好，背景更新越快，系统的实时性就越高[61]。常用的背景建模算法有均值法[62]、混合高斯背景建模法[63]。均值算法相对简单、运算量小，实时性高，但是当背景发生变化时检测效果不好，只能用于背景比较稳定的场景中。混合高斯背景建模法对背景变化不敏感，但是算法相对复杂，运算量大，实时性较均值算法较低。背景减除法的流程图如图 3-1 所示。

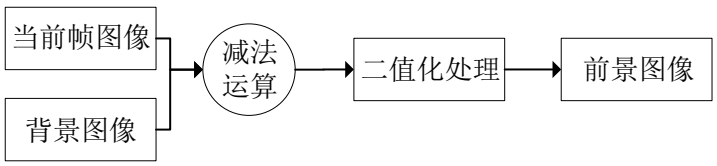


图 3-1 背景减除法的流程图

Fig. 3-1 Flow chart of background elimination method

背景减除法可以用式(3-1)进行描述。

$$X_t(x,y)=\begin{cases}255, & \left|I_t(x,y)-B_{t-1}(x,y)\right|>T\\ 0, & \left|I_t(x,y)-B_{t-1}(x,y)\right|\le T\end{cases} \tag{3-1}$$

式中，$X_t(x,y)$表示 t 时刻的提取出来的运动目标图像的二值化图像，$I_t(x,y)$表示t时刻的视频帧，$B_{t-1}(x,y)$表示$t-1$时刻的背景图像，T表示分割阈值。

3.1.3 帧间差分法

帧间差分法与背景减除法同为图像差分算法，但是帧间差分法不需要对背景进行建模，而是采用相邻两帧做差分运算，因此运算速度更快。图像传感器采集到的视频图像序列是连续的，即相邻帧具有相关性。一般情况下，物体与背景会存在颜色差距，灰度化后会反映到亮度信息上，即物体与背景的灰度等级不同。当视频图像序列中存在发生位移的物体时，相邻两帧图像相同像素坐标的灰度值作差，并取差值的绝对值，背景差分后的值较低，而运动物体发生了位移，差分后的灰度值为运动物体与背景的灰度值之差，这个值明显不为 0，这时选取一个合适的分割阈值，就可以将运动物体从背景中分离出来。帧间差分法原理简单，容易使用硬件实现，实时性好，对光线变化也有一定的适应能力，鲁棒性好，在场景中存在多个运动目标的情况也同样适用，但是当运动目标距离摄像头太近时二值化图像会出现内部空洞，当运动目标运动的速度太快时二值化图像会产生重影，只能提取出运动目标的轮廓信息[64]。帧间差分法的流程图如图 3-2 所示。

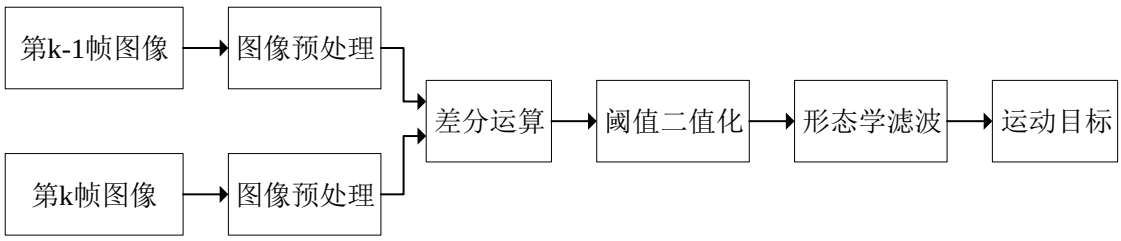


图 3-2 帧间差分法流程图

Fig. 3-2 Flow chart of inter-frame difference method

取相邻两帧图像相同像素坐标的灰度值作差，并取其绝对值，得到差分图像，如式(3-2)所示：

$$D_{\mathrm{n}}(x,y)=\left|\mathrm{f}_n(x,y)-f_{n-1}(x,y)\right| \tag{3-2}$$

式中，$D_{\mathrm{n}}(x,y)$为差分图像，$\mathrm{f}_n(x,y)$和$f_{n-1}(x,y)$分别表示视频图像序列第 n 帧和第$n-1$帧图像对应像素点的灰度值。选取合适的二值化阈值，对差分后的图像进

行二值化处理，如式(3-3)所示：

$$R_n(x,y)=\begin{cases}255, D_{\mathrm{n}}(x,y)>T\\0 \quad ,else\end{cases} \tag{3-3}$$

式中，T 为选取的阈值，$R_n(x,y)$ 为二值化图像，$D_{\mathrm{n}}(x,y)$ 为差分图像。帧间差分法的流程图如图 3-2 所示。

3.1.4 常用运动目标检测算法比较

以所得目标信息、鲁棒性、复杂度、实时性为指标，对三种常用运动目标检测算法进行了比较，如表 3-1 所示。

表 3-1 运动目标检测算法对比

Tab. 3-1 Comparison of moving target detection methods

运动目标检测算法	光流法	背景减除法	帧间差分法
所得目标信息	位置、大小、形状	位置、大小、形状	轮廓信息
鲁棒性	一般	差	较好
复杂度	大	中	小
实时性	差	较好	最好

综合考量后，决定使用复杂度小、速度快、适应能力强的帧间差分法。

3.2 数字图像处理技术

受外部环境的干扰，摄像头采集到的图像可能会存在噪声，并且采集到的图像一般为彩色，彩色图像包含大量的无用数据，这些无用信息会加大后续图像处理的运算量。通常在对视频图像处理前需要采取一系列的数字图像处理来去除视频图像冗余的信息，增强目标特征信息，简化数据以减少后续处理的运算量，从而加快运算速度[65]。

3.2.1 图像的灰度化

摄像头采集的图像一般都是 RGB 色彩空间的彩色图像，彩色图像既包含了物体的颜色特征又包含了物体的形态特征，由 RGB 三个颜色分量组成。直接对彩色图像进行处理需要对三个分量依次进行处理，处理效率不高。进行图像处理时，一般并不关注图像的色彩信息，而是关注图像的边缘、轮廓、梯度等形态特征信息，这些信息灰度图像就包含了，因此在图像处理中一般需要将彩色图像转化为灰度图，使用转化后的灰度图进行后续的图像处理。将纯黑色与纯白色之间

按明暗程度划分为一定数量的等级，这个等级就是灰度。图像的灰度化就是将彩色图像转化为使用灰度表示的图像的过程。灰度化目的是去除原图像的色彩信息，简化数据，以减少后续图像处理的运算量，从而加快运算速度，以 RGB888 为例，灰度化前需要对 24 位（256^3 种颜色）数据进行运算，而灰度化后只有 8 位（256 种灰度等级的颜色）数据。由此可见，灰度化处理可以简化数据，降低系统资源消耗，提高运算速度。

目前有多种算法可以实现灰度化，可以使用 RGB 最大的分量值、也可以使用三个分量的算术平均值或加权平均值[66]，其中根据人眼对色彩的敏感程度调节参数的加权平均值法使用最为广泛，如式(3-4)所示：

$$X = 0.3R + 0.59G + 0.11B \tag{3-4}$$

式中，X 表示该像素的灰度值，R、G、B 表示该点像素的红、绿、蓝颜色分量。为了验证三种灰度化算法的效果，在 MATLAB 上对算法进行了仿真，仿真结果如图 3-3 所示，可以得出加权平均值法灰度化效果最好的结论。

(a) 原始图像　(b) 最大值灰度化

(c) 平均值法灰度化　(d) 加权平均值法灰度化

图 3-3 灰度化算法仿真

Fig. 3-3 Grayscale algorithm simulation

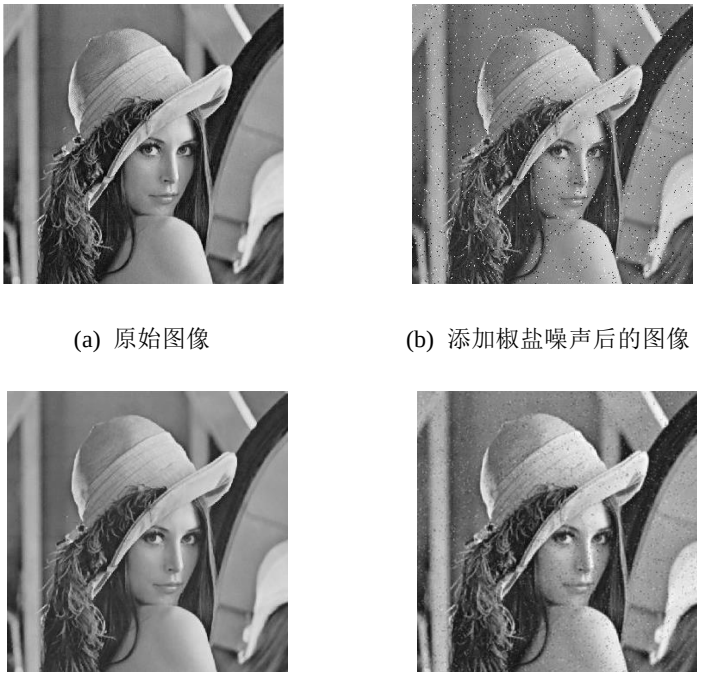
3.2.2 图像的去噪

摄像头在采集过程中难免会产生噪声，噪声会降低图像的质量，甚至会导致图像变得模糊不清，进而影响到后续图像处理的质量，并对最后运动目标检测的结果产生很大的影响。因此需要采取相应的滤波算法过滤掉图像中的噪声，提高图像的质量。图像处理中一般采用空间滤波器来过滤图像中的噪声。空间滤波器由领域和预定义操作组成，领域由中心像素和周围像素组成。对领域内的像素进行预定义的操作会产生的新的像素值，使用这个新像素值来代替原来领域中心位置的像素值，从而达到滤波的目的。常用的滤波算法主要有以下几种：

(1) 均值滤波[67]，将领域内像素灰度值的平均值赋值给领域中心点；

(2) 中值滤波[68]，将领域内像素灰度值的中值赋值给领域中心点。

为了对比两种滤波算法的过滤能力，使用 MATLAB 对两种算法进行了仿真，验证了两种算法对椒盐噪声和高斯噪声的过滤能力。其中对椒盐噪声的过滤效果如图 3-4 所示，对高斯噪声的过滤效果如图 3-5 所示，从图中可以得出以下结论：均值滤波过滤噪声的效果不好，并且会破坏图像的细节，滤波后的图像可能会失真；中值滤波可以有效地过滤噪声，并且过滤后的图像边缘没有模糊。



(a) 原始图像　(b) 添加椒盐噪声后的图像

(c) 中值滤波后的图像　(d) 均值滤波后的图像

图 3-4 两种滤波算法对椒盐噪声的过滤效果对比

Fig. 3-4 The filtering effect of two filtering algorithms on salt and pepper noise is compared

(a) 原始图像

(b) 添加高斯噪声后的图像

(c) 中值滤波后的图像

(d) 均值滤波后的图像

图 3-5 两种滤波算法对高斯噪声的过滤效果对比

Fig. 3-5 The filtering effect of two filtering algorithms on Gaussian noise is compared

3.2.3 图像的二值化

二值化就是选取合适的阈值，然后用该阈值与灰度图上像素点的灰度等级逐一比较，灰度值大于或等于阈值的像素点被判定为感兴趣的区域，使用白色表示，灰度值小于阈值的像素点被判定为背景，使用黑色来表示。灰度图经过二值化处理后像素值只有纯黑、纯白两种，而不再是灰度图的 256 种灰度等级，数据量进一步变小，后续的图像处理变得更加简单，并且能够使目标的轮廓变得更加明显，从而达到突出感兴趣区域的目的[69]。二值化主要有以下两种算法：

(1) 固定阈值：通过指定的阈值来进行二值化。这种算法计算量小、速度快。但是对环境的适应性不好，选取的阈值不当可能会造成图像关键信息丢失的现象。

(2) 自适应阈值：使用最大类间方差法计算的结果作为阈值来进行二值化[70]。最大类间方差法的公式如式(3-5)所示。每个灰度等级都能将图像分为前景和背景两类图像。

$$\sigma^2=\mathrm{w}_0(u_0-u)^2+w_1(u_1-u)^2 \tag{3-5}$$

式中 σ^2 为当前灰度等级下的类间方差， w_0 为前景图像占全部图像的比率， u_0 为

前景图像灰度的平均值，w_1为背景图像占全部图像的比率，u_1为背景图像灰度的平均值，u为整帧图像灰度的平均值。按照公式(3-5)计算每个灰度等级下的类间方差并选取最大值，取最大值对应的灰度等级作为当前帧二值化的阈值。自适应阈值能够动态更新阈值，二值化效果好，但是计算量大，速度慢，并且难以使用硬件描述性语言实现。

使用 MATLAB 对上述两种二值化算法进行了仿真，二值化的效果对比图如图 3-6 所示，从图中可以得出以下结论：自适应二值化的分割效果更好，能够保留更多的图像细节，但是计算复杂。

(a)原始图像　　(b)固定阈值二值化效果图（T=128）　　(c)自适应二值化效果图

图 3-6 两种二值化算法的效果对比

Fig. 3-6 Comparison of two binarization algorithms

3.2.4 图像的形态学滤波

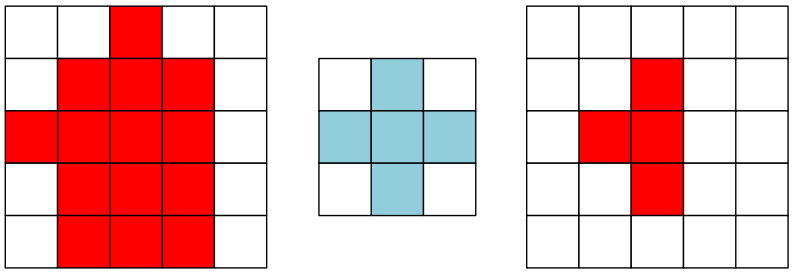
图像的形态学滤波是使用由 0 和 1 组成的结构元素分析图像的形状与结构的方法。二值化图像中存在噪声点会对检测结果产生不良影响，使用形态学滤波可以过滤掉大部分噪声点[71]。形态学滤波基本操作如下。

(1) 腐蚀（erode）

腐蚀运算使用一个核与图像进行卷积，求取局部最小值，然后将这个最小值赋值给领域中心点。假设 A 和 B 是 Z 中的两个集合，B 对 A 腐蚀定义为：

$$A \ominus B = \left\{ z \mid (B)_z \cap A \subseteq A \right\} \tag{3-6}$$

式中，A 是前景图像的一个集合，B 是一个结构元，z 是前景像素值。这个公式指出 B 对 A 腐蚀是所有点 z 的集合，平移后的 B 包含于 A。腐蚀算法例图如图 3-7 所示，使用结构元对图像进行腐蚀操作，只有结构元完全在图像中才保留该点，否则舍弃该点。腐蚀算法的仿真如图 3-8 所示，二值化图像经过腐蚀后，白色区域会变小，相邻暗的区域会连通起来，孤立的小白点会消失。



(a) 图像　　(b) 结构元　　(c) 腐蚀后的图像

图 3-7 腐蚀算法例图

Fig. 3-7 Erode effect diagram

(a)原二值化图　　(b)腐蚀后的效果图

图 3-8 腐蚀算法仿真图

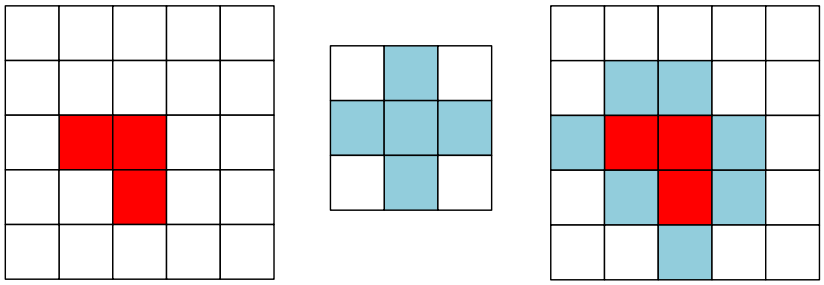
Fig. 3-8 Corrosion algorithm simulation diagram

(2) 膨胀（dilate）

与腐蚀是一个类似的操作，只不过膨胀是将求取覆盖区域的最大值赋值给领域中心点。假设 A 和 B 是 Z 中的两个集合，B 对 A 膨胀定义为：

$$A \oplus B = \left\{ z \mid (B)_z \cap A \neq \varnothing \right\} \tag{3-7}$$

B 对 A 的膨胀就是所有位移 z 的集合，条件是 B 的元素与 A 的元素至少一个元素重叠。膨胀算法例图如图 3-9 所示，使用结构元对图像进行膨胀操作，只要结构元与图像还有交集就保留该点，否则舍弃该点。膨胀算法的仿真如图 3-10 所示，膨胀后图像的白色区域变大，同时相邻白色区域会连通起来，并且可以消除图像中的小黑点。



(a) 图像　　(b) 结构元　　(c) 膨胀后的图像

图 3-9 膨胀算法例图

Fig. 3-9 Dilate effect diagram

(a) 原二值化图　　(b) 膨胀后的效果图

图 3-10 膨胀算法仿真图

Fig. 3-10 Simulation diagram of expansion algorithm

(3) 开运算（opening operation）

开运算是先腐蚀后膨胀的过程。开运算的仿真如图 3-11 所示，从图中可以看出通过开运算可以平滑二值化图像的轮廓，去除孤立的小白点、细小的毛刺，并且能够保留图像的位置与形状。

(a) 原二值化图　　(b) 开运算后的效果图

图 3-11 开运算仿真图

Fig. 3-11 Open operation simulation diagram

(4) 闭运算（closing operation）

闭运算是先膨胀后腐蚀的过程。闭运算的仿真图如图 3-12 所示，从图中可以看出。通过闭运算可以消除二值化图像的中细小的小黑孔，将轮廓中断裂部分连接起来，并且能够保留图像的位置与形状。

(a) 原二值化图　　(b) 闭运算后的效果图

图 3-12 闭运算仿真图

Fig. 3-12 Closed operation simulation diagram

3.3 运动目标检测 IP 核的设计

运动目标检测 IP 核的设计，处理流程如图 3-13 所示。将当前帧图像写入帧缓存中，再从帧缓存中读出上一帧的图像；将上一帧图像写入 FIFO 缓存器，对当前帧做打两拍处理；对两帧图像进行灰度化、中值滤波的预处理；将预处理后的两帧图像进行差分运算，并取差分结果的绝对值；对差分结果进行二值化、形态学滤波的后处理；在形态学滤波后的图像中统计运动目标的位置信息，并与当前帧进行叠加，输出叠加矩形框的图像。

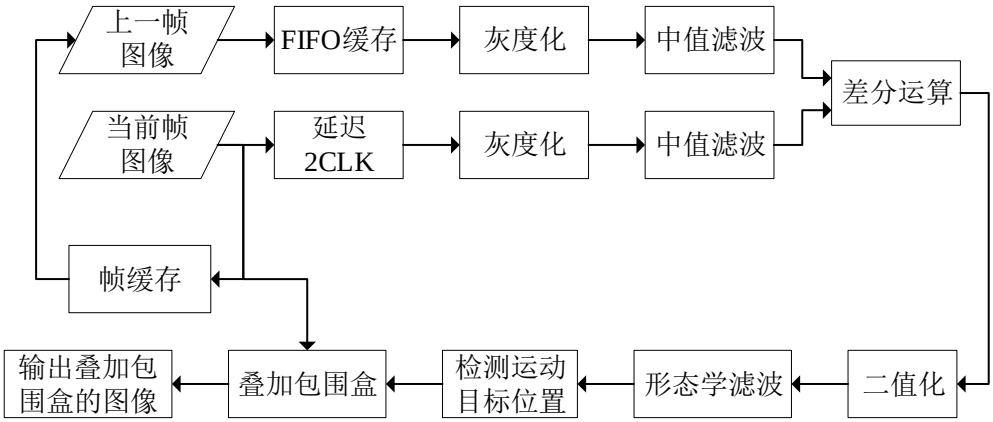


图 3-13 运动目标检测流程图

Fig. 3-13 Flow chart of moving target detection

3.3.1 灰度化算法实现

图像采集模块采集到的视频图像为 RGB888 格式，为了减少后续图像处理的运算量，加快运动目标的检测速度，需要对采集到的视频图像进行灰度化处理。首先完成 RGB888 到 YCbCr 的色彩空间的转换，并取 YCbCr 中的亮度分量 Y 值作为灰度值。RGB 转化 YCbCr 的公式如(3-8)所示：

$$\begin{cases} Y = 0.299R + 0.587G + 0.114B \\ Cb = -0.172R - 0.339G + 0.511B + 128 \\ Cr = 0.511R - 0.428G - 0.083B + 128 \end{cases} \tag{3-8}$$

由于可编程逻辑无法进行浮点运算，但是可以通过右移的方式实现除以 2 的指数幂，因此将公式右侧先扩大 256 倍将浮点数转化为整数，再右移 8 位，得到公式(3-9)：

$$\begin{cases} Y = ((77*R + 150*G + 29*B) >> 8) \\ Cb = ((-43*R - 85*G + 128*B) >> 8) + 128 \\ Cr = ((128*R - 107*G - 21*B) >> 8) + 128 \end{cases} \tag{3-9}$$

式(3-9)运算过程中可能会出现负数，因此对公式进行了变化，得到如下公式(3-10)。

$$\begin{cases} Y = (77*R + 150*G + 29*B) >> 8 \\ Cb = (-43*R - 85*G + 128*B + 32768) >> 8 \\ Cr = (128*R - 107*G - 21*B + 32768) >> 8 \end{cases} \tag{3-10}$$

公式(3-10)在可编程逻辑中采用三步完成，第一步计算括号内的各乘法项，第二步括号内各项相加，第三步括号内计算结果右移 8 位，整个过程消耗三个时钟周期，因此需要延迟 3 拍以同步数据信号。灰度化的效果图如图 3-14 所示。

图 3-14 灰度化效果图

Fig. 3-14 display of graying

3.3.2 中值滤波算法实现

有多种算法可以求取领域中九个像素灰度值的中值，比如冒泡排序法、选择排序法等。但是上述的算法只适合于软件实现，对于硬件描述性语言来说实现比较复杂，并且不能合理利用硬件并行运算速度快的优势，因此采用流水线操作的方式来求取中值实现中值滤波，从而提高运算效率。中值滤波算法框图如图 3-15 所示。

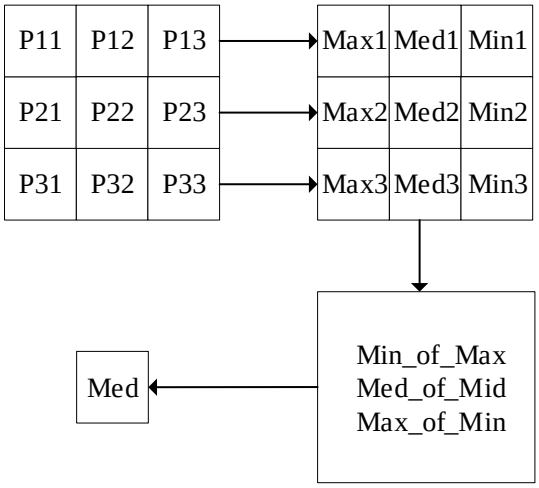


图 3-15 中值滤波算法框图

Fig. 3-15 Block diagram of median filtering algorithm

算法需要生成 3×3 的像素阵列，采用 Shift_RAM IP 核来生成 3×3 的像素阵列。Shift_RAM IP 核配置如图 3-16 所示，勾选上时钟使能信号，将数据宽度设置为 8，深度设置为一行数据的个数。设置完成后，就是相当于向该 IP 核内写入数据位宽为 8 位的数据，然后写入的数据会在 640 个周期后输出。Shift_RAM IP 核可以看作是一个移位寄存器，使用 Shift_RAM IP 核可以轻松实现图像的卷积运算。使用 RAM（Random Access Memory，随机存取存储器）、FIFO 存储两行数据也可以生成 3×3 的像素阵列，但是操作较为繁琐和复杂，使用 Shift_RAM IP 核实现比较简单。Shift_ RAM 一次只能缓存一行数据，所以采用例化两次 Shift_RAM IP 核以级联的方式来缓存前两行的数据，加上正在输入的第三行来生成 3×3 的矩阵，级联的方式如图 3-17 所示。

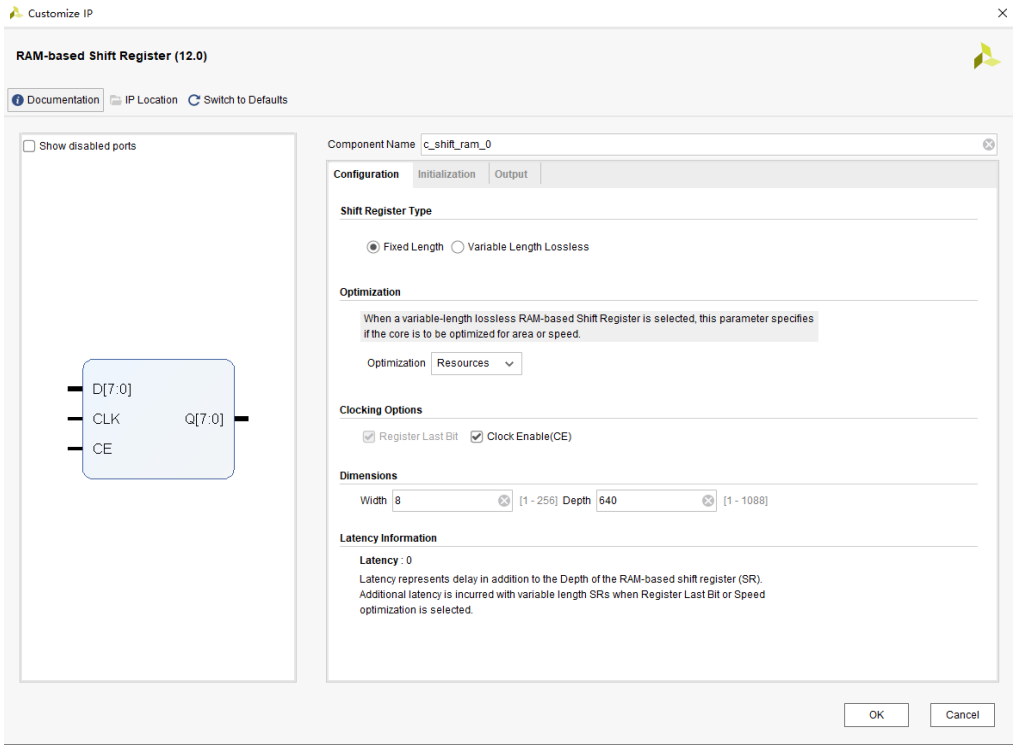


图 3-16 Shift_RAM IP 核配置图

Fig. 3-16 Shift RAM IP core configuration diagram

```
c_shift_ram_0    yourname(
.D(row3_data),
.CLK(clk),
.CE(Shift_clk_en),
.Q(row2_data)
);
c_shift_ram_0    yourname(
.D(row2_data),
.CLK(clk),
.CE(Shift_clk_en),
.Q(row1_data)
);
```

图 3-17 Shift_RAM IP 核级联示意图

Fig. 3-17 Shift RAM IP core cascade diagram

生成 3×3 矩阵的示意图如图 3-18 所示。首先使用硬件描述性语言编写一个三个数排序的模块，对该模块进行多次例化。采用流水线的方式快速求取中值，第一级流水线同时对每一行三个灰度值进行排序；第二级流水线对第一级流水线的排序后的三个最大值、三个中间值、三个最小值进行同时排序分别取中间值；第三级流水线对第二级流水生成的三个中间值进行排序取中间值，并将第三级流水线的结果赋值给邻域中心点的像素值。中值滤波的结果图如图 3-19 所示。

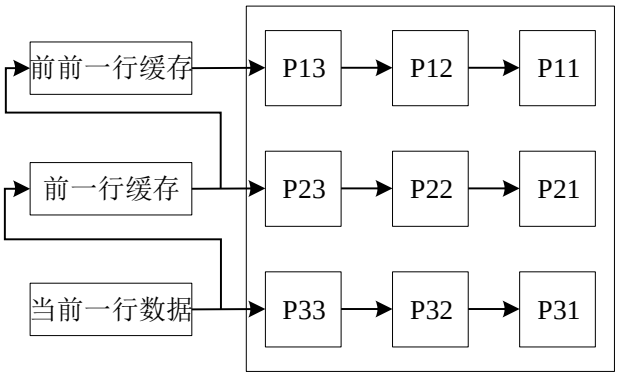


图 3-18 3×3 像素阵列的生成示意图

Fig. 3-18 A schematic of the generation of pixel arrays

图 3-19 中值滤波的结果图

Fig. 3-19 Median filtering result graph

3.3.3 差分处理的实现

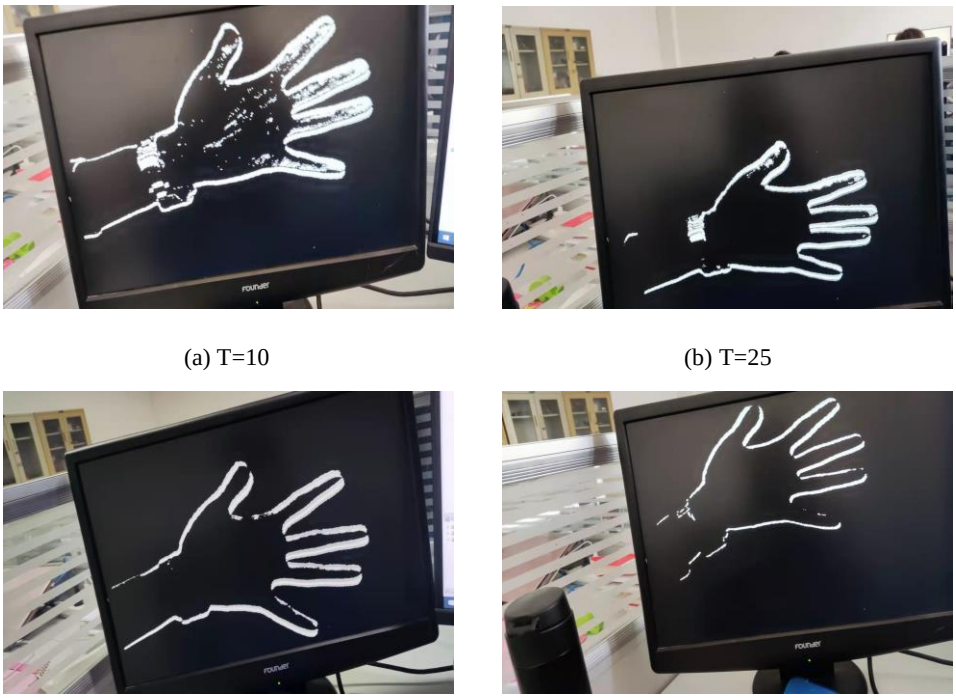
该自定义的 IP 有两组 AXI Sream 从接口，两组 AXI Stream 主接口。其中 AXI Stream Slave0 接口的数据是 OV5640 摄像头采集的视频流数据；AXI Stream Slave1 接口的数据是 VDMA1 读通道输出的视频流数据（缓存的上一帧的数据）；AXI Stream Master0 是输出到 VMDA0 的视频流数据，用于输出到 HDMI 显示器；AXI Stream Master1 是输出到 VDMA1 写通道的视频流，用于缓存一帧数据来做差分运算。

帧差运算需要相同像素坐标上的像素灰度值做差。该模块例化了灰度转化、中值滤波、二值化、形态学滤波等模块。为了使两帧数据同步，即像素级的对齐（同一时刻，对应的两个像素数据永远是同一个像素坐标），该模块例化了一个 FIFO。FIFO 是一种没有读写地址线的缓存器，先写入 FIFO 中的数据会先读出，将 VDMA1 中的数据写入其中，并在下一帧起始信号到来时从 FIFO 中读取

出来做帧差运算。在添加 FIFO 时将 FIFO 设置为同步 FIFO，并添加即将读空（almost_empty）和即将写满（almost_full）两个信号，分别用于指示 FIFO 即将读空和即将写满。编写 Verilog 代码操作 FIFO 的读端口和写端口来实现数据的缓存。为了保证写入 FIFO 的第一个数据为一帧数据的第一个有效像素数据，FIFO 的写使能信号（fifo_wr_en）只有当 s1_axis_tvalid、s1_axis_tready、s1_axis_tuser 三个信号同时为高电平时才为高电平，其中 s1_axis_tvalid 表示来自 VDMA1 的数据该时刻有效，s1_axis_tready 表示从设备已经准备完成，s1_axis_tuser 表示来自 VDMA1 的一帧的第一个数据到来。等待 FIFO 中缓存一定量的数据后，当检测到摄像头输入视频流的帧起始信号（start of frame），才打开 FIFO 读使能，即只有当 s0_axis_tvalid、s0_axis_tready、s0_axis_tuser、fifo_full 同时为高电平时才打开读使能信号，其中 s0_axis_tvalid 表示摄像头输入视频流数据有效，s0_axis_tready 表示从设备已经准备完成，s0_axis_tuser 表示摄像头输入的视频流的帧起始信号，fifo_full 表示 FIFO 写满。由于 FIFO 的读写各自需要消耗一个时钟周期，所以将来自摄像头的视频流数据也延迟两个时钟周期以同步数据。这样，如果检测到摄像头输入的视频流数据的帧起始信号，就将 FIFO 中缓存的上一帧的第一个数据读出，之后来自摄像头的视频流每进来一个数据，就从 FIFO 中读取一个数据做差分运算。对帧差运算的结果取绝对值时，在 Verilog 中不能直接使用绝对值的符号来求取绝对值，采用的是条件判断的方式，即先判断两个数的大小，再决定两个数相减的顺序，从而避免相减产生负数再取绝对值的问题。

3.3.4 二值化算法的实现

手动设置二值化的阈值，并设置一个帧差标志位。将帧差结果的绝对值与阈值作比较，当大于阈值则初步认定该点像素为运动目标输出 1，否则判断为该点像素为背景输出为 0。为了将二值化后的图像显示出来，这里将帧差标志位拼接成一个 24 位的数据作为颜色数据。为了选择合适的阈值，这里做了一个实验，分别将阈值设置为 10、25、50、75，二值化的实验结果如图 3-20 所示。实验结果显示当阈值设置得较小时，检测出来的运动目标轮廓很完整，但包含较多的噪声。增大阈值可以有效地过滤噪声，但是会使运动目标轮廓变得不完整。综合考虑，在当前环境下选择 25 作为阈值比较适合。



(a) T=10　　(b) T=25

(c) T=50　　(d) T=75

图 3-20 不同阈值下二值化的效果图

Fig. 3-20 Renderings of binarization at different thresholds

3.3.5 形态学滤波的算法实现

（1）腐蚀算法的实现

腐蚀算法实现需要调用之前完成的 Shift_RAM IP 来生成3×3的矩阵。对领域中的九个值进行与运算如式(3-11)所示，只有九个值都为 1 时，输出图像的该点才输出为 1，否则为 0。使用 Verilog 实现该算法，在第一个时钟周期将3×3矩阵的每一行的三个值进行与运算得到三个中间值如式(3-12)所示；在第二个时钟周期将这三个中间值进行与运算如式(3-13)所示，并将最终值赋值给领域中心点。整个过程消耗两个时钟周期，对同步信号做打两拍处理。

$$P_{\mathrm{a}} = P_{11} \& P_{12} \& P_{13} \& P_{21} \& P_{22} \& P_{23} \& P_{31} \& P_{32} \& P_{33} \tag{3-11}$$

$$\begin{cases} P_1 = P_{11} \& P_{12} \& P_{13} \\ P_2 = P_{21} \& P_{22} \& P_{23} \\ P_3 = P_{31} \& P_{32} \& P_{33} \end{cases} \tag{3-12}$$

$$P_{\mathrm{a}} = P_1 \& P_2 \& P_3 \tag{3-13}$$

式(3-11)、(3-12)、(3-13)中 P_a 表示腐蚀结果，P_{11}-P_{33} 表示3×3矩阵中的九个像素点的值，P_1、P_2、P_3 是中间运算结果。腐蚀效果如图 3-21 所示。

图 3-21 腐蚀效果图

Fig. 3-21 Corrosion effect drawing

（2）膨胀算法的实现

膨胀算法的实现同样调用之前完成的 Shift_RAM IP 来生成 3×3 的矩阵。对领域中的九个值进行或运算如式(3-14)所示，九个值至少有一个 1，该点输出就为 1，只有 9 个值全为 0 时该点才输出 0。在第一个时钟周期将3×3矩阵的每一行的三个值进行或运算得到三个中间值如式(3-15)所示；在第二个时钟周期将这三个中间值进行或运算如式(3-16)所示，并将最终值赋值给领域中心点。整个过程消耗两个时钟周期，对同步信号做打两拍处理。

$$P_b = P_{11} \mid P_{12} \mid P_{13} \mid P_{21} \mid P_{22} \mid P_{23} \mid P_{31} \mid P_{32} \mid P_{33} \tag{3-14}$$

$$\begin{cases} P_4 = P_{11} \mid P_{12} \mid P_{13} \\ P_5 = P_{21} \mid P_{22} \mid P_{23} \\ P_6 = P_{31} \mid P_{32} \mid P_{33} \end{cases} \tag{3-15}$$

$$P_b = P_4 \mid P_5 \mid P_6 \tag{3-16}$$

式(3-14)、(3-15)、(3-16)中 P_b 表示结果，P_{11}-P_{33} 分别表示矩阵中的九个像素点的值，P_4、P_5、P_6 表示中间运算结果。膨胀效果如图 3-22 所示。

图 3-22 膨胀效果图

Fig. 3-22 Expansion rendering

3.3.6 获得运动目标的位置信息

通过对帧起始信号和行结束信号进行计数来获得运动目标的像素坐标。具体做法为：定义了 x_cnt、y_cnt 两个 reg 型的变量两个对 x，y 方向进行计数。当系统复位时，将两个变量初始化为 0。当数据有效信号到来时，判断行结束信号是否到来，如果到来就说明一行数据结束，将 x_cnt 进行清零操作，并将 y_cnt 加一。同时，如果帧起始信号到来时，说明新的一帧数据到来，将 x_cnt 和 y_cnt 进行清零操作。通过这样的操作就获取了帧差后的像素坐标。

传统获取运动目标最大矩形框的方法是在所有记为运动目标的像素点（形态学滤波结果为 1 的像素点）的像素点中寻找最左、最右、最上、最下的像素点，并记录这些点的像素坐标。具体做法为定义五个 reg 型变量（up_reg、down_reg、left_reg、right_reg、flag_reg），其中前四个分别用于寄存帧差后运动目标的上下左右的坐标，flag_reg 用于标记矩形框的边界。遍历形态学滤波后的整帧图像，寻找运动目标的上下左右边界。当帧起始信号到来时，初始化上述的五个寄存器。当数据有效信号和形态学滤波结果同为高电平时，说明该点像素认定为运动目标，将该点的横纵坐标分别与上下左右寄存器作比较并更新相关位置寄存器中的数值，来获取运动目标的位置信息。当一帧数据结束时，上下左右寄存器中的数值就是运动目标的边界信息。

上述方法存在一定弊端。当场景中只有一个运动物体时，上述方法可以有效的获取这个运动有效地获取信息，但是当场景中出现多个运动目标时，这种方法会将多个运动目标认定为一个运动目标，即上述方法只适用于单运动目标检测，不适用于多运动目标。为解决上述方法不能有效获取多运动目标位置的问题，本文使用一种基于邻域的方法来获取运动目标位置信息，即通过判断运动目标的间距，当间距大于一定范围就认为是另一运动目标。获取运动目标位置信息流程图如图 3-23 所示。

手动设置一个最小间距的参数，通过该参数来判定是否为一个运动物体。同样需要通过帧起始信号和行结束信号来计算帧差后的像素坐标，这里不再赘述。首先，如果各运动目标的上下左右边界加上最小间距不大于图像的边界，则将上下左右边界向四周各扩散最小间距，否则将对应边界设置为图像的边界，称之为运动目标的邻域。检测并标记需要分两步进行。第一步，找到标记为运动目标的像素点，判断该像素点是否位于其他运动目标的领域范围内，若均不在则判定该像素点是新的运动目标，否则判定该像素点不是新的运动目标。第二步，将数据更新到运动目标列表中。如果该像素点均不在其他运动目标的领域范围内，将该点像素坐标赋值给新的运动目标位置寄存器；若在某一运动目标的领域范围之

内，则扩展该元素的边界信息，即将该像素点的像素坐标与该运动目标边界信息作比较，若 x 坐标小于左边界，则将其 x 坐标扩展为左边界，其他边界以此类推，不再赘述。在一帧数据统计结束后，寄存输出各个运动目标的位置信息。

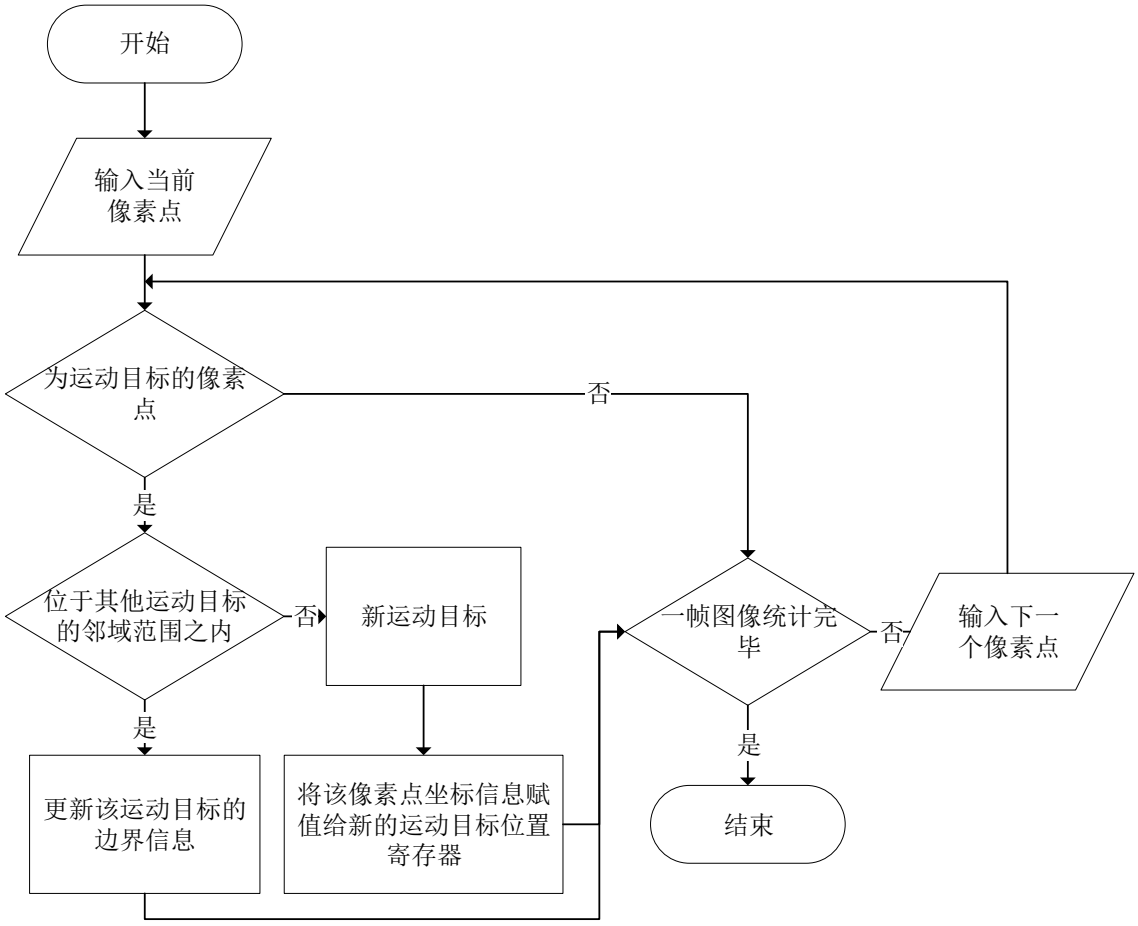


图 3-23 获取运动目标位置信息流程图

Fig. 3-23 Obtain the flow chart of moving target position information

3.3.7 添加矩形框

传统添加矩形框的做法为：首先，计算摄像头输入视频图像的像素坐标，即根据帧起始信号和行结束信号对 x、y 方向进行计数。设置了一个 reg 型变量 boarder_flag，用于标志像素点位于运动目标的矩形框上。当矩形框标志为高电平时，判断来自摄像头视频数据的坐标是否位于矩形框的上下左右边界上，是则将 boarder_flag 拉高，否则将 boarder_flag 置零。最终，根据 boarder_flag 的值来选择输出数据，当 boarder_flag 为 1 时，输出的数据为 24'hff_00_00（红色），

否则输出该点原来的数据。从而实现给运动图像添加矩形框，并将矩形框与原图像叠加的效果。

上述添加矩形框的方法适用于单运动目标，当出现对个运动目标时需要去除重叠边框。改进后添加矩形框的流程图如图 3-24 所示。改进的添加矩形框的算法为：首先对输入信号的帧起始信号的行结束信号进行计数，得到其横纵坐标。检测并标记需要分两步进行，第一步判断当前像素点是否位于边框上；第二步判断边框是否位于其他运动目标的范围内，若是则说明边框有重叠部分，此时内部边框将不再显示。只有该点像素坐标位于边框上且不位于其他运动目标的范围内才输出为 24'hff_00_00（红色），否则输出为原图像数据。该方法可以有效去除重叠边框，适用于给多个运动目标添加矩形框。添加矩形框的流程图如图 3-24 所示。

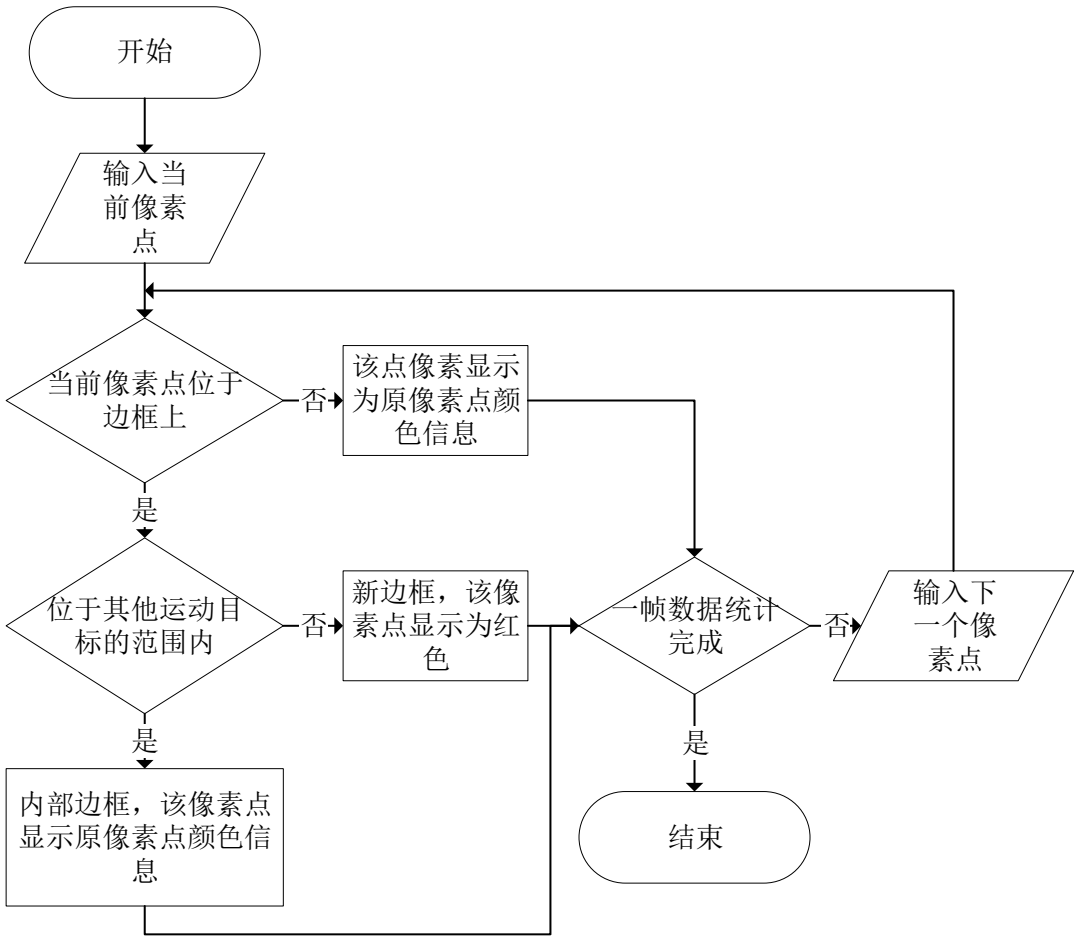


图 3-24 添加矩形框的流程图

Fig. 3-24 Flowchart for adding a rectangular box

3.4 本章小结

本章分析了三种经典的运动目标检测算法，以所得信息、鲁棒性、复杂度、实时性为指标，对三种经典的运动目标检测算法进行了比较，最终选择了帧间差分法作为本系统的算法。另外，对一些数字图像处理技术进行了介绍、分析，选择了合适的数字图像技术，为系统的建立奠定了理论基础。完成了运动目标检测算法 IP 核的设计，并将各阶段图像处理算法的处理效果进行了展示。设计了灰度转化模块，将彩色图像转化为灰度图像，减少了后续图像处理的运算量；设计了快速中值滤波模块，对灰度化的图像去噪；然后设计了差分处理模块和二值化模块，实现了帧间差分法，通过实验的方式选取了合适的二值化阈值；设计了形态学滤波模块，对二值化的结果进行开运算，提升检测的精度；优化了获取运动目标位置和添加矩形框的算法，实现了对多个运动目标的标记。

第4章 系统的总体设计

4.1 系统总体结构

系统围绕 Zynq-7020 芯片，采用软硬件协同设计和模块化设计的方法，将系统分割成一个个的模块。有效利用 Zynq 平台上擅长简单逻辑并行计算的可编程逻辑（PL）、擅长复杂程序串行执行的处理系统（PS），将这些任务合理地分配到可编程逻辑和处理系统分别完成。系统总体结构如图 4-1 所示。系统由图像采集模块、算法处理模块、数据存储模块、图像显示模块四个模块组成，其中算法处理模块与数据存储模块是双向数据交互。

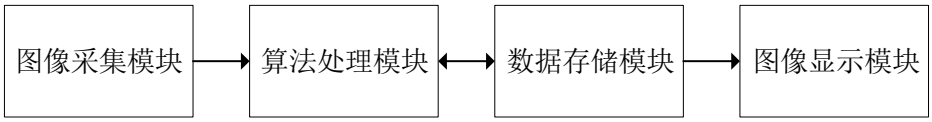


图 4-1 系统总体结构

Fig. 4-1 Overall system structure

系统上电后，处理系统完成 OV5640 的配置后，OV5640 摄像头将采集到的视频数据传输到 OV5640 图像采集模块，然后传输到算法处理模块进行处理；数据存储模块将输入的图像和处理后的图像缓存到系统存储器中；算法处理模块完成运动目标的检测；显示模块将检测结果实时显示在 HDMI 显示器上。系统分为可编程逻辑和处理系统两部分，可编程逻辑主要完成图像采集模块、算法处理模块、图像显示模块；处理系统则完成摄像头的配置、系统存储器的读写控制以及各个 IP 的驱动。每个模块的具体功能如下：

(1) 图像采集模块主要完成摄像头的配置、舍弃不稳定的前几帧、RGB565 到 RGB888 的转换的视频图像采集任务；

(2) 算法处理模块主要是运动目标检测 IP 核，实现了帧间差分法和相关数字图像处理技术；

(3) 数据存储模块通过 VDMA IP 来完成 DDR3 存储器的数据的读写。通过 VDMA1 写通道将输入的图像缓存到 DDR3 存储器“地址 1”，通过 VDAM1 读通道从 DDR3“地址 1”中读取出来传输到算法处理模块，通过 VDMA0 写通道将处理结果写入 DDR3“地址 0”，通过 VDMA0 读通道从 DDR3 存储器“地址 0”读取出来传输到显示模块；

(4) 图像显示模块主要是将运动目标的检测结果将 DDR3 存储器中读取出来并显示在 HDMI 显示器上。

系统设计框架图如 4-2 所示。

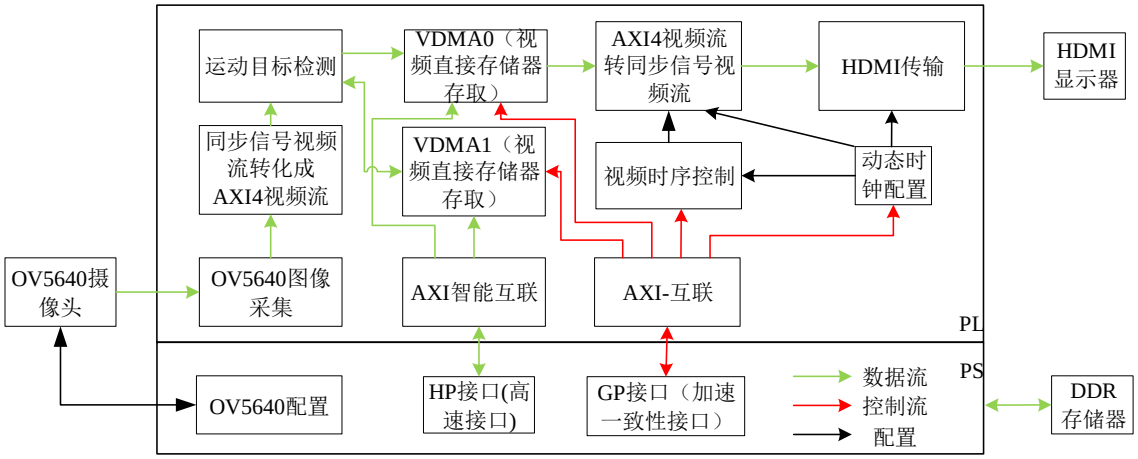


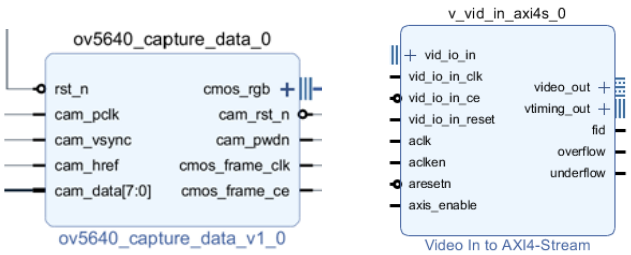
图 4-2 系统设计框图

Fig. 4-2 System design block diagram

4.2 各模块的设计

4.2.1 图像采集模块

图像采集模块主要包括图像采集 IP 核和 Video In to AXI4-Stream IP 核，如图 4-3 所示。



(a) 图像采集 IP 核　　(b) Video in to AXI4-Stream IP 核

图 4-3 图像采集模块的 IP 核

Fig. 4-3 IP core of image acquisition module

图 4-4 为 OV5640 摄像头 RGB565 模式时序图摄像头的输出时序图，HREF

信号的高电平期间，在 PLCK 的每个上升沿，OV5640 会输出一个 8bit 的图像数据。OV5640 传感器输出 RGB565 的格式每一个像素为 16bit，因此摄像头输入接口的两个时钟周期输出的数据组合成一个数据。而后续图像处理需要的是 RGB888 格式，通过在 RGB565 数据后分别补 3、2、3 个 0 来完成 RGB565 到 RGB88 的格式转换。处理系统配置完摄像头后，存在一定的时间延迟，即前几帧数据不稳定，因此图像采集 IP 核需要对帧进行计数，先等待 10 帧数据，等数据稳定后才开始数据采集工作。

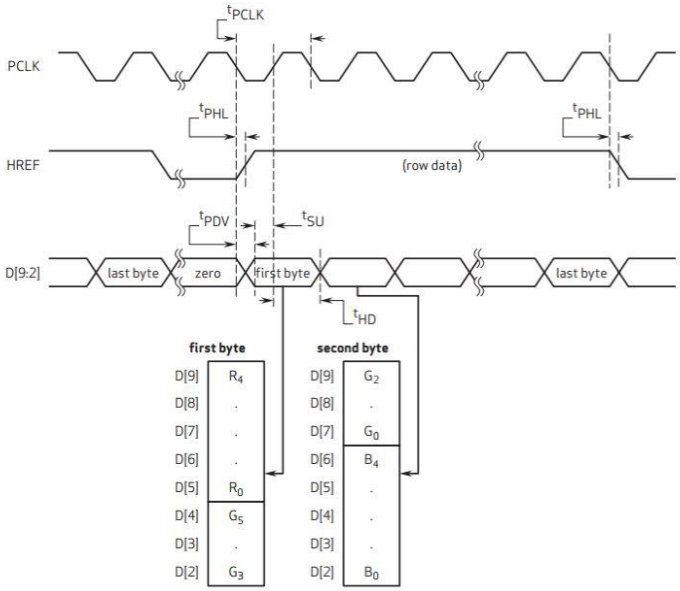


图 4-4 OV5640 RGB565 模式时序图

Fig. 4-4 OV5640 RGB565 mode sequence diagram

Video in to AXI4-Stream IP 核将摄像头输出由行场同步信号控制的视频数据流转化为 AXI4-Stream 格式，方便后续的图像处理操作。

4.2.2 算法处理模块

算法处理模块主要完成帧间差分法和相关数字图像处理技术。该模块如图 4-5 所示，有两路 AXI-Stream Slave 视频输入接口，其中 s0_axis 表示来自 OV5640 摄像头采集模块的当前帧视频流数据，s1_axis 表示来自 VDMA1 读通道的上一帧视频流数据；两路 AXI-Stream Master 接口，其中 m0_axis 表示处理后的也就是包含运动目标的视频流数据，通过 VDAM0 写通道写入 DDR3 中，最终通过 VDMA0 读通道传输到显示模块，m1_axis 表示当前帧视频流数据，通过 VDMA1 写通道写入 DDR3，用于下一帧检测使用。双击该自定义的 IP 核即可

对视频分辨率、二值化阈值、最小间距进行参数配置。

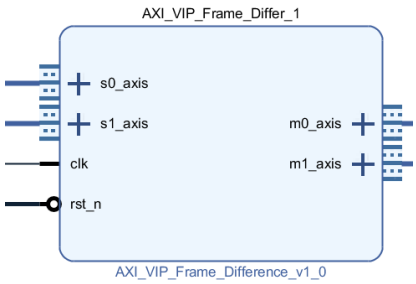


图 4-5 算法处理模块

Fig. 4-5 Algorithm processing module

4.2.3 数据存储模块

数据存储主要使用的 AXI_VDMA IP 核，如图 4-6 所示。图像数据进出系统存储器(DDR3)都经过 VDMA IP 核。通过调用 VDMA IP 核完成帧缓存，方便后续的帧间差分运算以及解决图像输入与图像显示速率不匹配的问题。本系统使用了 VDMA0、VDMA1 两个 VDMA IP 核。VDMA0 的作用是解决输入和显示设备传输速率不匹配的问题，VDMA1 的作用则是缓存上一帧数据用于实现帧差运算。通过 VDMA1 的写通道将当前帧 AXI4 Stream 格式的视频流转成 AXI4 Memory Map 格式，并最终写入 DDR3 的“地址 1”中，通过 VDMA1 的读通道从 DDR3“地址 1”中读取上一帧的图像并以 AXI4 Stream 格式输出到算法处理模块。通过 VDMA0 的写通道将处理后的图像写入 DDR3 的“地址 0”中，并通过 VDMA0 的读通道从 DDR3 的“地址 0”中读取处理后的图像以 AXI4 Stream 格式传输到显示模块。

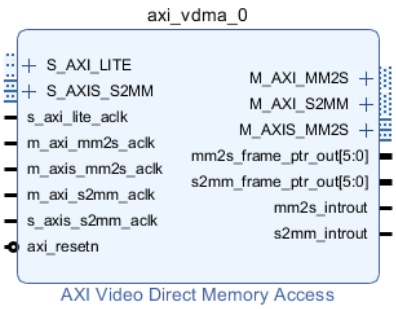


图 4-6 VDMA IP 核

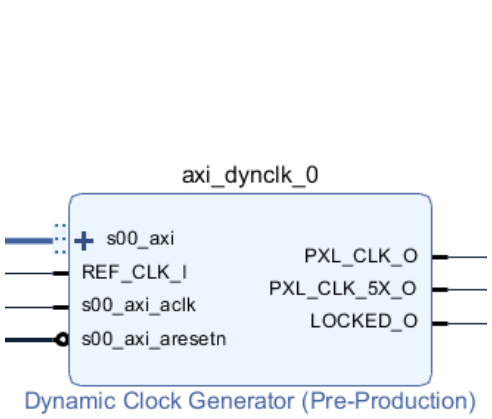
Fig. 4-6 VDMA IP core

一般情况下图像输出端的速率会大于图像输出端的速率，可能会造成输出到显示器上的图像是多帧图像叠加的现象。在许多实时处理的系统中，通常使用帧缓冲存储器来解决这一问题，同时也方便了帧差运算，因为存储一帧图像需要的存储容量大于片内的随机块存储器（Block Random Access Memory，BRAM），所以选择处理系统的 DDR3 存储器作为帧缓冲存储器。由于 DDR3 是 PS 部分的存储接口，PL 逻辑需要通过 AXI 接口访问 DDR3，本系统调用 VDMA IP 核来完成帧缓存。当 VDMA 选择动态同步锁相主模式时，主通道会跳过从通道当前操作的帧缓存，从通道工作在主通道上一次操作过的帧缓存。

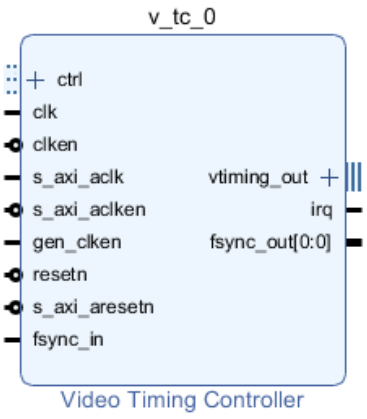
4.2.4 显示模块

(1) 显示模块的组成

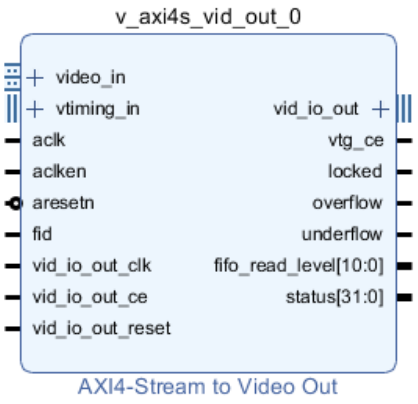
显示模块主要由 AXI-Dynclk IP 核、Video Timing Controller IP 核、AXI-Stream to Video Out IP 核、DVI Transmitter IP 构成，如图 4-7 所示。



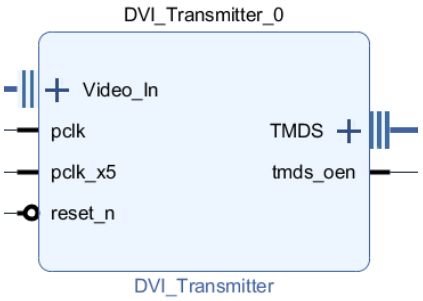
(a) AXI-Dynclk IP 核



(b) Video Timing Controller IP 核



(c) AXI-Stream to Video Out IP 核



(d) DVI Transmitter IP

图 4-7 显示模块所用到 IP 核

Fig. 4-7 Display module used IP core

AXI-Dynclk IP（动态时钟控制器）用于根据设置的视频分辨率动态地为RGB2DVI模块提供时钟。Video Timing Controller（视频时序控制器）IP核用于控制视频输出的时序参数。AXI4-Stream to Video Out IP用于将AXI4-Stream格式的数据流转换成RGB888格式。DVI Transmitter IP是自定义IP核，作用是将RGB888格式的数据转化为最小化传输差分信号（Transition Minimized Differential Signal，TMDS）标准传输视频数据，驱动HDMI接口，从而使用开发板HDMI接口输出包含运动目标的图像。

(2) DVI Transmitter IP自定义IP核

DVI接口只能传输视频不能传输音频，本系统目标为运动目标检测，不需要传输音频，因此只需实现DVI接口的驱动逻辑即可。DVI和HDMI使用的TMDS技术高速传输串行数据。TMDS传输使用两个引脚电压之差来决定传输数据的数值，电磁兼容性能较好。TMDS差分传输技术由四个串行通道组成，其中包括三个数据通道和一个时钟通道。TMDS差分传输主要分为编码和并串转换两个步骤。第一步为根据TMDS编码算法进行编码，在数据有效阶段将数据通道上8位的像素数据编码为10位的字符流，在视频消隐期将2位的控制字符编码为10位的字符流，这个阶段一定程度上实现了直流平衡。第二步为并串转换，将编码后生成的并行的数据字符流和控制字符流转换成串行数据流，最终将串行数据流在三个差分输出通道发送出去。DVI Transmitter模块框图如图4-8所示，由编码模块、并串转换模块、差分输出模块组成。

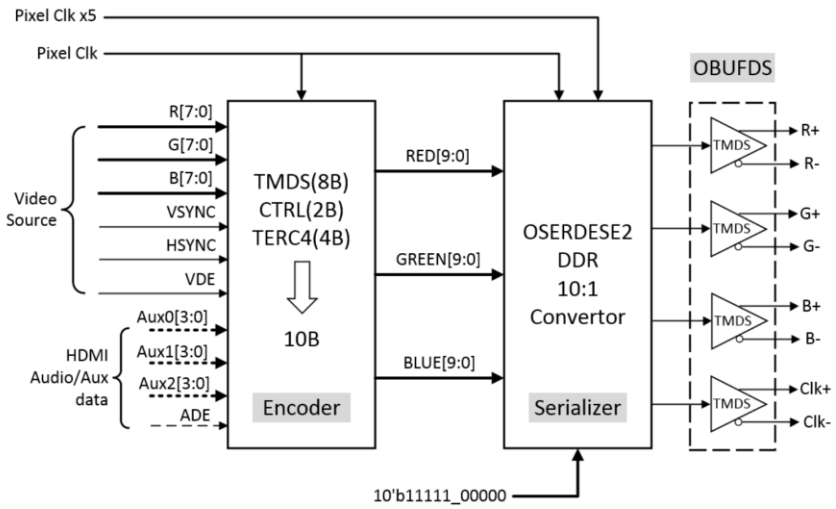


图4-8 DVI Transmitter 模块框图

Fig. 4-8 Block diagram of DVI Transmitter module

编码模块对每个通道输入的像素数据和控制字符使用 TMDS 编码算法进行编码。并串转换模块通过调用专门实现并串转换的硬件资源——OSERDESE2 原语来实现并行数据到串行数据的转换。将两个 OSERDESE2 模块级联起来以达到 10:1 的转换率。OSERDESE2 模块可以实现双倍数据速率，在 5 倍像素时钟频率的时钟下即可满足 10:1 的转换率。最后使用差分输出缓冲器（OBUFDS）将串行数据和时钟信号转换成差分信号输出。

4.3 硬件平台的搭建

通过 Xilinx 公司提供的 Vivado 2018.3 来创建工程，配置 Zynq、VDMA 等 IP 核来完成硬件平台的搭建。一般情况下，一个 IP 核含有多种功能，有些功能需要选择才能正常使用，有些功能是当前系统所不需要的，因此在 IP 核的调用过程中需要对 IP 核的功能进行裁剪，即通过配置 IP 的参数使其工作在预期的工作模式下。

4.3.1 关键 IP 核的参数配置

(1)ZYNQ7 Processing System IP 的配置

ZYNQ7 Processing System IP 核是 PS 端在 PL 端的逻辑接口 IP 核。使用该 IP 核将硬件 IP 核与 PS 程序集成在一起。双击添加的 ZYNQ7 处理系统 IP，进入处理系统的配置界面，如图 4-9 所示。

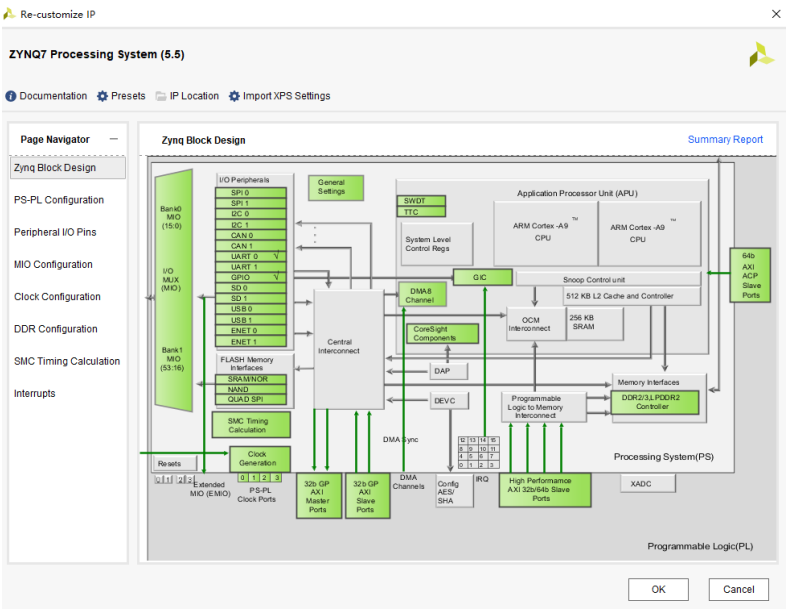


图 4-9 ZYNQ7 Processing System IP 核的配置界面

Fig. 4-9 Configuration page of the ZYNQ7 Processing System IP core

为方便调试，配置了 PS 的 UART0，根据原理图，将 MIO14 和 MIO15 配置成 UART0，保持默认的 115200 波特率。配置 PS 的 DDR3 控制器，选择 MT41J256M16RE-125。在 PS-PL Configuration 栏开启 S_AXI_HP0_Interface。将 FCLK_CLK0 配置为 100Mhz。使能扩展多路复用输入/输出（Extended Multiplexed Input/Output，EMIO），将位宽设置为 2，用来连接摄像头的串行摄像机控制总线（Serial Camera Control Bus，SCCB）接口。使能排队串行外围设备接口（Queued Serial Perripheral Interface，QSPI）Flash 以及安全数字卡（Secure Digital Memory Card，SD 卡）控制器外设，并将 Bank1 的电平标准设置为 1.8V。配置完成的 ZYNQ7 Processing System IP 如图 4-10 所示。

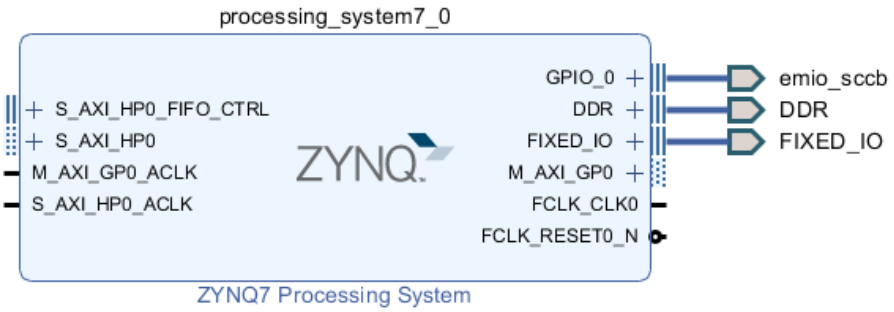


图 4-10 配置完成的 ZYNQ7 Processing System IP 核

Fig. 4-10 ZYNQ7 Processing System IP core is configured

（2）VDMA IP 核的配置

将帧缓冲存储位置的数量设置为 3，突发读的大小设置为 64，数据宽度设置为 24，行缓冲深度设置为 2048，并使能读通道和写通道。VDMA 的高级配置界面如图 4-11 所示。

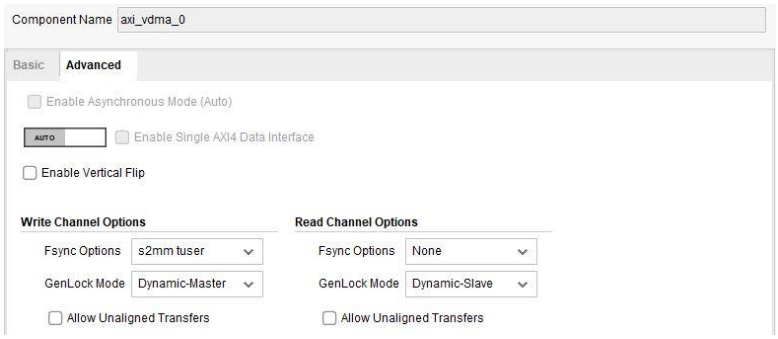


图 4-11 VDMA IP 核高级配置界面

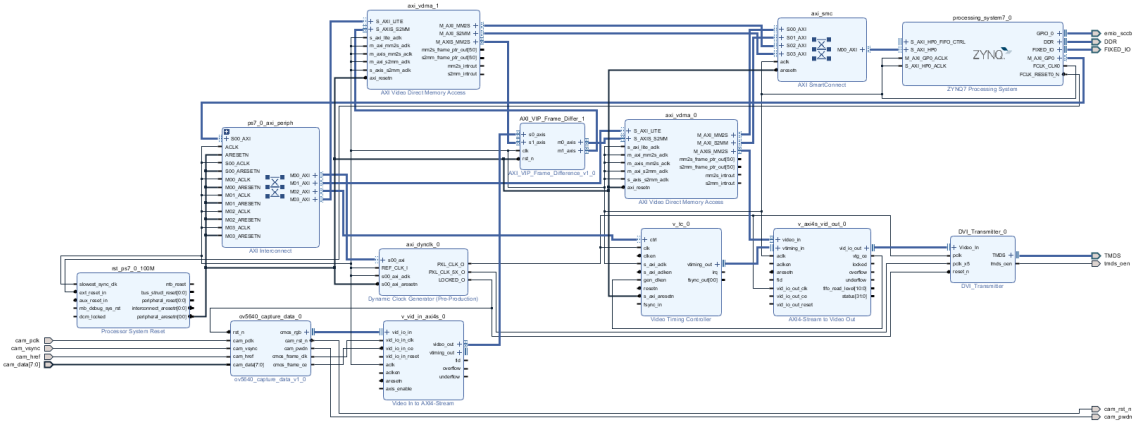
Fig. 4-11 VDMA IP core advanced configuration interface

在 VDMA 的高级配置界面将 VDMA 设置为动态同步锁相模式，其中写通

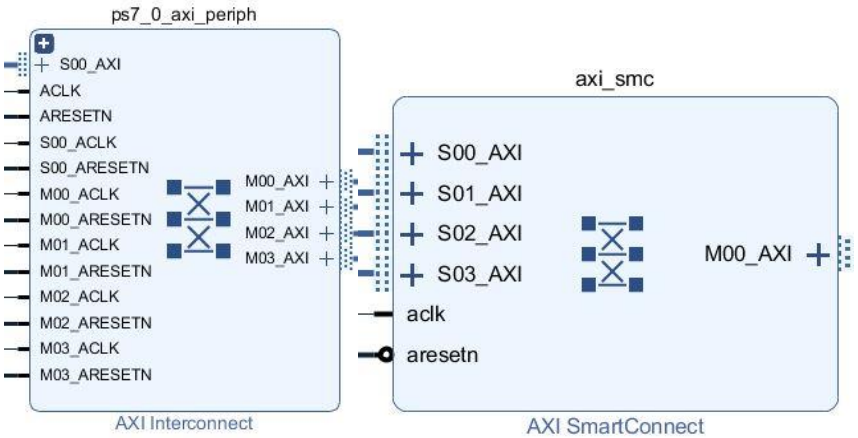
道设置为动态同步锁相主模式，读通道设置为动态同步锁相从模式。本系统添加了两个 VDMA IP 核，即 VDMA0、VDMA1。

4.3.2 系统集成

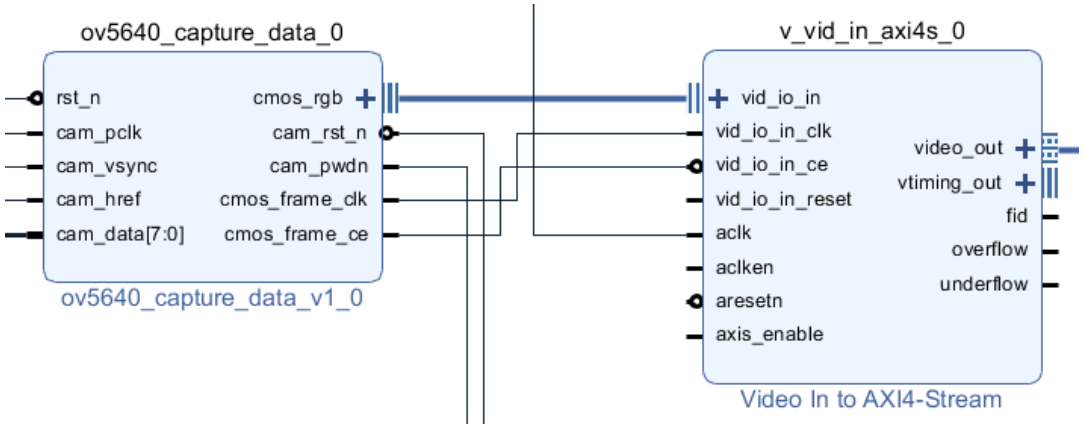
在完成各个 IP 核的配置后便可以进行连线，先进行手动布线，将一些关键信号进行连接，可以避免自动布线带来的问题，提高效率，然后使用软件自动连线功能，最后再手动连接自动连线后遗留的线。构建完成的 PL 端系统整体连线及各个分模块局部图如图 4-12 所示。自动连线后系统自动生成了 AXI 互联 IP 核、智能互联模块、复位模块。其中 ZYNQ IP 核的 M_AXI_GPO 接口通过 AXI 互联 IP 与外部低速外设的控制总线与 AXI_LITE 总线从而达到处理系统动态配置低速外设的目的；ZYNQ IP 核的 S_AXI_HP0 接口通过智能互联模块与 VDMA 的读、写通道存储器端映射的 AXI4 接口，从而实现对存储器（DDR3）的高效读写访问；复位模块用于复位总线上的外设。



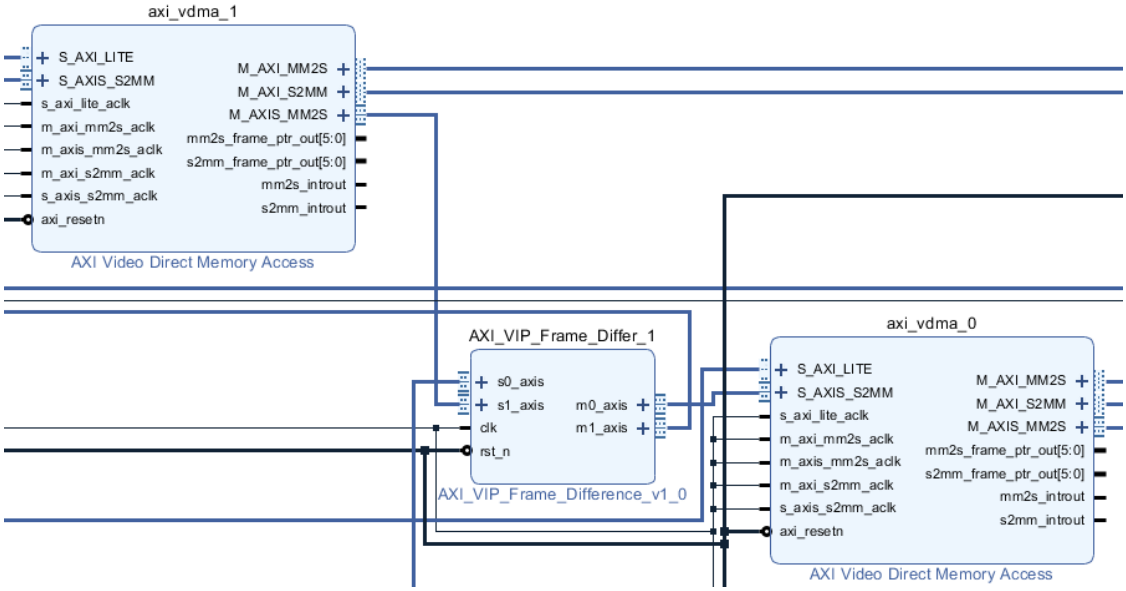
(a) PL 端系统整体连线



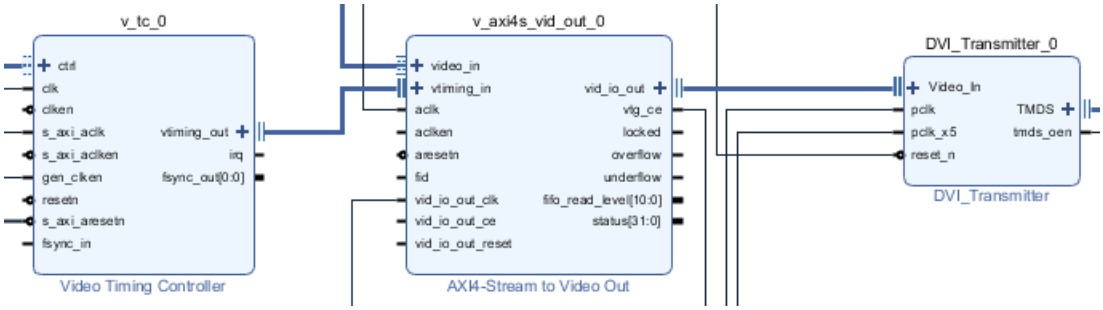
(b) AXI 总线互联模块



(c) 图像采集、格式转换模块



(d) 运动目标检测、VDMA 模块



(e) 显示模块

图 4-12 PL 端系统构建图

Fig. 4-12 PL side system construction diagram

完成后使用 Validate Design 功能进行验证是否有连线错误，验证结果如图 4-13 所示。

图 4-13 Validate Design 验证结果

Fig. 4-13 Validate Design Verifies the result

4.3.3 生成并导出比特流文件

在验证完系统设计不存在错误后，开始生成并导出比特流文件，流程图如 4-14 所示。首先选中完成的 Block Design 设计，选择“Generate Output Products”，选择 Global 模式，生成设计的综合、实现和仿真文件。然后执行“Create HDL Wrapper”，勾选自动更新顶层硬件描述语言（Hardware Description Language，HDL）模块；添加引脚约束，对像素时钟进行时序约束，绑定外设引脚，设置引脚的接口形式与电压大小。执行“Generate Bitstream”操作对设计进行综合、实现、生比特流文件；执行“Export hardware”导出硬件，并勾选包括比特流文件；执行 Launch SDK，在 SDK 中创建一个应用工程进入处理系统部分的设计。

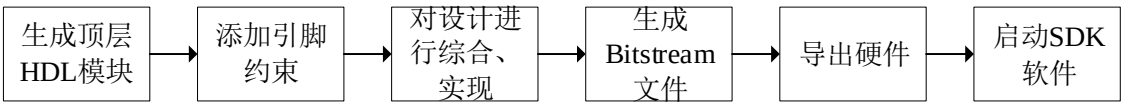


图 4-14 生成并导出比特流文件流程图

Fig. 4-14 Generate and export bitstream file flow charts

4.4 软件部分

4.4.1 OV5640 摄像头的配置

OV5640 具有多种功能，因此需要采用串行摄像头控制总线（SCCB）对其进行配置完成摄像头的驱动，使摄像头工作在预期的工作模式下以得到画质较好

的图像。由于 OV5640 的驱动只需在上电后执行一次，这一模块由处理系统来完成。OV5640 SCCB 写传输协议如图 4-15 所示。OV5640 SCCB 写传输协议先进行一次虚写操作，将使地址指针指向寄存器地址的位置，然后将 8 位的控制数据写入寄存器中。与集成电路总线（Inter Integrated Circuit，IIC）协议不同的是，SCCB 不需要判断否有应答信号 SCCB 不能进行连续读写，每操作一次都立即发出停止信号。

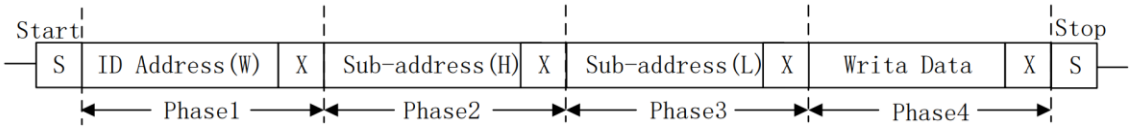


图 4-15 OV5640 SCCB 写传输协议

Fig. 4-15 OV5640 SCCB write transmission protocol

处理系统通过 SCCB 总线来配置 OV5640 摄像头的相关寄存器，使其输出的像素格式为 RGB565 格式，分辨率为 640*480，帧率为 60fps。由于 SCCB 总线的写传输协议与 IIC 协议很类似，可以直接使用 IIC 来完成摄像头的配置。

SCCB 接口总线通过 EMIO 连接到处理系统中，由处理系统实现 SCCB 的控制逻辑。首先对 EMIO 进行初始化，将 SCCB 端口设置为输出，并使能输出，将 SCCB 的 SCL 和 SDA 都拉高。编写相应函数分别实现 SCCB 的起始信号、停止信号、发送一个字节、接收一个字节和产生应答信号。例如 SCCB 的起始信号的产生过程为：将 SCCB 的 SCL 和 SDA 都拉高，延时 4μs，将 SDA 拉低，延时 4μs，将 SCL 拉低。通过调用这些函数来构成 SCCB 的读写操作的函数。

OV5640 上电后需要等待 20ms 才能对其进行配置，因此首先延迟 20ms。然后读取寄存器来获取 OV5640 摄像头的 ID，如果读取到正确的 ID，先将寄存器初始化为默认值，延时 1ms 后将电源休眠模式选为正常模式，开始以写相关寄存器的方式对 OV5640 摄像头进行配置，最终返回值为 0。在主函数中通过参数设置摄像头输出图像的分辨率，将参数传递给摄像头初始化函数，从而方便地从顶层修改摄像头的输出分辨率。

4.4.2 VDMA 的配置

VDMA 有两种配置方式，一种是直接操作寄存器，另一种是通过库函数。直接操作寄存器过程比较繁琐，本文通过 run_vdma_frame_buffer 函数配置 VDMA，该函数来源于 Xilinx 官方提供 VDMA 模板。run_vdma_frame_buffer 函数参数列表如图 4-16 所示，该函数最后一个参数用来配置 VDMA 的读写通道。

本系统使用该函数对分别对 VDMA0 和 VDMA1 进行配置，在主函数中给 VDMA 分配不同的帧缓存的起始地址，并通过全局变量传给 run_vdma_frame_buffer 函数。VDMA 的配置图如图 4-17 所示。

```
int run_vdma_frame_buffer(XAxiVdma* InstancePtr, int DeviceId, int hsize,
        int vsize, int buf_base_addr, int number_frame_count,
        int enable_frm_cnt_intr,vdma_run_mode mode)
```

图 4-16 run_vdma_frame_buffer 函数

Fig. 4-16 Run vdma frame buffer function

```
//配置VDMA
run_vdma_frame_buffer(&vdma0, VDMA_ID0, vd_mode.width, vd_mode.height,
                      frame_buffer_addr0,0,0,BOTH);
```

图 4-17 VDMA 的配置

Fig. 4-17 VDMA configuration

4.5 本章小结

本章主要完成了系统的总体设计，采用模块化设计的方法将系统分为图像采集模块、数据存储模块、算法处理模块，并采用软硬件协同设计的方法将各模块分配到处理系统与可编程逻辑分别实现。调用并配置相关 IP 搭建了硬件平台；完成了基于 OV5640 摄像头的图像采集模块；通过 VDMA IP 来完成 DDR3 存储器的数据的读写实现了数据存储模块；编写 DVI 驱动模块实现了 RGB 到 DVI 时序的转换，将处理结果实时显示在显示器上；在 Block Design 中将各个 IP 集成，生成比特流文件，导出到 SDK 中；在 SDK 中完成了处理系统端的设计。

第5章 系统调试与结果分析

5.1 系统调试

完成基于 Zynq 的运动目标检测系统后，为测试系统的性能，搭建了如图 5-1 所示的实验环境。图中 1 为显示器，2 为 SDK 软件调试界面，3 为 USB UART 接口，4 为 JTAG 下载器，5 为 HDMI 数据线，6 为 OV5640 摄像头模组，7 为 Zynq-7020 开发板，8 为电源线。其中 PC 通过两根 USB 数据线与开发板的进行连接，一根连接到 USB UART 口用于打印信息，另一根连接到 JTAG 下载器用于下载程序。摄像头连接在 Zynq 的扩展接口，通过 HDMI 线将检测结果实时输出到显示器中。完成硬件连接后，对系统进行调试，在 SDK 中将软件 ELF 文件和 BIT 文件下载到开发板中，可以利用 SDK 上的串口输出打印一些信息。

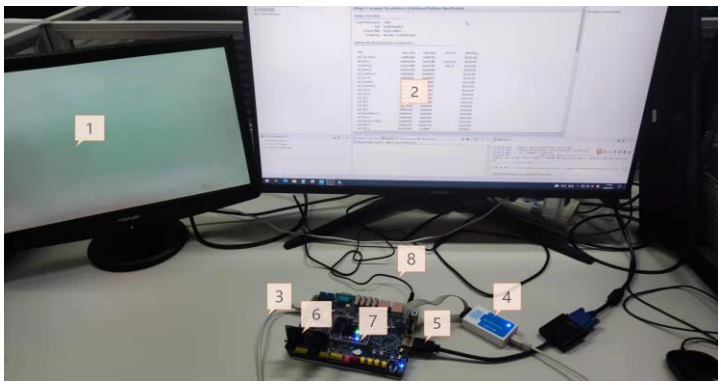
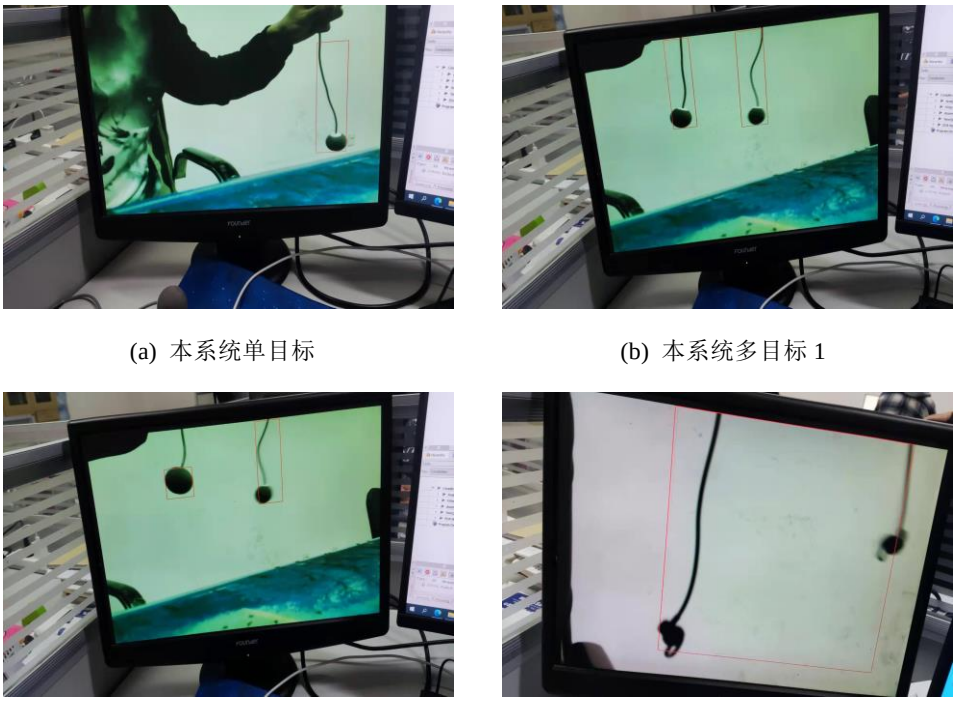


图 5-1 系统硬件连接图

Fig. 5-1 System hardware connection diagram

5.2 测试结果与分析

选择 UART 连接的对应端口，设置好波特率，点击运行后，串口打印出“OV5640 detected successful！”字样，说明摄像头正常工作。当场景中出现运动的物体时，显示器上就会实时显示被红色矩形框包围的运动目标，结果图是通过手机抓拍到的图像。测试结果如 5-2 所示。图 5.2(a)是本系统单目标检测的结果图，图 5.2(b)和图 5.2(c)是本系统多目标检测效果图，图 5.2(d)是传统方法多目标检测效果图，会将多个运动目标标记为一个运动目标。



(a) 本系统单目标　(b) 本系统多目标 1

(c) 本系统多目标 2　(d) 旧方法多目标

图 5-2 测试结果

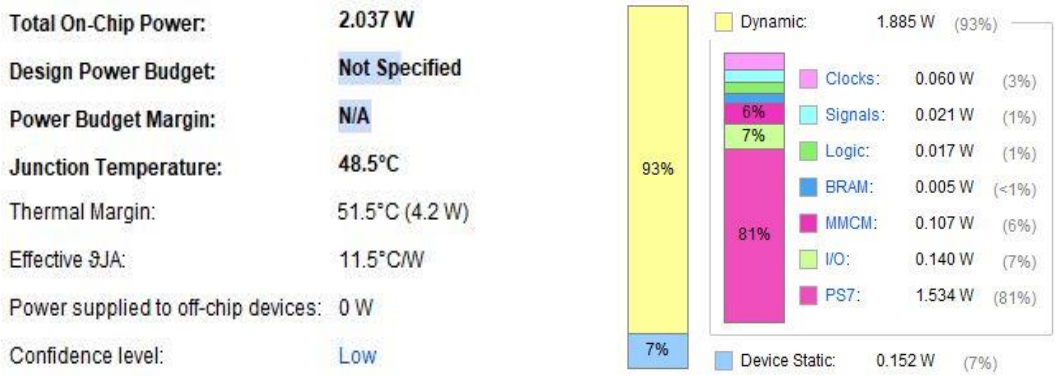
Fig. 5-2 test result

本系统可以更有效地标记多个运动目标，但是当两个运动目标间距较小的时候会标记为一个目标，这个间距是在程序中设置的。缩小间距可以解决这个问题，但是会出现将一个运动目标标记为多个运动目标的问题。本系统最多可以检测并标记 4 个运动目标，这是可以在程序中设置的，设置更多的目标会消耗更多的资源，本系统适用于稀疏多目标的检测。综上所述，本文完成了运动目标检测算法在 Zynq 平台的实现，能够有效地对多个稀疏目标进行标记，其中检测算法使用的是帧间差分法。

5.3 系统性能分析

5.3.1 功耗分析

功耗是保证嵌入式系统稳定可靠运行的重要指标。Vivado 在完成综合实现后点击“Report Power”可以生成详细的功耗报告，如图 5-3 所示。在系统功耗图中，系统的总功率为 2.037W，其中动态功耗为 1.855W，静态功耗仅为 0.152W，满足低功耗要求。



(a) 总功耗及温度信息　　(b) 各部分功耗信息

图 5-3 系统功耗

Fig. 5-3 The power consumption of the system

5.3.2 资源利用率分析

在 Zynq 可编程逻辑端的设计中，资源利用率是评估系统设计的一个重要指标，当所用某一项资源超过开发板的最大资源数时，需要采取办法调整资源占用情况或换为资源更丰富的开发板，资源占用率越低功耗越低，可以考虑在更加低功耗、经济的芯片中完成项目，同时对资源占用较小的工程进行时序分析更容易。使用 Vivado 搭建好硬件系统后，利用软件综合可以得到本系统的资源使用情况，资源利用率如图 5-4 所示，具体资源使用情况如表 5-1 所示。各类资源消耗均不超过 25%，为后续系统功能补充保留了大量的资源，由此证明本设计具有一定的优势。

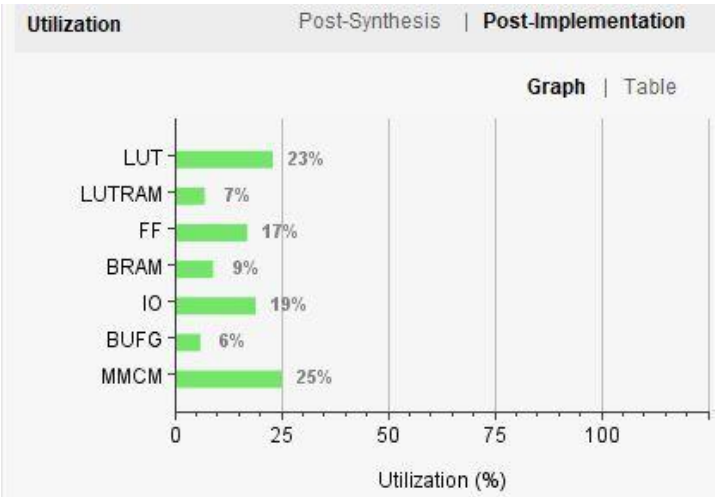


图 5-4 资源利用率

Fig. 5-4 resource utilization

表 5-1 资源利用统计

Tab. 5-1 Resource utilization statistics

资源	已使用资源	总资源数	资源利用率/%
LUT（查找表）	12476	53200	23.45
LUTRAM（分布式 RAM）	1158	17400	6.66
FF（触发器）	17742	106400	16.67
BRAM（块 RAM）	13	140	9.29
I/O（输入输出）	24	125	19.20
BUFG（全局缓冲）	2	32	6.25
MMCM（混合时钟管理器）	1	4	25.00

5.3.3 实时性分析

本系统的算法处理部分部署在 Zynq 的可编程逻辑部分，可编程逻辑采用多级流水线操作具有并行运算速度快的优势，主要体现在计算是并行的（单个时钟周期多条指令同时进行）、模块也是并行的，即每级流水线同时进行。可编程逻辑的时钟频率一般为几十到几百 MHz，虽然没有达到 PC 机的几 GHz 的数量级，但是可编程逻辑在进行图像处理时，多级流水线操作使得多个模块同时进行。PC 机在进行图像处理时是串行运算的，以帧为单位进行处理，只有等一帧数据处理完成才能进行下一级运算。可编程逻辑以像素为单位进行处理，比如灰度化后的像素就直接进入中值滤波的操作，而不必等到整帧图像都灰度化后再进行中值滤波。

本系统摄像头采集的图像分辨率为 640*480，一帧图像有 307200 个像素，在中值滤波、腐蚀、膨胀运算时生成 3*3 的像素阵列需要各自缓存 2 行数据，需要消耗 640*2*3=3840 个时钟周期，其他操作需要的时钟数均为个位数，可忽略不计。从一帧数据的第一个像素进入到流水线到一帧数据的最后一个像素离开流水线约需要 311040 个时钟周期，在 100MHz 的时钟频率下，处理一帧图像需要 311040/100M≈3.10ms，处理时间远低于一帧图像传输需要的 16.67ms，说明本系统可以实时处理分辨率 640*480，帧率为 60fps 的视频，满足实时性的要求。

5.3.4 系统性能对比

将本文设计的系统与国内外相关基于 FPGA 或 Zynq 的运动目标检测研究系统的性能作对比分析，系统性能对比结果如表 5-2 所示。从表中可以看出，只有文献[41]和文献[42]实现了多目标的检测，其中文献[41]使用了分区域的矩形标

记法，具有很大局限性；文献[42]采用了边缘标记法，标记效果不够明显。本系统采用基于邻域的标记法可以实现多目标的检测，并且结果显示效果更好，处理速度更快，一定程度上表明本系统具有优势。

表 5-2 系统性能对比结果表

Tab. 5-2 System performance comparison results table

文献	年份	是否为多目标检测	结果显示形式	处理的图像	是否能实时检测
文献[41]	2015	是	彩色图像	640*480@30fps	是
文献[45]	2019	否	二值化图像	752*480@60fps	是
文献[42]	2020	是	彩色图像	640*480@30fps	是
文献[47]	2020	否	灰度图	640*480@60fps	是
文献[44]	2021	否	灰度图	640*480@30fps	是
文献[72]	2021	否	彩色图像	640*480@30fps	是
本文	2022	是	彩色图像	640*480@60fps	是

5.4 程序固化

通过 JTAG 接口将 FPGA 配置文件和应用程序下载到 Zynq 器件中，这种做法每次上电都需要 PC 机将程序下载到开发板中，因此一般只在调试中使用。在通过测试后，将程序存储在非易失性存储器中，在每次上电或者复位后，程序自动运行，从而实现脱机运行。软件 ELF 文件和配置 FPGA 的比特文件可以存储到 PS 端的片外非易失性存储器，比如 SD 卡和外扩闪存（QSPI Flash）。前面在硬件系统中已经使能了 SD 卡与 QSPI Flash 引脚。

启动镜像的制作是在 SDK 中完成的。重新设置板级支持包（Board Support Package），勾选“xilffs”，以使用 FAT 文件系统。新建一个工程，并选择工程中的板级支持包，在模板中选择“Zynq FSBL”，执行编译生成 FSBL.elf 文件。

接下来使用 FSBL.elf 文件、BIT 文件、ELF 文件创建启动镜像，其中 BIT 文件是在 Vivado 中搭建硬件环境后手动生成的，ELF 文件是在 SDK 中编写软件程序后由软件自动编译生成的。选择“Create Boot Image”，在弹出的对话框中，选择输出文件的路径，按照顺序依次添加 FSBL.elf、BIT 文件、ELF 文件，如图 5-5 所示。

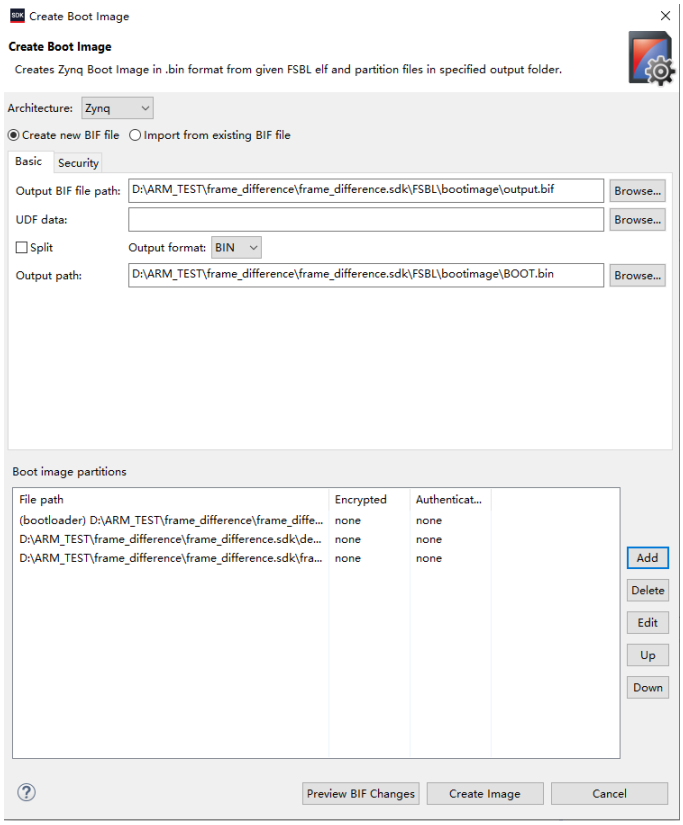


图 5-5 创建 Boot Image

Fig. 5-5 Create Boot Image

点击 Create Image 即可在之前设置的输出路径中看到 output.bif 和 BOOT.bin 文件，如图 5-6 所示，其中 output.bif 是 Boot Image 的配置文件，BOOT.bin 是启动文件。

图 5-6 生成的镜像文件

Fig. 5-6 Generated startup file

在 SDK 中选择“Program Flash”，选择生成的镜像文件 BOOT.bin 和 FSBL.elf 文件，点击“Program”，开始对 Flash 进行编程，将程序固化在外扩闪存中。关闭开发板电源，启动方式设置为 QSPI 启动，经测试，系统上电后程序可自动运行。另外，也可以使用内存卡的方式启动，将 BOOT.bin 拷贝到 SD

卡根目录下，将 SD 卡取出放置在开发板的 Micro SD 卡的卡槽中，启动方式选择为 SD Card，经测试，系统上电后程序自动运行。

5.5 本章小结

本章首先介绍了调试过程；其次与传统添加矩形框方法的结果进行了对比，对测试结果进行了分析；接着从功耗、资源利用率和实时性对系统性能进行了分析；最后阐述了制作启动镜像的过程，完成了程序固化。

第6章 总结与展望

6.1 全文总结

本文以多个运动目标的实时检测为研究目标，使用软硬件协同设计以及自顶向下模块化的设计思想，在 Zynq-7020 开发板上搭建了以图像采集、图像处理、数据存储和图像显示为框架的运动目标检测系统，改进了运动目标的标记方法，实现了多个运动目标的实时检测与标记，最后对系统性能进行了分析。主要做了以下工作：

(1) 从运动目标检测算法和视频图像处理平台两个方面，阐述了国内外运动目标检测技术的研究现状。简要分析了 Zynq 平台在图像处理领域的优势，明确了本文的研究方向是在 Zynq 上实现多个运动目标的实时检测。对硬件平台、一些关键技术、设计工具和设计方法进行了简单介绍。

(2) 通过对国内外文献进行系统调研，并对常用的运动目标检测算法进行了分析，通常以所得信息、鲁棒性、复杂度、实时性为指标，对三种常用运动目标检测算法进行了比较，选取了帧间差分法作为本文的运动目标检测算法。对数字图像处理技术中常用的图像处理技术进行了研究与分析，最终选取了灰度化、中值滤波、二值化、形态学滤波的方法。

(3) 采用模块化的设计方法设计了系统总体结构，将整个系统分为图像采集模块、算法处理模块、图像显示模块、数据存储模块。并使用软硬件协同设计将任务划分到可编程逻辑和处理系统两个部分。可编程逻辑实现图像处理的硬件算法加速，处理系统实现对外设的配置与各个 IP 的初始化，通过 AXI_HP 端口将图像数据缓存在 DDR3 中。调用相关 IP 搭建了基于 Zynq 的图像处理硬件平台。在片上集成了图像数据采集和 HDMI 驱动，进一步提高了系统的集成度。

(4) 设计了运动目标检测 IP 核，使用硬件描述性语言完成了图像的灰度化、中值滤波、差分处理、二值化、形态学滤波、添加矩形框的硬件算法加速。在该 IP 核中使用一种邻域的思想优化了运动目标的标记方法，即通过间距来判断是不是一个物体，并去除了重叠边框，实现了对多个运动目标的标记。在 SDK 中完成了摄像头的驱动、帧缓存的配置、以及各个 IP 核的初始化。

(5) 对系统进行了功能测试和分析。对比了传统 FPGA 运动目标检测的标记结果，实验表明本系统能够有效地实时检测并标记多个运动目标，功耗低，实时性高，资源占用率不高，为后续系统升级保留了大量资源。最后制作了启动镜像将程序固化，使系统能够上电自启。

6.2 工作展望

本文基本完成了静态背景下对多个运动目标的实时检测，并且能够对运动目标以添加矩形框的方式进行标记，系统的集成度高，满足低功耗的要求。但是由于个人能力有限，还有很多地方需要优化，后续能够进行的研究如下：

(1) 本系统只实现了静态背景下的多运动目标检测，没有采用比较复杂的检测算法，作用领域比较局限，之后可以融合其他算法对系统进行优化。

(2) 本系统在可编程逻辑中实现运动目标检测的 IP 核使用的是 Verilog 语言，该语言开发周期较长。下一步可以在 Vivado HLS 工具中使用 C、C++语言来设计 IP 核，从而提高抽象的层次，缩短开发周期。

(3) 下一步可以在处理系统中移植 Linux 系统，使用 OpenCV 与 Qt 对检测结果进一步分析，对运动目标进行分类。

参考文献

[1] 陆伟. 运动目标检测与跟踪算法的研究及应用[D]. 淮南市：安徽理工大学, 2016:1-3.

[2] 徐振雄. 视频中的运动目标检测与跟踪技术研究[D]. 武汉：武汉理工大学, 2018:2-3.

[3] Horn B K P, Schunck B G. Determining optical flow[J]. Artificial intelligence, 1981, 17(1-3): 185-203.

[4] Mahalingam V, Bhattacharya K, Ranganathan N, et al. A VLSI architecture and algorithm for Lucas–Kanade-based optical flow computation[J]. IEEE transactions on very large scale integration (VLSI) systems, 2009, 18(1): 29-38.

[5] Wang Z, Yang X. Moving target detection and tracking based on Pyramid Lucas-Kanade optical flow[C]. 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC). IEEE, 2018: 66-69.

[6] Gomaa A, Abdelwahab M M, Abo Zahhad M, et al. Robust vehicle detection and counting algorithm employing a convolution neural network and optical flow[J]. Sensors, 2019, 19(20): 4588.

[7] Sengar S S, Mukhopadhyay S. Detection of moving objects based on enhancement of optical flow[J]. Optik, 2017, 145: 130-141.

[8] Wren C R, Azarbayejani A, Darrell T, et al. Real-time tracking of the human body[J]. IEEE Transactions on pattern analysis and machine intelligence, 1997, 19(7): 780-785.

[9] Stauffer C, Grimson W E L. Adaptive background mixture models for real-time tracking[C]. Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition. IEEE, 1999, 2: 246-252.

[10] Barnich O, Van Droogenbroeck M. ViBe: a powerful random technique to estimate the background in video sequences[C]. 2009 IEEE international conference on acoustics, speech and signal processing. IEEE, 2009: 945-948.

[11] Mabrouk L, Houzet D, Huet S, et al. Single core SIMD parallelization of GMM background subtraction algorithm for vehicles detection[C]. 2018 IEEE 5th International Congress on Information Science and Technology (CiSt). IEEE, 2018: 308-312.

[12] Jain R, Nagel H H. On the analysis of accumulative difference pictures from image sequences of real world scenes[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1979, 2: 206-214.

[13] Lipton A J, Fujiyoshi H, Patil R S. Moving target classification and tracking from real-time video[C]. Proceedings fourth IEEE workshop on applications of computer vision. (WACV'98).

IEEE, 1998: 8-14.

[14] Hsu Y Z, Nagel H H, Rekers G. New likelihood test methods for change detection in image sequences[J]. Computer vision, graphics, and image processing, 1984, 26(1): 73-106.

[15] Abughalieh K M, Sababha B H, Rawashdeh N A. A video-based object detection and tracking system for weight sensitive UAVs[J]. Multimedia Tools and Applications, 2019, 78(7): 9149-9167.

[16] Husein A M, Halim D, Leo R. Motion detect application with frame difference method on a surveillance camera[C]. Journal of Physics: Conference Series. IOP Publishing, 2019, 1230(1): 012017.

[17] Wang S, Kong P, Wang M. Inter-Frame Differences and Convolution Neural Network based Abnormal Crowd Event Recognition[C]. Proceedings of the 2nd International Conference on Digital Signal Processing. 2018: 32-36.

[18] Nallasivam M, Senniappan V. Moving human target detection and tracking in video frames[J]. Studies in Informatics and Control, 2021, 30(1): 119-129.

[19] Sun W, Sun M, Zhang X, et al. Moving vehicle detection and tracking based on optical flow method and immune particle filter under complex transportation environments[J]. Complexity, 2020, 2020.

[20] 乐英，赵志成. 基于背景差分法的多运动目标检测与分割[J]. 中国工程机械学报，2020, 18(4): 305-309.

[21] Liu Ling, Chai Guo hua, Qu Zhong. Moving target detection based on improved ghost suppression and adaptive visual background extraction[J]. Journal of Central South University, 2021, 28(3).

[22] 李明月. 基于背景建模的运动目标检测算法的研究与实现[D]. 西安: 西安电子科技大学, 2018: 21-26.

[23] 何银飞. 基于改进的帧差法和背景差法实现运动目标检测[D]. 秦皇岛: 燕山大学，2016: 23-25.

[24] 王梦菊，吴小龙，杜海涛. 基于背景差分与帧间差分的目标检测改进算法[J]. 自动化技术与应用, 2018, 37(10): 89-92,96.

[25] 张丽平，范红，王璐瑶，等. 利用光流法实现运动目标提取的研究[J]. 计算机与数字工程, 2020, 48(1): 83-87.

[26] 赵淑胜. 光照变化环境下的运动目标检测和跟踪方法研究[D]. 济南: 山东建筑大学, 2018: 23-26.

[27] Zhang Wei, Sun Wenhua. Research on small moving target detection algorithm based on

complex scene[J]. Journal of Physics: Conference Series, 2021, 1738(1).

[28] Cho J, Jung Y, Kim D S, et al. Moving object detection based on optical flow estimation and a Gaussian mixture model for advanced driver assistance systems[J]. Sensors, 2019, 19(14): 3217.

[29] Kumar K, Nandan D, Mishra R K. Compact Hardware of Running Gaussian Average Algorithm for Moving Object Detection Realized on FPGA and ASIC[J]. Rev. d'Intelligence Artif, 2019, 33(4): 305-311.

[30] 郑淋萍. 基于嵌入式平台的复杂背景运动目标检测与提取[D]. 上海: 上海师范大学, 2020: 30-32.

[31] Yi Q. Design of Moving Object Detection System Based on FPGA[C]. 2018 10th International Conference on Communications, Circuits and Systems (ICCCAS). IEEE, 2018: 435-438.

[32] Sridevi N, Meenakshi M. An Efficient FPGA based Background Subtraction Algorithm for Object Detection[C]. 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2018: 397-401.

[33] Correia M V, Campilho A C. Real-time implementation of an optical flow algorithm[C]. 2002 International Conference on Pattern Recognition. IEEE, 2002(4): 247-250.

[34] Zhang S, Zhang T, Li Z, et al. Scale adaptive infrared small target detection with patch contrast measure[C]. Automatic Target Recognition and Navigation. 2020: 89-97.

[35] 高文刚, 李锦明, 张虎威, 等. 基于 DSP 的运动目标检测系统[J]. 仪表技术与传感器, 2016, 11: 54-57.

[36] Jianhui Gao. Research and Design of Moving Target Detection and Tracking System Based on Web Server[C]. Proceedings of the 5th International Conference on Communication, Image and Signal Processing (CCISP 2020), 2020: 264-268.

[37] 宋翰林. 基于 ARM 的运动目标检测与跟踪算法的实现[D]. 哈尔滨: 哈尔滨工程大学, 2017: 34-42.

[38] 何国锋. 基于嵌入式的运动目标检测与追踪[D]. 贵州: 贵州大学, 2017: 15-18.

[39] 邵鹏, 杨晨, 张晋敏. 基于 FPGA 的自适应阈值运动目标检测[J]. 应用光学, 2017, 38(6): 903-909.

[40] 张棋. 基于 FPGA 的运动目标检测与跟踪系统设计[D]. 南京: 南京理工大学, 2018: 23-24.

[41] 林培杰, 郑柏春, 陈志聪, 等. 面向多区域视频监控的运动目标检测系统[J]. 液晶与显示, 2015, (3): 484-491.

[42] 邢凯. 基于 FPGA 的多运动目标检测技术研究[D]. 昆明: 昆明理工大学, 2020: 19-20.

[43] 郭铮，张起贵. 基于 FPGA 的高清目标快速检测系统设计[J]. 电子设计工程, 2021, 29(04): 150-154.

[44] 王俊龙. 基于 FPGA 的运动目标检测系统的设计[D]. 太原：中北大学, 2021: 12-16.

[45] 周佩. 基于 Zedboard 的运动目标检测系统的实现[D]. 内蒙古：内蒙古大学, 2019: 31-32.

[46] 李鹏. 基于 FPGA 的运动目标检测算法实现[D]. 西安：西安电子科技大学, 2020: 13-16.

[47] 陈科成，苏成悦，何雷，等. 基于 ZYNQ 架构的运动物体检测系统[J]. 信息技术与信息化, 2020, 3: 63-66.

[48] 张祖昊，王云光. 基于 Zynq-7000 的实时视频图像处理系统框架设计[J]. 软件导刊, 2021.

[49] 张俊杰. 基于 Zynq 的动态图像测量处理系统设计与实现[D]. 西安：西安电子科技大学,2020.

[50] Ji Qingbo, Dai Chong, Hou Changbo, et al. Real-time embedded object detection and tracking system in Zynq SoC[J]. EURASIP Journal on Image and Video Processing, 2021, 2021(1).

[51] Kortli Yassin, J ridi Maher, Merzougui Mehrez, et al. Optical face detection and recognition system on low-end-low-cost Xilinx Zynq SoC[J]. Optik, 2020, 217(C).

[52] Younis D B, Younis B M K. Low Cost Histogram Implementation for Image Processing using FPGA[C]. IOP Conference Series: Materials Science and Engineering. 2020, 745(1): 012044.

[53] 赵莉，李司. 基于 ZYNQ 和 AD9361 的软件无线电平台设计与实现[J]. 移动通信，2018, 42(12): 63-67,73.

[54] 高辉，于恒. 基于 Vivado HLS 的特征点坐标提取和 AXI4-Stream 接口高速传输[J]. 信息技术, 2020, 44(04): 27-31.

[55] 唐成武. 基于 AXI4 总线的 DDR3 高速存储接口系统设计[D]. 武汉：华中科技大学, 2020: 13-15.

[56] 李启慧. 基于 ZYNQ 的摄像头采集系统设计与实现[J]. 电子设计工程, 2020, 28(08): 108-113.

[57] 张振利，韩凌锋. 基于 FPGA 的 HDMI 视频显示系统设计[J]. 现代电子技术，2021, 44(16): 35-39.

[58] 蔚瑞华，余有灵，张伟，等. 基于模块化思想的 FPGA 综合实验项目设计[J]. 实验技术与管理, 2016, 33(05): 44-47.

[59] 黄张祥，白瑞林，吉峰. 基于 SoC 软硬件协同设计的布匹瑕疵检测[J]. 光学技术，2017, 43(01): 50-55.

[60] Abdallah El Hamidi, Marwan Saleh, Nicolas Papadakis, et al. A proper generalized decomposition approach for optical flow estimation[J]. Mathematical Methods in the Applied Sciences, 2020, 43(08).

[61] Jeevith S.H., Lakshmikanth S. Detection and tracking of moving object using modified background subtraction and Kalman filter[J]. International Journal of Electrical and Computer Engineering, 2021,11(01).

[62] 慈文彦. 运动目标检测方法综述[J]. 信息技术, 2016, 12: 93-96,100.

[63] Khuhro M A, Huang D, Huang S, et al. A modified Gaussian mixture background model for moving object detection[J]. Journal of Computational and Theoretical Nanoscience, 2017, 14(08): 3672-3678.

[64] 贾亮, 张武臣. 基于帧间差分法的目标检测研究与 FPGA 实现[J]. 电脑与信息技术, 2021, 29(02): 20-23.

[65] 裴正雄, 彭安金. 基于 FPGA 的图像预处理系统[J]. 信息通信, 2019, 12: 79-80.

[66] 肖汉, 郭宝云, 李彩林, 等. 基于 OpenCL 的图像灰度化并行算法研究[J]. 江西师范大学学报(自然科学版), 2020, 44(05): 462-471.

[67] 彭姝姝. 基于均值滤波和小波变换的图像去噪[J]. 现代计算机, 2019, 12: 62-67.

[68] 马炼, 李林. 一种针对椒盐噪声的高速自适应中值滤波算法[J]. 计算机时代, 2021, 10: 68-71.

[69] 阳欣, 魏可, 宋宇鲲, 等. 图像二值化处理硬件加速引擎的设计[J]. 合肥工业大学学报(自然科学版), 2021, 44(11): 1495-1499,1517.

[70] 张蜀红. 改进 Otsu 算法的镀锌板表面缺陷检测方法[J]. 计算机测量与控制, 2021, 29(12): 57-61.

[71] Boato G, Dang-Nguyen D T, De Natale F G B. Morphological filter detector for image forensics applications[J]. IEEE Access, 2020, 8: 13549-13560.

[72] 刘汝卿, 李锋, 蒋衍, 等. 基于 FPGA 的运动目标实时检测系统设计[J]. 计算机测量与控制, 2022, 30(4): 56-59.

致　谢

时光飞逝，三年的硕士研究生学习生涯已经接近尾声。在论文即将完稿之际衷心地感谢三年来所有鼓励我、关心我和帮助过我的所有人。

首先要感谢我的导师林青松。在这三年中，林老师给予我最大程度的科研支持，为我置办了科研工具和一些基础的学习用品，为我提供了很好的科研环境。林老师科研能力强、对待工作一丝不苟，思维活跃。在最初确定研究方向时，林老师针对我读研是为了更好就业的目的，结合时代热点与社会需求，给我选取了 Zynq 方向，这也让我找工作的时候更具有优势。平时在科研中遇到了困难，林老师总是耐心地为我指点迷津。在后期的学位论文撰写和修改过程中，林老师提出来许多宝贵的修改意见。在这三年期间，林老师的谆谆教导对我产生了潜移默化的影响，我将受益终生。衷心地感谢林老师的教导与帮助。

其次要感谢研究生期间共同成长的朋友们。感谢已经毕业的刘怡明、李文生师兄们，他们无私分享了很多 Zynq 学习和开发经验，让我可以很快地解决科研路上遇到的问题，少走了很多弯路。感谢邓鹤、刘存领、于洪泽师弟们，他们在我写论文之际提供了许多帮助。感谢 427 的室友黄萌、黄伟伟、卢越生活上对我的包容。感谢他们的陪伴，让我的研究生生活变得多姿多彩。

然后感谢我的家人，是他们一直在背后支持并鼓励着我。我只想成为他们的骄傲，我将带着他们殷切地期望披荆斩棘，闯出一片天地来报答他们的养育之恩。

最后，感谢在感谢论文评阅和答辩评委组的各位老师和专家。

攻读学位期间的研究成果

[1] Shuai Zhang, Qingsong Lin, Cunling Liu, and Hongze Yu. Moving object detection based on Zynq [J], Scientific Journal of Intelligent Systems Research.（已录用）