分类号_____ 学号 M202072764

学校代码 10487 密级_____

华中科技大学

硕士学位论文

（学术型 ☑ 专业型 □）

基于国产器件的融合信息处理系统设计及跟踪算法实现

学位申请人：裘剑东

学 科 专 业：控制科学与工程

指 导 教 师：桑红石 副教授

答 辩 日 期：2023 年 5 月 15 日

A Thesis Submitted in Partial Fulfillment of the Requirements for the Master Degree in Engineering

The Information-Fused Processing System Design and Tracking Algorithm Implementation Based on Domestic Components

Candidate : QIU Jiandong

Major : Control Science and Engineering

Supervisor : Assoc.Prof. Sang Hongshi

Huazhong University of Science and Technology

Wuhan 430074, P. R. China

May, 2023

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：裘剑东

日期：2023年5月24日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于 保　密□，在_____年解密后适用本授权书。

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：裘剑东　　指导教师签名：聚红石

日期：2023年5月24日　　日期：2023年5月24日

摘 要

在复杂环境下，单一的传感器容易受到干扰而导致目标检测跟踪结果产生偏差。红外激光双模融合的目标检测跟踪技术能够实现更准确可靠的跟踪性能。为了在嵌入式平台上开展相关算法的初步研发，本文设计了一种基于国产器件的融合信息处理系统。根据设计需求与初步的目标检测跟踪方案，系统主要由一片高性能数字信号处理器（Digital Signal Processor, DSP），一片自研的专用协处理器和两片资源丰富的现场可编程门阵列（Field Programmable Gate Array, FPGA）组成。DSP 是系统中算法执行的主体部分；协处理器用于协助 DSP 完成特定计算任务；在 FPGA 内设计了灵活的硬件加速框架，硬件加速电路可以方便地接入系统并由 DSP 调用。

基于所设计的融合信息处理系统，建立起了用于红外激光融合目标检测跟踪的多核 DSP 软件框架，实时实现了核相关滤波（Kernel Correlation Filter, KCF）目标跟踪算法。多核 DSP 软件框架为后续算法的研发规划了方向；所实现的 KCF 算法仅需 12.6 ms 即可完成对任意大小目标的跟踪，利用两个内核同时执行 KCF 算法跟踪两个不同的目标也仅需 13.3 ms。

设计了扩展性强，数据传输效率高，软件编程便捷的算法硬件加速框架，可用于接入专用计算电路解决算法实时实现瓶颈；基于此框架搭建了目标跟踪算法验证平台，并且完成了四种在 DSP 上运行的跟踪算法的完整验证与评估。算法验证平台完全嵌入公开数据集的测试平台中，用户能够像评估上位机软件跟踪算法的性能一样，通过测试脚本自动对 DSP 上运行的目标跟踪算法进行完整测试；DSP 与上位机之间的数据传输速率达到 1 GB/s 左右，面对大量的测试序列也可以快速完成。

关键词： 软硬件协同设计；DSP；FPGA；视觉目标跟踪；KCF

Abstract

In complex environments, a single sensor is susceptible to interference that can skew the target detection tracking results. Infrared laser dual-mode fusion object detection and tracking technology can achieve more accurate and reliable tracking performance. In order to carry out the preliminary research and development of related algorithms on the embedded platform, this paper designs a fusion information processing system based on domestic components. According to the design requirements and preliminary object detection and tracking scheme, the system mainly adopts a high-performance Digital Signal Processor (DSP), a self-developed dedicated coprocessor and two resource-rich Field Programmable Gate Array (FPGA). DSP is the main part of algorithm execution in the system; The coprocessor are used to assist DSP with specific computational tasks ; A flexible hardware acceleration framework is designed within the FPGA, and the hardware acceleration circuit can be easily connected to the system and called by the DSP.

Based on the designed fusion information processing system, a multi-core DSP software framework for infrared laser fusion target detection and tracking is established, and the real-time Kernel Correlation Filter (KCF) target tracking algorithm is implemented. The multi-core DSP software framework plans the direction for the development of subsequent algorithms. The implemented KCF algorithm can complete the tracking of any size target in only 12.6 ms, and the KCF algorithm can be executed simultaneously with two cores to track two different targets in only 13.3 ms.

A hardware acceleration framework with strong scalability, high data transmission efficiency and convenient software programming is designed, and a target tracking algorithm verification platform is built based on this framework, and the complete verification and evaluation of four tracking algorithms running on DSP are completed. The algorithm verification platform is fully embedded in the test platform with public data sets, and users can automatically test the target tracking algorithm running on the DSP through the test script like evaluating the performance of tracking algorithm running on host computer; The data transmission rate between the DSP and the host computer reaches about 1 GB/s, and the test can be done quickly in the face of a large number of test sequences.

Keywords: Software and hardware co-design, DSP, FPGA, Vision Object Tracking, KCF

目 录

1 绪论

1.1 研究背景与意义

现实场景纷繁复杂，单一的视觉传感器容易受到外界环境影响而失效，导致目标检测跟踪结果不可靠。多传感器融合的目标检测跟踪算法是解决这一问题的一种重要思路。红外探测器能够捕获画面中的纹理细节，不论昼夜均能工作；通过激光雷达获取场景中目标的距离信息，能够丰富目标特征的表达。红外激光双模融合目标检测跟踪技术充分结合两种传感器的优势，从而实现更加稳定可靠的目标跟踪。

激光雷达的技术发展日新月异，从单点测距到三维成像，在红外与激光融合的目标检测跟踪技术中起到越来越重要的作用。随着激光雷达成像范围不断增大，新的设计需求要求系统能够接收并实时处理每秒 20000 帧 128×128×12 bit 的激光雷达原始数据和每秒 50 帧 640×512×8 bit 的红外图像数据，同时要求在复杂场景中实现高识别准确率与跟踪精度。这些需求增加了目标识别跟踪算法的复杂度和计算量，对融合信息处理系统的数据传输带宽和实时信息处理能力均提出挑战。课题组此前已有相关技术积累，但原有的融合信息处理系统已无法满足新的设计需求，因此设计新的性能更优，扩展性更强的红外激光融合信息处理系统具有重要意义。

在红外图像中针对单一目标的跟踪算法种类繁多，目前最新的性能最优的跟踪算法往往存在计算过程复杂，计算量大的问题。核相关滤波（Kernel Correlation Filter, KCF）算法计算过程简明，目标跟踪性能良好，在一般的嵌入式平台中都能实现较高的实时性。红外激光融合信息处理系统计算任务繁重，不仅需要完成目标的检测识别，还需要对多个目标同时跟踪，因此选用 KCF 算法作为本系统的主体跟踪方案具有较高的可行性。基于本系统实时实现 KCF 跟踪算法，并建立起红外激光融合信息处理系统的多核 DSP 软件框架，能够为后续工作的开展奠定基础。

将 KCF 算法移植到融合信息处理系统中实现，需要对算法的性能进行验证与评估。现有的评估方案主要分为三类，一类是在上位机编写跟踪算法的软件模型，间接评估算法的跟踪性能；一类是采用摄像头实时采集图像序列，由人对实际的跟踪结果

进行主观评价；另外只有少量文献中提到通过上位机下发图像序列，对跟踪算法的性能直接评估。受到现有评估方案的启发，为了完成算法移植也需要设计一套完整的跟踪算法评估验证平台。使移植后的跟踪算法能够像上位机软件算法一样，通过公开数据集进行直接且全面的验证与评估，便于为后续的算法研发明确方向。

1.2 国内外研究现状

1.2.1 红外激光信息融合处理系统

一种较常见的信息融合方式是通过红外摄像头确定目标的方位，再利用激光雷达测距，解算出目标的三维坐标后可以利用卡尔曼滤波等滤波算法对目标的坐标进行跟踪预测。比如早在 2009 年潘丽娜提出的激光红外传感器联合跟踪算法能够对匀速运动目标准确定位[1]；2015 年郝静雅等人提出的融合算法解决了红外激光数据异步的问题[2]；2022 年彭涛在滤波跟踪算法方面进行改进，实现了对复杂背景下的多目标跟踪[3]。这些做法都只利用了探测器与目标之间单一的距离信息，而还没有将激光雷达三维成像技术应用于目标跟踪。

近年来，激光雷达在自动驾驶领域已有广泛应用。车载激光雷达的作用范围一般不超过 200 米，但它采用探测器阵列旋转扫描的方式，能够获取全方位的距离信息，为目标的识别定位提供了更多依据。比如 2021 年战荫泽等人融合红外与激光点云数据，降低了车辆目标识别的漏检率和误检率[4]；同年郑欣悦等人采用 Yolov3 识别红外图像中的目标，结合激光点云信息提高了目标识别的准确率和相对定位精度[5]。近距离的激光雷达三维成像技术已投入商业化应用，但想要提升激光雷达的作用距离，实现远距离目标的三维成像跟踪仍需克服许多困难。

雪崩光电二极管是一种高灵敏度的光电探测器件，在其两端施加超过击穿电压的反向偏压时，能够实现单光子探测，它的这一工作模式称为“盖革模式”。能够正常工作在盖革模式的雪崩光电二极管称为单光子雪崩光电二极管（Single-photon Avalanche Photodiode, SPAD）。得益于它的高灵敏度，SPAD 能够实现远距离的目标探测，未来应用场景广泛，目前已是国内外的重点研究方向。2020 年黎正平等人实现了 202 km 的单光子远距离成像[6]，创造了世界纪录。2021 年康英杰等人将单光子

成像激光雷达与红外传感器融合，提出了利用红外图像来补全激光点云数据的方法，完善了对目标三维信息的表达[7]，但没有实现目标检测跟踪的应用。

完整的红外激光信息融合处理系统至少包括红外成像系统，激光探测系统，随动控制系统和信息处理系统。红外与激光探测器的接收光路需要精心设计，确保它们共光轴；随动控制系统能够快速响应并且精准指向；信息处理系统能够实时处理红外与激光数据，给出目标的跟踪结果。2014 年颜洪雷博士完成了红外与激光复合探测系统样机的设计[8]。该样机采用 SPAD 激光雷达，能够实现远距离测距，但还无法做到三维成像；由于经费与时间的限制，系统中采用的是非制冷式红外探测器，若采用制冷式红外探测器，还能够降低噪声，实现更远距离的探测。2017 年宋盛博士对该系统进一步完善，提出了一种针对红外弱小目标的检测方法，提升了激光探测器的响应速度[9]。2020 年本课题组的刘羽丰设计了基于红外图像与激光三维成像结果的车辆目标检测识别跟踪算法[10]，并在嵌入式系统上实现。整个系统由多方单位合作完成，本课题组主要负责红外与激光数据的融合处理，当时所需处理的激光原始数据同样是每秒 20000 帧，但每帧大小 64×64×12 bit，仅为新需求的四分之一。课题组设计了相应的融合信息处理板并开展融合算法的嵌入式实现工作。当时的系统搭载了两片 TI 公司的 TMS320C6455 DSP 和两片 Xilinx 的 XC7K410T FPGA，在目前看来不论是处理器性能还是逻辑资源都有提升空间。新一代融合信息处理系统的设计以更强的处理器算力，更丰富的逻辑资源与更灵活的系统互连为目标，为后续的融合算法研发提供高性能的嵌入式平台。

1.2.2 嵌入式平台实时目标跟踪

要在嵌入式平台上实现目标跟踪算法也需要对当前算法的研究趋势有一定的了解。视觉目标跟踪（Visual Object Tracking, VOT）挑战赛是视觉目标跟踪领域的国际顶尖赛事，2022 年的 VOT 挑战赛划分了多个赛道，包括短序列跟踪，长序列跟踪，实时序列跟踪等。不同类型的序列考验跟踪算法性能的侧重点不同，短序列跟踪侧重于跟踪算法对目标的跟踪精度；长序列跟踪侧重于跟踪算法的抗干扰能力；实时序列跟踪侧重于算法的实时性。VOT 挑战赛的结果全方位地反映了现今性能最领先的视觉目标跟踪算法，2022 年 VOT 挑战赛的结果表明，目前性能最优的跟踪算法大多基

于 Transformer 模型[11]。基于 Transformer 模型的目标跟踪算法近两年不断涌现，很快便成为了主流的目标跟踪算法，在 2020 年的 VOT 挑战赛上，参加长序列跟踪赛道与短序列跟踪赛道的算法只有两类，分别基于判决式相关滤波（Discriminative Correlation Filter, DCF）或基于孪生网络（Siamese Network, SN）[12]。而到了 2022 年，参加短序列跟踪赛道的跟踪算法中有一半是基于 Transformer 模型，40%基于 DCF，剩余 10%基于 SN。除此之外，在长序列跟踪赛道上排名前三的跟踪算法，实时序列跟踪赛道上的冠军算法也是基于 Transformer 模型[11]。但目前这些性能领先的目标跟踪算法基本都需要用到 GPU 加速，借助 AI 边缘计算平台 Jetson TX2，Nousi 等人实现了 SiamRPN 目标跟踪算法在嵌入式系统中的部署，能够达到 15 FPS（Frame Per Second）的处理速度[13]，这一实时性在一般的嵌入式系统上很难做到，而且 15 FPS 的处理速度在大部分情况下也是远远不够的。

基于 DCF 的目标跟踪算法是 VOT 挑战赛中比较常见的算法。DCF 算法中，KCF 算法[14]是一座里程碑。它从理论上完整地证明了相关滤波的计算过程相当于在目标周围密集采样，离散傅里叶变换能够加速其计算过程，使得 DCF 算法能够通过简单的计算获得精确的跟踪结果。KCF 算法在 2014 年的 VOT 挑战赛中是目标跟踪准确率最优的算法之一[15]。后续几年中，不断有学者基于 KCF 算法提出新的算法，比如 SRDCF 算法在 KCF 的基础上引入额外的正则项约束，抑制了图像的边界效应[16]；C-COT 算法引入由卷积神经网络提取的深度特征，不同分辨率的深度特征图提高了跟踪的准确率[17]；DiMP 算法将提取深度特征的模型改为在线训练的模式，更好地利用了背景信息[18]。这些改进的算法虽然提升了跟踪结果的准确性与稳定性，但也引入了更多复杂的运算，失去了 KCF 算法原有的计算效率高的特点，不利于嵌入式平台的实现。而 KCF 算法本身已经能够处理对于一般刚体目标的跟踪需求，因此本系统中采用 KCF 算法跟踪红外图像中的目标。

目标跟踪算法实时实现的性能很大程度上依赖于嵌入式平台的选择。近年来随着半导体行业逆全球化趋势加剧，从国外进口高端处理器受限，以往一些采用国外高端器件实现的系统都存在着极大的安全隐患，在国防军事领域这一问题尤为凸显。因此在处理器的选型上，尽可能采用国产器件推行国产化是目前的趋势。海思 Hi3559

片上系统（System on Chip, SoC）采用 ARM 内核，主频 1.8 GHz，集成四核 GPU 与双核 NPU，专用于图像信号处理；瑞芯微 RK3399 采用类似的 ARM 内核，集成更高性能的 GPU，广泛应用于工业和消费领域的各种终端设备。国内还有许多研发 AI 处理器的公司，比如全志科技，嘉楠科技，寒武纪，地平线等，它们所推出的处理器也都是面向工业控制，汽车电子，智能家居等嵌入式应用。这些处理器为产品开发提供了很大便利，但同时也失去了一定的灵活性。基于 DSP 与 FPGA 的硬件平台能够为用户提供极大的自由度，不论是 DSP 内的软件还是 FPGA 内的逻辑电路，用户都可以根据自己的需求来设计；另一方面，KCF 算法的计算过程主要涉及传统的数字信号处理技术，也比较适合采用 DSP 来实现。在国外的出口禁令推行之前，已有不少系统采用了国外的高端芯片进行设计，为了系统的延续性，且避免推翻先前的设计工作，这部分高端芯片需要能够原位替代或功能替代。银河飞腾 FT-M6678 对标 TI 的 TMS320C6678 DSP，是一款国产的高性能 DSP 芯片，在实现对国外芯片的功能替代的基础上，又扩宽了 EMIF（External Memory Interface）总线位宽，新增了硬件快速傅里叶变换（Fast Fourier Transform, FFT）加速器。复旦微电的 JFM7VX690T 对标 Xilinx 的 XC7VX690T，能够实现对国外芯片的原位替代。本系统的设计主要以这两款国产高性能器件为主。

表 1-1 近些年基于 DSP 实现的 KCF 目标跟踪算法处理速度

年份	处理器	帧率（FPS）	说明
2021	1 个 C64x+内核	42.0[19]	跟踪 64×64 大小的目标
2019	1 个 C64x 内核	25.0[20]	
2018	1 个 C64x 内核	7.2[21]	跟踪 128×128 大小的目标
2020	1 个 C674x 内核	46.6[22]	跟踪 80×60 大小的目标
2021	1 个 C66x 内核	26.3[23]	可见光+红外图像处理
2021	1 个 C66x 内核	42.2[24]	采用 FFT 加速，自适应调整学习率
2020	1 个 C66x 内核	25.0[25]	跟踪 128×128 大小的目标
2019	1 个 C66x 内核	40.0[26]	跟踪 64×128 大小的目标
2017	1 个 C66x 内核	41.4[27]	跟踪 32×64 大小的目标
2021	2 个 C66x 内核	53.0[28]	能够自适应调整尺度
2018	2 个 C66x 内核	63.0[29]	跟踪 49×27 大小的目标
2021	8 个 C66x 内核	67.5[30]	8 核流水线方式实现
2020	8 个 C66x 内核	75.0[31]	OpenMP 8 核
2019	8 个 C66x 内核	94.8[32]	跟踪 47×102 大小的目标，OpenMP 8 核
2020	Vision DSP	164.8[33]	跟踪 106×222 大小的目标
2020	AM5708 SoC	20.0[34]	基于 Linux 上位机软件开发

采用 DSP 实现 KCF 目标跟踪算法的案例有很多。TI 公司的 C6000 系列 DSP 占据了国际市场上高性能 DSP 的绝大部分份额，是目前主流的 DSP。Cadence 公司于 2021 年推出的高端 DSP Vision Q8，专门面向人工智能应用，其指令字长达 1024 bit，远超 TI 的 C66x 内核采用的 128 bit，但时钟频率最高为 600 MHz。因此 Cadence 公司推出的 DSP 更加适用于大规模并行计算的场合。TI 公司提供的较常见的高性能内核性能从高到低依次有 C66x，C674x，C64x+等，表 1-1 列举了近些年基于 DSP 实现 KCF 目标跟踪算法的处理速度。这些方案中很多采用了多核 DSP，但方案介绍中没有多核程序设计相关的内容则默认其只用了单个内核。DSP 上目标跟踪算法的处理速度与很多因素有关，比如软件优化等级，存储器的分配方式，具体的算法执行流程等等。表 1-1 中的方案对 KCF 算法进行了不同程度的修改，实验测试的流程也不尽相同，因此他们的处理速度并不能直接比较。但这些结果在整体上也能反映出基于 DSP 实现 KCF 算法的平均水平，若不采用多核 DSP，则很难达到 50 FPS 以上的处理速度。Cadence 公司的 Vision DSP 得益于它的超长指令字，实现 KCF 算法能够达到 164.8 FPS 的处理速度[33]。AM5708 是一款高性能应用处理器，其带有一个 Cortex-A15 内核，两个 Cortex-M4 内核，一个 C66x 内核，还有专门的视频编解码器和 GPU。但基于高层次的 Linux 上位机软件开发引入了许多额外的时间开销，使得其实际的实时性能并不理想。目前大多数实现方案采用的都是 C66x 内核，单个内核实现 KCF 算法能够达到 40 FPS 左右，利用多个内核实现能够明显提升算法的速度。采用 OpenMP 实现是一种较高抽象层次的实现方案，编译器能够完成串行算法的并行化，但效率较低，采用 8 倍的内核数量，仅能提高约两倍的处理速度。

目标跟踪算法在嵌入式平台上的实现主要关注算法的实时性，实时性指该算法的执行能够在确定的时间内完成。但是在多篇文献中给出的测试结果表明，KCF 算法执行的速度与跟踪的目标大小有关。比如王跃宗等人的实现方案只需要 10 ms 左右就可以完成 47×102 大小的目标的跟踪，而跟踪 87×290 大小的目标则需要 48 ms[32]。这一现象在实时性要求严苛的嵌入式系统中可能会带来隐患，但可以通过缩放实际跟踪目标的大小，并完全固定模板图像尺寸来解决这一问题。算法实时性的提升也可以从提高算法的运行速度来考虑。以往的 KCF 实现方案仅利用 DSP 内核实现

KCF 算法，还可以结合软硬件协同设计的思想，利用 DSP 的片上外设和系统中的 FPGA 对算法进行加速。另一方面，多核 DSP 的性能有待进一步发掘，KCF 算法没有大规模并行计算的环节，很难通过多个内核对算法加速。但多个内核能够完成更复杂的系统性的功能，建立红外激光融合的多核 DSP 软件框架，针对两个不同目标分别执行高实时性的 KCF 跟踪算法是本文重点解决的问题。

1.2.3 嵌入式平台跟踪算法评估

大多数嵌入式平台上的目标跟踪算法移植工作都缺少对移植后的算法性能的完整评估。归纳多篇文献的目标跟踪算法评估方案，主要有三种方式，一种是在上位机编写算法的软件模型，间接评估算法的跟踪性能，这一做法要求软件模型与嵌入式平台中的运行结果一致；一种是在嵌入式平台上通过摄像头采集实际的序列进行评估，由人主观评价跟踪性能的好坏；另外有少部分文献中提到采用上位机下发图像序列，直接评估嵌入式平台上算法的跟踪性能，但缺少完整全面的评估流程。

在上位机平台编写算法的软件模型，模拟嵌入式平台上的算法执行过程，可以间接评估算法的运行效果。比如 Danilowicz 等人采用 VOT2015 挑战赛的算法评估环境，通过软件模型来验证嵌入式平台上算法的跟踪性能[35]。但这种方式始终都是间接评估，无法保证该算法的软件模型与实际算法一致。Zhang 等人采用同样的一组跟踪序列，将嵌入式平台与上位机软件上的跟踪结果进行对比，其每一帧的跟踪位置偏差主要分布在 1~2.5 个像素范围内[25]。虽然这在一定程度上验证了算法功能移植的正确性，但仅通过一组序列的测试结果难以得到可靠的结论。

在搭载有摄像头的嵌入式平台上进行跟踪算法的实地测试是对整个系统功能的全面验证，观察实际的跟踪效果可以最直观地把握算法的跟踪性能。比如管宪宇在无人机上部署 KCF 目标跟踪算法，并在多个场景进行实地测试，对系统进行了全面的验证[36]。但这种做法也存在一些问题，比如现实场景中的目标缺少位置参考，目标跟踪结果的准确性全都依靠人的主观判断来评价；实际测试的序列并不能很好地覆盖到不同的情况，文献中展示的结果大多都只有少量序列，容易以偏概全。

许剑清采用 5 组实拍图像序列对嵌入式平台上运行的算法直接进行评估[29]。图像由上位机以 UDP 协议通过网络接口发送至 DSP，DSP 完成目标跟踪后向上位机传

回结果。张清洋在上位机将测试序列组织成相机数据的格式，并通过专门发送数据的板卡发送至嵌入式处理平台，对算法的实际跟踪效果进行验证[37]。这是最直接的算法评估方式，但这一做法完全可以更进一步，将算法验证的过程完全嵌入到公开数据集的验证平台中，使得嵌入式平台上跟踪算法的性能也能够像上位机软件算法一样快速而又全面地进行评估。

1.3 本文主要内容

本文提出并设计了一种基于国产 DSP 与 FPGA 的融合信息处理系统，用于红外激光融合的目标检测识别与跟踪算法研发。基于此系统建立起多核 DSP 软件框架，实现了高实时性的 KCF 目标跟踪算法，设计了算法硬件加速框架与算法验证平台，并对移植后的算法进行了全面的性能评估。主要内容共有 6 章，其组织结构关系如图 1-1 所示。

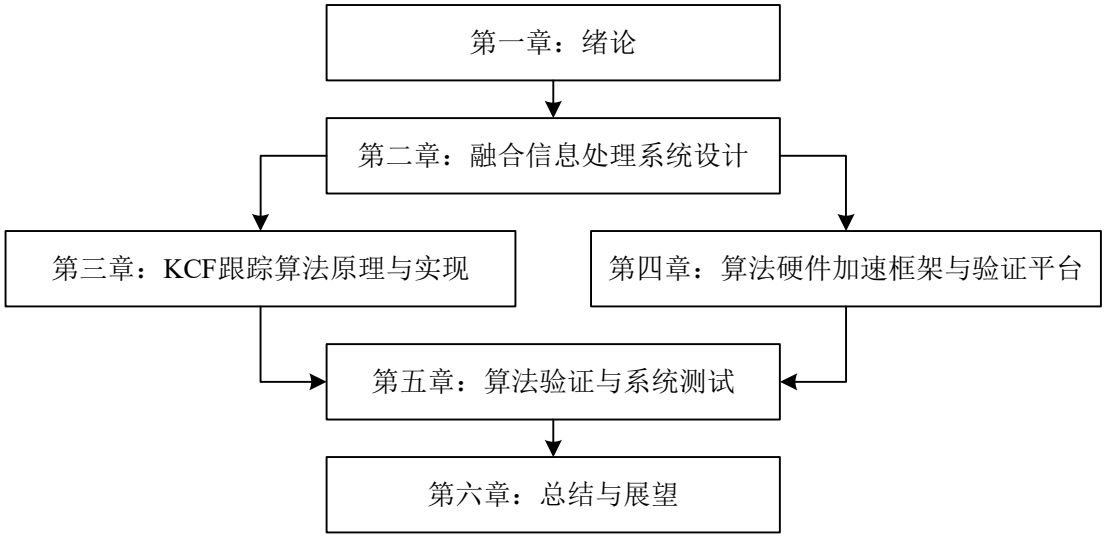


图 1-1 组织结构图

第一章 绪论：介绍了当前红外激光融合信息处理系统的研究现状，视觉目标跟踪算法的嵌入式实现以及评估方案。本文主要内容也将从红外激光融合信息处理系统的硬件设计与多核 DSP 软件框架设计，KCF 目标跟踪算法实时实现，目标跟踪算法的验证评估三部分展开。

第二章 融合信息处理系统设计：介绍了硬件系统的设计方案与多核 DSP 软件

框架的设计。根据整体的红外激光融合目标检测跟踪方案，规划了系统的功能布局，设计了由一片多核 DSP，一片专用协处理器与两片 FPGA 组成的高性能硬件系统。将具体的任务分配到 DSP 的各个内核，形成多核 DSP 软件框架，为后续的开发明确了方向。

第三章 KCF 跟踪算法原理与实现：详细分析了 KCF 算法的原理与实现细节，为实现算法移植奠定坚实的基础。结合银河飞腾 FT-M6678 DSP 的特点，充分利用片上外设，设计了批量二维 FFT 计算的快速实现方案，实现对任意尺度目标均能保持恒定处理速度的高实时性 KCF 目标跟踪算法。

第四章 算法硬件加速框架与验证平台：介绍了算法硬件加速框架设计与算法验证平台的搭建。实现了扩展性强，数据传输效率高，软件编程便捷的算法硬件加速框架。建立起基于公开数据集 UAV123 与 OTB100 的算法验证平台，用于全面地评估 DSP 上运行的目标跟踪算法。

第五章 算法验证与系统测试：基于算法验证平台对 DSP 执行的四种 KCF 目标跟踪算法进行验证与评估，同时也是对系统功能的完整测试。

第六章 总结与展望：总结了每一章的主要内容，给出了测试实验的结论，归结了本文主要的创新点。对于本设计中存在的不足点，提出了未来的改进方向。

2 融合信息处理系统设计

在传统的目标检测跟踪系统中，仅采用单一的视觉传感器获得的信息有局限性。该传感器一旦受到干扰，目标检测跟踪的准确率就会明显降低。为此，将不同的或者相同的多个传感器进行有效结合来提高系统的抗干扰能力和稳定性是一种解决问题的思路。采用制冷式红外探测器和可三维成像的 SPAD 激光雷达针对远距离目标进行检测跟踪有着广泛的应用前景。探测器获取的红外图像一般为 8~14 μm 的远红外波段信息，该波段具有穿透烟雾的能力，并且不论昼夜均可工作；SPAD 激光雷达的单光子检测能力灵敏度高，但探测视场小。将视场中目标的红外图像与其距离信息相结合，可以实现更加稳定可靠的跟踪效果。

根据设计需求与红外激光融合目标检测跟踪算法的实现方案，需要设计相应的融合信息处理系统。系统由一片高性能国产多核 DSP，一片课题组自研的专用协处理器与两片资源丰富的 FPGA 组成。除了能够满足实现算法的需求外，系统还能够通过 PCIe 链路与上位机连接，由上位机下发测试数据，对算法进行快速验证与评估。

融合信息处理系统中的多核 DSP 软件的设计也是重要的组成部分。红外激光融合的目标检测识别跟踪算法的实现需要充分发挥多核 DSP 的优势，使多个 DSP 内核高效协同工作。本章对多核 DSP 软件设计的一般方法以及融合信息处理系统中的多核 DSP 软件框架进行了介绍，为后续的软件开发规划了方向。

2.1 设计需求与初步方案

上位机能够向融合信息处理系统每秒发送 50 帧 640×512×8 bit 红外图像数据和 20000 帧 128×128×12 bit 激光成像数据，总的有效数据传输带宽需要达到 484 MB/s 以上。整个系统能够对收到的数据进行实时处理，完成对复杂环境下目标的准确识别与跟踪，并将目标检测与跟踪结果传回上位机或者由显示器显示。系统还应具有良好的环境适应性与可靠性，其中的器件尽可能采用国产器件。

针对这些设计需求，初步拟定采用红外与激光决策级融合的方案，并结合辅助目标间接定位技术，实现稳定可靠的目标检测跟踪，整体流程图如图 2-1 所示。

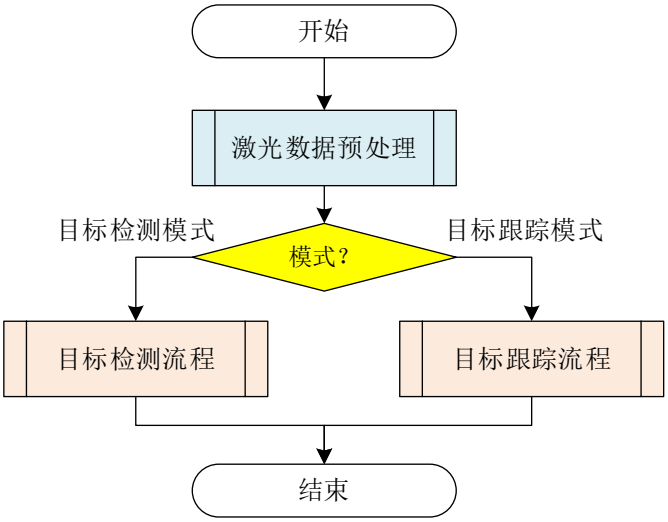


图 2-1 红外激光融合目标检测跟踪流程图

这一流程图表示系统在某一时刻收到红外实时图像后需要完成的计算任务，收到新的一帧红外实时图像后需要再次执行这一流程。整体流程主要由目标检测流程与目标跟踪流程两部分组成，系统启动之初处于目标检测模式，根据检测与跟踪结果的置信度来切换模式。当目标的置信度较低时，系统处于目标检测模式，直到得到较高置信度的结果才能切换到目标跟踪模式；在目标跟踪过程中，如果目标的置信度降低到一定的阈值以下，系统就切换到目标检测模式，重新检测目标。不论是目标检测或是跟踪模式，首先都需要对激光数据进行预处理，提升激光成像效果并且使红外与激光数据同步。辅助目标检测跟踪流程仅针对红外实时图像进行处理，是一个相对独立的流程。在目标跟踪流程中，可以借助辅助目标位置间接定位真实目标的位置。

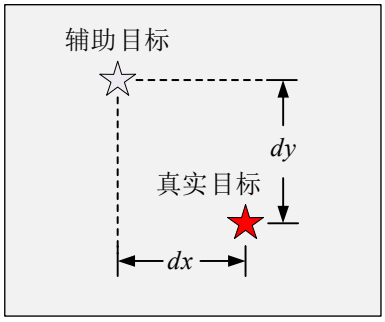


图 2-2 辅助目标间接定位

在红外实时图像中，真实目标附近可能存在特征明显且易于跟踪的图像块适合作为辅助目标。如图 2-2 所示，在连续帧之间，真实目标与辅助目标之间的相对位

置关系基本保持不变，从而可以通过跟踪辅助目标位置，再结合两者之间的相对位置关系来间接定位真实目标的位置。

2.1.1 激光数据预处理

激光雷达探测器产生的原始数据是一帧帧由时间计数值组成的二维图像。每个像素值都是激光飞行时间的计数值，反映了视场中不同位置与探测器的距离，因此称原始图像为距离像。原始距离像中存在许多噪声，需要采取降噪、连续帧累积的方式进行数据增强。连续多帧累积过程中，同一像素位置的距离值会有不同，选取出现次数最多的距离值作为该位置处的距离值，出现的次数作为强度值。由强度值可以构成强度像，反映距离值的置信度。原始激光数据量大，一般嵌入式处理器无法实时完成多帧距离像的累积成像，需要依靠并行电路加速。李玉涛基于 FPGA 设计了激光距离像强度像生成电路，可以通过硬件电路快速实现激光原始数据的预处理，得到多帧累积后的距离像与强度像[38]，其过程如图 2-3 所示。

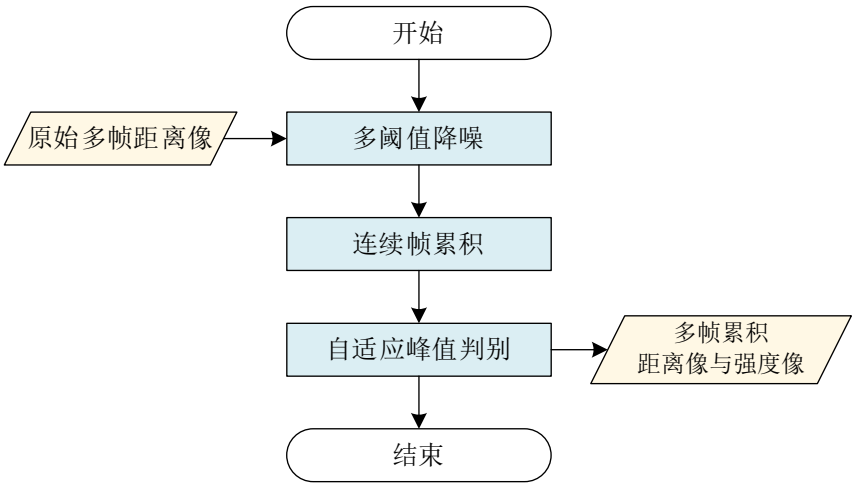


图 2-3 激光数据预处理流程图

2.1.2 目标检测流程

目标检测流程如图 2-4 所示。利用红外与激光数据分别检测目标，融合两者结果得出高置信度的目标位置。流程图左侧为基于红外实时图像的红外目标检测流程，右侧为基于激光点云数据的激光目标检测流程。

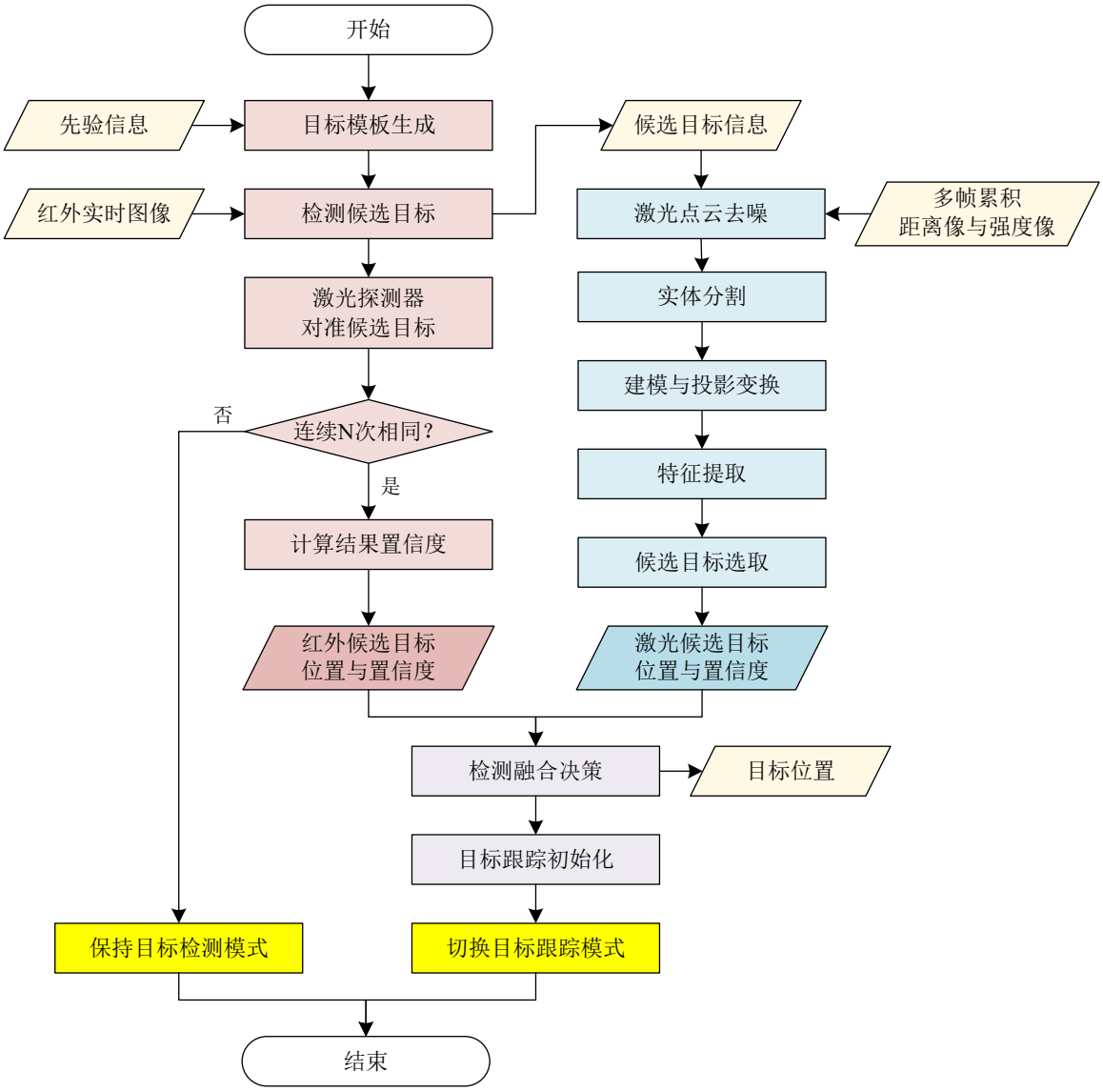


图 2-4 目标检测流程图

整个红外激光融合信息处理系统未来计划搭载在飞行器上使用，针对不同的地面目标首先需要生成红外图像的目标模板。模板图像源于卫星像片和其它一些先验信息，但由于卫星与飞行器观察地面目标的视角不同，因此需要根据飞行器的姿态与位置对卫星像片进行变换从而生成目标模板图。目标检测识别由神经网络实现，基于神经网络的目标检测识别技术具有稳定性高，抗干扰能力强，适用范围广等特点，是目前主流的目标检测识别技术。在获取可能的候选目标位置后，由于激光探测器的成像范围一般小于红外探测器，因此需要调整探测器的朝向，使探测器正对候选目标位置进行激光三维成像。经过多帧累积得到的距离像可以结合探测器角度与空间坐标

得到原始激光点云数据。原始激光点云数据可以结合激光强度像进行去噪。对激光点云进行实体分割可以进一步丰富探测器获取的原始信息。在激光目标检测流程中需要再次将预处理后的激光点云数据投影到探测器成像视角的方向上，经过投影后的激光点云数据可以采用类似于一般二维图像的处理方式提取特征图，进而完成激光候选目标的匹配检测。

在红外实时图像中连续 N 帧检测识别到相同目标后，给出目标相应的置信度，同时结合激光候选目标的位置与置信度进一步确认目标位置，完成目标检测流程。接着可以根据目标检测结果初始化后续目标跟踪所用的跟踪器，利用模板图像完成跟踪器训练。将系统的工作模式切换为目标跟踪模式，在随后的新一帧图像到来的时候执行目标跟踪流程。

2.1.3 目标跟踪流程

目标跟踪流程如图 2-5 所示，类似于目标检测流程，目标跟踪流程同样采用决策级融合的方式。分别利用红外实时图像与激光点云数据跟踪目标位置，得到红外/激光候选目标的位置与置信度。与目标检测流程不同的是，目标跟踪流程另外引入辅助目标间接定位机制，由一个相对独立的辅助目标检测跟踪流程提供辅助目标位置与置信度。在未找到辅助目标或者辅助目标丢失的情况下，其相应的置信度较小，对跟踪结果不产生影响。目标跟踪的最终结果由红外候选目标、激光候选目标以及辅助目标的位置与置信度一同决定。在目标能够正确跟踪的情况下，需要在确定目标位置后更新目标模板；若目标跟踪失败，则需要重新回到目标检测流程。

基于红外实时图像的目标跟踪主要采用多尺度相关滤波目标跟踪算法，首先从红外实时图像中提取目标的特征图。特征图相比于原图能更加准确地把握目标的固有信息，而忽略一些不重要的干扰信息。从激光成像结果中也可以提取激光特征图，融合红外与激光特征图后采用多尺度的相关滤波算法求解目标的位置与置信度即完成红外实时图像中的目标跟踪流程。基于激光点云数据的目标跟踪流程与检测流程类似，激光点云数据经过去噪与实体分割后，先投影到探测器成像视角的方向上得到二维图像，再从中提取激光特征图，利用模板匹配的方法获得激光候选目标位置与置信度。

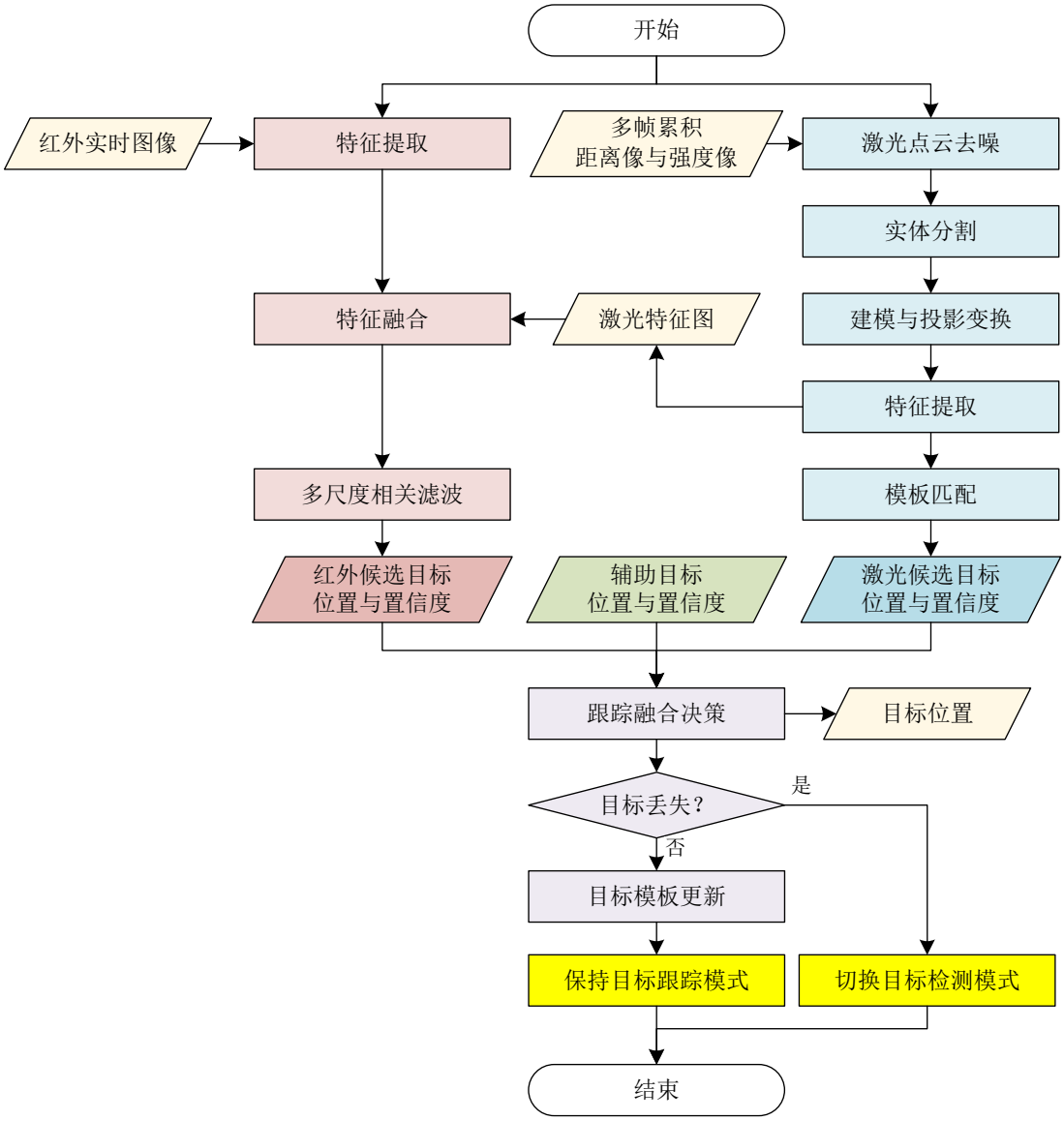


图 2-5 目标跟踪流程图

辅助目标检测跟踪流程如图 2-6 所示，这一过程包括检测与跟踪两种模式，仅涉及对红外实时图像的处理。系统启动之初，首先处于辅助目标检测模式。辅助目标检测流程采用涂直健等人提出的一种自动从图像中选取辅助目标的算法。这一算法能够自动选取红外图像中形状稳定，灰度显著的区域作为辅助目标[39]。采用形态学运算与图像分割算法即可获取一系列可以作为辅助目标的候选区域。再进一步从中选取特征最明显且与真实目标相对距离合适的区域作为最终的辅助目标，即完成辅助目标的检测。

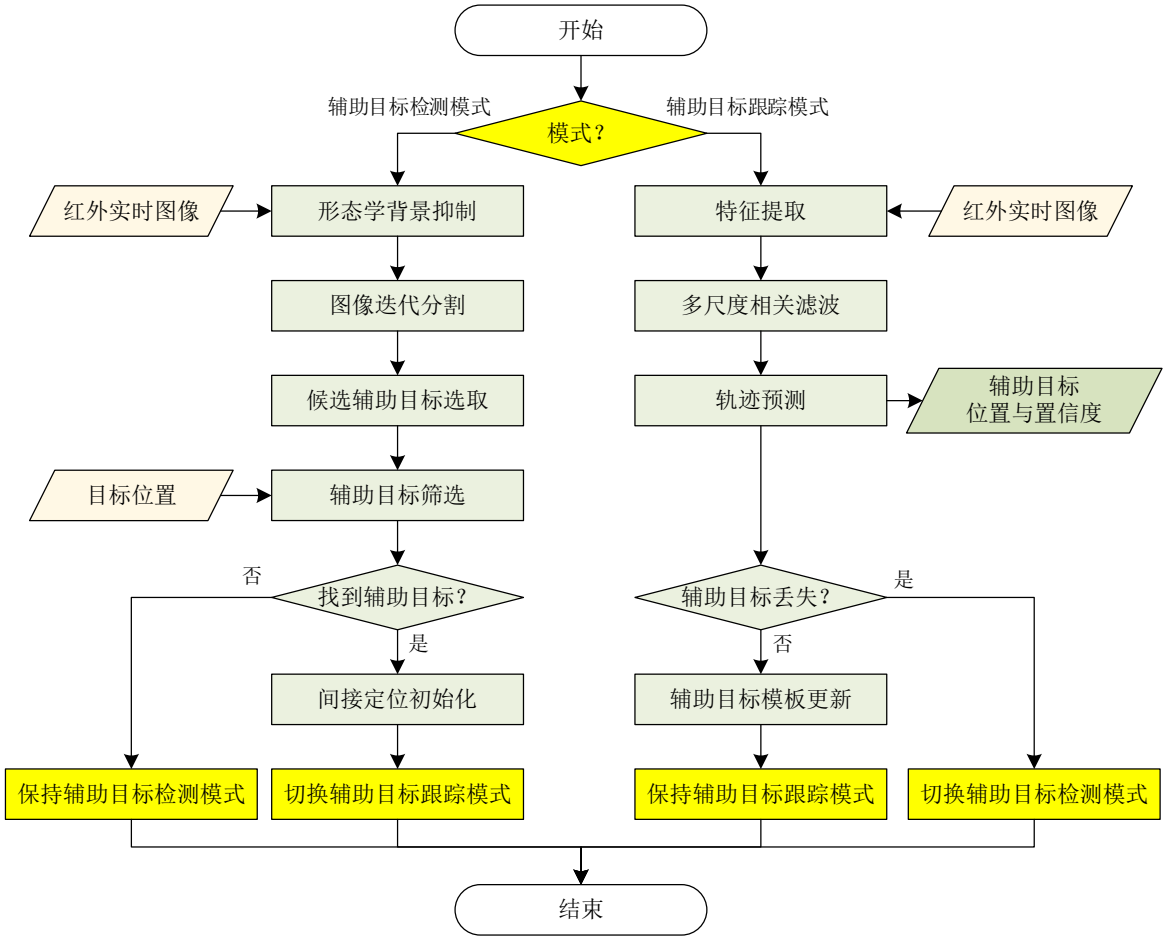


图 2-6 辅助目标检测跟踪流程图

辅助目标跟踪流程与红外目标的跟踪流程类似，同样采用相关滤波算法完成对目标的跟踪，只是少了融合激光特征图的过程。辅助目标跟踪的结果还可以结合轨迹预测算法，避免辅助目标跟踪结果突变，提升跟踪结果的稳定性。

2.2 硬件系统设计

2.2.1 系统结构与功能布局

融合信息处理板以国产高性能 DSP 银河飞腾 FT-M66778 为核心进行设计，采用一片专用协处理器与两片资源丰富的 FPGA 对算法中的部分计算过程加速。银河飞腾 FT-M66778 的 8 颗 FT-M66x DSP 内核，最高均支持 1.25 GHz 的主时钟频率，指令字长度为 128 bit，拥有 32GMACS@1GHz 的定点运算能力与 16GFLOPS@1GHz 的浮点运算能力。FPGA 采用的是国产的 JFM7VX690T，它拥有丰富的逻辑资源，属于

高性能 FPGA。课题组此前设计了一款专用协处理器——图像处理协处理单元（Image Processing Coprocessing Unit, IPCU），它能够快速完成多种形态学运算，为红外实时图像中的辅助目标检测提供硬件加速。

融合信息处理板的设计框图如图 2-7 所示。IPCU 主要用于辅助目标检测；通过 PCIe 连接上位机的 FPGA JFM7VX690T-2（以下简称 690T-2）主要用于实现激光成像电路；另有一片 FPGA JFM7VX690T-1（以下简称 690T-1）主要用于实现基于神经网络的目标检测识别电路；银河飞腾 FT-M66778 负责整体的调度工作，需要完成红外/辅助目标的跟踪，激光目标检测跟踪以及融合决策。

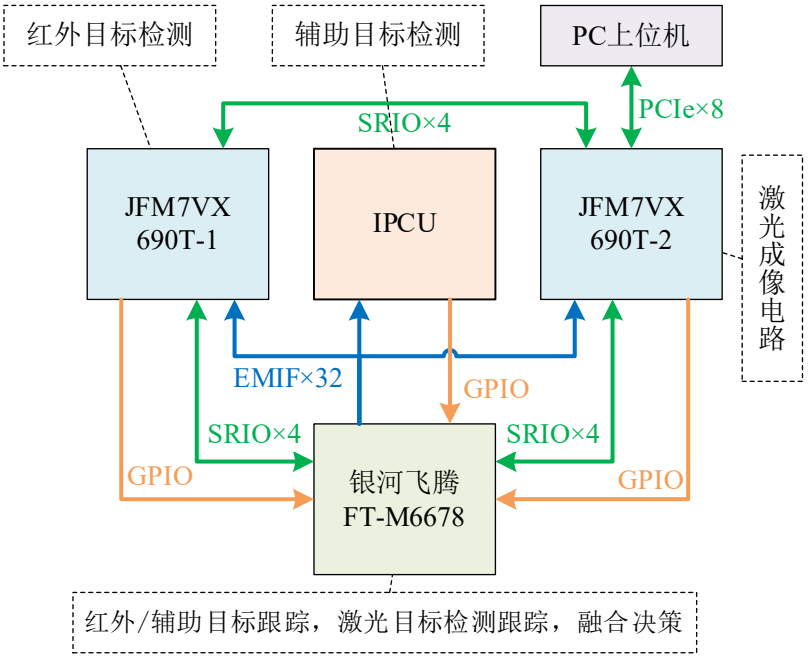


图 2-7 融合信息处理板设计框图

上位机与系统之间的 PCIe 链路便于上位机下发用于仿真的激光原始数据；IPCU 采用 EMIF 总线与 DSP 进行数据交互；DSP 与两片 FPGA 之间也都有 EMIF 总线连接，但主要用于控制命令传输，它们之间的数据传输通过高速链路实现。DSP 的 GPIO 用于接收来自 IPCU 或 FPGA 的中断信号。两片 FPGA 之间也有 Serial RapidIO（SRIO）链路互连，便于 690T-2 将收到的红外图像发送至 690T-1 进行处理。图中的 SRIO 链路速率均为 3.125 Gbps，PCIe 链路速率均为 5.0 Gbps。

实际的融合信息处理板与面向未来应用的设计略有不同，实际框图与实物图如

图 2-8 和图 2-9 所示。

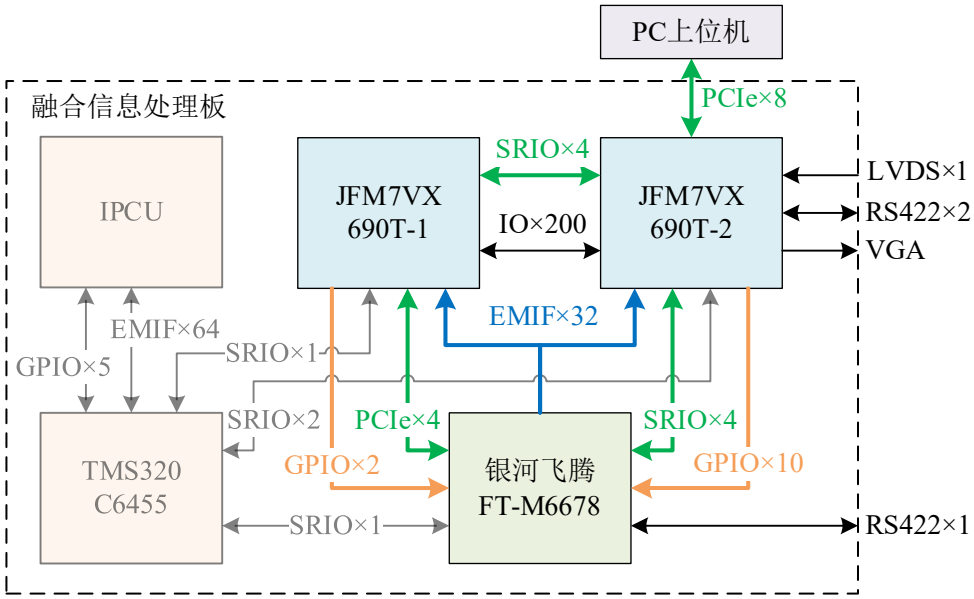


图 2-8 实际的融合信息处理板框图

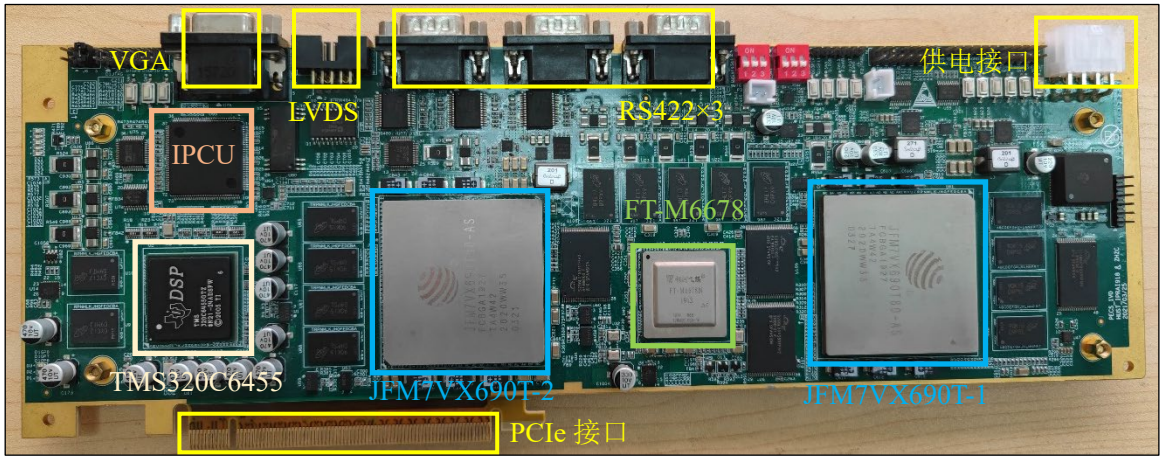


图 2-9 融合信息处理板实物图

主要考虑到课题组工作的延续性，在具体设计阶段仍然保留了上一版本设计中用到的 DSP——TMS320C6455，并将 IPCU 挂接在 C6455 上。之所以这么做是因为当时 IPCU 还没有完成流片，其功能还未经过测试。原理图的设计为了尽可能与前一版本兼容，保留了许多原有的设计。除了 C6455 与 IPCU 以外，LVDS 接口，RS422 串口以及 VGA 显示接口也都沿用了以往的设计。但系统的主体部分是一致的，同样是以银河飞腾 FT-M6678 DSP 与两片 JFM7VX690T FPGA 为核心器件。为了便于调试，系统的供电没有直接采用 PCIe 接口供电，而是类似于大功率的独立显卡供电的

方式，需要采用外部供电。电路设计细节详见附录 A.3 。

2.2.2 系统供电

DSP 与 FPGA 需要多种不同类型的供电，比如内核供电，端口的 IO 供电，内部独立模块的供电等等。不同类型的供电需要按照要求先后产生。

TI 公司的 LM3880 系列电源时序控制芯片能够根据输入的单个使能信号 EN，输出有先后次序的用于控制电源输出的控制信号，LM3880-1 的时序图如图 2-10 所示。其中的延时 t_d 由具体型号确定，比如 LM3880-1AE 的延时是 2 ms，LM3880-1AB 的延时是 30 ms[40]。

为了满足供电顺序的需求，采用三片 LM3880 芯片实现供电顺序的控制，系统启动过程中控制信号的时序图如图 2-11 所示。C6455 的内核供电需要至少晚于其 IO 供电 0.5 ms；FT-M6678 的内核供电需要至少晚于一般 IO 供电 20 ms，串行解串器相关的供电需要至少晚于 DDR3 的 IO 供电 10 ms；690T FPGA 只要求内部供电先于 IO 供电，并给出了推荐的供电顺序。

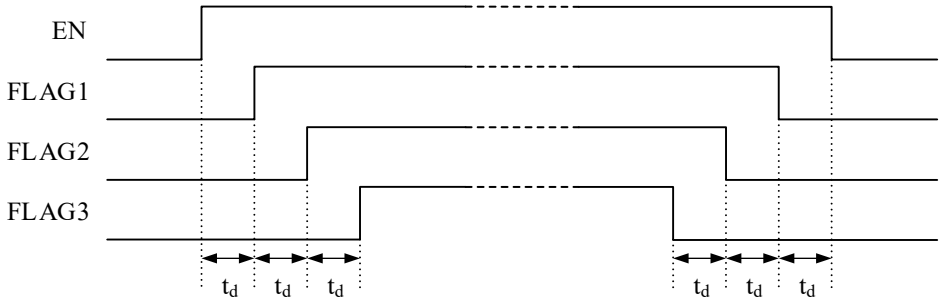


图 2-10 LM3880-1 电源时序控制芯片时序图

系统利用 LMZ31530RLG 电源模块将外部主机电源提供的 12V 电源转换为 3.3V 的非受控电源，系统中其余供电大多由该电源变换得到。TMS320C6455 和 690T 的供电顺序可以分别由一片 LM3880-1AE 来实现，FT-M6678 的供电顺序所要求的间隔较大，因此需要采用 LM3880-1AB 实现。系统中所有关于 DDR3 存储器的供电都由同一个电源模块产生，该电源模块由管理 FT-M6678 供电顺序的芯片 LM3880-1AB 的 Flag2 控制。而 FPGA 要求 IO 的供电晚于内部供电，因此 FPGA 也需要在 DDR3 存储器完成供电的时刻附近启动，采用 LM3880-1AB 的 Flag2 作为 FPGA 的供电顺序管理芯片的使能信号。为了在 DDR3 存储器启动之前完成 FPGA 的内部供电，负

责给 DDR3 存储器供电的电源模块采用缓启动的方式，缓启动过程持续 11 ms。这一做法虽然延迟了 DDR3 的 IO 供电，但也能满足 FT-M6678 的 DDR3 的 IO 供电至少早于其串行解串器供电 10 ms 的需求。

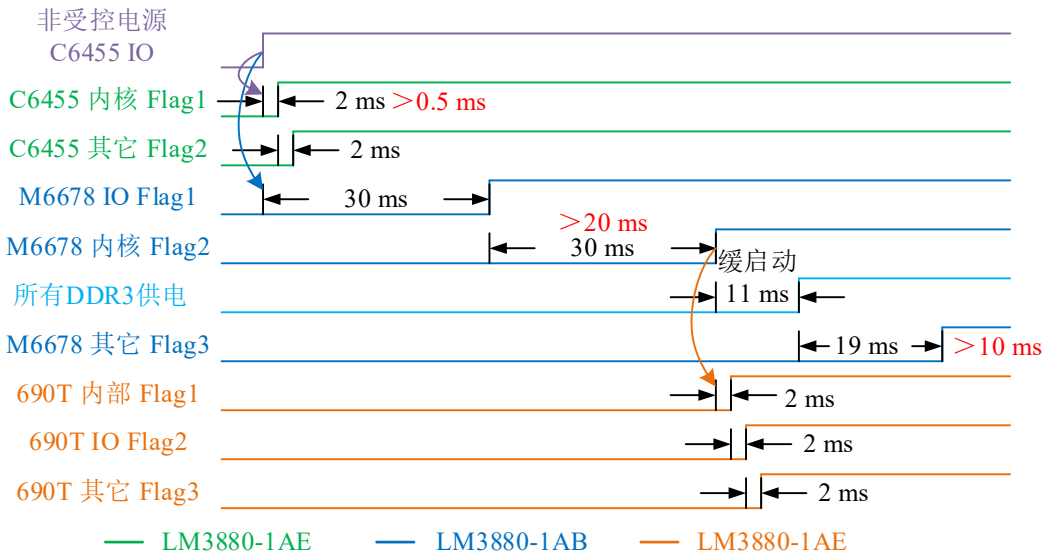


图 2-11 系统供电顺序时序控制图

2.2.3 板上存储器

融合信息处理板的板上存储器分布如图 2-12 所示。DSP 与 FPGA 都接有 DDR3 DRAM，每一片 DRAM 芯片都是 16 bit 位宽，采用位扩展的方式用 4 片 DRAM 芯片扩展为 64 bit 位宽。DSP 外接总计容量 2 GB 的 DRAM；两片 FPGA 分别外接总计容量 1 GB 的 DRAM。这些 DRAM 用于在系统运行时存放临时数据。DSP 的程序与 FPGA 的比特流文件都需要保存在 Flash 中，用于系统启动时的加载与配置。

DSP 的 8 个内核运行所需的代码和初始化数据都存放在通过 EMIF 外接的 NOR Flash 中，系统启动时 DSP 会自动从 NOR Flash 加载程序；配置两片 FPGA 所需的比特流文件存放在通过字节外设接口（Byte Peripheral Interface, BPI）外接的 128 MB NOR Flash 中，系统启动时 FPGA 会自动读取 Flash 内的数据并对自身进行配置。考虑到目标检测识别神经网络硬件电路的部署可能需要保存许多参数，690T-1 还外接了 128 MB 的 NAND Flash 用于保存量化后的神经网络参数；DSP 外接的 128 MB NAND Flash 可以用于保存一些目标跟踪过程中需要用到的先验信息或者一些在系统

断电前需要记录的测试结果。

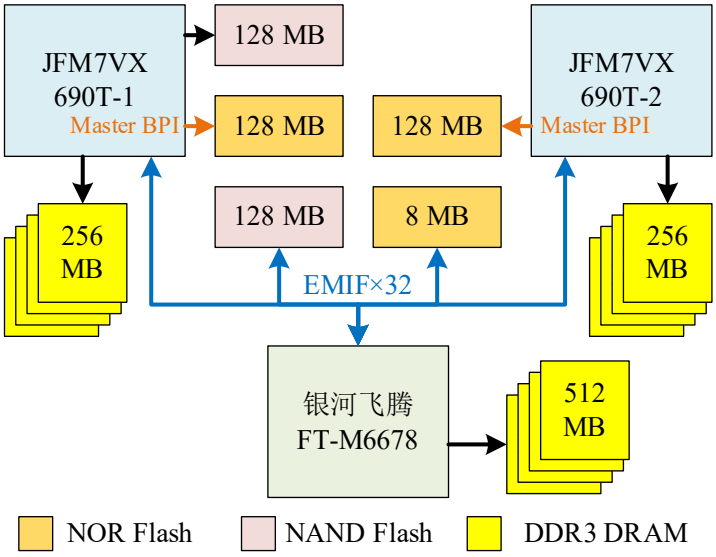


图 2-12 融合信息处理板板上存储器

2.3 多核 DSP 软件设计

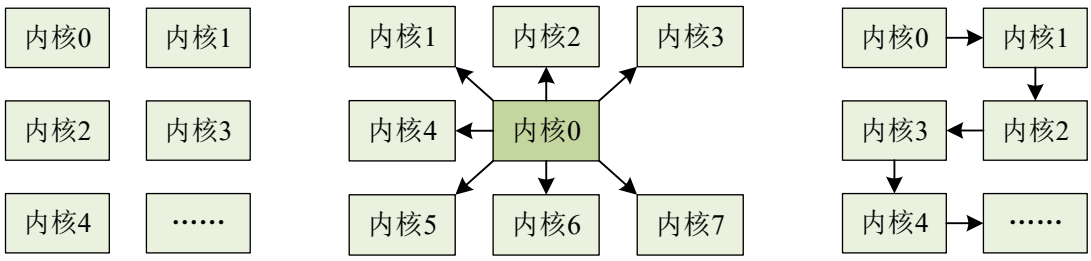
多核 DSP 软件程序设计的难点主要可以总结为三个方面：同步，互斥与优化。不同内核之间协同工作离不开核间的消息同步与数据共享，程序设计人员需要熟悉不同的核间同步机制并善加利用；多核 DSP 相对于单核 DSP 会新增部分共享外设，实现不同内核对共享资源的互斥访问也是程序设计人员需要注意的问题；软件程序的优化也不再局限于单个内核后，程序设计人员不仅需要熟悉单个内核的软件优化技术，而且需要对每个内核的计算任务进行合理分配，最大化软件执行的并行度从而提升效率。针对以上难点，掌握常用的软件开发技术工具，熟悉多核 DSP 的片内外设是解决难点的主要途径。本文根据软件开发流程，依次就软件结构、存储空间分配、现有软件包的支持、软件优化策略以及软件程序固化五个方面对多核 DSP 软件设计的一般方法进行介绍。最后为红外激光融合的目标检测跟踪系统设计了多核 DSP 软件框架，为后续的软件开发规划了方向。

2.3.1 常用软件结构

多核 DSP 的软件结构可以按照是否使用 OpenMP（Open Multi-Processing）分为两类。OpenMP 是面向并行系统的一套编译框架，它能够使用户在对代码做最小的改

动的情况下，将串行程序转换为多线程的并行程序。主要的做法就是使用#pragma 预处理指令，将循环结构中的计算任务分发到多个内核执行。

循环结构中用到的变量有的是多核共享的，称为共享变量，需要用 shared 关键字声明；有的是每个核私有的，称为私有变量，需要用 firstprivate 或者 private 关键字声明。每个核都有自己私有的存储空间，使用 firstprivate 声明的变量相当于每个核都会执行一次拷贝构造函数，使用 private 声明的变量相当于每个核都会调用一次默认构造函数，使用 shared 声明的变量则由每个核共享访问，不会有新的变量生成。目前 DSP 的编译器仅支持 OpenMP3.0 规范和部分 OpenMP4.0 规范，实际使用中存在一些限制。用于循环结构中的变量不能是结构体成员或者类对象的成员，也不能调用某个类的成员函数。这对于频繁使用类对象的代码不是很友好，而且实际测试发现 OpenMP 对算法的并行加速能力也比较有限，想要在 DSP 上充分利用 OpenMP 提升算法性能还需要克服许多技术难题。



(a) 独立式　　(b) 主从式　　(c) 流水式

图 2-13 不同的多核软件结构

使用 OpenMP 固然能很快实现串行算法的并行化，但无法对算法做到细粒度的把握，进一步优化难度大。因此大部分情况下需要为每个核单独设计程序，由用户管理多核之间的同步与互斥。主要的软件设计结构如图 2-13 所示。独立式软件结构即不同内核之间相互独立，工作互不相关，这是最简单的结构，每个内核的程序都如同单核程序设计一样。主从式软件结构由一个内核负责整体工作的调度，其余内核分别负责不同的具体任务。这一结构不适用于任务间依赖性较强的情况，因为容易出现“一核工作，众核围观”的现象，使得系统的效率不高。流水式软件结构可以将复杂的计算过程拆分为多个环节，由每个核分别负责流水线中的一环。这一做法需要保证每一级流水线的工作量均匀，否则容易出现流水线卡顿的现象，无法起到加速效果。

2.3.2 存储空间分配

在确定每个核的具体任务后，需要将系统内有限的存储资源按需分配给不同的内核。DSP 内的存储资源采用分层的结构，如表 2-1 所示。L1P 和 L1D 的容量很小，虽然它们可以配置为 SRAM 或者 Cache，但是一般都作为 Cache 使用，分别缓存指令和数据。片内的多核共享存储器控制器（Multicore Shared Memory Controller, MSMC）负责管理片上 4 MB 多核共享的静态随机访问存储器（Static Random Access Memory, SRAM），内核对这部分存储器的访问速度与 L2 缓存相当，仅次于 L1 缓存，属于容量较小的高速存储器。片外的 DDR3 存储器也是由多核共享，但它是动态随机访问存储器（Dynamic Random Access Memory, DRAM），容量较大但访问速度较慢。L2 作为 Cache 部分的大小取决于 DDR3 DRAM 的使用情况，如果几乎没有使用 DRAM，那么 L2 都可以配置成 SRAM；反之，如果 DRAM 使用较频繁，L2 也需要相应地增大 Cache 的占比。

表 2-1 FT-M6678 片内片外存储资源

名称	大小	类别	说明
L1D	32 kB	私有	一般作为 Cache 使用
L1P	32 kB	私有	一般作为 Cache 使用
L2	512 kB	私有	可以部分配置为 Cache 或 SRAM，访问经过 L1 缓存
MSMC SRAM	4 MB	共享	由 MSMC 管理的 SRAM，访问经过 L1 缓存
DDR3 DRAM	2 GB	共享	容量最大但访问速度最慢，需要 L2 缓存加速

在实际开发过程中，软件工程需要对处理器的硬件平台（platform）进行配置，其配置界面如图 2-14 所示。平台的配置主要包括处理器型号，系统主时钟频率与不同存储空间的使用方式。比如图中配置将 L1 都用作 Cache，L2 中的 256 kB 用作 Cache，剩余 256 kB 用作 SRAM。在 MSMC SRAM 与 DDR3 DRAM 中分别划分了 256 kB 和 128 MB 用作私有空间，同时还有 512 kB 的共享 SRAM 和 1 GB 的共享 DRAM。而后将 L2 SRAM 用作代码区和栈区，私有的 MSMC SRAM 用作数据区。

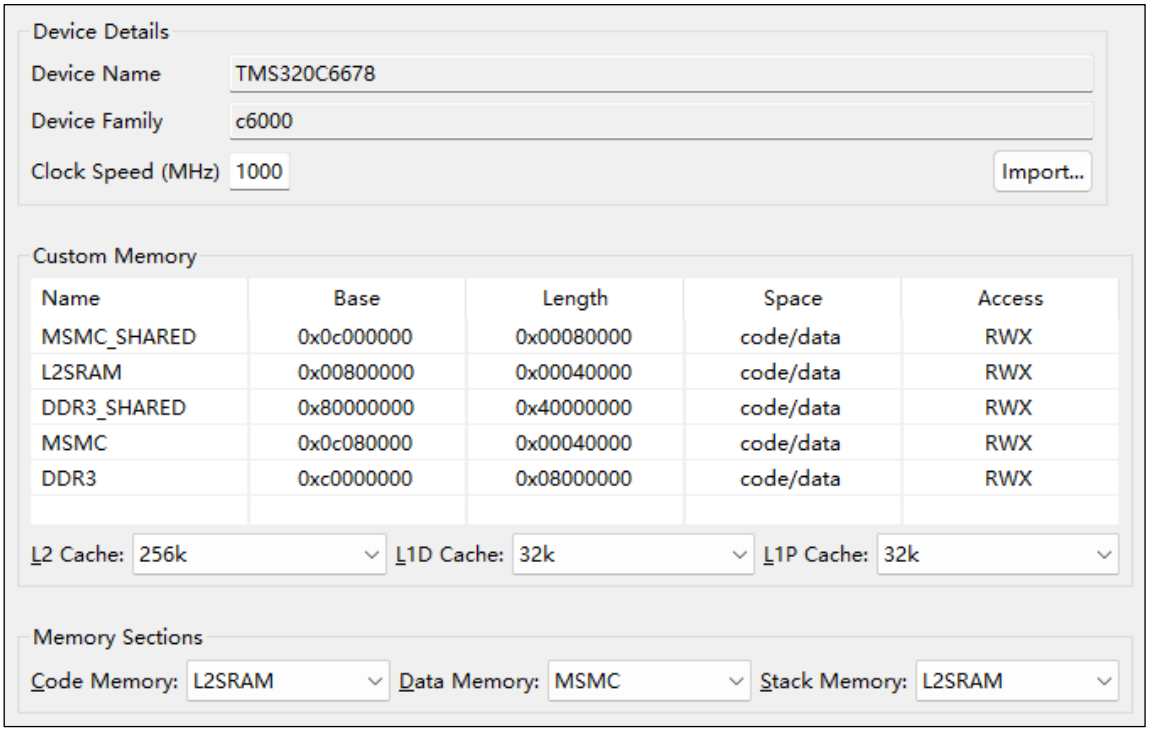


图 2-14 软件工程处理器硬件平台配置界面

2.3.3 RTSC 与 XDCTools

系统中使用的虽然是国产的银河飞腾 FT-M6678 DSP，但是对于 DSP 软件的开发仍然依赖于 TI 的 CCS 开发环境。在这一开发环境下，实时软件组件（Real-Time Software Components, RTSC）体系框架能够为开发过程提供很大的便利。RTSC 是在 2008 年发起的一个基于 C 语言的项目，为的是能够用管理软件包的方式管理 C 源代码，方便 C 语言开发者直接导入已有的组件进行开发，提高开发效率。在 RTSC 体系框架下，用户只需要通过编写脚本文件就可以调用已有的软件包，具体的开发流程如图 2-15 所示。

扩展 C 语言工具（eXpanDed C Tools, XDCTools）主要用于解析用户编写的配置脚本，并生成相应软件包的源文件与头文件。这部分由配置脚本生成的代码经过编译生成相应的库文件，得到的库文件可以与用户编译的目标文件链接在一起生成最终的输出文件。在 RTSC 体系框架下，SYS/BIOS 实时内核与用于核间通信的 IPC（Inter-Processor Communication）是常用的两个软件包，它们能够为用户开展新的设计提供极大的便利。

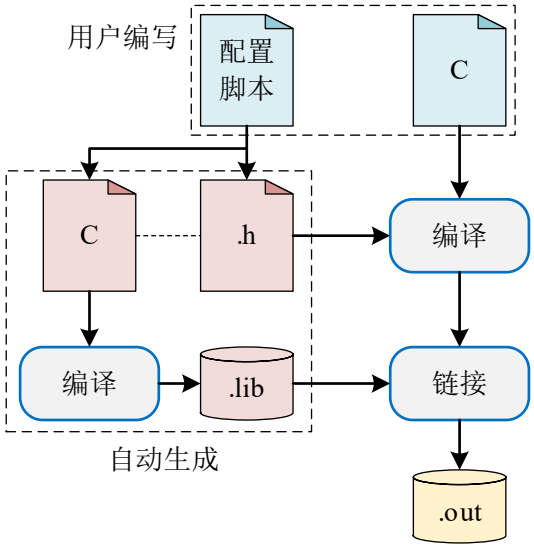


图 2-15 RTSC 框架下的开发流程

1) SYS/BIOS 实时内核

SYS/BIOS 实时内核提供了一个多线程的软件开发框架，不同的线程根据优先级从高到低依次分为硬件中断、软件中断和任务[41]。不同线程之间有着多种通信机制，比如信号量（Semaphore），事件（Event），信箱（Mailbox）等等。实时内核主要负责实现线程的调度，高优先级的线程能够抢断正在运行的低优先级的线程；正在运行的线程若缺少某种资源可以进入挂起状态，处理器转而去执行其它就绪的线程。SYS/BIOS 实时内核的引入能够使处理器更加充分地利用有限的计算资源，提高 DSP 软件的执行效率。

2) IPC 核间通信

多核 DSP 的核间通信一般通过 IPC 软件包来实现。不同核之间的信息同步与数据传输依赖于共享存储器，为了减少核间通信的时间开销，一般将用于核间通信的共享存储区安排在 MSMC SRAM 中。在现有的开发环境下，主要的核间通信机制包括消息队列（Message Queue），通知（Notify）和用于实现多核间互斥操作的 GateMP。

消息队列可以用于核间批量数据的传输。共享存储区中可以建立一个堆区并由多个内核同时管理。一般在单核程序设计时，需要特别注意堆区空间的分配与释放，若没有及时地释放动态分配的空间，那么就有可能产生内存泄漏的问题。而在通过消息队列传递数据时，由内核 A 动态分配的空间可以由另一个内核 B 来释放，消息队

列数据传递的实质只是将共享堆区中的某个指针传送给接收方，并由接收方来管理该指针指向的一片内存区域。

Notify 仅用于在内核间发送简短的信息，每个内核支持 28 种不同的 Notify 事件编号，在发送方发送 Notify 之前，接收方需要首先完成对应事件号的注册。每次发送的 Notify 可以额外携带 32bit 的信息。

当某些数据或硬件资源由多个内核共享时，GateMP 能够保证每个内核独占式地访问它们，保证了数据的一致性与硬件资源的正确使用。这些数据或硬件资源就像是被一扇门保护起来，这扇门每次只能有一个人进入，只有门里的人才能对数据进行读写操作或使用硬件资源，一个人离开后，下一个人才能接着使用。

消息队列，Notify 与 GateMP 已经能够满足大部分的核间通信的需求了。不同内核之间的同步与互斥可以利用 Notify 与 GateMP 实现，批量的数据传输可以由消息队列实现。

2.3.4 软件优化策略

想要充分发挥 DSP 的计算性能，软件层面的优化也是必不可少的。某些专门针对 DSP 进行优化的运算库，比如 mathlib 和 dsplib 都是用编译器提供的内联函数对特定的计算进行了优化。例如求一个单精度浮点数的倒数。若直接采用除法运算符“/”，那么时间开销大约需要几十个周期；而采用 DSP 的内联函数 RCPSP 则只需要一个周期。

编译器提供的内联函数直接与特定的 DSP 汇编指令相对应，熟悉 DSP 的指令集并适时地采用合适的内联函数是一个主要的优化思路。除此之外，还可以通过软件流水线技术进行优化。复杂的算法大多都包含许多循环结构，利用软件流水线技术减少循环迭代间隔（Iteration Interval）时间也能起到显著的加速效果[42]。若循环次数为 N，则整个循环的总时间为：

$$t_{\text{total}} = (N-1) \times t_{\text{ii}} + t_{\text{head}} + t_{\text{tail}} \tag{2-1}$$

其中 t_{head} 表示流水线刚开始填充需要的时间，t_{tail} 表示最后排空流水线需要的时间。减小 t_{ii} 是减少循环结构总时间的主要方式。

软件流水线的具体实现需要由编译器完成，用户只需要尽可能多地为编译器提供有用的信息来帮助它寻找最优的优化策略。随着指令集规模的扩展，指令并行程度的提升，以及软件流水线技术的引入，想要通过编写汇编代码来优化软件变得越来越困难，取而代之的是由编译器完成软件优化的工作，而用户则需要利用一些限定词或预编译指令来给编译器提供更多的信息。

1) 循环内部依赖

若在某一个循环结构中，首先利用指针 p_1 从存储器读取数据，最后利用指针 p_2 向存储器写回数据，两个指针分别访问两块不同的存储空间。在没有显式声明的情况下，编译器并不清楚它们访问的是否是同一块存储空间。所以为了保险起见，编译器采取保守的策略。每次循环都会等待数据写回 p_2 所指向的地址，然后再读取指针 p_1 指向的数据。一般一条加载数据的 LOAD 指令需要 5 个周期，一条存储数据的 STORE 指令需要 3 个周期[43]。这样一来 DSP 就会有很多时间在等待数据加载和写回，运算效率低下。

对于 p_1 和 p_2 两个指向不同数据空间的指针，可以在声明的时候加上“restrict”关键字进行修饰，告知编译器它们访问的是不同的数据空间，解除循环内部的依赖关系。这样一来 DSP 就可以利用 p_1 指针连续地取数据进行运算，也可以连续地将结果写回 p_2 指针指向的数据空间。在必要的地方添加“restrict”关键字能够有效地减小 t_{ii}，这一方法是最主要的软件流水线优化方式。

2) 循环展开

DSP 内有多个功能单元，这使得 DSP 可以在一个周期内同时执行多条指令。若某一次循环只使用了一半的功能单元，而另一半是空闲的，那么就可以考虑采用循环展开的方式进行优化。将连续的两次循环作为一次迭代，充分利用所有功能单元。但这么做就对循环次数 N 有一定的限制，N 必须是 2 的整数倍。

如果一个循环结构的循环次数是固定的，那么编译器会尽可能地尝试将循环展开来减小总时间。而如果循环次数是由一个变量控制，而且已知一些先验的信息，比如这个变量一定大于某个值或者是某个数的倍数，那么就可以用一条编译指令 MUST_ITERATE 来告诉编译器这些信息。编译器会据此尝试展开循环进行优化。

3) 地址对齐

有时为数据采用合适的地址对齐也能起到一定的优化效果。例如对于一批字节类型的数据，若将它们的首地址以字地址对齐，那么处理器可以利用一条 LDW 指令直接取四个数据；若没有对齐，处理器只能利用 LDB 指令逐个字节取数据。因此有时采用合适的对齐方式能够使取数据更快，但具体的效果还应该结合后续的数据处理流程进行分析。

2.3.5 软件启动流程

DSP 软件的设计之初，用户可以只关心应用程序的设计，通过在线仿真器调试程序时，每个内核会自动完成堆栈初始化并从主函数入口开始执行。当程序功能得到充分验证后，需要将其写入 Flash 中，完成程序固化。此时则需要用户添加额外的引导程序来实现系统复位后程序从外部存储器载入的功能。

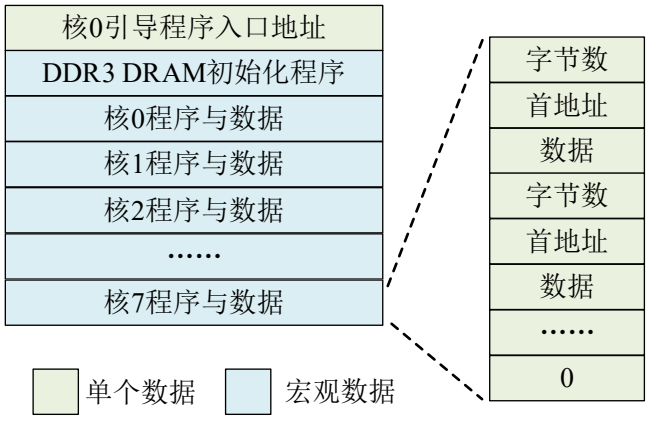


图 2-16 引导镜像文件格式

程序固化的实质是将内核运行所需的程序和数据按照引导镜像文件的格式进行组织，然后写入 Flash 的过程。在多核 DSP 系统中，整个引导过程基本由核 0 完成，核 0 在执行应用程序前需要执行一段引导程序。在引导镜像文件中首先给出引导程序的入口地址，引导镜像文件格式如图 2-16 所示。除了最开始的入口地址以外，引导镜像文件的剩余部分都是需要由核 0 搬运到指定地址的程序和数据，它们都按照同样的格式组织，并以字节数“0”作为结束标志。

DSP 通过 EMIF 启动引导的流程图如图 2-17 所示。当系统发生复位后，DSP 片内的一段写在 ROM（Read Only Memory）上的程序 RBL（ROM Boot Loader）由核

0 开始执行，RBL 的主要工作是搬运引导镜像文件中的程序和数据。搬运完成后核 0 便跳转到引导程序入口处开始执行引导程序。

实际工作中很可能会遇到每个核的程序或初始化数据需要搬运到 DDR3 DRAM 中存放，但是系统在复位后DRAM并没有初始化，因此需要有一段能够初始化DRAM 的程序在 RBL 搬运开始之前就执行。引导镜像文件中的 DDR3 DRAM 初始化程序就是起到这个作用，它需要在最开始被 RBL 搬运，而且搬运到特定的地址处。只有这样核 0 才会暂停执行 RBL 程序，转而执行 DRAM 初始化程序，在初始化完成后继续回来搬运剩余的程序和数据。

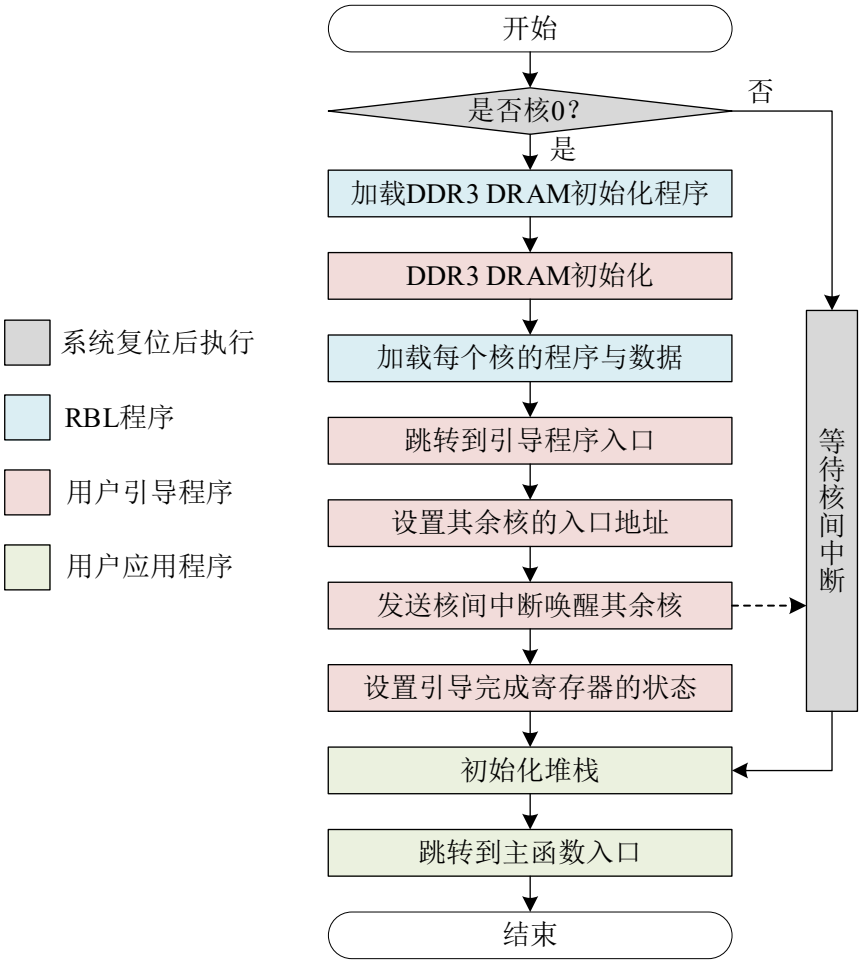


图 2-17 FT-M6678 从 EMIF 启动引导流程图

DDR3 DRAM 的初始化程序需要用户编写，因此也属于用户引导程序的一部分。需要特别注意的是用户引导程序在执行的过程中核 0 还没有初始化堆栈，因此这一

过程中不能有函数调用，也不能使用动态内存分配，大部分情况下用户引导程序都采用汇编语言编写。核 0 在用户引导程序中还可以执行其它的一些外设初始化操作，但主要工作就是为其它内核设置正确的堆栈初始化入口地址，并发送核间中断唤醒。随后核 0 设置引导完成寄存器，整个系统便进入正常的工作状态。

2.3.6 融合信息处理系统软件框架

融合信息处理系统的多核 DSP 内任务分工如图 2-18 所示，DSP 负责实现的功能需要分配到每一个内核。内核 0 负责整体的调度与融合决策，相当于主从式软件结构中的主核；内核 1 与内核 2 分别负责完成红外目标与辅助目标的跟踪；内核 3 负责管理多核共享的硬件外设与 FPGA 中的硬件加速电路调用；剩余的核都用于处理激光成像结果，提取激光特征图，完成激光目标的检测与跟踪。不同核之间都可以通过共享存储区交互数据。由内核 3 负责共享硬件的统一调用，避免不同内核同时调用而产生冲突的问题。

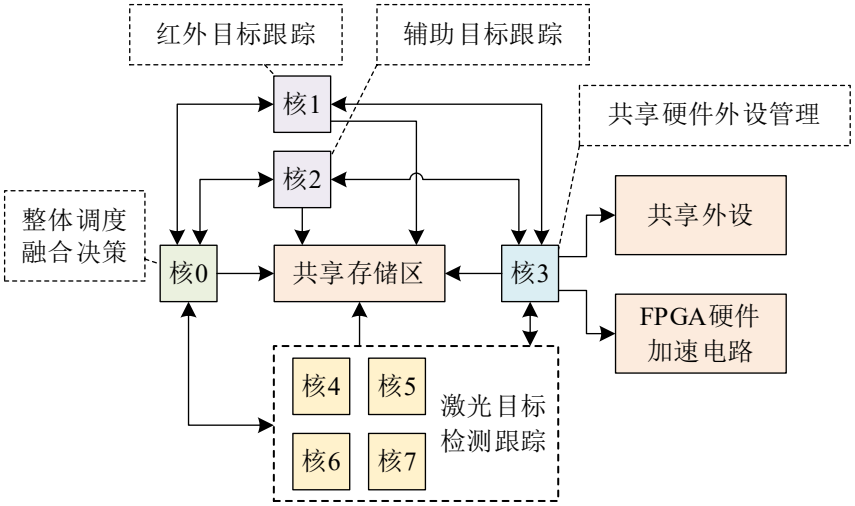


图 2-18 融合信息处理系统中 DSP 内核分工

除了内核 3 以外，整个软件框架采用以内核 0 为主，其余内核为辅的主从式软件结构。内核 0 负责接收数据，包括红外实时图像与激光成像电路的成像结果。数据接收过程采用乒乓缓冲机制，内核 0 能够周期性地收到数据接收完成中断信号。由一个专门的数据接收任务对接收的数据帧头进行校验，同时切换接收缓冲区地址，为下一次数据接收做准备。

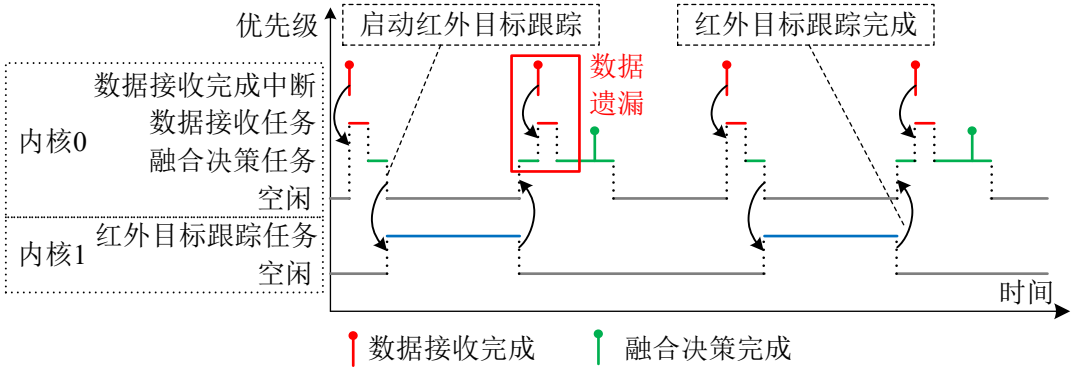


图 2-19 内核 0 采用单一的融合决策任务

在内核 0 中若仅设计单一的融合决策任务则可能会出现数据遗漏的情况，这一现象只有在算法实时性不足时才会出现。图 2-19 中展示了内核 0 调用内核 1 完成红外目标跟踪的流程，图中的横线表示在某一段时间内，某任务正在工作。每个内核中的多个任务按照优先级从高到低自上而下排列。内核 0 在收到第一帧红外实时图像后，融合决策任务调用内核 1 执行目标跟踪算法。而在收到第二帧图像时，内核 0 正在根据先前的跟踪结果执行融合决策运算，无法及时地启动新的一帧图像的处理。

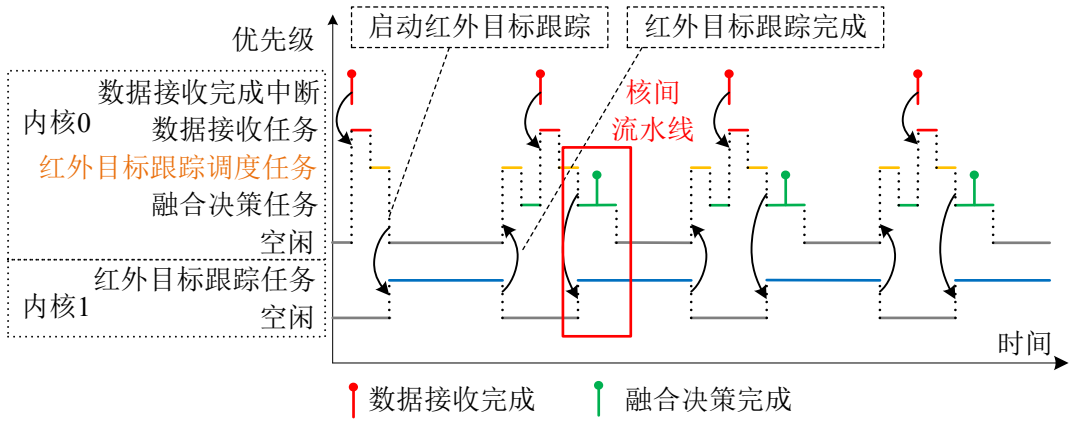


图 2-20 内核 0 引入调度任务实现核间流水线

为了解决这个潜在的问题，可以在内核 0 中设计专门的调度任务。更高优先级的调度任务可以抢断融合决策任务，从而在红外实时图就绪时能够及时地启动红外目标跟踪任务执行。图 2-20 表明在引入调度任务后等同于实现核间流水线，内核 0 在基于先前的跟踪结果进行融合决策时，下一帧就绪的图像已经开始处理。

表 2-2 中对每个内核中所需实现的任务进行了规划。内核 0 中的任务设计同样遵循主从式软件结构的设计思路。负责整体流程控制的主任务处于次高优先级，确保

其它任务不能抢断主任务的执行；数据接收任务采用最高优先级能够保证 DSP 正确接收每一组数据；所有目标检测跟踪与融合决策相关的任务都由主任务根据实际的工作模式来选择性地执行。

表 2-2 不同内核中的任务规划

内核	任务	优先级	说明
0	数据接收任务	31	负责红外实时图与激光成像结果数据接收
	主任务	7	算法整体流程控制，切换目标检测/跟踪模式
	红外目标检测任务	6	负责调用 FPGA 中的目标检测识别神经网络
	红外目标跟踪调度任务	5	负责调用内核 1 执行红外目标跟踪
	辅助目标检测任务	4	负责调用 IPCU 检测识别辅助目标
	辅助目标跟踪调度任务	3	负责调用内核 2 执行辅助目标跟踪
	激光目标检测跟踪调度任务	2	负责调用其它内核实现激光目标检测跟踪
	融合决策任务	1	根据不同阶段执行检测/跟踪融合决策
1	红外目标跟踪任务	1	红外目标跟踪
2	辅助目标跟踪任务	1	辅助目标跟踪
4~7	激光目标检测跟踪任务	1	激光目标检测跟踪

2.4 本章小结

采用远距离制冷式红外探测器与可三维成像的 SPAD 激光雷达进行目标检测跟踪具有广泛的应用前景。但是 SPAD 激光雷达产生的数据量较大，探测器成本高昂，整机系统调试环境复杂，为了对红外激光融合的目标检测跟踪算法进行初步研发，本章根据设计需求与初步的红外激光融合目标检测跟踪方案，设计了融合信息处理系统。系统以银河飞腾 FT-M6678 DSP 为核心，采用两块高性能 FPGA 与一片专用协处理器为 DSP 提供加速服务。系统通过 8 对 5.0 Gbps 的 PCIe 链路与上位机互连，便于上位机下发仿真测试数据进行算法调试。

除了硬件系统的设计，本章还提出了融合信息处理系统中的多核 DSP 软件框架，并对多核 DSP 软件的设计方法进行了介绍。包括常用的软件结构设计，各个内核的存储空间分配，SYS/BIOS 实时内核，核间通信方式，软件优化策略，软件启动流程。最后将红外激光融合信息处理系统中多核 DSP 需要实现的功能分配到具体的内核来完成，为后续的 DSP 软件开发规划了方向。

3 KCF 跟踪算法原理与实现 *

KCF 目标跟踪算法以高实时性著称，其简明的运算过程能够在大部分处理器上实现极高的帧率。但是在资源有限的 DSP 处理器上，存储资源的合理分配，硬件外设的适时调度，软件执行的细节优化等问题仍然需要审慎考虑。

将 KCF 算法移植到 DSP 上实现，首先应当熟悉 DSP 能够为计算过程提供哪些便利，同时应当熟悉算法原理与详细的计算过程。结合 DSP 提供的处理器指令集与硬件外设，针对性地设计移植方案才能够最大化算法执行效率。

本章首先对 KCF 算法的原理与实现细节进行了详细介绍，分析计算瓶颈。而后结合银河飞腾 FT-M6678 DSP 的硬件 FFT 加速器与增强型直接内存访问（Enhanced Direct Memory Access, EDMA）外设提出了批量二维 FFT 快速实现方法，有效提升了 KCF 算法的实时性。最后，为了实现红外与辅助目标同时跟踪的需求，阐述了在多核 DSP 上实现同时跟踪两个目标的设计思路与实现方案。

3.1 核相关滤波算法

相关滤波跟踪器基于相关运算理论，以训练一个滤波器为目标。将图像输入该滤波器后得到响应图，其中的峰值位置可以表征目标在新的一帧图像中的位置。滤波器的参数根据初始帧中的目标框初始化，在随后的序列里，滤波器参数不断更新，以适应目标的形态变化，进而达到跟踪的目的。KCF 算法借助循环矩阵的概念，对相关滤波跟踪器从另一个角度进行解释，使理论更加健全，为跟踪算法设计打开了新的思路。其中，核技巧的引入与多通道样本特征为算法跟踪性能带来了显著提升。

3.1.1 基本原理

跟踪算法所处理的图像块可以看作是 $m \times n$ 的二维样本，在训练过程中为每个样本设置合适的标签值，用标签值的大小表征每个图像块与目标图像块的相似程度，从而可以得到一组回归系数用于计算新的图像块与目标图像块的相似程度。而后从新

* 本章的主要内容以第一作者身份由《红外技术》期刊录用

的一帧图像中找出与目标最相似的图像块即达到了跟踪的目的。从而目标跟踪问题可以转化为二维样本的回归问题。为了便于说明，整个算法推导过程是从$n\times1$的向量样本出发的，二维样本的情况可以从前者推广得到。

现有 m 个 n 维列向量样本$\{x_1, x_2, x_3, \ldots, x_m\}$用于训练，每个样本对应有一个标签值 y_i，$i \in \{1, 2, 3, \ldots, m\}$。将这一组样本写成矩阵的形式，记作 X，$X = [x_1, x_2, x_3, \ldots, x_m]^{\mathrm{T}}$；样本的标签值也以对应的位置写成列向量的形式，记作 Y，$Y = [y_1, y_2, y_3, \ldots, y_m]^{\mathrm{T}}$。求回归系数$\omega_{n\times1}$，使得样本的预测值与真实值之间的平方误差之和最小，

$$\arg\min_{\omega} \left\| X\omega - Y \right\|_2^2 \tag{3-1}$$

这是一个最小二乘问题，可以令目标函数对ω的偏导数为零求解。在 X 列满秩的情况下，$X^{\mathrm{T}}X$ 可逆，可推导得到：

$$\omega = \left(X^{\mathrm{T}}X \right)^{-1} X^{\mathrm{T}}Y \tag{3-2}$$

更一般地，可以利用 Moore-Penrose 广义逆得到最佳的最小二乘解，但在这里不再展开。为了防止训练过拟合，KCF 算法采用的是岭回归（Ridge Regression）的训练方式，其在最小二乘回归的基础上引入了正则项$\lambda \left\| \omega \right\|_2^2$：

$$\arg\min_{\omega} \left\| X\omega - Y \right\|_2^2 + \lambda \left\| \omega \right\|_2^2 \tag{3-3}$$

同样令目标函数对ω的偏导数为零求解，可得：

$$\omega = \left(X^{\mathrm{T}}X + \lambda I \right)^{-1} X^{\mathrm{T}}Y \tag{3-4}$$

岭回归相较于最小二乘回归，其解中的$X^{\mathrm{T}}X$项需要加上对角线上都是λ的对角矩阵，像是一条山岭横跨在上方，因此得名“岭”回归。

为了引入核技巧，需要对问题的解转换到对偶空间表示。表示定理（Representer Theorem）指出，对于以下形式的线性正则化风险最小化问题：

$$\arg\min_{\omega} \sum_{i=1}^{m} L\left(\omega^{\mathrm{T}}x_i, y_i \right) + \lambda \left\| \omega \right\|_2^2 \tag{3-5}$$

其解ω一定位于样本x_i张成的子空间内，即：

$$\omega = \sum_{i=1}^{m} \alpha_i x_i = X^{\mathrm{T}} \alpha \tag{3-6}$$

其中 $\alpha \in \mathbb{R}^m$，只需要确定 α，自然就确定了原问题的解 ω，α 称作原问题在对偶空间的解。

表示定理所阐述的结论可以有比较直观的理解。$\omega = \omega_{\parallel} + \omega_{\perp}$，$\omega$ 可以分解为两部分，位于样本 x_i 张成的子空间内的一部分记作 $\omega_{\parallel}$，与所有样本 x_i 都垂直的一部分记作 $\omega_{\perp}$。将 ω 代入公式(3-5)可以得到：

$$\arg\min_{\omega} \sum_{i=1}^{m} L\left(\omega_{\parallel}^{\mathrm{T}} x_i + \omega_{\perp}^{\mathrm{T}} x_i, y_i\right) + \lambda \left\| \omega_{\parallel} + \omega_{\perp} \right\|_2^2 \tag{3-7}$$

$$= \arg\min_{\omega} \sum_{i=1}^{m} L\left(\omega_{\parallel}^{\mathrm{T}} x_i, y_i\right) + \lambda \left\| \omega_{\parallel} \right\|_2^2 + \lambda \left\| \omega_{\perp} \right\|_2^2 \tag{3-8}$$

从公式(3-8)可以看出，对于任意 $\omega = \omega_{\parallel} + \omega_{\perp}$，一定有一个 $\omega' = \omega_{\parallel}$ 能够使目标函数的值更小，所以最优解 ω^* 一定是所有样本 x_i 的线性组合。

解的维度与问题求解的复杂度正相关，表示定理的结论为求解问题(3-3)提供了两个角度。当样本个数 m 大于样本维度 n，可以直接通过公式(3-4)求解 ω，$\omega \in \mathbb{R}^n$；而当样本维度 n 远大于样本个数 m 的时候，用 α 来表示问题的解将会是更好的选择。联立公式(3-4)与公式(3-6)，可得：

$$\alpha = \left(XX^{\mathrm{T}} + \lambda I\right)^{-1} Y \tag{3-9}$$

其中 XX^{T} 称为 m 个样本向量的格拉姆矩阵，它是由这 m 个样本两两之间内积组成的矩阵。

利用 α 表示问题的解，不仅能在样本维度太大时简化计算，而且能够将样本训练与预测统一为样本间的内积计算。公式(3-9)是样本的训练过程，训练结果直接由样本两两之间的内积决定；当给出一个新的样本 $z_{n\times 1}$，计算预测值 y_z：

$$y_z = z^{\mathrm{T}} \omega = z^{\mathrm{T}} X^{\mathrm{T}} \alpha = \left(Xz\right)^{\mathrm{T}} \alpha \tag{3-10}$$

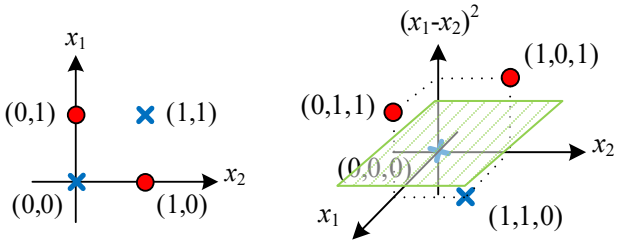
预测值 y_z 也是由新的样本 z 与训练样本 X 之间的内积 Xz 直接决定的。

在分类问题中，提高样本的维度，能够使原本线性不可分的问题变得线性可分。

比如典型的异或问题，如图 3-1 所示。两类样本在二维空间线性不可分，通过引入一个映射函数$\psi(x)$，

$$\psi\left(x\right)=\left[x_1,x_2,(x_1-x_2)^2\right]^{\mathrm{T}} \tag{3-11}$$

将二维样本映射到三维空间后，两类样本变得线性可分。



(a) 二维空间线性不可分　　(b) 三维空间线性可分

图 3-1 异或问题在二维空间与三维空间的线性可分性

前面所讨论的线性回归问题，也存在类似的局限性。如果样本与标签值之间并不存在某种线性关系，那么训练的结果也将是不可靠的，预测值与真实值会有较大偏差。因此需要引入核技巧（Kernel Trick），通过一个非线性函数$\psi(x)$将样本映射到更高维的空间。

高维空间中的样本，仍然只需要关心样本间的内积。核函数κ能够快速地计算得到两个样本 x 与x'在高维空间中的内积，而不需要关心它们在高维空间的具体形式$\psi(x)$或$\psi(x')$。

$$<\psi(x),\psi(x')>=\kappa(x,x') \tag{3-12}$$

典型的核函数如高斯核函数，或者称为径向基函数：

$$\kappa(x,x')=\exp\left(-\frac{1}{2\sigma^2}\left\|x-x'\right\|_2^2\right)=\exp\left(-\frac{1}{2\sigma^2}\left(\left\|x\right\|_2^2+\left\|x'\right\|_2^2-2x^{\mathrm{T}}x'\right)\right) \tag{3-13}$$

其中，σ是高斯函数的标准差。它所对应的映射函数$\psi(x)$将样本映射到无限维的空间，能够大大提高样本线性回归的准确性。引入核技巧后的岭回归只需要将原先的格拉姆矩阵XX^{T}替换为由高斯核函数计算得到的矩阵K即可，

$$\alpha=\left(K+\lambda I\right)^{-1}Y \tag{3-14}$$

其中$K_{ij}=\kappa(x_i,x_j)$。预测过程中也只需要将原先新样本z与训练样本X的内积替换为

高斯核函数的计算$\kappa(z,x_i)$，$i\in\{1,2,3,...,m\}$。核技巧的引入使岭回归不仅局限于线性回归，更有了一定的对非线性关系的回归能力。

3.1.2 循环矩阵加速

KCF 算法的基本原理介绍了样本训练与新样本推理的过程。但是如何快速地获取大量训练样本？如何高效地完成样本间的内积计算？这两个问题便由循环矩阵来解决。循环矩阵的概念仍然是对于一维样本来说的，顾名思义，循环矩阵是由样本经过循环移位构成的矩阵。现有一维样本$x_{n\times1}$，则它的循环矩阵$C(x)$为：

$$C(x)=[x,P^0x,P^1x,\ldots,P^{n-1}x]^{\mathrm{T}} \tag{3-15}$$

其中P^i是置换矩阵（Permutation Matrix），它的形式如下：

$$P^i=\begin{bmatrix}0 & I_i\\ I_{n-i} & 0\end{bmatrix} \tag{3-16}$$

根据循环移位的程度不同，可以为这 n 个样本设置合适的标签值用于训练。循环矩阵有两条重要的性质，能帮助简化样本训练与预测的计算过程。

性质 1. 循环矩阵可以用酉矩阵对角化。

酉矩阵U 又称为幺正矩阵，满足$U^{\mathrm{H}}U=UU^{\mathrm{H}}=I_n$，即酉矩阵与其共轭转置互为逆矩阵。循环矩阵对角化借助离散傅里叶变换（Discrete Fourier Transform, DFT）实现，将样本x看作长度为n的离散时间序列，其 DFT 变换后的序列可以用Fx表示。矩阵$F_{n\times n}$具有类似酉矩阵的特性，记酉矩阵$U=F/\sqrt{n}$，则循环矩阵$C(x)$可以表示为：

$$C(x)=U\,\mathrm{diag}(\hat{x})U^{\mathrm{H}} \tag{3-17}$$

具体推导过程在附录 A.1 中给出。

性质 2. 一组由循环移位得到的样本向量的格拉姆矩阵是循环矩阵。

$$C(x)C(x)^{\mathrm{T}}=C\big(C(x)x\big) \tag{3-18}$$

循环矩阵中本质上仅有一个样本，样本两两之间的内积只需要计算$C(x)x$就可以得到，$C(x)x$的循环移位构成了这一组样本的格拉姆矩阵。结合公式(3-15)和置换矩阵的性质也可以对公式(3-18)进行验证。

公式(3-14)中的矩阵 K 是高维特征空间中的格拉姆矩阵。可以理解为样本 x 映射到高维空间后得到 $\psi(x)$，其经过循环移位后得到一组样本，这一组样本向量的格拉姆矩阵是 K 是一个循环矩阵，

$$K = C(k^{xx}) = U\,\mathrm{diag}(\hat{k}^{xx})U^{\mathrm{H}} \tag{3-19}$$

其中 $k^{xx} = [\kappa(x, P^0x), \kappa(x, P^1x), \kappa(x, P^2x), \ldots, \kappa(x, P^{n-1}x)]$，重写公式(3-14)并整理：

$$\hat{\alpha}_{n\times 1} = \frac{\hat{Y}_{n\times 1}}{\hat{k}^{xx} + \lambda} \tag{3-20}$$

其中的除法运算表示两个向量之间对应位置分量的除法。进一步分析可以发现，由于 k^{xx} 是中心对称的，它的 DFT 变换结果 $\hat{k}^{xx}$ 是实数组成的向量；标签值 $Y_{n\times 1}$ 也可以通过精心设计使其中心对称，从而令 $\hat{Y}_{n\times 1}$ 也是由实数组成。最终能够保证除法运算是简单的实数除法，并且所得结果 $\hat{\alpha}$ 也是实数。

样本训练过程在引入循环矩阵后，不仅能够获得大量样本，而且使训练过程变得简洁明了。对于新样本 z 的预测过程，也可以将 z 循环移位得到一批新的样本一起进行预测，从而根据不同循环移位程度的样本得到的预测值的大小确定新的样本相对于训练样本 x 的偏移程度。

$$f(z) = C(z)\left(C(x)\right)^{\mathrm{T}}\alpha = \left(C(x)\left(C(z)\right)^{\mathrm{T}}\right)^{\mathrm{T}}\alpha \tag{3-21}$$

$C(x)\left(C(z)\right)^{\mathrm{T}}$ 也是一个循环矩阵，引入核函数后，将 $C\left(\psi(x)\right)\left(C\left(\psi(z)\right)\right)^{\mathrm{T}}$ 记作 K^{xz}，满足 $K_{ij}^{xz} = \kappa(P^i x, P^j z)$，用循环矩阵的形式表示：

$$K^{xz} = C(k^{xz}) \tag{3-22}$$

其中 $k^{xz} = [\kappa(x, P^0z), \kappa(x, P^1z), \kappa(x, P^2z), \ldots, \kappa(x, P^{n-1}z)]$。再利用循环矩阵可对角化的性质，对引入了核技巧的预测过程进行化简：

$$f(z) = \mathcal{F}^{-1}\left(\hat{k}^{xz} \odot \hat{\alpha}\right) \tag{3-23}$$

至此，不论是样本的训练或是预测，都能够用 DFT 加速运算，并且不再涉及矩阵乘法或求逆，只需要求向量之间逐分量的乘除法。最后，核函数的计算也可以优化，本质上都是利用 DFT 加速相关运算。例如，采用高斯核函数时，

$$k^{xx}=\exp\left(-\frac{1}{\sigma^2}\left(\|x\|_2^2\mathbf{1}_{n\times1}-\mathcal{F}^{-1}\left(\hat{x}\odot\hat{x}^*\right)\right)\right) \tag{3-24}$$

$$k^{xz}=\exp\left(-\frac{1}{2\sigma^2}\left(\|x\|_2^2\mathbf{1}_{n\times1}+\|z\|_2^2\mathbf{1}_{n\times1}-2\mathcal{F}^{-1}\left(\hat{x}\odot\hat{z}^*\right)\right)\right) \tag{3-25}$$

可以用公式(3-24)和公式(3-25)计算 k^{xx} 和 k^{xz}，其中的指数运算是针对向量中每个分量的运算。

3.1.3 特征提取与多通道样本

KCF 算法引入核技巧，使岭回归具有了一定的非线性回归能力，提升了算法的跟踪性能，除此之外，可靠的特征提取也是提升算法性能的关键。算法采用 FHOG 特征，FHOG 特征是由 Felzenszwalb 等人提出的方向梯度直方图特征（Histogram of Oriented Gradient, HOG）[44]。区别于传统的 HOG 特征，FHOG 特征将相反方向的梯度进行区分，并创新性的采用“soft binning”的策略，让每个像素的梯度幅值不仅对其所在的 Cell 有贡献，也让其对周围的 Cell 有贡献。FHOG 特征具有一定的光照不变性和抗形变能力，能更好地描述了梯度的细节，在实践中也表现出更好的性能。

特征提取从计算每个像素点的梯度幅值和方向开始。首先利用锚点位于中心的 1×3 和 3×1 的卷积模板 $[-1,0,1]$、$[-1,0,1]^{\mathrm{T}}$ 分别提取图像块横向的梯度 dx 和纵向的梯度 dy，得到梯度向量 $u=(dx,dy)$。图像块的边缘采用镜像扩展，使得边缘的梯度为零。梯度的大小 A 即梯度向量的长度，可以通过下式计算：

$$A=\sqrt{d_x^2+d_y^2} \tag{3-26}$$

传统的 HOG 特征以梯度向量与 x 轴正方向的夹角 γ 作为梯度方向，

$$\gamma=\begin{cases}\arctan(dy/dx), dy/dx\geq0\\ \pi/2, dx=0\\ \pi+\arctan(dy/dx), dy/dx<0\end{cases} \tag{3-27}$$

一般不区分相反方向的梯度，因此 γ 的值域是 $[0,\pi]$。直方图统计过程中，针对一个 Cell 内的像素进行统计，在 0~180 度内均匀地取若干角度作为方向梯度直方图的 bin。若某一梯度向量的角度 γ 处于两个相邻的 bin 之间，则采用类似线性插值的方式将梯

度幅值分别累加到两个 bin 中。例如图 3-2 中对 4×4 的 Cell 内的像素统计直方图，$\gamma > \theta_1$ 并且 $\gamma < \theta_2$，那么梯度幅值 A 需要乘以 η 才能累加到 θ_1 对应的 bin 中，并且需要乘以 $(1-\eta)$ 才能累加到 θ_2 对应的 bin 中，其中 η 满足：

$$\eta = \frac{\theta_2 - \gamma}{\theta_2 - \theta_1} \tag{3-28}$$

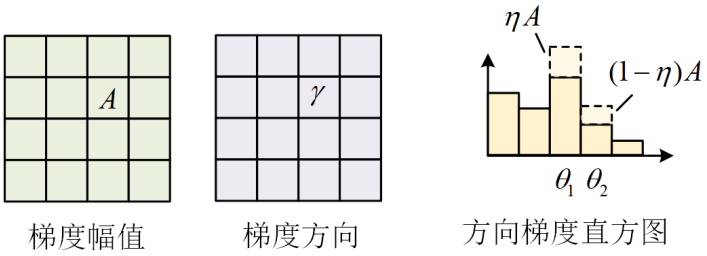


图 3-2 传统 HOG 特征的方向梯度直方图统计方式

FHOG 特征确定梯度向量方向与直方图统计的方式略有不同。可能是考虑到利用反正切函数计算角度过于繁琐，它通过比较梯度向量与一些单位基准向量内积的大小直接确定该梯度向量所在的 bin。

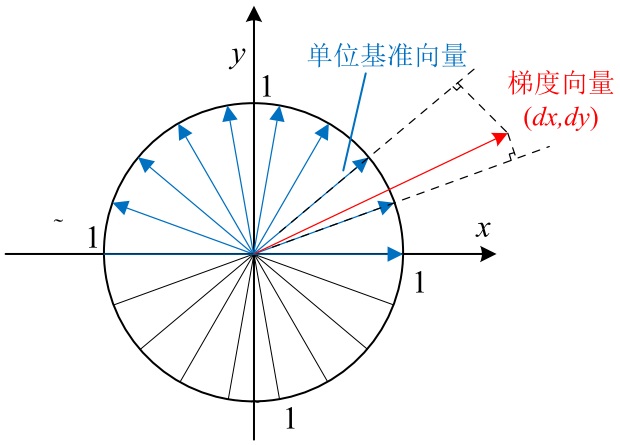


图 3-3 梯度向量示意图

单位基准向量是均匀分布在 x-y 平面的单位圆上的向量，如图 3-3 所示。记某个单位基准向量 v 与 x 轴正方向的夹角为 θ，将向量用坐标的形式表示，$v = (\cos\theta, \sin\theta)$。

$$d_{cos} = \frac{< u, v >}{|u| \cdot |v|} = \frac{dx\cos\theta + dy\sin\theta}{A} \tag{3-29}$$

向量 u、v 之间的余弦距离即计算它们的夹角余弦值，梯度向量与单位基准向量的余

弦距离计算如公式(3-29)所示，其值域为[−1,1]。将梯度向量与有限个数的单位基准向量依次计算余弦距离，找出使计算结果绝对值最大的单位基准向量即确定梯度向量的方向，结果的符号可以确定梯度向量具体是与基准向量同向或是反向。实际计算过程中由于仅需要比较不同θ下余弦距离的相对大小，因此可以省去归一化的过程，只需要计算不同θ时的向量内积$<u,v>$进行比较，令向量内积$<u,v>$的绝对值最大的θ即该梯度向量的方向，而不需要进行类似线性插值的处理。

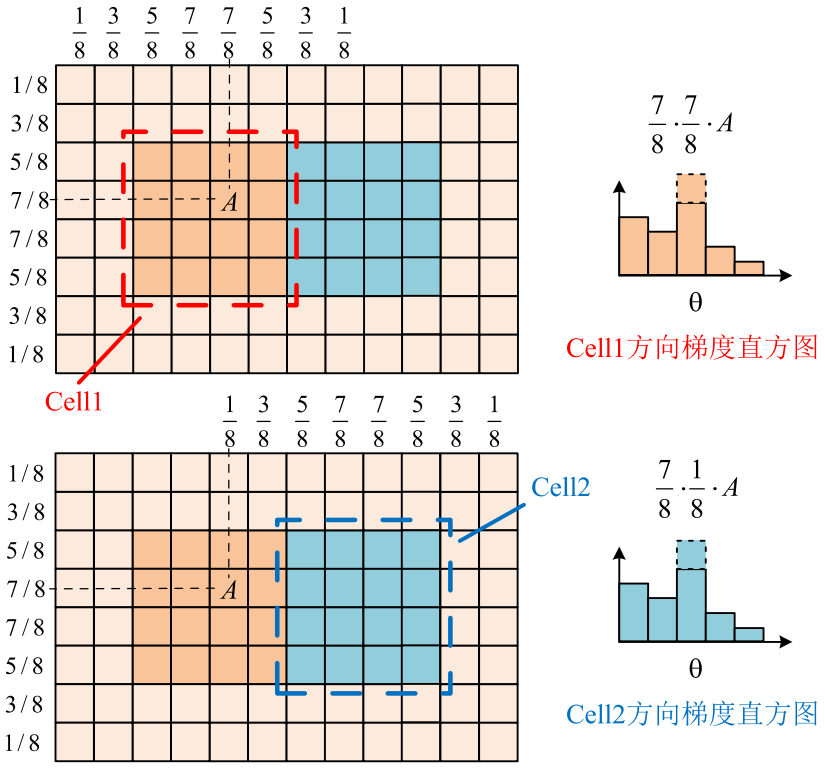


图 3-4 FHOG 特征梯度幅值累加方式

直方图统计的过程采用“soft binning”的策略。传统 HOG 特征中每个 Cell 对应一个方向梯度直方图，每个梯度向量仅对其所在的 Cell 对应直方图有贡献；而 FHOG 特征则令每个像素对其所在位置附近的四个直方图均有贡献，贡献的程度根据距离远近类似双线性插值的方式分配。每个 Cell 都会在行列方向上扩展，在扩展后的 8×8 的范围内的像素都对这个 Cell 的直方图有贡献，每个像素的梯度幅值都需要乘以行列方向上的比例系数再累加到直方图中。如图 3-4 中有两个相邻的 4×4 的 Cell，Cell1 内的一个像素的梯度幅值为A，梯度方向为θ，其对 Cell1 和 Cell2 的方向梯度

直方图均有贡献。另外在这两个 Cell 上方也有两个 Cell 在扩展成 8×8 后会覆盖到该像素。每个梯度向量都以类似的方式将它的梯度幅值以类似双线性插值的方式分成 4 份分别累加到 4 个不同的方向梯度直方图中。

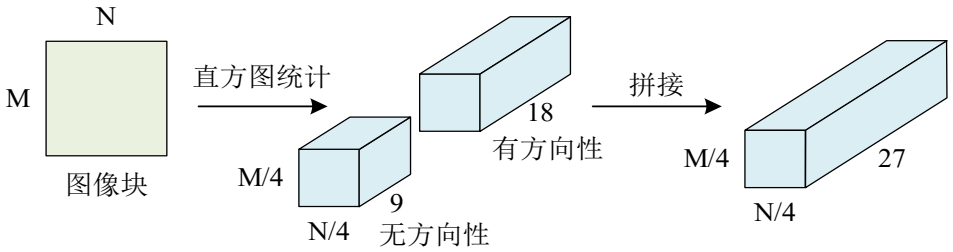


图 3-5 FHOG 特征方向梯度直方图统计

FHOG 特征采用 4×4 的 Cell，在 0~180 度内均匀分成 9 个方向。对相反的梯度方向不进行区分，得到长度为 9 的直方图；区分相反的梯度方向，则能够得到长度为 18 的直方图。级联两种直方图后可以得到 27 通道的 M/4×N/4 的方向梯度直方图统计结果，其过程如图 3-5 所示。

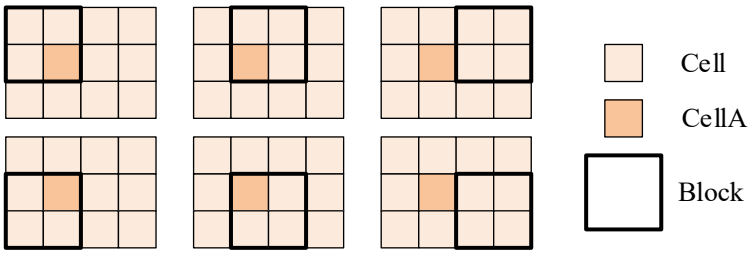


图 3-6 FHOG 特征中的 Block 窗口滑动

为了应对光照变化的影响，需要对直方图归一化，同时为了更加充分地利用相邻像素的信息，归一化不局限在单个 Cell 内进行，而是在多个 Cell 组成的 Block 内归一化。临近的 Block 之间存在重叠，Block 就像一个窗口在图像块上滑动，其过程如图 3-6 所示。4 个相邻的 Cell 组成一个 Block，除了位于图像块边缘的 Cell 以外，每个 Cell 存在于 4 个不同的 Block 中，如图中的 CellA 被四个 Block 所包含。因此一组直方图需要进行 4 次归一化的计算。每个 Cell 对应的长度为 27 的直方图经过归一化后能够得到长度为 108 的特征向量。有时可能遇到某个 Block 内的梯度非常小，导致归一化的分母趋近于 0，归一化后的结果远远超出平均水平，因此需要对归一化后的结果截断，防止产生过多噪声。

归一化后的 108 维特征向量不利于后续的计算，过多的特征维度也容易使样本训练产生过拟合，因此需要对样本特征降维。利用主成分分析（Principal Component Analysis, PCA）的方法对样本特征进行降维，一般需要利用协方差矩阵求解样本的特征值和特征向量，但是这里的特征向量是由归一化得到的，经过分析观察可以得出较为简单的计算方法，而不需要复杂的矩阵运算。

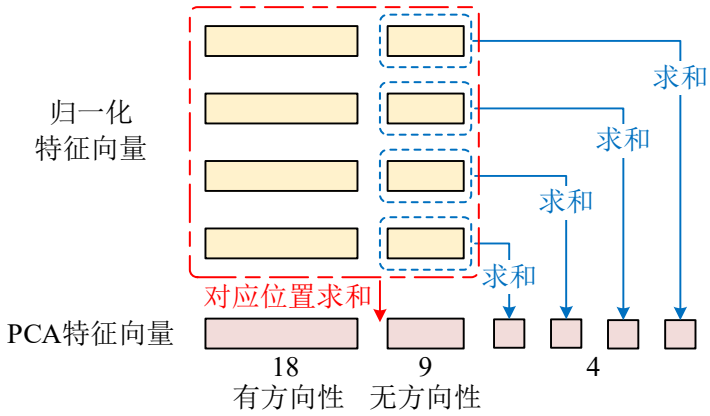


图 3-7 FHOG 特征 PCA 降维计算过程

PCA 降维的过程如图 3-7 所示，四组归一化后的 27 维特征向量对应位置的分量分别求和可以得到 27 维特征，这是方向梯度直方图的主要成分，其中包括 9 维不区分梯度向量方向的特征和 18 维区分梯度方向的特征；将四组无方向性的 9 维特征向量分别求和可以另外得到 4 维特征，它们用于表征 4 组不同的归一化结果。因此最终每个 Cell 对应一个 31（27+4）维的特征向量，Cell 的个数由图像块的大小决定。若一幅图像中除了位于图像边缘的 Cell 以外，共有 P×Q 个 Cell，那么最终能够得到 P×Q 大小的 31 通道特征图。

经过特征提取的图像块能够更好地抓住图像中的关键信息，相比于直接用未提取特征的图像块进行训练，特征提取后的训练结果更加稳定可靠。但是在提取 FHOG 特征后，样本由单通道变成了多通道。多通道样本如何训练？多通道样本是否会带来更多复杂的运算？KCF 算法针对这两个问题也给出了解答。

多通道样本的训练可以由样本级联实现。仍然以一维样本为例，如图 3-8 所示，现有一个一维样本 $x_{5\times1}$，将它循环移位 4 次后可以得到一组训练样本 $X_{5\times5}$，为这组样本分配标签值 $Y_{5\times1}$。如果引入一个新的通道的样本 $u_{5\times1}$，并且进行相同的循环移位，

它所产生的样本也应当赋予同样的标签值。将循环移位程度相同的样本级联可以得到新的样本矩阵 $X'_{5\times10}$，多通道样本的引入相当于提高了样本的维度。级联后样本间内积是级联前各样本间的内积之和。如果以 $z_{(m+n)\times1}=\{p_{m\times1},q_{n\times1}\}$ 表示长度分别为 m 和 n 的两个列向量级联得到长度为 $m+n$ 的列向量 z，那么：

$$z^{\mathrm{T}}z=p^{\mathrm{T}}p+q^{\mathrm{T}}q \tag{3-30}$$

幸运的是，3.1.1 中的结论表明，当把训练结果转换到对偶空间表示时，样本的训练和预测都统一到了样本间的内积计算。因此只需要在单通道样本的基础上，将涉及样本间内积计算的过程替换为多通道各自的内积计算，最后求和即可。

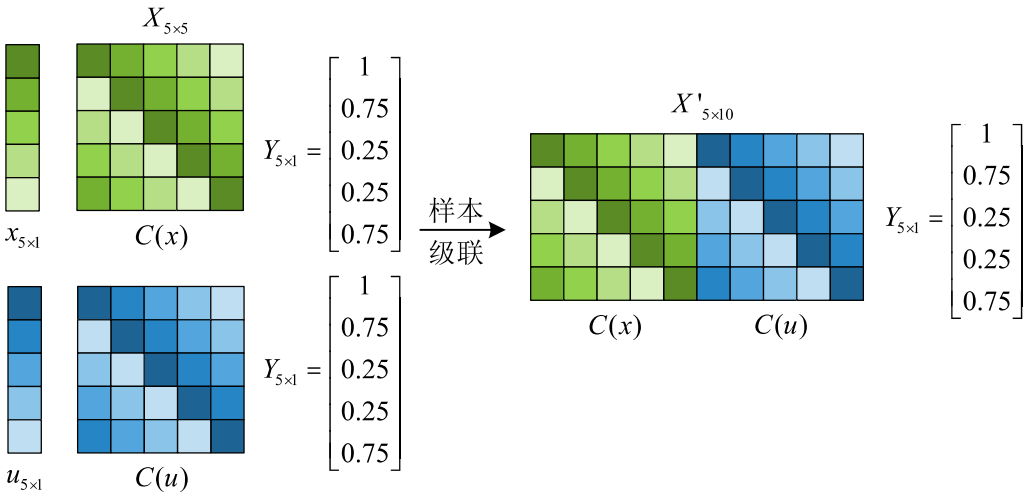


图 3-8 多通道样本级联

基于公式(3-24)和公式(3-25)可以给出在 m 通道 $n\times1$ 的样本的情况下，计算 k^{xx} 和 k^{xz} 的公式：

$$k^{xx}=\exp\left(-\frac{1}{\sigma^2}\left(\left(\sum_{c=0}^{m-1}\|x_c\|_2^2\right)\mathbf{1}_{n\times1}-\mathcal{F}^{-1}\left(\sum_{c=0}^{m-1}\left(\hat{x}_c\odot\hat{x}_c^*\right)\right)\right)\right) \tag{3-31}$$

$$k^{xz}=\exp\left(-\frac{1}{2\sigma^2}\left(\left(\sum_{c=0}^{m-1}\left(\|x_c\|_2^2+\|z_c\|_2^2\right)\right)\mathbf{1}_{n\times1}-2\mathcal{F}^{-1}\left(\sum_{c=0}^{m-1}\left(\hat{x}_c\odot\hat{z}_c^*\right)\right)\right)\right) \tag{3-32}$$

其中利用了离散傅里叶变换的线性性质，交换了求和与计算 IDFT 的顺序。相比于单通道的样本计算，多通道样本并没有显著增加计算量，计算得到 k^{xx} 和 k^{xz} 后就可以同单通道样本一样，利用公式(3-20)和公式(3-23)实现样本的训练与预测。

3.1.4 算法流程与实现细节

根据样本的预测值，可以得出目标相对于训练样本的偏移量。假设现已利用样本 x 完成了训练，输入新样本 z，满足 $z = P^1 x$，需要确定 z 与 x 的相对位置关系。

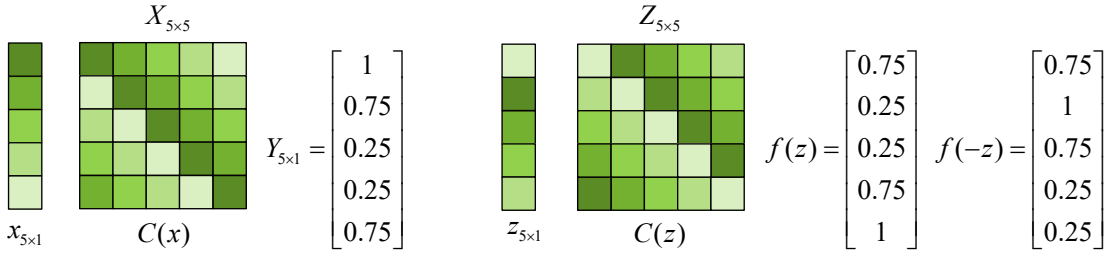


图 3-9 目标位置偏移量估计示意图

将样本 z 也循环移位，计算循环移位得到的样本的预测值。如图 3-9 所示，根据颜色对应关系，可以很快得出每个样本的预测值组成的列向量 $f(z)$。如果将由 x 变换到 z 循环移位的方向记作正方向，那么 $f(z)$ 就像是训练时的标签向量 Y 向反方向循环移位 1 个单位。这一现象说明 $f(z)$ 中峰值位置距离原点的偏移量 δx 的含义是：该新样本需要循环移位 δx 才能与训练样本一致。

这一结果在目标跟踪的过程中是反直觉的，比较理想的是 $f(z)$ 中的峰值位置能够直接表明当前目标位置相对于训练样本的偏移量。而 $f(-z)$ 的形式正好与理想结果一致，因此在实践中一般直接求 $f(-z)$。根据离散傅里叶变换的性质：

$$\hat{f}(-z) = \mathcal{F}\left(f(z)\right)^* = \left(\hat{k}^{xz} \odot \hat{\alpha}\right)^* = \hat{k}^{zx} \odot \hat{a} \tag{3-33}$$

其中，由于 $\hat{\alpha}$ 是实数，所以其取复共轭仍然是本身；$\hat{k}^{zx}$ 相比于 $\hat{k}^{xz}$ 的区别在于，$\hat{k}^{xz}$ 在计算过程中是对 $\hat{z}$ 取复共轭，而 $\hat{k}^{zx}$ 是对 $\hat{x}$ 取复共轭，类似于公式(3-32)，

$$k^{zx} = \exp\left(-\frac{1}{2\sigma^2}\left(\left(\sum_{c=0}^{m-1}\left(\left\|x_c\right\|_2^2 + \left\|z_c\right\|_2^2\right)\right)\mathbf{1}_{n\times1} - 2\mathcal{F}^{-1}\left(\sum_{c=0}^{m-1}\left(\hat{x}_c^* \odot \hat{z}_c\right)\right)\right)\right) \tag{3-34}$$

由于数字序列都是离散的，$f(z)$ 中的峰值位置，不一定恰好就是目标实际的偏移量，需要对目标实际的偏移量作更加细粒度的估计。这里有一个前提是训练样本中的目标确实存在于新的样本中，并且只是产生了位移和微小的形态变化，否则 $f(z)$ 中的峰值不明显，该预测结果是没有意义的。

真实的峰值位置可以根据峰值点两侧的值来估计，例如在 $f(z)$ 中找到峰值为 c，在峰值处建立直角坐标系，以峰值所在位置为原点，相邻点之间的距离为单位一。以峰值点附近的三个点来拟合二次函数，如图 3-10 所示。

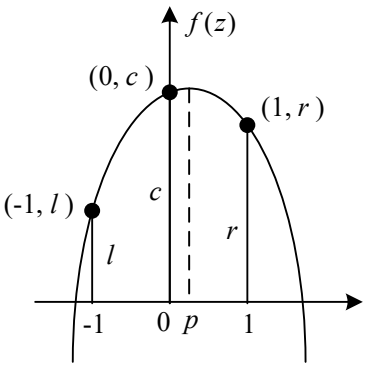


图 3-10 峰值点位置估计

将三个点的坐标 $(-1,l)$、$(0,c)$ 和 $(1,r)$ 代入二次函数 $g(t)=at^2+bt+q$ 中求三个系数 a, b, c，并估计峰值点的坐标 p：

$$\begin{cases} a-b+q=l \\ q=c \\ a+b+q=r \end{cases} \tag{3-35}$$

可以得到：

$$p=-\frac{b}{2a}=-\frac{r-1}{2\left(l+r-2c\right)}=\frac{r-1}{2\left(2c-l-r\right)} \tag{3-36}$$

在找到新的目标位置后，为了适应目标的变化，需要在新的位置重新训练并以固定的学习率线性更新训练样本 X 和训练结果 $\hat{\alpha}$。

$$X=(1-\eta)X_{old}+\eta X_{new} \tag{3-37}$$

$$\hat{\alpha}=(1-\eta)\hat{\alpha}_{old}+\eta\hat{\alpha}_{new} \tag{3-38}$$

在实现过程中还有许多细节需要考虑，比如样本的扩展与边界效应，图像块加窗处理，目标跟踪异常的情况等。由循环移位得到的样本在边界处往往存在不连续的情况。如果样本图像块仅包含目标，那么样本经过微小的移位后，其中的目标就不完整了。因此用于训练的样本是由目标图像块扩展得到的，一般会固定目标所在的感兴趣区域（Region Of Interest, ROI）的中心，将 ROI 的长和宽扩大 2.5 倍作为样本。在数

字信号处理中，为了抑制频谱泄漏，一般会对非整数倍周期采样的信号加窗处理，然后再对加窗后的序列进行 DFT 变换。二维图像虽然没有周期或非周期的说法，但是在对它进行 DFT 变换前也需要加窗，目的是防止图像边缘的不连续性也被作为目标特征的一部分参与训练。KCF 算法中加窗采用的是 Hanning 窗，它能够抑制图像边缘的信息。在目标跟踪的过程中，可能会遇到目标产生巨大形变或被遮挡或移出视场的情况，那么 KCF 跟踪的结果将是不可预计的。特别是当跟踪结果超出图像边界时，如果不加判断，那就很可能发生数据访问错误。因此对跟踪结果有效性的判断和数值范围限制的措施也是算法中不可或缺的一部分。

算法 1：KCF 算法

输入：N 张灰度图像 Images(n)，$n=0,1,2,\ldots,N-1$；初始目标 *roi*；训练更新学习率 η

输出：用于保存目标跟踪结果列表 *rois*

Initialization
    清空结果列表，$rois=[]$；
    初始化高斯形式的训练标签值 Y；
    $X_{tmp}=\text{getFeatures}(\text{Images}(0),roi)$
    $\text{Train}(X_{tmp},Y,1)$
End
For $img=\text{Images}(1:\text{N}-1)$
    $Z=\text{getFeatures}(img,roi)$
    $k^{zx}=\text{gaussCorrelation}(X,Z)$
    $f(z)=\mathcal{F}^{-1}(\hat{k}^{zx}\odot\hat{\alpha})$
    找到 $f(z)$ 中的最大值，更新 *roi*；
    保存更新后的结果，$rois.\text{append}(roi)$
    $X_{tmp}=\text{getFeatures}(img,roi)$
    $\text{Train}(X_{tmp},Y,\eta)$
End
Function $\text{Train}(X_{tmp},Y,\eta)$
    $k^{xx}=\text{gaussCorrelation}(X_{tmp},X_{tmp})$
    $\hat{\alpha}_{tmp}=\hat{Y}/(\hat{k}^{xx}+\lambda)$
    $X=(1-\eta)X+\eta X_{tmp}$
    $\hat{\alpha}=(1-\eta)\hat{\alpha}+\eta\hat{\alpha}_{tmp}$
End

KCF 算法的整体流程如算法 1 所示，getFeatures()函数实现 3.1.3 介绍的特征提取功能 gaussCorrelation()函数实现公式(3-31)和公式(3-34)的计算；X 和 $\hat{\alpha}$ 分别是训练

样本的特征图与训练结果，在目标跟踪过程中以学习率η线性更新。

KCF 目标跟踪算法的推导过程基于一维样本，而实际需要跟踪的是二维图像中的目标，虽然能够通过推广得到类似的结论，但是仍旧缺少直接的严谨推导与证明。尽管如此，算法中的循环移位与相关运算展现出了紧密的联系。概览整个算法，虽然在推导过程中没有提及相关滤波的理论，但实际的计算过程却与相关滤波密不可分。KCF 算法采用线性核，岭回归替换为最小二乘回归后就退化为 MOSSE 算法，因此其本质上是一种基于相关滤波的算法[14]。

3.2 批量二维 FFT 快速实现

3.2.1 软硬件 FFT 对比

银河飞腾 FT-M6678 DSP 片内带有 FFT 硬件加速器，支持单精度浮点数与定点小数，能够批量完成多个序列的 FFT 计算。KCF 算法的计算过程涉及大量二维 DFT 变换，因此可以利用 DSP 的硬件 FFT 加速器来加速算法中的相关运算。

DSP 中算法的处理时间与很多因素有关，比如代码/数据在存储器中的组织方式，Cache 容量的大小与映射方式，编译器的优化等级等等。FT-M6678 DSP 不仅支持硬件 FFT，也支持利用 dsplib 库调用软件 FFT。为了尽可能客观公正地对两者的性能进行对比，评估硬件 FFT 加速的潜力，共设计了两组对比实验。SW_1 和 HW_1 代表仅执行一次 FFT 计算，软硬件分别所需的周期数；SW_{100} 和 HW_{100} 代表连续进行 100 次 FFT 计算，软硬件分别所需的平均周期数。FFT 的序列长度覆盖 16~256 之间的 2 的整数次幂。相关实验环境及源码详见附录 A.3 ，实验结果如表 3-1 所示。

表 3-1 软硬件 FFT 计算时间对比

FFT 序列长度	SW_1	HW_1	硬件加速比例	SW_{100}	HW_{100}	硬件加速比例
16	986	5362	-443.81%	582.06	81.32	86.03%
32	1082	5844	-440.11%	662.76	134.95	79.64%
64	1362	6464	-374.60%	897.43	234.36	73.89%
128	2876	7024	-144.23%	2647.17	444.93	83.19%
256	3746	7693	-105.37%	4166.64	891.08	78.61%

在相同的处理器时钟下，若处理相同长度的序列，软件所用的周期数为t_s，硬件

所用的周期数为t_h，则硬件加速比例η定义为：

$$\eta = \frac{t_s - t_h}{t_s} \times 100\% \tag{3-39}$$

观察表 3-1 中的结果可以发现，单次调用硬件 FFT 加速器所需的时间较多，不如直接用软件计算方便；连续对多个序列进行 FFT 变换时，硬件计算相比于软件计算能够减少将近 80%的时间，起到显著的加速效果。

实验中待计算的序列均存放在片外双倍数据速率（Double Data Rate, DDR）存储器中，计算结果也需要写回 DDR 存储器。进一步观察软件 FFT 在不同的序列长度下所需的时间。当序列长度较短时，连续多次执行可以降低每次计算的平均时间；随着序列长度的增长，每次计算的平均时间超过了单次计算所需的时间。这是因为 DDR 存储器内跨行访问数据需要较长时间。在序列长度较短时，多次计算可以平摊存储器访问的额外时间开销；当序列长度显著增长时，这部分额外开销逐渐占据主导地位，形成计算瓶颈。硬件 FFT 的执行类似于处理器发送一条单指令多数据的命令，每次处理器下发启动命令直到收到完成中断信号视作一次硬件 FFT 调用。每次调用可以连续完成多次 FFT 计算。观察硬件 FFT 处理不同长度序列所需的时间，连续多次计算可以大幅度降低平均每次的开销。虽然单次 FFT 计算，硬件相较于软件所需的时间更久；但是在二维图像的 DFT 变换中，需要频繁连续计算多次一维 DFT，因此利用硬件 FFT 加速 KCF 算法的相关运算是可行的，只是 FFT 运算要求序列长度为 2 的整数次幂，这对图像的大小将产生一定的限制。

3.2.2 算法调整

原始图像块经过特征提取后得到多通道特征图，需要使每一通道的长宽都固定为 2 的整数幂才能利用 FFT 加速算法。采用双线性插值的方式缩放 M×N 的原始图像块为 T×T，T 满足：

$$\mathrm{T} = \left(2^n + 2\right) \times L, n \in Z^+ \tag{3-40}$$

其中 L 表示 FHOG 特征中 Cell 的长和宽，一般 $L = 4$。满足公式(3-40)的 T 的取值能够保证 FHOG 特征提取后得到的每个通道的特征图边长均为二的幂次。

n 的取值可以事先固定，或者根据初始目标的大小动态调整。若事先固定 n 的取值，算法实现简单，而且能确保对不同大小的目标所需的处理时间恒定，因为双线性插值的算法时间复杂度只与缩放结果的尺寸有关。若采用第二种方式，能够保证缩放后的结果与原始图像块最接近，甚至可以为长宽分别计算最合适的长度，令 $T_1 \times T_2$ 的矩形框与原目标框最相近。但这么做增加了算法的复杂度，并且使得算法的处理时间不确定，不利于满足实时性需求。综合考虑后，固定 n 的取值，即对任意大小的目标都缩放到固定的尺寸是一个较好的方案。

由于存在图像的缩放，从特征图中计算得到的目标位置偏移量 δx，δy 并不是实际目标的位移量。记图像块在行列方向的缩放系数为 γ_w 和 γ_h，满足：

$$\begin{cases} M / \gamma_h = T \\ N / \gamma_w = T \end{cases} \tag{3-41}$$

则目标在原图中的位移量 δx_r，δy_r 可以表示为：

$$\begin{cases} \delta x_r = \gamma_w L \delta x \\ \delta y_r = \gamma_h L \delta y \end{cases} \tag{3-42}$$

利用公式(3-42)可以修正由图像缩放和特征提取带来的估计偏差。

选取合适的 n 的值也很重要，若 n 的值较小，缩放后的图像块无法充分地表达目标与背景的信息；若 n 的值较大，则会导致计算量增加。本文算法取 $n=5$，经公式(3-40)计算得到缩放后的图像块大小为 136×136。

3.2.3 二维 FFT 实现方案

二维 FFT 可以分解为行列方向上的一维变换[45]。对 M×N 的矩阵 $f(x,y)$ 做 DFT 变换得到变换后的结果 $F(u,v)$，

$$F(u,v) = \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} f(x,y) e^{-j2\pi\left(\frac{ux}{N}+\frac{vy}{M}\right)} = \sum_{y=0}^{M-1} F'(u,y) e^{-j2\pi\frac{vy}{M}} \tag{3-43}$$

其中，$F'(u,y) = \sum_{x=0}^{N-1} f(x,y) \exp(-j2\pi ux / N)$，是 $f(x,y)$ 在 x 方向上的变换，而后由公式(3-43)完成 y 方向上的变换。二维图像数据一般在存储器中逐行存放，计算 FFT 的函数一般也要求输入数据的地址连续，因此计算二维 FFT 需要先逐行计算 FFT，

再将结果转置，使每一列的数据连续存放从而计算列 FFT，最后再次转置得到最终计算结果。

KCF 算法中的 FFT 运算主要用于加速相关运算，在图像处理中，参与相关运算的序列都是实数序列，相关运算的结果自然也是实数。针对待处理数据与处理结果的特点，本文设计了三种硬件 FFT 加速器的调用流程，分别是：R2R_FFT2d，R2C_FFT2d 和 C2R_IFFT2d，它们能够实现实数到实数，实数到复数的二维 FFT 变换与复数到实数的二维 IFFT 变换。计算过程中的中间结果转置与数据搬运工作都由 DSP 的 EDMA 外设实现，而不需要处理器内核的干预，真正做到由硬件实现完整的二维 FFT 运算。

1) R2R_FFT2d

满足一定条件的实矩阵经过 FFT 变换后仍然是实矩阵。例如一个 M×N 的矩阵 X，如果对任意的 $i \in \{1,2,3,\ldots,M-1\}$，$j \in \{1,2,3,\ldots,N-1\}$ 均满足：

$$X(i,j) = X(M-i, N-j) \tag{3-44}$$

那么 X 的二维 FFT 变换结果仍然是实矩阵。比如公式(3-20)中的样本标签值 y 和样本 x 的带核函数的自相关运算结果 k^{xx} 都满足(3-44)的条件，它们的 FFT 计算结果都是实数矩阵。

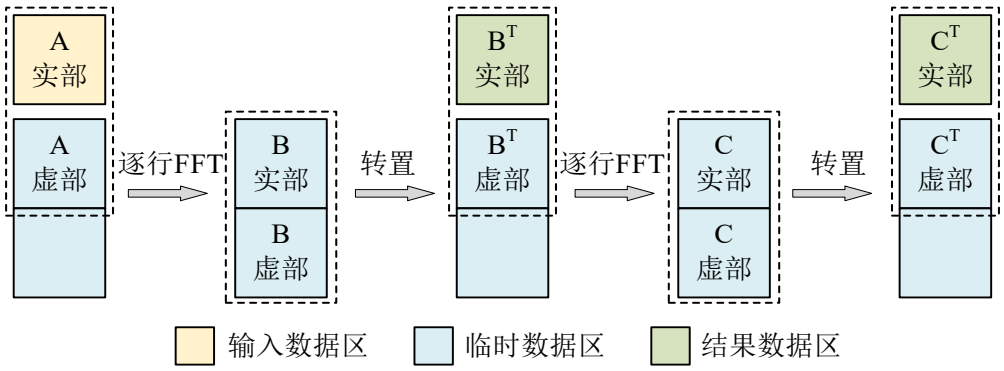


图 3-11 R2R_FFT2d 计算过程

R2R_FFT2d 的具体计算过程如图 3-11 所示。图中的输入数据区存放的是需要进行 FFT 计算的图像块；为了存放中间结果，需要额外开辟两倍于输入数据区的空间作为临时数据区。硬件 FFT 加速器支持实部虚部数据分开存储的方式，其计算结果也是实部虚部分开的，同时支持输入数据和输出结果的地址重叠。输入数据是实部，在临时数据区中填充零作为虚部，首先逐行执行 FFT 运算，其结果暂存在临时

数据区中。将结果转置后，使数据的组织形式与刚输入时一致，再次执行相同的运算并将实部结果写入到结果数据区中即完成实数到实数的二维 FFT 运算。

2) R2C_FFT2d

硬件 FFT 加速器支持实数 FFT 模式，该模式省去了加载虚部数据的过程，由硬件自动地提供全零的虚部数据。R2R_FFT2d 也可以在输入数据时采用这种模式，但是其为了统一行列 FFT 的计算形式，采用了由外部输入全零虚部的方式。

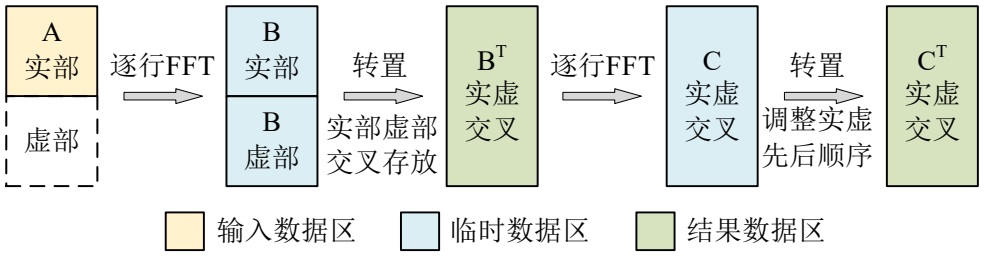


图 3-12 R2C_FFT2d 计算过程

实数到虚数的二维 FFT 是三种调用方式中最常用的一种，其计算过程如图 3-12 所示。二维 FFT 运算的结果后续还有更多的计算，实部虚部分开存储的方式不满足数据的空间局部性原则，容易导致 Cache 缓存访问缺失，因此复数以实部虚部交叉的形式存放才能更好地发挥缓存的作用。首次逐行 FFT 计算得到的结果仍然是实部与虚部分开存放，在后续的图像块转置的过程中可以同时将数据重新组织，形成实部与虚部交叉存放的形式。硬件 FFT 加速器也支持实部虚部交叉存放的数据输入，其计算结果再次转置的同时，可以根据不同的需要选择将实部数据放在低地址或者高地址。实部数据放在低地址，虚部放在高地址是大部分情况下复数在存储器中的存储方式。但是如果将虚部存放在低地址，DSP 的复数乘法指令可以只需要一条 LDDW 指令便可以将一个复数从存储器加载到寄存器中[43]。因此复数的实部与虚部存放在存储器中的先后顺序也是一个需要根据实际情况考量的问题。

3) C2R_IFFT2d

根据相关运算的结果是实数的特点，逆 FFT 的结果只需要保留实数的部分。因此每次执行二维 IFFT 运算，只需要用输入数据区一半大小的空间作为结果数据区。结果数据区大小不足以存放第一次 IFFT 计算并转置后的临时结果，为了避免过多地占用有限的存储空间，可以用输入数据区来暂存中间结果，这就导致二维 IFFT 运算

会破坏原始的输入数据，在使用的时候需要特别注意。

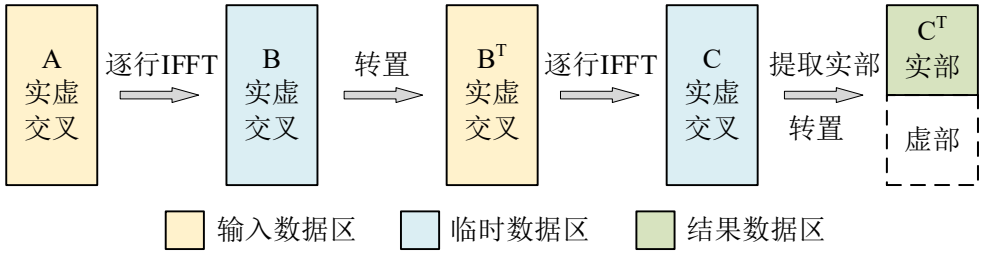


图 3-13 C2R_IFFT2d 计算过程

3.2.4 实时调用

利用 DSP 的硬件 FFT 加速器与 EDMA 实现特定的二维 FFT 运算是实现算法硬件加速的基础。还需要设计合理的硬件调用方式，使硬件在执行二维 FFT 运算时，DSP 能够同时执行其它计算任务，从而最大化算法的执行效率。

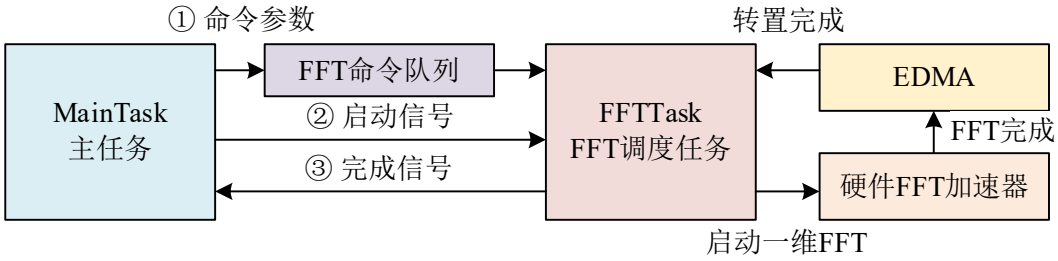


图 3-14 主任务与 FFT 调度任务间同步方式示意图

在 SYS/BIOS 实时内核的框架下，调用硬件实现二维 FFT 运算可以由一个单独的任务——FFT 调度任务负责，如图 3-14 所示。FFT 调度任务通过一个 FFT 命令队列接收命令参数，参数包括 FFT 执行的次数，FFT 的类型，源/目的数据的起始地址等等。仅利用队列无法实现任务的切换，因此还需要一个表示启动信号的信号量来实现任务间的同步。当 FFT 调度任务获得一个信号量后，就从命令队列中取出命令开始执行；如果无法获取信号量则进入休眠状态，等待新的信号量唤醒。每次二维 FFT 计算需要启动两次一维 FFT 计算，硬件 FFT 计算完成后能够自动启动 EDMA 执行事先配置好的转置与数据重组织的工作。FFT 调度任务收到两次 EDMA 完成中断即表明二维 FFT 运算完成，紧接着便向主任务发送完成信号。批量的 FFT 计算需求都可以通过 FFT 命令队列缓存，FFT 调度任务能够根据信号量的个数依次完成所有的二维 FFT 运算。

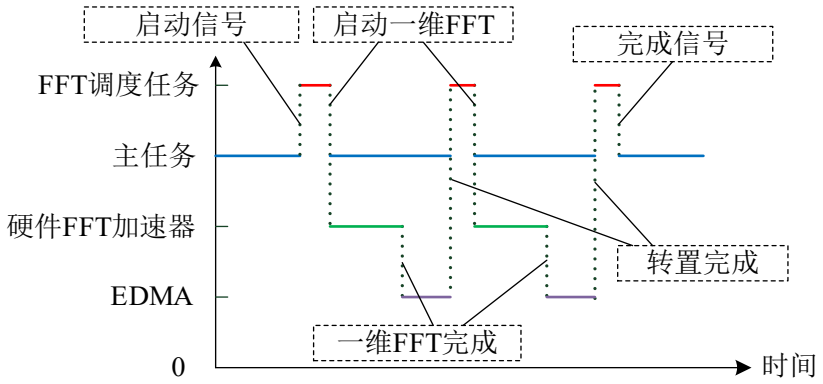


图 3-15 由硬件加速的二维 FFT 实现流程示意图

图 3-15 展示了完成二维 FFT 计算的具体流程。主任务中负责实现 KCF 算法，主任务的优先级略低于 FFT 调度任务，因此每当主任务需要启动二维 FFT 计算时，FFT 调度任务能够抢断主任务。主任务与 FFT 调度任务共享一个 DSP 内核，因此不能同时执行。FFT 调度任务在启动硬件 FFT 计算后只需要等待 EDMA 转置完成的中断，并在完成二维 FFT 计算后向主任务发送完成信号。在硬件计算的过程中，主任务仍然可以进行其它计算，在一定程度上使算法并行化。结合具体的 KCF 算法流程，在利用公式(3-34)计算 k^{xx} 时，需要计算多通道特征图的二范数平方和 $\sum_{c=0}^{m-1}\|z_c\|_2^2$，还需要同时计算多通道特征图的离散傅里叶变换 $\hat{z}_c$，此时便可利用硬件来连续完成多次二维 FFT 计算，由软件计算平方和，实现软硬件协同工作。此外，每当针对一张图像的训练完成时，可以直接启动对更新后的特征图的 FFT 计算，而不去关心它是否计算完成。直到新的一帧图像到来，需要用到其结果时再去统计 FFT 调度任务产生的完成信号个数，这样做也能隐藏部分用于 FFT 计算的时间。

3.3 实时目标跟踪

3.3.1 软件系统

根据 2.3.6 中规划的融合信息处理系统软件框架，DSP 中的内核 1 与内核 2 需要同时分别对红外目标与辅助目标进行跟踪。因此本文在融合信息处理系统软件框架的基础上实现了针对两个不同的目标同时采用 KCF 算法进行跟踪的功能。

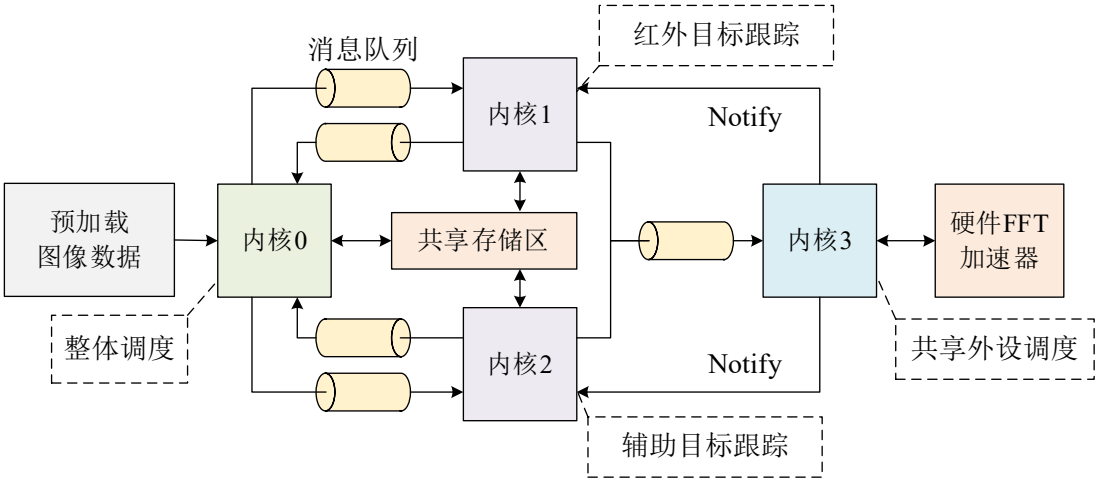


图 3-16 红外目标与辅助目标同时跟踪软件系统框图

用于红外目标与辅助目标同时跟踪的软件系统框图如图 3-16 所示。系统中由内核 0 负责整体调度。它利用乒乓缓冲区接收图像序列，图像接收缓冲区位于 DDR3 共享存储区中，可以由不同的内核同时读取原始图像数据。内核 1 与内核 2 分别负责跟踪两个不同的目标，它们与内核 0 之间分别建立双向的消息队列，用于传输启动命令与跟踪结果。当内核 0 收到一帧新的图像后，它分别启动内核 1 与内核 2 执行 KCF 目标跟踪算法，并等待它们反馈跟踪结果。内核 3 专门用于管理一些共享外设资源，例如硬件 FFT 加速器。内核 1 与内核 2 执行 KCF 算法需要调用硬件 FFT 加速器时，可以通过消息队列将 FFT 运算的相关参数发送给内核 3，由内核 3 来直接调用硬件 FFT 加速器完成运算。运算完成后，内核 3 通过 Notify 告知对应的内核运算完成。虽然共享硬件资源也可以由 GateMP 来管理，但是那样做就需要每个内核都能够响应硬件 FFT 的完成中断，并且屏蔽非自身调用产生的中断，这样做不如由单个内核独占式地管理共享硬件外设来得便捷。未来如果需要引入更多的共享硬件外设，也同样可以由内核 3 负责管理。

3.3.2 存储空间分配

DSP 内有限的存储资源需要合理地分配才能实现较高的效率。DSP 内的存储器有一定的层级关系。L1D 和 L1P 是最贴近内核的存储器，它们都只有 32 kB，一般都作为 Cache 使用，缓存内核对 L2 SRAM 的访问。每个内核都有 512 kB 的 L2 存储空间，可以部分配置为一般的 SRAM 或者 Cache，如果配置为 Cache，那么它能够缓存

内核对 DDR3 DRAM 的访问。如果程序中有部分数据或代码存放在 DDR3 DRAM 中，将部分 L2 存储空间配置为 Cache 能够有效提升内核对 DDR3 DRAM 的访问速度。整个片内共有 4 MB MSMC SRAM，内核对 MSMC SRAM 的访问速度与 L2 相当，如何利用这 4 MB 宝贵的存储空间是在程序设计之初就需要明确的问题。

表 3-2 实时目标跟踪系统存储器空间配置

内核	L2 Cache	L2 SRAM	MSMC SRAM	DDR3 DRAM
内核 0	256 kB	256 kB	256 kB	128 MB
内核 1	128 kB	384 kB	1.5 MB	128 MB
内核 2	128 kB	384 kB	1.5 MB	128 MB
内核 3	256 kB	256 kB	256 kB	128 MB
共享	/	/	512 kB	1 GB
已用总量/利用率	/	/	4 MB/100%	1.5 GB/75%

实时目标跟踪系统中对存储空间的配置如表 3-2 所示。每个核中用 L2 SRAM 存放程序代码、中断向量表与系统堆栈。由于内核 1 与内核 2 所执行的跟踪算法较为复杂，256 kB 的 L2 SRAM 略显不足，而且由于几乎不涉及对 DDR3 DRAM 的访问，可以适当减小 L2 Cache 的大小至 128 kB，将 L2 剩余的空间都用作 SRAM。系统中 MSMC SRAM 的 512kB 用于核间通信，其余空间按照需求分配给不同的内核作为数据区。内核 1 与内核 2 实现 KCF 算法需要较大的堆区用于存放特征提取过程中的临时结果，因此为它们分别分配了 1.5 MB 的 MSMC SRAM。内核 0 与内核 3 不涉及复杂的运算，因此对数据区的需求不大，只需要为它们分配 256 kB 即可满足需求。目前仅将 DDR3 DRAM 中的一小部分空间用作图像数据的接收缓冲区，因为高速的 MSMC SRAM 已经能够满足数据处理的需求，若后续需要使用更多的 DSP 内核或者需要处理更大规模的数据，则可将数据区移至 DDR3 DRAM 中。DDR3 中的数据访问经过 L2 Cache 缓存也能够实现接近 L2 的访问速率。

3.3.3 优先调度策略

为了对同时跟踪两个目标的正确性与实时性进行评估，采用一组 10 帧图像组成的图像序列进行测试。首先由调试器为每个内核加载各自的程序，程序加载完成后，每个核的程序指针都暂停在主函数的入口。然后利用调试器将 10 帧图像加载到内核 0 的 DDR3 DRAM 数据区，内核 0 启动 EDMA 将数据逐帧搬运到多核共享的 DDR3

DRAM 中的乒乓接收缓冲区，搬运数据的频率由内核 0 来控制，模拟实际工作中的数据接收过程。这 10 帧图像中的跟踪目标已事先利用标准的 KCF 算法跟踪处理，得到一个用于参考的结果，系统中的两个内核对同样的目标进行跟踪，其跟踪结果若与参考结果一致则能初步证明跟踪结果正确；由于采用乒乓缓存，每当系统对两个目标都开始跟踪时，就可以启动新的一帧图像的接收，DSP 通过记录每次启动数据接收的时刻来计算单帧图像的处理时间。

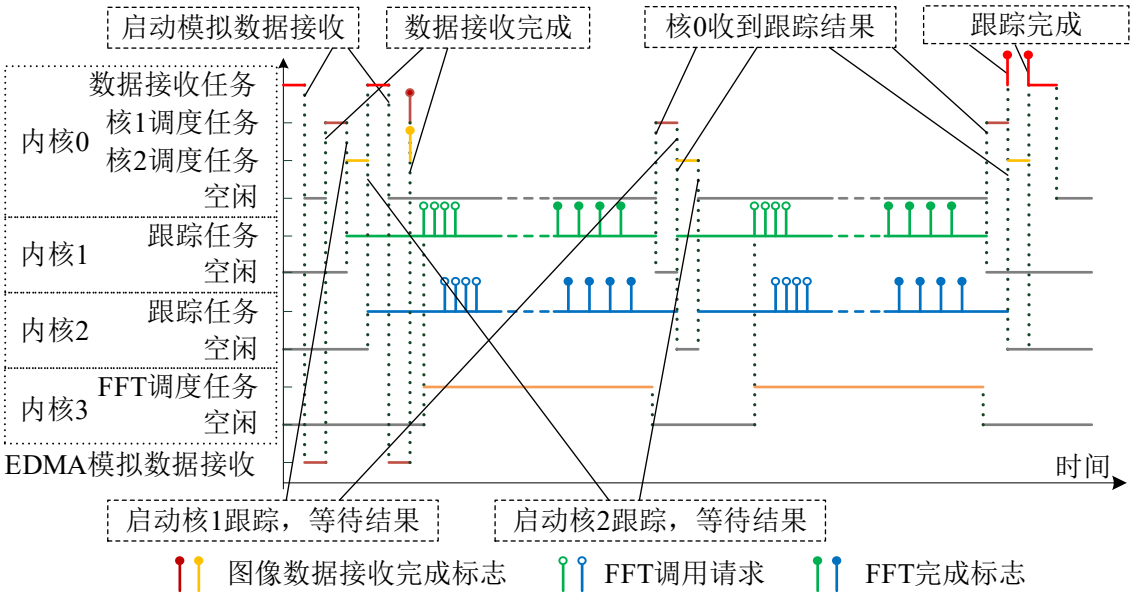


图 3-17 未采用优先调度的任务执行流程示意图

为了对具体的流程做更进一步的说明，图 3-17 以连续处理两帧图像为例，给出了不同内核中各个任务的执行流程示意图。内核 0 中安排了三个任务，它们的优先级从高到低分别是：数据接收任务、核 1 调度任务和核 2 调度任务。数据接收任务主要负责启动 EDMA 搬运图像数据至接收缓冲区，模拟图像数据的接收过程。除了最开始搬运的一帧图像以外，后续图像均在两个核都开始执行目标跟踪时启动搬运。程序运行之初，两个核很快都开始运行，所以连续两帧图像数据很快都搬运到了接收缓冲区中。由于图 3-17 中仅展示了两帧图像的处理流程，因此数据接收任务在完成两帧图像数据接收后不再模拟接收新的数据。

内核 0 中的核 1 调度任务与核 2 调度任务分别负责启动内核 1 与内核 2 的跟踪任务。调度任务只有在图像数据准备就绪，并且接收到上一帧的处理结果后才会向对

应的核发送启动的命令。在第一帧图像刚接收完成时，调度任务马上就能够启动跟踪任务，而在跟踪任务正在执行的过程中，新的数据虽然接收完成，但调度任务需要在收到上一帧的跟踪结果后才能启动新的一帧的处理。当序列中的图像都处理完成后，调度任务还会将跟踪结果打印输出。

两个内核的跟踪任务实现流程完全一致，因此它们很可能在同一时刻密集地通过内核 3 调用硬件 FFT 加速器。在没有设置优先级的情况下，消息队列中的 FFT 命令按照进入队列的先后顺序依次被内核 3 取出并执行。假如某一时刻内核 1 与内核 2 都在等待一批数据的 FFT 运算结果，而内核 3 却在交替执行两个内核的 FFT 运算命令，这使得它们相比于单独执行时，都需要额外等待更多的时间。

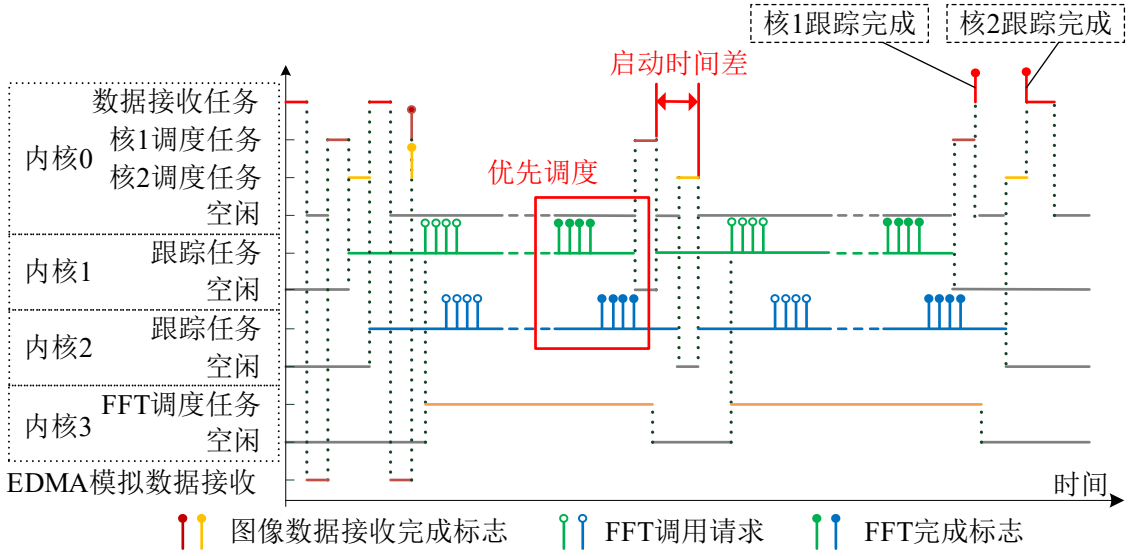


图 3-18 采用优先调度后的任务执行流程示意图

为了减少两个内核在同时调用硬件 FFT 时的额外等待时间，需要对 FFT 命令队列引入优先调度的机制，使两个内核先后调用硬件 FFT 加速器。可以将内核 1 调用硬件 FFT 的命令设置为更高的优先级，内核 3 优先执行内核 1 的 FFT 命令请求。采用优先调度后的任务执行流程示意图如图 3-18 所示。尽管最开始时两个内核同时开始执行跟踪任务，但是在处理完第一帧图像后，两个内核启动跟踪任务的时刻自然地产生了时间差，避免了同时相互等待的情况。两个内核虽然执行相同的跟踪算法，但它们所需的时间由于 FFT 命令的优先调度会产生不同，这也是为什么需要在内核 0 中利用两个调度任务分别对两个内核进行调度的原因。

3.3.4 测试结果与分析

相关实验环境及源码详见附录 A.3 ，测试采用的图像为 640×512 大小的红外灰度图像。由于调试器速率的限制，只能通过调试器加载少量图像用于测试，测试结果如表 3-3 所示。表中数据是关于三组实验的结果，包括单核单目标 KCF 目标跟踪与是否采用优先调度策略优化的多核双目标 KCF 跟踪。三组实验中均对同一目标跟踪，它们的跟踪结果完全一致，包括目标框左上角坐标(x,y)，目标框的宽度 width 和高度 height。

表 3-3 实时目标跟踪测试结果

帧数	x	y	width	height	单核（ms）	未优化（ms）	优化后（ms）
1	352	253	20	20	6.24	8.05	9.57
2	352	255	20	20	12.58	16.16	13.74
3	357	256	20	20	12.59	16.20	13.29
4	354	255	20	20	12.59	16.19	13.32
5	348	252	20	20	12.59	16.16	13.28
6	348	249	20	20	12.58	16.15	13.28
7	347	245	20	20	12.58	16.14	13.28
8	335	245	20	20	12.58	16.19	13.27
9	340	249	20	20	12.58	/	/
10	332	249	20	20	12.58	/	/

实时目标跟踪实验中，由于在两个内核刚开始处理第 9 帧时，内核 0 就已经开始了最后一帧图像的搬运，而程序仅记录了内核 0 开始搬运图像数据的时刻，因此无法计算最后两帧图像的处理时间。三组实验中第一帧图像均用于样本训练，因此处理第一帧图像所需的时间都较少。

对比单核单目标跟踪所用的时间与未采用优先调度策略优化的多核双目标跟踪所用的时间，双目标跟踪所需的时间明显增加，而且每帧图像处理所用的时间基本都是均匀的。观察优化后的双目标跟踪所用的时间，在样本训练阶段，由于额外的优先级仲裁，其所需的时间比未优化的跟踪方案更久；但是随后的第二帧图像，两个内核调用硬件 FFT 不再集中在同一时刻，DSP 处理一帧图像所需的时间有了明显减少。在随后的结果中可以观察到，由多核 DSP 实现的双目标跟踪相比于单个 DSP 内核实现的单目标跟踪，仅需额外不到 1 ms 的时间，即可完成相当于原本两倍计算量的工作。

3.4 本章小结

本章首先详细介绍了 KCF 算法的基本原理与实现细节，为后续将算法移植到 DSP 上奠定了基础。接着结合 KCF 算法的特点，提出了三种特定的二维 FFT 计算方式，包括实数到实数，实数到复数的二维 FFT 变换和复数到实数的二维 FFT 逆变换。这三种计算方式都可以采用 DSP 片内的硬件 FFT 加速器与 EDMA 实现。在 SYS/BIOS 实时内核的框架下，设计了专门的二维 FFT 调度任务，从而实现快速的批量二维 FFT 运算，有效地加速了 KCF 目标跟踪算法的执行。

红外激光融合的目标检测跟踪算法方案中需要对红外目标与辅助目标同时执行 KCF 跟踪算法，因此本章在最后基于融合信息处理系统的软件框架，实现了双目标实时跟踪的功能。利用优先调度策略，使硬件 FFT 加速器在不同内核间高效共享。在保证跟踪结果正确性的前提下，DSP 完成一帧图像中的双目标跟踪仅需 13 ms，相比于单个内核运行 KCF 目标跟踪算法，仅额外增加了不到 1 ms 的时间。

4 算法硬件加速框架与验证平台

算法硬件加速框架是指一种 FPGA 内部逻辑电路的设计框架，基于此框架，DSP 可以方便地调用 FPGA 中的硬件加速电路，从而对 DSP 执行的算法进行加速。算法验证平台是由 DSP 软件，FPGA 硬件逻辑电路和上位机软件三部分构成，DSP 上运行的目标跟踪算法可以通过上位机软件快速地完成验证与评估。

算法硬件加速框架具有可扩展性强，数据传输效率高和软件编程便捷三个特点。它主要由 AXI 交换结构，SRIO 收发控制器和 EMIF 与 AXI4-Lite 转换器三部分组成。AXI 交换结构使用户设计的硬件加速电路可以方便地接入 AXI 总线，形成多主多从的总线结构；SRIO 收发控制器使 DSP 内核无需关心数据收发过程，实现了 DSP 与 FPGA 之间的高速数据传输；EMIF 与 AXI4-Lite 转换器使得 DSP 仅需少量的存储器读写操作即可对硬件加速电路的寄存器进行配置，从而完成硬件加速电路调用。

算法验证平台基于公开数据集 UAV123 与 OTB100 进行设计，主要包括数据集验证平台中测试脚本的编写，SRIO 与 PCIe 桥接器的设计，DSP 与上位机软件的同步。测试脚本的编写使得算法验证过程能够直接嵌入原有的数据集验证平台中，用户可以通过 MATLAB 自动完成对 DSP 上跟踪算法性能的完整评估；SRIO 与 PCIe 桥接器的设计实现了 DSP 与上位机之间的流式数据传输，建立起高效的数据传输通路；DSP 与上位机软件以约定的方式进行同步，多组图像序列的测试可以连续依次进行，算法的跟踪结果与执行的时间开销都能够送回上位机用于分析。

4.1 算法硬件加速框架

4.1.1 设计思路

在传统的 DSP 与 FPGA 构成的数字信号处理系统中，通常由外部存储器访问接口（External Memory Interface, EMIF）实现 DSP 与 FPGA 之间的数据交互。DSP 通过 EMIF 接口将 FPGA 作为外部存储器访问，其访问速度受限于时钟频率，往往很难达到很高的速度。时钟频率的限制一方面由于随着时钟频率的提高，板上并行信号线之间的串扰也会更加严重，容易导致数据出错；另一方面，板上布线延时的增加使

得电路设计人员需要确保一组并行信号中的每根线的长度都保持一致；除此之外，FPGA 对输入信号的频率也有限制，种种原因导致用 EMIF 在 DSP 与 FPGA 之间传输批量数据，已经逐渐满足不了现如今的需求。只有当 FPGA 中设计的硬件加速电路能在很大程度上减轻软件计算的负担的情况下，这部分用于数据传输的时间开销才能够忽略不计。或者在 FPGA 设计中采用流式处理的方式，这能够起到一定的加速作用，但也会使得 FPGA 中的电路设计更加复杂，调试更加困难，同时由于 EMIF 接口不能同时用于读写数据，在 DSP 软件设计时也需要灵活地在发送数据与接收结果的过程中来回切换。总之，DSP 与 FPGA 之间的数据传输瓶颈是影响 FPGA 硬件加速器实际性能的关键因素，设计一个稳健高效的 FPGA 硬件加速框架刻不容缓。

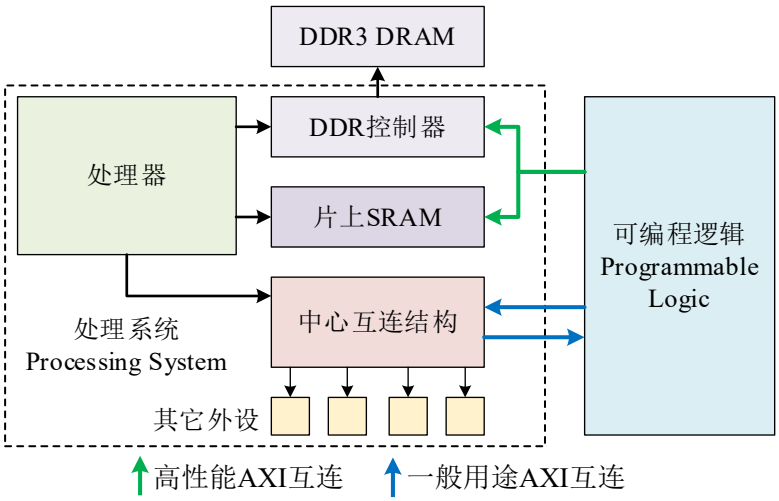


图 4-1 Zynq-7000 片上系统结构示意图

Xilinx 于 2010 年推出的 Zynq-7000 系列片上系统同时包含处理系统（Processing System, PS）与可编程逻辑（Programmable Logic, PL），类似于 DSP 与 FPGA 构成的信息处理系统。其内部结构示意图如图 4-1 所示，其中省略了许多细节，主要突出 PS 与 PL 之间的交互，其设计有很大的借鉴意义。

PS 与 PL 之间的互连采用 AXI4 协议，AXI4 协议是由 ARM 公司提出的用于片内系统互连的第四代高级可扩展接口（Advanced eXtensible Interface, AXI）协议，具有高带宽，低延迟的特点。AXI4 协议又能够细分为三种接口，分别是 AXI4 Memory Mapped（AXI4-MM），AXI4-Lite 和 AXI4-Stream。AXI4-MM 有时也简称为 AXI4，AXI4 与 AXI4-Lite 用于特定地址的数据访问，它们的区别仅在于 AXI4 额外支持数

据突发传输；AXI4-Stream 用于流式数据传输，不需要指定数据的地址。AXI4 与 AXI4-Lite 都有 5 个独立的通道组成，分别是读写地址通道（AR/AW），读写数据通道（R/W）和响应通道（B）。

图 4-1 中表示 AXI 互连的箭头的含义是：箭头的起点是主端设备，箭头的终点是从端设备，而不是指数据的流向，在后续的图示中也延续这一约定。AXI4 与 AXI4-Lite 的主端设备能够利用 AW 和 W 通道向从端设备写数据，同时也能够利用 AR 与 R 通道从从端设备读数据。图中的高性能 AXI 互连相比于一般用途的 AXI 互连有着更高的时钟频率与数据带宽。处理器只能通过一般用途的 AXI 互连主动访问 PL 中用户设计的硬件外设，这在大部分情况下只是用于控制寄存器的读写；而高速数据的传输仅能由 PL 端主动访问片上或片外存储器来实现。这一设计使得 PS 想利用 PL 实现一些加速功能时，仅需要配置一些寄存器启动加速电路，然后等待结果即可，PL 会负责主动读取数据并将处理结果写回指定的存储器空间。

4.1.2 框架可扩展性

受到 Zynq-7000 片上系统设计思想的启发，在 DSP 与 FPGA 构成的处理系统中也可以设计一套类似的 FPGA 硬件加速框架。在这一框架下，DSP 的内核可以像 Zynq 中的 PS 一样无需关心数据的传输过程，而只需要配置寄存器来启动加速电路，随后等待硬件处理完成的中断信号即可。DSP 不能再将 FPGA 作为被动的访问对象，而需要由 FPGA 掌控数据传输的主导权。

银河飞腾 FT-M6678 DSP 提供有两组 SRIO 接口，每组接口中有 4 对串行差分链路，每对链路支持 3.125Gbps 的传输速率，能够提供每秒千兆字节的传输速率，DSP 与 FPGA 双方都能够主动发起读写请求，能够从根本上解决 DSP 与 FPGA 之间的数据传输速率受限的问题。

数据传输的问题解决后，剩下的就是控制命令如何发送的问题。DSP 向 FPGA 发送控制命令一般不需要很快，所以有很多种方案可以选择。DSP 提供有 SPI，I2C，UART 接口用于访问外设，而 FPGA 内一般将电路模块的控制接口设计成 AXI4-Lite 接口，可以在 FPGA 内设计相应的接口转换电路来实现 DSP 对 FPGA 内模块的控制。虽然方案有很多，但不同的方案，设计的复杂程度也不同。AXI4-Lite 主端接口需要

实现特定地址的读写功能。比如 SPI 与 UART 接口本身没有地址的概念，做接口转换时需要额外约定确定寄存器地址的方式；I2C 协议虽然有寄存器地址的概念，但是设计接口转换电路也需要额外去约定设备地址与寄存器地址；DSP 的 EMIF 接口本身就支持特定地址读写，因此将 EMIF 接口转为 AXI4-Lite 接口是最方便的。

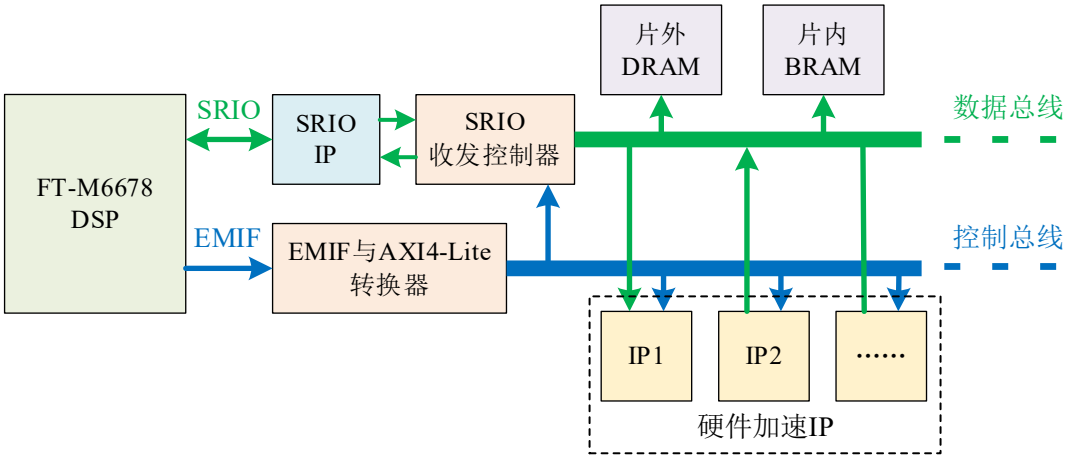


图 4-2 FPGA 硬件加速框架

综合以上考虑后，新的 FPGA 硬件加速框架如图 4-2 所示。框架内主要有一条数据总线与一条控制总线，两条总线都是由 Xilinx 提供的 AXI 交换结构实现的。FPGA 内的特定功能的电路模块称为 IP（Intellectual Property），用户设计的 IP 只需要遵循相应的 AXI 协议规范即可方便地接入总线。控制总线采用 AXI4-Lite 协议规范，数据总线采用 AXI4-MM 协议规范。总线上的主端设备和从端设备数量都最多支持 16 个，因此这一框架具有非常强的可扩展性。

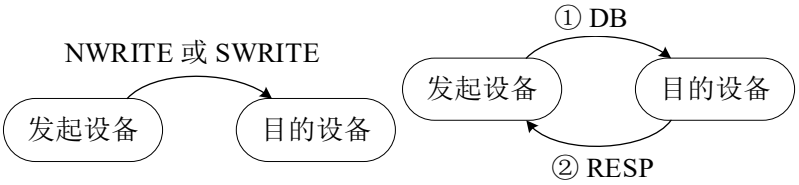
根据需求的不同，硬件加速 IP 的 AXI4-MM 接口可以设计为主端形式或者从端形式。比如图 4-2 中的 IP1 采用从端形式的接口，其它主端可以将数据直接写入 IP1 的内部存储区，或者从 IP1 的内部存储区中读取数据；IP2 采用主端形式的接口，它可以直接从 FPGA 片内或者片外的存储器中读写数据。DSP 对数据总线的访问需要经由 SRIO 收发控制器完成，DSP 通过读写 SRIO 收发控制器的寄存器可以启动数据从 FPGA 向 DSP 的传输；DSP 经过 SRIO 向 FPGA 发送的数据也会由 SRIO 收发控制器解析，并写入指定的地址空间。

DSP 对控制总线的访问需要经由 EMIF 与 AXI4-Lite 转换器来完成，其中包括 DSP 对 SRIO 收发控制器的操作也都是通过 EMIF 接口。FPGA 内的硬件加速 IP 都

需要设计一个从端 AXI4-Lite 接口用于用户控制与状态查询。

4.1.3 高速数据传输

DSP 与 FPGA 之间的高速数据传输得益于两者之间的 SRIO 链路互连。SRIO 是一种基于事务包（Transaction Packet）传输的系统级互连协议，可以实现点对点的对等通信，具有高带宽，低功耗，引脚数少，可扩展性强等特点。按照由高到低的实现层次，SRIO 可以划分为逻辑层，传输层与物理层，用户一般只需要关心逻辑层的设计而不需要关心中间的传输层与物理层的具体实现。



(a) 写事务　　(b) 门铃事务

图 4-3 SRIO 部分事务发送与响应流程

SRIO 支持多种不同的事务，而在目前的设计中主要涉及图 4-3 中的几种事务。为了实现数据从 DSP 到 FPGA 的搬运，虽然可以从 FPGA 发起读事务，读取 DSP 端的待处理的数据，但是相比于用 DSP 内部的 DMA 发起 NWRITE 事务直接发送，读的效率更低。因此在 FPGA 端只需要实现 NWRITE 事务包的接收，而不需要主动发起读请求。若是需要从 FPGA 向 DSP 发送数据，则只需要由 FPGA 发起 SWRITE 事务包，数据能够直接写入 DSP 的片内或片外 RAM 中。NWRITE 与 SWRITE 事务类似，都没有响应包；门铃事务 DB 则需要有响应包 RESP。

Xilinx 提供的 SRIO IP 支持 2.2 版本的 RapidIO 互连协议规范，最高支持 4 对链路，单条链路的速率最高为 6.25Gbps，受限于 DSP 对 SRIO 速率的支持，目前能够采用的单条链路速率为 3.125Gbps，全链路速率为 12.5 Gbps。SRIO 物理层采用 8b/10b 编码，因此理论上数据传输的最高速率为 1.25 GB/s。又因为串行链路传输中需要额外传输一些控制符号，实际数据传输速率必定略小于该理论值，但接近 1 GB/s 的数据传输速率以及能够满足绝大部分数据传输需求。

SRIO IP 在 SRIO 逻辑层协议规范的基础上对事务包的格式再次进行封装，采用 HELLO（Header Encoded Logical Layer Optimized）包格式，使用户更方便使用。IP

将事务包的收发分散到四个 AXI4-Stream 形式的通道上进行管理，如图 4-4 所示。四个通道的名称均以“I”或“T”开头，代表发起方（Initiator）或者接收方（Target）；而后紧跟的“REQ”或“RESP”代表该接口上的信号是属于请求（Requst）还是属于响应（Response）。例如用户可以在 TREQ 通道接收到来自 DSP 的 NWRITE 事务包，也可以从 IREQ 通道主动发送 SWRITE 事务包。

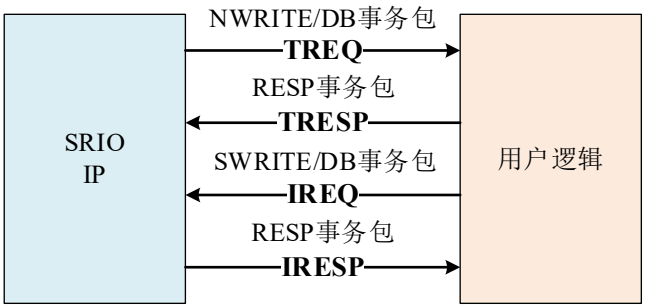


图 4-4 SRIO IP 的使用

在 Xilinx 提供的 SRIO IP 基础上，设计 SRIO 收发控制器，其接口定义如图 4-5 所示。可以通过控制器的从端 AXI4-Lite 接口控制 DB 与 SWRITE 事务的发送以及获取接收到的门铃信息；它将接收到的数据通过 AXI4-MM 接口的写通道写入指定地址，当需要发送数据时，也通过 AXI4-MM 接口的读通道从指定地址读取数据。

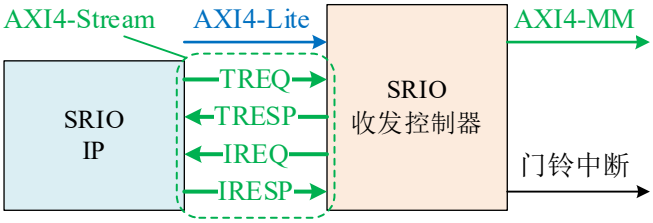


图 4-5 SRIO 收发控制器接口定义

由于数据的接收只与 TREQ，TRESP，AXI4-MM 的写通道有关，数据的发送只与 IREQ，IRESP 和 AXI4-MM 的读通道有关，因此数据的接收与发送可以分别独立设计，整体状态机如图 4-6 所示。SRIO 收发控制器的基本功能只需要按照状态转移所示进行设计，很容易就能够实现。AXI4 协议中每个通道都有 Valid 与 Ready 信号，只有当两个信号同时有效时数据才算有效握手传输，因此主从两端都能够对数据流的速率进行控制，设计的灵活性很高。但是这一设计目前还不够完善，比如系统状态的指示，错误参数的检测，寄存器具体功能的定义等都没有明确的说明。在后续的

4.2.2 中，SRIO 收发控制器将结合 PCIe 进行改进，并给出一个更加完整的设计。

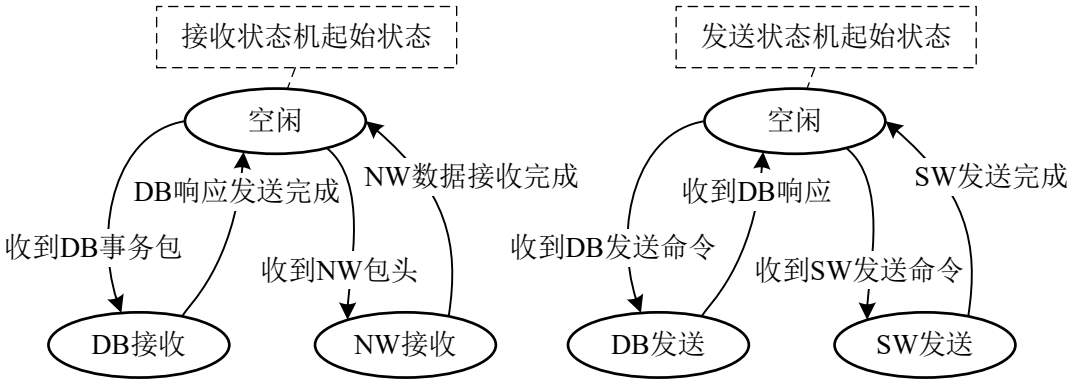


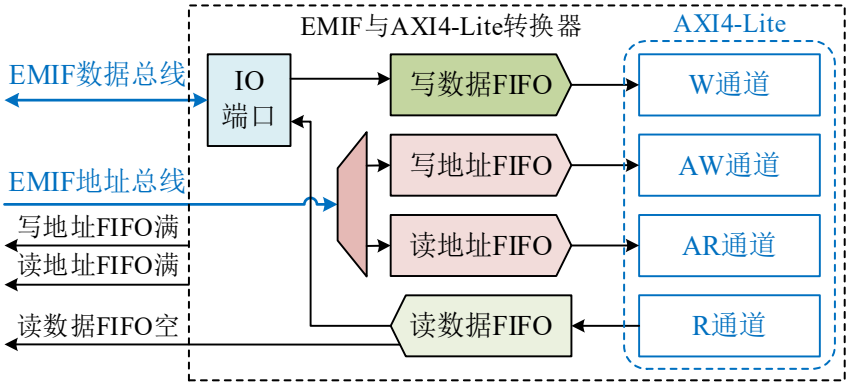
图 4-6 SRIO 收发控制器状态转移图

4.1.4 加速电路调用

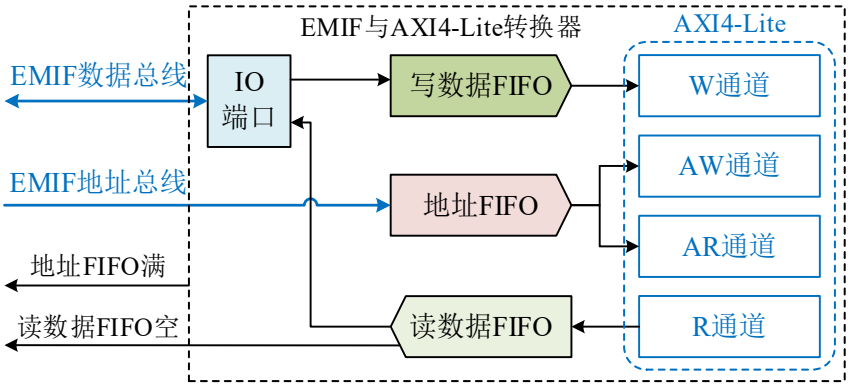
DSP 调用硬件加速电路主要依赖于 EMIF 与 AXI4-Lite 转换器，硬件加速 IP 的参数配置都可以由 DSP 的存储器读写操作完成，软件编程简单。DSP 与 FPGA 之间的数据传输都已经由 SRIO 收发控制器管理，因此 DSP 只需要等待硬件加速 IP 处理完成的中断信号，即可在本地存储器中获取硬件加速 IP 的运算结果。

EMIF 接口转换为 AXI4-Lite 接口的具体设计方案有多种选择，但整体的设计方案是类似的。首先考虑到 AXI4-Lite 的信号需要握手，所以读写命令都采用 FIFO 缓冲才能确保命令不遗漏。其次 AXI4-Lite 的读写通道是分开的，因此可以采用读写独立的地址 FIFO，如图 4-7 (a)所示。但在遇到某些读写顺序有严格要求的情况下，DSP 软件的设计应避免在 EMIF 总线上连续产生多个读写交替的请求，因为读写地址 FIFO 独立后，读写命令的前后顺序无法得到保证。考虑到 EMIF 总线不会同时产生读写信号，所以也可以采用地址 FIFO 合并的方案，如图 4-7 (b)所示，这样做就能够保证 DSP 的读写请求是按顺序完成的。

EMIF 的数据总线是双向的，需要在 FPGA 的 IO 端口上利用三态缓冲器控制 IO 的方向。三态缓冲器的输出在一般情况下保持高阻态，由输入缓冲器接收来自外部的数据；当 EMIF 总线的读信号有效时，三态缓冲器切换为正常输出。



(a) 地址 FIFO 独立的设计方案



(b) 地址 FIFO 合并的设计方案

图 4-7 EMIF 与 AXI4-Lite 转换器设计方案

接口转换的主要问题在于从 AXI4-Lite 接口读回数据的时刻不确定。EMIF 不论是以同步还是异步方式访问存储器，总能在发起读命令后的一定时间内获得数据。但是通过 AXI4-Lite 的 AR 通道发送读地址后，R 通道的数据 Valid 信号并不确定在何时有效。因此由 EMIF 读数据需要分两个步骤完成，第一步，发送读地址；第二步，在数据有效后读取结果。这两个步骤可以用一个 EMIF 的读命令来完成，每次 EMIF 发起读请求，表示发送读地址，同时读取上次读请求的地址的数据。但这一做法会使 AXI4-Lite 每次额外多读一次数据，这在大部分情况不会有问题，但多余的操作可能会引起意想不到的错误。因此如果能设置一种专门用于向 AR 通道发送读地址的方式，可能更加稳妥。可以在每次发送读请求时用一根额外的 GPIO 信号或 EMIF 的高位地址线来区分具体是发送地址还是仅读取数据，甚至也可以利用高位地址线将写请求分为正常的写命令与发送读地址。总之设计的方案很多，每种方案之间大同小异。“不管黑猫白猫，抓住老鼠就是好猫”，只要能够实现接口转换的功能，并且 DSP

软件设计时留心需要注意的地方，实现软硬件的紧密配合就是可行的设计方案。

本设计中采用的是利用写命令结合高位地址偏移量发送读地址的方案，读写地址共用一个 FIFO，在每次发送读写命令前需要确保地址 FIFO 非满，读取数据前需要判断读数据 FIFO 非空。读写寄存器的伪代码如下：

```
1  // 写寄存器
2  void writeReg(
3      volatile Uint32 *addr,
4      Uint32 val
5  ){
6      while(地址 FIFO 满信号有效);
7      *(addr) = val;
8  }
```

```
1  // 读寄存器
2  Uint32 readReg(
3      volatile Uint32 *addr
4  ){
5      while(地址 FIFO 满信号有效);
6      *((Uint32 *)(((Uint32)addr) +offset)) = 0;
7      while(数据 FIFO 空信号有效);
8      return *(addr);
9  }
```

典型的硬件加速电路调用流程如图 4-8 所示。FPGA 内设计了一个硬件加速 IP，它能够主动访问片外 DRAM，从中读取待处理的数据并将处理结果写回指定地址的存储区。DSP 对该硬件加速 IP 的调用只涉及少量的寄存器读写操作。

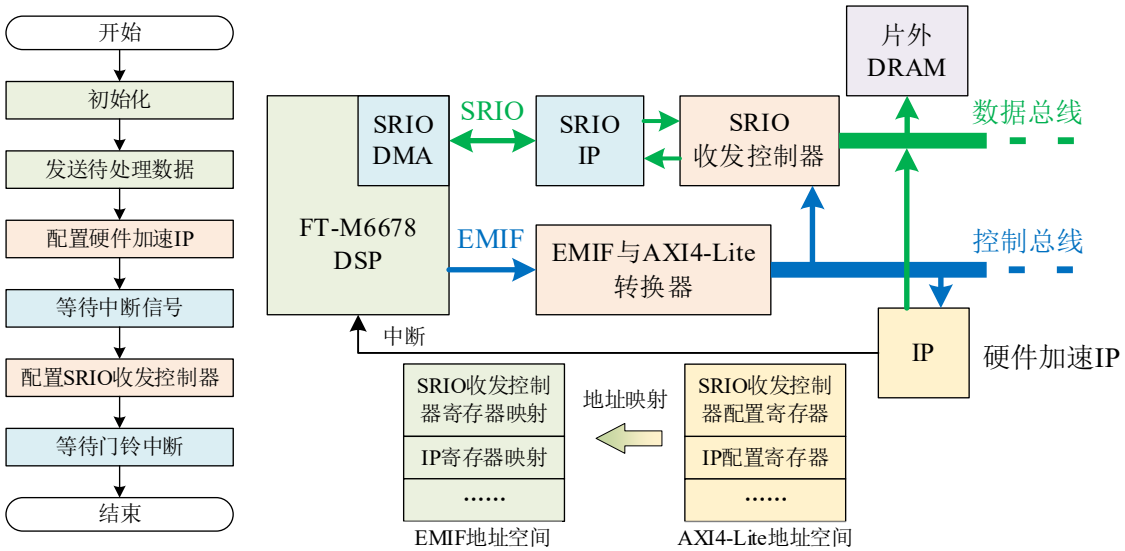


图 4-8 硬件加速电路调用流程举例

DSP 首先需要完成 SRIO 链路的初始化，配置 EMIF 接口，与 FPGA 建立连接。接着通过配置相关寄存器启动 SRIO DMA 将待处理数据发送至 FPGA 的片外 DRAM 中。数据发送完成后，通过 EMIF 接口配置硬件加速 IP 启动工作，并等待加速电路运算完成的中断信号。一旦收到中断信号，则再次通过 EMIF 接口启动 SRIO 收发控

制器向 DSP 发送处理后的结果。DSP 接收完成结果数据后会收到门铃事务中断，标志着硬件加速 IP 的处理结果已经存放在 DSP 可以直接访问的存储区中。

在整个 DSP 调用硬件加速 IP 的过程中，DSP 软件的操作就如同通过 EMIF 接口访问片外存储器一样。EMIF 与 AXI4-Lite 转换器将 FPGA 内硬件加速 IP 的寄存器配置空间映射到 DSP 的 EMIF 地址空间，使 DSP 与 FPGA 紧密结合，为 DSP 软件编程提供了极大的便利。

4.2 算法验证平台

目前已有许多关于视觉目标跟踪算法嵌入式实现的公开文献，它们大多只对实现方案本身做了详细的介绍，而缺少对实际跟踪性能的全面评估。测试结果中也往往只有特定的几个序列进行展示，这些序列可能直接来自于摄像头，跟踪的结果只能通过人来主观判断，而没有统一的参考结果。这么做的原因可能是在嵌入式系统中，大家更加关心系统的实时性，而系统实际的跟踪准确率并不是重点，因为算法本身可能已经在高性能的个人计算机（Personal Computer, PC）中实现并评估了其跟踪性能。另一方面，用于评估视觉目标跟踪算法的性能的平台如 OTB100，UAV123，LaSOT 等数据集，VOT 视觉目标跟踪挑战赛，都是建立在 MATLAB 或 Python 应用软件的基础上。嵌入式系统中缺少这类应用软件，因此无法直接对跟踪器的准确率做直接的评估，一般需要写一个模拟嵌入式平台跟踪结果的软件模型来间接评估其性能。这样得到的结果能够在一定程度上全面地反映嵌入式平台上的跟踪器在不同光照，不同背景，不同目标大小时的跟踪性能，但结果的可靠性取决于软件模型的准确性。为了方便快速地对算法的跟踪性能进行全面评估，在融合信息处理系统中建立算法验证平台是一个更好的选择。平台的建立能够省去软件模拟的过程，直接将数据集的测试序列下发至系统，并获取最直接的跟踪结果，用于算法验证与评估。

4.2.1 数据集测试平台

OTB100 与 UAV123 都是基于 MATLAB 的视觉目标跟踪数据集，每种跟踪器都有一个对应的跟踪器运行脚本，负责调用目标跟踪算法并保存跟踪结果。由于 MATLAB 无法直接通过硬件驱动程序实现上位机与 DSP 之间的数据传输，因此为了

实现算法的测试，需要编写相应的上位机软件与 DSP 通信。MATLAB 则通过调用上位机软件来间接完成目标跟踪算法的测试，具体的实现流程如图 4-9 所示。

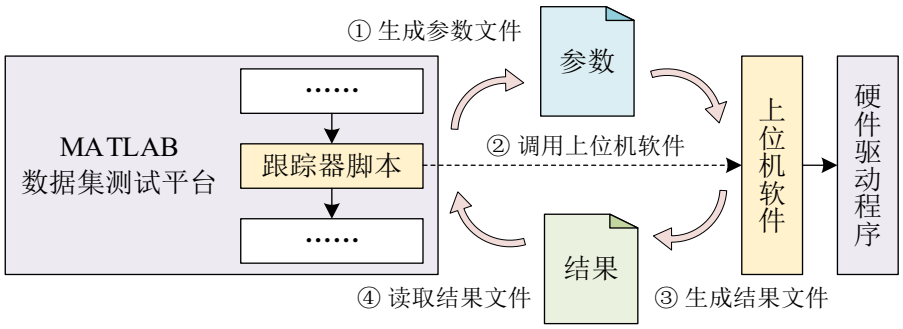


图 4-9 数据集测试平台实现跟踪器测试流程

跟踪器脚本首先根据待测序列生成参数文件，文件中包括初始目标框位置，序列帧数和序列图片路径等信息。随后 MATLAB 调用上位机软件，软件根据参数信息向 DSP 逐帧发送图像数据，并保存 DSP 传回的跟踪结果。最后跟踪器脚本读取结果文件，将结果保存为统一的格式，完成一组序列的测试。下一组序列的测试只需要改变参数文件的内容，重复同样的流程即可实现。

4.2.2 SRIO 与 PCIe 桥接器

基于本文所提出的融合信息处理系统，已有 PCIe 链路连接 FPGA 与上位机。Xilinx 提供的 IP“DMA/Bridge Subsystem for PCI Express”，简称 XDMA，用于在 FPGA 上实现 PCIe 接口，使用简单，配置灵活。

表 4-1 不同版本 PCIe 标准支持的链路速率

PCIe 标准	链路速率（Gbps）	x16 链路总带宽（Gbps）	年份
1.0	2.5	32	2003
2.0	5.0	64	2007
3.0	8.0	126	2010
4.0	16.0	252	2017
5.0	32.0	504	2019
6.0	64.0	1024	2021

PCIe 协议是 PCI 并行总线的扩展，按照由高到低的实现层次可以分为应用层，事务层，数据链路层和物理层。PCIe 协议支持的链路速率随着版本的提升不断提高，不同版本的协议支持的链路速率如表 4-1 所示。目前最新的协议规范支持单条链路 64.0 Gbps，Xilinx 提供的 PCIe IP 最高支持 PCIe 3.0，即单条链路速率 8.0 Gbps。

提供 PCIe 插槽的 PC 主机一般称为根复合体（Root Complex, RC），外接设备称为端点（End Point, EP）设备。在系统刚启动的时候，RC 能够根据 EP 的配置空间获取设备信息并进行相应的配置。其中，基址寄存器（Base Address Register, BAR）是 RC 用来确定 EP 中可用来访问的数据空间大小的寄存器。BAR 中低位 0 的个数表示数据空间的大小，这几位是在硬件上固定的，RC 无法改写。比如 BAR 的初始值是“0xFFFF0000”，它的低位有 16 个 0，表明这个 BAR 对应的数据空间大小是 64 kB。RC 确定了数据空间大小之后，再对高位的“1”进行改写，实现地址映射。比如改写成“0x68000000”，那么从 0x68000000 开始的 64 kB 范围就是这个 BAR 对应的数据空间。PCIe 配置空间中 MSI（Message Signaled Interrupt）相关的寄存器也需要 RC 在系统刚启动的时候进行配置。配置完成后，EP 可以根据配置信息向 RC 的特定地址写特定的数据来发送中断。

Xilinx 的 XDMA IP 为用户在 FPGA 上实现 PCIe 接口提供了很大的便利。这一 IP 的主要功能是基于 PCIe 链路实现 DMA 数据搬运。DMA 的通道数量与接口形式都是可以配置的。最高支持 4 对 DMA 通道，从板卡发往主机的通道称为 C2H（Card to Host），相反方向的通道称为 H2C（Host to Card）。若在 FPGA 端采用 AXI4-MM 形式的接口，即用于指定地址的数据传输，则不论通道数有多少，只提供一个 AXI4-MM 接口，其内部对通道的使用进行了管理；若采用 AXI4-Stream 形式的接口，即用于不指定地址的数据传输，则 IP 会对外提供每条通道的接口，由用户决定每条通道的具体用途。

表 4-2 XDMA 提供的可配置的 BAR 寄存器

BAR 寄存器	大小	用途
BAR0（可选）	用户设置	由 AXI4-Lite 访问的数据空间
BAR1	64 kB	DMA 配置寄存器
BAR2（可选）	用户设置	DMA 旁路，由 AXI4-MM 访问的数据空间

除了 DMA 的功能以外，XDMA 还提供了可选的 AXI4-Lite 主端接口与 DMA 旁路的 AXI4-MM 主端接口，如表 4-2 所示。AXI4-Lite 主端接口可以使上位机直接控制 FPGA 上设计的 IP，比如 SRIO 收发控制器可以由上位机直接配置并启动 SRIO 事务的发送；DMA 旁路的 AXI4-MM 主端接口可以使上位机直接访问 FPGA 片内或片

外的存储器，就像在访问上位机本地的数据空间一样方便。

Xilinx 在提供 IP 的同时，也附带提供了 Linux 操作系统下的设备驱动源码。在安装设备驱动后，XDMA 的不同 BAR 映射为上位机中的几个文件，设备驱动将用户对这些文件的读写操作转换为内核态中的寄存器配置与数据读写。在仅采用一对 XDMA 通道时，用户对文件 xdma0_h2c_0 写数据，设备驱动会配置相应的寄存器启动 DMA 从上位机下发数据；同样地，用户从 xdma0_c2h_0 读数据，相当于启动 DMA 从 FPGA 上传数据。xdma0_user 与 xdma0_bypass 两个文件对应的数据空间可以直接用 mmap 函数映射到用户的虚拟内存空间，使用户对特定地址的读写能够直接转换为 XDMA 对 FPGA 片内或片上存储器的访问。另外还有以 xdma0_events 开头命名的文件，用户可以根据读这些文件的结果来判断是否有 MSI 中断产生。

为了建立上位机与 DSP 之间的数据传输通路，可以在 SRIO 收发控制器的基础上引入 XDMA 的 H2C 与 C2H 通道，构成 SRIO 与 PCIe 桥接器，实现数据在 DSP 与上位机之间的流式传输。

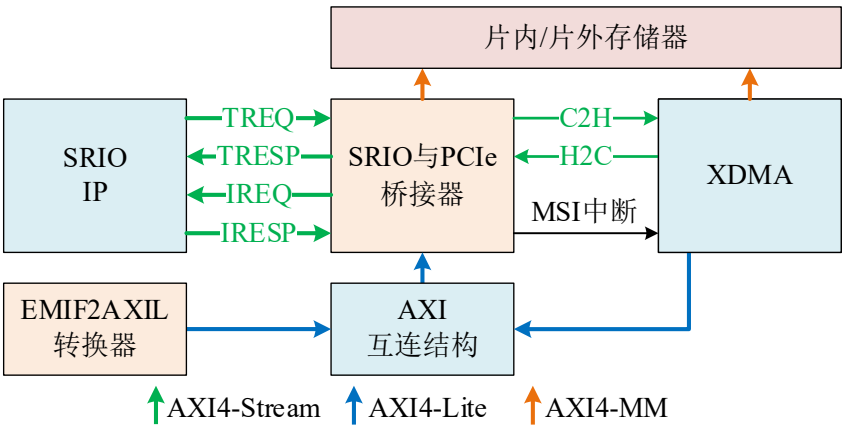


图 4-10 SRIO 与 PCIe 桥接器接口定义

SRIO 与 PCIe 桥接器对外的的接口形式如图 4-10 所示。其接口延续了 SRIO 收发控制器原有的形式，包括与 SRIO IP 之间的 4 条 AXI4-Stream 通道，用于控制的从端 AXI4-Lite 接口和用于主动访问存储器的 AXI4-MM 主端接口。不同的是新增加了两条与 XDMA 互连的 AXI4-Stream 通道和 MSI 中断请求信号，用于数据收发与中断发送；另外，寄存器的功能定义也更加丰富，而且能够实现更加灵活多样的数据传输。新的寄存器定义如表 4-3 所示，详细的寄存器位域信息见附录 A.2

表 4-3 SRIO 与 PCIe 桥接器寄存器定义

寄存器偏移地址	名称	说明
0x00	IDENTIFIER	标识寄存器
0x04	SRIO_CSR（Control and Status Register）	SRIO 控制与状态寄存器
0x08	SRIO_MODE	DB/SW/NW 的工作模式
0x0C	SW_SIZE	SW 的数据量
0x10	SW_DST（SW Destination Address）	SW 的目的地址
0x14	SW_SRC（SW Source Address）	SW 的源地址
0x18	DB_TXINFO（Doorbell Tx Information）	用于发送的门铃信息
0x1C	MSI_CSR	MSI 控制与状态寄存器
0x20~0x5C	MSI_INFO_n (n = 0~15)	用于发送的 MSI 信息

SRIO 与 PCIe 桥接器的实现不仅能够建立起上位机与 DSP 之间的数据传输通路，同时也使得 DSP，FPGA 与上位机三者之间的数据与控制信号的传输变得更加灵活。图 4-11 中给出了三者两两之间的通信方式。数据传输主要依赖 SRIO 的 NWRITE 与 SWRITE 事务，以下简称 NW 与 SW。桥接器为 NW 和 SW 分别设置了两种模式，本地模式与转发模式。本地模式下，NW 与 SW 事务均是 DSP 与 FPGA 之间的交互；转发模式则是由 FPGA 将 NW 事务接收的数据转发至 C2H 通道或者从 H2C 通道将数据组成 SW 事务包并向 DSP 转发。如果上位机想直接访问 FPGA 上的数据，可以直接通过 DMA 旁路的 AXI4-MM 主端接口进行读写访问。

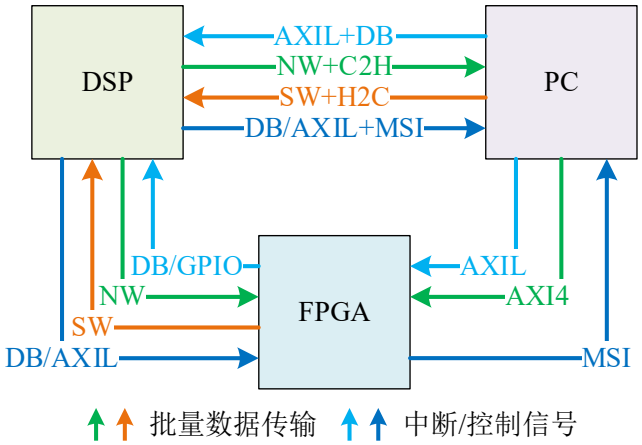


图 4-11 系统内部数据/中断传输方式

面对大批量原始激光数据的下发需求，不光可以采用 DMA 旁路的方式，还可以通过多启用 XDMA 的一条 DMA 通道来实现，如图 4-12 所示。新增加的 H2C_1 通道专门用于激光数据的传输，激光成像电路对原始激光距离像多帧累积后的结果可

以由 SRIO 与 PCIe 桥接器读取并发送至 DSP。

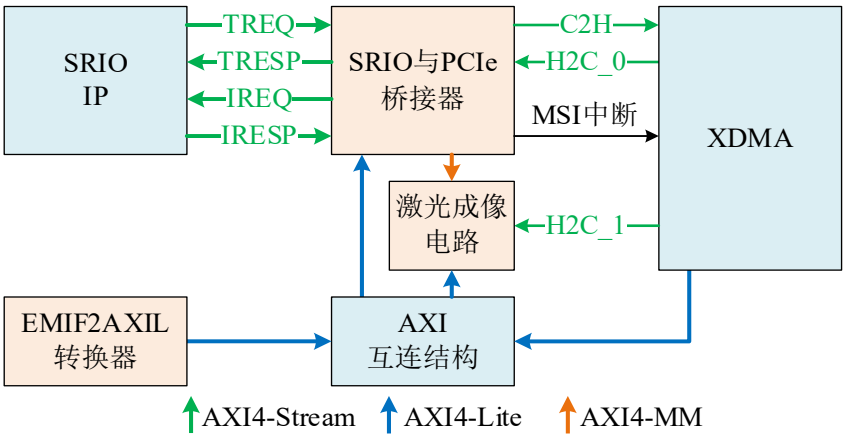


图 4-12 用于红外激光融合信息处理系统的 FPGA 内部结构框图

DSP 接收中断的方式主要有两种，一种是通过桥接器的 AXI4-Lite 接口启动 DB 事务的发送，DSP 能够收到门铃中断；另一种则是通过 FPGA 与 DSP 之间的 IO 互连产生 DSP 上的 GPIO 中断。DSP 也可以向 FPGA 发送门铃中断，并且桥接器可以在收到门铃中断后自动触发 MSI 中断的发送，使 DSP 能够更加快捷地向 PC 发送中断请求。这一功能可以由用户选择性的开启或关闭，有时仅需要向 FPGA 发送中断信号时，则不需要开启这一功能。

由于 DSP 与 PC 都有访问 FPGA 内 IP 的 AXI4-Lite 接口，所有需要通过访问寄存器实现的功能，DSP 与 PC 都能够做到。因此不管是发送 DB 中断还是 MSI 中断，不管是切换工作模式还是查看 IP 的状态，DSP 与 PC 都有主动权。这一特点使得整个系统的灵活性大大提高，DSP 既可以主动向 PC 请求数据，也可以由 PC 主动将数据下发，在不同的需求场景下，用户可以选择最合适的实现方式。

XDMA 支持发送多种不同的 MSI 中断，但在 Linux 操作系统中，仅支持一个中断向量，即不同的 MSI 中断都需要映射到同一个中断向量上。那么上位机就需要通过读取 MSI_INFO 寄存器查询中断信息来确定具体是产生了哪个 MSI 中断。在桥接器的设计中，对 MSI 中断的产生与处理采用了图 4-13 所示的方式。MSI 中断信号产生后，需要持续保持高电平，直到上位机读取了 MSI_INFO 寄存器之后才会重新将中断信号拉低。

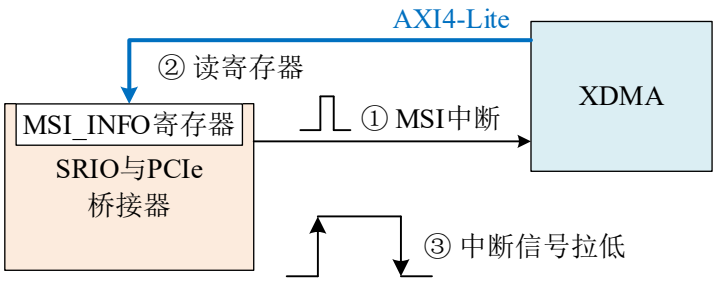


图 4-13 MSI 中断的产生与处理

为了使系统更稳健，桥接器的设计中 SRIO_CSR 与 MSI_CSR 是用于控制 SRIO 事务发送或 MSI 中断发送的寄存器。它们在设计过程中都将启动信号跳变的上升沿作为启动的标识，并在事务发送或中断未响应的过程中始终保持高电平，防止新的请求再次产生。如果有新的请求错误地产生，则相应的错误状态标志位将置一。除此之外，NW 与 SW 事务的传输对地址对齐有着严格的要求，而且 AXI4 协议不支持跨越 4 kB 边界的突发传输，因此在启动 SW 的时刻与收到 NW 事务包头的时刻，硬件都会对数据传输的地址进行检查。一旦出现错误，CSR 寄存器中的相应标志位也会置一，并且放弃错误的传输请求，随后用户可以通过读取 CSR 寄存器来查找问题。

4.2.3 DSP 与上位机软件同步

DSP 与上位机之间的同步方式如图 4-14 所示，主要由上位机启动序列测试，在测试过程中由 DSP 主动请求数据。

在启动测试之前，DSP 软件首先完成硬件外设的初始化，比如 DSP 与 FPGA 之间的链路初始化，硬件 FFT 加速器初始化，系统中断初始化等等。DSP 完成主要的初始化工作后便进入等待上位机下发启动信号的状态。

MATLAB 在启动上位机软件运行后，上位机软件根据参数文件生成序列信息，包括序列帧数，初始目标框位置以及每帧图像的长宽。DSP 在收到启动信号后，可以稍做准备，紧接着便请求上位机发送序列信息。在收到序列信息后，DSP 需要根据图像的大小与序列长度动态分配一定的空间来作为数据接收缓冲区与结果存放空间。随后便由 DSP 主动请求图像数据，逐帧进行目标跟踪。在整个序列处理完成后，DSP 将结果发送至上位机并释放动态分配的内存空间，准备处理下一组序列。上位机软件将结果写入指定的文本文件并确保 DSP 已就绪后就能够正常退出，等待

MATLAB 下一次调用。

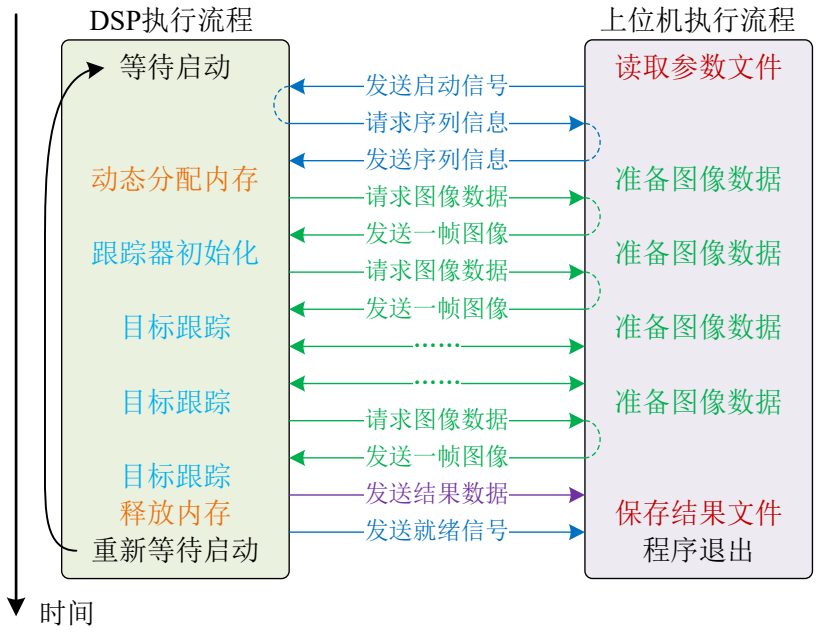


图 4-14 DSP 与上位机软件同步方式

算法执行时间由 DSP 负责统计，统计的时间中不包括数据传输的耗时，DSP 将处理一组序列所用的时间求和并随同跟踪结果一起发送至上位机。

4.2.4 数据传输速率测试

在设计完成后，需要对其实际的性能进行测试。SRIO 与 PCIe 桥接器实现了 DSP 与上位机之间的流式数据传输。在测试系统中，上位机与 FPGA 之间采用 8 条 PCIe 链路相连，每条链路速率为 5.0 Gbps，全链路速率为 40 Gbps。PCIe 的物理层也采用 8b/10b 编码，理论上的最高数据传输速率为 4 GB/s，大于 DSP 与 FPGA 之间的理论最大速率 1.25 GB/s。因此实际有效数据传输的速率必定小于 1.25 GB/s。

DSP 的 TSC 寄存器可以用于统计实际数据传输的用时。TSC 寄存器中的值按照 DSP 的主时钟频率不断增加，前后两次读取 TSC 寄存器中的值并计算两者的差值可以得到以 DSP 系统周期数为单位的准确时间长度。测试系统中 DSP 采用 1 GHz 的主时钟频率，因此用于统计时间的精度为 1 ns。

为了全面且快捷地对 DSP 与 PC 间的数据传输速率进行分析，测试流程中采用数据发送并读回校验的形式，并由 DSP 负责整个测试流程的控制，如图 4-15 所示。

首先由 DSP 在其片外的 DDR3 DRAM 中开辟数据空间并生成测试数据，DSP 通过 SRIO 的 DB 事务与 PCIe 的 MSI 中断告知 PC 启动 C2H 传输，紧接着也发送 NW 事务，并由桥接器完成数据转发。PC 在收到数据后会向 DSP 发送中断信号，DSP 收到中断的时刻标志着数据发送完成。而后 DSP 再次请求 PC 启动 H2C 传输，将刚刚发送过去的数据再传回 DSP。DSP 等待数据接收完成中断，并记录从发起请求到收到中断的时间作为接收数据的总时间。在数据接收完成后，DSP 需要对收到的数据进行校验，确保数据收发正确。单次传输的数据量以字节为单位，都是 2 的整数次幂，从最小 256 字节到到最大 8 兆字节共 16 种不同的数据量。每轮测试中，DSP 从收发较小的数据量开始，逐渐增大数据量，完成对不同数据量的收发测试，最终保存 16 组测试结果并将其发送至 PC。相同的测试流程多次执行，并对结果的均值与标准差进行分析，避免实验的偶然性。

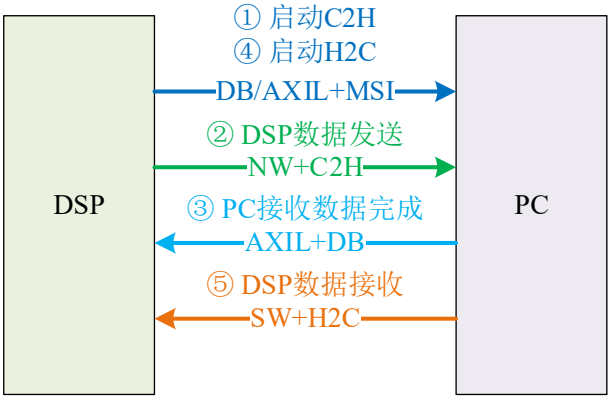


图 4-15 DSP 数据收发速率测试流程

相关实验环境及源码详见附录 A.3 。在数据传输速率测试的过程中，一些额外的时间开销，比如 DSP 与 FPGA 之间利用中断进行同步，软件调用函数时的上下文切换，相关外设的参数配置等，可以看作是固定的。而后随着数据量的增加，数据传输所需的时间也应当随着数据量线性增长，因此可以利用线性模型来拟合 DSP 收发数据量与数据传输时间的关系。将数据量用 x 表示，相应的数据传输时间为 y，则：

$$y = kx + b \tag{4-1}$$

其中 k 表示传输单位数据量需要的时间，习惯上用 $v = 1/k$ 来表示实际的数据传输速率；b 表示固有的时间开销。

实验中我们共进行 30 轮测试，计算 30 次结果的均值后利用线性模型进行拟合，结果如图 4-16 所示。图中横纵坐标系均采用对数坐标，虽然 y 与 x 呈线性关系，但是 $\log_2 y$ 与 $\log_2 x$ 并不满足线性关系，所以图中曲线表示线性拟合后的趋势线。线性拟合的结果根据最小二乘准则得到，根据拟合结果可以得出 DSP 发送数据的速率约为 0.88 GB/s，接收数据的速率约为 1.04 GB/s。相对于理论极限值，达到了 70%以上的有效带宽利用率。DSP 发送与接收数据的固有时间开销分别为 50.5 μs 与 67.0 μs 。进一步观察数据可以发现，小于 32 kB 的数据收发所用的时间与数据量并没有正相关性，这一现象可能与 DSP 内部的 SRIO 硬件实现或上位机操作系统的线程调度有关；大于 32 kB 的数据传输基本满足线性关系。因此在实际应用过程中，应当尽量整合小数据块，采用大批量数据传输，从而最大化数据传输的效率。

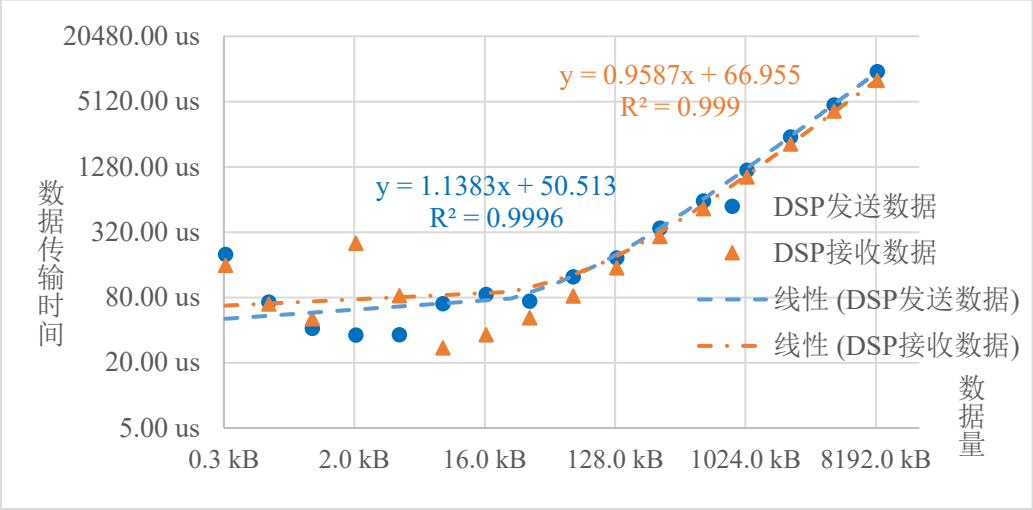


图 4-16 DSP 收发不同数据量的所需时间

除了 DSP 与 PC 间的数据传输的速率，每轮实验结果的一致性也是需要考量的指标。为了同时展现不同样本数据的波动性，首先需要对样本数据进行最大最小归一化。样本集 X 中某一样本 x 经过归一化后的结果为 x'：

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{4-2}$$

归一化后，样本的值域为[0,1]。对归一化后的样本用公式(4-3)计算 n 个样本的标准差 s 并绘制柱状图，即可得到图 4-17 中的结果。

$$s=\sqrt{\frac{\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2}{n-1}} \tag{4-3}$$

可以看到不同数据量传输所用的时间在归一化后，标准差都在 0.2 附近。这意味着如果某种数据量的传输在多次测试中，所用的最大时间为 A，多次测试的均值为 μ，那么主要的测试结果都集中在区间$[\mu-0.2A,\mu+0.2A]$范围内。因此 DSP 与 PC 之间的数据传输时间有较大的不确定性，主要原因可能是 PC 上位机所用的操作系统并不是实时操作系统，其响应无法做到很高的实时性。

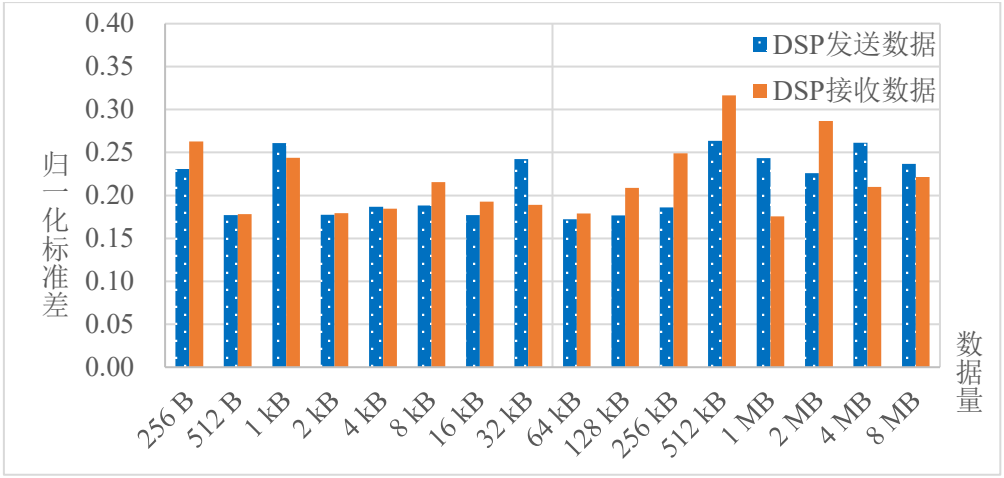


图 4-17 DSP 收发不同数据量所需时间的归一化标准差

4.3 本章小结

本章基于融合信息处理系统提出了算法硬件加速框架与算法验证平台。算法硬件加速框架具有可扩展性强，数据传输效率高和软件编程便捷三个特点。算法验证平台使用户能够利用公开的目标跟踪数据集对融合信息处理系统上运行的目标跟踪算法进行全面的验证与评估。

算法硬件加速框架主要由 AXI 交换结构，SRIO 收发控制器，EMIF 与 AXI4-Lite 转换器三部分组成。AXI 交换结构为框架提供了灵活的可扩展能力，用户设计的硬件加速 IP 可以方便地接入数据总线和控制总线；SRIO 收发控制器负责管理 DSP 与 FPGA 之间基于 SRIO 链路的数据传输，降低了 DSP 调用硬件加速 IP 的数据传输延时；EMIF 与 AXI4-Lite 转换器将硬件加速 IP 的寄存器映射到 DSP 的 EMIF 地址空

间，DSP 可以通过少量存储器访问操作完成硬件加速 IP 的调用。

算法验证平台基于公开数据集 UAV123 与 OTB100 设计，主要包括数据集验证平台中测试脚本的编写，SRIO 与 PCIe 桥接器的设计，DSP 与上位机软件的同步。测试脚本的编写使用户可以通过 MATLAB 自动完成对 DSP 上跟踪算法性能的完整评估。SRIO 与 PCIe 桥接器的设计主要在硬件加速框架的基础上引入 PCIe 链路，实现了 DSP 与上位机之间的数据流式传输，数据传输速率达到 1 GB/s 左右；DSP 与上位机软件的同步，使上位机软件能够调用 DSP 执行目标跟踪算法，并连续完成多组图像序列的测试，算法的跟踪结果与执行的时间开销都能够送回上位机用于分析。

5 算法验证与系统测试

本章基于融合信息处理系统中的算法验证平台对四种在系统中实现的KCF目标跟踪算法进行验证与评估，同时这也是对系统功能的测试。

为KCF算法引入自适应调整尺度的能力，或者设计硬件加速电路对KCF算法进行加速，可以得到四种在DSP上运行的KCF算法：尺度固定的软件执行的KCF算法，尺度固定的硬件加速的KCF算法，自适应尺度的软件执行的KCF算法和自适应尺度的硬件加速的KCF算法。基于算法验证平台对这四种算法进行验证与评估，并分析测试结果。

5.1 算法验证方案

5.1.1 测试数据集

算法测试采用OTB100与UAV123数据集，它们分别包含100个序列和123个序列。OTB100数据集中的序列图像大小不固定，从96×128的小图像到576×768的大图像都有覆盖，最长的序列有3872帧，最短的序列为71帧[46]。UAV123数据集包含的是由无人机拍摄的123个序列，每个序列的图像大小都是固定的720×1280，最长的序列为3085帧，最短的序列为109帧[47]。两个数据集中的序列信息如表 5-1所示。

表 5-1 OTB100与UAV123数据集信息

数据集	图像大小	最小帧数	最大帧数	平均帧数	总帧数
OTB100	不固定	71	3872	590.4	59040
UAV123	720×1280	109	3085	915.3	112578

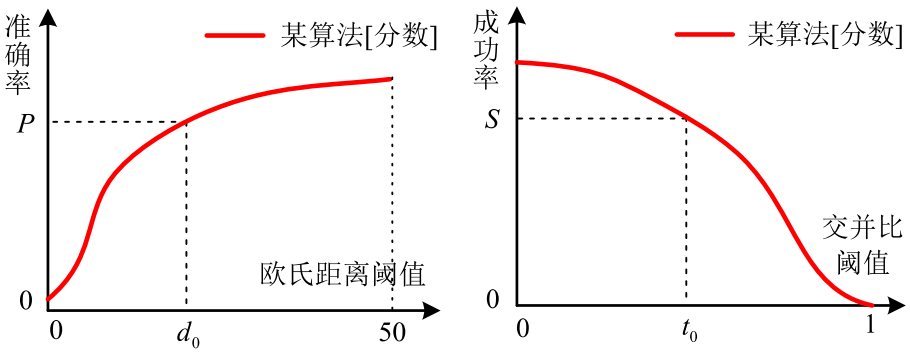
数据集中的每一帧都有对应的参考结果（Ground Truth, GT）。用数据集进行算法评估有三种测试方式，分别是OPE，TRE和SRE。单次运行评估（One-Pass Evaluation, OPE）是最常用的算法评估方式，即在序列的第一帧图像中给出目标位置，跟踪器需要在后续的图像中跟踪目标位置。但有时跟踪器可能对初始条件比较敏感，所以需要改变初始目标框的位置进行多次测试，这些测试称为时间稳定性评估（Temporally Robustness Evaluation, TRE）和空间稳定性评估（Spatial Robustness Evaluation, SRE）。

在本系统的算法测试过程中仅采用 OPE 测试。

算法的跟踪性能从准确率 P 与成功率 S 两个方面体现。跟踪结果目标框中心位置与 GT 目标框中心位置的欧氏距离 d 作为评价跟踪是否准确的标准。设置欧式距离阈值 d_0，如果 d 小于 d_0 则准确跟踪，反之跟踪失败。统计序列中准确跟踪的帧数占比得到在欧氏距离阈值 d_0 下的跟踪准确率 P。将不同 d_0 下的 P 进行绘制可以得到准确率曲线，如图 5-1 (a)所示。跟踪结果目标框与 GT 目标框的交并比 t 作为评价跟踪是否成功的标准。假设跟踪结果的目标框为 r_a，GT 目标框为 r_g，定义运算符⋂和⋃分别表示两个区域相交和相并的部分，$|\cdot|$ 表示区域内像素的个数，则：

$$t = \frac{|r_a \bigcap r_g|}{|r_a \bigcup r_g|} \tag{5-1}$$

设置交并比阈值 t_0，如果 t 大于 t_0 则表示跟踪成功。统计序列中成功跟踪的帧数占比得到在交并比阈值 t_0 下的跟踪成功率 S。将不同 t_0 下的 S 进行绘制可以得到成功率曲线，如图 5-1 (b)所示。



(a) 准确率曲线　　(b) 成功率曲线

图 5-1 算法性能评估曲线

若仅关心某一阈值下的跟踪算法准确率和成功率，可以仅按照特定阈值处的结果对算法进行排序。一般欧式距离阈值为 20，交并比阈值为 0.5。仅通过特定阈值下的准确率或成功率来比较不同算法的性能是不全面的，因此算法整体的准确率或成功率可以通过曲线下面积（Area Under Curve, AUC）来体现。典型的准确率曲线与成功率曲线绘制时，还会同时在图例中注明算法排序依据的分数——特定阈值下的准

确/成功率或曲线整体的 AUC，便于直观地比较各个算法的性能。

5.1.2 测试环境

Xilinx 提供了 XDMA 在 Linux 环境下的驱动源码，因此为了便于实现上位机软件的设计，测试过程基于 Linux 操作系统，详细的测试环境信息如表 5-2 所示。其中 MATLAB 是数据集测试所用的平台，读取图像数据依赖开源计算机视觉库 OpenCV。

表 5-2 测试环境

类型	属性
CPU 型号	Intel® Pentium® CPU G640 @ 2.80GHz
内存	DDR3 8 MB 1067 MHz
操作系统	Ubuntu 20.04.5 LTS
系统内核	5.15.0-69-generic
编译器	g++ 9.4.0 / gcc 9.4.0
依赖库	OpenCV 3.4.8
测试平台	MATLAB R2022b

5.2 算法实现与验证

5.2.1 待验证算法

传统的 KCF 算法缺少对目标尺度变化的估计，因此跟踪框的大小始终是固定的。对 KCF 算法做简单的调整就可以使算法具有尺度估计的能力，主要有两种方式，分别以 SAMF 算法[48]与 DSST 算法[49]为代表。SAMF 算法在 KCF 算法的基础上引入尺度金字塔，能够同时估计目标的位置与尺度。缺点是每引入一种尺度的估计，相当于额外进行一次目标检测的运算，其运算时间会显著增加。DSST 算法额外训练一个一维的尺度滤波器用于估计目标的尺度。在相同尺度个数的情况下，它相对于 SAMF 能够更快完成尺度的估计。但缺点是它对目标的位置估计与尺度估计是分开的，对于尺度估计的准确性依赖于位置估计的准确性。若位置估计出现偏差，则基于错误的位置进行尺度估计也很难得到正确的结果，这种情况在目标大小发生快速变化时很容易出现。

本文主要参考 SAMF 算法的做法，在不同尺度下分别进行相关运算，对目标的位置与尺度同时进行估计。为了将传统的 KCF 算法与能够自适应调整尺度的算法进行区分，称前者为 KCF_SF（Scale Fixed）算法，后者为 KCF_SA（Scale Adaptive）

算法，两者之间的主要区别如图 5-2 所示。主要区别集中在目标位置更新的过程中。KCF_SF 只需要在一个响应图中找出峰值坐标即可估计目标的位置。KCF_SA 则需要以上一帧中目标位置为中心，提取不同尺度的图像块，缩放成模板图像大小后提取多通道特征，再为每种尺度的图像块计算响应图。从多个响应图中找出峰值位置，根据峰值所在的响应图确定目标的尺度，根据峰值的二维坐标确定目标的位置。

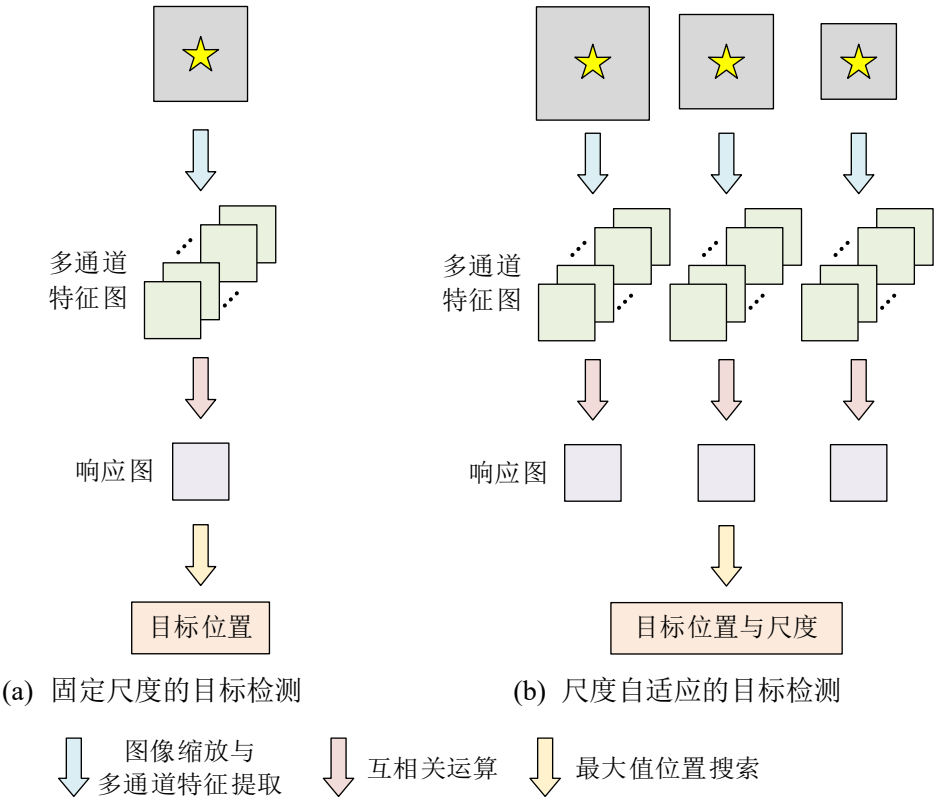


图 5-2 固定尺度与尺度自适应的 KCF 算法目标检测流程

为了能够有效判断目标尺度变化的趋势，在原有算法的基础上，至少需要额外进行两次目标检测的计算，即分别对较大的目标框和较小的目标框进行检测。3.3.4 中的测试结果表明，在 DSP 的单个内核上运行 KCF_SF 算法至少需要 12 ms，其中包括 6 ms 的目标检测时间与 6 ms 的重新训练时间。如果再加上两次目标检测的计算，那么单个内核对单帧图像的处理将至少需要 24 ms。这一结果将导致系统无法实时处理按照每秒 50 帧速率下发的红外实时图像，因此需要采用 FPGA 硬件加速电路对算法中的耗时计算环节进行加速。通过分析发现，算法中的特征提取过程是主要的耗时部分，为此课题组内的其他成员设计了 FHOG 特征提取硬件加速电路，有效地提高

了算法的实时性。为了便于区分不同的算法，将未调用 FHOG 特征提取硬件加速电路的算法名加上“SW”后缀，即表示主要由软件（Software）计算得到；将调用了加速电路的算法名加上“HW”后缀，表示算法经过硬件（Hardware）加速。最终得到四种算法分别为：KCF_SFSW，KCF_SFHW，KCF_SASW，KCF_SAHW。

5.2.2 FPGA 硬件加速电路

FHOG 特征提取硬件加速电路有一个 AXI4-Lite 从端接口，用于设置参数和启动控制；另有一个 AXI4-MM 主端接口，用于从指定地址的存储器中读取待处理图像或将结果写回指定地址的存储器中。

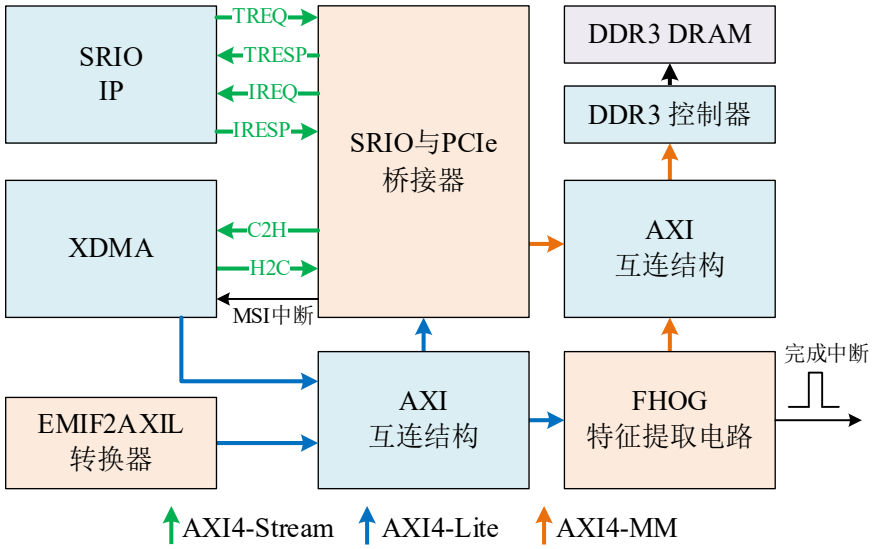


图 5-3 FPGA 内部结构框图

添加了 FHOG 特征提取硬件加速电路后的 FPGA 内部结构框图如图 5-3 所示。使用过程中 DSP 先将待处理图像数据通过 SRIO 与 PCIe 桥接器发送至 FPGA 的片外 DRAM 中，接着通过 AXI4-Lite 接口为 FHOG 特征提取电路设置参数并启动电路工作，当它完成特征提取后，结果写回片外 DRAM 中指定的地址空间并产生中断信号。DSP 收到 GPIO 中断，并再次通过桥接器读回特征提取后的结果。理论计算表明 FHOG 特征提取硬件加速电路能够将原本单个 DSP 内核计算所需的 4.8 ms 缩减到 0.8 ms，显著提高算法实时性。

由于 DSP 与 FPGA 之间采用无法抢断的单通道数据传输，每次桥接器启动 SW

事务后，数据传输无法终止，只能等待本次数据传输完成后才能启动下一次传输。因此当上位机正在通过桥接器向 DSP 发送数据时，用户就无法从 DRAM 中读取 FHOG 硬件加速电路处理后的结果，只能等待上位机的数据接收完成后才能去读取。这是硬件加速电路调用过程中需要注意的地方，也是本设计中存在的局限性。

5.2.3 FPGA 实现结果

FPGA 采用 Xilinx 的开发软件 Vivado 进行开发，软件中的框图设计如图 5-4 所示。其整体结构与图 5-3 内容一致，不同之处在于用于控制外设的 AXI 交换结构另外连接了 SRIO IP 与 DDR3 DRAM 控制器。

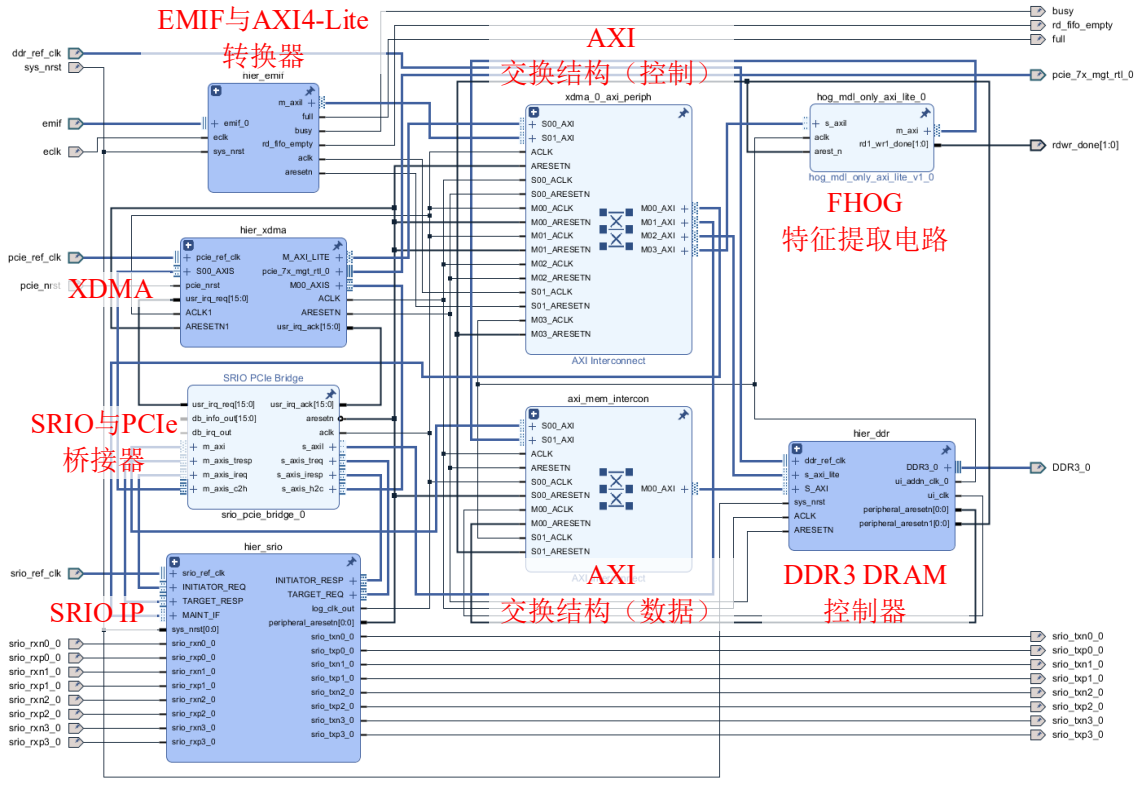


图 5-4 Vivado 软件界面框图

FPGA 内部电路经过实现后，由软件生成的资源利用与功耗分析结果如图 5-5 和图 5-6 所示。系统整体的资源使用开销少，功耗也在较低水平。系统中资源开销与功耗主要集中在 XDMA 与 DDR3 DRAM 控制器上。

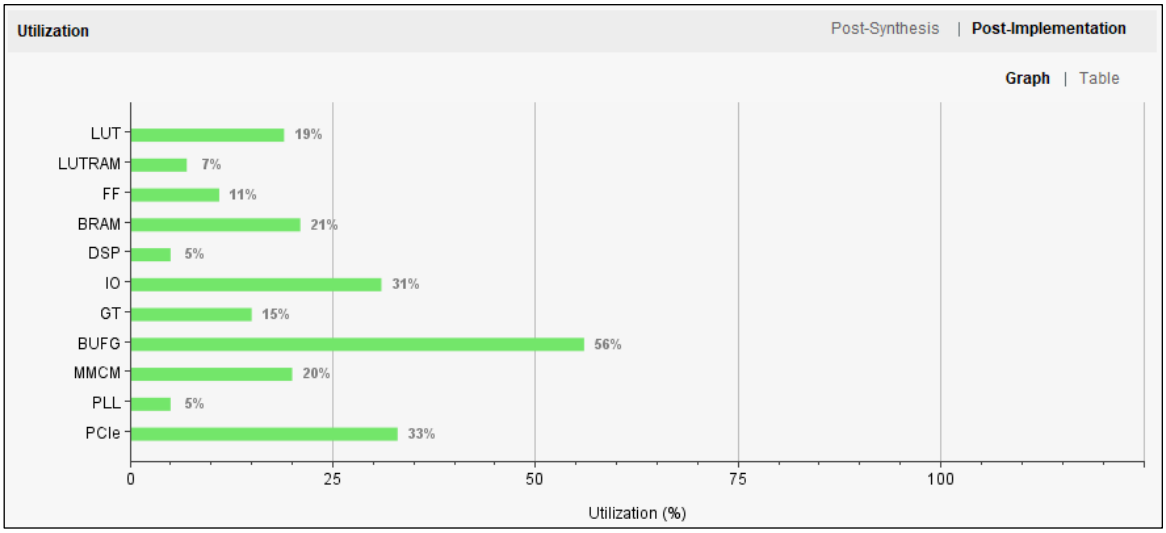


图 5-5 FPGA 整体资源开销

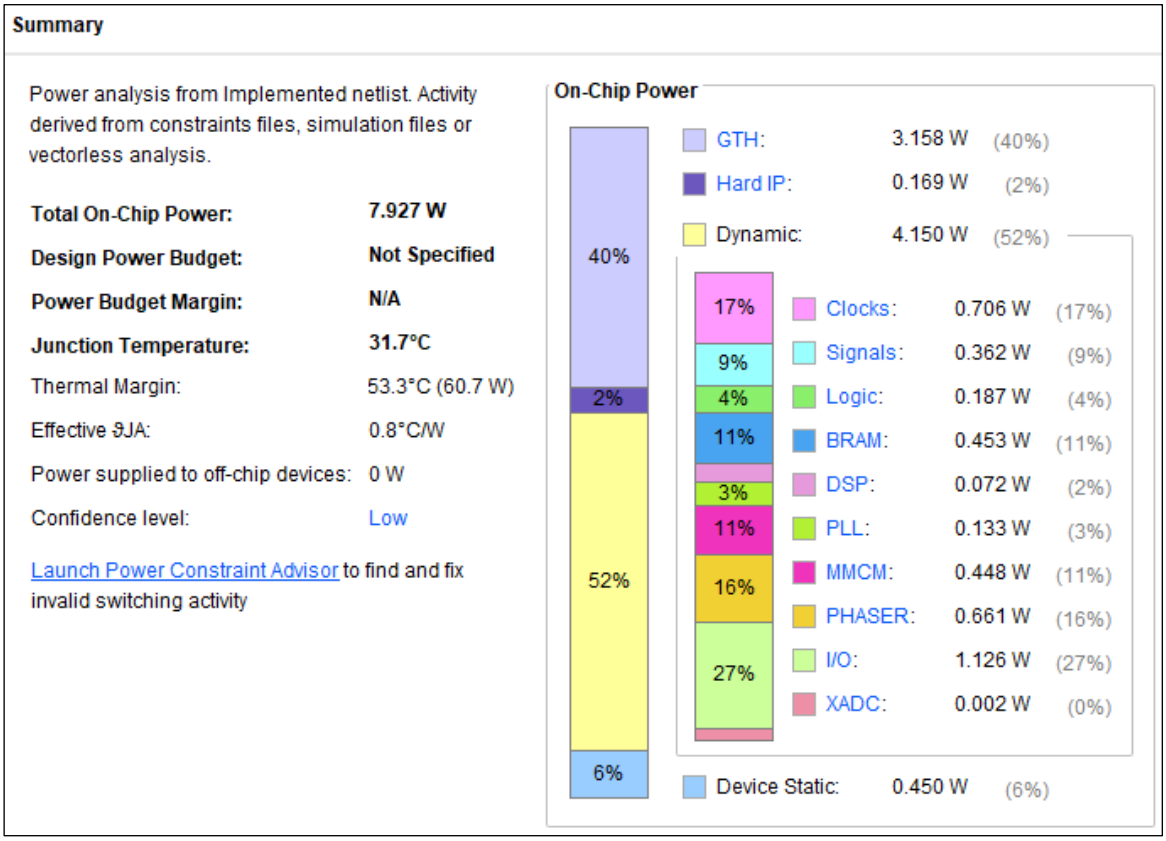


图 5-6 FPGA 整体功耗分布

5.2.4 测试结果与分析

相关实验环境及源码详见附录 A.3 。测试过程采用在线调试的方式，实际测试

场景如图 5-7 与图 5-8 所示。

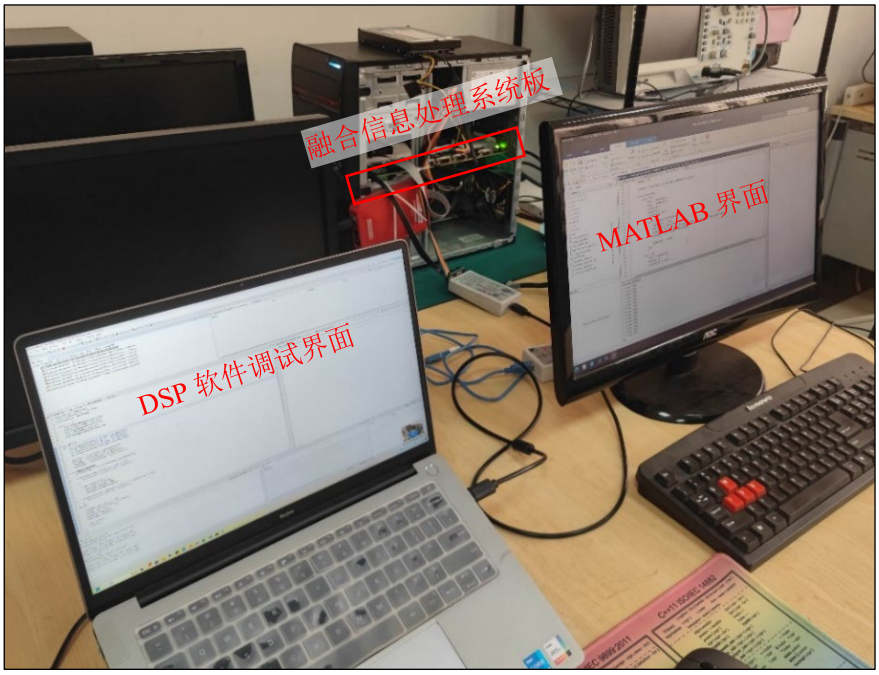


图 5-7 实际测试场景（远景）

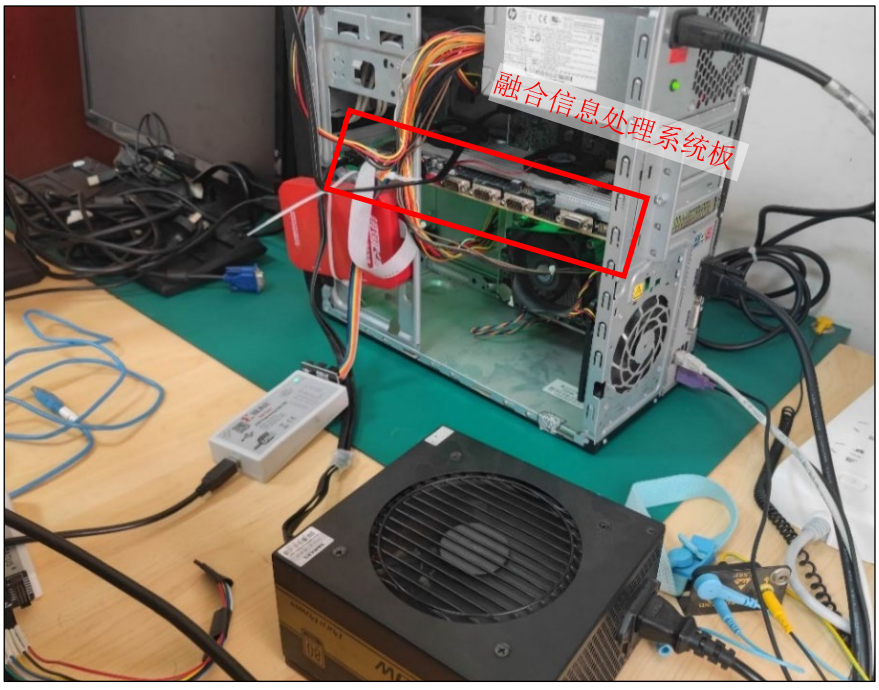
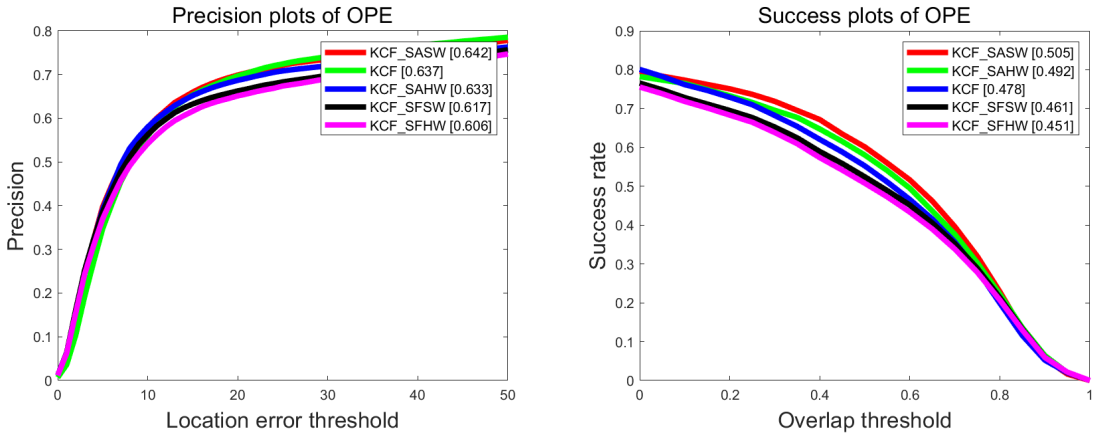


图 5-8 实际测试场景（近景）

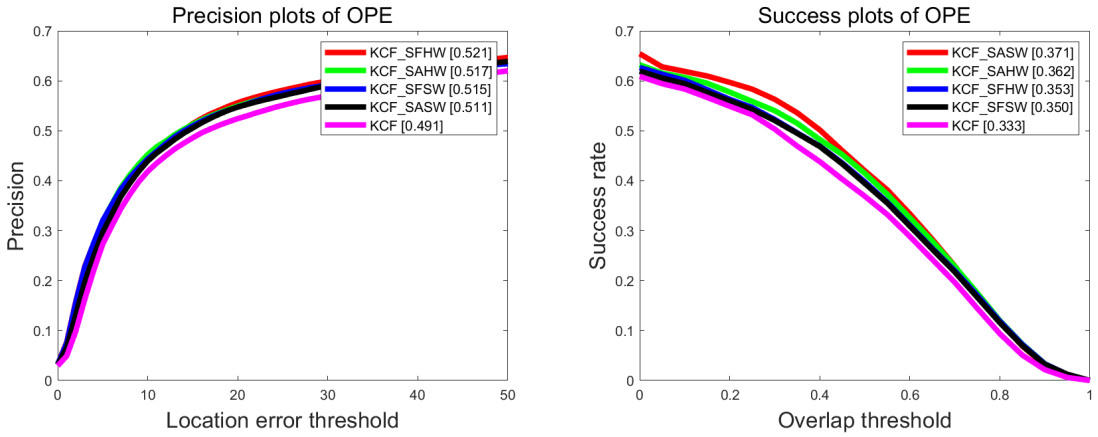
系统测试前都需要先将 FPGA 的比特流文件与 DSP 程序通过调试器进行加载，加载完成后上位机开机。DSP 软件首先开始运行，并进入等待启动测试的状态。上

位机系统在开机后需要加载硬件驱动程序，识别 PCIe 外设，接着上位机软件便可以运行。



(a) 准确率曲线　　(b) 成功率曲线

图 5-9 OTB100 总体测试结果



(a) 准确率曲线　　(b) 成功率曲线

图 5-10 UAV123 总体测试结果

四种跟踪器在 OTB100 数据集与 UAV123 数据集上测试后以 AUC 排序的结果分别如图 5-9 和图 5-10 所示，在 PC 上运行的 KCF 算法也作为参考算法一同展示其中。从两组数据集的测试结果中可以看出，在 DSP 上实现的四种跟踪算法与 KCF 参考算法的性能相近。四种算法在 UAV123 数据集中的测试准确率曲线几乎重合，没有明显的优劣区分。从整体上看，引入多尺度估计的算法（后缀带有“SA”）精度略优于固定尺度的算法（后缀带有“SF”）；仅由 DSP 软件实现的算法（后缀带有“SW”）精度略优于采用硬件加速的算法（后缀带有“HW”）。OTB100 数据集与 UAV123 数据集中的部分序列测试结果如图 5-11 和图 5-12 所示。

图 5-11 OTB100 部分序列跟踪结果

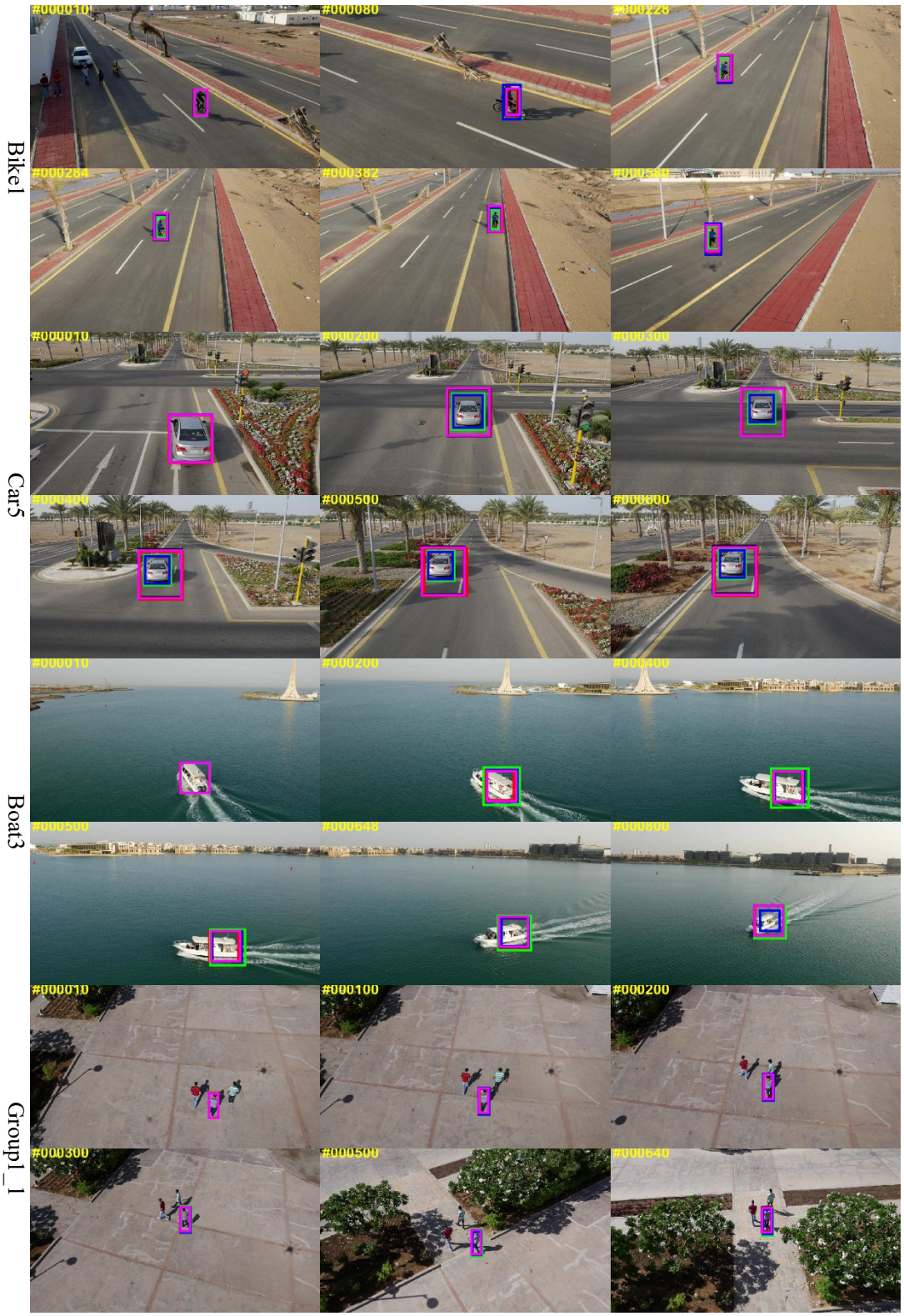


图 5-12 UAV123 部分序列跟踪结果

在算法的实时性测试方面，计算每种算法在不同测试序列下的帧率平均值可以得到表 5-3 所示结果。从表中结果可看出，固定尺度的 KCF 算法能够满足融合信息处理系统设计之初的每秒 50 帧的处理速率需求，但如果仅依靠 DSP 软件实现多尺度的检测，这一需求将无法满足，仅能达到 38 FPS。采用 FHOG 特征提取硬件加速电路后，算法实时性显著提高。

表 5-3 四种算法在两个数据集上的平均帧率

算法	OTB100 测试帧率（FPS）	UAV123 测试帧率（FPS）
KCF_SFSW	76.4	77.9
KCF_SFHW	164.9	171.7
KCF_SASW	38.3	38.6
KCF_SAHW	95.1	97.0

算法的实时性除了四种算法之间的比较以外，也将本系统中的测试结果与其它公开文献中的类似实验结果进行了比较。表 5-4 中除 KCF_SFSW 以外的其它四种方案均是在 TI 的 TMS320C6678 DSP 上实现的，TI 的 C6678 与 FT-M6678 有着相近的性能。KCF_NOSC (Non-Optimized Single Core) 表示由单个内核实现未优化的 KCF 算法；KCF_IPC 表示利用核间通信，以流水线的方式由多个内核实现的 KCF 算法；KCF_OMP 表示基于 OpenMP 实现的 KCF 算法；KCF_OPT 表示由单核实现的优化后的 KCF 算法。

表 5-4 不同 KCF 实现方案实时性对比

算法	帧率（FPS）
KCF_SFSW（本文）	76.4(OTB100), 77.9(UAV123)
KCF_NOSC[31]	12.5
KCF_OPT[27]	41.3
KCF_IPC[31]	64.9
KCF_OMP[31]	77.1

KCF_SFSW 相比于 KCF_OPT 的优势在于利用硬件 FFT 加速器进行加速，TI 的 C6678 缺少硬件 FFT 加速器的支持，因此 KCF_SFSW 能够实现更高的帧率。若与 KCF_IPC 或 KCF_OMP 相对比，KCF_SFSW 仅利用单个内核就实现了与两者相近的实时性。KCF_IPC 虽然使用八核流水线工作，但它的实际帧率并没有显著提升，可能不同核间任务划分不均匀，导致没有完全发挥出流水线的优势。KCF_OMP 的实时性与 KCF_SFSW 相当，但这一方案额外使用 7 个内核，计算效率较低。

5.3 本章小结

本章基于融合信息处理系统中的算法验证平台对四种在系统中实现的KCF目标跟踪算法进行了验证与评估。测试结果能够全面反映不同跟踪器的性能，同时也证明了算法验证平台的可行性。

系统中实现的四种 KCF 目标跟踪算法与上位机实现的 KCF 算法性能相当，采用尺度自适应调整的策略后，算法的准确率与成功率均略有提升。采用 FHOG 特征提取硬件加速电路加速后的算法能够在保证跟踪精度的前提下显著提升算法的实时性，固定尺度的 KCF 算法经过硬件加速后可以达到 170 FPS 左右，尺度自适应调整的 KCF 算法经过硬件加速也能够达到 95 FPS 以上。

6 总结与展望

6.1 本文主要内容及结论

本文提出并设计了一种基于国产 DSP 与 FPGA 的融合信息处理系统，用于红外激光融合的目标检测识别与跟踪算法研发，基于此系统建立起多核 DSP 软件框架，实现了高实时性的 KCF 目标跟踪算法，设计了算法硬件加速框架与算法验证平台并对四种目标跟踪算法进行了全面的验证与评估。

本文首先介绍了硬件系统的设计方案与多核 DSP 软件框架的设计。根据整体的红外激光融合目标检测跟踪方案，规划了系统的功能布局，设计了以一片多核 DSP，一片专用协处理器与两片 FPGA 组成的高性能硬件系统。将具体的任务分配到 DSP 的各个内核，形成多核 DSP 软件框架，为后续的开发明确了方向。

随后基于所设计的硬件系统与软件框架，实现了对任意尺寸目标的处理时间均恒定的高实时性 KCF 目标跟踪算法，搭建了算法硬件加速框架与算法验证平台。结合 DSP 的片内 FFT 硬件加速器与 EDMA 外设，提出了批量二维 FFT 快速实现方案，有效加速 KCF 算法执行，单个 DSP 内核仅需 12.6 ms 即可完成一帧图像的处理，利用两个内核分别对两个不同的目标进行跟踪也仅需 13.3 ms。算法硬件加速框架具有扩展性强，数据传输效率高，软件编程便捷的特点，使 FPGA 内的加速电路设计能够有效加速 DSP 中的复杂运算；基于公开数据集 UAV123 与 OTB100 的算法验证平台实现了上位机与 DSP 之间 1 GB/s 的流式数据传输，便于全面评估 DSP 上运行的目标跟踪算法性能。

最后基于算法验证平台对 DSP 执行的四种 KCF 目标跟踪算法进行验证与评估。测试结果表明在 DSP 上实现的 KCF 算法与原算法的性能相当，对特征提取环节采用硬件加速后，在保证算法跟踪精度的前提下显著提升了实时性。

6.2 本文的主要创新点

1) 完成 KCF 算法移植，实现了对任意尺寸目标的处理时间均恒定的高实时性 KCF 目标跟踪算法。结合 DSP 的片内 FFT 硬件加速器与 EDMA 外设，提出了批量二

维 FFT 快速实现方案，有效加速算法执行。单个 DSP 内核仅需 12.6 ms 即可完成一帧图像的处理，利用两个内核分别对两个不同的目标进行跟踪也仅需 13.3 ms。

2) 提出一种 FPGA 内的逻辑电路设计框架，用于为 DSP 中执行的算法提供硬件加速，这一框架称为算法硬件加速框架。框架主要由 AXI 交换结构，SRIO 收发控制器，EMIF 与 AXI4-Lite 转换器组成，具有扩展性强，数据传输效率高，软件编程便捷的特点。用户设计的多个 FPGA 硬件加速 IP 可以按照标准的 AXI 协议方便地接入控制总线与数据总线供 DSP 调用；DSP 与 FPGA 之间采用高速的 SRIO 链路互连，SRIO 收发控制器使 DSP 内核无需关心数据收发过程，实现高效的数据传输；EMIF 与 AXI4-Lite 转换器使 DSP 只需要少量的存储器访问操作即可完成对 FPGA 内寄存器的配置，从而快速实现硬件加速电路的调用。
3) 搭建了基于 OTB100 与 UAV123 数据集的嵌入式目标跟踪算法验证平台，并且完成了对四种在 DSP 上运行的跟踪算法的完整验证与评估。主要包括数据集验证平台中测试脚本的编写，SRIO 与 PCIe 桥接器的设计，DSP 与上位机软件的同步。测试脚本的编写使得算法验证过程能够直接嵌入原有的数据集验证平台中，用户可以通过 MATLAB 自动完成对 DSP 上跟踪算法性能的完整评估；SRIO 与 PCIe 桥接器的设计实现了 DSP 与上位机之间的流式数据传输，建立起高效的数据传输通路，数据传输速率可以达到 1 GB/s 左右；DSP 与上位机软件以约定的方式进行同步，多组图像序列的测试可以连续依次进行，算法的跟踪结果与执行的时间开销都能够送回上位机用于分析。

6.3 展望

本设计仍存在许多可以改进的空间，主要有以下两点：

1) SRIO 收发控制器可以对收到的事务包逐个进行处理，但是在启动 SW 传输时，只有某次传输请求完成后，才能启动下一次的 SW 传输。这对于实时嵌入式系统中的应用不太友好，某些比较紧急的传输任务可能会因为一些不紧急的数据正在传输而被迫等待。因此可以在 SRIO 收发控制器中引入多通道仲裁机制，面向用户开放不同优先级的数据传输通道，由控制器完成不同优先级传输任务的调度。

2) 融合信息处理系统的设计仅初具雏形。在融合信息处理系统多核 DSP 软件框架与算法硬件加速框架的基础上仍有许多工作需要完成。比如激光数据的处理，算法融合决策，红外目标检测识别等等。未来还可以在这一方向上继续深入研究，更充分地利用现有的硬件资源，实现完整的红外与激光数据融合的实时目标检测识别与跟踪。

致 谢

三年时光，匆匆飞逝，行文至此，感慨万千。一路走来，我的学习成长之路离不开许许多多的人给予我的帮助、鼓励与支持，在这里一并表示衷心的感谢！

首要感谢的是我的导师桑红石老师。仍然记得第一次和桑老师交流的时候，我说我对嵌入式开发方面感兴趣。在随后的三年里桑老师完全遵照我的意愿为我制定研究方向，提供健全的实验设备与宽敞的调试环境；在我遇到困难时，帮我分析问题，开拓思路。桑老师的全力支持使我度过了充实而又有意义的三年。

我也要特别感谢我的女朋友熊梦瑶。小熊在大学刚入学时对任何新的事物都充满好奇心，一次自制电路的故障使我们相识。在随后的几年里，我们一起学习进步，并肩前行，探索未知。在我心情低落的时候，她总会及时地陪伴与鼓励我；在我出差的时候，她会特意为我准备晕车药；在最近临近论文提交的截止日期，她为我买了好多咖啡并督促我写论文。研究生三年以来，和小熊逐渐走近并最终走到一起是我收获的最大的财富。回想起我们之间的点点滴滴，有欢声笑语也有矛盾争吵，细细回忆起来都令我们不禁莞尔。一件件小事已经将我们紧紧联系在一起，以后我们也要继续努力奋斗！相信我们会有美好的未来！

除此之外，我还要感谢在在电工电子科技创新中心认识的老师和同学们。包括亦师亦友的王贞炎老师和李蔚琳学姐，靠谱的电赛队友曹琳和金楚琪，见多识广的胡总还有其他一同参与培训和比赛的队员们、老师助教们。本科阶段的电赛经历为我开启了嵌入式开发领域的大门，未来我也会继续在这条道路上坚定地走下去。

实验室的同学们就像我的家人一样，给我提供了多方面的帮助。吴相涛、李双、李立师兄在我刚加入实验室时为我提供了耐心的指导；我的好兄弟于春梁经常能及时地提醒我去上课或参加线上会议，在我们一起找工作时也给了我很多帮助；林勉和钱怡雯设计的 FHOG 特征提取硬件加速电路使我的毕业设计更加完整；风趣幽默的况云峰、魏启昊、任劳任怨的孙雨浩、谢梓龙都是我们实验室不可或缺的一部分。

最后，我要特别感谢我的父母和外婆。近些年我长年在外求学，每年只有过年的时候能回家看望他们。非常感谢他们对我的关心与支持！

参考文献

[1] 潘丽娜. 激光与红外传感器联合跟踪算法研究. 光电技术应用, 2009, 24(3): 25-28

[2] 郝静雅. 红外/激光雷达数据融合跟踪算法研究: [硕士学位论文]. 西安电子科技大学, 2015

[3] 彭滔. 基于激光和红外双传感器的低空多目标跟踪技术: [硕士学位论文]. 中北大学, 2022

[4] 战荫泽, 张立东, 秦颖. 基于激光雷达与红外图像融合的车辆目标识别算法. 激光与红外, 2021, 51(9): 1238-1242

[5] 郑欣悦, 赖际舟, 吕品, 袁诚, 范婉舒. 基于红外视觉/激光雷达融合的目标识别与定位方法. 导航定位与授时, 2021, 8(3): 34-41

[6] 黎正平. 远距离单光子三维成像的技术研究: [博士学位论文]. 中国科学技术大学, 2020

[7] 康英杰. 单光子成像激光雷达及其与短波红外图像融合技术研究: [硕士学位论文]. 山东大学, 2021

[8] 颜洪雷. 红外与激光复合探测关键技术研究: [博士学位论文]. 中国科学院研究生院（上海技术物理研究所）, 2014

[9] 宋盛. 红外与激光双模复合探测关键技术研究: [博士学位论文]. 中国科学院大学(中国科学院上海技术物理研究所), 2017

[10] 刘羽丰. 激光红外复合车辆检测算法及多 DSP 软件实现: [硕士学位论文]. 华中科技大学, 2020

[11] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J. K. Kamarainen, 等. The Tenth Visual Object Tracking VOT2022 Challenge Results. 2022

[12] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J. K. Kamarainen, 等. The Eighth Visual Object Tracking VOT2020 Challenge Results. 2020

[13] P. Nousi, I. Mademlis, I. Karakostas, A. Tefas, I. Pitas. Embedded UAV Real-Time Visual Object Detection and Tracking. 见: 2019 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2019: 708-713

[14]J. F. Henriques, R. Caseiro, P. Martins, J. Batista. High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(3): 583-596

[15]M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Č. Zajc, G. Nebehay, 等. The Visual Object Tracking VOT2014 challenge results. 2014

[16]M. Danelljan, G. Häger, F. S. Khan, M. Felsberg. Learning Spatially Regularized Correlation Filters for Visual Tracking. 见: 2015 IEEE International Conference on Computer Vision (ICCV), 2015: 4310-4318

[17]M. Danelljan, A. Robinson, F. Shahbaz Khan, M. Felsberg. Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In: Computer Vision – ECCV 2016, Cham, Springer International Publishing, 2016: 472-488

[18]G. Bhat, M. Danelljan, L. Van Gool, R. Timofte. Learning Discriminative Model Prediction for Tracking. 见: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019: 6181-6190

[19]王向军, 罗仁, 徐小东. 小型嵌入式平台实时目标跟踪算法的研究与实现. 传感技术学报, 2021, 34(1): 27-33

[20]时旭东, 施华君, 陆国强. 基于 DSP 的嵌入式目标跟踪系统. 计算机系统应用, 2019, 28(11): 87-95

[21]鲁国智. 基于嵌入式系统的视觉跟踪算法研究: [硕士学位论文]. 杭州电子科技大学, 2018

[22]吴玉兴. 基于 SDI 相机的 FPGA 和 DSP 实时目标跟踪系统设计与实现: [硕士学位论文]. 中国石油大学(华东), 2020

[23]于文怡. 基于显著性和 KCF 的嵌入式运动平台实时 RGB-T 融合跟踪算法研究: [硕士学位论文]. 西安电子科技大学, 2021

[24]曹庆礼. 智能目标跟踪算法研究及嵌入式系统实现: [硕士学位论文]. 西安电子科技大学, 2021

[25]T. Zhang, S. Zhang, Z. Li, S. Z. T. Zhang, S. Zhong. A method of adaptive learning rate tracking for embedded device based correlation surface evaluation. https://doi.org/10.1117/12.2538083, 2020, 11430: 93-101

[26]魏鹏辉. 基于嵌入式平台的智能目标跟踪算法的研究与实现: [硕士学位论文]. 西安电子科技大学, 2019

[27]杜文彬, 毛征, 梅伟军, 贾文洋, 韩嘉隆. 基于 DSP6678 的 KCF 算法实现及优化系统设计. 国外电子测量技术, 2017, 36(7): 62-67

[28]李传超. 基于 FPGA+DSP 架构的目标跟踪系统设计与实现: [硕士学位论文]. 华中科技大学, 2021

[29]许剑清. 基于 FPGA+DSP 架构的图像目标跟踪器研制: [硕士学位论文]. 哈尔滨工业大学, 2018

[30]W. Wu, Q. Jia, F. Luo, T. Yang, H. Huang. A Parallel Optimization Method for KCF based on Inter-core Communication of Multi-core DSP. 见: 2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2021: 1736-1740

[31]W. Wu, Q. Jia, S. Li, B. Teng. A parallel optimization method for kernel correlation filter based on multi-core DSP. 见: IET International Radar Conference (IET IRC 2020), 2020: 1416-1420

[32]王跃宗. 高目标响应背景感知 KCF 算法及并行化实现: [硕士学位论文]. 中国石油大学(华东), 2019

[33]X. Gong, Z. Le, H. Wang, Y. Wu. Study on the Moving Target Tracking Based on Vision DSP. Sensors 2020, Vol. 20, Page 6494, 2020, 20(22): 6494

[34]C. Yang, Z. Chen, Y. Zhang, Y. Li. Design of Embedded Target Tracking System Based on MobileNet and KCF. 见: 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2020: 1578-1581

[35]M. Danilowicz, T. Kryjak. Real-Time Embedded Object Tracking with Discriminative Correlation Filters Using Convolutional Features. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2022, 13569 LNCS: 166-180

[36]管宪宇. 基于视觉的无人机地面目标跟踪方法研究: [硕士学位论文]. 天津大学, 2019

[37]张清洋. 基于 DSST 算法的目标跟踪系统设计与实现: [硕士]. 华中科技大学,

2017

[38]李玉涛. 盖革 APD 面阵激光雷达距离像强度像生成电路研究: [硕士学位论文]. 华中科技大学, 2019

[39]涂直健, 张天序, 桑红石. 自动选取辅助目标的建筑物目标间接定位方法. 应用光学, 2019, 40(4): 603-611

[40]TI. LM3880 Three-Rail Simple Power Supply Sequencer. Texas Instruments, 2013

[41]TI. SYSBIOS (TI-RTOS Kernel) v6.40 User's Guide. Texas Instruments, 2014

[42]TI. TMS320C6000 Programmer's Guide. Texas Instruments, 2011

[43]TI. TMS320C66x DSP CPU and Instruction Set Reference Guide. Texas Instruments, 2010

[44]P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, 32(9): 1627-1645

[45]R. C. Gonzalez, R. E. Woods. 数字图像处理. 阮秋琦, 阮宇智, 译. 第三版. 电子工业出版社, 2011

[46]Y. Wu, J. Lim, M. H. Yang. Online object tracking: A benchmark. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2013: 2411-2418

[47]M. Mueller, N. Smith, B. Ghanem. A benchmark and simulator for UAV tracking. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, 9905 LNCS: 445-461

[48]Y. Li, J. Zhu. A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration. In: Computer Vision - ECCV 2014 Workshops, Cham, Springer International Publishing, 2015: 254-265

[49]M. Danelljan, G. Häger, F. S. Khan, M. Felsberg. Discriminative Scale Space Tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(8): 1561-1575

附录 1　攻读硕士学位期间取得的研究成果

发表与接收论文

[1] **裘剑东**，桑红石. 核相关滤波目标跟踪算法在 DSP 上的实现. 红外技术.（已录用）

附录 2　其它附录

A.1　循环矩阵对角化推导

循环矩阵 $X_{n\times n}$ 由一个一维列向量样本 $x_{n\times 1}$ 经过循环移位得到。

$$x=\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix}, X=C(x)=\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_0 & x_1 & \cdots & x_{n-2} \\ x_{n-2} & x_{n-1} & x_0 & \cdots & x_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & x_3 & \cdots & x_0 \end{bmatrix} \tag{A-1}$$

如果将样本 x 看作一个长度为 n 的数字序列，那么关于这个序列的相关运算和卷积运算可以用循环矩阵的形式方便地表示。

在数字信号处理技术中，序列 x 的与序列 x' 的相关运算与循环卷积如公式(A-2)和(A-3)所示，得到的序列 y_{corr} 和 y_{conv} 分别是相关运算与循环卷积的结果。

$$y_{corr}[i]=\sum_{\tau=0}^{n-1}\tilde{x}'[\tau]\tilde{x}[i+\tau], i=0,1,2,\cdots,n-1 \tag{A-2}$$

$$y_{conv}[i]=\sum_{\tau=0}^{n-1}\tilde{x}'[\tau]\tilde{x}[i-\tau], i=0,1,2,\cdots,n-1 \tag{A-3}$$

其中 $\tilde{x}$ 和 $\tilde{x}'$ 分别表示序列 x 的与序列 x' 的周期延拓，用于离散傅里叶分析的数字序列都隐含有这种周期性，而我们只以其主值区间 $[0,n-1]$ 内的值来表示这一无限长的周期序列。在引入循环矩阵后，我们可以用更紧凑的方式表示样本序列的相关运算：

$$y_{corr}=C(x)x' \tag{A-4}$$

通过观察也可以得出循环卷积的矩阵表示方式：

$$y_{conv}=\left(C(x)\right)^{\mathrm{T}}x' \tag{A-5}$$

循环卷积可以利用 DFT 将时域的卷积替换为频域的乘积，DFT 的分析式与合成式分别如公式(A-6)和(A-7)所示：

$$\hat{x}[k]=\sum_{i=0}^{n-1}x[i]W_n^{ki}, k=1,2,3,\ldots,n-1 \tag{A-6}$$

$$x[i]=\frac{1}{n}\sum_{k=0}^{n-1}\hat{x}[k]W_n^{-ki},i=1,2,3,\ldots,n-1 \tag{A-7}$$

其中 $\hat{x}$ 表示 x 的 DFT 变换，W_n^{ki} 称为旋转因子。$W_n=e^{-j2\pi/n}$ 的任意整数幂次始终落在复平面的单位圆上，改变幂指数的大小就像使它在复平面上旋转一样。DFT 可以用矩阵的形式表示，便于后续推导，将公式(A-6)写成矩阵的形式：

$$\hat{x}=\begin{bmatrix}1 & 1 & 1 & 1 & \cdots & 1\\ 1 & W_n^1 & W_n^2 & W_n^3 & \cdots & W_n^{n-1}\\ 1 & W_n^2 & W_n^4 & W_n^6 & \cdots & W_n^{2(n-1)}\\ 1 & W_n^3 & W_n^6 & W_n^9 & \cdots & W_n^{3(n-1)}\\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & W_n^{n-1} & W_n^{2(n-1)} & W_n^{3(n-1)} & \cdots & W_n^{(n-1)(n-1)}\end{bmatrix}\begin{bmatrix}x[0]\\ x[1]\\ x[2]\\ x[3]\\ \vdots\\ x[n-1]\end{bmatrix}=Fx \tag{A-8}$$

DFT 矩阵 F 有着非常好的性质，首先从结构上可以看出，它是一个对称矩阵，满足 $F^{\mathrm{T}}=F$，$F^{\mathrm{H}}=F^{*}$，它有着非常接近酉矩阵特性：

$$FF^{\mathrm{H}}=F^{\mathrm{H}}F=\begin{bmatrix}n & 0 & 0 & \cdots & 0\\ 0 & n & 0 & \cdots & 0\\ 0 & 0 & n & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & n\end{bmatrix}=n\mathbf{I}_n \tag{A-9}$$

对序列 y_{conv} 进行 DFT 计算，可以通过交换两次求和的顺序推导出时域卷积与频域相乘的关系：

$$\begin{aligned}\tilde{Y}_{conv}[k]&=\sum_{i=0}^{n-1}\left(\sum_{\tau=0}^{n-1}\tilde{x}'[\tau]\tilde{x}_2[i-\tau]\right)W_n^{ki}\\ &=\sum_{\tau=0}^{n-1}\tilde{x}'[\tau]\sum_{i=0}^{n-1}\tilde{x}[i-\tau]W_n^{k(i-\tau)}W_n^{k\tau}\\ &=\sum_{\tau=0}^{n-1}\tilde{x}'[\tau]W_n^{k\tau}\tilde{X}[k]\\ &=\tilde{X}'[k]\tilde{X}[k]\end{aligned} \tag{A-10}$$

将这一关系用矩阵形式表示：

$$FC(x)^{\mathrm{T}}x'=\hat{x}\odot\hat{x}' \tag{A-11}$$

其中 $\hat{x}\odot\hat{x}'$ 表示 $\hat{x}$ 与 $\hat{x}'$ 的哈达玛积，即对应位置的分量分别相乘后得到一个新的向量，

进一步将公式(A-11)进行转换后可以得到：

$$C(x)^{\mathrm{T}} x' = F^{-1}\,\mathrm{diag}(\hat{x})\hat{x}' = \frac{1}{n} F^{\mathrm{H}}\,\mathrm{diag}(\hat{x}) F x' \tag{A-12}$$

其中 $\mathrm{diag}(\hat{x})$ 表示将 $\hat{x}$ 的每个分量放在对角线上构成对角矩阵， $\mathrm{diag}(\hat{x})\hat{x}'$ 表示 $\hat{x}$ 与 $\hat{x}'$ 的哈达玛积。公式(A-12)对任意的 n 维列向量 x'都成立，因此可以消去，再结合 DFT 矩阵的酉矩阵特性，引入酉矩阵 U ，满足 $U = F/\sqrt{n}$ ，最终可以得到：

$$C(x) = U\,\mathrm{diag}(\hat{x})U^{\mathrm{H}} \tag{A-13}$$

A.2 SRIO 与 PCIe 桥接器寄存器说明

0x00 IDENTIFIER

字段	位域	复位值	读写权限	描述
ID	[31:0]	0x10370918	R	标识寄存器

0x04 SRIO_CSR

字段	位域	复位值	读写权限	描述
DB Start	0	0	R/W	写 1 发送 TXDBINFO 寄存器中的门铃信息。发送完成自动清零；发送未完成时写 1 无效，并记录错误状态。
SW Start	1	0	R/W	写 1 向 DSP 发送 SW 事务包。发送完成自动清零；发送未完成时写 1 无效，并记录错误状态。
RSVD0	[15:2]	0	R	保留
ERR_DB	16	0	RW1C	在有 DB 发送请求未完成时尝试再次发送门铃
ERR_SW	17	0	RW1C	在有 SW 发送任务未完成时尝试再次启动 SW
ERR_NW Cross	18	0	RW1C	NW 跨越 4k 边界
ERR_NW Unalign	19	0	RW1C	NW 地址不是 8 字节对齐或 Size 不是 8 字节倍数
ERR_SW DstUnalign	20	0	RW1C	SW 源地址不是 256 字节对齐
ERR_SW SrcUnalign	21	0	RW1C	SW 目的地址不是 256 字节对齐
RSVD1	[31:22]	0	R	保留

0x08 SRIO_MODE

字段	位域	复位值	读写权限	描述
DB NoMSI	0	0	R/W	0：收到门铃后会发起 MSI 中断； 1：门铃不会产生 MSI 中断。
SW Mode	1	0	R/W	0：SW 来自 XDMA 的 H2C 通道； 1：SW 来自 AXI-MM，需要由 IP 主动从外部读数据。
NW Mode	2	0	R/W	0：NW 发往 XDMA 的 C2H 通道； 1：NW 接收的数据转到 AXI-MM 接口。
RSVD	[31:3]	0	R	保留

0x0C SW_SIZE

字段	位域	复位值	读写权限	描述
SW Size	[26:0]	0	R/W	发送 SW 的数据量，以双字(8 bytes)为单位，实际传输的数据量为 Size+1。最大单次传输数据量 $2^{27}\times 8B=1GB$。
RSVD	[31:27]	0	R	保留

0x10 SW_DST

字段	位域	复位值	读写权限	描述
Destination Address	[31:0]	0	R/W	当 SW 的目的地址

0x14 SW_SRC

字段	位域	复位值	读写权限	描述
Source Address	[31:0]	0	R/W	当 SW 的源地址，只有在 SW Mode = 1 的时候有效

0x18 DB_TXINFO

字段	位域	复位值	读写权限	描述
Normal DBINFO	[15:0]	0	R/W	一般用途的门铃信息，直接发送。
SW DBINFO	[31:16]	0	R/W	SW 事务完成后发送的门铃信息。

0x1C MSI_CSR

字段	位域	复位值	读写权限	描述
MSI_IRQ	[15:0]	0	R/W	有 MSI 请求，但 PC 还没有响应
ERR_MSI	[31:16]	0	RW1C	在有 MSI 请求还没被 PC 响应时再次产生请求
RSVD	其余	0	R	保留

0x20~0x5C MSI_INFO_n（n = 0~15）

字段	位域	复位值	读写权限	描述
INFO	[31:0]	0	R/W	需要由 PC 读取的信息，清除 IRQ。

A.3 实验测试环境

本文设计已开源：https://gitee.com/jayden1998/gp-sup

表 A-1 主要实验环境

	PC1	PC2	PC3
处理器	Intel® Core™ i5-11300H @ 3.10GHz	Intel® Pentium® CPU G640 @ 2.80GHz	Intel® Core™ i7-10700F @ 2.90GHz
操作系统	Windows 11 家庭版 22H2	Ubuntu 20.04.5 LTS Kernel 5.15.0-69-generic	Windows 11 专业版 21H2
软件	CCS 6.2.0 Matlab R2021a	Matlab R2022b gcc 9.4.0 g++ 9.4.0 cmake 3.16.3	Vivado 2019.2 OrCAD Capture CIS 17.4
主要用途	DSP 软件设计 数据集测试结果分析	上位机软件设计 数据集测试平台	FPGA 逻辑电路设计 硬件原理图设计

A.3.1 软硬件 FFT 对比

实验平台	PC1
SYS/BIOS	6.34.4.22
XDCTools	3.25.6.96
dsplib	3.4.0.4

A.3.2 双目标实时跟踪

实验平台	PC1
SYS/BIOS	6.34.4.22
XDCTools	3.25.6.96
IPC	1.25.3.15

A.3.3 数据传输速率测试

实验平台	PC1 + PC2 + PC3
SYS/BIOS	6.34.4.22
XDCTools	3.25.6.96

A.3.4 算法验证与评估

实验平台	PC1 + PC2 + PC3
SYS/BIOS	6.34.4.22
XDCTools	3.25.6.96
OpenCV	3.4.8