Y1874177

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：曙宜锡 签字日期：2010年 6月13日

学位论文版权使用授权书

本学位论文作者完全了解 天津大学 有关保留、使用学位论文的规定。特授权 天津大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：曙宜锡 导师签名：

签字日期：2010年 6月13日 签字日期：2010 年 6 月 13 日

中文摘要

本文主要研究了基于空间变换核天文图像降晰算法及硬件实现，以达到南极天文站（AST3）对图像实时性和低功耗的要求。本文实现了一种基于传输触发体系结构（Transport Triggered Architecture，TTA）的完全可配置可扩展 T*CORE 处理器的设计，该处理器作为硬件加速协处理器实现了基于空间变换核天文图像降晰算法中的关键计算部分。TTA 架构可以实现最大限度的指令集并行，能够在一个周期内执行多条指令，因此 TTA 架构非常适合数据密集型的计算任务。利用空间变换核降晰算法中数据传送规律，改进了数据存储方式和计算流程，将大小为N×N空间变换核的天文图像处理每点所需数据传输由原算法N×N点降至平均只需$(2N-1)^2/N^2(<4)$点，数据复用率达到 $N^4/(2N-1)^2$(约 $N^2/4$)。可配置处理器的设计是通过自动化软硬件协同设计平台针对特定应用进行快速设计空间探测，定制最优的硬件架构及相应的功能单元，其可定制性使其具备传统 ASIC 方法性能高、功耗低的优势，同时其处理器架构又保证了很大的设计灵活性，弥补了当前主流的基于通用处理器实现的缺陷。

在上述研究的基础之上，本文设计了基于可配置可扩展处理器 T*CORE 的异构多核 SoC（System on Chip）架构平台，来实现天文图像降晰算法，并利用 6 个 T*CORE 处理器作为协处理器并行处理的方法，提高运算速度，最终实现课题研究的实时性和低功耗要求。在此基础之上也进一步证明了以可配置可扩展处理器T*CORE为基本组成模块的SoC设计新方法的巨大优势和良好发展前景。在FPGA上的实验结果表明，与 PC 纯软件处理相比，本文提出的基于 SoC 设计的嵌入式方案在实现运算时加速比达到了 29.06 且功耗仅为纯软件处理时的 2.98%。

关键词： 空间变换核；降晰算法；可配置可扩展处理器；传输触发体系结构；SoC

ABSTRACT

This thesis presents the study on the algorithm of Space-varying Kernel of astronomical image blurring and its hardware implementation to achieve real-time and low power requirements for data processing in AST3 project. A fully configurable and extensible T*CORE processor hardware design which is based on transport triggered architecture (TTA) is implemented. This processor as hardware acceleration coprocessor achieves the key computation task of the algorithm of Space-varying Kernel of astronomical image blurring. TTA architecture can achieve maximum parallel instruction set and be able to execute multiple instructions in one cycle, so the TTA architecture is very suitable for data-intensive computing tasks. By utilizing the rules of data transfer in the space-varying kernel algorithm for astronomical image blurring, computation flow and the ways of data storage are redesigned, which can reduce the number of pixels that had to be transmitted from $N \times N$ to $(2N-1)^2/N^2(<4)$ when processing a single pixel in an astronomical image with $N \times N$ Space-varying Kernels. Accordingly, the reuse rate of data can reach $N^4/(2N-1)^2$, almost $N^2/4$ times. The design of configurable processor is done by use of automatic software and hardware co-design platform for a specific application for rapid design space exploration, customizing optimal hardware architecture and corresponding functional unit. The property of customization of configurable processor has the advantages of the traditional ASIC approach such as high performance, low power consumption, while its processor architecture ensures the great design flexibility comparing with the implemention based on general-purpose processor.

Based on these studies, this thesis presents a design of heterogeneous multi-core SoC(System on Chip) platform based on configurable and extensible processors T*CORE to achieve the astronomical image blurring algorithm. Six T*CORE processors are used for parallel processing as a way to ultimately realize the real-time and low power requirements. The experimental results demonstrate that our design outperforms the existing processing of PC by 29.06 times in speed, and the power consumption is reduced to 2.98%. From the result, it also shows the great advantage and good prospects of the new SoC design method based on the configurable and extensible processors T*CORE as the basic component module.

KEY WORDS: Space-varying kernel; image blurring; configurable and extensible processor; Transport Triggered Architecture; SoC

目　录

第一章 绪论

1.1 课题研究背景、目的和意义

本课题选自国家自然基金资助的南极天文实时测光系统与关键技术项目。AST3（Three Antarctic Schmidt Telescope）是 2010 年初（第二十六次南极科考）在南极昆仑站上安装的三个 50cm 口径的施密特望远镜。AST3 的每个望远镜配备了 10K×10K 的 CCD 相机，可以覆盖约 3×3 度的天空，空间采样 1”/pixel。CCD 相机以漂移扫描（drift-scan）方式工作。望远镜工作模式有两种，一种是准跟踪一个固定的天区，在一次漂移扫描完成后，将望远镜重新指向这个天区，进行重复扫描； 另一种是无跟踪观测，固定望远镜，指向一个赤纬方向，随地球自传每天扫描同一个赤纬环带。在两种观测模式下，数据量一定。

假定无跟踪对赤纬－60 度观测，每个天体有近 24 分钟的积分时间。在漂移扫描方向上，CCD 可以有 10 个通道读出，所以每 2.4 分钟便可以有一幅 200MB 的图像。这样，每天总的数据量是 120G×3＝360GBytes。四个月总的观测时间，可以有 43.2TB 原始数据。

南极昆仑站上目前只能进行夏季科考，在三到五年后才能够成为常年的越冬站，AST3 将在无人值守的情况下全天连续进行越冬观测。AST3 的观测图像将能够发现众多瞬变源，但由于图像数据量大，无法通过卫星传回，因此要求数据在本地服务器通过实时测光系统进行处理、测量，并将测量结果和部分重要图像传回。数据处理包括图像差异分析、PSF 拟合测光、孔径测光、光变曲线分析、标准模板建立等。

数据处理从最初的数据采集开始，最后能够得到所有观测到的天体和发现的瞬变源的星表和光变曲线。后续数据的处理部分依赖于前期观测的一些结果，所以，在能够进行正常数据处理流程之前，必须选择并利用前期的观测数据建立一些标准。这些标准包括选择观测天区的测光标准星（photometric standards），星空坐标的标准星（astrometric standards）。最后利用观测数据建立观测天区的模板，并进行测光和位置的定标。

数据处理的数据流程图（AST3 Data Flow）如图 1-1 所示。数据采集下来后，首先要进行 CCD 图像的一些必要处理（Image Processing pipeline），然后需要探测图像里的源，并利用模板和坐标标准星对其位置进行定标。随后需要对图像的

质量进行测量，同时对图像进行最重要的测光过程。测光过程包括三种不同的方法：1、图像差异分析（Difference Image Analysis），即图像相减测光；2、点扩散函数拟合（PSF－fitting）测光；3、孔径测光。测光结果中的星表将包括所有源的位置和星等，其中瞬变源将被特别挑出，以他们为中心的小图（Stamp Image）也会被切下来单独存放，以供和星表一起传回。

由于AST3实时测光系统（AST3_RPS，Realtime Photometry System for AST3）需要在无人值守的条件下长时间持续运行，并且每2.4分钟需要完成200MB数据的处理，因而在软件系统的可靠性、计算性能方面要求严格。本课题的研究目标是在成熟测光算法的基础上，并行化和优化核心算法以满足实时处理的要求，并进一步将计算密集型的图像处理算法通过专用硬件加速模块实现，为未来更大规模的实时数据处理奠定基础。因此，本课题的研究工作主要围绕AST3的实时测光数据处理进行，包括三方面内容：高可靠硬件系统SoC构架方案研究、高性能密集型数据处理方案研究、天文图像空间变换核降晰算法的分析优化及硬件设计。

在AST3天文图像处理中基于空间变换核的图像降晰处理占据51.02%的计算时间（图1-1中的红色区域），是所有处理流程中最费时的部分，因此研究空间变换核降晰算法并对其中的关键点优化，硬件并行加速处理是非常有意义的。本文研究作为AST3项目的一部分，主要研究图像相减算法中的空间变换核降晰运算，对算法进行优化，并把算法移植到嵌入式SoC架构平台中进行运算，通过设计专门的硬件加速器提高运算速度，以达到实时性的要求。对于天文图像空间变换核降晰算法中的数据密集型运算，采用一种灵活的构架——可配置可扩展处理器，作为SoC架构系统的一个协处理器，大大提高了天文图像处理的运算速度，也满足了对系统实时性和低功耗的要求。

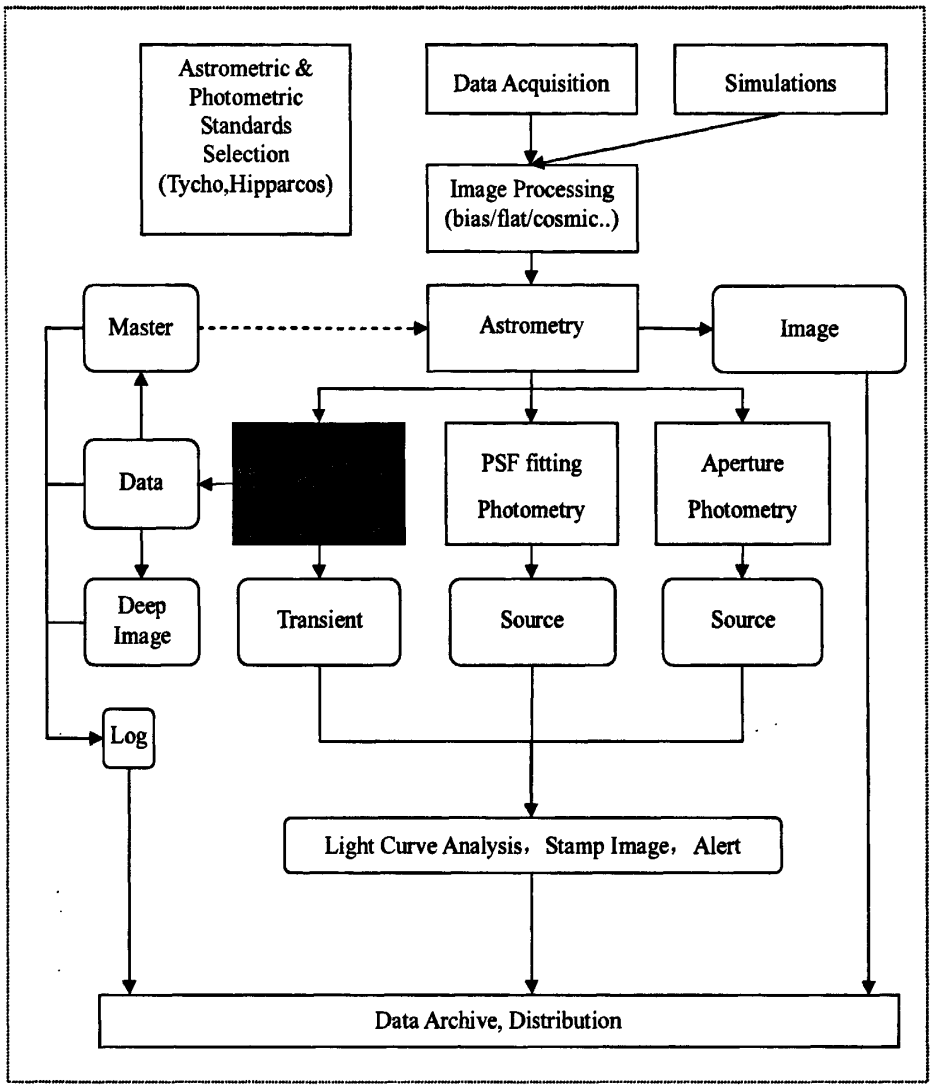


图 1-1 天文图像数据处理流程图

1.2 国内外天文图像相减处理研究状况

目前在天文观测中，使用 CCD 检测变源的算法中都通过将当前获取图像与模板图像相减来检测星星光强和位置的变换。图像相减是求两张相近照片的差异，从中提取差异信息的一种运算。通过在不同时期拍摄的两张照片相减，在医学上可用来发现病灶的变化；在军事上可以发现地面军事设施的增减；在农业上可以预测农作物的长势；在工业上可以检查集成电路掩膜的疵病，等等。还可用于地球资源探测、气象变化以及城市发展研究等各个领域。图像相减是相干光学

处理中的一种基本的光学数学运算，是图像识别的一种主要手段。实现图像相减的方法很多，本文通过 OIS 算法实现图像相减并将变源检测出来。

首先尝试使用图像相减来获取变源的是 Tomaney 和 Crotts。为了将两张图片完美的相减，一张图片和另一张的清晰度必须非常准确的匹配。Tomaney 和 Crotts 通过降低清晰度较好的图片以匹配清晰度较差的图片，由图片相减获取的质量完全依赖于变换核的求解，并且寻找恰当的核是一个非常精巧的过程。Tomaney 和 Crotts 提出通过对图像中每个亮星做作傅里叶变换获取变换核。然而由于图像的高频部分受到噪音的影响，因而必须使用高斯分布来检测边缘。此算法不保证相减操作后能够获取很好的效果。此外，该算法对复杂的 PSF 变换处理很差，并且没有解决天空背景不同时的处理[1]。

紧接着 Kochanski 提出了最优算法，该算法通过寻找一个核变换使两幅图像差异最小化，然而该算法是一个非线性的最小二乘拟合过程，就计算时间而言，该算法是不能实现的[2]。Alard 和 Lupton 基于 Kochanski 思想通过核变换分解将算法转换为线性最小二乘法问题，使算法在实际得以应用[3][4]。其开源软件 ISIS 可以免费下载(http://www2.iap.fr/users/alard/package.html[5])。然而要完成 AST3 项目，依靠 Alard 算法仍然不能胜任，使用 ISIS 软件在天津大学计算机学院的 DELL2950 服务器上处理 2K×2K 的图片需要 40 秒，最保守估计处理 10K×10K 图片需要 40×25=1000 秒=16.7 分种，其中大部分时间都花费在计算上，而硬件加速器处理大量数据密集型的运算将是最有效，再加上南极环境所要求的低功耗，因此采用硬件加速器来处理大量数据密集型的运算将是最有效的。

1.3 本文研究目标和主要内容

由于图像差异分析是发现变源的核心计算过程，要求保证实时性。因此，本文研究的主要目标是基于空间变换核天文图像降晰算法，将图像差异分析中的关键计算模块通过设计专用硬件模块加以实现，以达到实时性和低功耗的要求。本文给出了一种区别于以往部分可配置可扩展处理器架构的基于传输触发体系结构的完全可配置可扩展 T*CORE[6]处理器的硬件设计，以实现数据密集型的图像降晰运算。利用空间变换核降晰算法中数据传送规律，改进了数据存储方式和计算流程，从而大大减少了数据传输次数。可配置处理器的设计可通过自动化软硬件协同设计平台针对特定应用进行快速设计空间探测，定制最优的硬件架构，可定制性使其具备传统 ASIC 方法性能高、功耗低的优势，同时其处理器架构又保证了很大的设计灵活性，弥补了当前主流的基于通用处理器实现的缺陷。

在上述研究的基础之上，本文设计基于可配置可扩展处理器 T*CORE 的异

构多核 SoC 架构平台，来实现天文图像空间变换核降晰算法，并利用多个T*CORE处理器并行处理的方法，提高运算速度，最终实现课题研究的实时性和低功耗要求。在此基础之上也进一步证明了以可配置可扩展处理器 T*CORE 为基本组成模块的SoC设计新方法的巨大优势和良好发展前景。

1.4 本文组织结构

本文共分七章，论文结构如下所示：

第一章，介绍了本文的课题研究背景、目的和意义，阐述了天文图像处理的发展现状、存在问题以及本文的主要研究工作。

第二章，主要讨论了与本文的研究工作相关的背景知识和主要技术研究基础。

第三章，给出了天文图像空间变换核降晰算法分析及其对关键部分算法的优化处理，为下一步进行硬件设计做好准备。

第四章，提出了可配置可扩展 T*CORE 处理器的设计，给出了各关键组件的硬件设计方案。它用于处理数据密集型的计算，作为 SoC 系统的一个协处理器，通过嵌入式CPU核来控制，来实现一个低功耗、实时性的SoC嵌入式系统。

第五章，构建了基于空间变换核天文图像降晰算法硬件实现的 SoC 系统。对算法的软硬件功能进行划分，并在基于FPGA的可编程嵌入式开发系统中构建SoC系统平台，实现软硬件协同处理操作过程。

第六章，主要针对各种数据比较，来选择最优 T*CORE 构架，并以此确定最适合SoC平台的各硬件功能模块。最后对各种最终实验结果进行总结比较。

第七章，对本论文进行了总结并提出了以后发展的目标。

第二章 可配置可扩展处理器研究基础

本章讨论与本文研究工作相关的理论和方法基础。第一节主要提出一种新型体系结构（TTA）。第二节主要介绍本项目组设计的基于TTA架构的可配置可扩展处理器T*CORE处理器。第三节主要阐述了SoC设计方法的发展与挑战，并提出一种基于可配置可扩展处理器作为SoC基本组成模块的SoC设计方法。

2.1 传输触发架构（TTA）

传输触发体系结构（Transport Triggered Architecture,TTA）是由TU Delft计算机工程系的Henk Corporaal教授提出的一种新型处理器体系结构，是一款优秀的指令级并行处理器架构，根据TTA架构他们做出了一款处理器MOVE32INT[7]，并且还开发了一套能够辅助设计人员将一些特殊应用移植到TTA架构的集成开发环境中，称之为MOVE Framework。TTA架构是传统VLIW体系结构的超集，如图2-1所示[8][9][10][11][12]。传统的VLIW架构处理器属于操作触发体系结构（Operation Triggered Architecture，OTA），通过对指令的译码获取操作类型，然后触发数据传输完成相应计算。而TTA正好相反，通过将数据传输到相关计算部件来触发相应的运算。不同于VLIW在一条指令中打包多个操作，TTA则是在一条指令中打包多个数据传输[13][14]。因此VLIW也被称为单指令多操作（Single Instruction Multiple Operations，SIMO）体系结构，而TTA被称为单指令多传输（Single Instruction Multiple Transport，SIMT）体系结构。TTA与VLIW相比另一大同点是，TTA不需要在每个FU和RF之间都建立一条独立的数据连结。

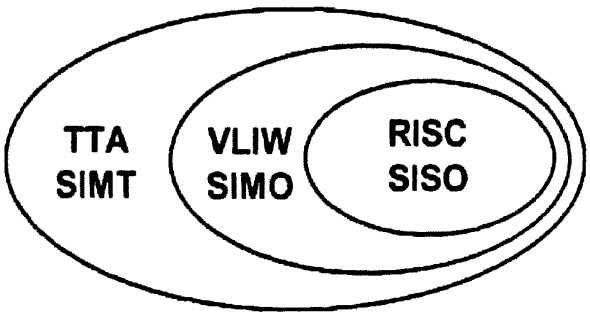


图2-1 处理器的分类

TTA 是根据传输网络中的特定数据类型指定操作方式，它只有一种类型的操作，即从一个 FU 到另一个 FU 的 move 操作。通常，FU 通过三种寄存器类型（operand、trigger 和 result registers）互连到传输网络中。所有的 move 操作本质上就是从寄存器到寄存器的操作（r_x->r_y），这些寄存器堆可以看作是一种特殊的 FU。功能单元 FU 用来进行运算，在 FU 中主要包含以下三种寄存器：

1. Operand 寄存器（O）：操作数寄存器，这种寄存器在读写操作上和一般的寄存器相同，但是当触发运算时，会作为参与运算的另一个操作数。

2. Triggered 寄存器（T）：触发寄存器，当这种寄存器被传入新的数据时，会触发相应的运算。一个 Triggered 寄存器可以触发多种运算。

3. Result 寄存器(R)：结果寄存器，用来存放 FU 运算的结果。

一个典型的 TTA 操作和传输架构处理器如图 2-2 所示，由各类功能单元 FU、包含通用寄存器（GPRs）的寄存器堆 RF 以及各种互连网络组成。互连网络分为 bus（总线）和 Socket 两部分，所有的 FU 和 RF 通过 Socket 与总线连接，彼此交换数据。每个 Socket 相当于一个交叉开关，每个时钟周期可以将总线上的一个数据传输到相应的 FU 或 RF，或者将 FU、RF 中的数据传输到一条或多条总线之上。通过 Socket，总线与 FU、RF 构成了松耦合的连接模式，设计者可以通过任意的增/减 FU、RF 来调整处理器计算能力，或通过改变总线宽度、数目来调整互连网络的传输能力，只要各自遵循与 Socket 的标准接口即可，这非常适合于可配置处理器的理念。通过这些改变，可以使该处理器对特殊的应用而进行量身定制，最大限度的发挥处理器的性能[15][16]。另外，各个 Socket 与总线采用部分连接方式，去除不必要的数据连接，以获取更低的功耗。

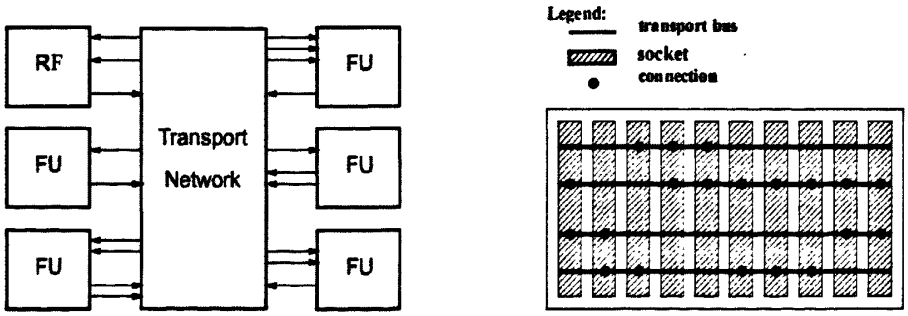


图 2-2 典型 TTA 操作和传输处理器框架

基于 TTA 架构的处理器具有模块化、高灵活性、完全可配置、完全可扩展等特点，拥有更加巨大的设计空间。处理器的设计者可以根据特定应用的需求任意定制处理器的各个参数，包括：FU 的种类、RF 中通用寄存器的数目、总线的宽度、总线的数目、特殊指令格式等，从而使处理器架构达到与应用的最佳匹配，

获得最高的性能、最低的成本（芯片面积）、最少的功耗，拥有最优的性能/价格比。另外，基于 TTA 架构处理器的数据传输在体系结构一级可见的特性使得程序设计者可在更细粒度上对程序进行控制，通过操作数共享、旁路技术以及死结果的去除，使处理器能够进一步提升性能、降低成本、功耗。基于这些优势，本文将选用 TTA 作为完全可配置可扩展 T*CORE 处理器的基本架构，以展开后续研究。

2.2 可配置可扩展处理器 T*CORE 处理器

T*CORE 处理器是天津大学超大规模集成电路设计与应用研究所设计的具有自主知识产权的基于 TTA 架构的完全可配置可扩展的处理器，其可配置性和可扩展性的优点为快速进行特定应用功能处理器设计提供了可能，体现了 T*CORE 处理器的灵活性和高性能。T*CORE 处理器主要是基于传输触发体系结构进行设计实现的一系列可配置可扩展处理器。T*CORE 并不是指某一具有固定指令集的处理器，而是一个标准处理器模板，也就是说，设计者需要根据应用需求所得到的处理器体系结构配置参数选取并配置处理器的各关键组件，并依据相应的标准接口将组件进行适当的组合和信号连接，构成面向某一具体应用的处理器。从传输触发体系结构的可知，此架构与其他处理器架构相比，其性能、成本、功耗上的优势主要是因为数据传输细节在体系结构一级可见，程序设计者可以在更细粒度上对程序进行控制。然而这一特点也正是基于传输触发体系结构处理器的最大缺陷，大量可见传输细节导致属于同一操作的多个数据传输可能将分布在多个指令当中，而属于同一指令的各个数据传输也不一定描述的是同一个操作，这样大大增加了处理器状态保存和更新的难度，导致基于传输触发体系结构的处理器需要过于昂贵的成本才可实现中断处理，进而也使移植基于硬件的抢占式（pre-emptive）多线程操作系统的难度剧增。因此，不同于以往基于传输触发体系结构的处理器以通用目的作为设计目标，如 MOVEINT32，T*CORE 系列处理器被定位为轻量级协处理器，去除了以往面向通用目的进行设计时对于中断、Cache 等复杂逻辑的支持，大大降低了实现难度，进一步节省了芯片成本、功耗，从而使其更适应于以处理器为中心的新型 SoC 设计的要求。

2.3 基于可配置可扩展处理器的 SoC 设计方法

2.3.1 SoC 设计方法的发展与挑战[17]

未来的 SoC 中将会用到更多的处理器，以便更加灵活的支持不断出现的新应用。设计方法也会改进来应对新的挑战，它会对设计工具提出新的要求，产生新的设计技术。这些趋势主要体现在以下方面：

1. IP 复用将不仅在硬件领域，在软件设计领域同样需要；

2. 基于平台的设计，今后的设计将在一个应用平台上完成，该平台将包括一个或多个处理器和逻辑单元；

3. 可编程、可配置、可扩展的处理器核的使用，会使得原有的设计流程和设计者思维发生变化；

4. 系统级验证，利用高级语言搭建验证平台和编写验证向量，需要相应的工具支持；

5. 软、硬件协同综合，使得在同样的约束条件下，系统达到最优的设计性能。

这些都要求设计层次向着更高的抽象层次发展。当前普遍的 SoC 设计方法都是基于以一个或多个高性能通过 RISC 处理器核（如 ARM、MIPS 等）为核心，构成一整个片上系统体系结构。然而，通用处理器核的性能往往不能满足如信号处理、视频图像、协议实现、信息安全等数据密集型处理任务。现有的典型设计方案，大多是在 SoC 片上加载一个算法加速单元，通过硬连线逻辑的设计方法来实现。但可以看到，这样的设计方法会导致芯片面积和成本的急剧上升，同时算法加速单元的效率也可能会由于处理器核本身功能上非专用的缺陷而受到限制。

从系统级设计方法可以看到，要最有效地发挥 SoC 的性能，尽可能早地与软件结合确定设计方案是一种行之有效的方法。而如果能够针对软件设计出专用性强、算法实现效率高的特殊处理器核，则会更方便地实现 SoC 的高效性。于是，一种新型的处理器内核——可配置的、可扩展的微处理器核（Configurable Extensible Microprocessor Core）应运而生了。这样的处理器往往通过一个生成器或编译器直接生成，这种过程的特点是可以将高层次的应用需求以指令集、代码或者高级语言直接转换为高效的硬件设计和软件工具，从而极其方便地为设计者如何使用、整合、互连形成处理器提供了积极有效的帮助，也为以这样的一个或多个处理器为核心构成 SoC 提供了极大的参照便利，更重要的是能够使其尽早结合软件应用来实现处理器和 SoC 性能的优化。

使用这些可配置、可扩展的专用处理器作为 SoC 系统中子系统的底层构件，完全可以根据软件的需求做合适的调整，省略其中不必要的硬件组成部分，使得部分合适的软件能对其专用或共享，而其他软件程序则可以根据自身要求再“裁减”一套新的处理器核。从这点可以看出，处理器的效率肯定会得到大幅度的改善。

诚然，现今也有相当多的设计者使用多个高性能通用微处理器核构成 SoC，同样也能够提高处理性能。但需要提出的是，在这样地的环境下，至少一个处理器核的功能肯定不会得到完全发挥，其内部对应用而言无用的逻辑会使整个速度和功耗指标受到损失。但是专用可配置处理器在这点上的优势较为明显，自动生成的微处理器核只包含应用所需要的功能，因此基础指令集中一定不含有应用目标中不会使用和无关的功能，由此相应带来的面积、功耗和速度等方面的负担也会得相应的减少。因此，使用可配置可扩展的专用处理器作为 SoC 的功能模块，会成为 SoC 设计中的一种必然选择。

2.3.2 基于可配置可扩展处理器的 SoC 设计流程

使用可配置可扩展处理器作为基本组成模块的多处理器 SoC 的出现极大地影响了 SoC 设计流程。以可配置可扩展处理器为基本组成模块的 SoC 设计流程如图 2-3 所示[18]。

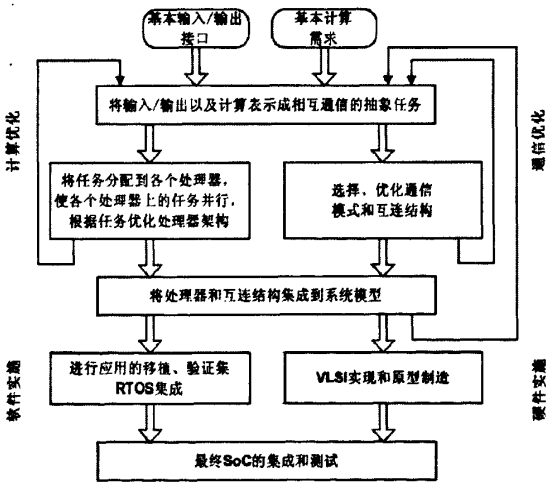


图 2-3 可配置可扩展处理器为基本组成模块的 SoC 设计流程

整个流程的基本输入是 SoC 接口和功能需求。基本的输入/输出接口规格说明包括物理层电气规格说明和更高层次的逻辑规格说明，还有需符合的协议。基本的计算需求包括联系输入和输出地算法说明和算法行为的时间约束，后者包括

最大迟滞，最小带宽和其他需求指标，如成本、功耗等。整个流程包括四个主要的子流程：计算优化、通信优化、软件优化、软件实现和硬件实现。几乎所有反馈环都发生在这些单元内。计算优化致力于找到实现系统计算需求的复杂嵌套过程所需的恰当的处理器的数目和类型。通信优化重点在于提供必要的通信软件编程模型和处理器、存储器和I/O接口之间的硬件互连，以获得低成本迟滞的通信，尤其对于系统中要求最为苛刻的数据流。这个流程依赖于两个重要原则。第一，对于复杂系统架构，早期的系统级仿真是给设计者提供对功能和性能问题的深入了解的一个重要工具。第二，从最初推测的实现到最终的应用，处理器配置、存储器、互连和输入/输出实现的快速的递增式的精化，让设计者快速地获知并利用关于成本和性能的信息。

此处，传统RTL设计者和嵌入式软件开发者都能很轻松很有效地利用处理器生成工具进行新处理器配置和专用指令集的设计。因此，利用可配置可扩展处理器为基本组成模块的SoC设计流程意味着更少的设计错误，更少的硬件设计迭代，更多的模块划分选择，更早的模块软件测试和验证，并且很可能也意味着更早的产品产出。可以预测，以多个处理器和可配置的处理器为中心的复杂SoC设计必将成为未来的主流。

第三章 天文图像空间变换核降晰算法分析与优化

本章主要对天文图像空间变换核降晰算法进行详细的归纳与分析，找出算法规律，并对其中的关键部分算法进行优化处理，为后续的硬件设计做准备。

3.1 OIS 算法简介

OIS(Optimal Image Subtraction Method: 最优图像相减算法)算法使用一组大约 150 颗不变的恒星(在 Alard(1998)这篇论文之后称作“stamps”)对欲相减的两张图像 I(当前给定的源图像)，R(参考图像作为模板)进行对准，即坐标对齐。然后依据 I，R 构建一个卷积核 K，使用 K 对 R 降晰使 R 和 I 清晰度相同。执行相减操作，从而得到变源。

其中，求解 K 的过程称为反卷积或反降晰。基于空间变换核的降晰算法是整个天文图像相减处理的核心运算部分，因此本文仅对空间变换核降晰算法进行分析和优化。

给定一组基准向量（局部核函数）$\{K_n; n=1,2,...N\}$，建立全局核函数 K：

$$K=\sum_{n=1}^{N} a_n K_n \qquad \text{公式（3-1）}$$

局部核函数 K_n 及局部核函数系数 a_n 分别如下公式所示：

$$K_n(u,v)=e^{-(u^2+v^2)/2\sigma_k^2}u^i v^j \qquad \text{公式（3-2）}$$

$$a_n(x,y)=\sum_{i=0}^{d}\sum_{j=0}^{d-m} a_{n,i,j}x^i y^j \qquad \text{公式（3-3）}$$

其中，n 是关于 i, j, k 的广义指数分布 $n=(i,j,k)$，$a_n(x,y)$ 为关于 x ,y 的二元多项式，d 为多项式的阶，$0\leqslant m\leqslant d$。

定义核降晰运算如下：

$$[R\otimes K]_{x,y}=\sum_{u=x-\Delta w}^{x+\Delta w}\sum_{v=y-\Delta w}^{y+\Delta w} R_{u,v}K_{u-x+\Delta w+1,v-y+\Delta w+1} \qquad \text{公式（3-4）}$$

其中，$[R\otimes K]_{x,y}$ 为 R 经过 K 降晰后产生的矩阵元素。$R_{u,v}$ 为图像矩阵 R(x,y) 元素，$K_{u-x+\Delta w+1,v-y+\Delta w+1}$ 为核函数矩阵 $K(u,v)$ 元素，$\Delta w=\frac{1}{2}(L-1)$，L 为卷核函数矩阵边长。

下式定义在整个图像之上。

$$I = R \otimes K \qquad \text{公式（3-5）}$$

将公式（3-1）、（3-2）、（3-3）代入公式（3-5）式后，使用最小二乘法通过求解得到系数向量$\{a_{n,r,s}; n \in [1,N], r \in [0,d], s \in [0,d-r]\}$，该向量与$K_n$相乘得到核函数K。

$$K = \sum_{l=1}^{N} \left(\sum_{r=0}^{d} \sum_{s=0}^{d-r} a_{n,r,s} e^{-\frac{u^2+v^2}{2\sigma_k^2}} u^l v^j \right) \qquad \text{公式（3-6）}$$

结合公式（3-6）得到R降晰结果C。

$$C = R \otimes K \qquad \text{公式（3-7）}$$

3.2 天文图像空间变换核降晰处理算法基础

本小节主要是对天文图像空间变换核降晰算法中一些关键算法做一些介绍，阐述所用到的一些算法思想，分析其基本原理，为下一步对算法进行优化提供理论基础，能更好的进行优化处理。

3.2.1 PSF概述

光学系统的理想状态是物空间一点发出的光能量在像空间也集中在一点上，但实际的光学系统成像时，物空间一点发出的光在像空间总是分散在一定的区域内，其分布的情况称为PSF(Point Spread Function：点扩展函数)，描述成像到CCD上的点源强度随位置变化的函数。点扩散函数是评价光学系统成像质量的基本工具，在数字图像复原及识别中更是一个关键的参数[19]。此函数为1对多的非线性函数，在本文中通过核函数K描述。PSF成像传输系统[20]如图3-1所示。

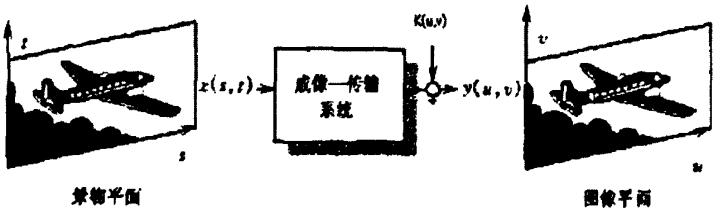


图3-1 PSF成像传输系统

3.2.2 降晰函数

成像系统从景物中获取图像的过程是一个降质过程，几何光学的分析表明，

光学系统散焦造成图像降晰的点扩展函数是一个均匀分布的圆形光斑。Gauss 降晰函数是许多光学测量系统和成像系统最常见的降晰函数。这是因为对于这些系统而言，影响系统点扩展函数的因素比较多，众多因素结合的结果总是使点扩展函数趋于 Gauss 型[21]。Gauss 降晰函数一般形式为：

$$h(m,n)=\begin{cases} Ke^{-\frac{u^2+v^2}{\alpha}} & (m,n)\in C \\ 0 & other \end{cases} \qquad \text{公式（3-8）}$$

其中，K 为归一化系数，α 为正常数，C 为 Gauss 降晰函数 $h(m,n)$ 的圆形支持域。

3.2.3 核函数

核函数的基本作用就是接受两个低维空间里的向量，能够计算出经过某种变换后在高维空间里的向量内积值。此方法就是用非线性变换 $\Phi(x)$ 将 n 维矢量空间中的随机矢量 x 映射到高维特征空间ℵ：

即：$x\rightarrow\Phi(x)\in\aleph$ 公式（3-9）

在高维特征空间中设计的线性算法，若其中各坐标分量间的相互作用仅限于内积，则不需要知道非线性变换 $\Phi(x)$ 具体形式，只要满足 Mercer 条件的核函数替换线性算法中的内积，就能得到原输入空间中对应的非线性算法[21]。可以看出：核函数方法是一项将标准的线性方法推广为非线性方法的强有力的技术，核函数本质上是对应于高维空间中的内积，从而与生成高维空间的特征映射一一对应。核函数方法正是借用这一对应关系隐性的使用了非线性特征映射(当然也可以是线性的)。这一方法使得我们能够利用高维空间让数据变得易于处理，不可分的变成可分的，同时又回避了高维空间带来的维数灾难，不用显式表达特征映射[22]。Mercer 条件此处不予说明。

由 3.2.2 分析可知，CCD 成像系统的点扩展函数可以用 Gauss 降晰函数表示。本文使用 Gauss 核函数描述恒星的 PSF。在 3.2.2 中 Gauss 降晰函数可以表示整个图片的 PSF，然而，在天文图像处理中 CCD 图片都很大，如在 AST3 项目中图像大小为 10K×10K，并且天文图片的 PSF 函数是空间变换的。这样使用一个随空间变换的核函数描述基于空间变换的 PSF。定义如下：

$$K_{i,j}(u,v)=(i-\frac{m}{2})^u(j-\frac{m}{2})^v e^{-\frac{(i-m/2)^2+(j-m/2)^2}{\sigma^2}} \qquad \text{公式（3-10）}$$

其中，$(i,j)\in C$，C 为图像支持域。(i,j) 为图像上的点 σ 为归一化参数，(u,v) 为拟合 K 多项式的阶参数的排列组合。

如上所示核函数 K 为四维的函数，处理 AST3 大小为 10K×10K 图像数据时计算量太大，中间的存储单元太多，计算机很难实现。并且不能直接使用 FFT

进行反卷积和图像估计。因此可以考虑使用局部核函数将全局核函数分解为若干块。使得每块可以近似为非空间的降晰函数[20]。

分割算法如下：

（1）首先选定一种像素领域，这样图像就可以由若干个领域覆盖。领域可以交叠也可以不交叠。这样图像矩阵就相应地变换为若干个分块矩阵，每一块就是领域大小的像素块。

示例如下：

$$R=\begin{pmatrix} R_1 & R_2 & \cdots & R_s \\ R_{s+1} & R_{s+2} & \cdots & R_{2s} \\ \vdots & \vdots & \vdots & \vdots \\ R_{N-s+1} & R_{N-s+2} & \cdots & R_N \end{pmatrix},\ I=\begin{pmatrix} I_1 & I_2 & \cdots & I_s \\ I_{s+1} & I_{s+2} & \cdots & I_{2s} \\ \vdots & \vdots & \vdots & \vdots \\ I_{N-s+1} & I_{N-s+1} & \cdots & I_N \end{pmatrix} \quad \text{公式（3-11）}$$

其中，对 $\forall n \in [1,N]$，R_n 和 I_n 都坐标对齐。

（2）然后针对每一块确定一个局部核函数(局部核函数作用于全图，不过在每块之外为恒 0)，以分块的局部核函数之和作为整个图像的全局核函数 K。

$$K(u,v)=\sum_{n=1}^{N} a_n K_n \quad \text{公式（3-12）}$$

本文使用 3 个高斯轮廓作为核函数 K，然后拟合 R 和 I 的星像，使得 R 和 I 的可见度归一化。

高斯轮廓参数的选取在文献[3]中没有说明，可以参考文献[23]和文献[24]。在文献[23]中 Astier 选取主要依据实践，文献[24]中 Israel 研究出对于不同性噪比(S/N)和抽样率时选取高斯轮廓参数的方法。

3.3 基于最小计算时间的空间变换核的解法

I，R 已经对齐并且在同一天区、使用相同的望远镜并且 CCD 相机具有相同的滤镜，但是拍摄时间不同，大小为 $m \times m$。I 的清晰度比较差，R 的清晰度比较好作为参考图像(模板)。R 使用卷积核 K 降晰 R，使其清晰度与 I 相同。本文简单的总结 OIS 图像相减的基本原理(Alard & Lupton 1998)。算法的核心是找到一个将参考图像(R)转换为与给定图像(I)具有相同清晰度的卷积核(K)，得到核函数 K 后对 R 实施卷积：用 I 减去 $[R \otimes K](x,y)$ 的结果，最后分析亮度或者位置发生变化的物体。

为了叙述简单，整个算法分成两个阶段。第一阶段假设核函数 K 不随空间变换，也就是说 PSF 在整个图像中不变。第二阶段利用第一阶段对解法的理解，阐述随空间变换的 Gauss 核函数求解过程。

3.4 不同天空背景时图像降晰处理

在上面核函数 K 的求解过程中只是考虑大气扰动，没有考虑天空背景对 PSF 的影响，因此很有必要添加天空背景不同时的影响。Alard 和 Lupton 在 I 和 R 天空背景不同时可以通过添加二元多项式 $bg(x,y)$ 矩阵来修改模型 $I = R \otimes K$（由于离散系统中二元多项式可以看作关于 x, y 变换的矩阵，所以此处对于函数和矩阵不做严格区分）[3]。考虑天空背景时的模型为：

$$I = R \otimes K + bg \qquad \text{公式（3-13）}$$

$$bg_{x,y} = \sum_{i=0}^{d_{bg}} \sum_{j=0}^{d_{bg}} c_{i,j} x^i y^j \qquad \text{公式（3-14）}$$

其中：d_{bg} 为拟合背景多项式的阶，(x, y) 为像素的坐标。

3.5 空间变换核降晰算法流程分析及数据格式优化

本文首先使用 Alard 和 Lupton 所实现的软件 ISIS[5] 分析了天文图像降晰算法的运算流程、运算时间、数据取值范围等。其中图像降晰算法数据流程如图 3-2 所示，各部分所需计算时间如表 3-1 所示。

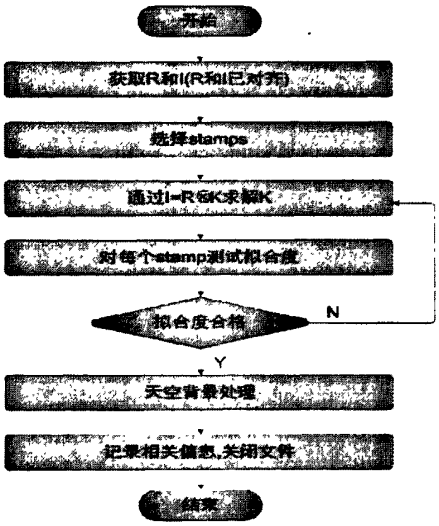


图 3-2 算法流程图

表 3-1 空间变换核降晰算法各模块所占计算比重

功能	对应公式	所占计算比重
求局部核函数	(3-2)	16.2%
求局部核函数系数	(3-5)	5.6%
求解核函数	(3-6)	3.9%
执行降晰运算	(3-7)	74.3%

从表 3-1 可知，空间变换核降晰算法中计算量最大部分为公式（3-7）执行降晰运算，其次为公式（3-2）计算核函数部分。公式（3-2）中指数部分 $-(u^2+v^2)/2\sigma_{k^2}$ 由于含有因子 σ_k 使得 $e^{-(u^2+v^2)/2\sigma_{k^2}}$ 范围确定，采用查表法来实现将大大减少运算时间。从而使公式（3-7）成为唯一的计算瓶颈。以图像大小为 2K×2K、卷积核为 19×19 为例，再依据公式（3-7）执行降晰运算时需要的计算为 $2K\times2K\times19\times19=1.51\times10^9$ 次，因此，通过设计硬件提高公式（3-7）的计算速度是提高整个空间变换核降晰算法实现的计算速度及降低功耗的最有效手段。由于核函数求解的实现非系统瓶颈部分，本文在设计计算空间变换核降晰运算单元时不予深入探讨。

3.5.1 降晰运算分析

空间变换核降晰算法实现的处理流程可以归纳为三步：(a)依据程序配置参数对图像分块。(b) 依据公式（3-3）求解各块的核函数。(c)依据卷积公式（3-5）执行降晰运算。详细处理流程如下：

1) 将图像划分为若干小格，不足一格按照一格处理（虚线处），图中 I 颜色区域为一格。

2) 将 R 中核 K 一半大小的边界丢弃不做处理，最终需要处理为 R 中虚线区域。

3) 对小格的中心（如 R 中(x,y)）使用 make_kernel 构建对应的核矩阵 Kernel（III 颜色区域）。

4) 依据定义公式（3-2）执行运算求解 C[x,y]的值。

5) 对 2)划分的区域，从左到右至上而下执行 3)、4)步骤直至虚线区域中的小格全部处理。

6) C 的边缘处全为 0（第二步中未作处理的部分）。

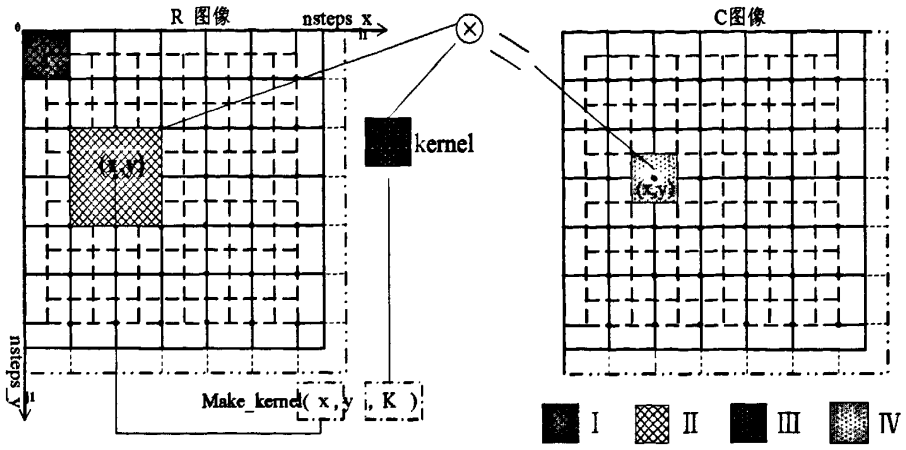


图 3-3 降晰计算示意图

对每一块 M 计算区域是由 19×19 个点组成的，根据执行降晰计算公式（3-7），其中每一点的降晰计算过程都可由如图 3-4 所示来表示。对应于图 3-4 中 M 区域块中 n 点的计算过程。

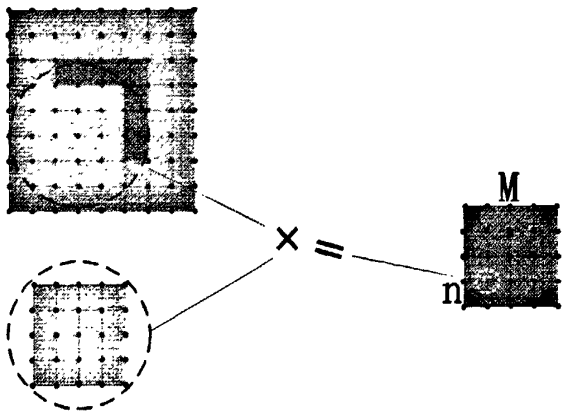


图 3-4 执行每一像素点的降晰运算过程

从降晰计算过程可以看出R图像中数据的复用率很高，假设核大小为N×N，计算图 3-3 中 C 图像时需要 N^4 数据，然而这些数据全部在R图像II颜色区域大小仅为$(2N-1)^2$，数据重复达 $N^2/4$。因此，硬件系统提高空间变换核降晰运算速度的核心之一为提高数据复用率，即减少数据存/取时间。

3.5.2 数据存储格式优化

由于 ISIS 软件是运行在大型计算机上，在整个数据处理流程中均使用 IEEE 754 标准浮点双精度 64bit 格式表示数据。这种数据表示方法虽然保证了很高的

数据精度，但在基于嵌入式方法实现时由于存储资源和运算资源有限，大量的64bit浮点数运算不仅需要设计大量存储空间和运算单元，并导致系统硬件庞大、功耗上升，从而无法用于设在极地的天文观测实时处理等应用中。然而由于中间数据取值范围较大，若使用IEEE 754单精度浮点数表示，天文图像信噪比(SNR)太低不能满足天文处理的精度要求。为此本文使用特定基线测试向量来分析性能和准确性，在详细分析了数据处理过程中数据的范围后设计了自定义浮点数据格式用于硬件设计之中。空间变换核降晰算法中取值范围最大的几个中间变量如表3-2所示。

表3-2 空间变换核降晰算法中变量变化范围

名称	最大值	最小值
K_n	3.654052647388	0.000000000001
a_n	831.3745742220	0.000000029103
K	6402.242176810	0.000000000001
$a_{n,i,j}$	48062.89723834	0.000001861958

针对上述数据的特点，本文自定义了一种新的浮点数格式，其设计方法可参照[25]所阐述的设计方法。设计浮点数表示方式的关键：一是在浮点的总字长给定的情况下，如何选择尾数基址，使浮点数的表示范围最大、表数精度和表数效率最高；二是在尾数基址确定之后，如何根据表数范围和表数精度的要求具有确定尾数长度p和阶码长度q。从表3-2我们可知，降晰运算中最大值N = 10^{12}，表数精度不低于δ = 2.98023e^{-008}，所以我们可以根据第二条思想确定p和q，来确定浮点数的格式。计算p和q的公式所下所示。

$$q > \frac{\log\left(\frac{\log^N}{\log^2}+1\right)}{\log^2} \qquad p > \frac{-\log^\delta}{\log^2} \qquad \text{公式（3-15）}$$

由上面的两个公式计算可得，q = 5.35273.　p = 24.9905。所以本文自定义的浮点数格式如图3-5所示。

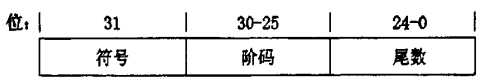


图3-5 自定义的浮点数格式

其中，阶码使用移码表示，阶数的基为2，浮点数的总字长32位。可以表

数范围最大正数为：$\left(1-r_m^{-p}\right)\times r_m^{r_e^q-1}=9.22337e^{+018}$；最小正数为：$\frac{1}{r_m}\times r_m^{-r_e^q}=2.71051e^{-20}$；最大负数为：$-\frac{1}{r_m}\times r_m^{-r_e^q}=-2.71051e^{-20}$；最小负数为：$-\left(1-r_m^{-p}\right)\times r_m^{r_e^q-1}=-9.22337e^{+018}$。其表数范围完全可以满足降晰运算的数据表示范围。

以采用标准的双精度浮点数格式运算的结果作为标准，对采用自定义的浮点数格式进行运算及采用单精度浮点数格式进行运算的结果进行比较，所得信噪比如表 3-3 所示。

表 3-3 不同数据类型图像降晰结果性噪比

类型	SNR(db)
单精度	75.220086
自定义	93.263391

由上可知，自定义的数据存储格式从表数范围、精度、误差都满足天文图像处理的要求，而所需要的硬件资源将大大减少，因此完全适合 T*CORE 架构的新型数据类型。

3.6 本章小结

本章主要是对天文图像空间变换核降晰算法进行分析和优化，阐述了本文所要用的算法思想，并根据天文图像空间变换核降晰算法的运算规律，设计了一种新的数据类型，在满足精度要求的前提下，减少运算复杂度，节省硬件资源。最后可以根据算法分析所要实现的特定应用功能，给出了一种特定 T*CORE 处理器架构，具体详细设计可见下一章所述。

第四章 可配置可扩展 T*CORE 处理器设计

本章将对基于传输触发体系结构的完全可配置可扩展 T*CORE 处理器进行架构设计。由于 T*CORE 处理器的所有体系结构参数对于设计者都是可见的，因此，设计者可根据应用的需求构建最优的 T*CORE 处理器架构方案。

4.1 T*CORE 处理器总体架构

T*CORE 处理器模板的整体架构如图 4-1 所示，由核心计算组件、T*CORE 与外界通信的接口组件以及 T*CORE 的指令和数据存储器组件等几大部分组成。T*CORE 核心计算组件是根据特定应用需求定制的，主要负责完成该应用的所有计算任务，是 T*CORE 处理器的核心。数据存储器(Data_RAM_1，DATA_RAM_2 . . . DATA_RAM_n) 使用接口组件 WRAPPER 进行封装，用于选择外界或是 T*CORE 本身 LD/ST 单元访问该数据存储器，而外界访问拥有较高的优先级。这样，在一个 SoC 系统中，其它的 Master(例如 MCU、DMA 等)就可以通过该端口来访问 T*CORE 的数据存储器，以便于数据的交互。指令存储器 INS_RAM 也通过一个 WRAPPER 进行封装，用于选择外界或 T*CORE 本身去访问该指令存储器。当外界访问 T*CORE 的指令存储器时，主要完成系统上电时的初始化任务，将 T*CORE 所运行的程序导入到指令存储器中。

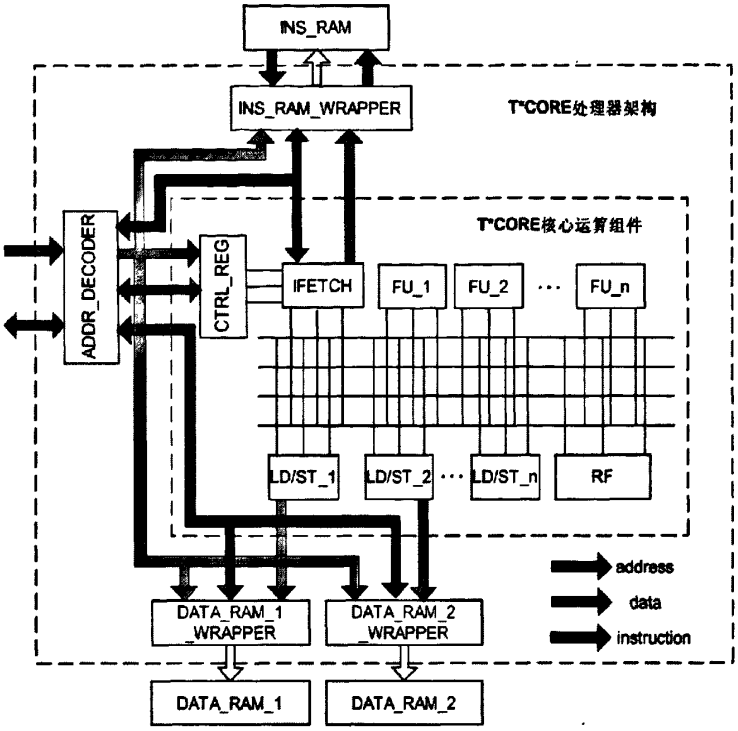


图 4-1 T*CORE 处理器模板架构

4.2 T*CORE 核心运算组件设计

T*CORE 核心计算组件是 T*CORE 处理器最为主要的部分，在控制寄存器 CTRL_REG 的控制下完成所有计算任务。T*CORE 核心运算组件设计主要由控制寄存器 CTRL_REG、指令控制单元 IFETCH、功能单元 FU、数据访问单元 LD/ST、通用寄存器堆 RF、Socket 以及内部互连网络等几部分组成。在核心计算组件中，控制寄存器 CTRL_REG 用于 T*CORE 协处理器与外部主处理器之间进行通信，控制 T*CORE 的运行。指令控制单元 IFETCH 用于从指令存储器中读取指令，并译码出控制信号。功能单元 FU 负责各种通用或特殊计算，如加、减、乘、FMAC、循环操作等。数据访问单元 LD/ST 也可看成一种 FU，负责从数据存储器中获取计算所需数据或将计算所得结果写回到存储器中。Socket 可看成交叉开关，分为 input Socket 和 output Socket 两大类，input Socket 负责将数据从数据总线传输到相应的 FU 当中完成计算，而 output Socket 则负责将 FU 的计算结果传输到数据总线之上。内部互连网络由三类总线构成：识别码（identifier）总线、数据总线以及控制总线，控制总线又分为 guard bus 和 lock bus。识别码总线传输译码所得的选通信号，数据总线传输计算所需数据，guard bus 用于判断

所选操作是否执行，lock bus 用于停止整个 T*CORE 的运行。每个组件的具体逻辑设计将在下一节进行详细介绍。

4.3 可配置可扩展 T*CORE 处理器详细设计

T*CORE 处理器是本文针对天文图像空间变换核降晰处理而设计的一款基于 TTA 架构的可配置可扩展处理器。T*CORE 并非某一具体固定指令集处理器，而是可以根据应用需求选取并配置处理器各关键组件，并依据相应地标准接口将组件进行适当组合和信号连接，构成面向某一具体应用的处理器。由于 T*CORE 具有模块性，可扩展性以及可配置性的特点，使其非常适合作为协处理器配合功能强大的通用处理器完成计算数据密集型的任务[26]。根据第三章对空间变换核降晰算法分析，本文设计了一种基于这种特定功能的 T*CORE 处理器架构。T*CORE 协处理器内部架构图如图 4-2 所示。

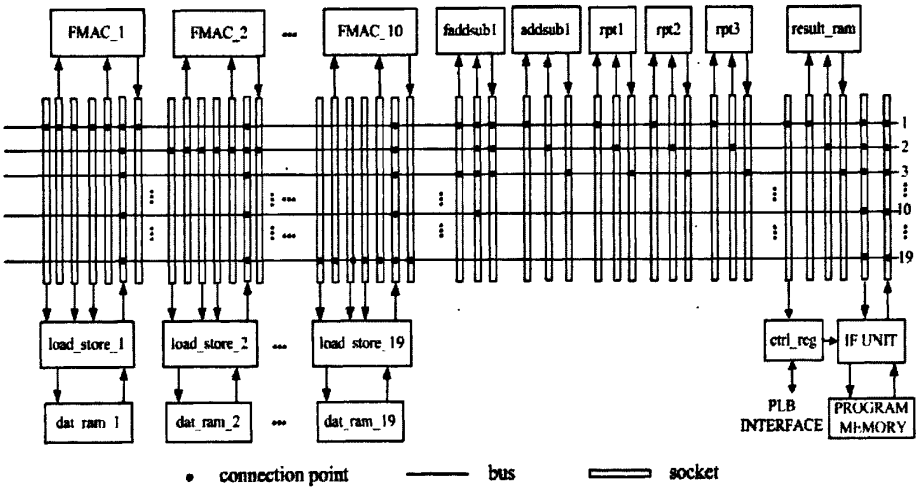


图 4-2 T*CORE 协处理器内部架构

其中，FMAC 单元可以在 2 个时钟周期内完成一次乘累加运算，也可以单独做加法器使用。根据空间变换核矩阵的运算，本设计采用了 19 条总线，10 个 FMAC 单元并行计算的架构。dat_sram_i 中保存欲处理图像数据及对应空间变换核矩阵数据，result_ram 用于保存处理结果。load_store_i 单元用于从对应的第 i 个 dat_sram_i 中存取数据。rpt_i 单元用控制循环操作，而且循环操作是可以嵌套的，本架构使用三个 rpt 单元来实现三层循环嵌套操作。program memory 用于存放执行降晰运算的 T*CORE 汇编代码，控制 T*CORE 处理器的执行流程，其中可通过对 ctrl_reg 赋值启动和暂停 T*CORE 处理器及判断运算是否执行完成操

作，T*CORE 处理器也支持中断操作，这样可以方便对多个 T*CORE 处理器并行执行的操作进行控制。各功能单元 FU、寄存器堆 RF 通过 socket 与总线连接，彼此交换数据。因为总线与各 FU 之间通过 socket 构成松耦合的连接模式，为了节省硬件资源，本文 socket 连接部分采用部分连接方式，除 fmac_r 采用全连接外，其它大都采用单连接或部分连接，这样不仅不会影响功能，还可以大大节省硬件资源，这非常适合于可配置处理器的理念，做到量身定制，最大限度的发挥处理器的性能。

本文设计 T*CORE 处理器关键一点是如何设计符合算法需要的 FU 功能单元，用来作为一个专用的硬件加速器，加快算法执行速度，以满足实时性的要求，因此本文在下面几个小节重点介绍了如何设计各 FU 功能单元。

4.3.1 FMAC 单元模块设计

该单元主要用于完成执行降晰运算中大量存在的乘累加操作，同时为了共享硬件资源，FMAC 单元也可以单独作为乘法器作用，其结构示意图如图 4-3 所示。通过剖析图像相减算法的空间变换核降晰算法及 3.5.2 可知，需要自定义一种新型的 32 位浮点数格式，可根据浮点数的运算规则进行硬件设计。此外，根据基于传输触发体系结构 T*CORE 处理器的编程特点，启动 FMAC 操作后，计算所需的数据被依次连续的送入 FMAC 功能单元，当全部数据都被输入后，经过 2 个时钟周期的延时后，便可从结果寄存器中取出计算结果，因此 FMAC 单元的本身固有延迟为 Latency(FMAC) = 2。

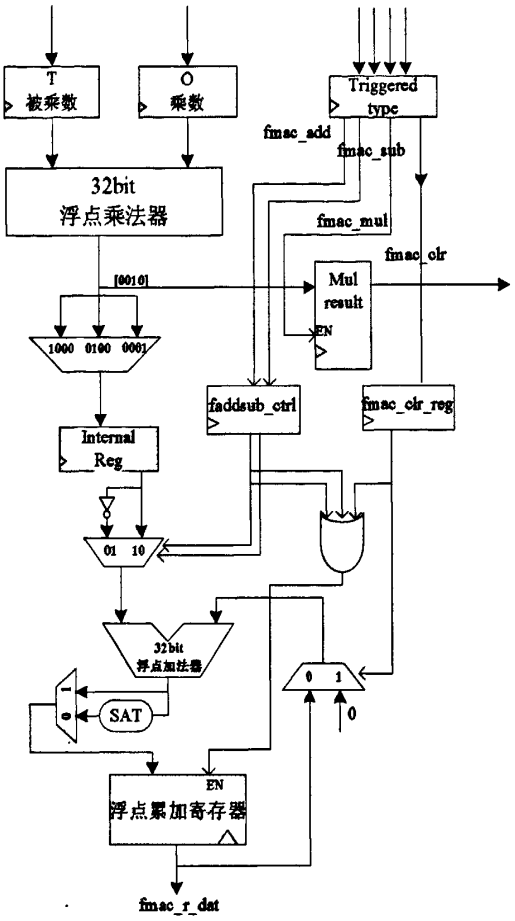


图 4-3 FMAC 硬件结构示意图

FMAC 模块主要是由 FADDSUB 模块和 FMUL 模块组合而成的，因此也可以单独进行浮点乘法和浮点加法操作。完成一个完整的 FMAC 操作需要 2 个时间周期。该 FU 的寄存器包括：

1. Operand 寄存器，用来存放乘数。

2. Trigger 寄存器，用来存放被乘数，它可以触发 3 种运算，分别是：

(1) 乘累加（fmac_t_add），用累加寄存器的值加上乘法的结果。

(2) 乘累减（fmac_t_sub），用累加寄存器的值减去乘法的结果。

(3) 乘法（fmul_t_mul），一般的乘法运算，这个乘法结果不会累加寄存器的值，但会把原来累加寄存器中的值清除为 0，一般也用来作重新进行乘累加运算时的初始化过程。

3. 结果寄存器（fmac_r），用来寄放浮点运算的结果值。

对于 FADDSUB 和 FMUL 模块的设计，只要根据浮点数的加、减、乘的计算流程，对指数和尾数分别进行运算，就可以得出最后的浮点计算结果。

4.3.2 RPT 单元模块设计

该单元主要用于完成程序流控制操作，其作用等价于高级语言中的循环功能，这可大大方便编写降晰运算中多循环操作功能函数的汇编代码，而且 rpt 功能单元本身面积并不大，因此利用此功能单元还可以节省总的硬件资源。

RPT 的结构示意图如图 4-4 所示。

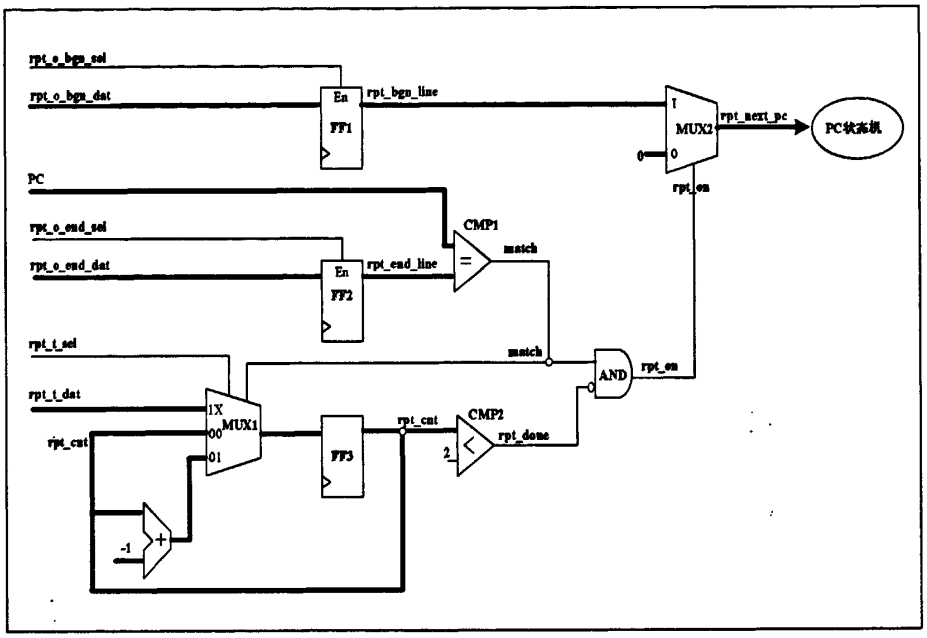


图 4-4 RPT 硬件结构示意图

该 FU 的寄存器包括：

1. Operand 寄存器（rpt_o_bgn 和 rpt_t_end），用来存放循环开始标识符和循环结束标识符。

2. Trigger 寄存器（rpt_t），用来存放循环次数，并触发循环操作。

由 4.3.5 节可知，数据存储和运算过程有规律，从而可以大大减少搬数和取数的时间。因为在对每一块数据进行卷积运算时，操作过程都是一样的，只是所需计算的数据不同。因此我们可以通过循环，来简化操作，从而缩短汇编代码的行数，节省 PROGRAM MEMORY 的存储空间。本架构采用三层循环嵌套，每个 RPT 的作用如下：

(1) rpt1 用于控制每次进行卷积运算的块数。

(2) rpt2 用于控制在计算一个像素点时，所需要进行 19 次浮点乘累加运算。

(3) rpt3 用于产生 19 个 rpt2 运算出来的像素点。

4.3.3 LOAD/STORE 单元模块设计

该功能单元除了传统意义上对于数据存储中数据的读取之外，根据天文图像空间变换核降晰算法包含大量对于数组的连续存取操作提供了直接和偏移寻址模式以支持对于数组数据的快速存取，从而提高处理器性能。

LOAD/STORE 的结构示意图如图 4-5 所示。

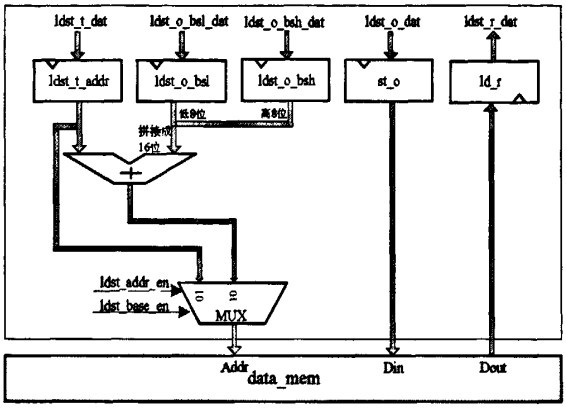


图 4-5 LOAD/STORE 硬件结构示意图

LOAD/STORE FU 主要用于访问 T*CORE 数据存储器，它支持直接寻址和基址寻址的寻址方式。完成一个完整的 LOAD 操作需要 3 个时钟周期，而完成一个完整的 STORE 操作只需要 2 个时钟周期。该 LOAD/STORE 支持 2 种触发方式，对应于图中 MUX 的两个片选信号 ldst_addr_en（一维绝对寻址）、ldst_base_en（一维基址寻址），并且，基址由两个 operand 寄存器 ldst_o_bsh 和 ldst_o_bsl 拼接而成，以支持更大的寻址范围（单独一个基址 operand 寄存器寻址只有 256，两个寄存器值拼接后寻址范围为 65536）。该 FU 的寄存器包括：

1. Operand 寄存器 1（ldst_o），用来存放存储操作中的数据。

2. Operand 寄存器 2（ldst_o_bsh、ldst_o_bsl），用来存放基址地址。

3. Trigger 寄存器，用来存放偏移地址，同时能触发 4 种操作：

(1)直接存储（ldst_t_sta），这个操作将 Trigger 寄存器中的值作为地址，将 ldst_o 中的值作为数据，存储到数据存储器中。

(2)基址存储（ldst_t_stb），这个操作将 Trigger 寄存器中的值和{ldst_o_bsh,ldst_o_bsl}中的值相加作为地址，将 ldst_o 中的值作为数据，存储到数据存储器中。

(3)直接取值（ldst_t_lda），这个操作将 Trigger 寄存器中的值作为地址，从存储器中取出数据。

(4)基址取值（ldst_t_ldb），这个操作将 Trigger 寄存器中的值和{ldst_o_bsh,ldst_o_bsl}相加的值作为地址，从存储器中取出数据。

4. 结果寄存器（ldst_r），延时 2 个时钟周期后，将 ldst_t_lda 和 ldst_t_ldb 中的数据值取出来，送入下一个操作指令中。

本文设计的 LOAD/STORE 功能单元用 trigger 寄存器用于存放偏移地址，这样在进行 T*CORE 程序设计时每次只需更改 trigger 寄存器的值，而不需更改 operand 寄存器中的基地址，即可高效地完成对于数组的依次连续存取，相比使用传统功能的 LOAD/STORE 单元完成对于数组的依次存取，所消耗时间节省 30%左右。

4.3.4 通用寄存器（R0~R8）

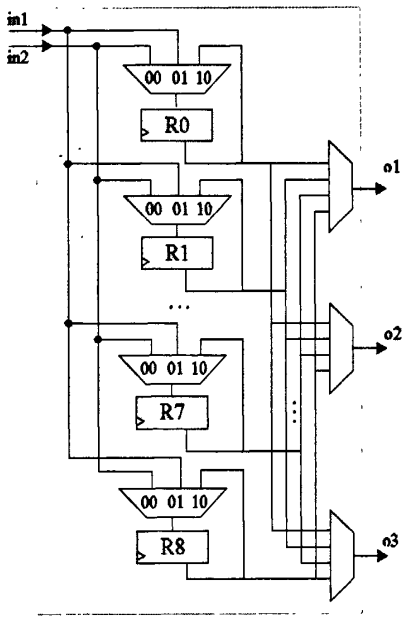


图 4-6 通用寄存器堆 RF 结构示意图

通用寄存器堆是一种特殊的 FU，用来缓存运算过程中的临时数据。在 RISC 架构中，中间运算的结果必须先存回通用寄存器里，这样就大大加重对了通用寄存器的依赖，当通用寄存器个数不够用的时候，性能会大大的降低。TTA 架构允许 FU 之间直接旁路数据传递，例如把 ldst_r 的结果直接送到 fmac_o 中，不用再经过寄存器堆传递，这样就大大提升了任务的执行效率，同时对通用寄存器的依赖也降低到最小[27]。考虑到本论文的 T*CORE 架构采用 19 条总线，19 个 LOAD/STORE 单元，10 个 FMAC 的结构，因此通用寄存器堆被设计成包括 9

个通用寄存器（R0~R8），用来存放运算过程中需要临时暂存的数据。

当然，当通用寄存器个数不够的时候，中间结果会通过 LOAD/STORE 单元暂存到数据寄存器中，这样就会影响代码执行的效率，又因为 T*CORE 是可配置可扩展处理器，灵活性强，因此在设计的时候充分考虑到需要用到的通用寄存器个数，做到量体裁衣，这样就可以避免出现不够用的情况，代码执行效率也可以大大提高。通过编写代码可以知道，9 个通用寄存器是完全够用的。

4.3.5 数据存储方式及计算流程

本设计根据所采用的降晰运算的规律，分析了数据复用的可行性，在设计 T*CORE 处理器时，采用了下述特殊的数据存储方式。

定义如下符号：

R_i (i=1,2,…,37)：图像第 i 列数据。

K_i(i=1,2,…,37)： kernel 第 i 列数据。

R_i+R_ +j：将图像 R 第 i 列数据与第 j 列数据按先 i 后 j 顺序存储，列中顺序不变

~K_i：逆序排列的图像 K 第 i 列数据。

DATA_RAM_i_j: DATA_RAM_i 中第 j 个数据。

DATA_RAM 中数据存储方式为如下：

DATA_RAM_i 中数据：R_i + R_(i+19)+(~K_i)，i=1,2,…18。

DATA_RAM_19 中数据：R_19+ R_19 +(~K_19)。

DATA_RAM 中存放图像数据和 kernel 数据两部分，数据至上而下上编号为 1-93。数据存储方式如图 4-7 所示。

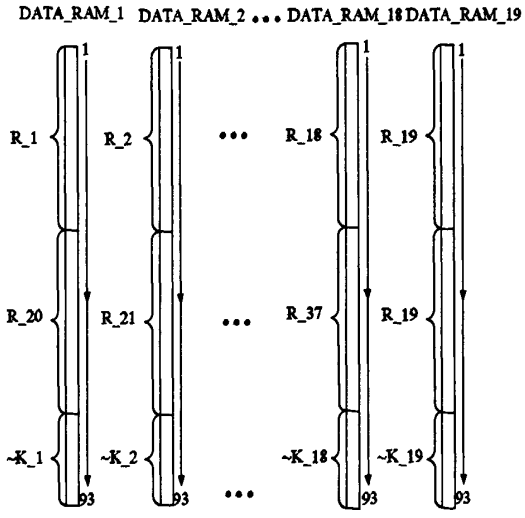


图 4-7 T*CORE DATA_RAM 中数据存储方式

计算流程如下：

（1） Offset1=0，Offset2=0。

（2） 从 DATA_RAM_i 中地址为 74（~K_i 起始位置）、DATA_RAM_(i+Offset1)中地址为 Offset2 的位置（图像数据起始位置）开始连续取 19 个数送至 MAC_i，执行乘加运算。其中，i=1,2,…,19 记 n=i+Offset1，若 n>19,则 n=n mod 19 且 DATA_RAM_(i+ Offset1)中图像数据地址为：Offset2+37。

（3） 19 个 MAC 中计算结果两两相加，将值保存于 RFs。

（4） Offset2++。

（5） 循环执行(2)、(3)、(4)18 次。

（6） 将 RFs 数据传至主控制器，Offset1++。

（7） 循环执行(2)、(3)、(4)、(5)、(6)18 次。

T*CORE 计算过程中数据地址变化如图 4-8 所示。

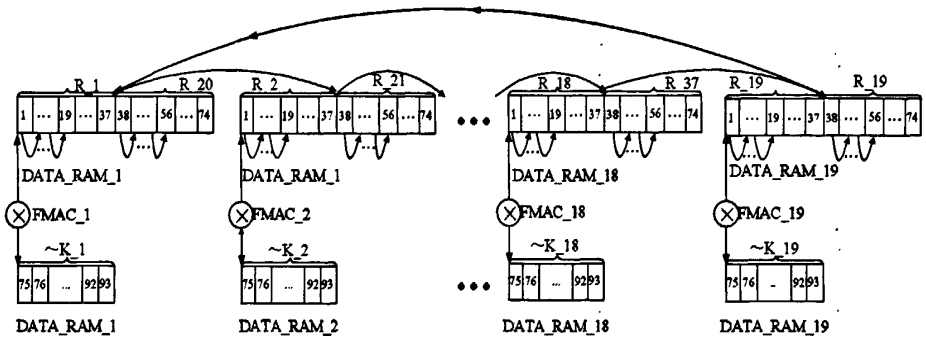


图 4-8 T*CORE 数据地址变化

依据原算法，假如核大小为 N×N，计算 1 个点主控制器需从外存储器搬 N×N 个点，而依据上述控制逻辑平均一个点只需从外存储器搬用 $(2N-1)^2/N^2 \approx 4$ 个点，数据复用率为原算法 $N^2/4$ 倍。

4.3.6 T*CORE 处理器电子系统级建模

本节主要是应用 T*CORE 处理器电子系统级建模工具，来生成硬件架构，并进行功能仿真，选择最优的硬件架构。此建模工具是面向可配置可扩展 T*CORE 处理器的周期精确位精确（Cycle Accuracy and Bit Accuracy，CABA）SystemC 电子系统级模型。此模型将程序设计中的面向对象思想融入到 T*CORE 处理器的建模当中，使模型具有模块化、松耦合、易扩展等优势，解决了由于可配置可扩展处理器架构的灵活性给建模所带的复杂度和难度。T*CORE 处理器的

电子系统级模型不仅可以作为标准的指令集仿真器（Instruction Set Simulator，ISS）进行功能仿真，而且由于其精确模拟了 T*CORE 处理器的所有硬件行为和时序关系，因此该高层次抽象模型可以提供不同架构之下各种处理器性能、成本指标，便于用户进行空间探测，决定最优架构。

本文通过建模工具配置生成的 T*CORE 架构模型采用了 19 条总线、10 个 FMAC 单元、19 个 LOAD_STORE 单元、19 个 DAT_SRAM 单元、1 个 FADDSUB、1 个 ADDSUB、3 个 RPT、1 个 RESULT_RAM、1 个 PROGRAM_RAM、通用寄存器堆以及一些外围控制接口组成。总线 socket 采用部分连接的方式。通过编写 T*CORE 处理器的汇编代码，利用 ISS 仿真器进行功能仿真，对所设计的 T*CORE 处理器架构进行功能验证及性能评估。不仅解决了由于可配置可扩展处理器其本身架构的多样性给建模带来的困难，而且相比传统的基于结构化编程思想的指令集仿真器设计又大大提高了仿真速度。该仿真工具提供的大量参数指标，可用于选择最优架构配置方案。

表 4-1 给出了在这种架构配置方案下的 T*CORE 处理器运算图像降晰算法时所得到的各方面参数指标。

测试1	基本参数指标					
	代码量 (行数)	周期数	寄存器读	寄存器写	旁路	
	624	107626	276222	15219	89.4%	
测试2	FU利用率					
	bus1	fmac	faddsub	addsub	ldst	rpt
	98.7%	45.78%	1.09%	5.91%	87.10%	0.13%

表 4-1 本架构配置方案的基本参数指标及 FU 利用率

其中测试 2 给出的只是部分 FU 的利用率，通过 ISS 仿真器建模工具可以得到更详细的性能评估数据。利用率是指单个 FU 触发操作的次数/所有的 FU 触发操作次数。

4.3.7 基于 T*CORE 处理器的汇编器

本节主要论述基于 T*CORE 处理器的汇编器工具，该汇编器工具可以把 T*CORE 汇编代码转换成 T*CORE 可识别的二进制代码，以便对 T*CORE 处理器进行操作。因此，要把图像降晰算法的 C 代码转化为 T*CORE 汇编代码，根据 T*CORE 相应的指令格式编写代码，并做适当的优化处理。

本文的 T*CORE 框架的每条指令长度是 20bits，共 19 条总线，所以指令 memory 宽度是 380bits，一条总线类似于指令格式的一个 slot，所以一个 cycle 可以并行执行 19 条总线上的指令操作。由于 T*CORE 处理器采用了 TTA 结构，因此实质上在 T*CORE 内部仅仅包含一种操作，即数据传输操作。所有的复杂操作都被分解成数据传输操作，当数据送到相应的 FU 功能单元后就触发相应的运算操作。T*CORE 处理器在一条指令中打包多个数据传输。T*CORE 处理器的指令格式如图 4-9 所示。

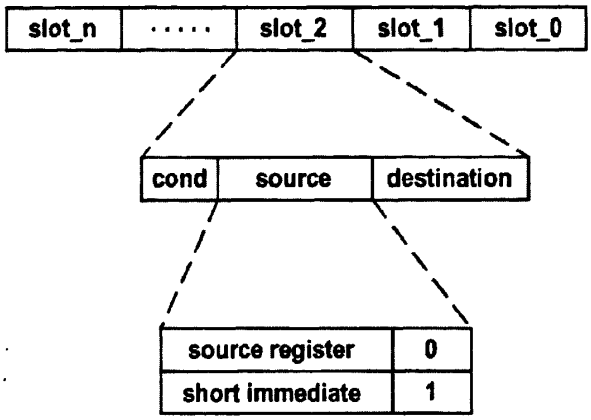


图 4-9 T*CORE 处理器指令格式

T*CORE 处理器指令格式采用小端格式（little endian），即指令最低有效位存储在低地址空间中。有多少条总线，就有多少个 slot，每个 slot 单元由 cond 段、source 段和 destination 段组成。其中 cond 段用于判断 source 段是短立即数还是源寄存器编码，source 段和 destination 段用于表示数据传输的源地址和目的地址。

正确编写完 T*CORE 汇编代码后，利用汇编器工具生成二进制代码。首先读取上节提到的仿真器工具生成的架构配置信息，进行架构解析；然后再读取并行汇编代码，就可以生成 T*CORE 可执行的二进制代码。

第五章 天文图像空间变换核降晰算法的SoC系统设计

本章主要针对图像降晰计算部分通过专用硬件模块进行实现，对算法进行软硬件功能划分及接口定义。并采用了Xilinx公司的SoC设计平台开发板Virtex-5，该开发板即为一个完整的SoC系统，可以将以T*CORE处理器为协处理器的SoC构架平台移植到开发板的FPGA当中，这样就可以对系统的功能进行验证。

5.1 算法的软硬件功能划分

图像相减测光算法从功能上分为：确定PSF，确定天区、对齐、图像相减、定标以及产生星表等。为了充分发挥专用硬件的计算优势、加快计算速度以满足实时处理的需要，需要对上述功能进行软硬件的划分，提取对算法性能影响较大的瓶颈功能，以确定哪些功能由软件完成（PC机），哪些功能由专用硬件加速器完成。主要遵循的划分原则是软件主要进行并行度低，计算量小但控制灵活的工作，而硬件加速器完成并行度高计算量大的工作。

图像相减测光算法目前已经有了多种实现，本课题针对专用硬件设计的特点选取并行度高，数据类型和结构便于硬件处理的算法进行硬件移植，通过快速处理单元、高速I/O以及并行存储的设计达到实时处理（2.4分钟处理200M图像数据）要求，同时有效控制功耗。

首先对选取算法进行适当修改与优化，主要表现在两个方面：代码结构优化与数据类型修正。前者主要通过循环拆解（loop unrolling）等方法去除不必要的代码分支，加大程序本身的并行性。后者将在不影响功能及处理效果的前提之下，将一些复杂数据类型转换为简单数据类型，将一些浮点数定点化，以此进一步降低计算复杂度，为硬件加速性能提升打下坚实基础。

在修改优化后的代码之上，通过大量实验分析出该算法对实时性影响最大的部分，并以此基础确定软硬件的划分。针对该实时性的瓶颈部分，进行功能分析，将其分解为独立的便于流水处理的功能模块。此外，还需针对此部分算法的数据流进行分析，将对整幅图像的处理划分为对各个独立子图的处理，进一步加大并行性。

在软硬件功能划分、功能并行性分析的基础之上，使用硬件描述语言对需要

专用硬件处理部分进行 RTL（Register Transport Level，寄存器传输级）描述，完成功能设计，借助已有的成熟的 EDA 工具完成仿真、综合、布局布线等硬件设计工作，从而确保此硬件在功能上的正确性。

根据算法数据流分析的结果，利用多个图像差异分析算法专用硬件处理单元搭建并行处理架构，使其能够完成对整幅图像的同时并行处理以及多幅图像的并行处理。

依据逻辑运算单元的数据并行性，完成并行存储系统的设计，使数据并行存取能力与数据运算能力相匹配，充分发挥硬件加速器的性能。

在保证了专用硬件设计实时处理的前提之下，需要确定与 PC 机的互连接口。此接口应为高速互连接口，能够保证图像数据以一定的速率高速稳定地被传输到专用硬件/传回 PC，保证专用硬件的高利用率、流水线不停顿、不断流，从而在 I/O 层面上确保实时处理的需要，本论文采用基于 PCI-Express 接口的设计方案，通过即插即拔的方式与 PC 机进行互连。

所用的硬件相关设计工作将主要基于 FPGA（Field Programmable Gate Array，现场可编程逻辑门阵列）实现。通过 FPGA 可以快速搭建系统原型、完成相应的功能验证，其可配置、可编程性大大提高了设计效率，降低了设计风险。而本文的专用硬件设计目前主要针对于天文图像相减算法中执行空间变换核降晰运算部分，即卷积运算，而今后可能根据需要还要进行适当的扩充，增加新的功能，而 FPGA 设计的灵活性保证了这种不确定设计，仅需适当的修改与扩充即可进行快速的系统验证。

从 3.5 节的表 3-1 以及天文图像空间变换核降晰算法运算的特点可知，执行降晰运算的执行时间占总时间的 74.3%，并且都是计算密集型任务，因此数据处理的瓶颈在于执行降晰运算，即卷积运算，而这些运算主要是进行乘累加运算，非常适合于硬件实现。因此，我们把这部分数据量大，计算复杂度又高的运算移植到开发板的 FPGA 中进行计算。对此，本文设计了一种基于 T*CORE 可配置可扩展处理器的 SoC 构架平台。其中 T*CORE 处理器作为协处理器，用于执行降晰运算，处理数据密集型的任务。由于执行降晰运算是图像降晰算法的最后一步，而且前面三步运算量并不大，数量并行性也不是太高，所以可以把这三步都放在软件上实现，并对其进行代码结构的优化和数据类型的修正，以实现软件层次上的优化。最后通过 PCI-E 把 PC 机与开发板互连，进行数据交互。

由于天文图像空间变换核降晰算法所需计算的数据量特别大，为了不使搬运数据的时间影响图像降晰计算的总时间，因此必须使用一种高速传输接口，在规定的时间内完成图像数据搬运操作，而 Virtex-5 的 PCI-E 接口的传输速度能达到 312.5Mb/s，完全能满足 I/O 层次上的实时性要求，详细的 PCI-E 端口模块设计

见 5.3.1。具体的数据流向如图 5-1 所示。

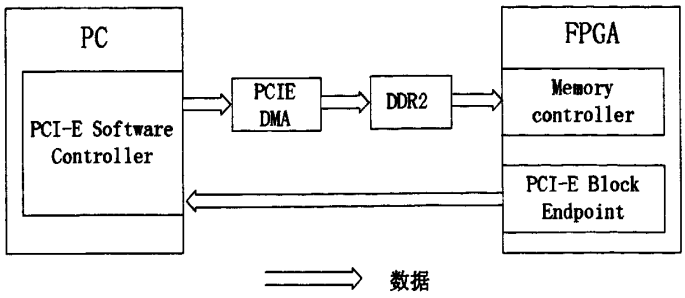


图 5-1 数据交互图

首先把在 PC 端计算完成的前三部分数据读入内存，做成一组数据流，发送到 PCI-E 端口的接收模块，然后通过 PCI-E 端口的发送模块把这些数据传送到开发板的 DDR2 中，待这些数据传送完成后，MicroBlaze 核会接收到发送完成中断请求信号，然后 MicroBlaze 会给 T*CORE 协处理器发送开始进行运算信号，T*CORE 处理器就进行最后一步执行降晰运算。待计算完成后又会给 MicroBlaze 发送计算完成中断请求信号，通知主核已经完成这部分的运算。最后再把计算完的数据通过 PCI-E 的发送端口传回到 PC 主机端，完成整个流程的运算。

5.2 SoC 系统架构平台设计

针对空间变换核降晰运算数据量大的特点，本文设计了基于 TTA 架构的可配置可扩展处理器作为协处理器的 SoC 实现方案。其 SoC 架构平台如图 5-2 所示，基于 TTA 架构的可配置可扩展处理器 T*CORE 将完成空间变换核降晰运算，CPU 核主要用于控制，硬件系统通过 PCI-E 接口与 Host(PC 机)进行数据传输。系统工作时 CPU 通过 DMA 将存放在 DDR2 中的一次计算图像数据送入 T*CORE，等到 T*CORE 中数据处理完毕，CPU 先将处理结果经 PCI-E 传至 Host，然后，将新的数据送至 T*CORE 依次循环，直至图像数据处理完毕。

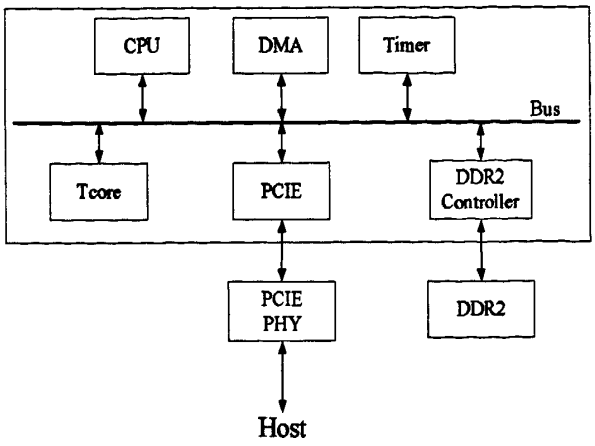


图 5-2 硬件系统 SoC 结构图

5.2.1 PCI-E 接口模块设计

PCI Express（PCI-E）是一种强大可靠、可伸缩、灵活并且合乎成本效益的 I/O 互连方式，采用目前业内流行的点对点串行连接，比起 PCI 以及更早期的计算机总线的共享并行架构，每个设备都有自己的专用连接，不需要向整个总线请求带宽，而且可以把数据传输率提高到一个很高的频率，达到 PCI 所不能提供的高带宽。PCI Express 标准是一个高性能的通用互连架构，可用于多种计算和通信平台。它是一种基于包的点到点串行接口，支持每方向每通道 2.5Gbit/s 的原始带宽。其可靠且基于信用度的内置流量控制可消除因接收缓冲器溢出而导致的丢包问题，而重试功能则可确保数据传输的高度可靠性。一个 PCI Express 连接可以被配置成 x1，x2，x4，x8，x12，x16 和 x32 的数据带宽。x1 的通道能实现单向 312.5 MB/s(2.5 Gb/s)的传输速率。Xilinx 公司的 Virtex5 系列 FPGA 芯片内嵌 PCI-Express Endpoint Block 硬核，为实现单片可配置 PCI-Express 总线解决方案提供了可能。

Xilinx 公司提供的 Endpoint Block Plus for PCI-E 解决方案适用于 Virtex-5 FPGA 架构，是一种可靠的高带宽可缩放串行互连构建模块。核例化 Virtex-5 器件中的 Virtex-5 Integrated Block for PCI Express，支持 Verilog HDL 和 VHDL 两种语言。其顶层功能模块如图 5-3 所示，从中可以看出，核接口分为系统（SYS）接口、PCI Express（PCI_EXP）接口、配置（CFG）接口以及事务（TRN）接口。

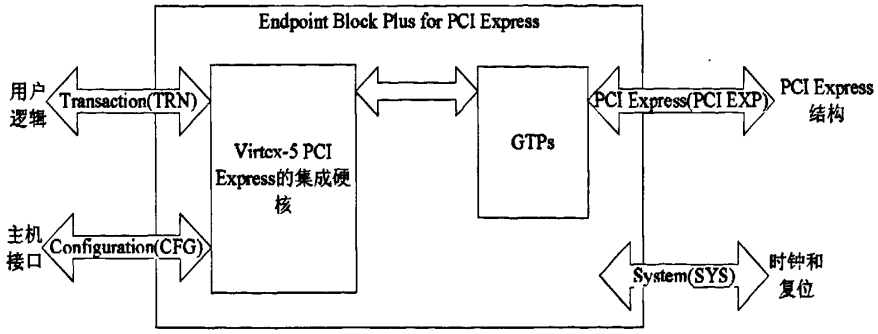


图 5-3 顶层功能模块和接口示意图

本文在研究 PCI-Express 接口协议和 PCI-Express Endpoint Block 硬核的基础上，使用 Virtex5LX110T FPGA 芯片设计 PCI Express 接口硬件电路，其接口硬件电路设计如文献[28]所示，实现 PCI-Express 数据传输。

按照文档中的所描述步骤，对 PCI-E 接口模块进行配置，最后可以生成一个 config 文件夹，里面包含有两个文件一个是 xilinx.sys 文件，另一个是 rev6.ace 数据包。把这两个文件拷贝到 CompactFlash 卡里，通过开发板上的 CF 卡接口与开发板相接，以便主机通过读取这些配置信息来识别开发板。这样就可以为下一步进行 PC 与 FPGA 相互传输数据做好准备。PC 端通过 PCI-E 往 FPGA 搬数，需要编写软件程序实现，所以要实现这些操作，就必须掌握 PCI-E 协议，编写相应的程序来实现。由于时间和精力的限制，在本论文中并未实现，只是采用 PCI-E 专门的互连工具 PciTree 来代替实现。该工具完成可以实现从 PC 往 FPGA 内部传输数据，在 FPGA 端由于有专门的 IP Core，因此往 PC 端传输数据就可以通过其提供的 API 函数予以实现。

在实现数据传输过程中，数据从 PC 内存通过 PCI-Express 接口向下传输到 FPGA 内部，FPGA 内部 DDR 控制逻辑再将数据传输到的 DDR 内存芯片中存储，向下传输完毕后，FPGA 内部逻辑从 DDR 芯片中将存储的数据读出，并且给每个数据按字节加‘1’，然后通过 PCI-Express 接口，再将数据传回 PC 内存，PC 内存程序对数据进行校验。

5.2.2 T*CORE 用户自定义 IP Core 添加

嵌入式系统的开发基本是由 XPS 自动完成的，本节介绍如何在 XPS 中添加用户定义设备的操作。虽然外部设备各类繁多，总线接口也不尽相同，XPS 仍给出了相应的向导来简化此过程。

在系统组件面板中的总线接口给出了总线、处理器和 IP 间的互连关系。用户创建的任何 IP 都必须适应已生成的系统，为满足这一条件，必须做到两点：

1.确定 IP 所需要的接口，对于用户定制的外围设备，必须指出它们所连接的总线，如处理器本地总线 PLB；2.实现和验证定制的功能，IP 必须导入到 EDK，必须将外围设备加入到 XPS 创建的处理器系统中。

用户自定义 IP Core 与处理器系统的互连是通过系统提供的 IP 接口（IP interface，IPIF）实现的。IPIF 是个已经被验证、最优化的并且参数可调的接口，同时它提供了一系列简单总线协议，即 IP 互连（IPIC）。用户利用 IPIF 模块，使其参数匹配自己的需求，就可以极大地减少设计和测试的工作量。

本文自定义的 IP Core T*CORE 就是通过上述规则实现的。首先通过“Create or Import Peripheral”向导命令，定制生成一个模板 IP Core，并会生成相关的诸多文件，包括软件驱动和硬件结构代码。以上步骤只完成了 IP Core 的声明及其和 PLB 总线端口的连接，没有实现任何用户逻辑。因此用户需要在模板中添加用户逻辑，完成对 PLB 总线的响应。由于模板创建的是 CoreConnect 兼容结构，因此不需要为其添加任何额外的逻辑。对于定制 IP 而言，所做的只是对 user_logic.v 文件的修改。

对于 T*CORE 而言，需要把代码的顶层文件(tcore_chip)例化到 user_logic 模块中，并设置好 T*CORE 的读写时序，把相应的 T*CORE 信号线连接到模板 IP 提供的 IPIF 接口信号线上，完成对定制 IP 的生成。最后再把定制 T*CORE 添加到 XPS 工作中，就可以把 T*CORE 加入到 SoC 系统里。其 IP 核例化过程如下图 5-4 所示。

```
wire wr;
wire rd;
assign wr = Bus2IP_CS & (! Bus2IP_RNW );
assign rd = Bus2IP_CS & ( Bus2IP_RNW );

tcore_chip   tcore_chip (
             .clk_p(Bus2IP_Clk),
             .reset_b_p(~Bus2IP_Reset),
             .rw_p(Bus2IP_RNW),
             .cs_p(Bus2IP_CS),
             .addr_p(addrr[9:29]),
             .datai_p (data_in_val),
             .int_flag_p(IP2Bus_IntrEvent),
             .datao_p (dout)
         );

  assign IP2Bus_Data    = data_out_val;
  assign IP2Bus_WrAck   = wr;
  assign IP2Bus_RdAck   = rd;
  assign IP2Bus_Error   = 0;
```

图 5-4 T*CORE 模块例化

其中 int_flag_p 信号连接到 IPIF 中断信号上，由中断控制单元进行中断控制，当 T*CORE 处理器运算完成后会自动发出运算完成中断请求信号，就可自动触发进行下一次运算，而不需要用轮询方式一直查询 T*CORE 处理器是否运算完成，这样可以大大增加多核处理器的并行流水处理能力。

5.3 T*CORE 数据计算流程

在 SoC 系统平台主程序中，要编写程序对 PC 端与开发板端进行数据流控制，例如使用双核 T*CORE 进行并行计算时，其数据流程如下图 5-5 所示：

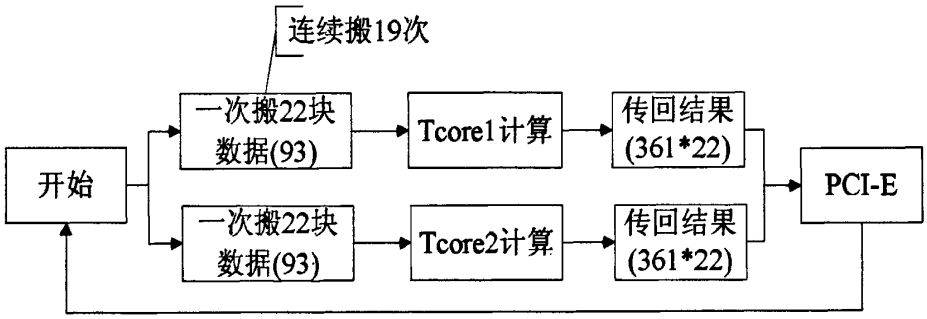


图 5-5 T*CORE 运算流程图

首先，通过 DMA 把 DDR2 中的图像数据搬送到 T*CORE1 和 T*CORE2 中，这一过程要连接搬送 19 次，每次 93 个数据，然后同时启动 T*CORE1 和 T*CORE2，计算完成后会自动发出运算完成中断请求信号，当 CPU 主核响应到这个中断请求信号后，再分别把存放在结果 RAM 里的运算结果通过 PCI-E 端口把结果传回到 PC 中，重新搬运新的图像数据进行下一次的运算，直至全部图像数据处理完毕。这种采用中断控制方式的多核处理器执行流程，可以大大提高并行流水处理能力，增强处理器的执行效率。

第六章 实验结果

本章主要阐述了如何通过分析比较，最终确定合适的 SoC 设计方案及 T*CORE 架构。并对最终的实验给出数据，来验证这种设计方案的合理性和可行性。

6.1 T*CORE 可配置可扩展处理器的数据统计

根据对天文图像空间变换核降晰算法流程的分析，可以知道执行降晰运算时，对图像进行分块卷积运算，每块计算完后，得到一块大小为 19×19 的像素区域。每次计算完成后，通过 DMA 再给 T*CORE 中搬数，进行下一次运算。为了减少 DMA 的启动和中断时间，可以通过增大 T*CORE 内部 RAM 方式，并把 19 个 RAM 设置成地址连续，这样就可以一次往 RAM 里搬送 22 块需要计算的数据，一次计算就可以得出 22 块计算后的值。然后通过 PCI-E 端口把计算后的结果值传回到 PC 主机端，再搬送新的 22 块数据进行下一次运算。

由于受到 Virtex-5 开发板资源的限制，因此设计 T*CORE 时就必须做到量体裁衣，而且这也正是 T*CORE 处理器的优势之一。下表为本文设计 T*CORE 综合时占用的开发板资源信息。

表 6-1 T*CORE 综合面积资源

Slice Registers	Slice LUTs	DSP48Es	BlockRAM	Total Memory (KB)
5585(8%)	26300(38%)	20(31%)	68(46%)	2384(45%)

下表 6-2 为设计 T*CORE 基本信息，以满足天文图像降晰算法的需求。

表 6-2 单核 T*CORE 处理器关键性能、成本参数

最大主频	代码行数	存储器面积(Bytes)		完成一次运算所需
(MHZ)		Data RAM	Instr RAM	的时钟周期
125	624	152K	32K	107626

总线Socket连接部分除了ldst_r部分采用全连接外，其它部分采用部分连接或单连接，这样不仅节省了面积，也降低了功耗。由于受到硬乘法器的限制，必须把原来的19次同时乘累加运算分成两次运算完成，这样运算时间就会增加，所以完成一个运算需要107626cycles，若有足够的硬乘法器资源，就可以同时进行19次乘累加操作，这样完成一次运算所需的时钟周期只需要83788。本文使用DMA进行搬数操作，搬一个数需要四个cycle。

下表6-3为单核和双核T*CORE运算的总时间统计情况，其中单核运算采用19个FMAC同时运算的架构，所以完成一次运算只需要83788个时钟周期。

表6-3 单核T*CORE和双核T*CORE运算时间统计

	传数（cycle）	计算（cycle）	传回（cycle）	总时间（cycle）
单核	184384512	977314896	118786176	1280485584
双核（RAM地址不连续）	87045080	629481530	17080840	733607450
双核（RAM地址连续）	82737240	629481530	17080840	729299610

其中，最大主频为125MHz，DMA发起和处理中断的时间为460cycles，双核比单核运算提高了1.75倍，仅T*CORE计算提高了1.553倍。所以使用双核T*CORE完成整个运算所需的时间为5.84s。

根据上文可知，天文图像空间变换核降晰算法占整个ISIS流程的51.02%，执行卷积降晰运算又占图像相减算法的74.3%，而根据实时性要求，一个总的ISIS流程时间不能超过2.4分钟，所以完成2K×2K图像的执行降晰运算必须在2.184s之内完成，所以必须设计利用多个T*CORE处理器并行处理的SoC架构来加快运算速度，节省时间，以达到实时性的要求。所以双核T*CORE的SoC架构还不能达到实时性的要求，而目前的Virtex-5开发板的资源只能挂两个T*CORE处理器，因此我们必须选择更大资源的FPGA开发板，本文选择Virtex-6 XC6VLX240T[29]型号的开发板，来设计一个新的SoC构架。

6.2 改进的T*CORE可配置可扩展处理器设计

由于硬乘法器资源够用，所以就可以同时进行19次乘累加运算，因此必须对原来的架构做一些小的改动，如图6-1所示。

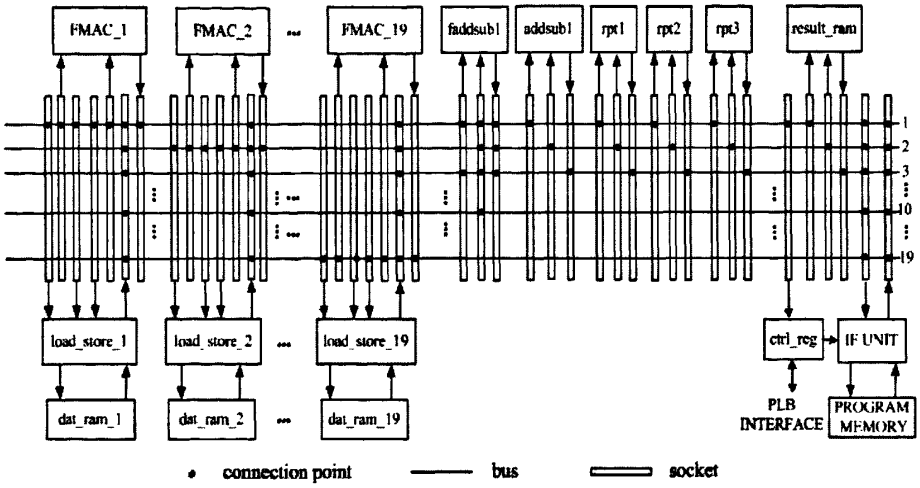


图 6-1 改进的 T*CORE 架构

和原来设计的架构相比，只是把原来的 10 个 FMAC 改为 19 个 FMAC 功能单元，别的设计还是一样。还需要对并行汇编代码做一些修改，同时利用 19 个 FMAC 功能单元进行运算，其它部分和原来的操作还保持一致性，这也正体现了 T*CORE 处理器设计的灵活性。表 6-4 给出了在改进的 T*CORE 架构下处理器的关键性能和成本参数。

表 6-4 单核改进的 T*CORE 处理器关键性能、成本参数

最大主频	代码行数	存储器面积(Bytes)		完成一次运算所需
(MHZ)		Data RAM	Instr RAM	的时钟周期
125	725	152K	32K	83788

6.3 T*CORE 可配置可扩展处理器设计仿真效果图

为了更快的验证 T*CORE 处理器设计的正确性，可以利用 Modelsim 仿真平台对 T*CORE 处理器的运算结果进行仿真，确保其结果的正确性。图 6-2 为 T*CORE 处理器仿真示意图。

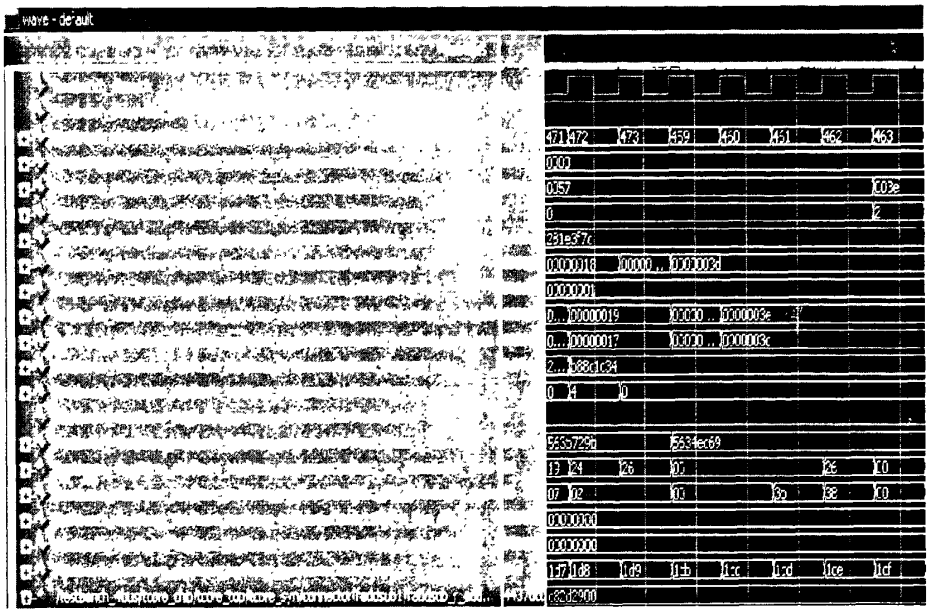


图 6-2 T*CORE 处理器仿真图

6.4 改进的 SoC 设计数据分析与结果统计

从 6.1 节可知，要想达到实时性，必须在 2.184s 之内完成 2K×2K 图像的执行降晰运算，因此本文设计 6 个 T*CORE 协处理器核并行运算的 SoC 架构设计，其最后的 SoC 设计图如图 6-3 所示。

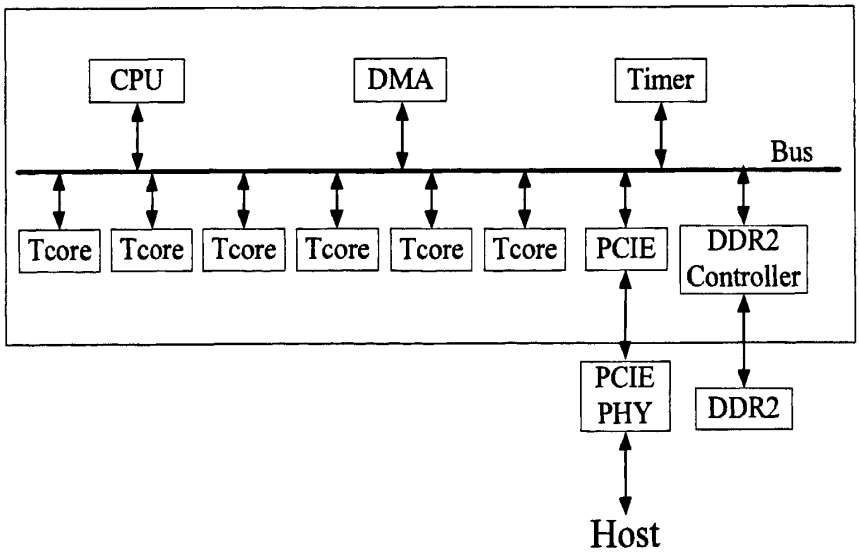


图 6-3 改进的 SoC 架构图

最后的运算时间如表 6-5 所示，其中 19 个 RAM 还保持地址连续的特点，这样可以节省传数的时间。由表 6-5 可知，在主频为 125MHz 时，完成整个图像的执行降晰运算流程总时间为 2.102s<2.184s，可以达到处理数据实时性的要求，功耗也远比用 PC 来完成运算要少的多。在 FPGA 上的实验结果表明，运用六核 T*CORE 处理器的 SoC 架构能够实现实时处理图像降晰算法的目标。

表 6-5 六核 T*CORE 的运算时间统计

	传数（cycle）	计算（cycle）	传回（cycle）	总时间（cycle）
六核（RAM地址连续）	82737240	162885008	17080840	262703088

通过使用嵌入式 FPGA 综合工具，改进的 SoC 架构所占用的 FPGA 资源面积如表 6-6 所示。

表 6-6 改进的 SoC 架构综合面积资源

Slice Registers	Slice LUTs	DSP48Es	BlockRAM	Total Memory (KB)
53343(18%)	174711(58%)	231(30%)	402(96%)	13904(93%)

虽然本文使用开发板配套的 XPS 综合工具对改进的 SoC 架构进行综合及布局布线，得到最后的资源使用情况，但由于还没有 Virtex-6 开发板以完成最后的实验，只是依据 Virtex-5 开发板所设计的双核 T*CORE 处理器的计算流程得到的计算时间进行仿真及估算，得到最后分析的结果，因此结果可能会因为总线负载过大而受到一定的影响。

使用 Xilinx 公司的 XPower 分析器，通过对本设计的在 FPGA 上的硬件实现进行基于实际布线结果的动态功耗分析。表 6-7 给出了本设计实现空间变换核降晰运算时的功耗以及所花的时间与 PC 上软件实现的功耗以及所花的时间，其中 PC 功耗仅计算处理器（Intel Core2 Duo CPU E7300）本身的功耗，数据来源于 Inter 公司官方网站。

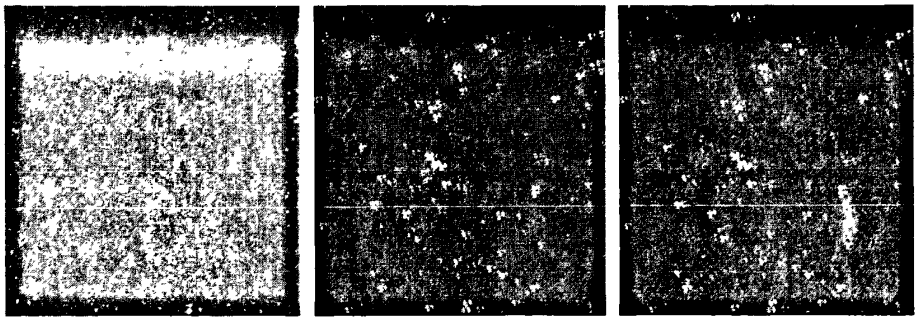
表 6-7 系统性能比较

方案	主频(HZ)	功耗(W)	计算时间(ms)
纯软件	2.66G	130	61062
本文设计	125M	3.88	2102

从上面分析可知，本文提出的设计方案处理效果在精度可以满足天文测光系统观测要求的同时使功耗大幅度降低，计算速度显著提高，达到天文图像处理快速低功耗处理的需求。

6.5 实验效果比较

本文设计方案处理的结果与通过 PC（Intel Core2 Duo CPU E7300 2.66G）纯软件处理的采用双精度浮点数据的结果相比较，其图像效果如图 6-4 所示，图像信噪比如表 6-7 所示。



a)模板图像　　b)PC 处理结果　　c)本文处理结果

图 6-4 图像效果对比

表 6-8 图像信噪比

SNR	绝对误差
89.471026	≤0.005

从图 6-4 及表 6-8 的结果可以看出，本设计达到了满意的处理效果。

6.6 多核 SoC 架构的探索

对于多处理器架构的多核 SoC 系统来说，可以知道一个很重要的定律，即 Amdahl 定律，它主要描述的是系统优化某部件，所获得的系统性能的改善程度，取决于该部件被使用的频率，或所占总执行时间的比例。Amdahl 定律存在三种性能模型：固定大小、固定时间和存储器限制模型。固定大小模型是指在计算问题规模不变的情况下，通过增加多核架构来提升系统性能加速比，但在这种模型下，有悲观的观点认为多核 SoC 系统架构可扩展性的规模的有极限的，而随着处理器的增加，加速比虽然也会线性增长，但增长是有一个极限值的[30][31]。但另一种乐观的观点认为，对于固定时间模型而言，通过设计多核架构，在固定的时间内，能够处理更多的任务，也就是说随着处理器的增加，系统性能加速比是会呈现线性增长的[32][33]。其固定时间模型加速比公式如图 6-5 所示。w 表示总时

间，f表示程序可并行部分所占的比例，m表示处理器的数目。从公式可以看出其加速比呈现线性增长的趋势。

$$\begin{aligned}\text{Speedup}_{FT} &= \frac{\text{Sequential Time of Solving } w'}{\text{Parallel Time of Solving } w'} \\ &= \frac{\text{Sequential Time of Solving } w'}{\text{Sequential Time of Solving } w} \\ &= \frac{w'}{w} = \frac{(1-f)w + fmw}{w} = (1-f) + mf.\end{aligned}$$

图 6-5 固定时间模型加速比公式

本论文所要解决的问题也是在固定的时间之内要完成整个天文图像降晰运算，因此所设计的多核 SoC 架构也符合 Amdahl 定律，随着 T*CORE 处理器数目的增多，处理速度也会呈现趋于线性增长的趋势。图 6-6 为本文设计多核架构的加速比分析情况。

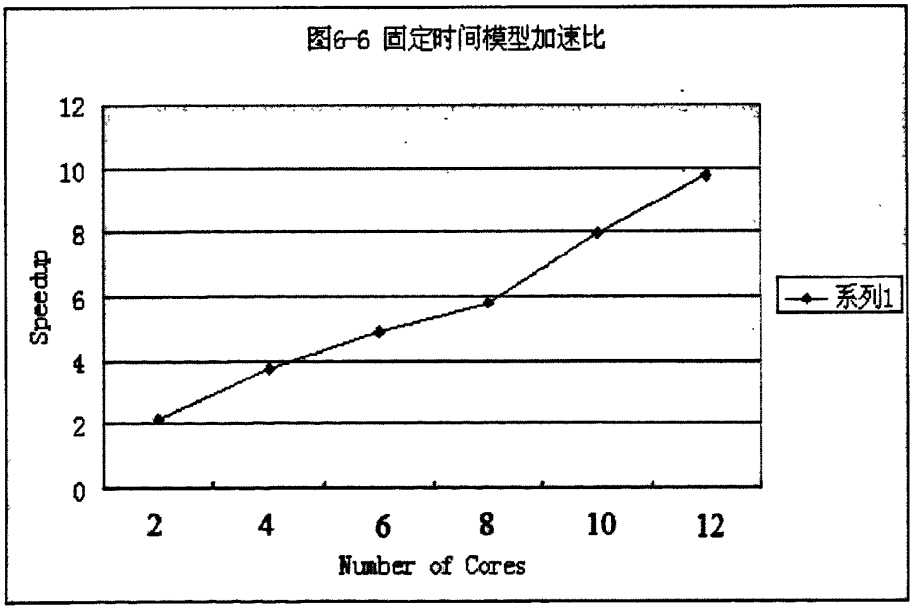


图 6-6 基于固定时间模型多核架构加速比

从图中可知，随着处理器核数目的增加，其加速比也趋向于线性增长的趋势，这一结论也进一步验证了本文中利用 6 个 T*CORE 处理器，获得更快运算速度的原因。

第七章 总结与展望

本文通过对天文图像空间变换核降晰算法的研究，实现了一种可配置处理器为协处理器的硬件架构，并利用并行处理的思想，设计了6个T*CORE可配置可扩展处理器并行运算的SoC硬件架构。T*CORE处理器是一款基于传输触发结构（TTA，Transport Triggered Architecture）的可配置可扩展处理器。TTA架构可以实现最大限度的指令集并行，能够在一个周期内执行多条指令，因此TTA架构非常适合数据密集型的计算任务。T*CORE很好的继承了TTA架构的这些优点，它能够实现单周期内执行19条指令，支持乱序执行，支持位数可变的立即数，兼有处理器的灵活性和ASIC的高效性，是在这两者之间全理的折衷。T*CORE可配置可扩展处理器可以根据不同的应用需求，设计不同的处理器架构，以达到处理器体系结构与应用需求的最优匹配。正是基于以上特点，本文设计了一种基于T*CORE处理器作为协处理器的SoC系统，以用来处理数据密集型的计算任务，并结合并行计算的思想，采用多个T*CORE处理器并行运算，加快数据的处理速度，最终实现实时性的要求。

设计了一种可复用的数据的存储方式，减小了主控制器从外部存储器取数频率，有效的解决了主控制器取数重复率过高的问题，将每点所需传输的数据从N×N点降至平均不足4点，数据复用率达到$N^2/4$。同时，通过对数据的分析将数据格式由IEEE-754标准双精度64位降为32位自定义浮点数，从而大大降低了硬件消耗及功耗并提高了计算效率。对于大量数据的传输，设计了一种高速的传输方式——PCI-E互连传送方式，大大缩短了主机与FPGA开发板之间的数据传送时间。实验结果表明，与PC纯软件处理相比本文提出的设计加速比达到了29.06且功耗仅为纯软件处理时的2.98%。

虽然本文对天文图像空间变换核降晰算法的SoC进行了设计实现，但仍存在很多问题亟待在未来的工作中去解决：

1. 提高T*CORE可配置可扩展处理器的工作频率，优化T*CORE处理器的汇编代码。本文设计是把天文图像降晰算法移植到嵌入式开发板中，运算速度应该尽可能的快，所以我们应该一方面提高T*CORE处理器的工作频率，找出其关键路径。本文设计的T*CORE的关键路径是FMAC单元的operand寄存器到FMAC单元的中间结果寄存器之间的路径，该路径中有一个26位的乘法器而且该乘法器需要在一个时钟周期之内算出结果，故该路径为整个T*CORE的关键路径，如果想使T*CORE工作在更高的频率，就只能采用流水线的乘法器，这

样会增加乘法指令和乘累加指令的延迟，需要在编程时加以注意。而且用 Virtex-5 开发板的综合工具，由于 FU 功能单元较多，布线延时占整个延时的 69.5%，因此我们应该尽可能的去掉一些不必要的 socket 连接，减少布线延时。另一方面应该对目前的汇编代码做些优化，看能不能去掉一些不必要的 nop 空操作，并且把循环部分展开，减少循环次数，提高运算速度。

2. 完善整个运算流程的自动流水操作。目前对于 PC 端的数据发送操作，并没有一个操作 PCI-E 的软件程序，来自动完成数据传输操作，只是通过 PCI-E 专门的工具来替代这个过程，手动完成 PC 与开发板之间互连的数据传输。这也是今后必须去解决的一个问题。

3. 整个图像相减流程的硬件实现。本论文目前只完成图像相减算法占总时间 74.3%的执行降晰运算，对于前三部分求局部核函数、求局部核函数系统、求解核函数并没有在硬件上去实现。因此希望下一步也能把这三个部分移植到硬件平台上实现，设计新的硬件加速器，提高整个算法流程的处理速度。

参考文献

[1] Crotts, A.P.S, Tomaney, A.B MEGA, A wide-field survey of microlensing in M31, ASP Conference Proceedings, 1996, 552(8)：243~245

[2] Greg P. Kochanski, J. Anthony Tyson, Philippe Fischer, Flickering Faint Galaxies: Few and Far Between, The Astronomical Journal, 1996, 111(4): 1444

[3] C. Alard and Robert H. Lupton, A method for optimal image subtraction, The Astrophysical Journal, 1998, 503(2): 325~331

[4] C. Alard, Image subtraction using a space-varying kernel, Astronomy & Astrophysics Supplement, 2000, 144(2): 363~370

[5] http://www2.iap.fr/users/alard/package.html

[6] Wei Guo, Jizeng Wei, Yongbin Yao, Zaifeng Shi, Su Wang, Design of a configurable and extensible Tcore processor based on Transport Triggered Architecture, the 2009 World Congress on Computer Science and Information Engineering, Los Angels，2009: 536~540

[7] Henk Corporaal, "MOVE32INT Architecture and Programmer's Reference Manual", April 20, 1994

[8] H. Corporaal, From VLIW to TTA, Chichester, UK: John Wiley & Sons, 1997, 35～40

[9] CORPORAAL. H, ARNORLD M, Using transport triggered architectures for embedded processor design, Integrated Computer-Aided Eng, 1998, 5(1): 19～38

[10] HOOGERBRUGGE. J, Code Generation for Embedded System, PhD thesis, Netherland: Delft University of Technology, 1996

[11] Corporaal. H, TTAs, Missing the ILP Complexity Wall, Journal of Systems Architecture, 1999, 45(12): 949～973

[12] 岳虹，沈立，戴葵，基于 TTA 的嵌入式 ASIP 设计，计算机研究与发展，2006，43（4）：752～758

[13] 赵学秘，王志英，岳虹，传输触发体系结构指导下的 ASIP 自动生成，计算机辅助设计与图形学学报，2006，18（10）：1491～1496

[14] 赵学秘，王志英，岳虹，TTA-EC：一种基于传输触发体系结构的 ECC 整体算法处理器，计算机学报，2007，30（2）：225～233

[15] J. Heikkinen, J. Sertamo, T. Rautiainen, etal, "Design of transport triggered architecture processor for discrete cosine transform", In Proc. 15th Annu. Int. ASIC/SOC Conf., pp. 87~91, September 2002

[16] T. Pitkanen, R. Makinen, J. Heikkinen, etal., "Low power, high-performance TTA processor for 1024-point fast Fourier transform", In Embedded Computer Systems, Architectures, Modeling, and Simulation, vol. 4017, pp. 227~236, 2006

[17] 郭炜，SoC 设计方法与实现，北京：电子工业出版社，2005：10~13

[18] 吴武臣，侯立刚，复杂 SoC 设计（Chris Rowen），中国北京：机械工业出版社，2006，63～65

[19] 罗公亮，核函数方法，冶金自动化，2002，117(3)：1~3

[20] 邹谋炎，反卷积与信号恢复，长沙：国防工业出版社, 2001

[21] S. Mika, J. Weston, Fisher discriminant analysis with kernels, Neural networks for signal processing IX, Piscataway: IEEE, 1999, 221(9): 41~48

[22] V N. Vapnik, The nature of statistical learning theory, NY: Springer-Verlag, 1995

[23] P. Astier, Laboratoire de physique nucléaire et de haunts energies, Paris: Private communication, 2004

[24] H. Israel, F V. Hussmann, S. Shush, Optimizing optimal image subtraction, Astronomische Nachrichten, 2007(328): P16

[25] 郑纬民，汤志忠，计算机系统结构（第 2 版），北京：清华大学出版社，2004：38~53

[26] Jizeng Wei, Wei Guo, Jizhou Sun. Design and Implementation of Co-design Toolset for Tcore processor, the 2008 IEEE Asia-Pacific Conference on Circuits and Systems, Macau，2008: 1664~1667

[27] Jan Hoogerbrugge and Henk Corporaal, "Register file port requirements of transport triggered architectures", In MICRO-27, pp. 191~195, 1994

[28] XUPV5-LX110T PCIe x1 Endpoint Plus Design Creation.pdf

[29] Virtex-6 FPGA User Guide, Xilinx, 2010

[30] Gene M.Amdahl, Validity of the Single-Processor Approach to Achieving Large-Scale Computing Capabilities, Federation of Information Processing Societies Conf, 1967, pp. 483~485

[31] Hill and Marty, Amdahl's Law in the Multicore Era, IEEE COMPUTER, 2008

[32] Sun XH and Chen Y, Reevaluating Amdahl's law in the multicore era, J. Parallel and Distributed Computing, 2010(2)

[33] J. Gustafson, Reevaluating Amdahl's Law, Comm. ACM, May 1988

发表论文和科研情况说明

参与的科研项目：

国家自然科学基金项目“AST3实时数据处理关键技术与系统”，项目编号：10978016，参加人，负责天文图像空间变换核降晰算法的SoC设计实现

致　谢

本论文的工作是在我的导师郭炜教授的悉心指导下完成的，郭炜教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在此衷心感谢两年来郭炜老师对我的关心和指导。

郭炜教授悉心指导我们完成了实验室的科研工作，在学习上和生活上都给予了我很大的关心和帮助，在此向郭炜老师表示衷心的谢意。

魏继增老师对于我的科研工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢。

在实验室工作及撰写论文期间，课题组的师少飞、邸强、常轶松、王粟、谭星亮、王勇等同学对我论文中的研究工作给予了热情帮助，在此向他们表达我的感激之情。

另外也感谢家人，他们的理解和支持使我能够在学校专心完成我的学业。