

第七届

全国大学生集成电路创新创业大赛

报告类型\*： 设计报告

参赛杯赛\*： 芯来RISC-V杯

作品名称\*： 基于E203的心音分类系统设计

队伍编号\*： CICC5996

团队名称\*： 我想静静

目录

[1 概述 3](#_Toc136468290)

[2 需求分析 3](#_Toc136468291)

[2.1项目要求 3](#_Toc136468292)

[2.2项目需求 4](#_Toc136468293)

[3 总体设计方案 4](#_Toc136468294)

[3.1 硬件平台 5](#_Toc136468295)

[3.2 蜂鸟E203 SoC概述 6](#_Toc136468296)

[3.3 系统设计简介 7](#_Toc136468297)

[4 硬件分系统设计方案 8](#_Toc136468298)

[4.1 乘法优化 8](#_Toc136468299)

[4.2 FPU浮点指令集扩展 8](#_Toc136468300)

[4.2.1 功能描述 8](#_Toc136468301)

[4.2.2 接口定义 9](#_Toc136468302)

[4.2.3 本系统在全局的位置 9](#_Toc136468303)

[4.2.4 仿真 11](#_Toc136468304)

[4.3 NICE核的硬件应用及扩展 11](#_Toc136468305)

[4.3.1 功能描述 11](#_Toc136468306)

[4.3.2 模块介绍 12](#_Toc136468307)

[4.3.3算法设计与实现 12](#_Toc136468308)

[4.3.4 行为仿真 13](#_Toc136468309)

[4.4 NICE扩展的软件库实现 14](#_Toc136468310)

[4.4.1 功能描述 15](#_Toc136468311)

[4.4.5 算法设计与实现框图 15](#_Toc136468312)

[5 软件系统设计方案 15](#_Toc136468313)

[6 系统性能与测试 15](#_Toc136468314)

[6.1 经优化后benchmark测试 15](#_Toc136468315)

[7 进一步工作 18](#_Toc136468316)

[参考文献 18](#_Toc136468317)

# 1 概述

本设计完成了基于Nuclei MCU200T开发平台和蜂鸟E203开源CPU的一个低成本低功耗的心音分类的解决方案，展现了一个在成本和资源受限的情况下，如何完成较复杂的声音识别任务的思路与具体方案。我们通过python对心音的波形文件进行预处理和生成频谱二阶谱图，并利用二分类训练模型对图像进行二分类训练。最后将模型参数保存到数组中，然后通过串口打印，同时我们也在协处理器上实现了卷积神经网络加速器来对心音图像进行加速分类运算。

硬件设计中，我们在原有E203 CPU core的基础上，针对E203内核微架构的乘除法模块进行了一定的优化，并对E203 core进行了单精度浮点指令集（F）的扩展。同时，我们运用协处理器接口，对输入数据进行卷积，激活（Tanh函数），池化（去平均值），再卷积等运算，完成对处理器运算算子的加速。

软件设计中，我们将收集到的心音信号图像数组加载到协处理器中，并将权重，偏置参数导入协处理器，然后启动硬件卷积神经网络模块，最终取出结果，再通过I2C接口驱动OLED屏将结果显示（OLED屏未完成验证）。

整个应用系统，针对心音进行识别分类，极大降低了人工的工作量，提升了判断准确率，在最终的性能测试中，我们针对200个正常或异常心意信号进行二分类识别，分类结果全部正确。

# 2 需求分析

## 2.1项目要求

经过总结，题目的要求为以下几点：

1. 对蜂鸟E203 RISC-V内核的微架构实现进行一定的优化，并在Benchmark基准测试进行系统性能测试时，蜂鸟SoC测试分数展现出一定提升；
2. 通过NICE协处理器接口或直接实现RISC-V指令子集来实现对蜂鸟E203 RISC-V内核运算算子的扩展；
3. 增加I2C接口，对得到的数据通过LED屏显示；

通过对题目的总结，我们对项目需求做了以下分析：

## 2.2项目需求

1. 开源CPU IP核：确定一个risc-v IP核是整个方案设计的首要问题。根据题目要求，CPU 应具有如下特点：开源协议允许二次开发；有统一的片上总线；面积和资源占用较小；支持GDB交互调试；最好有集成的SoC；最好是verilog语言实现，具有好的可读性。
2. 处理器微架构及其优化方向：工程师在设计处理器时，按照指令集规定的指令，设计具体的译码和运算电路来支持这些指令的运行；指令集在CPU处理器内部的具体硬件电路的实现，我们就称为微架构。在设计一个微架构时，一般需要考虑很多问题：处理器是否支持分支预测，单发射还是多发射，顺序执行还是乱序执行，流水线需要多少级，主频需要多高，Cache需要多大，需要几级Cache。根据不同的配置选项如高性能、低功耗等不同的场景，基于一套指令集设计出不同的微架构。
3. 硬件平台：逻辑资源足够实现CPU和配套外设；Jtag接口（比特流文件和软件下载）；包括多个外接的IO接口（外接显示器和逻辑分析仪）；成本要足够的小。
4. 音频处理算法硬件加速电路：从性能角度考虑，我们希望能够通过增加外设或协处理器的方式来增加音频识别处理的速度。
5. 以SDK为基础，建立软件基础工程。
6. 一些正常或异常的心音文件：我们需要一些正常或异常的心音文件来作为本次设计成功与否的判断作证，并利用这些心音文件来训练我们的算法模型。
7. 一个完整的卷积神经网络加速模块：卷积神经网络加速模块将用以加速卷积算法，快速处理心音信号，完成对心音信号的分类识别。卷积神经网络由卷积层、激活层及池化层组成。卷积层使用卷积核对特征图进行卷积操作以提取图像特征，池化层对特征图进行降采样以减少数据量。
8. OLED显示器（没做完）：通过i2c来驱动OLED显示屏以显示分类结果。

# 3 总体设计方案

针对上面的需求分析，我们最终选择蜂鸟 E203 作为我们CPU 内核，一个以功耗低资源占用少为优势的两级流水处理器；Nuclei MCU200T作为我们的硬件开发平台，该平台是芯来科技设计的低成本RISC-V开发平台，作为蜂鸟E203的配套开发板，芯来科技对该开发板做了相当完整的支持。我们针对E203内核进行了优化，旨在提高处理器性能，并对其支持的指令集做了扩展，对NICE core进行开发，以满足心音识别任务的硬件需求。

软件方面，我们在心音数据的基础上，运用函数将该数组，卷积核，权重，偏执参数等加载到协处理器，再运用函数激活以及启动卷积，最终取出结果。并且，为降低处理器工作量，提升处理器运行速度，我们设计了极少的几条协处理器指令，软件层面完成了对这几条自定义指令的编译支持。

## 3.1 硬件平台

Nuclei MCU200T是芯来科技的一款低成本FPGA开发平台，实现了“一板两用”（作为S0C原型板/作为常规FPGA开发板）。该开发平台使用Xilinx XC7A200T-2 FPGA芯片，并含有丰富的硬件资源。

板上提供SPI Flash芯片，用于FPGA存储比特流文件（MCS格式）；开发板提供两种JTAG接口：FPGA\_JTAG接口，用于连接USB-JTAG编程线缆，用于FPGA的Bitstream编程；MCU\_JTAG接口，这是RISC-V Core调试的板载JTAG接口。并提供LED，Push Buttons，XADC port，CAN port，SD卡槽，GPIO等资源。

整体的硬件连接，FPGA\_JTAG接口用于下载比特流文件，MCU\_JTAG接口用于RISC-V Core调试通信，检测内核对于心音数据的处理结果。

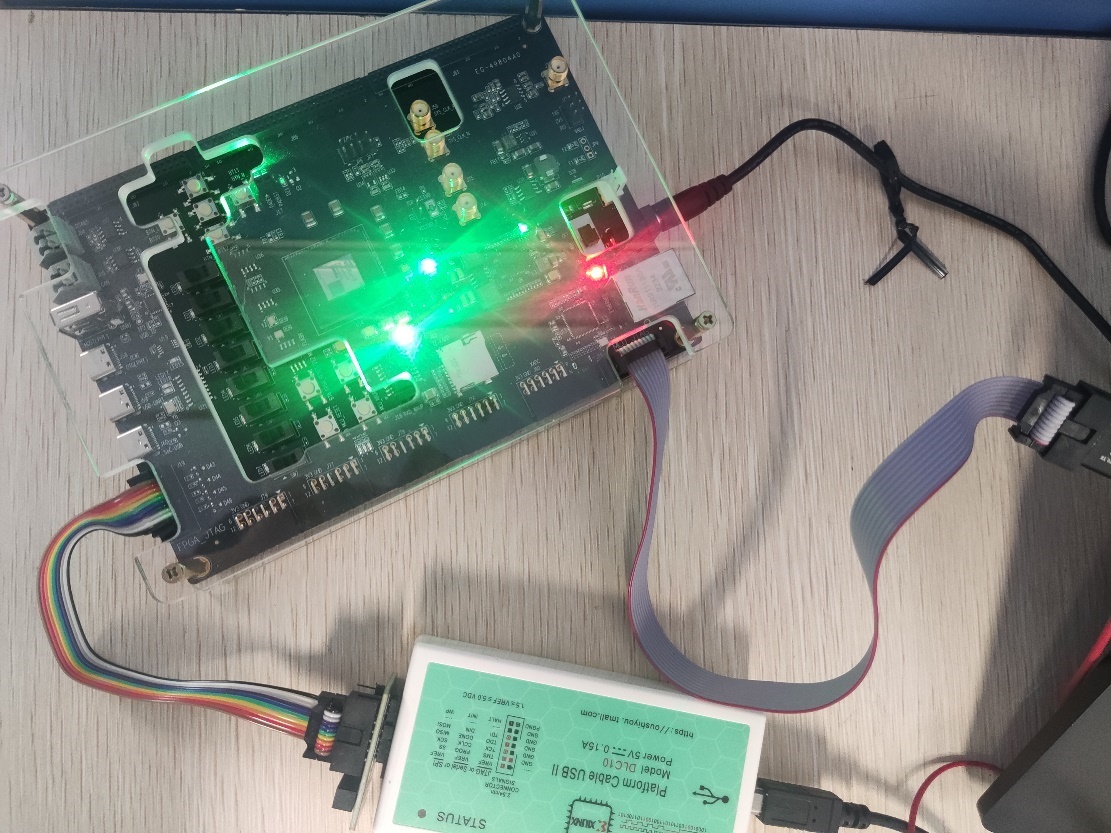


Figure 3：项目整体硬件连接

## 3.2 蜂鸟E203 SoC概述

蜂鸟E203 主要面向极低功耗与极小面积的场景，是一款对标Cortex-M0或M0+的处理器。处理器核采用两级流水线结构；支持RV32I/A/M/C指令子集的配置组合，支持机器模式；提供标准的JTAG调试接口；私有的ITCM和DTCM，实现指令与数据的分离存储。



Figure 4：E203 流水线结构

E203处理器配套SoC提供紧耦合系统IP模块，包括中断控制器，计时器，UART，QSPI和PWM等；中断接口通过PLIC与处理器核相连；提供ICB系统总线用于访存内存数据和访问外设；提供调试接口用于SoC级别的JTAG调试。

## 3.3 系统设计简介

根据需求分析，我们在E203源码的基础上，对E203 RISC-V内核的微架构进行了一定的优化，对乘除法模块中乘法部分进行了简单优化，使得原有乘法实现由17个周期减少为9个周期。

对蜂鸟E203 RISC-V内核进行了RISC-V指令子集——单精度浮点（F）指令集的扩展。本设计采用Berkeley SoftFloat浮点运算模块，符合IEEE浮点运算标准，支持加减乘除，平方根，格式转换等浮点运算，我们需要自己设计浮点译码，浮点寄存器堆，浮点csr等单元，完成FPU的控制逻辑，将该模块接入到E203 RISC-V core中。

对蜂鸟E203 RISC-V内核通过NICE协处理器进行运算算子的扩展，通过设计扩展自定义指令，实现对卷积神经网络的加速计算。

之所以选用卷积神经网络，是因为卷积神经网络一般用于图像处理，在进行处理过程中，会自动进行分类处理，使得结构十分简化。我们在实现卷积神经网络中卷积操作主要工作步骤包括：输入特征图、多次卷积和池化循环操作、激活函数操作、全连接以及输出特征。卷积神经网络操作主要是卷积层，激活层，池化层以及反向传播过程。

下图为E203 core示意图，其中红框部分为在原有内核基础上增加和修改的部分。

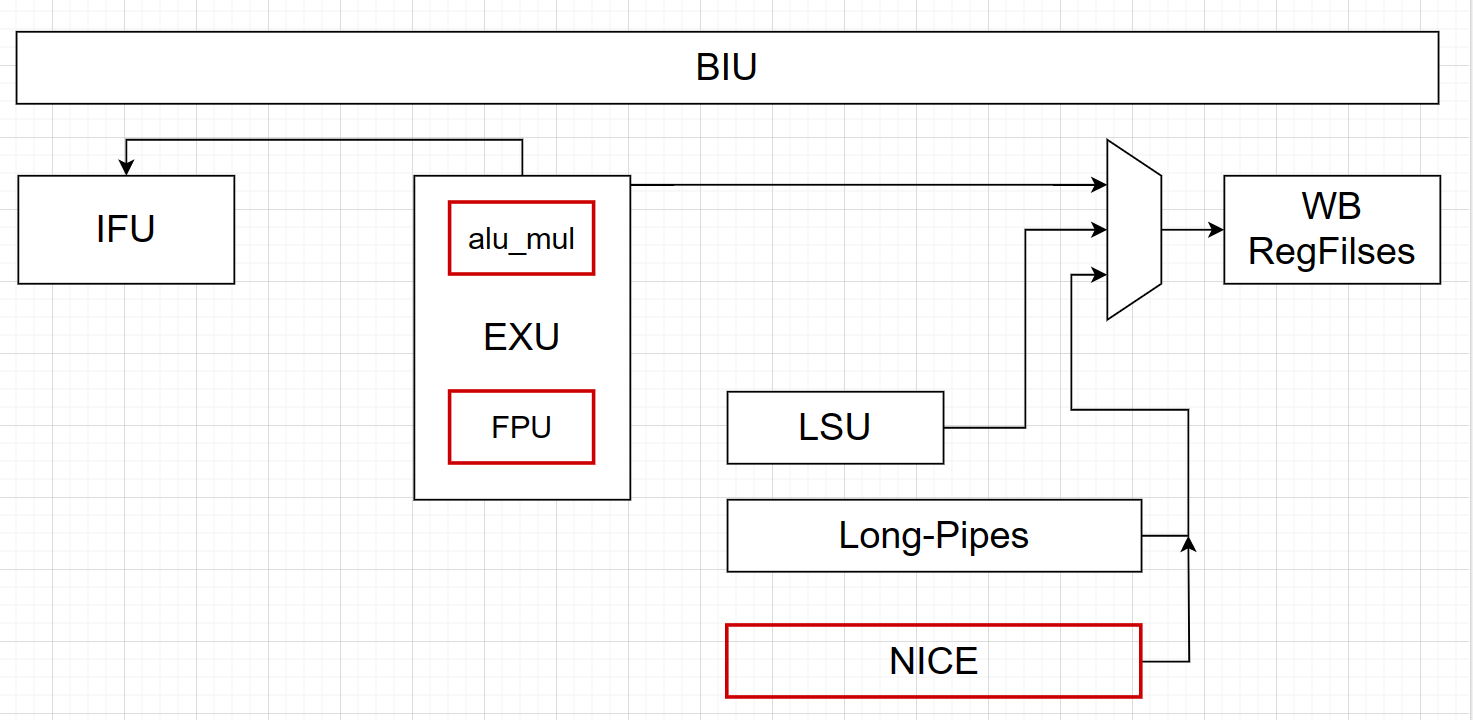


Figure 5：E203 core

# 4 硬件分系统设计方案

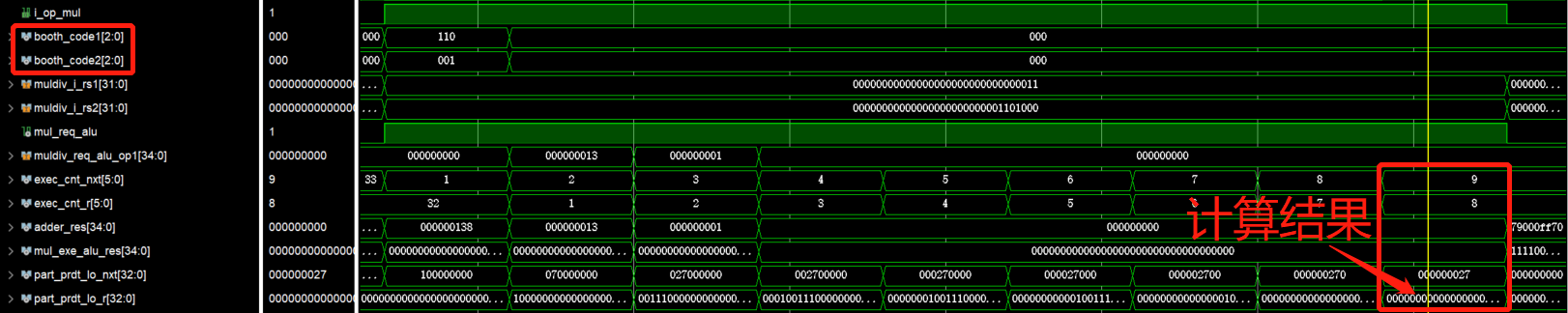
根据需求分析，我们针对E203 RISC-V core的乘除法模块，FPU，NICE协处理器进行了不同程度的优化或扩展。

## 4.1 乘法优化

在E203 core原乘除法模块中，采用Booth4乘法器，对于有符号整数乘法操作，可以使用常用的 Booth 编码产生部分积，然后使用迭代的方法，每个周期使用加法器对部分积进行累加，经过多个周期的迭代之后得到最终的乘积。针对32位乘法，每次生成一个booth code，计算两位，并且对于无符号乘法进行一位符号扩展后统一当作有符号数进行运算，因此需要17个迭代周期。

在对乘法进行优化时，曾考虑更换其他乘法实现方式，如LUT乘法器，Wallace树乘法器，移位相加乘法器等。但考虑到在资源消耗与运行速度两方面的平衡，在优化中并未选择这些乘法器。如Wallace树乘法器，仅仅两个4位的数相乘，就需要3个全加器与3个半加器的支持，而随着两个数位数的增加，所需要的加法器将会呈现指数级增长，针对32位乘法，将需要大量半加器与全加器的资源消耗，以支持Wallace乘法器优秀的运行速度，且很难使用到E203通用的计算通路。

本设计最终决定继续采用Booth4乘法器，但将在每个周期计算两个booth code，即每次计算4位，对于无符号乘法进行一位符号扩展后统一当作有符号数进行运算，共需要9个迭代周期来完成乘法的计算，在E203计算通路中，将两个的加法计算变成3个，其中最后一个加数的计算将只在乘法计算时生效。在完成工作内容后，对其进行模拟仿真，结果如下：



在上图中左侧框图中可以看到，每次计算时钟将生成两个booth code，即一次计算可以完成4位，并最终在第九个周期时，完成32位数的计算（图中右侧红色框图中）。

完成优化后，Benchmark基准测试进行系统性能测试时，蜂鸟E203 SoC测试分数展现出一定提升。

## 4.2 FPU浮点指令集扩展

### 4.2.1 功能描述

该FPU项目由Berkeley雇佣监督开发，该项目是二进制浮点的硬件实现，它符合IEEE浮点运算的标准，支持多种浮点格式，使用parameters模块独立确定指数的宽度和有效字段。可选的格式包括：16位半精度、32位单精度、64位双精度和128位四精度的标准格式（本次项目仅使用其32位单精度格式）。支持加减法、乘法、融合乘加、除法和平方根（迭代实现）、比较、与其他浮点格式的转换、与整数、有符号数和无符号的转换等。

### 4.2.2 接口定义

该浮点单元包括多个子单元，此处只介绍接入本项目的顶层接口定义。包括三个浮点数运算操作数，浮点运算类型，舍入模式，浮点计算结果，浮点数运算异常等。详细接口信息如下：

Table 3：FPU模块IO接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| Name | I/O | Bits | Description |
| frs1 | I | 64 | 浮点数运算第一个操作数 |
| frs2 | I | 64 | 浮点数运算第二个操作数 |
| frs3 | I | 64 | 浮点数运算第三个操作数 |
| ftype | I | 5 | 具体的浮点操作指令类型 |
| fcontrol | I | 1 | 浮点舍入控制 |
| rounfingmode | I | 3 | 舍入模式 |
| fmt | I | 2 | 浮点运算格式，single/double float |
| farithematic\_res | O | 64 | 浮点运算结果 |
| exception\_flags | O | 5 | 浮点数运算过程中出现的异常 |
| fflags\_valid | O | 1 | 异常输出有效 |

### 4.2.3 本系统在全局的位置

本设计在E203内核的EXU中，嵌入了该FPU单元。

首先在E203内核EXU模块中实例化该FPU单元，但该FPU单元仅为其具体浮点执行模块，我们需要在该浮点单元基础上，添加浮点译码模块，浮点寄存器堆，并在EXU\_CSR中加入浮点控制与状态寄存器的内容（csr），使得浮点指令与整数指令等相同，能够具有完整的取值-译码-执行-访存-写回。

首先设计浮点译码模块，并将其连接到EXU\_DECODE模块中，如下图所示。

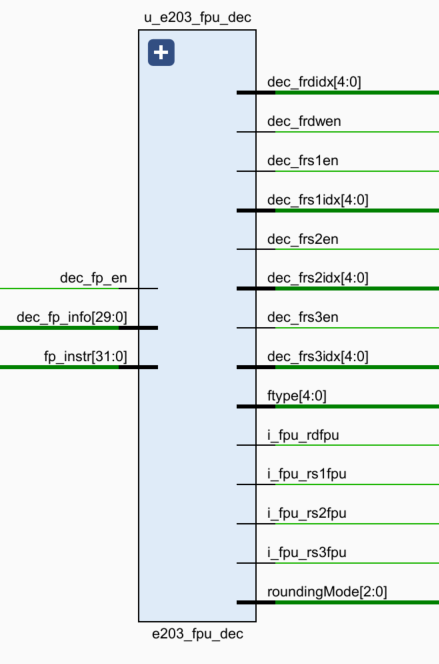


Figure 7：浮点译码单元

之后需要在IFU模块中，在预译码时，译码出FPU指令使能信号（dec\_fp\_en），指令源寄存器使能信号（dec\_frs1en，dec\_frs2en，dec\_frs3en），源寄存器索引（dec\_frs1dx，dec\_frs2dx，dec\_frs3dx）以及源寄存器来源是整形寄存器堆或浮点寄存器堆（i\_fpu\_rs1fpu，i\_fpu\_rs2fpu，i\_fpu\_rs3fpu）。

完成译码部分的工作后，需要在EXU模块中加入浮点寄存器堆（fpu\_regfile），如下图所示，浮点寄存器堆的verilog写法参照E203内核源码的整形寄存器堆写法。

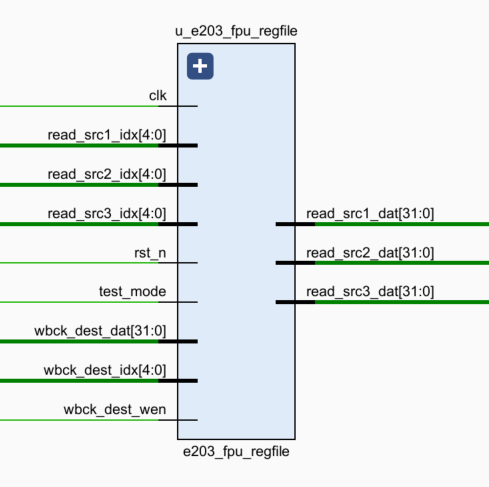


Figure 8：浮点寄存器堆

最终在EXU单元中，加入FPU模块的写回控制（EXU\_WBCK，EXU\_OITF，EXU\_DISP等中）以及FPU csr即可完成整个FPU单元的取值-译码-执行-访存-写回过程（注：至截止日，本设计中浮点单元存在某些写回异常情况，后续需对其进行更正）。

### 4.2.4 仿真

我们针对FPU模块，搭建了该模块的仿真平台，并对不同的功能进行了行为仿真。此处将以两个例子来展示。

例一：单精度浮点数的乘加取反指令（fnmadd rd,rs1,rs2,rs3），表示将浮点数rs1与浮点数rs2相乘，再将其加rs3的相加结果写入目的寄存器rd。

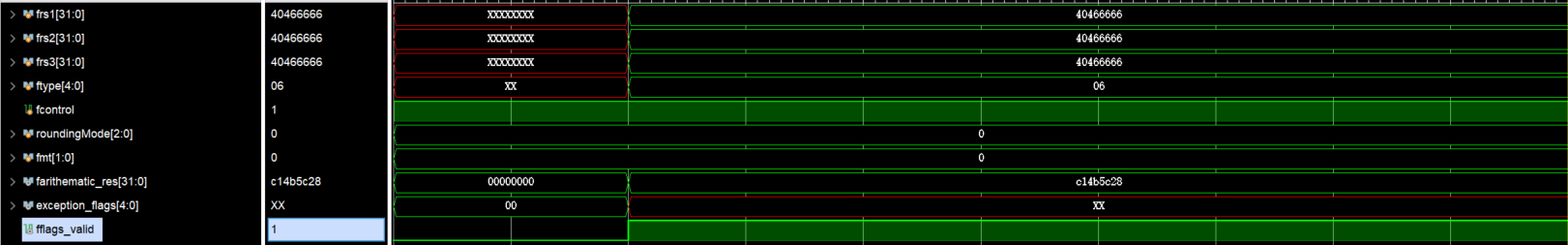


Figure 9：fnmadd指令行为仿真

图中可以看到浮点数frs1，frs2，frs3分别为32位二进制40466666，即表示十进制3.1，执行rd=-（rs1\*rs2+rs3）之后的结果为-12.72，即32位二进制表示为c14b5c28。

例二：单精度浮点数向字节转换指令（fcvt.w.s rd,rs1），表示将浮点数rs1转换为32位二进制补码表示的整数，再将结果写入目的寄存器rd。

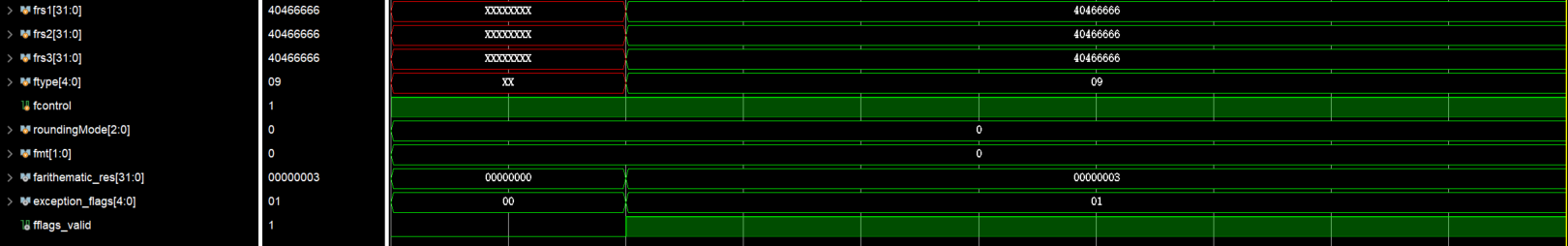


Figure 10：Fcvt.w.s指令行为仿真

图中可以看到浮点数frs1为 32位二进制40466666，即表示十进制3.1，经过格式转换之后的结果3。

## 4.3 NICE核的硬件应用及扩展

### 4.3.1 功能描述

通过NICE协处理器接口的添加，实现对蜂鸟E203 RISC-V内核进行运算算子的扩展，以配合本项目所需的卷积神经网络，提升运算速度，更好完成大量心音数据的分类。

我们设计的卷积神经网络协处理器由NICE接口，特征图存储器，权重存储器，激活模块，池化模块，结果存储器，加法树组成。

### 4.3.2 模块介绍

Table 5：EMAC 模块IO接口定义

|  |  |
| --- | --- |
| Name | Description |
| NICE接收模块 | 接收来自E203处理器的指令和数据 |
| 特征图存储器 | 存放NICE接口收到的待处理特征图，9个深度为7的32位存储器 |
| 权重存储器 | 存放卷积操作的卷积核参数，5个深度为5的32位存储器 |
| 激活模块 | 为卷积神经网络引入非线性 |
| 池化模块 | 对特征图进行压缩，去除冗余信息，简化网络复杂度，减小计算量，提高计算速度等 |
| 结果存储器 | 5个深度为7宽度为32的存储器 |
| 加法树模块 | 对卷积神经网络进行加速计算 |

### 4.3.3算法设计与实现

我们在E203的基础上，通过独立于E203的NICE 接口将协处理器与E203内核相连，实现在E203接收到神经网络指令时将指令传输给协处理器完成，在协处理器完成指令后返回E203内核。

卷积神经网络由卷积层、激活层及池化层组成。卷积层使用卷积核对特征图进行卷积操作以提取图像特征，池化层对特征图进行降采样以减少数据量。卷积神经网络涉及多种数学运算，例如：乘法、加法、除法、数值比较、指数等，需要用到大量的数据。如果使用RISC-V基本指令以及标准扩展指令来完成卷积神经网络的计算，需要用到多种类型的指令且需要无数次访存以读写数据。为降低工作量，提升运行速度，我们设计了极少的几条协处理器指令，如下表所示。

|  |  |
| --- | --- |
| Name | Description |
| JSLD | 将需要进行卷积操作的原图像数据加载到处理器内部 |
| FCONV | 将第一卷积层所需的卷积核数据加载到协处理器内部 |
| SCONV（待使用） | 将第二卷积层的卷积核参数加载到协处理器内部 |
| JACT | 启动协处理器完成单次激活函数操作 |
| JCONV | 启动协处理器完成卷积操作 |

本文所设计指令很少，但却可以完成整个卷积神经网络加速计算，这可以减少数据在E203处理器和协处理器之间的流动，从而减少访存功耗，尤其是指令JCONV，一条指令可以完成两层卷积计算、激活操作函数以及池化操作，极大地减少了访存。

### 4.3.4 行为仿真

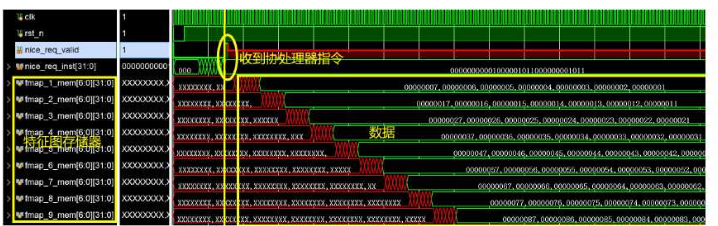
首先对指令JLSD进行仿真分析，指令JLSD主要负责将需要进行卷积操作的原图像数据从E203处理器的存储器搬移至协处理器内部的特征图存储器中。从开始再执行该指令前须在DTCM中存储待处理的图像数据（最大9\*7）并定义其起始地址（0x90000000）,我们通过Nuclei Studio 预编译生成点Verilog文件对该指令进行仿真，仿真图如下。f\_map\_1~9\_mem为协处理器的特征图存储器的子存储器，当收到协处理器指令JLSD后，数据连续不断的转移至特征图存储器当中。且特征图存储器数据与我们c文件定义的数组数据相同，因此该指令准确无误的完成了待处理特征图的搬移，协处理器完成了该指令。

Figure 10：JLSD指令仿真

接下来我们对FCONV进行仿真，仿真过程和上面一致，仿真结果图如下

Figure 10： FCONV指令仿真

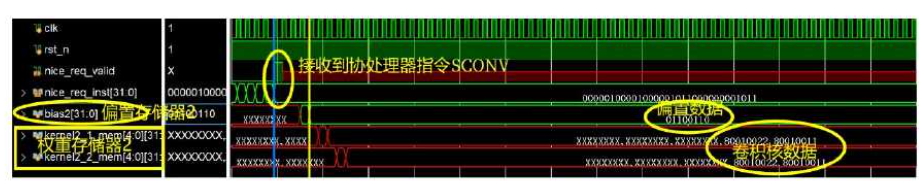
SCONV指令也与上面一致，仿真结果图如下

Figure 10： SCONV指令仿真

JACT指令验证，JACT指令因为用到了浮点数，我们编写了五条指令带来验证0001100\_00001\_00000\_000\_00001\_0010011,0000000\_00101\_00000\_000\_11111\_0010011,

0000000\_11111\_00001\_110\_00010\_0011011,0000110\_00001\_00001\_110\_00010\_1111011.

0000110\_00010\_00001\_110\_00010\_1111011

第一条指令将通用寄存器0中的数据与立即数0b110000001 相加并把结果存于通用寄存器1，执行过后通用寄存器1数值为0b110000001。第二条代码将通用寄存器0数据与0b101相加，执行过后通用寄存器1数值为0b101.第三条代码将寄存器1中的数据逻辑左移0b101位，此时通用寄存器1中的数据位0x00003020,并转化为定点数值12，320\*0.0000152587890625=0.18798828125.第四条代码使用sigmoid 函数把结果写入通用寄存器二（后面没用）第五条对寄存器1的数值进行Tanh激活函数操作且把结果写回通用寄存器2.下面是仿真图

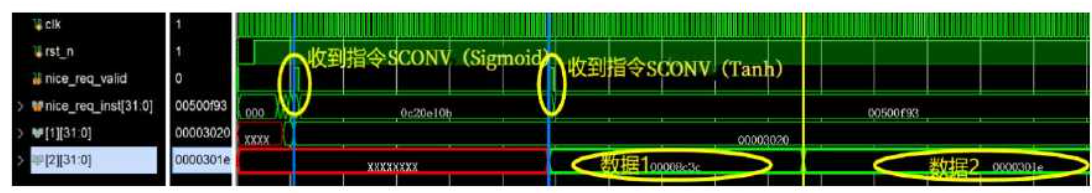


Figure 10：jact指令仿真

JCONV卷积功能验证，指令JCONV负责启动协处理器完成卷积操作，并将结果写回至E203处理器的存储器中。在执行该指令前，应把特征图，卷积核，偏置数据搬至协处理器。为验证该指令，我们同样通过.verilog文件进行仿真一共4\*4特征图卷积3\*3的卷积核，结果如图所示。

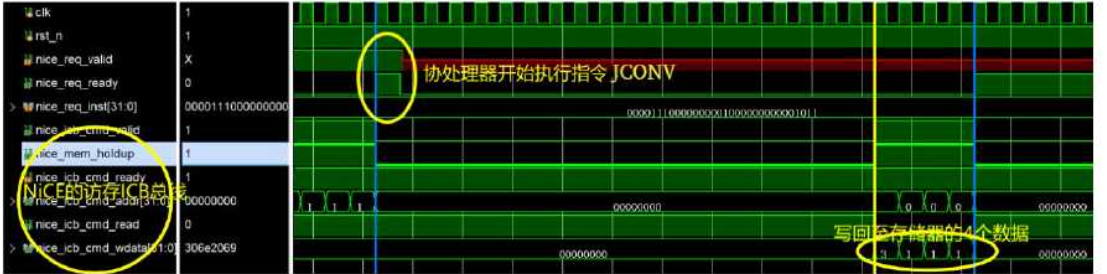


Figure 10：JCONV指令仿真

## 4.4 NICE扩展的软件库实现

### 4.4.1 功能描述

主要编写了jlsd（），fconv（），sconv（），jact（），jlweight（），jconv（）五个自定义函数。

Jlsd()用来将图像数组加载到协处理器，fconv()将卷积核加载到协处理器里，jlweight()将权重值加载到协处理器器中（包含两个偏置数），jact()启动激活函数，jconv()负责启动卷积池化全连接，归一化再输出数据。

### 4.4.5 算法设计与实现框图

加载数据->卷积->激活->卷积->全连接->归一化->输出结果，结果比对判断心音是否异常，并进行uart串口打印。

# 5 软件系统设计方案

心音音频文件通过python预处理成二阶频谱图，然后按像素（0-255区间）保存为文本文件，再通过读取txt文件到图像数组之中（目前是复制粘贴模式），再读取模型文件中的权重参数到权重数组和卷积核数组之中，然后通过协处理器的命令，对通过神经网络后的模型结果进行保存，最后通过对结果对比来uart输出结果(i2c驱动OLED显示也可以)。

# 6 系统性能与测试

## 6.1 经优化后benchmark测试

经过对微架构优化后，benchmark各项跑分有所提高：

1.coremark跑分

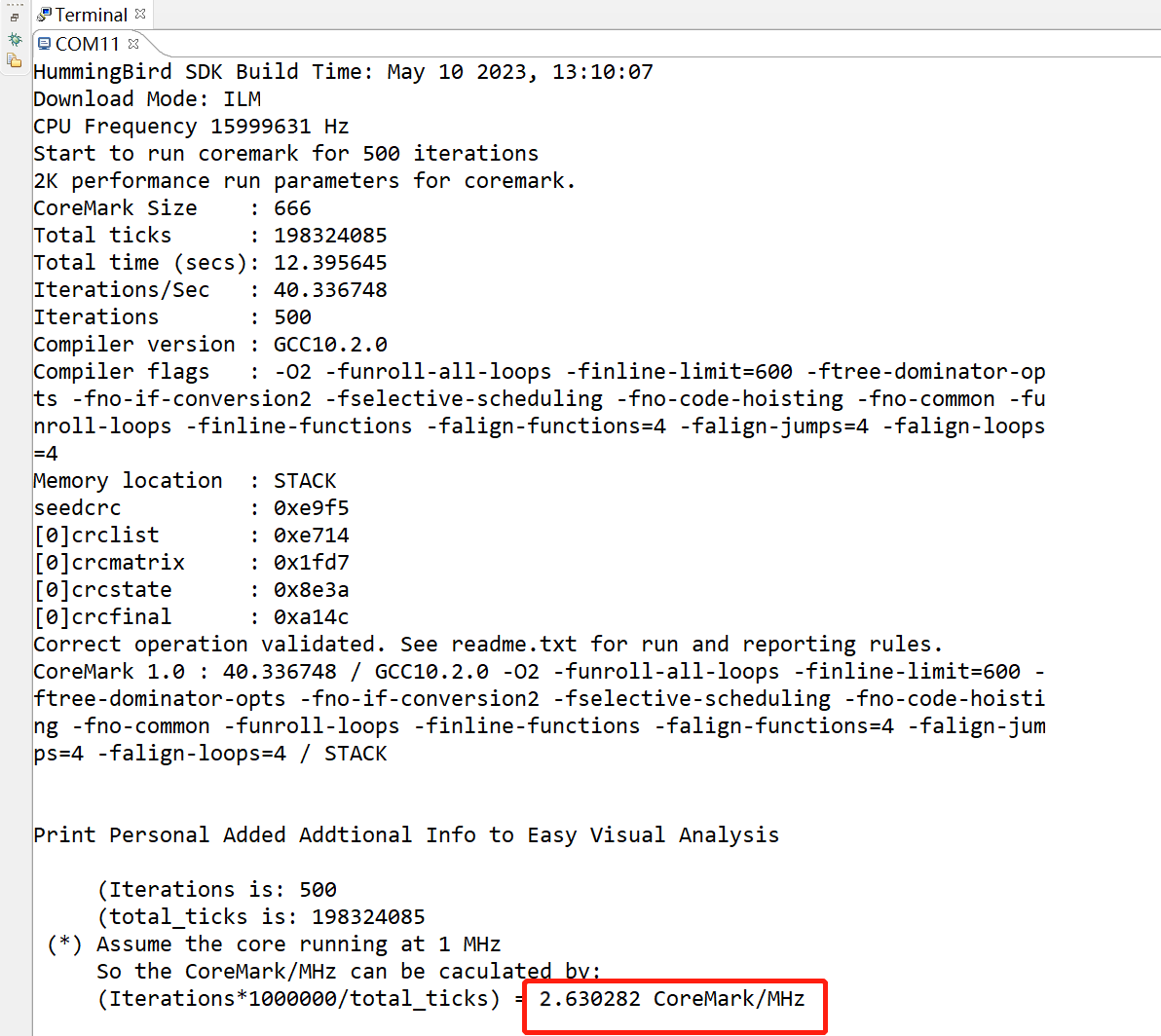


Figure 27：coremark跑分

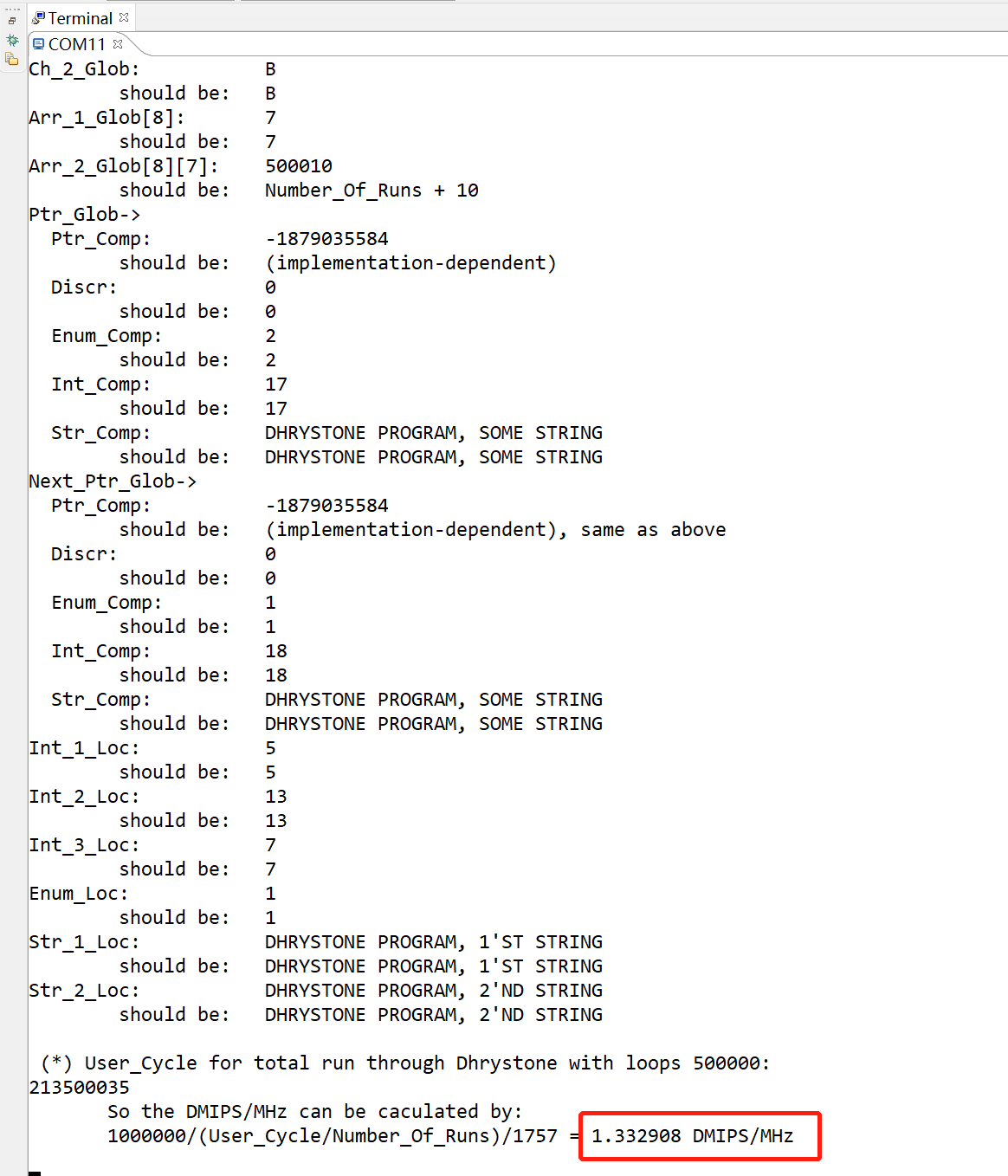
2.dhrystone跑分

Figure 28：dhrystone跑分

3.whestone跑分

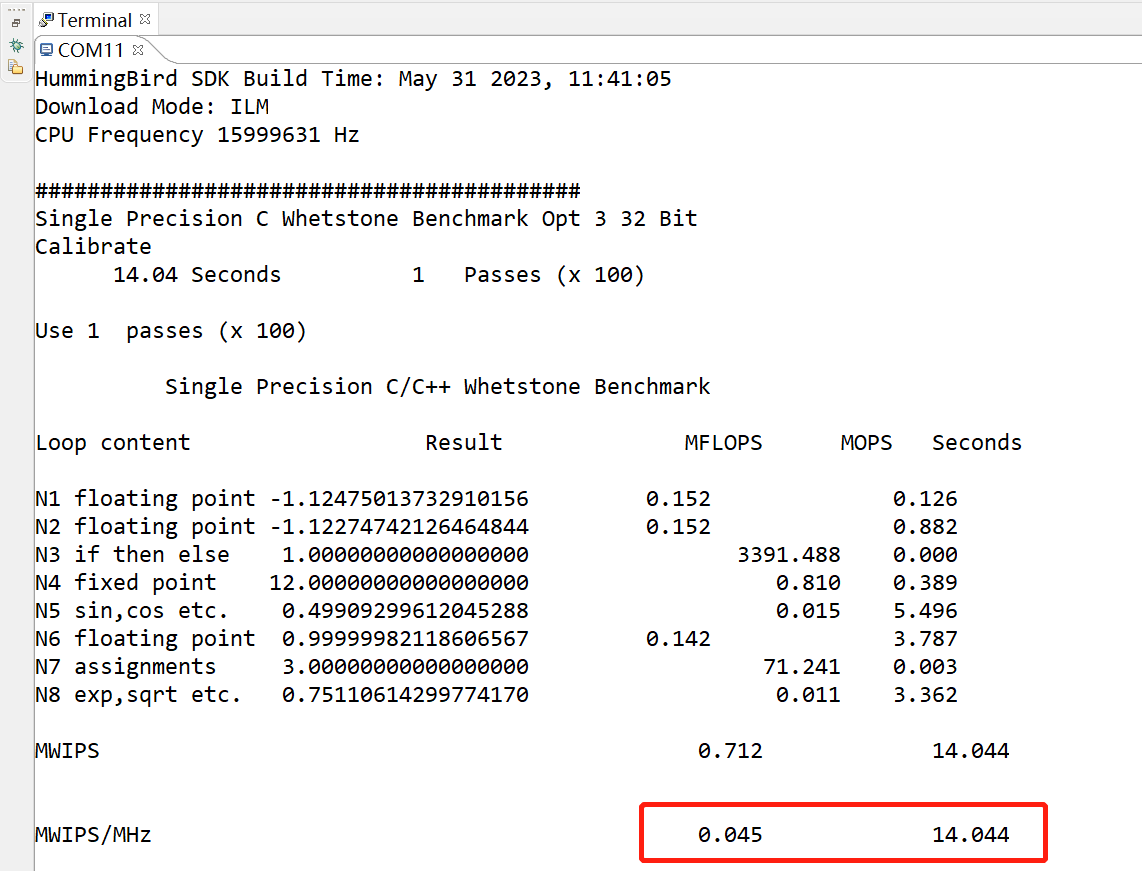


Figure 29：whestone跑分

# 7 进一步工作

进一步工作主要从四个方面开展：

1. 针对微架构优化较为单一，之后在保证低功耗小面积的前提下，可考虑从硬件除法器设计，微架构的调整，分支预测等方面进行优化。
2. 完善单精度浮点指令集的扩展，并深度开发Berkeley HardFloat单元，完成对半精度，单精度，双精度等的支持。
3. 此次设计主要以内核开发为主，并未在外设部分做开发，之后需根据该项目需求，增加外设支持，完善系统应用。
4. 本次项目设计为心音分类系统，但项目整体流程不够完善，之后将考虑从心音采集，量化，分类完善整体系统级应用，以支持完整心音分类流程，使得整个系统级应用前景更加明晰。

# 参考文献

1. 寇晓斌,杨琴,王亮亮.主流处理器体系结构与架构发展现状综述[J].微型机与应用, 2014, 33(16):1-2+5.DOI:10.19358/j.issn.1674-7720.2014.16.001.
2. Waterman A, Asanović K, Hauser J. The RISC-V instruction set manual: Volume II: Privileged architecture[R]. Si Five Inc., 2021.
3. Waterman A. The RISC-V Instruction Set Manual Volume I: Unprivileged ISA Chapter 4 “Zihintpause” Pause Hint, Version 1.0[EB/OL].[2023-03-20]. <https://www.riscv.org/wp-content/uploads/2017/05/riscv-hint-pause.pdf>.
4. SI-RISCV. 蜂鸟E203开源内核简介 [EB/OL]. [2022-03-26]. <https://github.com/SI-RISCV/e200_opensource/blob/master/doc>.
5. SI-RISCV. 蜂鸟E203开源内核 [EB/OL]. [2022-03-26]. <https://github.com/SI-RISCV/e200_opensource>.