**2020年新工科联盟-Xilinx暑期学校团队项目设计文档**

|  |  |
| --- | --- |
| **作品名称** | 给ESP32增加FPU |
| **板卡型号** | SPARTAN-7 XC7S15 |
| **所在班级** | 2020A班 |
| **成员姓名、学号、学校** | 韩辉珺22017102东南大学仪器科学与工程  陆佳晨22017113东南大学仪器科学与工程 |
| **Github链接** | https://github.com/CarolineJun/esp32\_FPU |

**第一部分**

设计概述 /Design Introduction

（1.请概括地描述一下你的设计，可包括本设计目的、学习到的知识点、应用方向或者设想的应用场景等；2. 经组内成员讨论后以表格的形式描述项目中各成员在项目中发挥的作用或者贡献百分比；

3.作品的展示照片）

1.简介

设计目的：本次设计的主题是给ESP32增加FPU（给ESP32增加浮点运算功能），从而可以利用FPGA与ESP32通讯、处理数据。

学习的知识点：ESP32在计算浮点数运算的时候，不只是微控制器在独立工作，协助计算的还有FPGA，现场可编程逻辑门阵列。现场可编程逻辑门阵列的开发依靠xilinx提供的vivado软件，其中有丰富的IPcore给我们用来实现不同的功能。当协处理器现场可编程逻辑门阵列通讯的时候，可以利用QSPI协议实现现场可编程逻辑门阵列与ESP32之间形成全双工的，高速的，可靠的，同步时钟的数据交换和地址交换和指令交换。利用现场可编程逻辑门阵列可以并行处理数据，大大降低进行浮点数运算的时候所需要的时间。

应用场景：微控制器(MCU)深入人们应用生活，几乎大小设备都看得到MCU踪影，在MCU导入DSP数字信号处理器、FPU浮点运算单元功能后，MCU更大幅扩展元件可适用范围，这几年来，在众多MCU大厂纷纷针对旗下商品推出多样整合方案，不管是产品策略还是市场区隔，也让MCU市场更加丰富多元...MCU(Microcontroller Unit)深入生活应用是不容易质疑的趋势，尤其是MCU在功能优化或市场区隔目的下，进行DSP(Digital Signal Processor)数字信号处理器或FPU(Floating Point Unit)浮点运算单元功能整合，使得MCU的可应用场域大幅扩展。MCU整合FPU可以在进阶数值运算的精密度大幅提升、处理效能也能获得改善。针对IoT应用开发的MCU方案，整合DSP可优化感测器数据撷取品质与提升信号处理效能。如果以FPU或DSP导入目的，一般在MCU中追加FPU、DSP整合架构，主要目的还是在考量成本下的设计方向，尤其在早期半导体元件，SOC(System on Chip)系统单芯片与MCU存在一段价格差距，如果仅需要SDP或FPU进行运算加速，又不想选用高单价SOC，这时整合DSP或FPU硬件加速单元的MCU产品、不仅可以更好的提供运行效能，同时又能在成本控制上表现更加优异。MCU整合芯片封装成本骤降 增加MCU功能扩充应用空间

以早期的SOC产品来看，搭载DSP与FPU硬件加速器是SOC产品的重要特性，其中DSP与FPU的应用方向主要以音讯、影像等处理加速运算为主，而在制程技术持续优化，SOC的成本逐步与MCU拉近，MCU在32位元甚至64位元架构下，也开始有结合DSP或是FPU硬件加速单元的解决方案。

先看看MCU加上硬件加速单元的优点，在MCU追加FPU导入，最直接的效益是早期利用MCU处理类似FPU运算内容，会因为MCU本身的运算架构限制，让运算结果得出时间会相对拉长，而在导入硬件加速器处理浮点运算时，因为硬件呼叫或是资料传递就能透过硬件算出数据，MCU本身耗在浮点运算的记忆体资源可以因硬件加速整合减少至少10%。

当然，从目的性来看，不管MCU有无整合FPU硬件加速单元，浮点运算需求使用MCU现有的运算能力也能得出结果，只是前提是计算过程会耗用较多运算时间与硬件资源，对于可等待、无需提供即时反应的系统自然可以不考虑整合FPU的MCU方案，但若是对系统效能、回馈反应速度要求高的整合需求，MCU结合FPU的效益提升不仅仅是运算资源耗用优化、节能优势等效果，反而是加快系统回应与效能提升的效用，才是MCU结合FPU硬件加速最直接、重要的功能改进，也让MCU可以因应更高复杂度的整合工作。

**高阶数值运算 运用硬件加速满足设计需求**

在早期MCU元件仍以8位元架构为主流的应用方向，MCU在资料处理与运算处理上，本来就有因架构的问题而有其处理限制，例如，MCU进行小数点、分数处理运算时，因为4位元或是8位元位数有限，就必须采用有限数值进行处理，透过数值结果的限制换取处理复杂度简化与效能要求目的，而这种因为数值处理产生的误差即“截断误差”，截断误差也会因为使用MCU进行数据运算的限制，而令误差数值产生扩大现象。

而在MCU整合FPU硬件加速，在运算同类型的数据处理时，例如在IoT物联网或是终端感测器应用中，常有将外部类比感测数据转换成数字资料的资料撷取、处理需求，这时透过MCU整合的FPU/DSP硬件加速单元，不仅可将感测数据更快速处理完成、加快系统回应，同时，也能导入进阶运算减少数据演算的误差。

在实际应用中，FPU硬件加速器本身并无法完全解决误差扩大问题，所以会有FPU、DSP等不同硬件加速整合架构下的应用目的考量，举例来说，透过DSP硬件加速器，可针对特殊数据类型更高速、可靠的运算处理输出，像是DSP可利用指令来进行多种运算，处理如快速快速傅立叶转换(fast Fourier transform；FFT)或有限脉冲回应(Finite impulse response；FIR)进阶运算中重要且耗资源的运算需求，甚至透过单周期的指令便能处理单一指令多重资料(Single Instruction Multiple Data；SIMD)运算需求，MCU在进行进阶数值处理方面还可获得进阶增强效益。

**FPU/DSP不同硬件加速单元具互补作用**

虽说整合FPU或DSP基本在架构与应用方向就不同，但实际上两者分别是针对数据运算、讯号处理对应至各式演算法应用，两者功能可以说是各有互补效用，比较难被独立拆分。以ARM Cortex-M4来看，若仅提供DSP硬件加速处理器反而没设置FPU浮点运算加速器反而会造成应用限制，因为在Cortex-M4应用场合如果仅有数字信号处理加速硬件支援，少了浮点运算支援，对开发需求端若碰到需要数值进阶运算加速，就会造成设计上的弹性限制，或是导致还需透过外部功能芯片支援，或利用原有的运算资源因应数值进阶计算需求，反而会因为数值处理效能限制了Cortex-M4的应用可能性。

同样的状况也发生在仅有FPU而没有设置DSP的微控制器应用方案上，对DSP或是FPU应用功能是相辅相成，独立整合对于微控制器的配置并未能产生综效，反而会成为发展路径的限制。

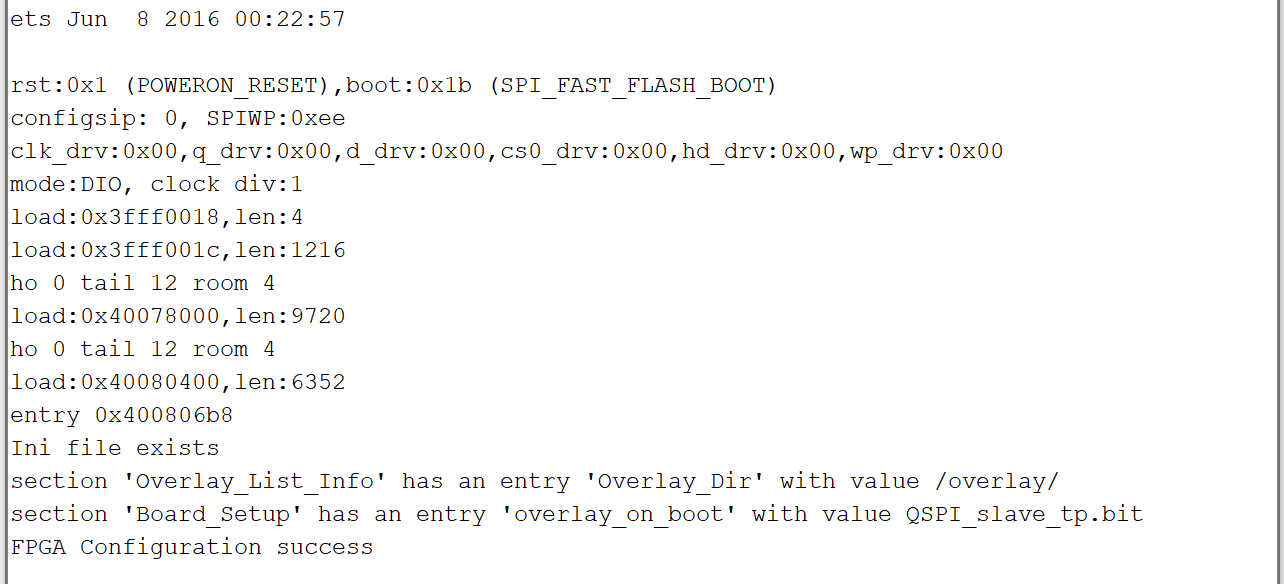
再者，从新一代IoT产品发展方向，透过感测器融合(Sensor Fusion)应用方向为例，若是Sensor Fusion概念为将多感测器整合在单一系统中协同运行，系统需要高阶数值与讯号处理能力，才可以将关键数值讯号自复杂数据中提取出来。

至于感测器融合可以再搭配即时的调整、控制与校正处理，由DSP加上FPU协同处理达到高精密度、高效率进行撷取数据的精密分析，尤其是现有的Sensor Fusion已做到陀螺仪、加速度器、温度、压力甚至触控感测都做在同一个模组中，必须透过DSP与FPU预先筛出相对精密且兼顾处理效率的讯号撷取与预处理的感测数据，提供相对高效的系统更具效率的感测数值处理机制。

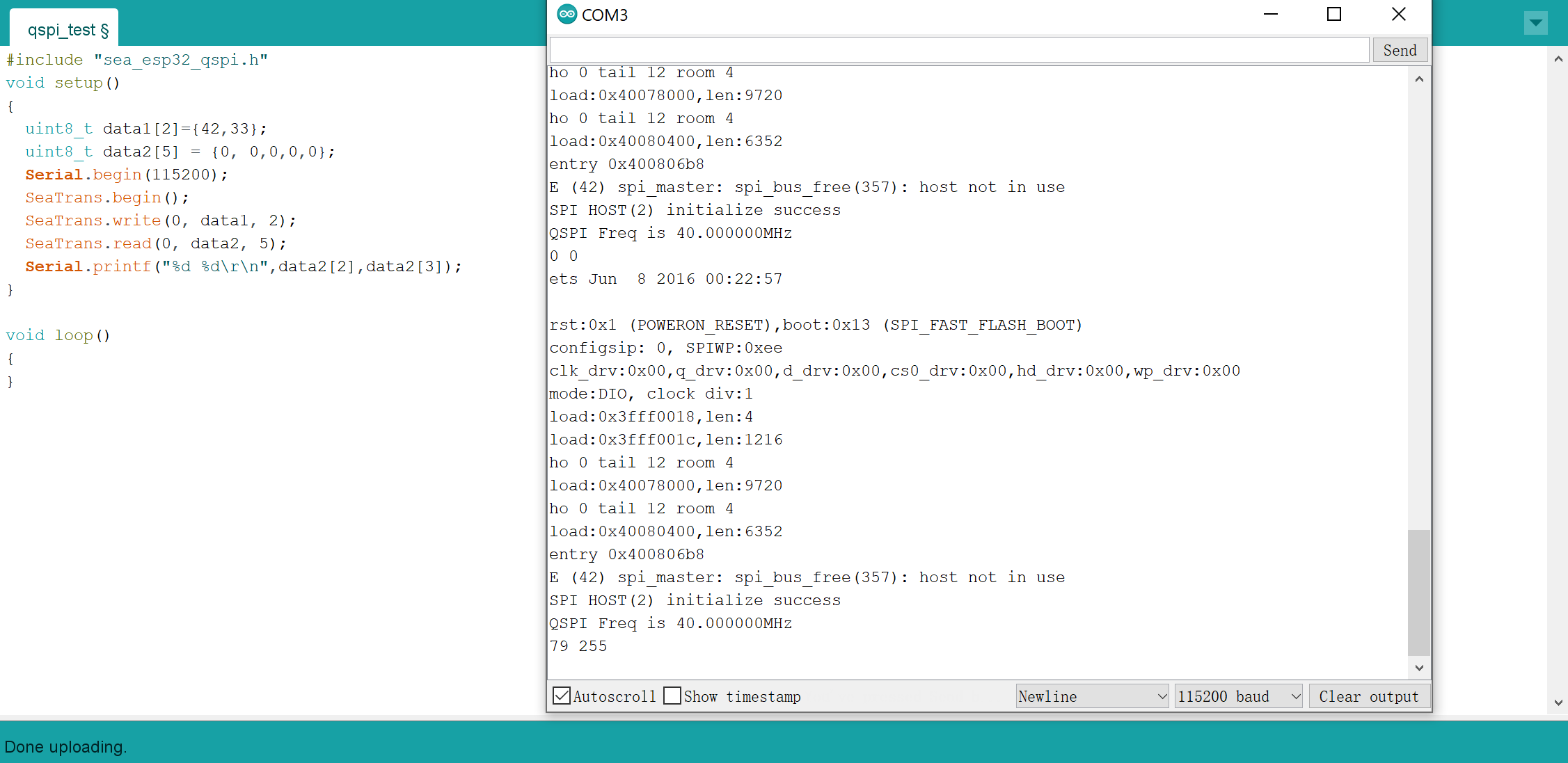
2.组员分工

|  |  |
| --- | --- |
| 韩辉珺 | 进行ESP32与FPGA的通讯  完成报告 |
| 陆佳晨 | 实现在FPGA上的浮点数运算  完成报告 |

3.作品展示



FPGA的bitstream加载成功。



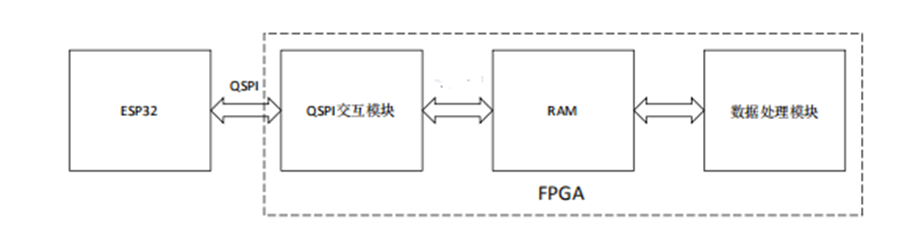
完成浮点运算并把数据传回。

**第二部分**

系统组成及功能说明 /System Construction & Function Description

（请对作品的1. 计划实现及已实现的功能；2. 项目系统框图；3. 使用的技术方向做说明）

ESP32在计算浮点数运算的时候，不只是微控制器在独立工作，协助计算的还有FPGA，现场可编程逻辑门阵列。现场可编程逻辑门阵列的开发依靠xilinx提供的vivado软件，其中有丰富的IPcore给我们用来实现不同的功能。系统框图如下图所示。



项目系统框图

第一章 基于QSPI协议的微控制器和FPGA通信

1.1 QSPI通信协议简介

QSPI是Quad SPI的简写，表示4线SPI，是Motorola公司推出的SPI接口的扩展，比SPI应用更加广泛。其中共有6个接口使协处理器与主机互联。使用该接口，用户可以一次性传输包含多达16个8位或16位数据的传输队列。一旦传输启动，直到传输结束，都不需要CPU干预，极大的提高了传输效率。该协议在ColdFire系列MCU得到广泛应用。与SPI相比，QSPI的最大结构特点是以80字节的RAM代替了SPI的发送和接收数据寄存器。

图片包含 游戏机, 截图, 键盘

描述已自动生成

图 1 QSPI协议时序图

如图 1所示，QSPI协议中的6个数据线分别为nCS，SCLK，IO0，IO1，IO2，IO3。nCS接口被用来启动，当该接口置零时开始工作。SCLK是一个同步时钟计时，每一拍都传送一个数据。当进入命令输入阶段时，在IO0传递8位命令。在传输的过程中我们先传递高位后传递低位。当进入地址传输阶段，为了传递的速度更加快速，我们选择用四个传输口IO0，IO1，IO2，IO3分别传输8位数据的第0和4位，第5和1位，第6和2位，第7和3位。以此类推，就可以传输一个24位的地址在6拍之内。由于我们要通过QSPI对两端进行读和写入的操作，所以我们选择空出4拍的富余，让FPGA进行分析处理数据。在数据的传回阶段，我们同样的采取了，分4个传输口传输的方式，以此来加快传输的速度。在最后，我们并没有结束符，这是因为我们可以通过nCS来控制系统的运行。同时，当nCS拉低时，我们可以一直传输多位数据。

1.2 QSPI通信硬件连接[1]

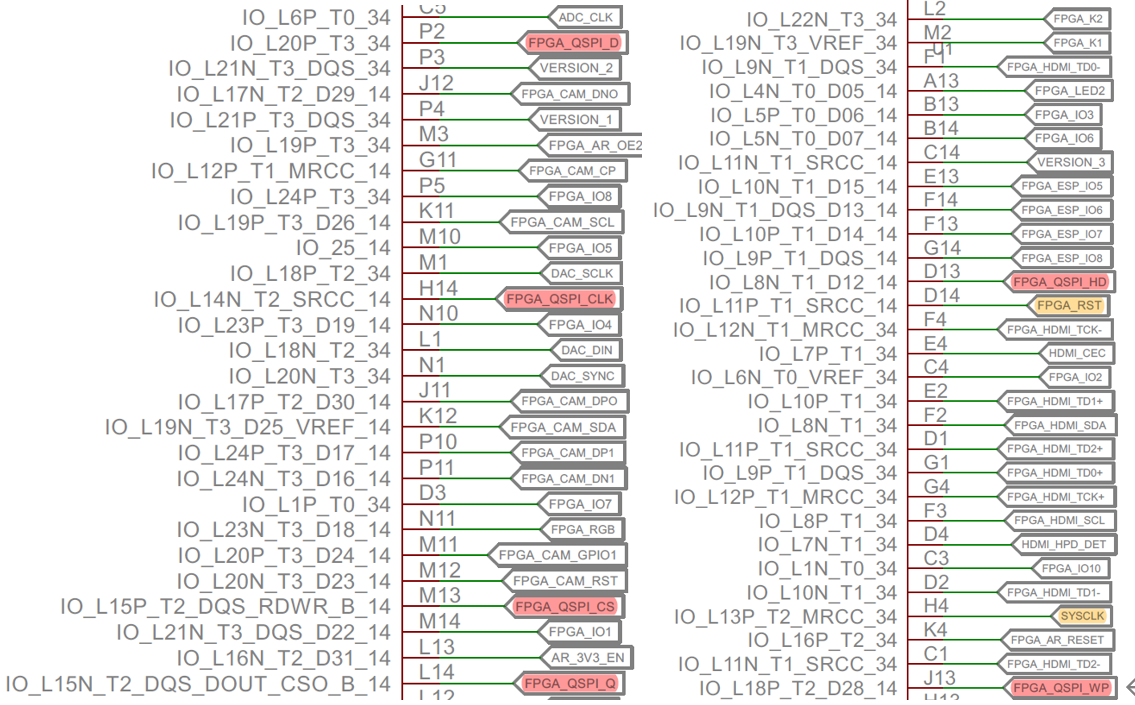


图 2 FPGA与QSPI通信相关的部分管脚原理图

SEA board 上搭载的Spartan-7 FPGA开发板XC7S15-FTGB196C的部分原理图如图 2所示，其中标红的是FPGA上和QSPI通信相关的管脚。P2管脚对应FPGA\_QSPI\_D,H14管脚对应FPGA\_QSPI\_CLK,M13管脚对应FPGA\_QSPI\_CS,D13管脚对应FPGA\_QSPI\_HD,L14管脚对应FPGA\_QSPI\_Q,D13管脚对应FPGA\_QSPI\_HD,J13管脚对应FPGA\_QSPI\_WP。

图 3为ESP32-dD0WDQ6的部分原理图，其中标红的是QSPI通信使用的管脚。其中，GPIO21对应ESP\_QSPI\_HD,GPIO22对应ESP\_QSPI\_WP,GPIO19对应ESP\_QSPI\_Q,GPIO23对应ESP\_QSPI\_D,GPIO18对应ESP\_QSPI\_CLK,GPIO5对应ESP\_QSPI\_CS。

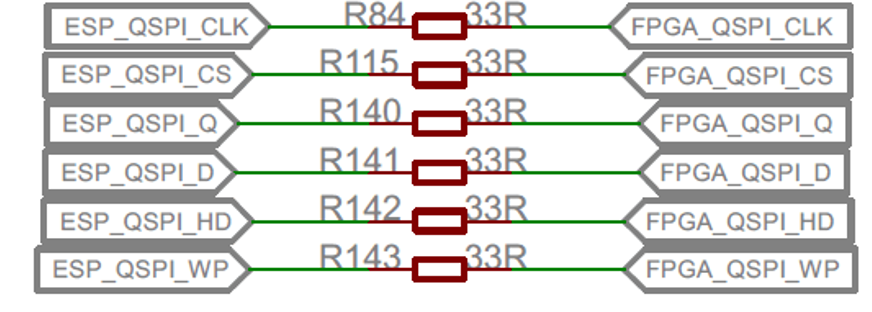


图 4 ESP32与FPGA的QSPI接口电路

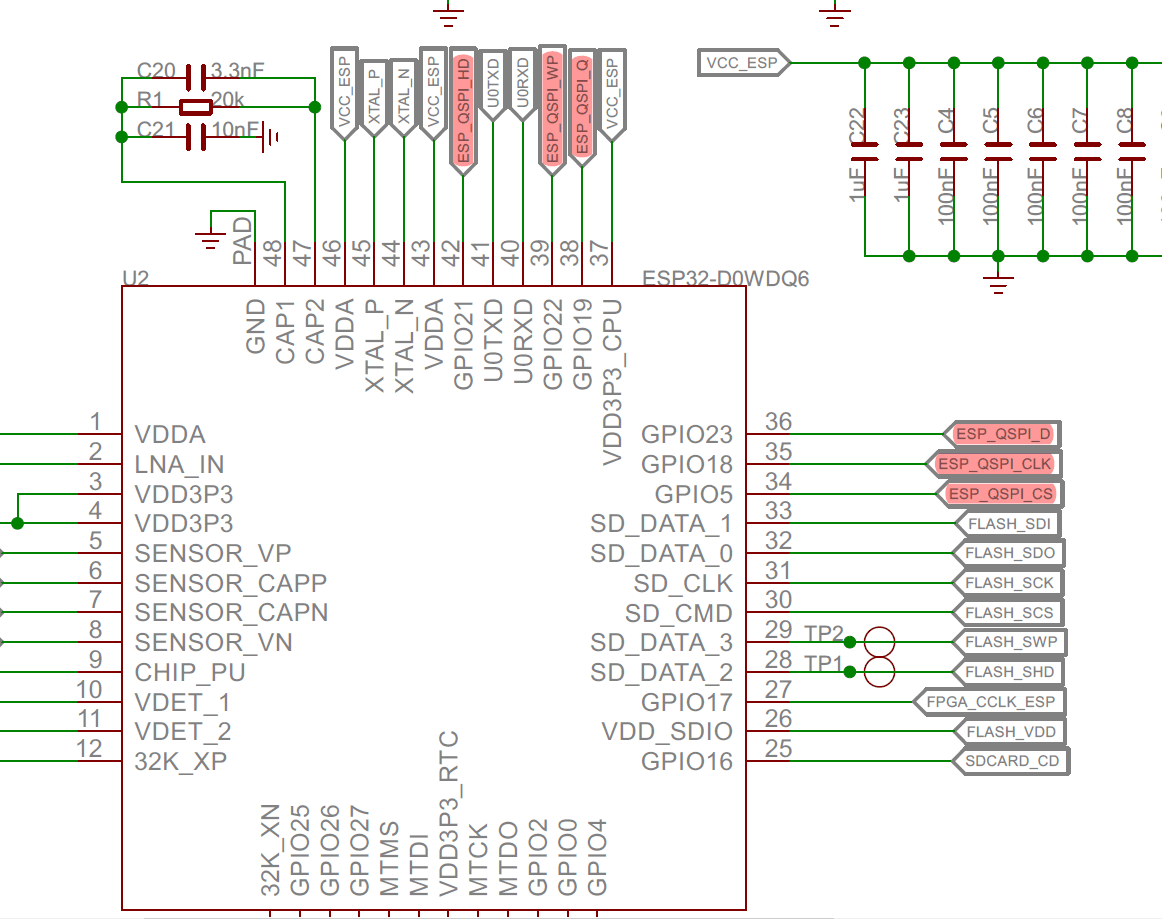


图 3 ESP32与QSPI通信相关的部分管脚原理图

图 4为ESP32连接FPGA时对应的管脚图，ESP与FPGA两端分别与一个电阻相连，以此来完成整个硬件电路的设计。

1.3 QSPI通信数据处理

**1.3.1 ESP32侧数据发送**

我们选择开发ESP32的软件为Arduino，本来打算利用MicroPython来做开发的，因为MicroPython的开发是逐语句的、实时的。但是SEA board公司提供的QSPI通讯库只提供了Arduino版本的，所以我从github上下载了QSPI通信的库[[1]](#footnote-1),加载到Ardunio中。主要的函数有以下两个：ESP32向存储器写数据的函数和ESP32向存储器读数据的函数，主要参数有两个部分是地址和数据。

**bool** sea\_esp32\_qspi::write(**uint16\_t** addr, **char** data[], **uint16\_t** length)

{

    qspiWritePolling(&this->qspi, addr, WR\_CMD,(**uint8\_t** \*)data, length);

}

**bool** sea\_esp32\_qspi::read(**uint16\_t** addr, **char** data[], **uint16\_t** length)

{

    qspiReadPolling(&this->qspi, addr, RD\_CMD, (**uint8\_t** \*)data, length);

}

**1.3.2 FPGA侧数据处理**

由于FPGA需要处理的数据是六条信号线上的生数据，我们自行编写状态机来分时处理不同类型的数据。由图 5所示，count为对SCLK上升沿的计数器，根据count的不同，在不同的时序阶段处理对应的数据。



图 5 FPGA处理QSPI数据的时序状态机

第二章 FPGA内嵌浮点数计算单元FPU

1.1 表示浮点数计算的方法[2]

在传递的数据中，共分为三个部分。第一部分为8位操作符，第二部分为两个16位数据。前八位用来表示4个不同的运算，分别用00000001来表示+，用00000010表示-，用00000100表示\*，用00001000表示/。之后的前16位里前八位用来表示浮点数的整数部分，后八位用来表示浮点数的小数部分。以此类推，输入浮点数据。

为了定点到浮点的转换，定点表示为有符号整数类型，这与Xilinx System Generator使用的数据类型一致。 定点值使用加权的二进制补码表示为以2的固定幂形式。定点数的二进制表示形式包含三个字段，如图5所示（尽管它仍然是加权的二进制补码）。

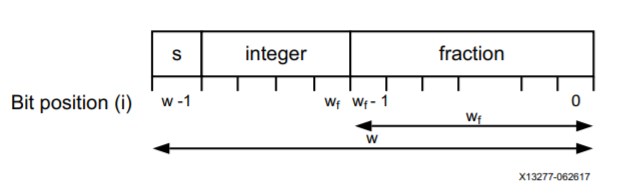


图 6 定点表示中的位字段

在图 6中，位的位置已标记为索引i。 基于此，定点数的值由下式给出：

例如，当指定总宽度为32和小数宽度为0时，将获得32位带符号整数表示形式。 在转换操作中采用舍入到最接近。 要提供符号位，整数字段的宽度必须至少为1，要求小数宽度不大于w-1。固定浮点运算符还可以选择执行32位和64位运算。位有符号和无符号整数转换，以将标准软件整数数据格式转换为浮点数。

1.2 判断并执行运算

先判断要进行的运算，后运用相对应的IPcore来执行接下来要进行的运算。

**1.2.1浮点数的加减运算**

浮点加法运算的实现包括以下几个步骤符号判断: 对阶、尾数加减操作、规格化、舍入操作、溢出判断。具体实现时通常把规格化、舍入操作、溢出判断作为一个步骤实现。浮点数的格式显然可以分为两部分, 即符号和数据的绝对值。若符号相同则符号不便绝对值相加; 若符号不同则须比较两绝对值的大小然后两绝对值作差运算。符号不同时首先判断和的符号, 显然若两浮点数的阶不同时和的符号当与阶数大的操作数相同; 若阶数不同则继续比较对阶操作首先比较两浮点数的阶数大小。然后需要两数对阶。对阶的原则是小阶对大阶，小阶对大阶的好处是，当小阶不同于大阶时，只需要移除小阶数的尾数部分的低位部分，加法流程图见图 7。

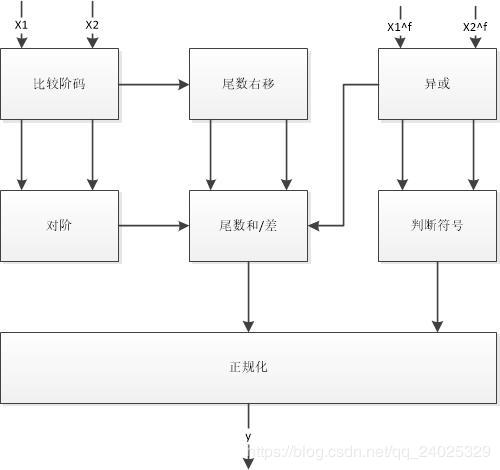


图 7 加/减运算流程

**1.2.2浮点数的乘法运算**

浮点数的乘法运算相对比较简单，只需要将两个操作数的符号位进行异或运算，再将阶码部分做和、尾数部分做积即可。同时需要检查操作数的运算结果是否有溢出问题。乘法流程见图 8:

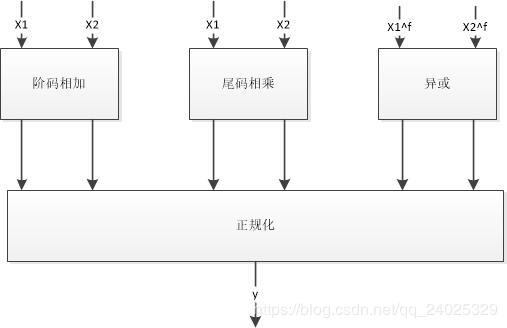


图 8 乘/除运算流程

参考文献

1. Seeed Studio, “Spartan Edge Accelerator Board v1.0”, Aug. 2019
2. XILINX, “Floating-Point Operator v7.1 LogiCore IP Product Guide”, Nov. 2019

**第三部分**

完成情况及性能参数 /Final Design & Performance Parameters

（作品已实现的功能及性能指标）

实现功能

1.ESP32与FPGA通信传递操作数和指令；

2.FPGA进行浮点数加减乘除。

性能参数如下表所示

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Total Power | LUT | FF | BRAMs | IO | BUFG |
|  | 113 | 191 | 0.00 | 8 | 3 |
| 2.706 | 112 | 191 | 0.00 | 8 | 3 |
|  | 0 | 0 | 0.50 | 0 | 0 |

**第四部分**

总结 /Conclusions

（谈一谈完成暑期学校课程后的收获与感想。请每位组员分开写。）

22017102 韩辉珺

我在这次项目开发的过程中，感觉自己有很多不足。譬如说我不会怎么调试自己写的VHDL，不像调试C++那样，这样不会调试的后果是在写大型项目的时候纠错遇到了很多问题。在综合的时候，我也遭遇多次报错，要求我加各种限制，作为一个小白我不太明白为什么要加这些限制，但加了之后就可以生成bitstream文件了，虽然做出来，但我还是觉得我只是停留在了皮毛。

22017113 陆佳晨

这次的课程与项目设计是全程在线下进行的，体验非常特别，对学校的良苦用心深感佩服。在课程的学习中我从一个更新的角度认识了FPGA。而且在家调试平台做实验也是很全新的体验，很多资料中说明比较模糊的地方要通过与同学交流或者上网再查资料来解决，虽然有一些麻烦，但相对还是比较有趣的。这次的项目让我对ESP32与FPGA的联动有了更多的认识。总而言之这半个月的折腾姑且还算收获颇丰。

1. <https://github.com/CutClassH/sea_esp32_qspi> [↑](#footnote-ref-1)