

本科生毕业论文（设计）

题 目 旋翼飞行器飞行控制芯片的
导航解算电路设计与实现

姓 名 学 号

院 系

专 业

指导教师 职 称

2015 年 5 月 15 日

目 录

摘要	I
Abstract	I
1 绪论	1
1.1 研究背景	1
1.2 研究现状	2
1.3 研究意义	3
1.4 研究内容	4
1.5 论文章节	4
2 导航解算电路算法原理分析与仿真验证	5
2.1 导航解算算法基础	5
2.1.1 地球参考模型	5
2.1.2 常用坐标系	6
2.1.3 姿态角	8
2.1.4 姿态矩阵	8
2.2 导航解算算法中的互补滤波原理与应用	10
2.2.1 陀螺仪输出	10
2.2.2 加速度计输出	10
2.2.3 输出噪声	11
2.2.4 互补滤波补偿修正	11
2.3 导航解算算法中的四元数原理与应用	13
2.3.1 四元数的更新	13
2.3.2 四元数的正交化	14
2.3.3 四元数与姿态矩阵	15
2.4 旋翼飞行器导航解算的力学编排及验证分析	15
2.4.1 旋翼飞行器运动姿态的建模	16
2.4.2 理想状态的动力学模型	17
2.4.4 仿真结果及分析	18
2.4.5 惯导算法的改进	19
2.5 导航解算算法的仿真验证	20
2.5.1 仿真数据的采集	20
2.5.2 无互补滤波的算法仿真	20
2.5.3 导航解算算法仿真	22
2.5.4 仿真验证结论	23
3 导航解算电路的设计基础	24
3.1 数据格式	24
3.1.1 浮点数的表示	24
3.1.2 定点数的表示	25
3.2 反正切运算的算法原理及硬线逻辑设计	25
3.2.1 基于 CORDIC 算法实现反正切运算的算法原理	25
3.2.2 反正切运算的 IP 核设计	27

3.3 浮点数运算的硬线逻辑设计	28
3.3.1 浮点数加法/减法器的设计	29
3.3.2 浮点数乘法器的设计	29
3.3.3 浮点数除法器的设计	30
3.3.4 浮点数开方器的设计	30
3.3.5 浮点数比较器的设计	30
3.3.6 浮点数与定点数转换器的设计	30
4 导航解算电路的功能描述与 RTL 实现	31
4.1 加速度计测量误差计算电路的设计	31
4.1.1 加速度计测量误差计算电路的接口设计	31
4.1.2 加速度计测量误差计算电路框图	32
4.2 电子罗盘测量误差计算电路的设计	33
4.2.1 电子罗盘测量误差计算电路的接口设计	33
4.2.2 电子罗盘测量误差计算电路框图	34
4.3 陀螺仪测量误差修正计算电路的设计	35
4.3.1 陀螺仪测量误差修正电路的接口设计	35
4.3.2 陀螺仪测量误差修正电路框图	36
4.4 四元数更新电路的设计	37
4.4.1 四元数更新电路的接口设计	37
4.4.2 四元数更新电路框图	38
4.5 姿态转换矩阵更新电路的设计	39
4.6 姿态角解算电路的设计	41
4.6.1 姿态角解算电路的接口设计	41
4.6.2 姿态角解算电路框图	41
4.6.3 反正切函数电路的设计	42
4.7 顶层电路的设计	44
4.7.1 顶层电路的接口设计	44
4.7.2 顶层电路框图	45
5 导航解算电路的综合与仿真验证	47
5.1 基于 ISE 的综合	47
5.1.1 系统工程的建立	47
5.1.2 综合	48
5.1.3 门数及面积	48
5.2 导航解算电路的仿真验证	48
5.2.1 加速度计测量误差计算电路的仿真与验证	49
5.2.2 电子罗盘测量误差计算电路的仿真与验证	49
5.2.3 陀螺仪测量误差修正电路的仿真与验证	50
5.2.4 四元数更新电路的仿真与验证	51
5.2.5 姿态矩阵更新电路的仿真与验证	51
5.2.6 姿态角解算电路的仿真与验证	52
5.2.7 顶层电路的仿真与验证	53
5.3 仿真结果分析	54
6 结论与展望	55
参考文献	56
致 谢	58

附录 A	IP 核的例化.....	59
附录 B	顶层电路状态机的 RTL 实现.....	61
B.1	顶层电路状态迁移的 RTL 实现.....	61
B.2	顶层电路状态操作的 RTL 实现.....	62
B.3	顶层电路中子模块电路的例化.....	63

旋翼飞行器飞行控制芯片的导航解算电路设计与实现

摘要: 针对目前基于 MCU 和 DSP 芯片的旋翼飞行器系统出现的导航问题, 设计了旋翼飞行器飞行控制芯片中的导航解算电路。在对惯性导航系统的基本原理进行了分析和推导的基础上, 研究了捷联式惯性导航系统中的数学平台工作原理和基于四元数与欧拉角的姿态解算过程, 创新性的提出了基于互补滤波的补偿修正算法对传感器的测量数据进行处理, 提出了一种易于硬线逻辑实现的捷联式惯性导航解算算法, 验证了该算法的准确性和可靠性。并利用 Xilinx ISE 软件进行基于 Xilinx ZedBoard 板载 Zynq-xc7z020-1clg400 芯片的算法硬线逻辑电路设计与 RTL 实现, 最后用 Modelsim SE 软件对电路系统功能模块进行了仿真验证。结果表明, 系统具有高精度、低误差、解算速度快、并行运算能力强等优点, 具有良好的可靠性和可行性。

关键词: 旋翼飞行器 惯性导航 姿态算法 仿真验证 电路设计

Design and Implementation on Navigation Calculating Circuit of Rotor Aircraft Flight Control Chip

Abstract: In view of the present vehicle navigation system which is based on MCU and DSP chip problems, we have designed a navigation decoding circuit in the rotor aircraft flight control chip . On the basic principle of inertial navigation system which has been on the basis of the analysis and derivation, we have research on the inertial navigation system works and mathematical platform which is based on quaternion and Euler angle gesture solver process, we also use a innovative complementary filter algorithm to modify the sensor measurement data and put forward a kind of easy way to hard line logic implementation of strap down inertial navigation decoding algorithm, at late we verify the accuracy and reliability of the algorithm. On this basis,we gave the functional description of the circuit and use Xilinx ISE to design and work out hard line logic of the algorithm based on Zynq-xc7z020-1clg400 chip on ZedBoard of Xilinx Inc, the system has carried on the synthesis and validation by using Modelsim to stimulate the the function of the module. Results show that the system has the higher precision, the lower error, the faster decoding speed, and the advantages of parallel computing capability is strong, so it has a good reliability and feasibility.

Key words: quadrotors; inertial navigation; attitude algorithm; simulation; circuit design

1 绪论

1.1 研究背景

在当下的信息化战争中，低投入、高精度是信息化战争的重要指标。无人机的普及，对现代化信息战争有着许多无可比拟的优势。旋翼飞行器以其较强的稳定性、低廉的成本、较高的效益在战争中得到广泛的应用^[1]。

由于旋翼飞行器的优势，其智能化的普及和低成本的特点收到越来越多爱好者的喜爱。除了基本的用来学习或娱乐的功能外，拓展了许多功能和外部设备。在加入了高级的控制算法后，旋翼飞行器的优势则更加明显。

尽管旋翼飞行器在许多领域得到了运用，但是总体而言依然处于初步发展阶段。旋翼飞行器将机器人技术、电子信息技术、通信技术、信号处理技术、视频图像处理技术、控制算法等多门学科交叉融为一体，具有很好的可开发性和市场需求。

旋翼飞行器整个系统中，最重要也是最复杂的就是控制算法中的导航算法。无论是旋翼飞行器正常平稳的飞行、还是在飞行过程中做出让人叹为观止的暴力特技，都需要使用导航算法实时获取旋翼飞行器的各项飞行参数，反馈给控制系统对旋翼飞行器的旋翼电机进行控制调制^{[2][3]}。

导航，顾名思义就是引导航行的意思，即载体沿着预定航线以要求的精度在指定的时间内引导至目的地^[1]。导航除了起始点和目标点的位置外，还需要随时知道航行体的实时位置、航行速度、航行体的姿态角等参数^{[4][5][6]}。

以航空为例，测量飞行器的位置、速度和姿态等导航参数，通过飞行控制系统引导其按预定的航线航行的整套设备成为导航系统。导航系统只提供各种导航参数而不直接参与对飞行器的控制，因此它是一个开环系统。

惯性导航系统（Inertial Navigation System, INS）是一种主要依靠加速度计、电子罗盘和陀螺仪等惯测仪表，而不依赖于任何外部信息、也不向外部辐射能量的自主式导航系统^{[1][4]}。这就决定了惯导系统具有其他导航系统无法比拟的优异特性。首先由于惯导系统不受外界环境的干扰，其工作环境不仅包括空中、地球表面，还可以在水下，这对军事应用具有重要的意义。其次，惯导系统所提供的导航数据十分完整。还具有数据更新率高、精度小和稳定性好的特点，所以惯导系统在军事以及民用领域发挥着越来越大的作用。

惯性导航系统各式各样，按照不同的分类方法，结果也大相径庭。若从选取以及构成导航坐标系的方法和途径上来分，有捷联式惯性导航系统和平台式惯性导航系统两种类型。

平台式惯导系统是采用物理平台构成导航坐标系的系统，惯性仪表均安装在稳定的物理平台上。在平台式惯性导航系统中，运载体运动参数是以平台坐标系为基准来测量的，因而提取有用信号的计算量小，但是结构复杂、尺寸大，不易于低成本化和硬线逻辑实现。

为了解决平台式惯性导航系统的平台搭建复杂过程，人们提出一种新型的惯性导航方法——捷联式惯性导航系统（SINS, Strapdown Inertial Navigation System）。捷联式惯性导航系统也就是将惯性仪表直接安装到运载体上，采用数学计算平台代替物理平台的导航系统^[5]。

随着电子、控制、航天等理论技术的进一步完善，捷联式惯性导航系统技术进一步完善并有取代平台式惯性导航系统的趋势。

1.2 研究现状

旋翼飞行器是微型飞行器的其中一种，也是一种智能机器人。旋翼飞行器的发展可以追溯至二十世纪初期，法国 Breguet 兄弟制造了第一架四旋翼式直升机，如图 1-1(a)。1924 年，出现了一种叫做 Oemichen 的四旋翼直升机首次实现了 1km 的垂直飞行，如图 1-1(b)。1956 年，Convertawing 造了一架四旋翼直升机，如图 1-1(c)，该飞行器的螺旋桨在直径上超过了 19 英尺，用到了两个发动机，并且通过改变每个螺旋桨提供的推力来控制飞行器。美国陆军研制的 VZ-7，被称为 Flying Jeep，如图 1-1(d)。在此之后的数十年中，旋翼垂直起降机没有什么大的进展。

近十几年来，随着微系统、传感器、控制理论以及旋翼垂直起降机制理论等技术的发展，旋翼垂直起降机又引起人们极大的兴趣。

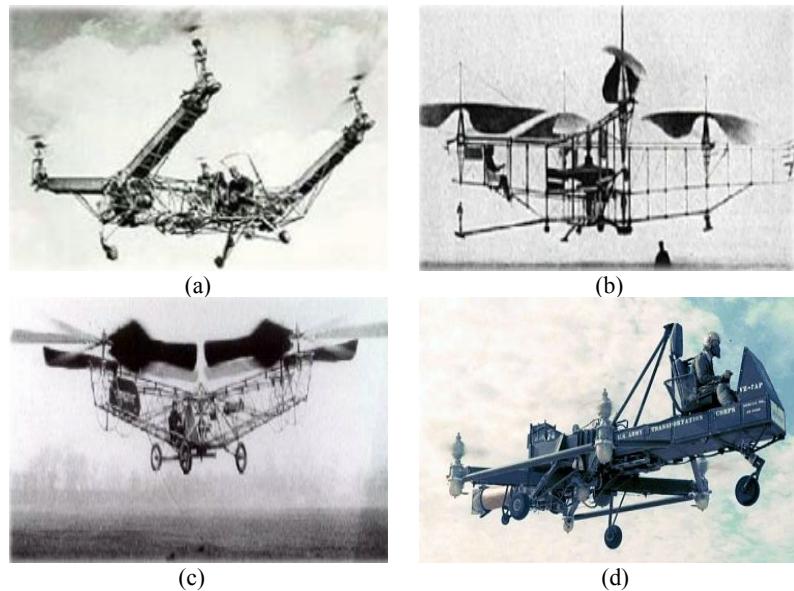


图 1-1 旋翼飞行器的发展

3D Robotics 公司在旋翼飞行器飞行控制系统领域有令人瞩目的研究成果，该公司旗下的 3 款飞控，分别是 ArduPilot (APM)、PX4 和 PixHawk。APM 是比较古老的版本，处理核心使用 Arduino 16 位 Mega 系列单片机。PX4 是 APM 的升级版，使用 STM32F407 单片机，处理速度更快，摆脱了 Arduino 的瓶颈。后来将 PX4FMU 和 PX4IO 结合在一起推出了 PixHawk 飞控，拥有更高的集成度。

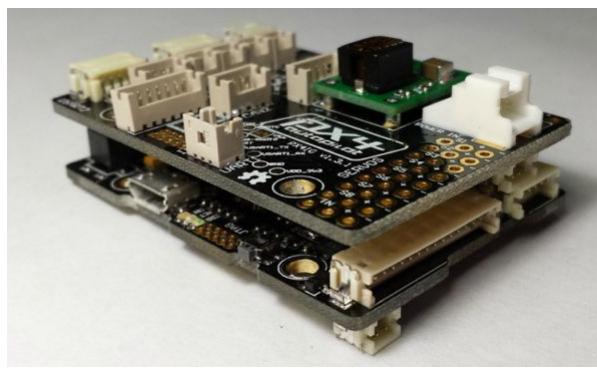


图 1-2 PX4 飞控系统

在 2006 年，Holger Buss 和 Ingo Busker 创造了 MK，一个伟大的 MikroKopter 四轴社区。在 2007 年中，MikroKopter 便像一个“空中的钉子”，像一只鸟一样，稳步的停留在空中。这对于开源四轴飞行器是一个很大的里程碑。

KK 飞控可以说是最经典的多轴飞控，价格便宜，是操纵旋翼机的入门级飞控系统。虽然它支持三轴、四轴、六轴等多种飞行模式，却没有自稳系统。

目前国外的旋翼飞行器使用基本都是基于 MCU 控制器的飞控系统，使用时需要根据自己的旋翼飞行器进行飞控的 C 程序调试和编译^[10]。MCU 成本较高，板级面积较大，不利于在飞行器机身上稳定，串行处理大量信息数据速度较慢。

国内旋翼飞行器虽然发展较晚，但已有若干家公司和高等院校拥有在国际上领先的技术，其中最为著名的就是深圳大疆创新科技有限公司（DJI）。DJI 面向各级消费者推出了不同类型的旋翼航模，DJI 的旋翼飞行器以其稳定的自主悬停技术独树一帜，图 1-3 为 DJI 推出的世界上首款可用于航拍的小型垂直起降一体化多旋翼飞行器—Phantom 1。



图 1-3 DJI Phantom 1

国内高等院校如哈尔滨工业大学、西北工业大学、北京航空航天大学等，也在旋翼飞行器的控制理论方面取得了不错的研究成果。

虽然国内还没有成熟的飞控系统，但是如匿名四轴、圆点博士小四轴等飞控系统并不输于国外的飞控产品。

目前国内外飞控的发展大相径庭，都是利用 MCU 做为处理器进行设计，还未有一款专用飞控芯片投入市场。国内高校在进行飞控系统芯片级设计时，仅仅将 FPGA 做为数据处理器使用，更多的还是通过 MCU 或 DSP 来进行导航解算。

1.3 研究意义

在高新技术突飞猛进发展的引领下，对导航系统精度和速度的需求也不断提高，也使得导航系统趋向于微型化和高度集成化。大多数使用 MCU 或者 DSP 设计的导航解算系统由于高成本、低精度、低运算速度逐渐无法满足市场的需求。可编程逻辑器件在设计和实现复杂数字逻辑系统方面，具有无可比拟的优势和发展潜力，使用 FPGA 进行导航系统的设计，不仅满足以上需求，而且由于 FPGA 设计的高可靠性、较短的设计周期、自顶向下的设计流程等特点，大大提高了系统的整体性能^[2]。

FPGA（现场可编程门阵列，Field Programmable Gate Array），从早期的晶体管级电路、中小型集成电路发展到属于专用集成电路（ASIC）的一种半定制类逻辑电路。FPGA 的并行数据处理方式和流水线体系结构的设计特点，使其较 MCU 和 DSP 拥有更快的处理速度和更高的运行能力。在使用 FPGA 进行设计过程中，可以对其内部数字线路进行资源利用的配置，改变整个 FPGA 的功能来实现设计目标^[3]。

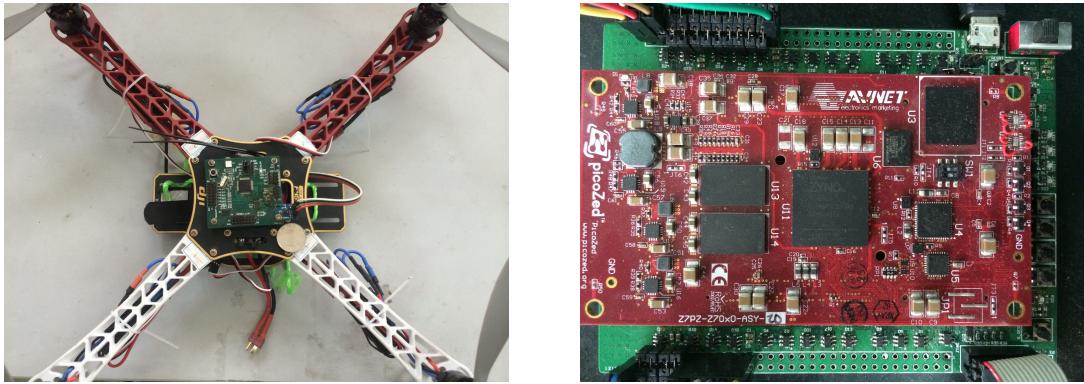


图 1-4 本文中设计使用的旋翼飞行器和 FPGA

1.4 研究内容

本文提出了一种适合硬线逻辑实现的导航算法，通过算法的理论分析和仿真研究，验证进行算法硬线逻辑实现的可行性。给出了电路的功能描述和 RTL 实现，对电路进行了仿真验证。结果显示电路具有良好的可靠性和可行性。主要研究以下两个方面内容：

(1) 惯性导航算法的研究与验证

惯性导航算法包含获取姿态的算法、得到速度算法、获得位置的算法等。其中获取姿态的过程又称为姿态更新，目前已有的算法有：旋转矢量法、欧拉角法、方向余弦法和四元数法^{[9][10]}，各有利弊。四元数法计算量相对来说比较小，算法简单易于实现且精度相对比较高^[11]。在使用四元数法解微分方程时，选用龙格-库塔方程进行求解。

(2) 算法硬线逻辑电路的设计与仿真

在算法研究与优化的基础上，设计了电路的架构，将整个电路系统模块化处理并充分对电路进行优化，给出各个模块电路的架构框图、状态迁移和接口描述。通过硬件描述语言 Verilog HDL 描述了算法的 RTL 实现^[12]。给出了 TestBench 的仿真，验证了所设计电路的可靠性和可行性，为以后成果转化和流片过程奠定基础。

1.5 论文章节

论文各个章节的内容安排如下：

第一章，介绍了本设计的研究背景、国内外发展状况、研究意义和研究内容。

第二章，介绍了惯导系统中的地球参数，简单推导了捷联式惯性导航系统的基本算法原理，创新性的提出了使用互补滤波算法对传感器的采集数据进行修正。给出了设计中惯性导航系统的力学编排模型和分析，系统解算过程的影响因素，对算法进行了改进来消除对系统的影响。最后通过 Matlab 软件对本设计中使用的算法进行了仿真验证。

第三章，给出了电路设计过程中信号的数据格式、基于 CORDIC 算法的反正切函数 IP 核的设计和浮点数运算操作器的设计流程。

第四章，在第二、三章理论研究的基础上，进行了硬线逻辑电路的设计和优化，给出了顶层电路和各个子模块电路的设计方法，介绍了电路的接口、工作流程和状态迁移的条件。

第五章，对第四章所设计的硬线逻辑电路进行了仿真验证。简要介绍了基于 Xilinx ISE 软件的工程建立和综合过程。给出了基于 Modelsim 软件的顶层电路和各个子模块电路的仿真波形，对设计的可靠性和可行性进行了验证。

最后一章对论文研究内容进行总结，提出了本设计的创新点，给出了设计过程中的不足和后续的研究计划。

2 导航解算电路算法原理分析与仿真验证

本章简要推导了捷联式惯性导航系统的基本算法原理，并给出了惯性导航系统的力学编排模型和分析，通过 Matlab 软件对本设计中使用的算法进行了仿真验证，为后续算法的硬线逻辑实现奠定了理论基础。图 2-1 为捷联式惯性导航系统姿态解算的原理框图。

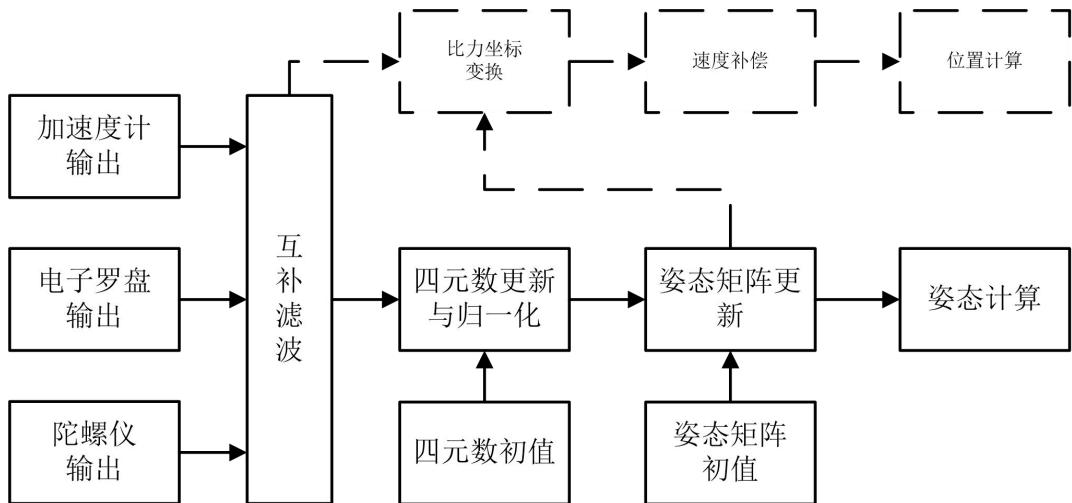


图 2-1 捷联式惯导系统姿态解算算法原理框图

2.1 导航解算算法基础

在近地飞行时，捷联式惯性导航系统中载体相对于地球进行定位，即惯性测量仪表测得的载体相关数据必需通过一定的转换关系转换到地理坐标系才能应用于导航^{[1][13]}。另一方面，导航系统的精度在某种程度上与地球的自身特性及几何参数密切相关。本设计中所研究的导航以地球作为参照体，因此有必要介绍本设计中所涉及的地球几何参数和力学特性。

2.1.1 地球参考模型

在接近地面的捷联惯性导航系统中，载体是相对地球来定位的。也就是惯性仪表测得的载体上的相关数据必须通过一定的转换关系转换到地理坐标系才能应用于导航。而另一方面，导航系统的精度在某种程度上与地球的自身特性及几何参数密切相关。

(1) 地球自转角速度^[14]

根据国际天文协会的数据，地球的自转角速度为

$$\omega_{ie} = 7.2921 \times 10^{-5} \text{ rad/s} \quad (2.1)$$

(2) 地球参考椭球的曲率半径

地球不是一个规则的球体，由于地球自转的影响，地球呈扁圆状，沿赤道的方向突出，南极稍微凹入。在实际问题中，通常用其他几何模型对地球进行近似，主要有以下三种：

① 圆球：将地球视为一个圆球体，球心位于地心，若半径采用 1964 年国际天文学会通过的数据^[14]，则 $R_e = 6378140m$ ，同时满足

$$x^2 + y^2 + z^2 = R^2 \quad (2.2)$$

这种模型又被称为第一近似模型，当对精度要求不高的场合下可以使用该模型。

② 大地水准体：由于地球是一个不规则的球体，若将海平面看做基准平面，把基准面延伸到全部陆地，这样所形成的的表面为大地水准面，把这个大地水准面包围起来

形成的几何体称为地球体或者大地体，用这个来确定地球的形状就是这种模型的基本描述。通常认为地球重力场的一个等位面就是地球体的表面，即认为重力方向和大地体法线方向一致。一般的可认为，海拔高度就是相对大地水准面提出的。

(3) 参考旋转椭球体：在这种模型里，将大地体近似为一个旋转椭球体，中心位于地心，旋转轴为地球的自转轴。长半轴和短半轴分别为 R_e 和 R_p ，绕地球自转轴旋转形成一个椭球体，参考椭球体的赤道平面近似为一个圆平面。这个参考椭球体可用以下二次方程描述

$$\frac{x^2}{R_e^2} + \frac{y^2}{R_e^2} + \frac{z^2}{R_p^2} = 1 \quad (2.3)$$

(4) 扁率

$$e = 1/298.257 \quad (2.4)$$

(5) 子午圈曲率半径

$$R_M = R_e(1 - 2e + 3e \sin^2 L) \quad (2.5)$$

(6) 卯酉圈曲率半径

$$R_N = R_e(1 + e \sin^2 L) \quad (2.6)$$

其中 L 是当地纬度。

(7) 地球重力场特性

与地球形状直接相关的是地球的重力场特性。假设地球是一个均匀规则的球体，不考虑地球自转，则地球表面各点引力相等。但在实际中，地球形状不规则且由于地球自转影响，地球表面物体的单位质量除了受地心引力影响，还受地球自转带来的离心力作用。在地球上，重力加速度的大小和方向与距离地面高度和纬度相关。综合考虑纬度和高度变化的影响^[14]，采用如下公式计算重力加速度

$$g(L, h) = g_0(1 + 0.0052884 \sin^2 L - 0.0000059 \sin^2 2L) - 0.0000003086h \quad (2.7)$$

其中 $g_0 = 9.80616 m/s^2$ ，为赤道上的重力加速度。

(8) 垂线与纬度

地球表面某点的纬度，是该点的垂线方向和赤道平面之间的夹角（线面角），而经度是本地经度平面与本初子午线平面的夹角（面面角）。地球的纬度和垂线是相关的。垂线有测地垂线、重力垂线和地心垂线三种，相对应的纬度有测地纬度、天文纬度、地心纬度三种。不同的纬度定义会造成导航不同程度的偏差。据相关研究表明，测地纬度和天文纬度的差别很小，可以不加以区分，均可作为地理纬度；但是导航系统中地心纬度和地理纬度由于偏差较大，必须加以区分。本文中的纬度均指地理纬度。

2.1.2 常用坐标系

(1) 地心坐标系（i 系，惯性参考坐标系）

在研究惯性系统时，通常将相对恒星所确定的参考系称为惯性空间，空间中静止或做匀速直线运动的参考坐标系称为惯性参考坐标系^{[1][15]}。

惯性参考坐标系可分为太阳中心惯性坐标系和地心惯性坐标系。由于本设计中研究对象主要为在地球表面附近载体的导航问题，因此选用地心惯性坐标系做为整个研究的惯性参考坐标系^[1]。

地心惯性坐标系符合右手直角坐标系规则。坐标系原点 O_i 在地球中心， Z_i 轴沿地球自转轴指向北极点， X_i 轴、 Y_i 轴在赤道平面内。

由于地心坐标系不随地球自转而转动，因此也称地心惯性坐标系为地球固定坐标系。

(2) 地球坐标系（e 系）

地球坐标系符合右手直角坐标系规则。坐标系原点 O_e 在地球中心， Z_e 轴沿极轴指

向北， X_e 轴在赤道平面与本初子午线的交线上， Y_e 轴在赤道平面内。地球坐标系与地球固连，随地球一起转动。

(3) 地理坐标系(n 系，导航坐标系)

地理坐标系符合右手直角坐标系规则，一般选取“东北天”为顺序。坐标系原点 O_n 与载体所在的点重合， Z_n 轴沿当地地理垂线指上， X_n 轴沿当地纬线指东， Y_n 轴沿当地经线指北。

(4) 地平坐标系(t 系)

地平坐标系右手直角坐标系。坐标系原点 O_t 与载体所在的点重合， X_t 轴保持水平指右， Y_t 轴与轨迹相切指向轨迹前进方向， Z_t 轴沿当地垂线方向指上。

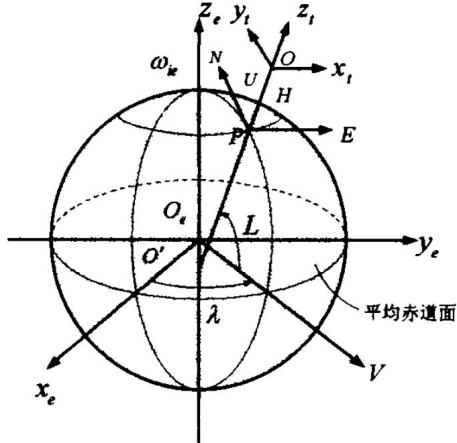


图 2-2 地球坐标系、地理坐标系与地平坐标系示意图

(5) 载体坐标系(b 系)

载体坐标系符合右手直角坐标系规则。坐标系圆点 O_b 与载体质心重合， X_b 轴沿机体横轴指右， Y_b 轴沿机体纵轴指前， Z_b 轴沿机体垂直指上。

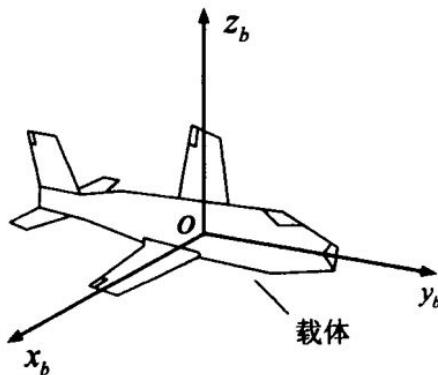


图 2-3 载体坐标系

设东向加速度计测量载体沿东西方向的运动加速度 a_e ，北向加速度计测量载体沿南北方向的运动加速度 a_n ，垂直加速度计测量载体沿天地方向的运动加速度 a_u 。

$$\begin{cases} v_e(t) = v_e(0) + \int_0^t a_e dt \\ v_n(t) = v_n(0) + \int_0^t a_n dt \\ v_u(t) = v_u(0) + \int_0^t a_u dt \end{cases} \quad (2.8)$$

那么可以得到载体三向速度分量为

$$\text{载体的瞬时位置为} \begin{cases} \lambda = \lambda_0 + \int_0^t \frac{v_e}{(R_N + h) \cos L} dt \\ L = L_0 + \int_0^t \frac{v_n}{R_M + h} dt \\ h = h_0 + \int_0^t v_u dt \end{cases} \quad (2.9)$$

2.1.3 姿态角

用 θ 表示俯仰角，指载体纵轴与纵向水平轴之间的夹角，以纵向水平轴为起点，向上为正，取值范围为 $-90^\circ \sim 90^\circ$ 。在地理坐标系中，载体绕俯仰轴（X 轴）转动 θ 即为俯仰角，一般用 Pitch 表示。

用 γ 表示横滚角，指载体纵向对称面与纵向铅垂面之间的夹角，从载体铅垂面开始计算，右倾斜为正方向，取值范围为 $-180^\circ \sim 180^\circ$ 。在地理坐标系中，载体绕横滚轴（Y 轴）转动 γ 即为横滚角，一般用 Roll 表示。

用 φ 表示航向角，指子午线与载体水平坐标系纵轴之间的夹角。通常以地理坐标系北向为起点，偏东方向为正，取值范围为 $0^\circ \sim 360^\circ$ 。在地理坐标系中，载体绕航向轴（Z 轴）转动 φ 即为航向角，一般用 Yaw 表示。

2.1.4 姿态矩阵

在惯性系统中，姿态矩阵也就是地理坐标系 n 与载体坐标系 b 之间的转换关系^{[1][16]}。地理坐标系 $ox_ny_nz_n$ 坐标绕 z_n 轴方向旋转，航向角 φ 得到 $ox'y'z'$ ， $ox'y'z'$ 绕 y' 轴旋转俯仰角 θ 得到 $ox''y''z''$ ， $ox''y''z''$ 再绕 x'' 轴旋转横滚角 γ 则得到载体坐标系 $ox_by_bz_b$ ，根据整个运算旋转过程的值，可以得到载体坐标系与地理坐标系之间的旋转角度的运算关系式，用方向余弦阵表示为

$$\begin{aligned} C_n^b &= C_\gamma C_\theta C_\varphi = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma \\ 0 & -\sin \gamma & \cos \gamma \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos \theta \cos \varphi & \cos \theta \sin \varphi & -\sin \theta \\ \sin \gamma \cos \theta \cos \varphi - \cos \gamma \sin \varphi & \sin \gamma \sin \theta \sin \varphi + \cos \gamma \cos \varphi & \sin \gamma \cos \theta \\ \cos \gamma \sin \theta \cos \varphi + \sin \gamma \sin \varphi & \cos \gamma \sin \theta \sin \varphi - \sin \gamma \cos \varphi & \cos \gamma \cos \theta \end{bmatrix} \end{aligned} \quad (2.10)$$

易知

$$\begin{aligned} C_b^n &= \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \\ &= \begin{bmatrix} \cos \theta \cos \varphi & \sin \gamma \cos \theta \cos \varphi - \cos \gamma \sin \varphi & \cos \gamma \sin \theta \cos \varphi + \sin \gamma \sin \varphi \\ \cos \theta \sin \varphi & \sin \gamma \sin \theta \sin \varphi + \cos \gamma \cos \varphi & \cos \gamma \sin \theta \sin \varphi - \sin \gamma \cos \varphi \\ -\sin \theta & \sin \gamma \cos \theta & \cos \gamma \cos \theta \end{bmatrix} \end{aligned} \quad (2.11)$$

姿态矩阵 C_n^b 是三个姿态角（俯仰角 θ 、横滚角 γ 和航向角 φ ）的函数，即可由 C_n^b 的元素确定姿态角主值的四元数表达式

$$\begin{cases} \theta_0 = \arcsin(-C_{13}) \\ \gamma_0 = \arctan\left(\frac{C_{23}}{C_{33}}\right) \\ \varphi_0 = \arctan\left(\frac{C_{12}}{C_{11}}\right) \end{cases} \quad (2.12)$$

为了方便后续电路的实现，用反正切函数表达俯仰角

$$\theta_0 = \arcsin(-C_{13}) = \arctan\left(\frac{-C_{13}}{\sqrt{1-C_{13}^2}}\right) \quad (2.13)$$

反正切函数 $\arctan(x)$ 的输出范围为 $(-\pi, -\frac{\pi}{2}) \cup (-\frac{\pi}{2}, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)$ ，得到姿态角的主值后，还需根据姿态角的定义域确定姿态角的真值。俯仰角 θ 的定义域为 $(-\frac{\pi}{2}, \frac{\pi}{2})$ ，因此俯仰角的真值与其主值是一致的^[17]。横滚角 γ 的定义域为 $(-\pi, \pi)$ ，由于当 $C_{33} = 0$ 时， $\frac{C_{23}}{C_{33}}$ 不合法，因此根据 C_{23} 的符号可判断 γ 在 $C_{33} = 0$ 时的取值。航向角 φ 的定义域为 $(0, 2\pi)$ ，当 $\frac{C_{12}}{C_{11}} < 0$ ，即 $\arctan(\frac{C_{12}}{C_{11}}) < 0$ 时，需要对真值补偿 2π ^{[17][18]}。

综上，姿态角的更新表达式为

$$\begin{cases} \theta = \theta_0 \\ \gamma = \begin{cases} \gamma_0 & , C_{33} \neq 0 \\ \frac{\pi}{2} & , C_{33} = 0 \cup C_{23} > 0 \\ -\frac{\pi}{2} & , C_{33} = 0 \cup C_{23} < 0 \end{cases} \\ \varphi = \begin{cases} \varphi_0 & , \frac{C_{12}}{C_{11}} \geq 0 \\ \varphi_0 + 2\pi & , \frac{C_{12}}{C_{11}} \leq 0 \\ \frac{\pi}{2} & , C_{11} = 0 \cup C_{12} > 0 \\ \frac{3\pi}{2} & , C_{11} = 0 \cup C_{12} < 0 \end{cases} \end{cases} \quad (2.14)$$

2.2 导航解算算法中的互补滤波原理与应用

2.2.1 陀螺仪输出

机体坐标系(b 系)相对于导航坐标系(n 系)的转动角速度在机体坐标系(b 系)中的投影

$$\boldsymbol{\omega}_{nb}^b = \mathbf{C}_1^b \begin{bmatrix} \dot{\theta} \\ 0 \\ -\dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\gamma} \\ 0 \end{bmatrix} \quad (2.15)$$

导航坐标系相(n 系)对于惯性参考系(i 系)的转动角速度在导航坐标系(n 系)中的投影

$$\boldsymbol{\omega}_{in}^n = \boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n \quad (2.16)$$

其中 $\boldsymbol{\omega}_{ie}^n = \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{ie}^n \cos L \\ \boldsymbol{\omega}_{ie}^n \sin L \end{bmatrix}$ 为地球自转角速度相对于参考系的转动角速度在导航系中的投

影， $\boldsymbol{\omega}_{en}^n = \begin{bmatrix} -V_y^n \\ \frac{V_x^n}{R_M + h} \\ \frac{V_x^n \tan L}{R_N + h} \end{bmatrix}$ 为导航系相对于参考系的转动角速度在导航系中的投影^[19]。

机体坐标系(b 系)相对于惯性坐标系(e 系)的转动角速度在机体坐标系(b 系)中的投影即陀螺仪理想输出参量

$$\boldsymbol{\omega}_{ib}^b = \boldsymbol{\omega}_{in}^b + \boldsymbol{\omega}_{nb}^b = \mathbf{C}_n^b \boldsymbol{\omega}_{in}^n + \boldsymbol{\omega}_{nb}^b \quad (2.17)$$

由于陀螺仪本身存在误差，考虑陀螺仪元件的误差 $\boldsymbol{\epsilon}^b$ ^[20]，因此陀螺仪的输出为

$$\tilde{\boldsymbol{\omega}}_{ib}^b = \boldsymbol{\omega}_{ib}^b + \boldsymbol{\epsilon}^b \quad (2.18)$$

一般的，在不考虑机体及外部环境的震动等噪声影响下，陀螺仪误差由常值漂移和随机误差影响^{[20][21]}

$$\boldsymbol{\epsilon}^b = \boldsymbol{\epsilon}_b + \boldsymbol{\epsilon}_r + w_g \quad (2.19)$$

其中 $\boldsymbol{\epsilon}_b$ 为常值漂移， $\boldsymbol{\epsilon}_r$ 为一阶马尔科夫过程， w_g 为白噪声。

2.2.2 加速度计输出

加速度计测量的量是比力。在导航坐标系中。比力与机体相对地球加速度之间的关系可以表示为

$$f^n = \mathbf{C}_n^t a^t + (2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n) \times \mathbf{V}^n + \mathbf{g}^n \quad (2.20)$$

式中 a^t 为机体相对于地球的加速度在轨迹坐标系中的投影， \mathbf{C}_n^t 为轨迹坐标系与导航坐标系的转换矩阵， $2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n$ 为机体相对于地球速度与地球自转角速度的相互影响而形成的哥氏加速度， \mathbf{g}^n 为地球的重力加速度在导航系的投影。

加速度计理想输出为

$$f^b = \mathbf{C}_n^b f^n \quad (2.21)$$

由于加速度计本身存在误差，考虑加速度计元件的误差 ∇_a^b ^[20]，因此加速度计的输出为

$$\tilde{f}^b = f^b + \nabla_a^b \quad (2.22)$$

一般的，在不考虑机体及外部环境的震动等噪声影响下，加速度计误差由零偏漂移和随机误差影响^{[20][21]}

$$\nabla_a^b = \nabla_a + \nabla_r + w_a \quad (2.23)$$

其中 ∇_a 为零偏漂移， ∇_r 为一阶马尔科夫过程， w_a 为白噪声。

2.2.3 输出噪声

在解算过程中，需要考虑加速度计和陀螺仪的元件误差，因此需要对理想输出进行加噪处理^[21]。

已知马尔科夫过程相关时间 $M_t = 1/3600$ ，设陀螺仪常数漂移 $GCND = 1^\circ/h$ ，白噪声漂移 $GWND = 0.02^\circ/h$ ，加速度计零偏漂移 $ACND = 10\mu g$ ，白噪声漂移 $AWN = 50\mu g$ ， k 时刻白噪声为 $WN(k)$ ，马尔科夫过程为 $M(k)$ ，可得

$$\begin{cases} \varepsilon^b(k+1) = GCND + GWND \cdot WN(k+1) + e^{-Mt \cdot M(k)} + \frac{WN(k)}{2 \cdot Mt} \cdot (1 - e^{-2 \int M_t dt}) \cdot GWND \cdot WN(k) \\ \nabla_a^b(k+1) = ACND + AWND \cdot WN(k+1) + e^{-Mt \cdot M(k)} + \frac{WN(k)}{2 \cdot Mt} \cdot (1 - e^{-2 \int M_t dt}) \cdot AWND \cdot WN(k) \end{cases} \quad (2.24)$$

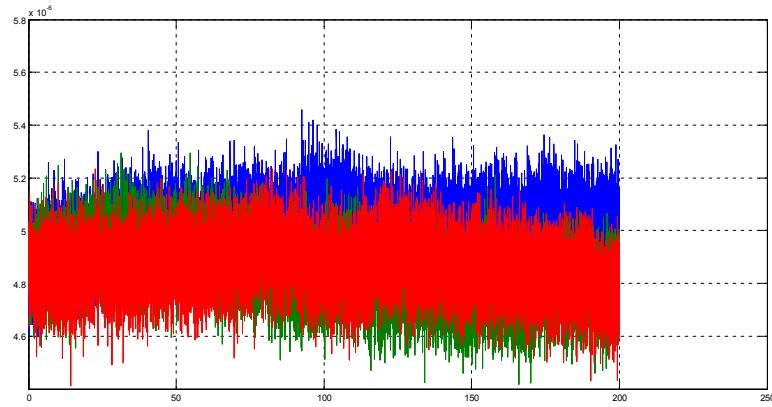


图 2-4 陀螺仪噪声

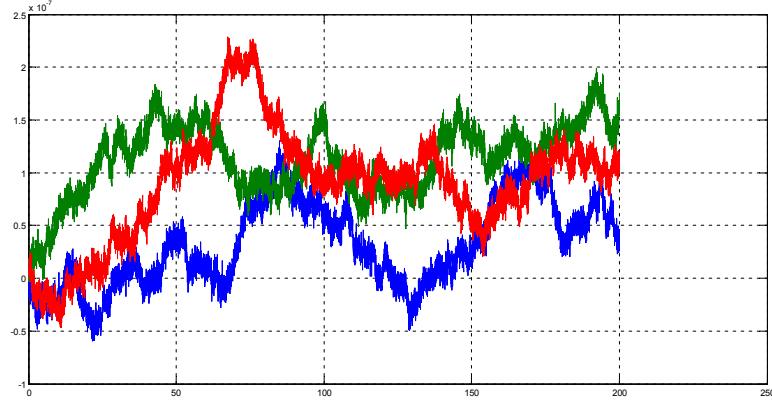


图 2-5 加速度计噪声

即

$$\begin{cases} \tilde{\omega}_{ib}^b = \omega_{ib}^b + \varepsilon^b \\ \tilde{f}_b = f_b + \nabla_a^b \end{cases} \quad (2.25)$$

因此，在进行数据使用前，需要对陀螺仪数据进行误差修正。

2.2.4 互补滤波补偿修正

互补滤波算法是一个非常关键的算法，也是本文的一个重要创新点。该算法主要作用是通过对关键的传感器的测量值进行去噪滤波处理来提升输出数据的精度^[21]。

在实际环境中，惯性仪表如陀螺仪和加速度计输出的数据都存在误差。而在后续算法过程中，这两个器件的输出数据是否准确关系十分重大。所以对两个器件的输出必须

经过一定的处理以提高其精度^{[22][23][24]}。

由于陀螺仪漂移的存在，会导致其测得的数据越来越不准。互补滤波就是在短时间内较多参考陀螺仪的数据，而随着时间的增加，对陀螺仪测得的数据的信任度降低的时候，采用加速度计测得的数据对其进行补偿校正。从原理上说来，是由于陀螺仪和加速度计其各自的特性决定的^[25]。如加速度计由于其动态响应慢，在高频段存在误差，可以通过低通滤波器来抑制其高频误差；而陀螺仪由于常值零漂低频段存在较大误差，所以需滤波器来抑制其低频误差。而互补滤波器就是根据其在频域上的互补特性来设计，以抑制干扰误差。这样就能够将两个惯性器件的优点结合起来，得到更加精确的数据。互补滤波的设计也是本设计的一个创新所在。

设加速度计在某时刻的三轴测量值为 $\mathbf{a} = [a_x \ a_y \ a_z]^T$ ，又由于重力加速度在导航坐标系中的比力分量为 $\mathbf{g}_n = [0 \ 0 \ 1]^T$ ，可求得重力加速度在载体坐标系的比例分量为

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \mathbf{C}_n^b \cdot \mathbf{g}_n = \mathbf{C}_n^b \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2(q_1q_3 - q_0q_2) \\ 2(q_2q_3 + q_0q_1) \\ 1 - 2(q_1^2 + q_2^2) \end{bmatrix} \quad (2.26)$$

设电子罗盘在同一时刻的三轴测量值为 $\mathbf{m} = [m_x \ m_y \ m_z]^T$ ，由于电子罗盘测量值为载体坐标系相对参考坐标系的磁力线变化，需要将测量值从载体坐标系转换到导航坐标系中的磁场矢量^{[26][27]}

$$\mathbf{h} = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = \mathbf{C}_n^b \cdot \mathbf{m} = \begin{bmatrix} m_x[1 - 2(q_2^2 + q_3^2)] + 2m_y(q_1q_2 - q_0q_3) + 2m_z(q_1q_3 + q_0q_2) \\ 2m_x(q_1q_2 + q_0q_3) + m_y[1 - 2(q_1^2 + q_3^2)] + 2m_z(q_2q_3 - q_0q_1) \\ 2m_x(q_1q_3 - q_0q_2) + 2m_y(q_2q_3 + q_0q_1) + m_z[1 - 2(q_1^2 + q_2^2)] \end{bmatrix} \quad (2.27)$$

设导航坐标系中的磁场矢量为 $\mathbf{b} = [b_x \ b_y \ b_z]^T$ 。在导航坐标系中，垂直面上也存在有磁场分量 b_z ，但是不同位置的垂直分量是不同值，所以一般的可将 b_z 近似为 h_z 。由于地理地磁水平夹角为 0° （忽略磁偏角等因素，固定方向指北），所以 $b_y = 0$ 。而磁场水平分量的大小是一致的，即为 $b_x^2 + b_y^2 = h_x^2 + h_y^2$ ，而 $b_y = 0$ ，可求得 $b_x = \sqrt{h_x^2 + h_y^2}$ 。

综上可得导航坐标系中的磁场矢量

$$\mathbf{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} \sqrt{h_x^2 + h_y^2} \\ 0 \\ h_z \end{bmatrix} \quad (2.28)$$

最后将导航坐标系中的磁场矢量投影到载体坐标系中，可得

$$\mathbf{w} = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \mathbf{C}_n^b \cdot \mathbf{b} = \begin{bmatrix} b_x[1 - 2(q_2^2 + q_3^2)] + 2b_z(q_1q_3 + q_0q_2) \\ 2b_x(q_1q_2 + q_0q_3) + 2b_z(q_2q_3 - q_0q_1) \\ 2b_x(q_1q_3 - q_0q_2) + b_z[1 - 2(q_1^2 + q_2^2)] \end{bmatrix} \quad (2.29)$$

综合以上各式，将加速度计的测量矢量和参考矢量做叉乘可得到加速度计的测量误差，将电子罗盘的测量矢量与参考矢量叉乘得到电子罗盘测量误差。两者相加，可以得到的用来修正陀螺仪测量值的误差矢量

$$\mathbf{e} = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = \mathbf{a} \times \mathbf{v} + \mathbf{m} \times \mathbf{w} = \begin{bmatrix} (a_y v_z - a_z v_y) + (m_y w_z - m_z w_y) \\ (a_z v_x - a_x v_z) + (m_z w_x - m_x w_z) \\ (a_x v_y - a_y v_x) + (m_x w_y - m_y w_x) \end{bmatrix} \quad (2.30)$$

设陀螺仪的采样起始时刻为 t_k 、更新周期为 T ，那么测量修正值可用下式表示

$$\begin{aligned}
\mathbf{g}(t_k + T) &= \begin{bmatrix} g_x(t_k + T) \\ g_y(t_k + T) \\ g_z(t_k + T) \end{bmatrix} \\
&= \mathbf{g}(t_k) + K_p \cdot \mathbf{e}(t_k + T) + K_i \sum_{t=1}^{t_k+T} \int \mathbf{e}(t) dT = \begin{bmatrix} g_x(t_k) + K_p \cdot e_x(t_k + T) + K_i \sum_{t=1}^{t_k+T} \int e_x(t) dT \\ g_y(t_k) + K_p \cdot e_y(t_k + T) + K_i \sum_{t=1}^{t_k+T} \int e_y(t) dT \\ g_z(t_k) + K_p \cdot e_z(t_k + T) + K_i \sum_{t=1}^{t_k+T} \int e_z(t) dT \end{bmatrix} \quad (2.31)
\end{aligned}$$

其中, K_p 为误差比例系数常量, K_i 为误差积分系数常量。一般的, K_p 为 K_i 的 10—100 倍^{[6][28]}。

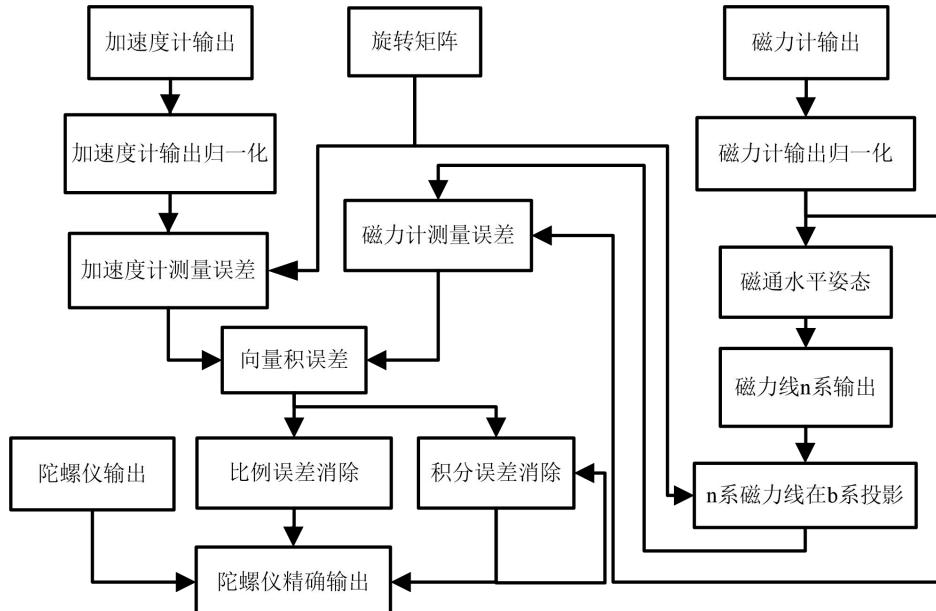


图 2-5 互补滤波算法原理框图

2.3 导航解算算法中的四元数原理与应用

2.3.1 四元数的更新

设有四元数 $\mathbf{Q} = q_0 + q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k}$, 则有四元数更新微分方程^{[31][32]}

$$\dot{\mathbf{Q}} = \frac{1}{2} \mathbf{Q} \bullet \boldsymbol{\omega}_{nb}^b \quad (2.35)$$

式中 $\mathbf{Q} = [q_0 \ q_1 \ q_2 \ q_3]^T$, 表示姿态四元数,

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\omega_{nbx}^b & -\omega_{nby}^b & -\omega_{nbz}^b \\ \omega_{nbx}^b & 0 & \omega_{nbz}^b & -\omega_{nby}^b \\ \omega_{nby}^b & -\omega_{nbz}^b & 0 & \omega_{nbx}^b \\ \omega_{nbz}^b & \omega_{nby}^b & -\omega_{nbx}^b & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (2.36)$$

式中, $\boldsymbol{\omega}_{nb}^b = \boldsymbol{\omega}_{ib}^b - \boldsymbol{\omega}_{in}^b$ 。 $\boldsymbol{\omega}_{ib}^b$ 即为陀螺仪的输出量 $\mathbf{g} = [g_x \ g_y \ g_z]^T$; $\boldsymbol{\omega}_{in}^b$ 为导航坐标系相对于惯性参考系的转动角速度在导航坐标系中的投影, 且有

$$\boldsymbol{\omega}_{in}^b = \boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n \quad (2.37)$$

式中 $\boldsymbol{\omega}_{ie}^n = [0 \ \boldsymbol{\omega}_{ie}^n \cos L \ \boldsymbol{\omega}_{ie}^n \sin L]^T$ 为地球自转角速度相对于参考系的转动角速度在导航

系中的投影， $\boldsymbol{\omega}_{\text{en}}^n = \begin{bmatrix} -V_y^n & V_x^n & V_x^n \tan L \\ R_M + h & R_N + h & R_N + h \end{bmatrix}^\top$ 为导航系相对于参考系的转动角速度在导航系中的投影， R_M 、 R_N 、 L 和 h 分别为地球参考椭球的长轴半径、短轴半径、载体所在地理位置的纬度和载体所在地理位置的高度。

由于旋翼机的在供电设备最大允许条件下，最大航行高度和姿态变化时的角速度变化与地球的半径和自转角速度相比可以忽略，即 $\boldsymbol{\omega}_{\text{in}}^n = 0$ 。因此可近似为

$$\boldsymbol{\omega}_{\text{nb}}^b \approx \boldsymbol{\omega}_{\text{ib}}^b = [g_x \quad g_y \quad g_z]^\top \quad (2.38)$$

综合以上，则有四元数微分方程^[32]

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -g_x & -g_y & -g_z \\ g_x & 0 & g_z & -g_y \\ g_y & -g_z & 0 & g_x \\ g_z & g_y & -g_x & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (2.39)$$

解算四元数，采用一阶龙格库塔法，设采样起始时刻为 t_k 、更新周期为 T ，在一个更新周期内对陀螺仪的输出角速率进行一次采样，那么起始角速度和采样角速度分别为

$$\mathbf{g}(t_k), \quad \mathbf{g}(t_k + T)$$

一阶龙格-库塔迭代公式^[32]

$$\boldsymbol{Q}(t_k + T) = \boldsymbol{Q}(t_k) + T \mathbf{K} \quad (2.40)$$

其中斜率矩阵 \mathbf{K} 计算如下

$$\mathbf{K} = \frac{1}{2} [\boldsymbol{Q}(t_k) \cdot \boldsymbol{\omega}_{\text{nb}}^b(t_k)] \quad (2.41)$$

综上得到 $t_k + T$ 时刻四元数更新的表达式

$$\begin{aligned} \boldsymbol{Q}(t_k + T) &= \begin{bmatrix} q_0(t_k + T) \\ q_1(t_k + T) \\ q_2(t_k + T) \\ q_3(t_k + T) \end{bmatrix} = \boldsymbol{Q}(t_k) + \frac{1}{2} \int \dot{\boldsymbol{Q}}(t_k + T) dt \\ &= \begin{bmatrix} q_0(t_k) + 0.5(-q_1(t_k) \cdot g_x - q_2(t_k) \cdot g_y - q_3(t_k) \cdot g_z) \cdot T \\ q_1(t_k) + 0.5(q_0(t_k) \cdot g_x + q_2(t_k) \cdot g_z - q_3(t_k) \cdot g_y) \cdot T \\ q_2(t_k) + 0.5(q_0(t_k) \cdot g_y - q_1(t_k) \cdot g_z - q_3(t_k) \cdot g_x) \cdot T \\ q_3(t_k) + 0.5(q_0(t_k) \cdot g_z + q_1(t_k) \cdot g_y - q_2(t_k) \cdot g_x) \cdot T \end{bmatrix} \end{aligned} \quad (2.42)$$

2.3.2 四元数的正交化

四元数在更新的同时需要进行规范化处理，以保证方向余弦矩阵是一个正交矩阵，即保证数学平台的三个坐标轴符合右手直角坐标系规则。

设有归一化前四元数

$$\boldsymbol{Q} = q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k} + q_4 \mathbf{k} \quad (2.43)$$

\boldsymbol{Q} 的模为

$$\sqrt{N_{\boldsymbol{Q}}} = |\boldsymbol{Q}| = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} \quad (2.44)$$

将四元数除以它的模可得

$$\hat{\boldsymbol{Q}} = \frac{\boldsymbol{Q}}{\sqrt{N_{\boldsymbol{Q}}}} = \frac{q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k} + q_4 \mathbf{k}}{\sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}} \quad (2.45)$$

式中

$$\hat{q}_i = \frac{q_i}{\sqrt{N_q}} \quad (2.46)$$

四元数范数可表示为

$$\|\mathbf{Q}\| = q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \quad (2.47)$$

显然实现四元数的归一化，从而也就是完成了姿态矩阵 \mathbf{C}_b^n 的正交化。

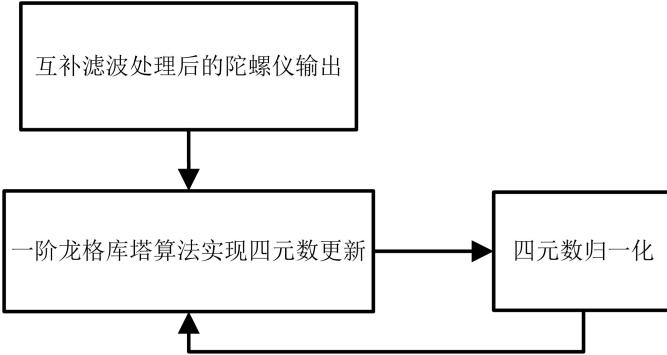


图 2-6 四元数更新算法原理框图

2.3.3 四元数与姿态矩阵

四元数理论是数学的一个分支，本文中将从四元数理论在捷联式惯性导航系统中的应用出发，不加证明地给出四元数中的元素与姿态矩阵中的元素之间的关系^{[30][31]}。

四元数是一个实数单位为 1 和三个虚数单位 $i \ j \ k$ 组成的含有四个元的数，可用下式表达

$$\mathbf{Q} = q_0 + q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k} \quad (2.32)$$

一个坐标系相对于另外一个坐标系的转动，可以用四元数唯一的表示出来。那么载体坐标系相对于导航坐标系运动时可如下表示

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 + q_1q_2) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} \quad (2.33)$$

即有

$$\mathbf{C}_n^b = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 + q_1q_2) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \quad (2.34)$$

因此姿态矩阵中的九个元素都可使用四元数一一对应的表示出来。

2.4 旋翼飞行器导航解算的力学编排及验证分析

旋翼飞行器具有呈十字交叉的多个螺旋桨，它通过改变四个螺旋桨的升力来实现不同的运动，主要方法是改变螺旋桨的转速。旋翼有且仅有 4 个输入力，却需要产生 6 个自由度方向的运动，属于典型的欠驱动系统^[22]。而且四旋翼飞行器具有高度的耦合动特性，一个螺旋桨速度发生变化，将会同时改变 6 个状态量。四旋翼的姿态控制特性是发散的，升力的偏斜会使得姿态偏离中立位置越来越远，直到升力不足以支撑重力导致飞行器坠落。

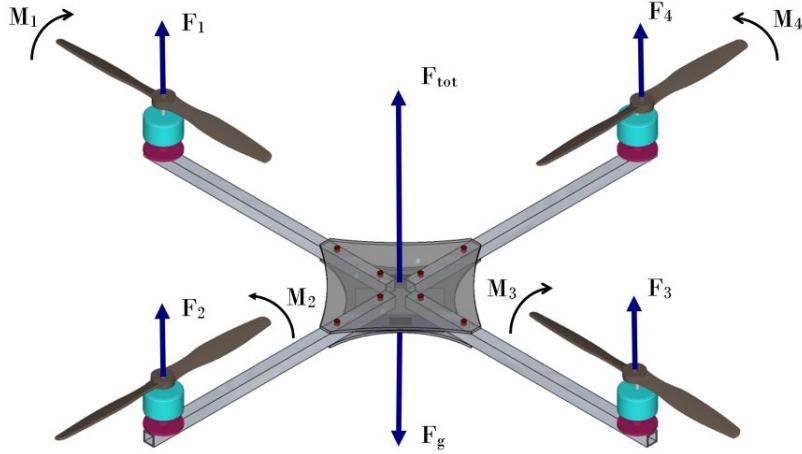


图 2-7 旋翼飞行器力学模型

2.4.1 旋翼飞行器运动姿态的建模

根据欧拉旋转定理，任何共原点的两个坐标系之间的关系，是一个绕着包含原点的固定轴的旋转^[33]。从 b 系到 n 系的转动，只需要沿 n 轴旋转 Ω 弧度（根据右手定则确定正方向）。姿态四元数可描述该旋转

$$\mathbf{n} = [n_x \quad n_y \quad n_z]^T \quad (2.48)$$

$$\mathbf{Q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \cos \frac{\Omega}{2} \\ n_x \cdot \sin \frac{\Omega}{2} \\ n_y \cdot \sin \frac{\Omega}{2} \\ n_z \cdot \sin \frac{\Omega}{2} \end{bmatrix} \quad (2.49)$$

由式 (2.11) 可得

$$f^n = \mathbf{C}_b^n f^b \quad (2.50)$$

设 r 为导航坐标系原点（飞行场地坐标零点）到飞行器重心的位移矢量； $F_1 \ F_2 \ F_3 \ F_4$ 为 4 个旋翼的升力；忽略机体空气阻力、机体陀螺效应、螺旋桨的陀螺效应（文献和实际试飞证明这些效应的影响很小），则可以得到飞行器质心的平动规律

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + C_b^n \begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} \quad (2.51)$$

设 L 为机架臂长（旋翼旋转中心到机体中心的距离）， I 为机体的转动惯量矩阵， $M_1 \ M_2 \ M_3 \ M_4$ 为 4 个旋翼的反扭力。我们可以得到飞行器质心的转动规律

$$I \begin{bmatrix} \dot{\omega}_{nbx}^b \\ \dot{\omega}_{nby}^b \\ \dot{\omega}_{nbz}^b \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} \omega_{nbx}^b \\ \omega_{nby}^b \\ \omega_{nbz}^b \end{bmatrix} \times I \begin{bmatrix} \omega_{nbx}^b \\ \omega_{nby}^b \\ \omega_{nbz}^b \end{bmatrix} \quad (2.52)$$

旋翼机依靠桨盘与空气的相对运动产生升力，同时桨盘的旋转受到空气阻力。在亚音速条件下，机翼的升力和阻力与气流的动压头成正比，且可以认为升、阻力系数与速度无关^[22]。桨的升、阻力系数和参考面积记作 $C_l \ C_d \ S$ ，可得

$$P = \frac{1}{2} \rho v^2, \quad F_r = Cl \cdot P \cdot S, \text{ 其中 } \rho \text{ 为空气密度} \quad (2.53)$$

设一个桨和电机的转动惯量 I_r , 桨的阻力系转速为 ω_r , 它产生的反扭力为

$$M_r = Cd \cdot P \cdot S \quad (2.54)$$

第 i 个旋翼的转速 ω_i 与其升力 F_i 和反扭力 M_i 有如下关系

$$F_i = k_F \cdot \omega_i^2, \quad M_i = k_M \cdot \omega_i^2 \quad (2.55)$$

一般的, $k_F \approx 5 \times 10^{-8} N/rpm^2$, $k_M \approx 1.5 \times 10^{-9} N/rpm^2$

由 BLDC 的原理可知, 桨、电机、调速器组成的系统有如下的动态特性:

$$\dot{\omega}_i = \frac{1}{\tau} (\omega_i^{set} - \omega_i) \quad (2.56)$$

因此旋翼是一个一阶系统, 角速度响应时间常数为 τ 。然而实际的电子调速器会对输入进行线性化, 使得输入油门信号和桨升力成正比

$$F_i = Throttle \times Gain \quad (2.57)$$

一般的, 角速度响应时间常数取值 $\tau \approx 0.05$ 。

2.4.2 理想状态的动力学模型

对于姿态解算算法在旋翼飞行器中的应用, 考虑到旋翼飞行器的静态和动态的力学编排是不同的^[22], 验证算法的正确性需要从旋翼飞行器的力学编排和响应速度进行分析。

使用 Simulink 建立如图 2-6 所示的旋翼飞行器的惯导系统的开环响应模型。所谓开环响应模型, 即没有引入 PD 和 PI 控制器对解算数据进行反馈处理^{[34][35][36]}。开环响应模型中的姿态解算部分如图 2-9 所示, 采用了前文所设计的算法。

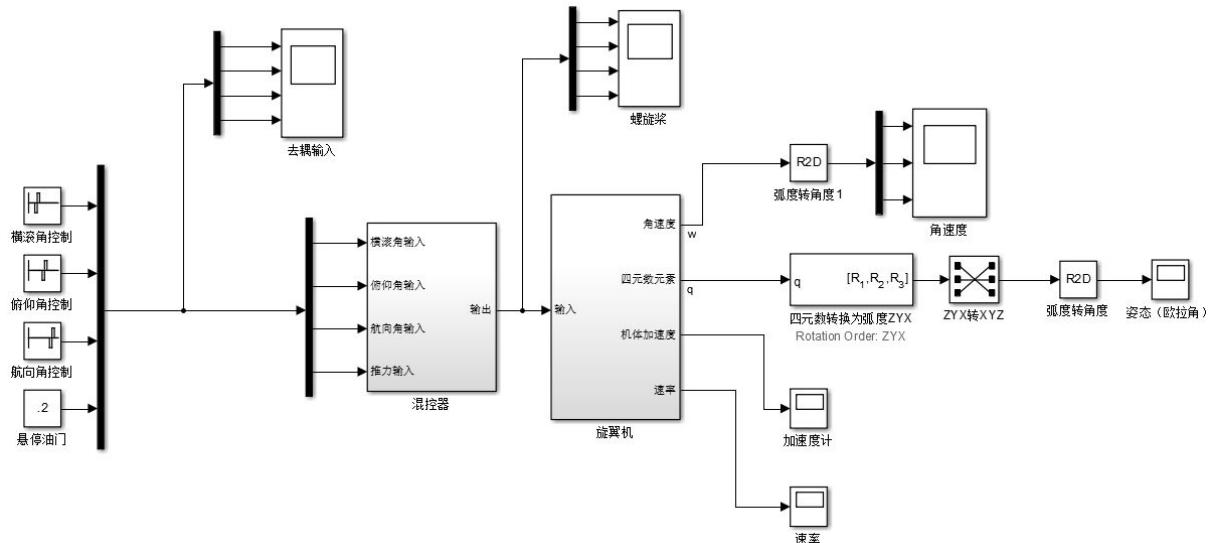


图 2-8 惯导系统的开环响应模型

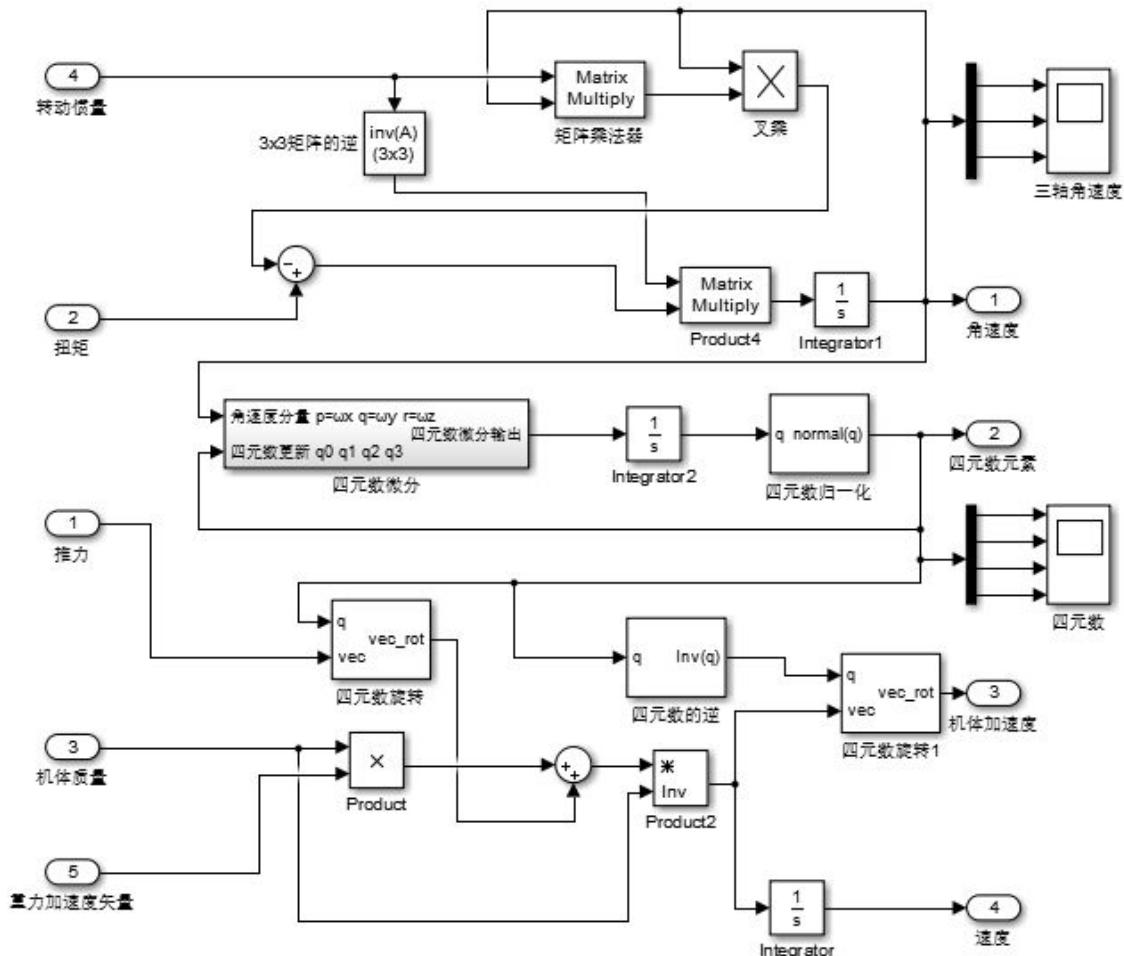


图 2-9 姿态解算的理论模型

旋翼飞行器中的混控器对 4 路油门进行简单解耦，为了方便将俯仰角控制、横滚角控制、航向角控制和悬停油门限制在了 $[-1, +1]$ 范围内，航向角限制在 $[-0.2, +0.2]$ 范围内，输出限制在 $[0, +1]$ 。

2.4.4 仿真结果及分析

仿真后输出的控制信号波形和解算姿态角分别如图 2-10、图 2-11 所示。

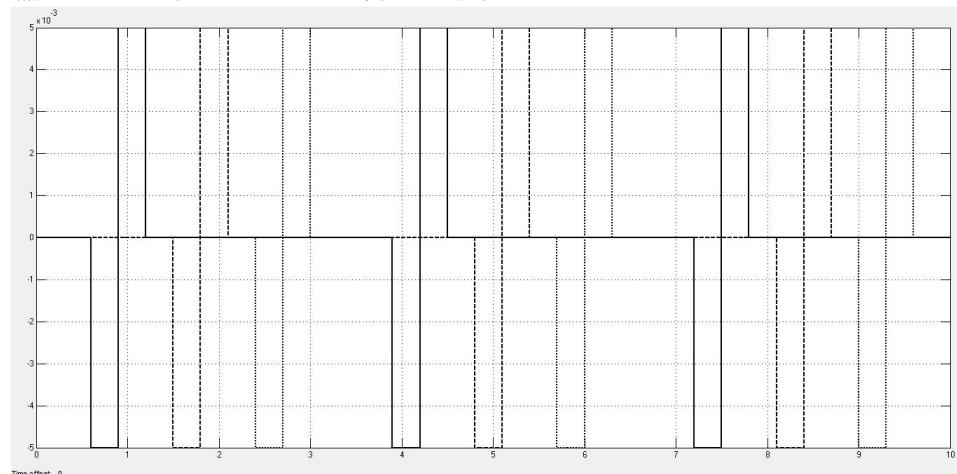


图 2-10 控制信号输入

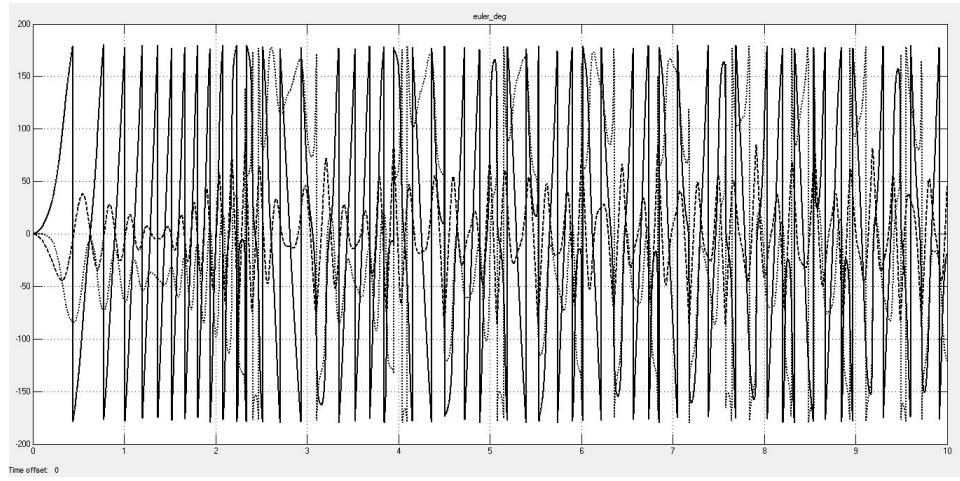


图 2-11 三轴姿态角输出

综合图 2-10、2-11，可发现以下规律：姿态角 3 个方向间耦合度很小，姿态响应较控制信号的输入慢，姿态变化导致升力方向变化进而引起旋翼机运动。在旋翼飞行器进行倾斜运动时，需要及时进行垂直速度补偿，否则将会减速而下降。若消除机体姿态角解算自然发散的趋势，既需要对输入数据和输出数据进行特殊耦合，这里使用前文所使用的互补滤波算法进行处理。

2.4.5 惯导算法的改进

根据 2.4.3 节得到的结论，在这里引入互补滤波脉冲做为欧拉角的输入。输入和输出分别如图 2-12 和 2-13 所示。

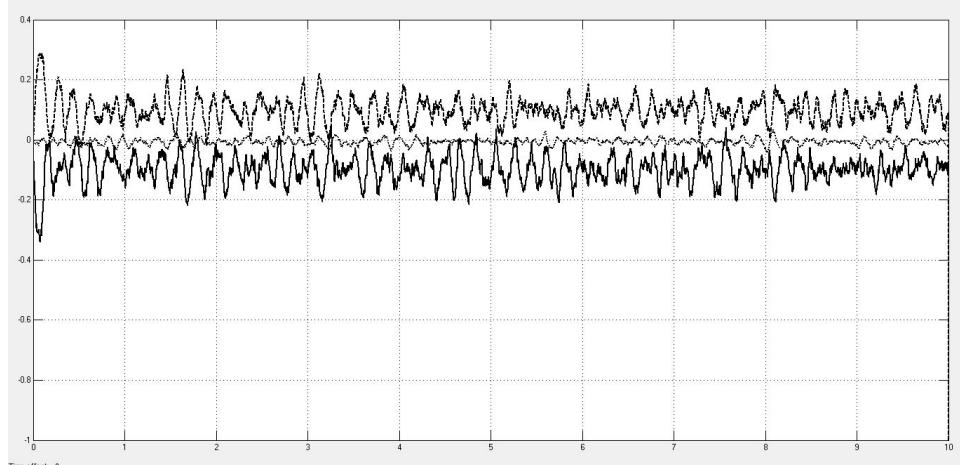


图 2-12 互补滤波控制信号输出

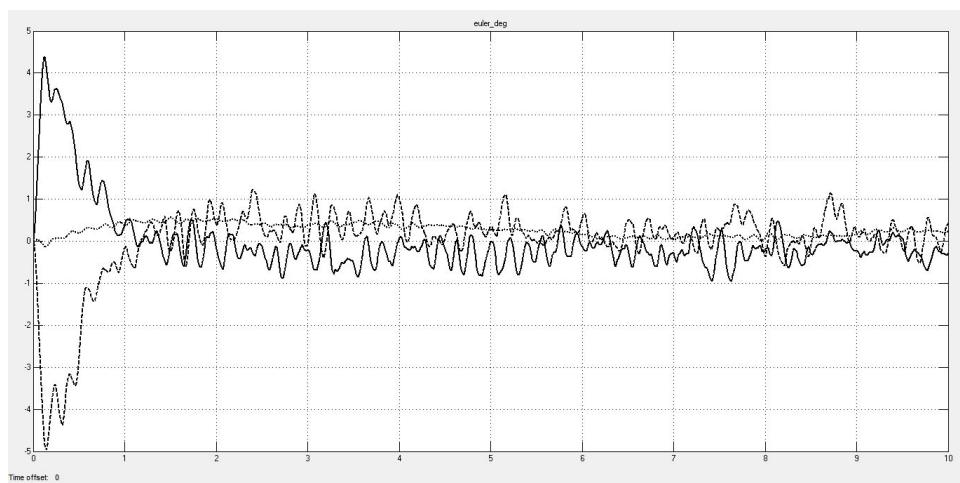


图 2-13 改进后姿态角输出

2.5 导航解算算法的仿真验证

2.5.1 仿真数据的采集

算法的仿真验证选择使用直接由惯性仪表（陀螺仪、加速度计和电子罗盘）得到的实测数据进行仿真分析。实测数据能真实反映随机误差和量化误差，如果能得到误差相对比较小的算法，那么在实际中执行时遇到的困难和问题比较少^{[36][37]}。

加速度计和陀螺仪选用了 InvenSense 公司的 9 轴运动处理组件 MPU6050，该芯片集成了 3 轴 MEMS 陀螺仪和 3 轴 MEMS 加速度计，免除了组合陀螺仪与加速器时之轴间差的问题。

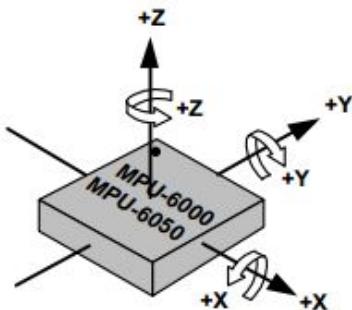


图 2-14 MPU6050 工作原理图

电子罗盘选用 Honeywell 公司的 HMC5883L 芯片。该芯片采用各向异性磁阻技术，具有在轴向高灵敏度和线性高精度的特点，测量范围从 1 毫高斯到 8 高斯。

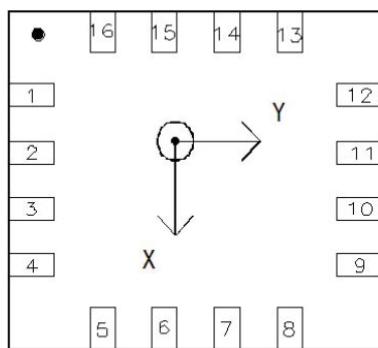


图 2-15 HCM5883L 工作原理图

采集数据平台处理器选用 STM32 芯片，采样时间为 0.002s，能够真实的仿真实际系统中惯性仪表的测量速度，使仿真结果更有说服性。

每次采集数据共有十二个值，分别为加速度计三轴测量值、电子罗盘三轴测量值、陀螺仪三轴测量值和通过处理器解算得到的三个姿态角。

本次仿真使用的采集数据共 27447 组。

2.5.2 无互补滤波的算法仿真

为了验证互补滤波算法对系统误差的影响，使用 Matlab 对无互补滤波补偿修正算法进行仿真验证。仿真结果如图 2-14、图 2-15 和图 2-16 所示，每幅图由上到下依次为 MCU 实测姿态角、本文算法解算姿态角和解算误差。

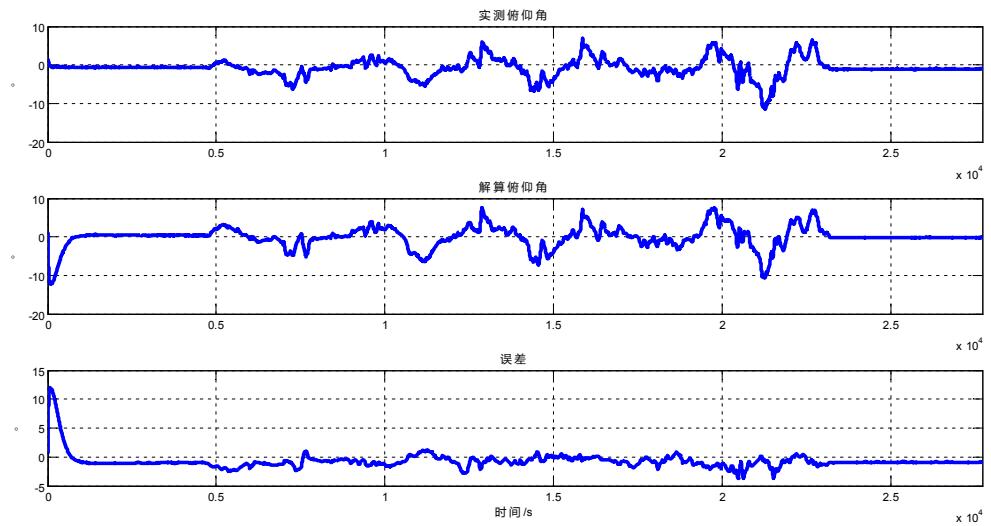


图 2-14 俯仰角实测值与无互补滤波修正的解算值比较

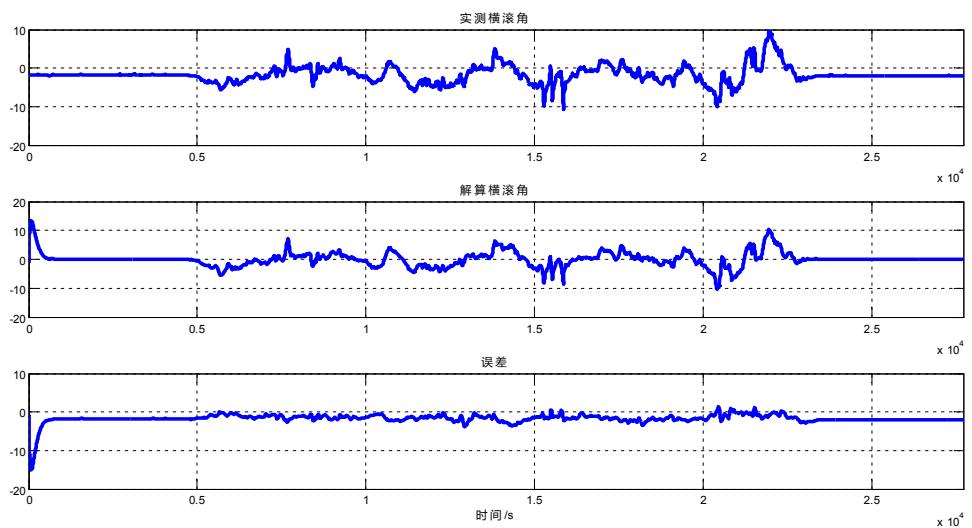


图 2-15 横滚角实测值与无互补滤波修正的解算值比较

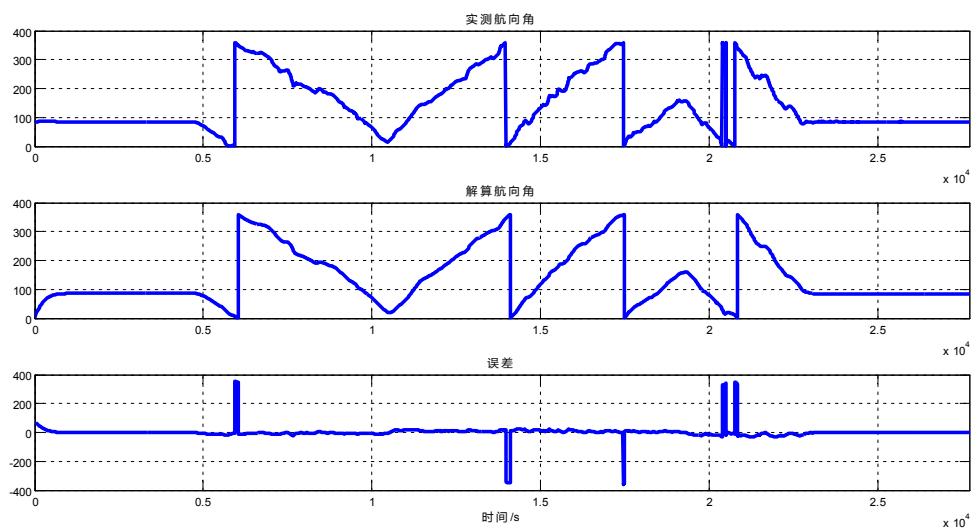


图 2-16 航向角实测值与无互补滤波修正的解算值比较

2.5.3 导航解算算法仿真

惯测数据经过互补滤波补偿修正算法的仿真结果如图 2-17、图 2-18 和图 2-19 所示。

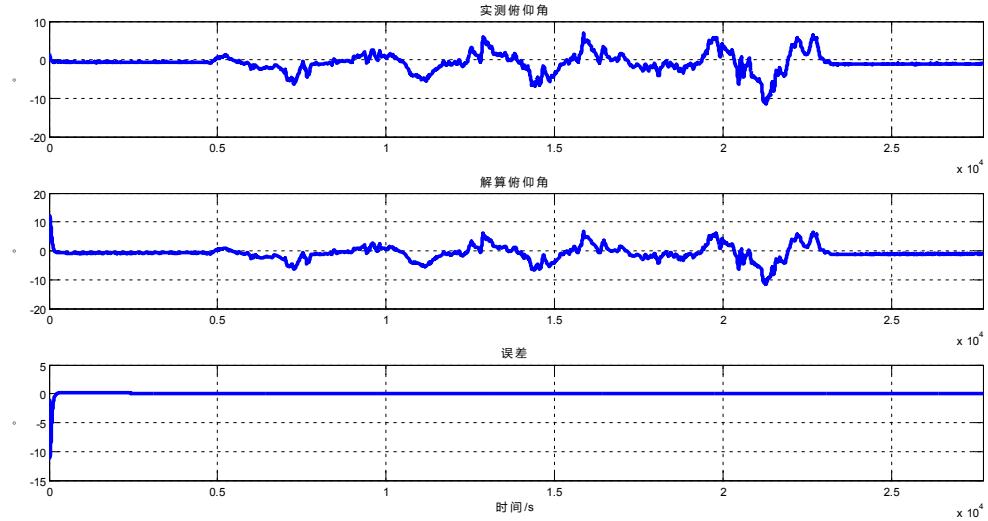


图 2-17 俯仰角实测值与互补滤波修正的解算值比较

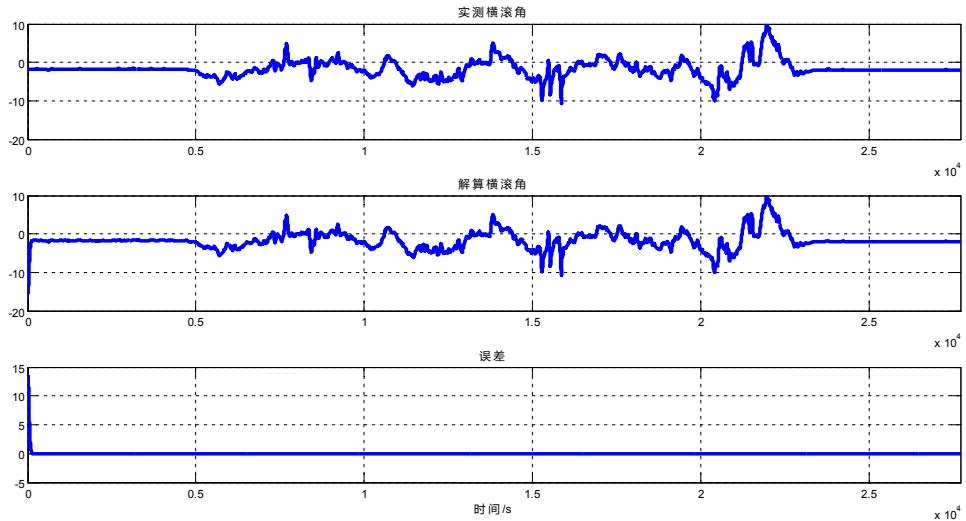


图 2-18 横滚角实测值与互补滤波修正的解算值比较

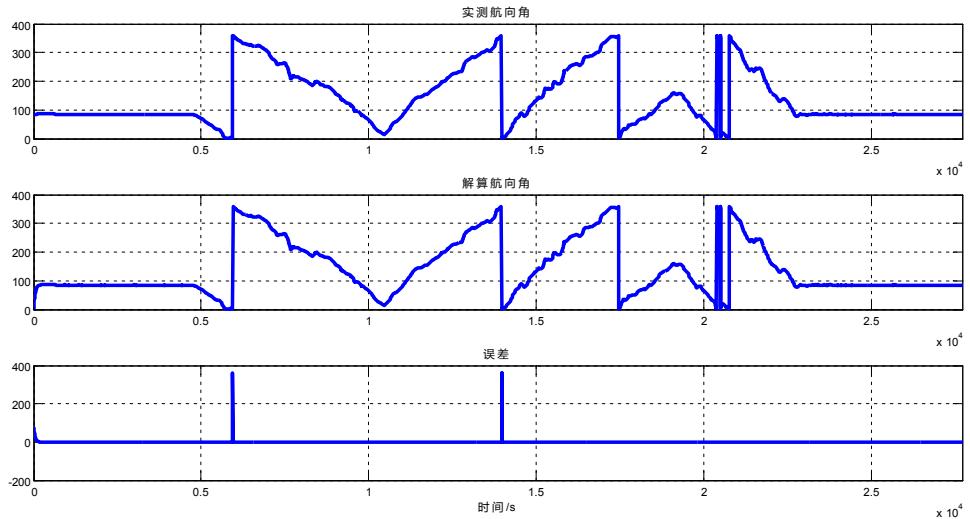


图 2-19 航向角实测值与互补滤波修正的解算值比较

2.5.4 仿真验证结论

传感器 9 轴惯测数据没有经过互补滤波算法处理后解算误差较大，对于旋翼机此类运行距离和运行姿态变率较小的飞行器，较大的误差会引起控制上出现较大偏差，造成严重影响。加入互补滤波算法处理后，解算误差基本为 0，可见互补滤波算法对本设计的影响显著，这也是本文创新性的引入互补滤波对 9 轴数据进行处理的结果。

通过对比可得到如下结论

- ① 在开始之后的短时间内误差较大，这是由于已设定四元数和姿态矩阵的初始值，系统需要将测量值代入，对四元数和姿态矩阵做收敛处理，这个过程需要一段时间。
- ② 误差基本为零，实测姿态角与解算姿态角的波形基本一致。
- ③ 算法准确，具有良好的可靠性和可行性，可以在此基础上进行硬线逻辑电路的设计。

3 导航解算电路的设计基础

根据第 2 章的算法推导，电路中的数据主要为浮点数，不仅需要加减乘除等常用的运算操作，而且需要开方、反三角函数等较复杂的运算操作。本章先给出电路设计过程中的数据格式定义，然后通过使用 Xilinx 公司的 CORE Generator 工具定制本设计中需要使用的运算逻辑 IP 核。

3.1 数据格式

电路设计中的数据格式主要为 32 位有符号单精度浮点数和 16 位有符号定点数。下面对这两种数据格式的表示方法做简要介绍。

3.1.1 浮点数的表示

电路中使用的浮点数表示方法完全符合 IEEE 754 标准，一般的用两个位宽参数来描述浮点数：总位宽 w 和小数位宽 w_f 。在 IEEE 754 标准中浮点数用符号位+指数+小数的组合形式表示，分别记为 s 、 E 和 $b_0 b_1 \dots b_{w_f-1}$ ，浮点数的值 m 可表示为

$$m = (-1)^s 2^E b_0 b_1 \dots b_{w_f-1} \quad (3.1)$$

其中，符号位 s 当 m 为正数时为 0，当 m 为负数时为 1。将上式进行加权展开，有

$$m = (-1)^s 2^E (1 + b_1 \cdot 2^{-1} + b_2 \cdot 2^{-2} + \dots + b_{w_f-1} \cdot 2^{-(w_f-1)}) \quad (3.2)$$

在进行将浮点数表示二进制形式时，需要使小数部分满足 $1 \leq b_0 b_1 \dots b_{w_f-1} < 2$ 。图 3-1 给出了浮点数的二进制表示形式。

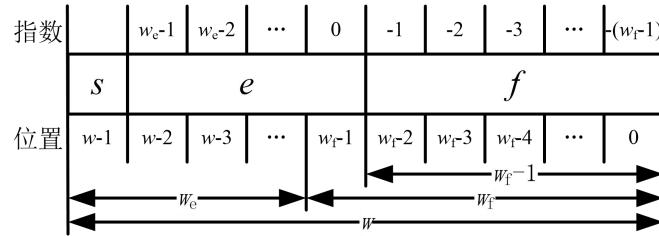


图 3-1 浮点数的二进制表示形式

在图 3-1 中， e 为指数域，位宽 $w_e = w - w_f$ ，且有

$$e = \sum_{i=0}^{w_e-1} e_i 2^i \quad (3.3)$$

指数值的符号 E 可表示为

$$E = e - (2^{w_e-1} - 1) \quad (3.4)$$

综上可得浮点数的二进制表示形式^{[38][39]}

$$m = (-1)^s 2^{\sum_{i=0}^{w_e-1} e_i 2^i - (2^{w_e-1} - 1)} (1 + b_1 \cdot 2^{-1} + b_2 \cdot 2^{-2} + \dots + b_{w_f-1} \cdot 2^{-(w_f-1)}) \quad (3.5)$$

另外在表 3-1 中给出一些特殊浮点数的二进制表示形式。

表 3-1 特殊浮点数的二进制表示形式

特殊浮点数	s	e	f
NaN	未定义	2^{w_e-1} , 即 $e=11\dots11$	10…00
$\pm\infty$	∞ 符号	2^{w_e-1} , 即 $e=11\dots11$	00…00
± 0	0 的符号	0	00…00

其中 NaN 的符号位是没有定义的，无穷大和零是有符号数，对符号的处理与有限非零数处理相同。

3.1.2 定点数的表示

电路中需要实现浮点数和定点数的相互转换，定点数与浮点数的表示类似，采用 2 的固定幂次乘以二进制补码数的表示形式，同样用 s 、 e 和 f 三个值域来表示，如图 3-2 所示。

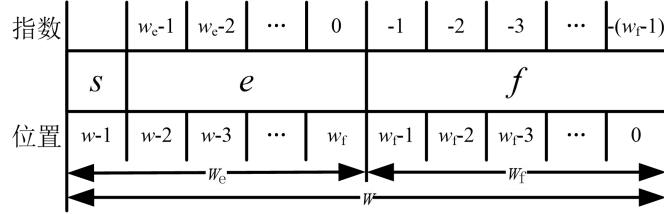


图 3-2 定点数的二进制表示形式

定点数的二进制表示形式为^{[38][39]}

$$n = (-1)^s 2^{w_e-1-w_f} + b_{w_e-2} \cdots b_{w_f} b_{w_f-1} \cdots b_1 b_0 = (-1)^{b_{w_e-1}} 2^{w_e-1-w_f} + \sum_{i=0}^{w_e-2} 2^{i-w_f} b_i \quad (3.6)$$

电路中，对于定点类型运算，其输入和输出都是定点数格式的，对于其他操作，输入和输出都是同类型浮点数格式。

对于某些特殊情况，需要满足下列限制。

① 指数位宽与小数位宽需满足

$$\text{最小指数位宽} = \text{ceil}[\log_2(\text{小数位宽}+3)]+1 \quad (3.7)$$

② 格式转换中，浮点数的指数位宽与定点数的总位宽需满足

$$\text{浮点数的最小指数位宽} = \text{ceil}[\log_2(\text{定点数总位宽})+3]+1 \quad (3.8)$$

3.2 反正切运算的算法原理及硬线逻辑设计

由于电路中需要使用多处定点数反正切运算，需要通过 Xilinx CORDIC IP 核通过运用坐标旋转数字计算来实现反正切运算^[40]。

3.2.1 基于 CORDIC 算法实现反正切运算的算法原理

对于矢量 (X, Y) 旋转 θ 弧度，记旋转后的矢量为 (X', Y') ，则有

$$\begin{cases} X' = X \cdot \cos \theta - Y \cdot \sin \theta \\ Y' = X \cdot \sin \theta + Y \cdot \cos \theta \\ \theta' = 0 \end{cases} \quad (3.9)$$

CORDIC 算法还可以实现连续 n 次小角度矢量旋转，每次旋转角度 $\arctan(2^{-i})$ ，称做微旋转^[36]。第 $i+1$ 次旋转的表达式如下

$$\begin{cases} x_{i+1} = x_i - \alpha_i \cdot y_i \cdot 2^{-i} \\ y_{i+1} = y_i + \alpha_i \cdot x_i \cdot 2^{-i} \\ \theta_{i+1} = \theta_i + \alpha_i \cdot \arctan(2^{-i}) \end{cases} \quad (3.10)$$

其中， $\alpha_i = \pm 1$ ， $i = 0, 1, 2, \dots, n$ 。

每一次微旋转都可以通过简单的移位和加法/减法操作来实现。通过 n 次迭代，可以得到第 n 次旋转的表达式

$$\begin{cases} X' = \prod_{i=1}^n \cos(\arctan(2^{-i}))(X_i - \alpha_i Y_i 2^{-i}) \\ Y' = \prod_{i=1}^n \cos(\arctan(2^{-i}))(Y_i + \alpha_i X_i 2^{-i}) \\ \theta' = \sum_{i=1}^n \theta - \alpha_i \cdot \arctan(2^{-i}) \end{cases} \quad (3.11)$$

其中， $\alpha_i = \pm 1$ ， $i = 0, 1, 2, \dots, n$ 。

① 矢量旋转：矢量 (X, Y) 经过旋转 θ 弧度之后得到新的矢量 (X', Y') 。通过参数 α_i 来选择矢量旋转方向：若旋转角 $\theta \geq 0$ ，则 $\alpha_i = -1$ ；若旋转角 $\theta < 0$ ，则 $\alpha_i = 1$ 。图 3-3 给出了 CORDIC 算法实现矢量旋转的示意图。

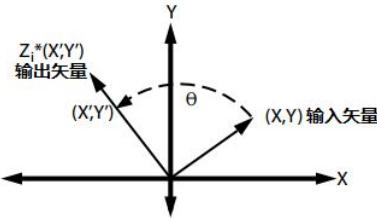


图 3-3 CORDIC 算法实现矢量旋转示意图

CORDIC 算法实现矢量旋转可如下表达

$$\begin{cases} X' = Z_n(X \cdot \cos \theta - Y \cdot \sin \theta) \\ Y' = Z_n(X \cdot \cos \theta + Y \cdot \sin \theta) \\ \theta' = 0 \end{cases} \quad (3.11)$$

式中， Z_n 是 CORDIC 算法的缩放因子

$$Z_n = \left(\prod_{i=1}^n \arccos(\arctan(2^{-i})) \right)^{-1} \quad (3.12)$$

② 矢量变换：矢量 (X, Y) 沿着单位圆旋转直到 Y 分量为 0 得到新的矢量 (X', Y') 。通过参数 α_i 来选择矢量旋转方向并最终实现 $Y' \rightarrow 0$ ：若旋转角 $Y_{i-1} \geq 0$ ，则 $\alpha_i = -1$ ；若旋转角 $Y_{i-1} < 0$ ，则 $\alpha_i = 1$ 。该过程的输入参数为矢量 (X, Y) 的幅度 X' 和相位 θ' ，输出变换后的相位和幅度。图 3-4 给出了 CORDIC 算法实现矢量变换的示意图。

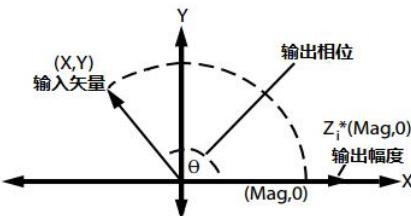


图 3-4 CORDIC 算法实现矢量变换示意图

CORDIC 算法实现矢量变换可如下表达

$$\begin{cases} X' = Z_n \cdot \sqrt{X^2 + Y^2} \\ Y' = 0 \\ \theta' = \arctan(Y / X) \end{cases} \quad (3.13)$$

式中, Z_n 是 CORDIC 算法的缩放因子

$$Z_n = \left(\prod_{i=1}^n \arccos(\arctan(2^{-i})) \right)^{-1} \quad (3.14)$$

③ 反正切算法: 矢量(X, Y)通过使用 CORDIC 算法沿单位圆旋转直到 Y 分量为 0 得到新矢量(X', Y')的旋转角度输出 $\arctan(Y / X)$ 。

3.2.2 反正切运算的 IP 核设计

Xilinx 坐标旋转数字计算 IP 核 CORDIC v4.0 能够实现解三角方程的功能^[40], 并且提供粗略旋转功能, 将输入采样全部旋转到第一象限进行数据处理, 同时可自动补偿 CORDIC 算法的缩放因子。CORDIC 核的接口说明和工作时序说明分别如图 3-5 和 3-6 所示。

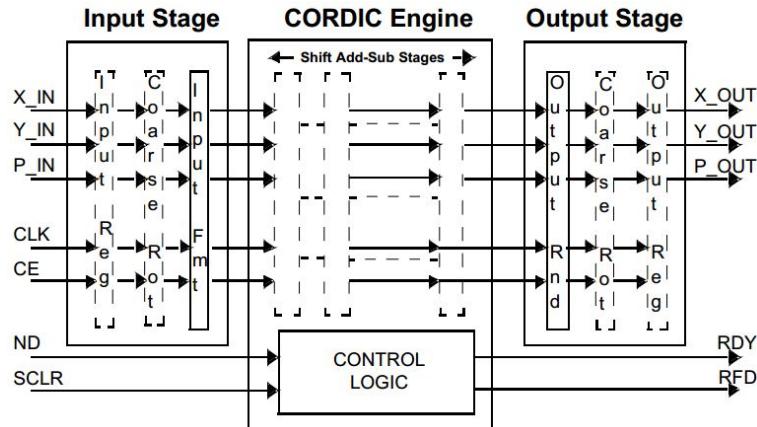


图 3-5 CORDIC 算法操作器接口图

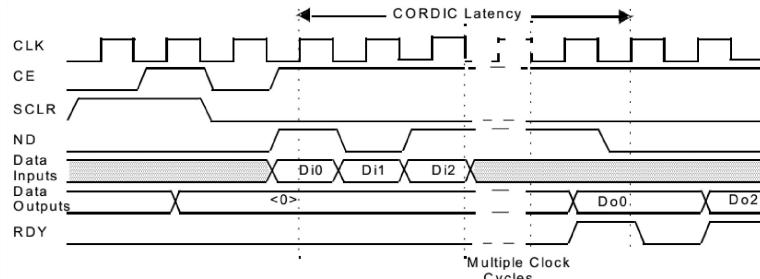


图 3-6 CORDIC 算法操作器时序图

对于姿态解算电路中使用的反正切函数仅需求解弧度值, 一般只用到了时钟信号 clk、时钟使能信号 ce、高电平清零信号 sclr、输入信号 x_in、输入信号 y_in、输出信号 p_out, 由于加速度计、陀螺仪和电子罗盘的数据采样频率为 500Hz, 而电路工作频率为 100MHz, 同时为了保证数据处理的时钟余度足够以使数据充分收敛, 还需使用到操作开始标志 nd、操作结束标志 rdy。

反正切函数输出数据表达式即为

$$p_{out}[DataWidth-1:0] = \arctan\left(\frac{y[DataWidth-1:0]}{x[DataWidth-1:0]}\right) \quad (3.15)$$

其中 $p_{out}[DataWidth-1]$ 为输出的符号位, 输出的单位为弧度。

CORDIC 矢量变换算法得到的输出相位的准确性受到输入矢量(X, Y)的高位幅度宽度的限制。另外求长度为 0 的矢量(0,0)的反正切值是非法操作, 其输出值是不可预期的。

基于 CORDIC 算法的反正切函数 IP 的输入输出范围如表 3-2 所示。

表 3-2 arctan 的输入输出范围

信号	范围	说明
X	[-1, 1]	输入矢量的 X 坐标
Y	[-1, 1]	输入矢量的 Y 坐标
PHASE	$[-\pi, \pi]$	输出角度

若输入值超出表 3-2 所列的范围, arctan 会产生不可预期的结果。因此在 X 和 Y 信号输入之前需要进行归一化处理

$$\begin{cases} X_{in} = \frac{X}{\sqrt{X^2 + Y^2}} \\ Y_{in} = \frac{Y}{\sqrt{X^2 + Y^2}} \end{cases} \quad (3.16)$$

使输入信号在规定的范围之内, 以满足 IP 对数据处理的要求。

考虑到使定制的 IP 占用逻辑资源数量达到最少, 配置过程中使用多时钟数据传输的串行结构并使用最优化的流水线处理方式, 输入为总位宽为 16、小数位宽为 14 的定点数, 输出为总位宽为 16、小数位宽为 13 的定点数。

3.3 浮点数运算的硬线逻辑设计

通过第 2 章中给出的基本算法原理, 整个算法流程中的浮点数操作有浮点数加法/减法运算、浮点数乘法运算、浮点数除法运算、浮点数取平方根运算、浮点数比较运算等、浮点数到定点数转换运算、定点数到浮点数转换运算。

Xilinx 浮点数操作器 IP 核 Floating-point v5.0 具有在 FPGA 上进行浮点计算功能^[38], 其操作、字长、延时和接口都是可配置的, 定制非常方便, 减小了电路设计的复杂度。浮点数操作器的接口说明和工作时序说明分别如图 3-7 和 3-8 所示。

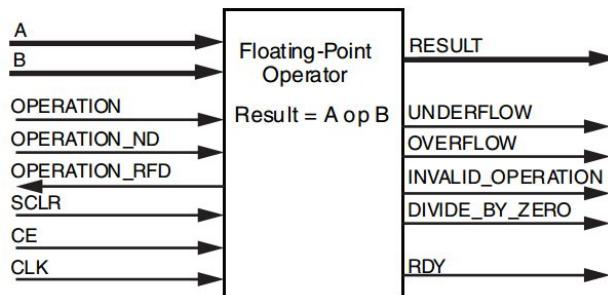


图 3-7 浮点数操作器接口图

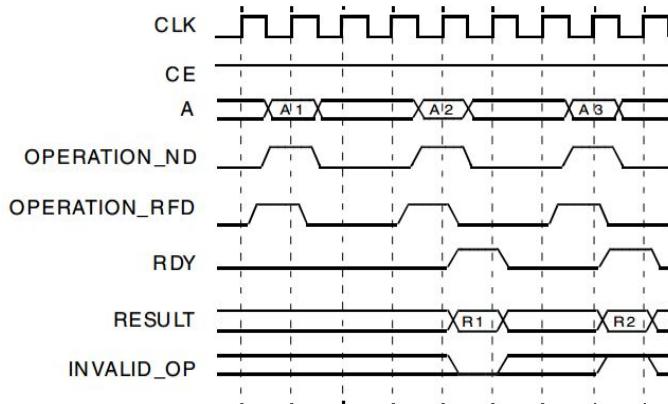


图 3-8 浮点数操作器时序图

对于姿态解算电路, 一般只用到了时钟信号 clk、时钟使能信号 ce、高电平清零信

号 sclr、输入信号 a、输入信号 b、输出信号 result，在进行加法/减法和比较运算时，还需要使用 operation。同时为了保证数据处理的时钟余度足够以使数据充分收敛，因此还需使用到操作开始标志 operation_nd、操作结束标志 rdy。

3.3.1 浮点数加法/减法器的设计

浮点数加法/减法器的输入端为 a[DataWidth-1:0]、b[DataWidth-1:0]、operation[7:0]，输出端为 result[DataWidth-1:0]。

运算表达式为

$$\text{result}[\text{DataWidth - 1 : 0}] = \text{a}[\text{DataWidth - 1 : 0}] \pm \text{b}[\text{DataWidth - 1 : 0}] \quad (3.17)$$

由于浮点数加法/减法器并不能同时实现加法和减法运算，因此在输入数据同时需要通过 operation[7:0]端口对操作器进行配置，具体配置操作如表 4-3 所示。

表 3-3 浮点数加法/减法器的配置说明

操作类型	operation[7:0]
加法	8'b0000 0000
减法	8'b0000 0001

由于浮点数加法/减法器需要调用的次数较多，为了保证定制的 IP 占用逻辑资源数量达到最少，配置过程中使用最佳数量的 DSP48，延时与 LUTs、FFs 和 Speed 的关系如图 3-9 所示，将浮点数加法/减法器设置为 6 个延时。



图 3-9 浮点数加法/减法器延时与资源用量的关系

3.3.2 浮点数乘法器的设计

浮点数乘法器的输入端为 a[DataWidth-1:0]、b[DataWidth-1:0]，输出端为 result[DataWidth-1:0]。

运算表达式为

$$\text{result}[\text{DataWidth - 1 : 0}] = \text{a}[\text{DataWidth - 1 : 0}] \times \text{b}[\text{DataWidth - 1 : 0}] \quad (3.18)$$

由于浮点数乘法器需要调用的次数最多，配置过程中最大化使用 DSP48，延时与 LUTs、FFs 和 Speed 的关系如图 3-10 所示，将浮点数加法/减法器设置为 4 个延时。



图 3-10 浮点数乘法器延时与资源用量的关系

3.3.3 浮点数除法器的设计

浮点数乘法器的输入端为 $a[\text{DataWidth}-1:0]$ 、 $b[\text{DataWidth}-1:0]$ ，输出端为 $\text{result}[\text{DataWidth}-1:0]$ 。

运算表达式为

$$\text{result}[\text{DataWidth}-1:0] = a[\text{DataWidth}-1:0] \div b[\text{DataWidth}-1:0] \quad (3.19)$$

3.3.4 浮点数开方器的设计

浮点数开方操作器的输入端为 $a[\text{DataWidth}-1:0]$ ，输出端为 $\text{result}[\text{DataWidth}-1:0]$ 。

运算表达式为

$$\text{result}[\text{DataWidth}-1:0] = \sqrt{a[\text{DataWidth}-1:0]} \quad (3.20)$$

3.3.5 浮点数比较器的设计

浮点数比较器的输入端为 $a[\text{DataWidth}-1:0]$ 、 $b[\text{DataWidth}-1:0]$ 、 $\text{operation}[7:0]$ ，输出端为 $\text{result}[7:0]$ 。

输入 $a[\text{DataWidth}-1:0]$ 为比较数，输入 $b[\text{DataWidth}-1:0]$ 为被比较数，若 $a[\text{DataWidth}-1:0]$ 与 $b[\text{DataWidth}-1:0]$ 比较结果满足预设比较方式，则 $\text{result}[7:0]$ 输出 $8'b0000\ 0001$ ，否则输出 $8'b0000\ 0000$ 。

由于浮点数比较器并不能同时实现不同的比较运算，因此在输入数据同时需要通过 $\text{operation}[7:0]$ 端口对操作器进行配置，具体配置操作如表 3-4 所示。

表 3-4 浮点数比较器的配置说明

操作类型	s_axis_operation_tvalid[7:0]
大于	8'b0010 0100
等于	8'b0001 0100
小于	8'b0000 1100

3.3.6 浮点数与定点数转换器的设计

根据 3.2.2 节中反正切函数 IP 核的设计方案，需要将输入由 32 位有符号单精度浮点数转换为总位宽 10 位小数位宽 8 位有符号定点数，经过反正切 IP 处理后的输出再由总位宽 10 位小数位宽 7 位有符号定点数转换为 32 位有符号单精度浮点数。

在设计中例化的 IP 核详见附录 A。

4 导航解算电路的功能描述与 RTL 实现

导航解算电路是一个由有限状态机控制的多模块串行操作电路，它包括顶层控制电路（INS_top）、三轴加速度计测量误差计算电路（INS_acc）、三轴电子罗盘测量误差计算电路（INS_mag）、三轴陀螺仪测量误差修正电路（INS_gyro）、四元数更新电路（INS_quat）、姿态转换矩阵更新电路（INS_matx）和三轴姿态角解算电路（INS_euler）。

根据第 2、3 章中的算法推导和电路设计基础，按照“自顶向下”的设计原则，从顶层电路开始，分别给出电路中每个模块的接口描述、RTL 实现原理和各个模块的状态迁移图。

在系统复位状态下，电路中所有接口对应的寄存器都为空，姿态转换矩阵复位为 $C_n^b = \text{diag}[1 \ 1 \ 1]$ ，四元数复位为 $Q = [q_0 \ q_1 \ q_2 \ q_3]^T = [1 \ 0 \ 0 \ 0]^T$ 。在将前级电路（传感器接口电路）采样的九轴数据全部读取完毕后，芯片总控电路 CONTROLLER 将 in_data_en 拉高，电路检测到开始进行工作，直到 out_INS_finish 被拉高，标志一次解算工作完毕，并将解算后的三轴姿态角及本次解算过程中更新的姿态转换矩阵九个元素输出。

4.1 加速度计测量误差计算电路的设计

4.1.1 加速度计测量误差计算电路的接口设计

加速度计测量误差计算电路用来计算加速度计测量数据与参考比力之间的相对误差，图 4-1 给出了加速度计测量误差计算电路的设计接口框图，表 4-1 则给出了接口中各个输入和输出信号的具体描述。

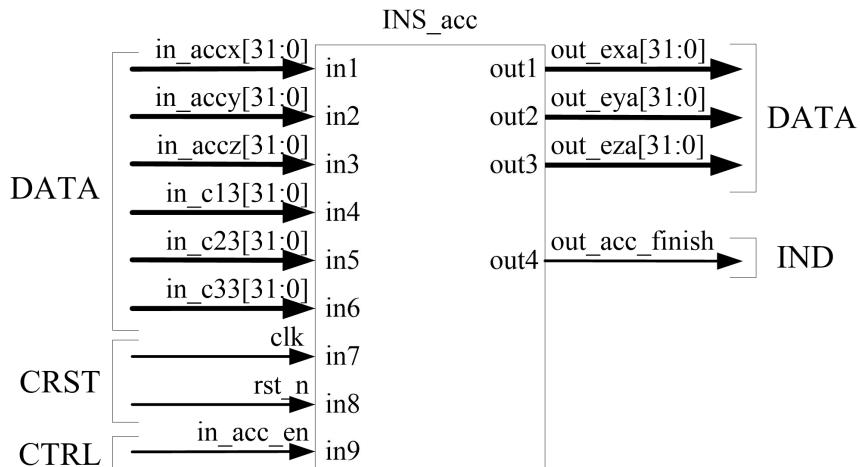


图 4-1 加速度计测量误差计算电路的接口框图

表 4-1 加速度计测量误差计算电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步复位)
数据接口 (DATA)			
in_accx[31:0]	输入	clk	加速度计 x 轴测量值
in_accy[31:0]	输入	clk	加速度计 y 轴测量值
in_accz[31:0]	输入	clk	加速度计 z 轴测量值
in_c13[31:0]	输入	clk	前一次姿态矩阵更新后第 1 行第 3 列元素值

续表 4-1

数据接口 (DATA)			
in_c23[31:0]	输入	clk	前一次姿态矩阵更新后第 2 行第 3 列元素值
in_c33[31:0]	输入	clk	前一次姿态矩阵更新后第 3 行第 3 列元素值
out_exax[31:0]	输出	clk	加速度计 x 轴测量误差值
out_eyay[31:0]	输出	clk	加速度计 y 轴测量误差值
out_ezaz[31:0]	输出	clk	加速度计 z 轴测量误差值
控制与指示接口 (CTRL、IND)			
in_acc_en	输入	clk	电路使能信号 (高电平有效), 当 3 轴加速度数据 数据全部读取完毕, 该信号被置为高电平
out_acc_finish	输出	clk	加速度计测量误差计算标志信号 (高电平有效)

4.1.2 加速度计测量误差计算电路框图

加速度计测量误差计算电路由控制电路工作时序的有限状态机、浮点数加/减法器、浮点数乘法器、浮点数除法器、浮点数开方器和与其他电路互联的接口。为了描述简便, 根据第 2 章中算法推导, 该电路可分为三轴测量值归一化电路 (ACC_NOR) 和测量误差计算电路 (ACC_CAL), 由于电路设计过程中使用了大量逻辑优化, 这里不再给出加速度计测量误差计算电路底层 IP 核间的电气连接和状态机控制电路的电气连接原理图。加速度计测量误差计算电路的框图和状态迁移图分别如图 4-2 和图 4-3 所示。

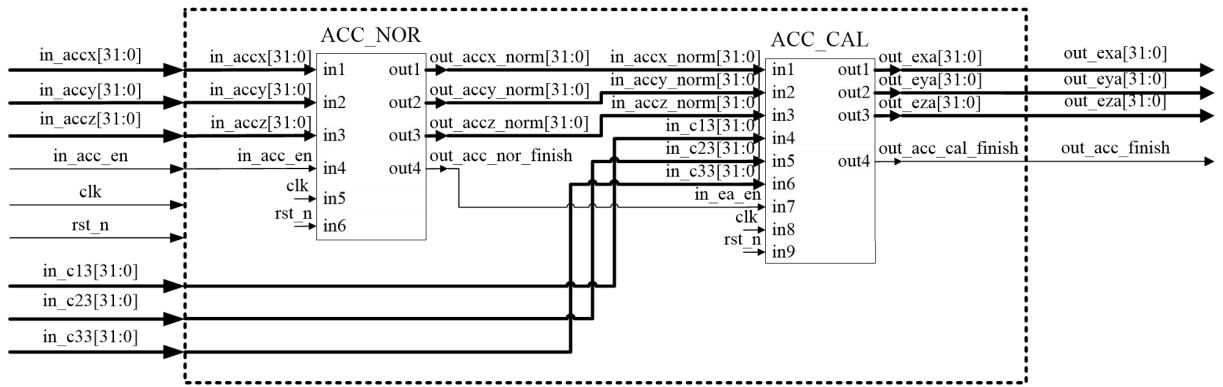


图 4-2 加速度计测量误差计算电路的框图

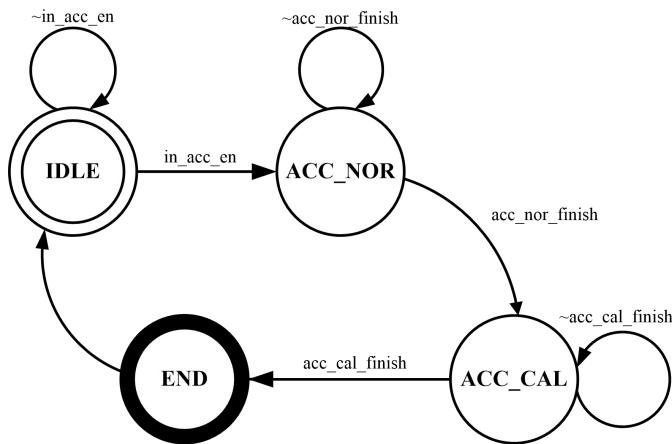


图 4-3 加速度计测量误差计算电路的状态迁移图

加速度计测量误差计算电路采用 5 级流水线结构进行数据处理, 内部共有 66 个状态, 包括高低电平信号检测状态和标志位采样状态。为方便描述, 合并为 4 个子状态, 如图 4-3 所示, 分别为 IDLE (初始状态)、三轴测量值归一化状态 (ACC_NOR)、测

量误差计算状态（ACC_CAL）和结束状态（END）。

在系统复位时，状态置为到初始状态将电路中所有接口对应的寄存器都置为空，所有辅助信号都置为低电平。当控制信号 in_acc_en 被顶层电路的状态机置为高电平后，进行加速度计测量数据和上次解算过程得到的姿态转换矩阵第 3 列元素的读取操作，状态跳转至 ACC_NOR，将三轴测量值归一化电路的辅助控制信号置为高电平使其开始工作；当三轴测量值归一化完成后，输出辅助信号 acc_nor_finish 被置为高电平后，状态跳转至 ACC_CAL，测量误差计算电路开始工作；当测量误差计算电路将数据处理完毕，输出辅助信号 acc_cal_finish 被置为高电平后，状态跳转至 END；在 END 状态下将计算得到的三轴误差值输出，并跳转至初始状态 IDLE。至此，一次加速度计测量误差计算过程结束。

4.2 电子罗盘测量误差计算电路的设计

4.2.1 电子罗盘测量误差计算电路的接口设计

电子罗盘测量误差计算电路用来计算电子罗盘测量数据与参考磁力之间的相对误差，图 4-4 给出了电子罗盘测量误差计算电路的设计接口框图，表 4-2 则给出了接口中各个输入和输出信号的具体描述。

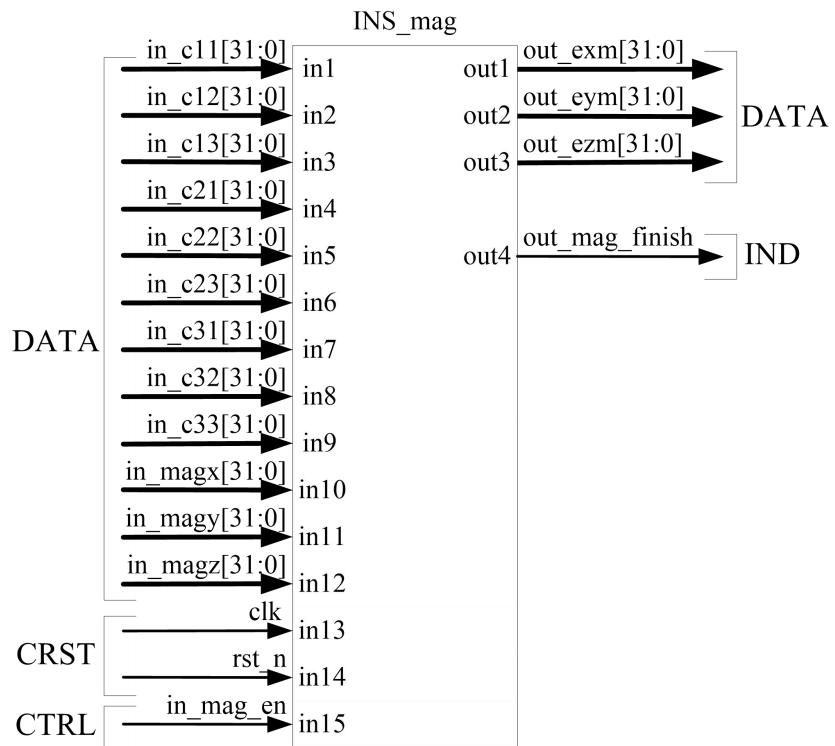


图 4-4 电子罗盘测量误差计算电路的接口框图

表 4-2 电子罗盘测量误差计算电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步复位)
数据接口 (DATA)			
in_magx[31:0]	输入	clk	电子罗盘 x 轴测量值
in_magy[31:0]	输入	clk	电子罗盘 y 轴测量值
in_magz[31:0]	输入	clk	电子罗盘 z 轴测量值
in_c11[31:0]	输入	clk	前一次姿态矩阵更新后第 1 行第 1 列元素值

续表 4-2

数据接口 (DATA)

in_c12[31:0]	输入	clk	前一次姿态矩阵更新后第 1 行第 2 列元素值
in_c13[31:0]	输入	clk	前一次姿态矩阵更新后第 1 行第 3 列元素值
in_c21[31:0]	输入	clk	前一次姿态矩阵更新后第 2 行第 1 列元素值
in_c22[31:0]	输入	clk	前一次姿态矩阵更新后第 2 行第 2 列元素值
in_c23[31:0]	输入	clk	前一次姿态矩阵更新后第 2 行第 3 列元素值
in_c31[31:0]	输入	clk	前一次姿态矩阵更新后第 3 行第 1 列元素值
in_c32[31:0]	输入	clk	前一次姿态矩阵更新后第 3 行第 2 列元素值
in_c33[31:0]	输入	clk	前一次姿态矩阵更新后第 3 行第 3 列元素值
out_exm[31:0]	输出	clk	电子罗盘 x 轴测量误差值
out_eym[31:0]	输出	clk	电子罗盘 y 轴测量误差值
out_ezm[31:0]	输出	clk	电子罗盘 z 轴测量误差值

控制与指示接口 (CTRL、IND)

in_mag_en	输入	clk	电路使能信号 (高电平有效), 当 3 轴电子罗盘数据全部读取完毕, 该信号被置为高电平
out_mag_finish	输出	clk	电子罗盘测量误差计算标志信号 (高电平有效)

4.2.2 电子罗盘测量误差计算电路框图

电子罗盘测量误差计算电路由控制电路工作时序的有限状态机、浮点数加/减法器、浮点数乘法器、浮点数除法器、浮点数开方器和与其他电路互联的接口。为了描述简便, 根据第 2 章中算法推导, 该电路可分为三轴测量值归一化电路 (MAG_NOR)、参考磁力线计算电路 (MAG_FLU) 和测量误差计算电路 (MAG_CAL), 由于电路设计过程中使用了大量逻辑优化, 这里不再给出电子罗盘测量误差计算电路底层 IP 核间的电气连接和状态机控制电路的电气连接原理图。电子罗盘测量误差计算电路的框图和状态迁移图分别如图 4-5 和图 4-6 所示。

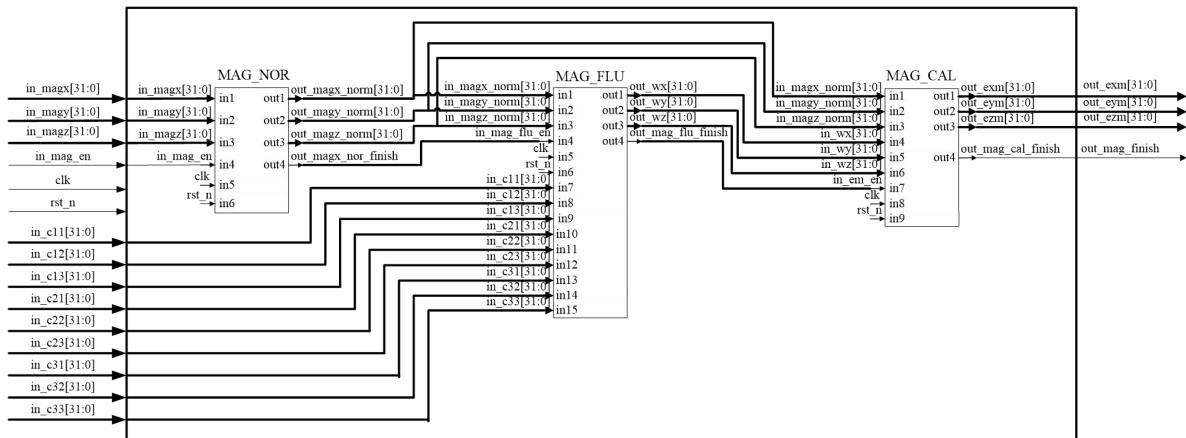


图 4-5 电子罗盘测量误差计算电路的框图

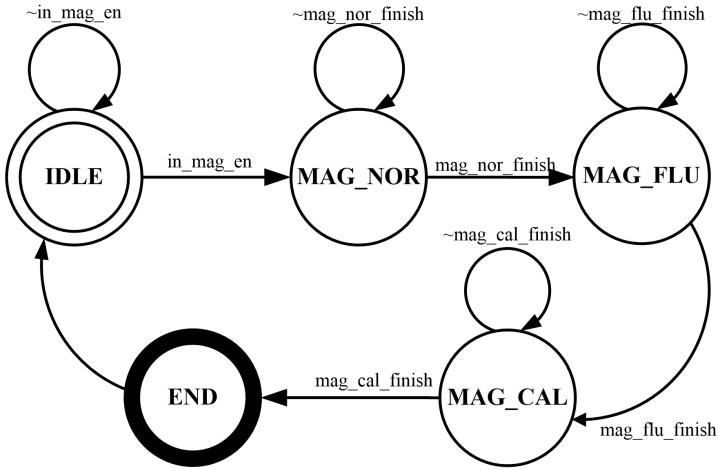


图 4-6 电子罗盘测量误差计算电路的状态迁移图

电子罗盘测量误差计算电路采用 5 级流水线结构进行数据处理，内部共有 138 个状态，包括高低电平信号检测状态和标志位采样状态。为方便描述，合并为 6 个子状态，如图 4-6 所示，分别为 IDLE（初始状态）、三轴测量值归一化状态（MAG_NOR）、参考磁力线计算状态（MAG_FLU）、测量误差计算状态（MAG_CAL）和结束状态（END）。

在系统复位时，状态置为到初始状态将电路中所有接口对应的寄存器都置为空，所有辅助信号都置为低电平。当控制信号 in_mag_en 被顶层电路的状态机置为高电平后，进行电子罗盘测量数据和上次解算过程得到的姿态转换矩阵 9 个元素的读取操作，状态跳转至 MAG_NOR，将三轴测量值归一化电路的辅助控制信号置为高电平使其开始工作；当三轴测量值归一化完成后，输出辅助信号 mag_nor_finish 被置为高电平后，状态跳转至 MAG_FLU，参考磁力线计算电路开始工作；当参考磁力线计算完成后，输出辅助信号 mag_flu_finish 被置为高电平后，状态跳转至 MAG_CAL，测量误差计算电路开始工作；当测量误差计算电路将数据处理完毕，输出辅助信号 mag_cal_finish 被置为高电平后，状态跳转至 END；在 END 状态下将计算得到的三轴误差值输出，并跳转至初始状态 IDLE。至此，一次电子罗盘测量误差计算过程结束。

4.3 陀螺仪测量误差修正计算电路的设计

4.3.1 陀螺仪测量误差修正电路的接口设计

陀螺仪测量误差修正电路使用加速度计测量误差计算值和电子罗盘测量误差计算值通过互补滤波算法对陀螺仪测量误差进行修正以满足后续电路的使用需求，图 4-7 给出了陀螺仪测量误差修正电路的设计接口框图，表 4-3 则给出了接口中各个输入和输出信号的具体描述。

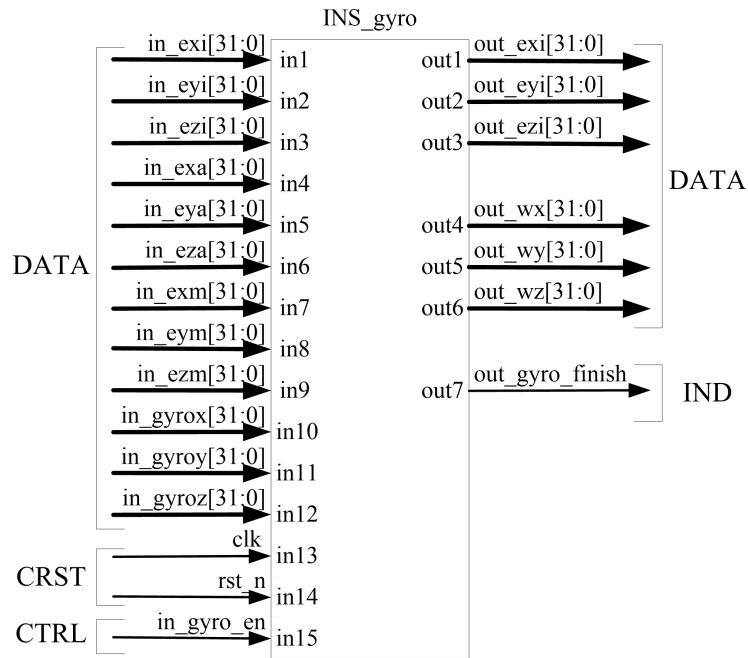


图 4-7 电子罗盘测量误差计算电路的接口框图

表 4-3 电子罗盘测量误差计算电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步复位)
数据接口 (DATA)			
in_gyrox[31:0]	输入	clk	陀螺仪 x 轴测量值
in_gyroy[31:0]	输入	clk	陀螺仪 y 轴测量值
in_gyroz[31:0]	输入	clk	陀螺仪 z 轴测量值
in_exa[31:0]	输入	clk	加速度计 x 轴测量误差值
in_eya[31:0]	输入	clk	加速度计 y 轴测量误差值
in_eza[31:0]	输入	clk	加速度计 z 轴测量误差值
in_exm[31:0]	输入	clk	电子罗盘 x 轴测量误差值
in_eym[31:0]	输入	clk	电子罗盘 y 轴测量误差值
in_ezm[31:0]	输入	clk	电子罗盘 z 轴测量误差值
in_exi[31:0]	输入	clk	前一次 x 轴积分误差
in_eyi[31:0]	输入	clk	前一次 y 轴积分误差
in_ezi[31:0]	输入	clk	前一次 z 轴积分误差
out_wx[31:0]	输出	clk	陀螺仪 x 轴测量误差修正值
out_wy[31:0]	输出	clk	陀螺仪 y 轴测量误差修正值
out_wz[31:0]	输出	clk	陀螺仪 z 轴测量误差修正值
续表 4-4			
控制与指示接口 (CTRL、IND)			
in_gyro_en	输入	clk	电路使能信号 (高电平有效), 当 3 轴陀螺仪数据和 6 轴误差数据全部读取完毕, 该信号被置为高电平
out_gyro_finish	输出	clk	陀螺仪测量误差修正标志信号 (高电平有效)

4.3.2 陀螺仪测量误差修正电路框图

陀螺仪测量误差修正电路由控制电路工作时序的有限状态机、浮点数加/减法器、浮

点数乘法器和与其他电路互联的接口。为了描述简便，根据第 2 章中算法推导，该电路可分为比例积分误差计算电路（GYRO_KPI）和测量误差修正电路（GYRO_RSV），由于电路设计过程中使用了大量逻辑优化，这里不再给出陀螺仪测量误差修正电路底层 IP 核间的电气连接和状态机控制电路的电气连接原理图。陀螺仪测量误差修正电路的框图和状态迁移图分别如图 4-8 和图 4-9 所示。

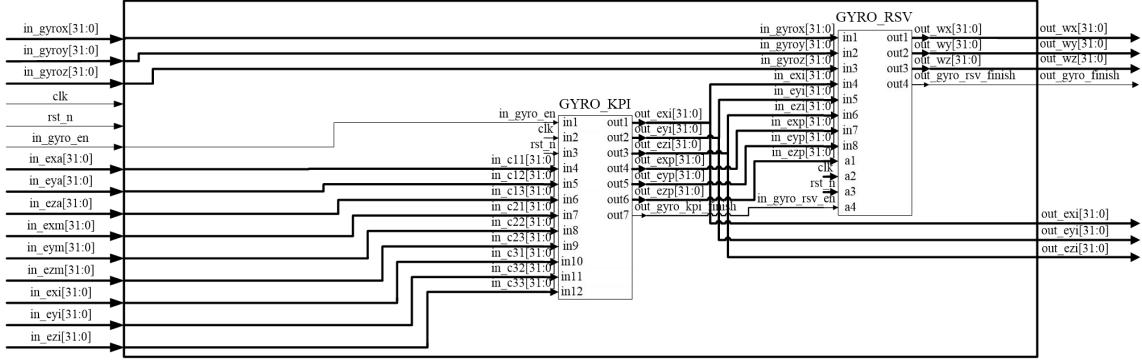


图 4-8 陀螺仪测量误差修正电路的框图

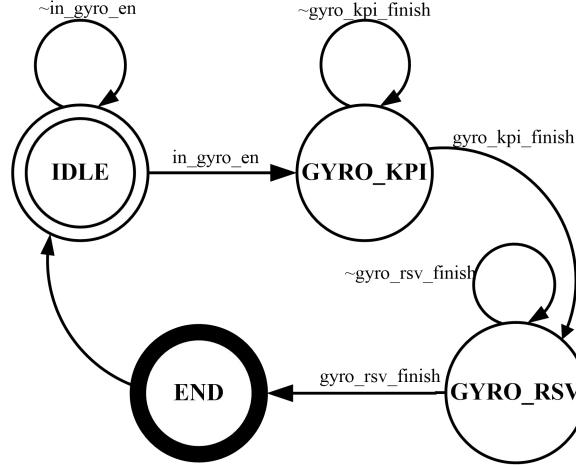


图 4-9 陀螺仪测量误差修正电路的状态迁移图

陀螺仪测量误差修正电路采用 2 级流水线结构进行数据处理，内部共有 82 个状态，包括高低电平信号检测状态和标志位采样状态。为方便描述，合并为 4 个子状态，如图 4-9 所示，分别为 IDLE（初始状态）、比例积分误差计算工作状态（GYRO_KPI）、测量误差修正工作状态（GYRO_RSV）和结束状态（END）。

在系统复位时，状态置为到初始状态将电路中所有接口对应的寄存器都置为空，所有辅助信号都置为低电平。当控制信号 `in_gyro_en` 被顶层电路的状态机置为高电平后，进行电子罗盘测量数据、加速度计测量误差计算数据和电子罗盘测量误差计算数据的读取操作，状态跳转至 GYRO_KPI，将比例积分误差计算电路的辅助控制信号置为高电平使其开始工作；当比例积分误差计算完成后，输出辅助信号 `gyro_kpi_finish` 被置为高电平后，状态跳转 GYRO_RSV，测量误差修正电路开始工作；当测量误差修正电路将数据处理完毕，输出辅助信号 `gyro_rsv_finish` 被置为高电平后，状态跳转至 END；在 END 状态下将计算得到的三轴修正值与三轴积分误差值输出，并跳转至初始状态 IDLE。至此，一次陀螺仪测量误差修正过程结束。

4.4 四元数更新电路的设计

4.4.1 四元数更新电路的接口设计

四元数更新电路使用陀螺仪测量误差修正值通过四元数微分方程和一阶龙格-库塔

算法对四元数进行更新，图 4-10 给出了四元数更新电路的设计接口框图，表 4-4 则给出了接口中各个输入和输出信号的具体描述。

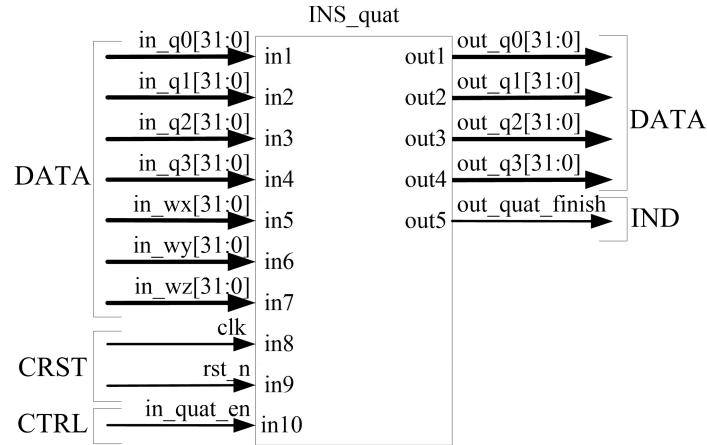


图 4-10 四元数更新电路的接口框图

表 4-4 四元数更新电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步复位)
数据接口 (DATA)			
in_wx[31:0]	输入	clk	陀螺仪 x 轴测量误差修正值
in_wy[31:0]	输入	clk	陀螺仪 y 轴测量误差修正值
in_wz[31:0]	输入	clk	陀螺仪 z 轴测量误差修正值
in_q0[31:0]	输入	clk	前一次四元数更新标量值
in_q1[31:0]	输入	clk	前一次四元数更新 i 方向矢量值
in_q2[31:0]	输入	clk	前一次四元数更新 j 方向矢量值
in_q3[31:0]	输入	clk	前一次四元数更新 k 方向矢量值
out_q0[31:0]	输出	clk	本次四元数更新标量值
out_q1[31:0]	输出	clk	本次四元数更新 i 方向矢量值
out_q2[31:0]	输出	clk	本次四元数更新 j 方向矢量值
out_q3[31:0]	输出	clk	本次四元数更新 k 方向矢量值
控制与指示接口 (CTRL、IND)			
in_quat_en	输入	clk	电路使能信号 (高电平有效), 当 3 轴陀螺仪修正数据和四元数 4 分量数据全部读取完毕, 该信号被置为高电平
out_quat_finish	输出	clk	四元数更新完毕标志信号 (高电平有效)

4.4.2 四元数更新电路框图

四元数更新电路由控制电路工作时序的有限状态机、浮点数加/减法器、浮点数乘法器和其他电路互联的接口。为了描述简便，根据第 2 章中算法推导，该电路可分为一阶龙格-库塔解算电路 (QUAT_LK) 和四元数归一化电路 (QUAT_NOR)，由于电路设计过程中使用了大量逻辑优化，这里不再给出四元数更新电路底层 IP 核间的电气连接和状态机控制电路的电气连接原理图。四元数更新电路的框图和状态迁移图分别如图 4-11 和图 4-12 所示。

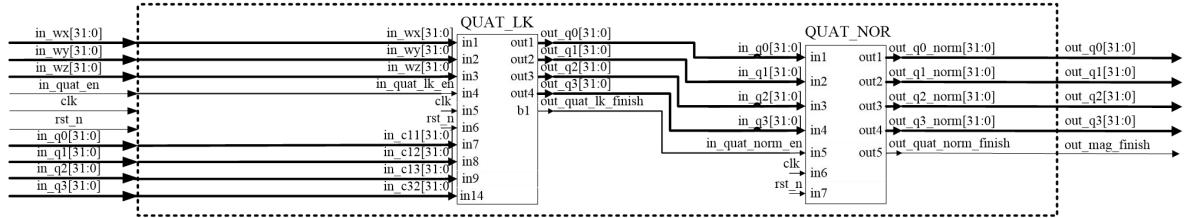


图 4-11 四元数更新电路的框图

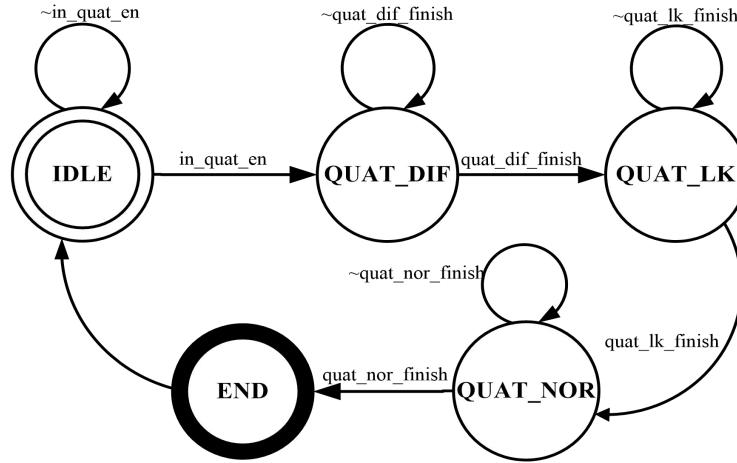


图 4-12 四元数更新电路的状态迁移图

四元数更新电路采用 5 级流水线结构进行数据处理，内部共有 126 个状态，包括高电平信号检测状态和标志位采样状态。为方便描述，合并为 5 个状态，如图 4-12 所示，分别为 IDLE（初始状态）、四元数微分状态（QUAT_DIF）、一阶龙格-库塔解算状态（QUAT_LK）、四元数归一化状态（QUAT_NOR）和结束状态状态（END）。

在系统复位时，状态置为到初始状态将电路中所有接口对应的寄存器都置为空，四元数初始化为 $\mathbf{Q} = [q_0 \ q_1 \ q_2 \ q_3]^T = [1 \ 0 \ 0 \ 0]^T$ ，所有辅助信号都置为低电平。当控制信号 in_quat_en 被顶层电路的状态机置为高电平后，进行上次更新后四元数数据的读取操作，状态跳转至 QUAT_LK，一阶龙格-库塔解算电路开始工作；当一阶龙格-库塔解算完毕，输出辅助信号 $quat_lk_finish$ 被置为高电平后，状态跳转至 QUAT_NOR，四元数归一化电路开始工作；当四元数归一化完毕，输出辅助信号 $quat_nor_finish$ 被置为高电平后，状态跳转至 END；在 END 状态下将本次更新的四元数输出，并跳转至初始状态 IDLE。至此，一次四元数更新过程结束。

4.5 姿态转换矩阵更新电路的设计

姿态转换矩阵更新电路使用四元数更新值对姿态转换矩阵进行更新，图 4-13 给出了姿态转换矩阵更新电路设计接口框图，表 4-5 则给出了接口中各个输入和输出信号的具体描述。

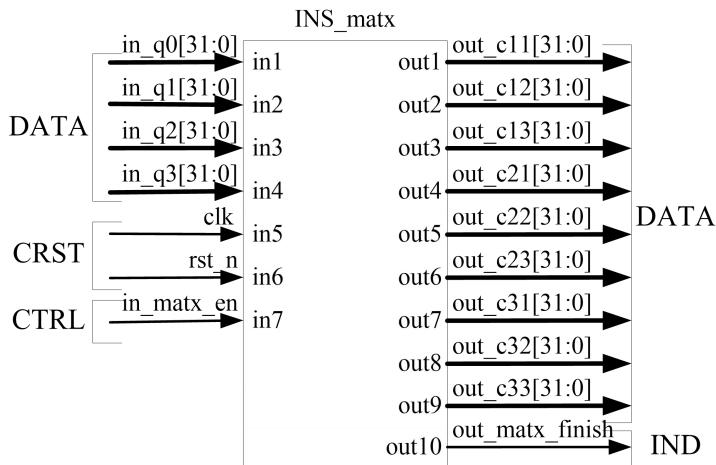


图 4-13 姿态转换矩阵更新电路的接口框图

表 4-5 姿态转换矩阵更新电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步复位)
数据接口 (DATA)			
in_q0[31:0]	输入	clk	前一次四元数更新标量值
in_q1[31:0]	输入	clk	前一次四元数更新 i 方向矢量值
in_q2[31:0]	输入	clk	前一次四元数更新 j 方向矢量值
in_q3[31:0]	输入	clk	前一次四元数更新 k 方向矢量值
out_c11[31:0]	输出	clk	姿态矩阵更新后第 1 行第 1 列元素值
out_c12[31:0]	输出	clk	姿态矩阵更新后第 1 行第 2 列元素值
out_c13[31:0]	输出	clk	姿态矩阵更新后第 1 行第 3 列元素值
out_c21[31:0]	输出	clk	姿态矩阵更新后第 2 行第 1 列元素值
out_c22[31:0]	输出	clk	姿态矩阵更新后第 2 行第 2 列元素值
out_c23[31:0]	输出	clk	姿态矩阵更新后第 2 行第 3 列元素值
out_c31[31:0]	输出	clk	姿态矩阵更新后第 3 行第 1 列元素值
out_c32[31:0]	输出	clk	姿态矩阵更新后第 3 行第 2 列元素值
out_c33[31:0]	输出	clk	姿态矩阵更新后第 3 行第 3 列元素值
控制与指示接口 (CTRL、IND)			
in_matx_en	输入	clk	电路使能信号 (高电平有效), 当四元数分量数据全部读取完毕, 该信号被置为高电平
out_matx_finish	输出	clk	姿态矩阵更新完毕标志信号 (高电平有效)

姿态转换矩阵更新电路由控制电路工作时序的有限状态机、浮点数加/减法器、浮点数乘法器和其他电路互联的接口。姿态矩阵更新电路采用 5 级流水线结构进行数据处理, 内部共有 78 个状态, 包括高低电平信号检测状态和标志位采样状态。为了描述简便, 根据第 2 章中算法推导, 该电路没有复杂的算法电路, 因此姿态转换矩阵更新电路的框图和状态转移图在此省略。

在系统复位时, 状态置为到初始状态将电路中所有接口对应的寄存器都置为空, 姿态矩阵初始化为 $C_n^b = \text{diag}[1 \ 1 \ 1]$, 所有辅助信号都置为低电平。当控制信号 `in_matx_en` 被顶层电路的状态机置为高电平后, 进行更新后四元数数据的读取操作, 并根据第 2 章中姿态矩阵九元素更新公式由状态机控制底层浮点数运算 IP 核进行多流水线运算及 IP 核复用。当 `out_matx_finish` 被置为高电平, 更新后的姿态矩阵九元素输出, 至此, 一次

姿态转换矩阵更新过程结束。

4.6 姿态角解算电路的设计

4.6.1 姿态角解算电路的接口设计

姿态角解算电路使用更新后的姿态转换矩阵对姿态角进行解算处理，图 4-14 给出了姿态角解算电路的设计接口框图，表 4-6 则给出了接口中各个输入和输出信号的具体描述。

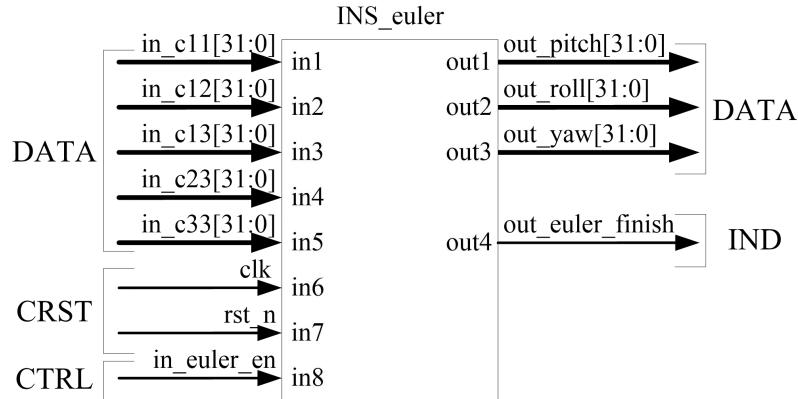


图 4-14 姿态角解算电路的接口框图

表 4-6 姿态角解算电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步复位)
数据接口 (DATA)			
in_c11[31:0]	输入	clk	姿态矩阵更新后第 1 行第 1 列元素值
in_c12[31:0]	输入	clk	姿态矩阵更新后第 1 行第 2 列元素值
in_c13[31:0]	输入	clk	姿态矩阵更新后第 1 行第 3 列元素值
in_c23[31:0]	输入	clk	姿态矩阵更新后第 2 行第 3 列元素值
in_c33[31:0]	输入	clk	姿态矩阵更新后第 3 行第 3 列元素值 x
out_pitch[31:0]	输出	clk	俯仰角解算值
out_roll[31:0]	输出	clk	横滚角解算值
out_yaw[31:0]	输出	clk	航向角解算值
控制与指示接口 (CTRL、IND)			
in_euler_en	输入	clk	电路使能信号 (高电平有效), 当姿态矩阵数据全部读取完毕, 该信号被置为高电平
out_euler_finish	输出	clk	姿态解算完毕标志信号 (高电平有效)

4.6.2 姿态角解算电路框图

姿态角解算电路由控制电路工作时序的有限状态机、浮点数加/减法器、浮点数乘法器和其他电路互联的接口。为了描述简便，根据第 2 章中算法推导，该电路可分为俯仰角解算电路 (EULER_P)、横滚角解算电路 (EULER_R) 和航向角解算电路 (EULER_Y)，由于电路设计过程中使用了大量逻辑优化，这里不再给出姿态角解算电路底层 IP 核间的电气连接和状态机控制电路的电气连接原理图。四元数更新电路的框图和状态迁移图分别如图 4-15 和图 4-16 所示。

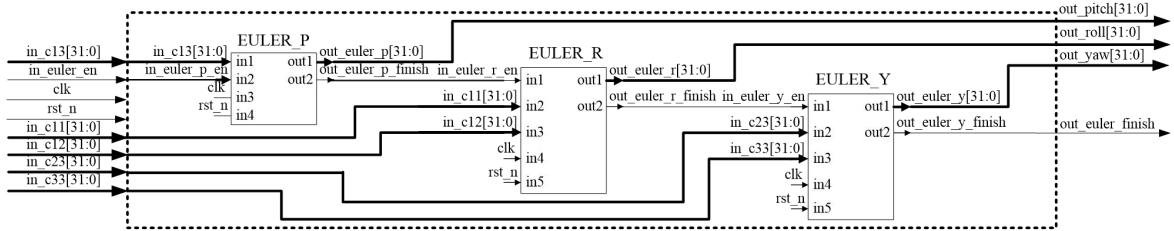


图 4-15 姿态角解算电路的框图

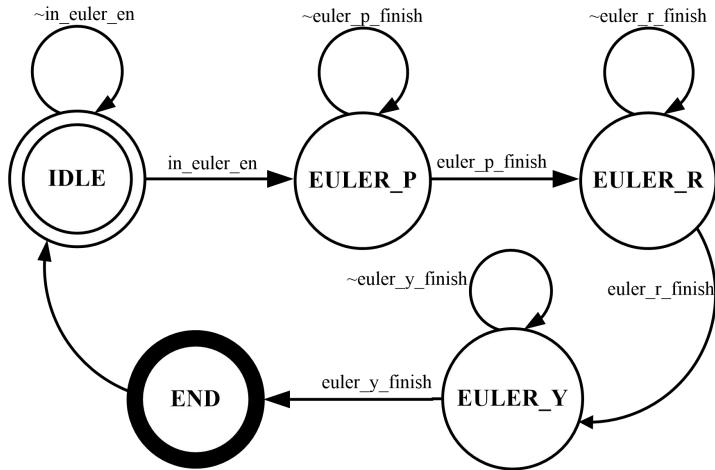


图 4-16 姿态角解算电路的状态迁移图

姿态矩阵更新电路采用 3 级流水线结构进行数据处理，内部共有 98 个状态，包括高低电平信号检测状态和标志位采样状态。为方便描述，合并为 6 个状态，如图 4-16 所示，分别为 IDLE（初始状态）、俯仰角解算状态（EULER_P）、横滚角解算状态（EULER_R）、航向角解算状态（EULER_Y）和结束状态状态（END）。

在系统复位时，状态置为到初始状态将电路中所有接口对应的寄存器都置为空，所有辅助信号都置为低电平。当控制信号 `in_euler_en` 被顶层电路的状态机置为高电平后，进行上次更新后姿态转换矩阵的读取操作，状态跳转至 `EULER_P`，俯仰角解算电路开始工作；当俯仰角解算完毕，输出辅助信号 `euler_p_finish` 被置为高电平后，状态跳转至 `EULER_R`，横滚角解算电路开始工作；当横滚角解算电路完毕，输出辅助信号 `euler_r_finish` 被置为高电平后，状态跳转至 `EULER_Y`，航向角解算电路开始工作；当航向角解算完毕，输出辅助信号 `euler_y_finish` 被置为高电平后，状态跳转至 `END`；在 `END` 状态下将本次解算的姿态角输出，并跳转至初始状态 `IDLE`。至此，一次姿态角解算过程结束。

4.6.3 反正切函数电路的设计

(1) 反正切函数电路的接口设计

反正切函数电路进行通过反正切操作求得弧度值，图 4-17 给出了反正切函数电路的设计接口框图，表 4-7 则给出了接口中各个输入和输出信号的具体描述。

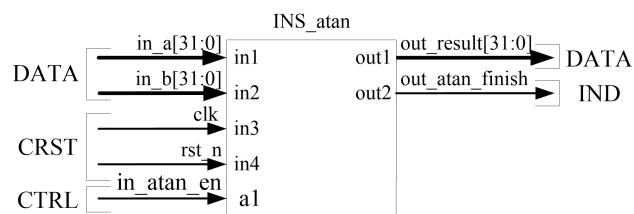


图 4-17 反正切函数电路的接口框图

表 4-7 反正切函数电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步复位)
数据接口 (DATA)			
in_a[31:0]	输入	clk	余弦值
in_b[31:0]	输入	clk	正弦值
out_result[31:0]	输出	clk	弧度值
控制与指示接口 (CTRL、IND)			
in_atan_en	输入	clk	电路使能信号 (高电平有效), 当数据全部读取完毕, 该信号被置为高电平
out_atan_finish	输出	clk	反正切运算完毕标志信号 (高电平有效)

(2) 反正切函数电路框图

反正切函数电路由控制电路工作时序的有限状态机、浮点数加/减法器、浮点数乘法器和其他电路互联的接口。为了描述简便, 根据第 3 章中 IP 核的设计, 该电路可分为浮点数转定点数电路 (float32tofix16)、反正切运算电路 (atan) 和定点数转浮点数电路 (fix16tofloat32), 由于电路设计过程中使用了大量逻辑优化, 这里不再给出姿态角解算电路底层 IP 核间的电气连接和状态机控制电路的电气连接原理图。反正切函数电路的框图和状态迁移图分别如图 4-18 和图 4-19 所示。

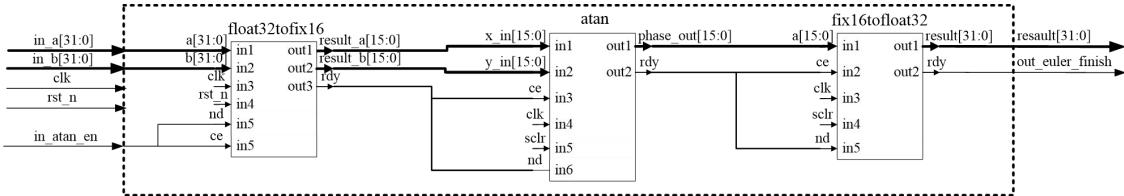


图 4-18 反正切函数电路的框图

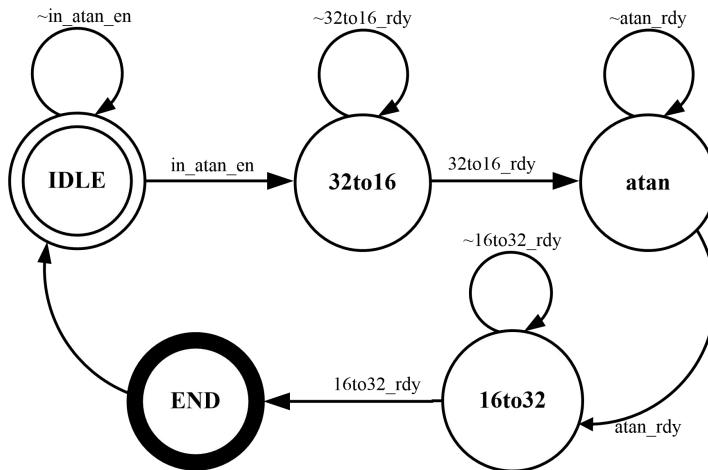


图 4-19 反正切函数电路的状态迁移图

反正切函数电路中共有 5 个状态, 如图 4-19 所示, 分别为 IDLE (初始状态)、浮点数转定点数状态 (32to16)、反正切运算状态 (atan)、定点数转浮点数状态 (16to32) 和结束状态状态 (END)。

在系统复位时, 状态置为到初始状态将电路中所有接口对应的寄存器都置为空, , 所有辅助信号都置为低电平。当控制信号 in_atan_en 被顶层电路的状态机置为高电平后,

进行上次更新后姿态转换矩阵的读取操作，状态跳转至 32to16，浮点数转定点数电路开始工作；当浮点数转定点数完毕，输出辅助信号 32to16_rdy 被置为高电平后，状态跳转至 atan，反正切运算电路开始工作；当反正切运算电路完毕，输出辅助信号 atan_rdy 被置为高电平后，状态跳转至 16to32，定点数转浮点数电路开始工作；当定点数转浮点数完毕，输出辅助信号 16to32_rdy 被置为高电平后，状态跳转至 END；在 END 状态下将本次计算的弧度值输出，并跳转至初始状态 IDLE。至此，一次反正切运算过程结束。

4.7 顶层电路的设计

4.7.1 顶层电路的接口设计

顶层电路的接口主要为与其他电路进行互联的接口，图 4-20 给出了顶层电路的设计接口框图，表 4-8 则给出了接口中各个输入和输出信号的具体描述。

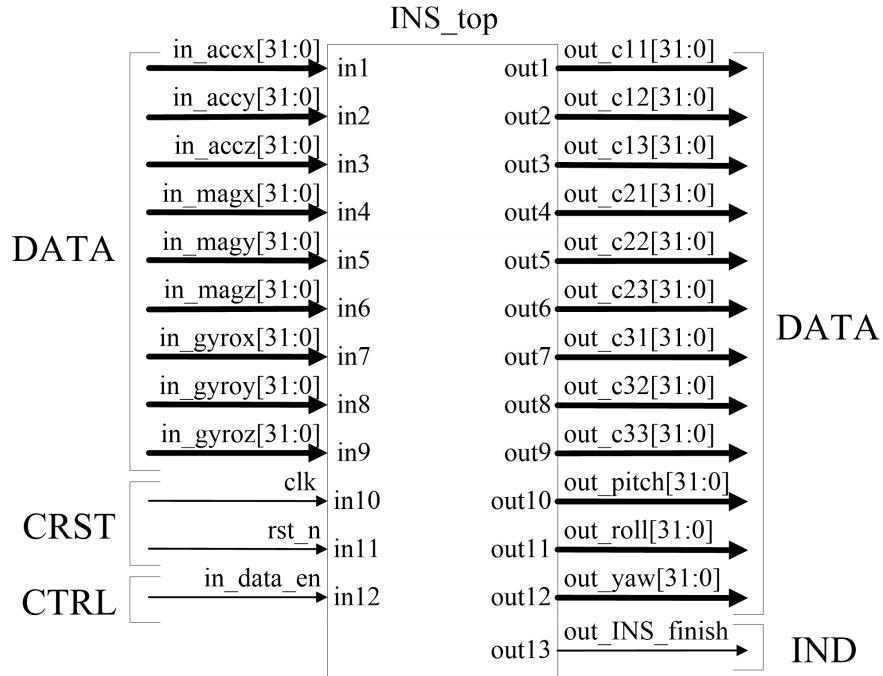


图 4-20 顶层电路的接口框图

表 4-8 顶层电路的接口信号

接口名称	输入/输出	时钟域	描述
时钟复位接口 (CRST)			
clk	输入	N/A	时钟信号 (典型值 100MHz)
rst_n	输入	N/A	复位信号 (低电平有效, 异步有效).
数据接口 (DATA)			
in_accx[31:0]	输入	clk	加速度计 x 轴测量值
in_accy[31:0]	输入	clk	加速度计 y 轴测量值
in_accz[31:0]	输入	clk	加速度计 z 轴测量值
in_magx[31:0]	输入	clk	电子罗盘 x 轴测量值
in_magy[31:0]	输入	clk	电子罗盘 y 轴测量值
in_magz[31:0]	输入	clk	电子罗盘 z 轴测量值
in_gyrox[31:0]	输入	clk	陀螺仪 x 轴测量值
in_gyroy[31:0]	输入	clk	陀螺仪 y 轴测量值
in_gyroz[31:0]	输入	clk	陀螺仪 z 轴测量值
out_c11[31:0]	输出	clk	姿态矩阵更新后第 1 行第 1 列元素值
out_c12[31:0]	输出	clk	姿态矩阵更新后第 1 行第 2 列元素值

续表 4-8

数据接口 (DATA)

out_c13[31:0]	输出	clk	姿态矩阵更新后第 1 行第 3 列元素值
out_c21[31:0]	输出	clk	姿态矩阵更新后第 2 行第 1 列元素值
out_c22[31:0]	输出	clk	姿态矩阵更新后第 2 行第 2 列元素值
out_c23[31:0]	输出	clk	姿态矩阵更新后第 2 行第 3 列元素值
out_c31[31:0]	输出	clk	姿态矩阵更新后第 3 行第 1 列元素值
out_c32[31:0]	输出	clk	姿态矩阵更新后第 3 行第 2 列元素值
out_c33[31:0]	输出	clk	姿态矩阵更新后第 3 行第 3 列元素值
out_pitch[31:0]	输出	clk	俯仰角解算值
out_roll[31:0]	输出	clk	横滚角解算值
out_yaw[31:0]	输出	clk	航向角解算值

控制与指示接口 (CTRL、IND)

in_data_en	输入	clk	电路使能信号 (高电平有效), 当 9 轴数据全部读取完毕, 该信号被置为高电平
out_INS_finish	输出	clk	姿态解算完毕标志信号 (高电平有效)

4.7.2 顶层电路框图

顶层电路中主要包括控制整个电路工作 时序的有限状态机和与其他电路互联的接口。其框图和状态迁移图分别如图 4-21 和图 4-22 所示。

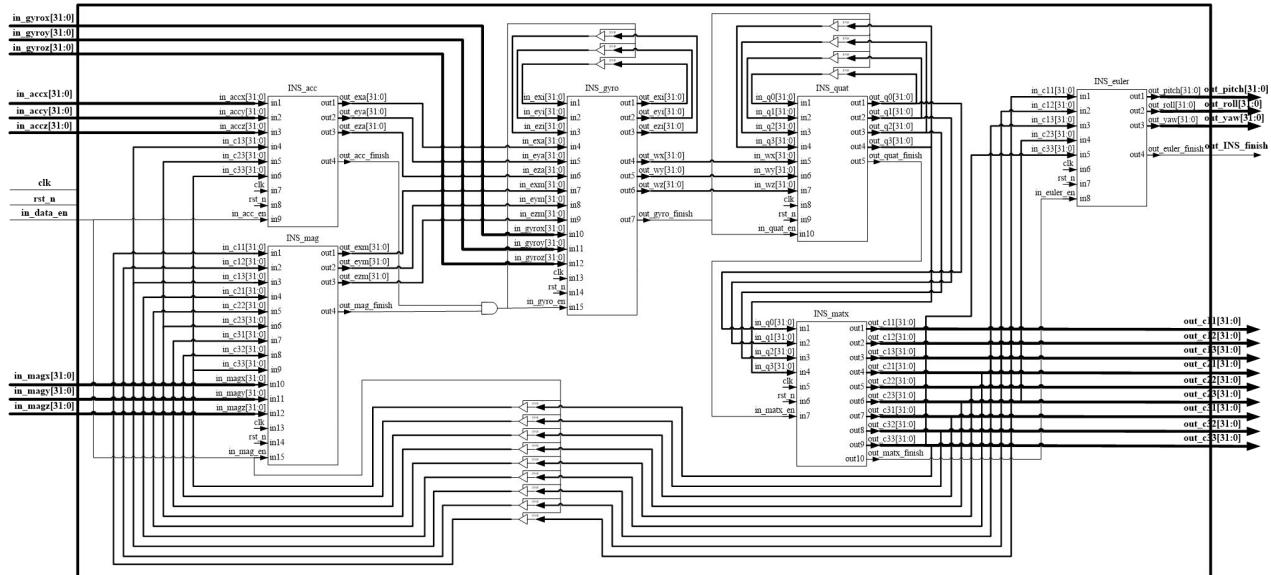


图 4-21 顶层电路的框图

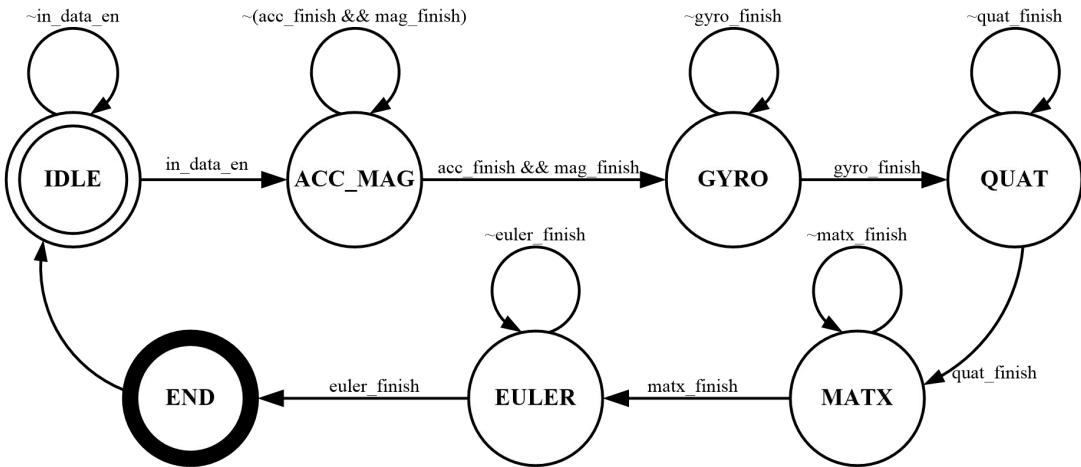


图 4-22 顶层电路的状态迁移图

顶层电路主要由一个对子模块的辅助信号进行判断并发送控制信号的有限状态机组成。通过状态迁移过程来控制三轴加速度计测量误差解算电路（INS_acc）、三轴电子罗盘测量误差解算电路（INS_mag）、三轴陀螺仪测量误差修正电路（INS_gyro）、四元数更新电路（INS_quat）、姿态转换矩阵更新电路（INS_matx）和三轴姿态角解算电路（INS_euler）的工作顺序。

顶层电路中共有 6 个子状态，如图 4-22 所示，分别为 IDLE（初始状态）、加速度计和电子罗盘测量误差模块工作状态（ACC_MAG）、陀螺仪测量误差修正模块工作状态（GYRO）、四元数更新模块工作状态（QUAT）、姿态转换矩阵更新模块工作状态（MATX）、姿态角解算模块工作状态（EULER）和结束状态（END）。

在系统复位状态下，姿态转换矩阵置为 $C_n^b = \text{diag}[1 \ 1 \ 1]$ ，四元数置位为

$Q = [q_0 \ q_1 \ q_2 \ q_3]^T = [1 \ 0 \ 0 \ 0]^T$ ，状态置为到初始状态将电路中所有接口对应的寄存器都置为空，所有辅助信号都置为低电平。当控制信号 in_data_en 被 CONTROLLER 置为高电平后，进行九轴传感器测量数据的读取操作，状态跳转至 ACC_MAG，将加速度计测量误差计算电路和电子罗盘测量误差计算电路的辅助控制信号置为高电平使两电路开始工作；当加速度计测量误差计算电路和电子罗盘测量误差计算电路都将数据处理完毕，输出辅助信号 acc_finish 和 mag_finish 都被置为高电平后，状态跳转至 GYRO，将陀螺仪测量误差修正电路的辅助控制信号置为高电平使电路开始工作；当陀螺仪测量误差修正电路将数据处理完毕，输出辅助信号 gyro_finish 被置为高电平后，状态跳转至 QUAT，将四元数更新电路的辅助控制信号置为高电平使电路开始工作；当四元数更新电路将数据处理完毕，输出辅助信号 quat_finish 被置为高电平后，状态跳转至 MATX，将姿态转换矩阵更新电路的辅助控制信号置为高电平使电路开始工作；当姿态转换矩阵更新电路将数据处理完毕，输出辅助信号 matx_finish 被置为高电平后，状态跳转至 EULER，将姿态角解算电路的辅助控制信号置为高电平使电路开始工作；当姿态角解算电路将数据处理完毕，输出辅助信号 euler_finish 被置为高电平后，状态跳转至 END；在 END 状态下将解算得到的姿态角和姿态转换矩阵九个元素输出，并跳转至初始状态 IDLE。至此，一次姿态解算过程结束。

顶层电路的 RTL 实现详见附录 B。

5 导航解算电路的综合与仿真验证

本章对第4章中设计的电路进行了综合和时序仿真验证。综合后，首先对各个子模块电路进行了验证，然后对顶层电路进行了验证，在验证的同时分别给出了验证结果和使用Matlab软件仿真结果的误差，最后给出了误差分析。

为了方便描述，在使用Modelsim进行验证时，使用虚拟的测试用例（TESTCASE）对设计的电路进行功能验证和时序验证。

5.1 基于ISE的综合

5.1.1 系统工程的建立

完成整个系统的综合，首先需要使用ISE开发环境建立系统工程。首先建立新的系统工程目录“INS_prj”，选择芯片类型为ZedBoard开发板载芯片Zynq-xc7z020-1clg400，如图5-1所示。

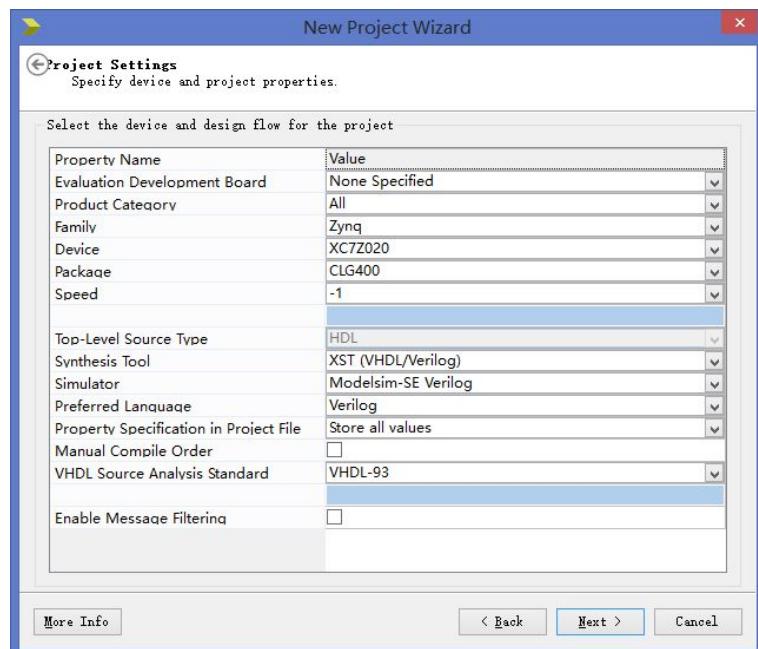


图5-1 工程设置

工程建立完毕，向工程内添加在第4章中已写好的RTL文件，并将“INS_top”设置为顶层文件，如图5-2所示。

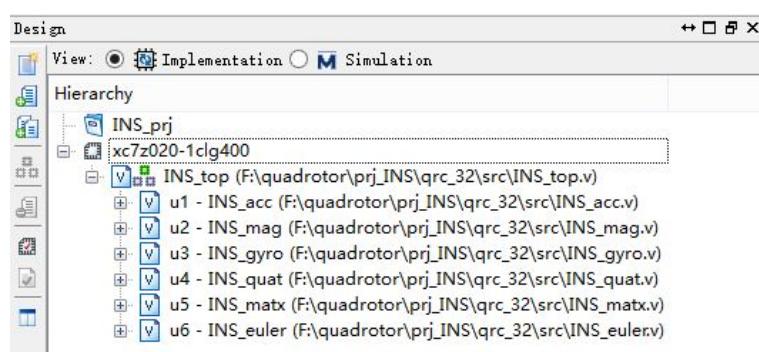


图5-2 工程文件目录

5.1.2 综合

在“Design”对话框内选择“Synthesize”或者“Implement Design”对工程文件进行综合，如图 5-3 所示。

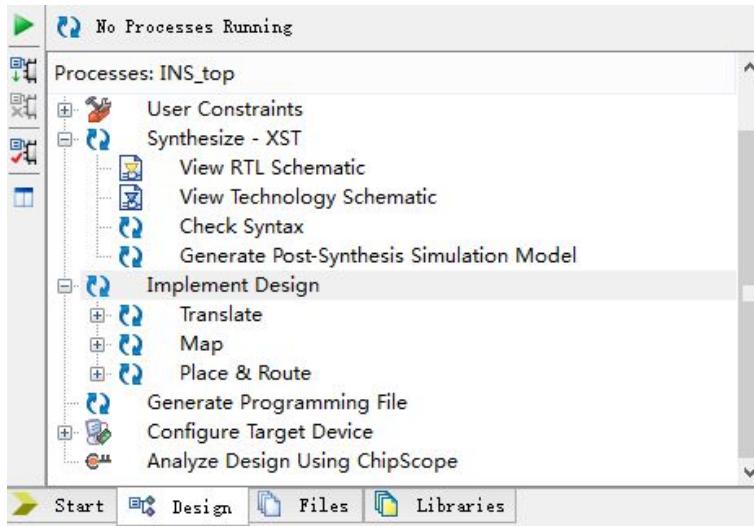


图 5-3 综合选项

5.1.3 门数及面积

综合后的报告如图 5-4 所示。

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	20172	106400	18%
Number of Slice LUTs	15111	53200	28%
Number of fully used LUT-FF pairs	9879	25404	38%
Number of bonded IOBs	676	125	540%
Number of BUFG/BUFGCTRLs	2	32	6%
Number of DSP48E1s	40	220	18%

图 5-4 综合报告

由图 5-4 可看出，整个工程综合后，寄存器占总数量 18%，查找表占用 28%，触发器占用 38%，全局时钟缓冲器占用 6%，DSP48E 占用 18%。由于本设计是用在芯片中的内部电路，因此端口超出了实际芯片的端口数量。除端口外，其他资源占用很少，因此本设计在芯片中的工作是安全的。

5.2 导航解算电路的仿真验证

综合完毕后，建立顶层电路和子模块电路的测试文件 TESTBENCH，如图 5-5 所示。

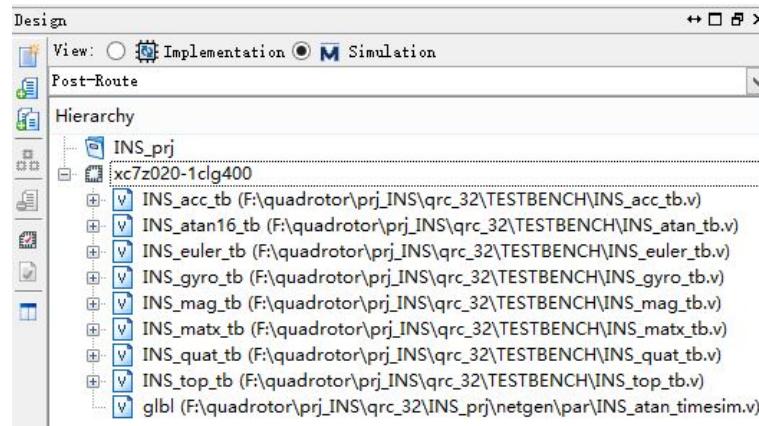


图 5-5 测试文件

测试文件建立完成后，在“Design”对话框内选择“Simulate Post-Place&Route Model”调用 Modelsim 软件对电路进行仿真，如图 5-6 所示。

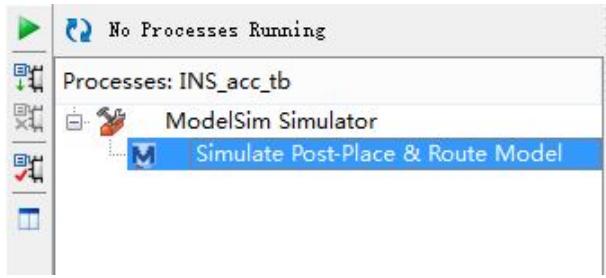


图 5-6 测试文件

5.2.1 加速度计测量误差计算电路的仿真与验证

对加速度计测量误差计算电路进行仿真验证，图 5-7 中的信号由上向下依次为时钟信号、低电平复位信号、使能信号 acc_en，电路输入信号分别为 accx=1、accy=2、accz=3、c13=3、c23=2、c33=1，同时 acc_en 置为高电平。经过 5.6us 运算之后，输出信号 exa=-1.06904、eya=2.13809、eza=-1.06904，并将 acc_finish 置为高电平。

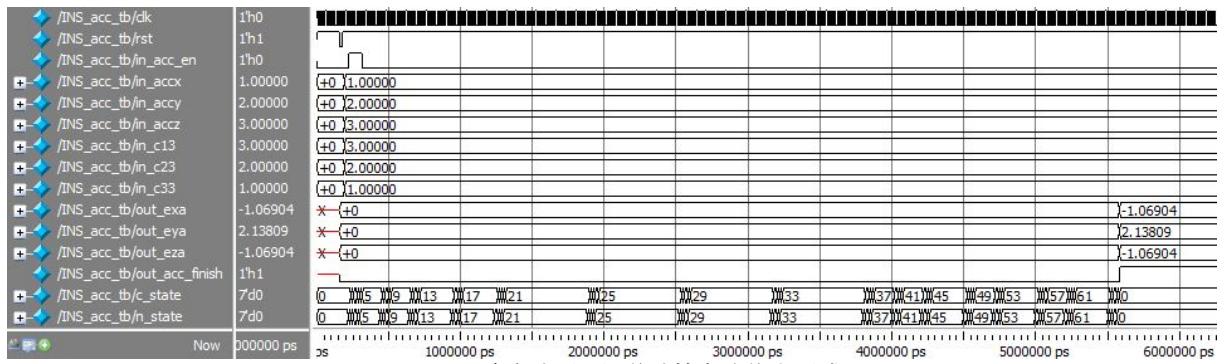


图 5-7 加速度计测量误差计算电路仿真测试

仿真输出结果与在 Matlab 中仿真结果对比如表 5-1 所示。结果显示加速度计测量误差计算电路计算结果与使用 Matlab 软件计算结果相同，误差为 0。

表 5-1 加速度计测量误差计算电路仿真对比

输出 \ 环境	Modelsim	Matlab	误差
exa	-1.06904	-1.06904	0.0
eya	2.13809	2.13809	0.0
eza	-1.06904	-1.06904	0.0

5.2.2 电子罗盘测量误差计算电路的仿真与验证

对电子罗盘测量误差计算电路进行仿真验证，图 5-8 中的信号由上向下依次为时钟信号、低电平复位信号、使能信号 mag_en，电路输入信号分别为 magx=2、magy=2、magz=2、c11=1、c12=1、c13=1、c21=1、c22=1、c23=1、c31=1、c32=1、c33=1，同时 mag_en 置为高电平。经过 11.5us 运算之后，输出信号 exm=-8.04738、eym=-3.21895、ezm=4.82842，并将 mag_finish 置为高电平。

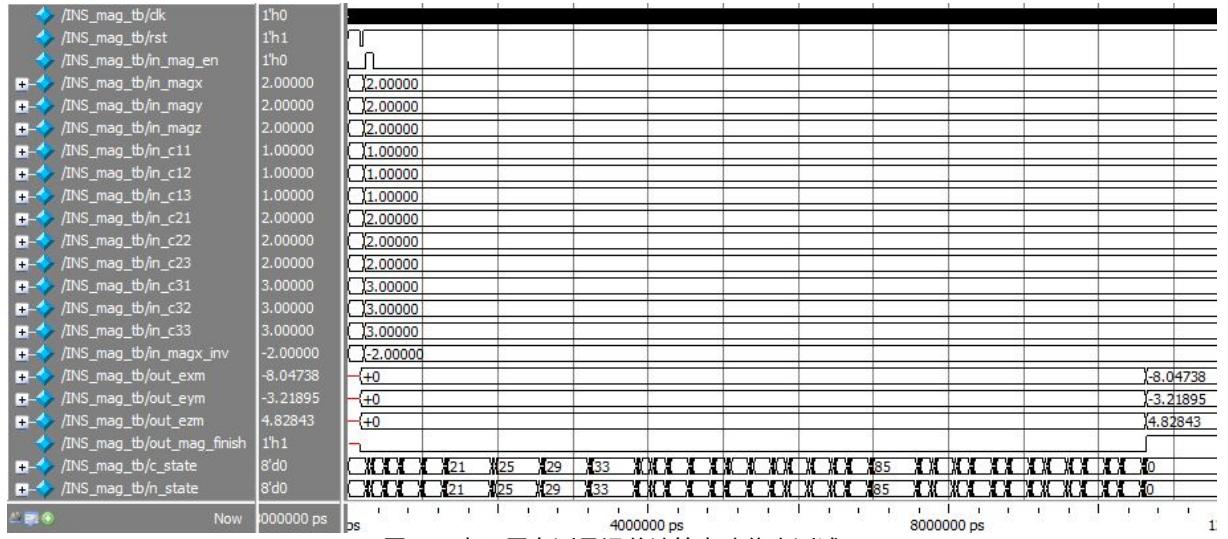


图 5-8 电子罗盘测量误差计算电路仿真测试

仿真输出结果与在 Matlab 中仿真结果对比如表 5-2 所示。结果显示电子罗盘测量误差计算电路计算结果与使用 Matlab 软件计算结果相同，误差为 0。

表 5-2 电子罗盘测量误差计算电路仿真对比

输出	环境	Modelsim	Matlab	误差
exm		-8.04738	-8.04738	0.0
eym		-3.21895	-3.21895	0.0
ezm		4.82843	4.82843	0.0

5.2.3 陀螺仪测量误差修正电路的仿真与验证

对陀螺仪测量误差修正电路进行仿真验证，图 5-9 中的信号由上向下依次为时钟信号、低电平复位信号、使能信号 gyro_en，电路输入信号分别为 exi=1、eyi=2、ezi=3、exa=1、eya=2、eza=3、exm=1、eym=2、ezm=3、gyrox=1、gyroy=2、gyroz=3，同时 gyro_en 置为高电平。经过 5.8us 运算之后，输出信号 exi=1.0020、eyi=2.0040、ezi=3.0060、wx=-21.0195、wy=42.0389、wz=63.0584，并将 gyro_finish 置为高电平。

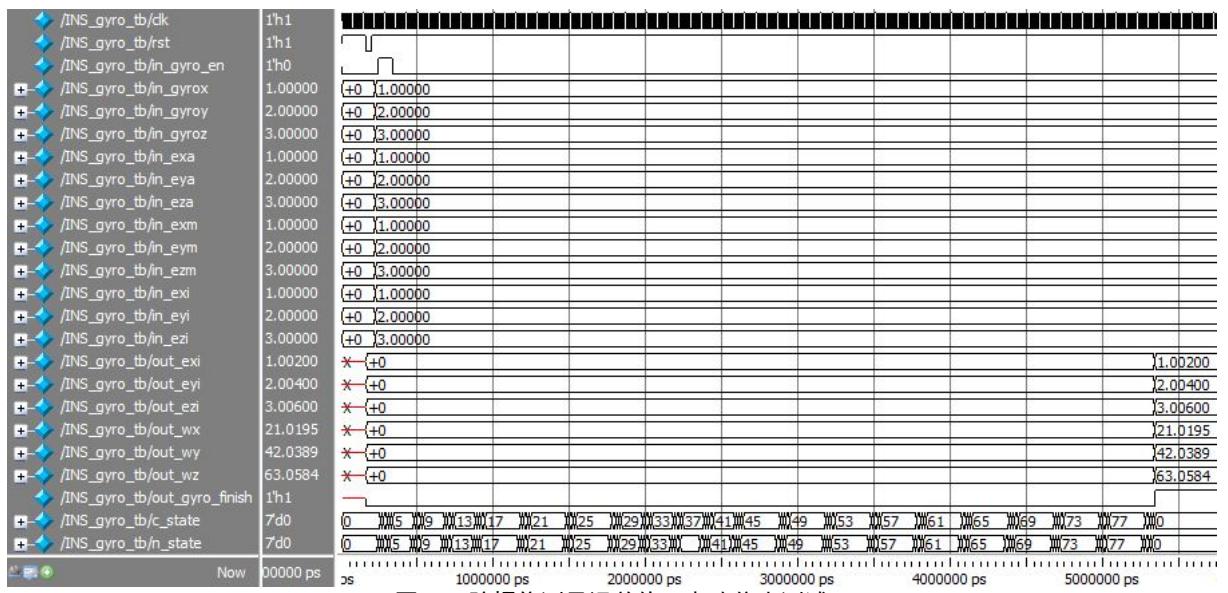


图 5-9 陀螺仪测量误差修正电路仿真测试

仿真输出结果与在 Matlab 中仿真结果对比如表 5-3 所示。结果显示陀螺仪测量误差

修正电路计算结果与使用 Matlab 软件计算结果相同，误差为 0。

表 5-3 陀螺仪测量误差修正电路仿真对比

输出	环境	Modelsim	Matlab	误差
exi		1.0020	1.0020	0.0
eyi		2.0040	2.0040	0.0
ezi		3.0060	3.0060	0.0
wx		21.0195	21.0195	0.0
wy		42.0389	42.0389	0.0
wz		63.0584	63.0584	0.0

5.2.4 四元数更新电路的仿真与验证

对四元数更新电路进行仿真验证，图 5-10 中的信号由上向下依次为时钟信号、低电平复位信号、使能信号 quat_en，电路输入信号分别为 $q_0=1$ 、 $q_1=2$ 、 $q_2=3$ 、 $q_3=4$ 、 $wx=1$ 、 $wy=2$ 、 $wz=3$ ，同时 quat_en 置为高电平。经过 10us 运算之后，输出信号 $q_0=0.178981$ 、 $q_1=0.365085$ 、 $q_2=0.547902$ 、 $q_3=0.731083$ ，并将 quat_finish 置为高电平。

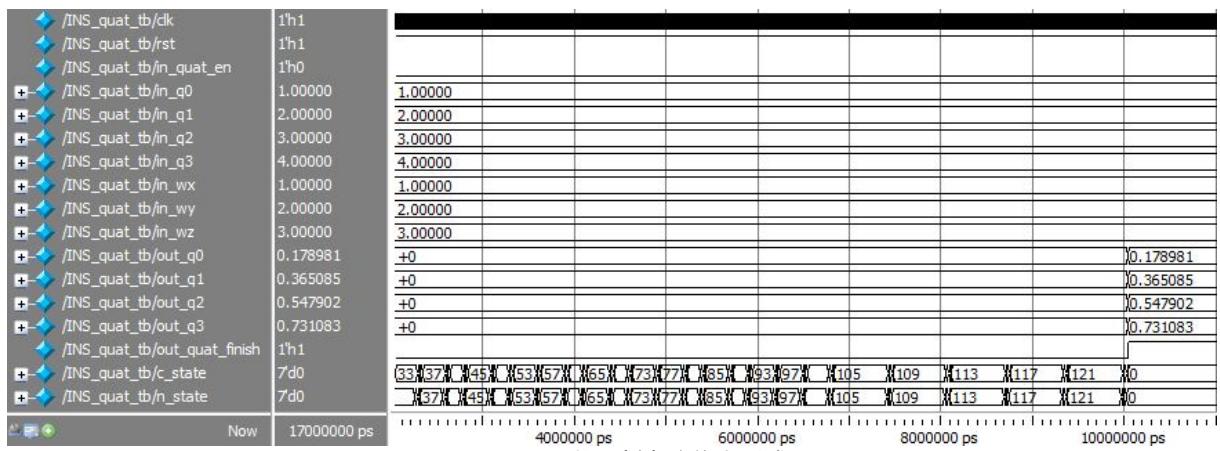


图 5-10 四元数更新电路仿真测试

仿真输出结果与在 Matlab 中仿真结果对比如表 5-4 所示。结果显示四元数更新电路计算结果与使用 Matlab 软件计算结果基本相同，误差低于 0.05%。

表 5-4 四元数更新电路仿真对比

输出	环境	Modelsim	Matlab	误差
q_0		0.178981	0.178921	0.00006
q_1		0.365085	0.365511	-0.000426
q_2		0.547902	0.547719	0.000183
q_3		0.731083	0.731022	0.000061

5.2.5 姿态矩阵更新电路的仿真与验证

对姿态矩阵更新电路仿真验证时，使用的 TESTCASE： q_0 、 q_1 、 q_2 、 q_3 分别为 1、2、3、4，四元数更新电路的仿真波形如。图 5-11 中的信号由上向下依次为时钟信号、低电平复位信号、使能信号 matx_en，电路输入信号分别为 $q_0=1$ 、 $q_1=2$ 、 $q_2=3$ 、 $q_3=4$ ，同时 matx_en 置为高电平。经过 4.6us 运算之后，输出信号 $c_{11}=-49$ 、 $c_{12}=20$ 、 $c_{13}=10$ 、 $c_{21}=4$ 、 $c_{22}=-39$ 、 $c_{23}=28$ 、 $c_{31}=22$ 、 $c_{32}=20$ 、 $c_{33}=-25$ ，并将 matx_finish 置为高电平。

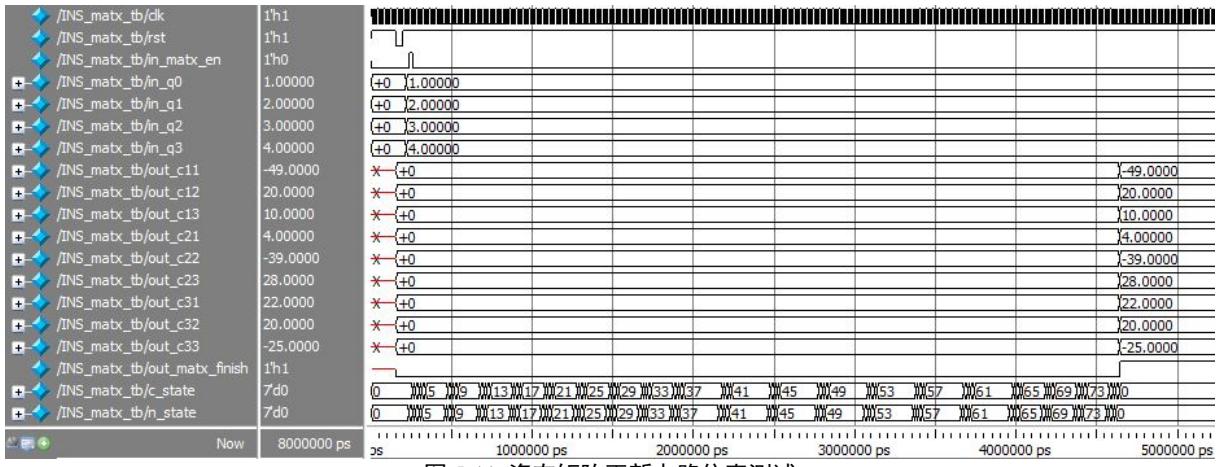


图 5-11 姿态矩阵更新电路仿真测试

仿真输出结果与在 Matlab 中仿真结果对比如表 5-5 所示。结果显示姿态矩阵更新电路计算结果与使用 Matlab 软件计算结果相同，误差为 0。

表 5-5 姿态矩阵更新电路仿真对比

输出	环境	Modelsim	Matlab	误差
c11		-49	-49	0.0
c12		20	20	0.0
c13		10	10	0.0
c21		4	4	0.0
c22		-39	-39	0.0
c23		28	28	0.0
c31		22	22	0.0
c32		20	20	0.0
c33		-25	-25	0.0

5.2.6 姿态角解算电路的仿真与验证

对姿态角解算电路仿真验证时，图 5-12 中的信号由上向下依次为时钟信号、低电平复位信号、使能信号 euler_en，电路输入信号分别为 c11=2、c12=-1、c13=-0.5、c23=-2、c33=1，同时 euler_en 置为高电平。经过 9us 运算之后，输出信号 pitch=30.0047、roll=-63.4296、yaw=333.4360，并将 euler_finish 置为高电平。

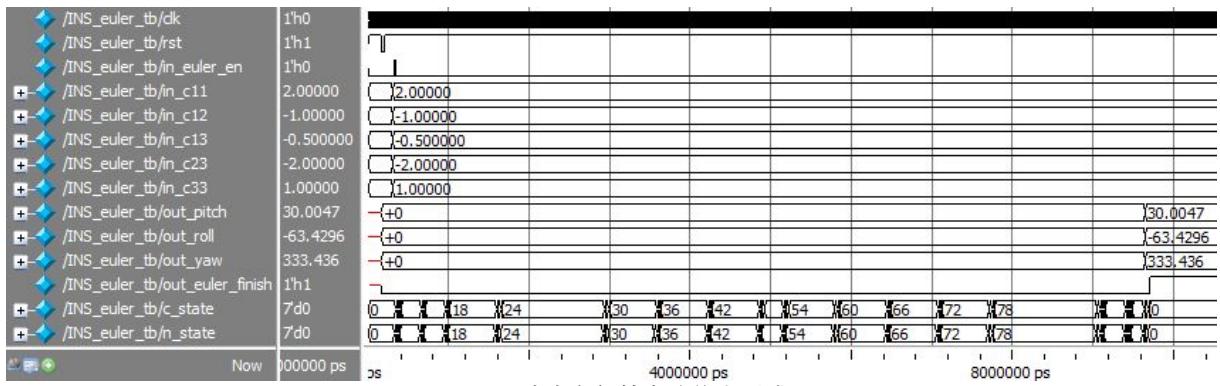


图 5-12 姿态角解算电路仿真测试

仿真输出结果与在 Matlab 中仿真结果对比如表 5-6 所示。结果显示姿态角解算电路计算结果与使用 Matlab 软件计算结果基本相同，误差低于 0.01%。

表 5-6 姿态角解算电路仿真对比

输出	环境	Modelsim	Matlab	误差
	pitch	30.0047	30.0000	0.0047
	roll	-63.4296	-63.4349	0.0053
	yaw	333.4360	333.4349	0.0011

5.2.7 顶层电路的仿真与验证

对顶层电路进行仿真验证，中的信号由上向下依次为时钟信号、低电平复位信号、使能信号 data_en，电路输入信号分别为 accx=1、accy=1、accz=1、magx=1、magy=1、magz=1、gyrox=1、gyroy=1、gyroz=1，同时 data_en 置为高电平。经过 40us 运算之后，输出当前解算的三个姿态角角度，并将 INS_finish 置为高电平。仿真过程进行五次解算。

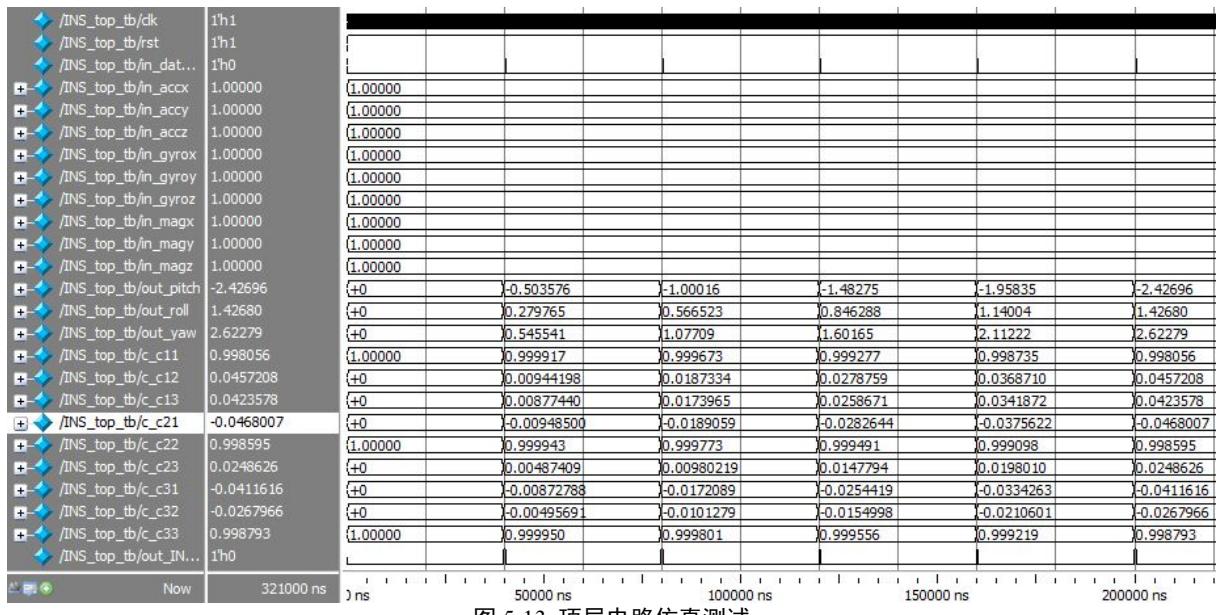


图 5-13 顶层电路仿真测试

仿真输出结果与在 Matlab 中仿真结果对比如表 5-7 所示。

表 5-7 顶层电路仿真对比

输出	环境	Modelsim	Matlab	误差
第 1 次解算				
pitch		-0.5036	-0.5027	-0.0009
roll		0.2798	0.2793	0.0005
yaw		0.5455	0.5410	0.0045
第 2 次解算				
pitch		-1.0000	-0.9968	-0.0032
roll		0.5665	0.5572	0.0093
yaw		1.0771	1.0762	0.0009
第 3 次解算				
pitch		-1.4827	-1.4821	-0.0006
roll		0.8463	0.8338	0.0125
yaw		1.6017	1.6058	-0.0041
第 4 次解算				
pitch		-1.9584	-1.9589	0.0005
roll		1.1400	1.1092	0.0308
yaw		2.1122	2.1299	-0.0177

续表 4-7

输出	环境	Modelsim	Matlab	误差
第 5 次解算				
pitch		-2.4270	-2.4272	0.0002
roll		1.4268	1.3835	0.0433
yaw		2.6228	2.6488	-0.026

5.3 仿真结果分析

由表 5-1~表 5-5，子模块电路仿真数据与 Matlab 计算数据基本相等，说明电路稳定可靠。

表 5-6、表 5-7 中，姿态角与 Matlab 计算数据有误差，这是因为在姿态角解算过程中，使用的反正切运算 IP 核的为 16 位定点数，在浮点数与定点数转换过程中产生了数据误差，这些是后续电路改进过程中需要注意的问题。电路的解算值与参考值误差集中分布在 $[-0.02^\circ, 0.02^\circ]$ 内，并且通过数据对比可以看出解算值和参考值一样都是在收敛过程中进行的。解算时间约为 40us，而 MCU 的解算速度约为 280us^[4]，解算时间节省 6/7。由于电路数据格式的限制导致的误差不可消除。由于“数学平台”中解算微分方程时进行的积分运算和互补滤波过程中的比例积分运算产生了部分积累误差。另外在电路实际工作环境中，解算误差相较于飞行器的飞行状态而言可以忽略不计，并且当引入闭环 PID 反馈控制后，误差会相应的减小。在仿真过程中，对调用的 IP 核的输入和输出标志信号可能会在时钟边沿丢失，从而造成计算错误，需要将标志信号进行保持以确保信号不会产生“竞争与冒险”现象，设计中采用了多个空状态进行标致信号的保持与采样来避免处理过程中产生“竞争与冒险”或“亚稳态”现象发生。

通过以上数据分析可知电路在姿态角解算方面具有良好的可靠性和稳定性。

6 结论与展望

本文在前人的研究成果和现有的经验基础上，从工程设计和仿真验证角度对捷联式惯性导航算法进行了硬线逻辑实现的设计，整个电路系统的建立包括了对捷联式惯性导航算法的推导和基于 ISE 与 Xilinx IP CORE Generator 的电路设计。

设计过程主要完成了下列内容：

(1) 深入研究了捷联式惯性导航系统的基本部件和工作机制，给出了基于四元数的载体姿态转换矩阵的求解算法，推导了姿态角的解算过程，并研究了数学平台的解析功能。

(2) 使用 Matlab 软件进行了载体飞行轨迹的模拟，并完成了姿态解算算法的验证，同时验证了系统设计的合理性和可行性。

(3) 结合上述研究内容以及芯片的设计需求，通过理论分析确定了使用 FPGA 芯片为系统测试平台处理器的设计方案，改变传统的使用通用 MCU 进行导航和姿态解算的设计思路。

(4) 使用 Verilog HDL 语言对设计的电路进行 RTL 实现，并通过仿真测试验证了电路的可靠性和可行性。并且在设计过程中对电路中的固定常数使用了可调定义，并且采用模块化的设计思路，通过大面积的并行操作和面积复用，大大减少了解算时间和资源消耗，降低了系统成本，并且易于维护。

本设计的创新点有以下：

(1) 对陀螺仪和加速度计输出采用互补滤波处理和用四元数法来解算姿态矩阵，并将算法原理成功使用硬线逻辑实现。

(2) 采用四元数法求解姿态微分方程，得到相应的姿态矩阵进而得到姿态角，在最后的结果中可以看出姿态角的误差满足系统误差的要求。

(3) 相对其他处理方法，互补滤波能够一定程度上提供姿态解算的精度，且相对来说算法简单易实现。从最后的结果比较可以看出，互补滤波处理后，姿态角的平均误差有所减小，特别是在后段时间的误差明显减小。

(4) 硬线逻辑设计过程中使用大量逻辑优化措施，使得整个系统占用的门数和面试达到最小，大大减少了设计成本，并为后续流片工作奠定了良好的基础。

(5) 大胆地使用并行处理方式来进行惯性导航的姿态角解算，与传统的 MCU 解算相比，在解算精度显著提高的前提下，使得解算时间明显减少为 MCU 解算时间的 1/7。

在进行研究设计工作的同时，发现了一些不足之处，需要在后续工作中进行完善。本设计主要基于 Matlab 软件进行了捷联式惯性导航系统算法的验证、基于 ISE 软件的对 Verilog HDL 语言描述的 RTL 代码进行了综合和基于 Modelsim 软件的功能验证。但是由于时间关系，没有使用真实传感器采集的导航数据进行测试分析，设计的 RTL 实现都是在基于理想化状态下的算法进行编写，解算后的误差基本都是由于算法设计过程中产生的误差，仅仅依靠简单的互补滤波算法进行的补偿在实际工作环境中效果不明显。实际环境中进行长时间或长距离导航时将会对解算精度产生极大影响。

在后续算法优化中，需要引入 GPS 或北斗导航系统，对误差进行实时校正。并使用较高级的滤波算法，例如卡尔曼滤波算法或粒子滤波算法对系统进行完善和提升。

在后续仿真测试中，将加入一内置 ROM 用来存放实测数据，对电路系统进行更接近实际环境的测试。

以后的工作，将对目前不够完善的内容进行深入研究，待设计完善并通过板级测试后，进行流片测试及成果化。

参考文献

- [1] 秦永元. 惯性导航[M]. 北京: 科学出版社, 2006: 1-12, 203-215, 287-304.
- [2] 贾胜文. 基于 ARM7CPU 和 FPGA 的低成本小型捷联惯导系统的设计[D]. 硕士学位论文. 西安: 西北工业大学, 2006: 4-19.
- [3] 侯伟先. 基于 FPGA 的惯测系统 IP 核设计与实现[D]. 硕士学位论文. 长沙: 国防科学技术大学, 2009: 2-9.
- [4] S.Bouabdallah. Design and Control of Quadrotors with Application to Autonomous Flying[D]. PhD Thesis. Lausanne: EPFL, 2007: 15-28.
- [5] 秦永元, 张洪钺, 汪淑华. 卡尔曼滤波与组合导航原理(第二版) [M]. 西安: 西北工业大学出版社, 2012: 1-4.
- [6] 黄德鸣, 程禄. 惯性导航系统[M]. 北京: 国防工业出版社, 1986: 16-26.
- [7] 宫经宽. 航空机载惯性导航系统[M]. 北京: 航空工业出版社, 2010: 1-32.
- [8] 李玉. 低成本捷联惯性导航算法研究及其 Matlab 仿真[D]. 学士学位论文. 哈尔滨: 哈尔滨工业大学, 2014: 1-4.
- [9] R.Brockhaus. Flugregelung[M]. New York: Springer-Verlag, 1994: 5-14.
- [10] Robert M, Tarek H, Jean M P. Complimentary filter design on the special orthogonal group SO(3)[A]. In: Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference 2005[C]. Seville, Spain, Dec 12-15, 2005: 1477-1484.
- [11] S.Madgwick, A.Harrison, R.Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm[A]. In: 2011 IEEE International Conference on Rehabilitation Robotics[C]. Rehab Week Zurich, ETH Zurich Science City, Switzerland, June 29-July 1, 2011.
- [12] 夏宇闻. Verilog 数字系统设计教程(第3版) [M]. 北京: 北京航空航天大学出版社, 2013: 6-7, 17-49, 120-127.
- [13] 栗瑞江. 低成本 SINS/GPS 组合导航系统研究[D]. 硕士学位论文. 西安: 西北工业大学, 2001: 4-6.
- [14] 国际天文协会“基本天文学术语”工作组. 国际天文学会“基本天文学术语”及其解释[M]. 国际天文协会, 2007: 1-5.
- [15] 邓正隆. 惯性技术[M]. 哈尔滨: 哈尔滨工业大学出版社, 2006: 1-56.
- [16] 黄溪流. 一种四旋翼无人直升机飞行控制器的设计[D]. 硕士学位论文. 南京: 南京理工大学, 2010: 1-4.
- [17] 张帆, 曹喜滨, 邹经湘. 一种新的全角度四元数与欧拉角的转换算法[J]. 南京理工大学学报, 2002, 26(4): 376-380.
- [18] 周亢, 闫建国, 屈耀红. 全角度姿态角解算方法研究与仿真[J]. 系统仿真学报, 2009, 21(6): 1697-1700.
- [19] 陈哲. 捷联惯导系统原理[M]. 北京: 宇航出版社, 1986: 141-152.
- [20] 储海荣, 段镇, 贾宏光, 等. 捷联惯导系统的误差模型与仿真[J]. 光学精密工程, 2009, 17(11): 2779-2785.
- [21] 于永军, 刘建业, 熊智, 等. 高动态载体高精度捷联惯导算法[J]. 中国惯性技术学报, 2011, 19(2): 136-139.
- [22] D.Mellinger, N.Michael, V.Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors[J]. The International Journal of Robotics Research, 2012, 31(5): 664-674.

- [23] M.Euston, P.Coote, R.Mahony, et al. A complementary filter for attitude estimation of a fixed-wing UAV[A]. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems[C]. Nice, France, Sept 22-26, 2008: 340-345.
- [24] 马春艳, 刘莉, 杜小菁, 等. 末制导炮弹姿态更新算法研究[J]. 北京理工大学学报, 2005, 25(9): 808-811.
- [25] 闫超. 超紧 SINS/GPS 组合导航系统技术研究[D]. 硕士学位论文. 哈尔滨: 哈尔滨工程大学, 2009: 6-21.
- [26] 罗武胜, 徐涛, 杜列波. 基于加速度计和磁强计的定向钻进姿态测量及方位校正[J]. 国防科学技术大学学报, 2007, 29(1): 106-110.
- [27] 谷宏强, 袁亚雄, 李斌. 捷联惯导系统快速自对准误差模型的建立[J]. 火力与指挥控制, 2008, 33(12): 137-139.
- [28] 雷芳, 王华, 焦国太, 等. 弹道修正弹药的姿态测量技术研究[J]. 弹箭与制导学报, 2009, 29(4): 123-125+132.
- [29] 景丽. 基于卡尔曼滤波组合导航算法的计算量与精度分析[D]. 硕士学位论文. 哈尔滨: 哈尔滨工业大学, 2014: 6-19.
- [30] B.L.Yuan, D. Liao, S.L.Han. Error compensation of an optical gyro INS by multi-axis rotation[J]. Measurement Sci. Tech. Vol.23, No.2, Feb, 2012.
- [31] R.Azor, I.Bar-Itzhack, J.Deutschmann, et al. Angular-Rate Estimation Using Quaternion Measurements[J]. NASA STI, 1998, 1: 1-6.
- [32] 张春慧, 吴简彤, 何昆鹏, 郭新伟. 四阶龙格-库塔法在捷联惯导系统姿态解算中的应用[J]. 声学与电子工程, 2005, 01: 37-38+48.
- [33] 郦正能, 程小全, 方卫国, 等. 飞行器结构力学(第2版)[M]. 北京: 北京航空航天大学出版社, 2010: 22-27.
- [34] 胡寿松. 自动控制原理(第六版)[M]. 北京: 国防工业出版社, 1984: 47-80.
- [35] Gene F. Franklin, J.David Powell, Abbas Emami-Naeini. Feedback Control of Dynamic Systems[M]. Sixth Edition. 北京: 电子工业出版社, 2014: 63-75.
- [36] L.Yang, XM.Zhao, F.Hui, X.Shi. FPGA Implementation of VSCS-LMS Algorithm for MEMS Gyroscope[A]. In: 2012 2nd International Conference on Computer Science and Network Technology[C]. Changchun, China, 2012: 1710-1714.
- [37] 唐斌, 黄一敏. 基于 Matlab 的无人机全过程飞行仿真[J]. 沈阳航空工业学院学报, 2007, 24(1): 13-16.
- [38] Xilinx Inc. LogiCORE IP Floating-Point Operator v5.0 [Z]. 2011.
- [39] 李庆扬, 王能超, 易大义. 数值分析(第5版)[M]. 北京: 清华大学出版社, 2008: 4-20.
- [40] Xilinx Inc. LogiCORE IP CORDIC v4.0 [Z]. 2011.

致 谢

附录 A IP 核的例化

```

atan ip1 (
    .x_in(x_in),
    .y_in(y_in),
    .nd(nd),
    .phase_out(phase_out),
    .rdy(rdy),
    .clk(clk),
    .ce(ce),
    .sclr(sclr)
);
// 反正切运算
// input [15 : 0] x_in
// input [15 : 0] y_in
// input nd
// output [15 : 0] phase_out
// output rdy
// input clk
// input ce
// input sclr

add ip2 (
    .a(add_in_a),
    .b(add_in_b),
    .operation_nd(add_start),
    .clk(clk),
    .sclr(sclr),
    .ce(ce),
    .result(add_out),
    .rdy(add_end)
);
// 加法器
// input [31 : 0] a
// input [31 : 0] b
// input operation_nd
// input clk
// input sclr
// input ce
// output [31 : 0] result
// output rdy

sub ip3 (
    .a(sub_in_a),
    .b(sub_in_b),
    .operation_nd(sub_start),
    .clk(clk),
    .sclr(sclr),
    .ce(ce),
    .result(sub_out),
    .rdy(sub_end)
);
// 減法器
// input [31 : 0] a
// input [31 : 0] b
// input operation_nd
// input clk
// input sclr
// input ce
// output [31 : 0] result
// output rdy

mult ip4 (
    .a(mult_in_a),
    .b(mult_in_b),
    .operation_nd(mult_start),
    .clk(clk),
    .sclr(sclr),
    .ce(ce),
    .result(mult_out),
    .rdy(mult_end)
);
// 乘法器
// input [31 : 0] a
// input [31 : 0] b
// input operation_nd
// input clk
// input sclr
// input ce
// output [31 : 0] result
// output rdy

div ip5 (
    .a(div_in_a),
    .b(div_in_b),
    .operation_nd(div_start),
    .clk(clk),
    .sclr(sclr),
    .ce(ce),
    .result(div_out),
    .rdy(div_end)
);
// 除法器
// input [31 : 0] a
// input [31 : 0] b
// input operation_nd
// input clk
// input sclr
// input ce
// output [31 : 0] result
// output rdy

```

```

);

sqrt ip6 (
    .a(sqrt_in), // 开方器
    .operation_nd(sqrt_start), // input [31 : 0] a
    .clk(clk), // input operation_nd
    .sclr(sclr), // input clk
    .ce(ce), // input sclr
    .result(sqrt_out), // input ce
    .rdy(sqrt_end) // output [31 : 0] result
);

f32tof16 ip7 (
    .a(li_in), // 浮点转定点
    .operation_nd(li_start), // input [31 : 0] a
    .clk(clk), // input operation_nd
    .sclr(sclr), // input clk
    .ce(ce), // input sclr
    .result(li_out), // input ce
    .rdy(li_end) // output [15 : 0] result
);

f16tof32 ip8 (
    .a(il_in), // 定点转浮点
    .operation_nd(il_start), // input [15 : 0] a
    .clk(clk), // input operation_nd
    .sclr(sclr), // input clk
    .ce(ce), // input sclr
    .result(il_out), // input ce
    .rdy(il_end) // output [31 : 0] result
);

equ1 ip9 (
    .a(equl_in), // 相等比较器
    .b(equl_in0), // input [31 : 0] a
    .operation_nd(equl_start), // input [31 : 0] b
    .clk(clk), // input operation_nd
    .sclr(sclr), // input clk
    .ce(ce), // input sclr
    .result(equl_out), // input ce
    .rdy(equl_end) // output [0 : 0] result
);

less ip10 (
    .a(less_in), // 小于比较器
    .b(less_in0), // input [31 : 0] a
    .operation_nd(less_start), // input [31 : 0] b
    .clk(clk), // input operation_nd
    .sclr(sclr), // input clk
    .ce(ce), // input sclr
    .result(less_out), // input ce
    .rdy(less_end) // output [0 : 0] result
);

```

附录 B 顶层电路状态机的 RTL 实现

B.1 顶层电路状态迁移的 RTL 实现

```
case(c_state)
    IDLE      : begin
                    if(in_data_en)
                        n_state = INS_ACC_MAG_P;
                    else
                        n_state = IDLE;
                end
    INS_ACC_MAG_P   : n_state = INS_ACC_MAG;
    INS_ACC_MAG    : n_state = INS_ACC_MAG_TMP;
    INS_ACC_MAG_TMP : begin
                    if(acc_finish && mag_finish)
                        n_state = INS_GYRO_P;
                    else
                        n_state = INS_ACC_MAG_TMP;
                end
    INS_GYRO_P     : n_state = INS_GYRO;
    INS_GYRO       : n_state = INS_GYRO_TMP;
    INS_GYRO_TMP   : begin
                    if(gyro_finish)
                        n_state = INS_QUAT_P;
                    else
                        n_state = INS_GYRO_TMP;
                end
    INS_QUAT_P     : n_state = INS_QUAT;
    INS_QUAT        : n_state = INS_QUAT_TMP;
    INS_QUAT_TMP   : begin
                    if(quat_finish)
                        n_state = INS_MATX_P;
                    else
                        n_state = INS_QUAT_TMP;
                end
    INS_MATX_P     : n_state = INS_MATX;
    INS_MATX        : n_state = INS_MATX_TMP;
    INS_MATX_TMP   : begin
                    if(matx_finish)
                        n_state = INS_EULER_P;
                    else
                        n_state = INS_MATX_TMP;
                end
    INS_EULER_P     : n_state = INS_EULER;
    INS_EULER        : n_state = INS_EULER_TMP;
    INS_EULER_TMP   : begin
                    if(euler_finish)
                        n_state = LAST;
                    else
                        n_state = INS_EULER_TMP;
                end
    LAST           : n_state = LAST_TMP;
    LAST_TMP        : n_state = END;
    END             : n_state = END_TMP;
    END_TMP         : n_state = FINISH;
    FINISH          : n_state = IDLE;
    default         : n_state = IDLE;
```

```

    endcase
B.2 顶层电路状态操作的 RTL 实现
case(c_state)
    IDLE      : begin
                    if(in_data_en)
                        out_INS_finish <= 1'b0;
                    else
                        out_INS_finish <= out_INS_finish;
                end
    INS_ACC_MAG_P   : begin
                    acc_start <= 1'b1;
                    mag_start <= 1'b1;
                end
    INS_ACC_MAG     : begin
                    acc_start <= 1'b0;
                    mag_start <= 1'b0;
                end
    INS_GYRO_P      : gyro_start <= 1'b1;
    INS_GYRO        : gyro_start <= 1'b0;
    INS_QUAT_P      : quat_start <= 1'b1;
    INS_QUAT         : quat_start <= 1'b0;
    INS_MATX_P      : matx_start <= 1'b1;
    INS_MATX         : matx_start <= 1'b0;
    INS_EULER_P      : euler_start <= 1'b1;
    INS_EULER         : euler_start <= 1'b0;
    LAST             : begin
                    c_exi <= p_exi;
                    c_eyi <= p_eyi;
                    c_ezi <= p_ezi;
                    c_q0 <= p_q0;
                    c_q1 <= p_q1;
                    c_q2 <= p_q2;
                    c_q3 <= p_q3;
                    c_c11 <= p_c11;
                    c_c12 <= p_c12;
                    c_c13 <= p_c13;
                    c_c21 <= p_c21;
                    c_c22 <= p_c22;
                    c_c23 <= p_c23;
                    c_c31 <= p_c31;
                    c_c32 <= p_c32;
                    c_c33 <= p_c33;
                end
    END              : begin
                    out_pitch <= out_pitch_tmp;
                    out_roll <= out_roll_tmp;
                    out_yaw <= out_yaw_tmp;
                end
    FINISH          : out_INS_finish <= 1'b1;
    default          : begin
                    acc_start <= acc_start;
                    mag_start <= mag_start;
                    gyro_start <= gyro_start;
                    quat_start <= quat_start;
                    matx_start <= matx_start;
                    euler_start <= euler_start;
                end
end
endcase

```

B.3 顶层电路中子模块电路的例化

```

INS_acc_u1 (
    .clk(clk)                                // 加速度计测量误差计算电路
    .rst(rst)                                 // input clk
    .in_acc_en(acc_start)                      // input rst
    .in_accx(in_accx)                         // input acc_en
    .in_accy(in_accy)                         // [31 : 0] accx
    .in_accz(in_accz)                         // [31 : 0] accy
    .in_c13(c_c13)                            // [31 : 0] accz
    .in_c23(c_c23)                            // [31 : 0] c13
    .in_c33(c_c33)                            // [31 : 0] c23
    .out_exa(exa)                             // [31 : 0] c33
    .out_eya(eya)                            // output [31 : 0] exa
    .out_eza(eza)                            // output [31 : 0] eya
    .out_acc_finish(acc_finish)                // output [31 : 0] eza
);
                                            // output acc_finish

INS_mag_u2 (
    .clk(clk)                                // 电子罗盘测量误差计算电路
    .rst(rst)                                 // input clk
    .in_mag_en(mag_start)                     // input rst
    .in_magx(in_magx)                         // input mag_en
    .in_magy(in_magy)                         // [31 : 0] magx
    .in_magz(in_magz)                         // [31 : 0] magy
    .in_c11(c_c11)                            // [31 : 0] magz
    .in_c12(c_c12)                            // [31 : 0] c11
    .in_c13(c_c13)                            // [31 : 0] c12
    .in_c21(c_c21)                            // [31 : 0] c13
    .in_c22(c_c22)                            // [31 : 0] c21
    .in_c23(c_c23)                            // [31 : 0] c22
    .in_c31(c_c31)                            // [31 : 0] c23
    .in_c32(c_c32)                            // [31 : 0] c31
    .in_c33(c_c33)                            // [31 : 0] c32
    .out_exm(exm)                            // output [31 : 0] c33
    .out_eym(eym)                            // output [31 : 0] exm
    .out_ezm(ezm)                            // output [31 : 0] eym
    .out_mag_finish(mag_finish)                // output [31 : 0] ezm
);
                                            // output mag_finish

INS_gyro_u3 (
    .clk(clk)                                // 陀螺仪测量误差修正电路
    .rst(rst)                                 // input clk
    .in_gyro_en(gyro_start)                   // input rst
    .in_gyrox(in_gyrox)                        // input gyro_en
    .in_gyroy(in_gyroy)                        // [31 : 0] gyrox
    .in_gyroz(in_gyroz)                        // [31 : 0] gyroy
    .in_exi(exa)                             // [31 : 0] gyroz
    .in_eyi(eya)                            // [31 : 0] exa
    .in_ezi(eza)                            // [31 : 0] eya
    .in_exm(exm)                            // [31 : 0] eza
    .in_eym(eym)                            // input [31 : 0] exm
    .in_ezm(ezm)                            // input [31 : 0] eym
    .in_exi(c_exi)                           // input [31 : 0] ezm
    .in_eyi(c_eyi)                           // [31 : 0] exi
    .in_ezi(c_ezi)                           // [31 : 0] eyi
    .out_exi(p_exi)                          // [31 : 0] ezi
    .out_eyi(p_eyi)                          // output [31 : 0] exi
    .out_ezi(p_ezi)                          // output [31 : 0] eyi
);
                                            // output [31 : 0] ezi

```

```

.out_wx(wx) , // output [31 : 0] wx
.out_wy(wy) , // output [31 : 0] wy
.out_wz(wz) , // output [31 : 0] wz
.out_gyro_finish(gyro_finish) // output gyro_finish
);

INS_quat_u4 (
    .clk(clk) , // input clk
    .rst(rst) , // input rst
    .in_quat_en(quat_start) , // input quat_en
    .in_q0(c_q0) , // input q0
    .in_q1(c_q1) , // input q1
    .in_q2(c_q2) , // input q2
    .in_q3(c_q3) , // input q3
    .in_wx(wx) , // input [31 : 0] wx
    .in_wy(wy) , // input [31 : 0] wy
    .in_wz(wz) , // input [31 : 0] wz
    .out_q0(p_q0) , // output [31 : 0] q0
    .out_q1(p_q1) , // output [31 : 0] q1
    .out_q2(p_q2) , // output [31 : 0] q2
    .out_q3(p_q3) , // output [31 : 0] q3
    .out_quat_finish(quat_finish) // output quat_finish
);

INS_matx_u5 (
    .clk(clk) , // input clk
    .rst(rst) , // input rst
    .in_matx_en(matx_start) , // input matx_en
    .in_q0(p_q0) , // input [31 : 0] q0
    .in_q1(p_q1) , // input [31 : 0] q1
    .in_q2(p_q2) , // input [31 : 0] q2
    .in_q3(p_q3) , // input [31 : 0] q3
    .out_c11(p_c11) , // output [31 : 0] c11
    .out_c12(p_c12) , // output [31 : 0] c12
    .out_c13(p_c13) , // output [31 : 0] c13
    .out_c21(p_c21) , // output [31 : 0] c21
    .out_c22(p_c22) , // output [31 : 0] c22
    .out_c23(p_c23) , // output [31 : 0] c23
    .out_c31(p_c31) , // output [31 : 0] c31
    .out_c32(p_c32) , // output [31 : 0] c32
    .out_c33(p_c33) , // output [31 : 0] c33
    .out_matx_finish(matx_finish) // output [31 : 0] matx_finish
);

INS_euler_u6 (
    .clk(clk) , // input clk
    .rst(rst) , // input rst
    .in_euler_en(euler_start) , // input euler_en
    .in_c11(p_c11) , // input [31 : 0] c11
    .in_c12(p_c12) , // input [31 : 0] c12
    .in_c13(p_c13) , // input [31 : 0] c13
    .in_c23(p_c23) , // input [31 : 0] c23
    .in_c33(p_c33) , // input [31 : 0] c33
    .out_pitch(out_pitch_tmp) , // output [31 : 0] pitch
    .out_roll(out_roll_tmp) , // output [31 : 0] roll
    .out_yaw(out_yaw_tmp) , // output [31 : 0] yaw
    .out_euler_finish(euler_finish) // output [31 : 0] euler_finish
);

```