2018 Synopsys ARC 杯电子设计竞赛

技术论文

论文题目：

iRhythm 网络音箱

参赛单位：华中科技大学

队伍名称：华智参赛队

指导老师：何顶新老师

参赛队员：李锐戈 吴曦 马志朋

完成时间：2018 年 5 月 26 日

基本情况表

<table>
<tr><td>队伍名称</td><td colspan="3">华智</td><td>单位名称</td><td colspan="2">华中科技大学</td></tr>
<tr><td>项目名称</td><td colspan="6">iRhythm 网络音箱</td></tr>
<tr><td>项目负责人</td><td colspan="3">李锐戈</td><td>联系方式</td><td colspan="2">18707169086</td></tr>
<tr><td>指导老师</td><td colspan="3">何顶新</td><td>职务</td><td colspan="2">副教授</td></tr>
<tr><td rowspan="5">参赛
队员
信息</td><td>姓名</td><td>学历</td><td colspan="2">证件号码</td><td>专业</td><td>分工情况</td></tr>
<tr><td>李锐戈</td><td>硕士</td><td colspan="2">441402199408250459</td><td>自动化</td><td></td></tr>
<tr><td>吴曦</td><td>硕士</td><td colspan="2">652122199505040110</td><td>自动化</td><td></td></tr>
<tr><td>马志朋</td><td>本科</td><td colspan="2">411527199503292531</td><td>船舶与海洋工程</td><td></td></tr>
<tr><td></td><td></td><td colspan="2"></td><td></td><td></td></tr>
<tr><td>项目时间</td><td colspan="6">2018 年 2 月 1 日 - 2018 年 6 月 10 日</td></tr>
<tr><td>队伍简介</td><td colspan="6">我们拥有熟悉算法、
软硬件设计与应用的队员们，丰富的想法，无限的创意</td></tr>
<tr><td>参与项目</td><td colspan="6">iRhythm 网络音箱</td></tr>
<tr><td>获奖情况（校级及以上）</td><td colspan="6">无</td></tr>
<tr><td>研究专长</td><td colspan="6">电路设计，嵌入式，物联网，EDA</td></tr>
<tr><td>其他</td><td colspan="6">无</td></tr>
</table>

摘　要

iRhythm 是基于 ARC EM Starter Kit 的网络音箱的设计原型。它可以连接在线音乐播放网站，模拟浏览器发送“GET”，“POST”等 HTTP 请求，从在线网络音乐播放网站的相应中抓取网站数据，分析其中的数据，从中提取获得音乐的下载链接。下载在线音乐到本地，和本地 SD 卡中的音乐文件一起，被解码和播放。

对 mp3 解码算法进行大幅度优化：使用 ARC DSP 指令加速运算，减少解码核心算法的运算时间；代码的深度优化，降低存储器读写的频率，CCM 的配合，减少存取时间。使得原本太复杂，难以在低速处理器上实现的 mp3 解码算法，在 ARC EM Starter Kit 上成功实现。

通过使用 FPGA，可以为没有音频输出接口的 ARC EM Starter Kit 扩展一个 I2S 音频输出外设：通过板载的 SPI 外设，加上外置 FPGA 扩展的 SPI 转 I2S 协议接口电路的配合，一个音频输出外设便产生了。而此时的 I2S 音频格式可以被音频 D/A 芯片直接识别，输出模拟的音频信号。

通过外部的音频功率放大器，微小的模拟音频信号被放大，音乐最终能够从扬声器中播放出来。提供更高的听觉感受！

通过板上的按键，用户可以对系统进行操控。通过操作按键，用户可以在 OLED 中查看当前播放的音乐，查看当前系统运行的性能等数据，查看播放列表，选择下一首歌曲等。

设计多个实验，对比系统在不同优化级别，不同运行状态下的运行性能状况,分析对比造成性能差异的原因；分析网络下载速度的瓶颈所在。经过实验可知，完全优化后，运算所需时间是优化前的约 43%，运算性能得到大幅度提高；在本设计中，外网速度限制了下载速度性能的发挥，如果提供更高的带宽，EMSK 使用串口控制 ESP8266 可以达到 305KB/s 的下载速度；ARC EM Starter Kit 开发板还有极大的潜能可以提供发掘。

关键词：mp3 解码；DSP 指令；优化；扩展外设

ABSTRACT

iRhythm is a prototype of Internet Radio designed on ARC EM Starter Kit. By using http command like *get*, *post*, it can connect online audition websites, crawl music information, analyze the music data, and get download link among data, download the music, decompress the file and play. The music file in SD card can also be decoded and played.

Mp3 decode algorithm has been optimized drastically. By using ARC DSP Instruction, optimized code, CCM on board, the efficiency of algorithm has been increased obviously. Though mp3 decode algorithm is too complex to complete in such a low speed core, we did make it this time!

With the help of a FPGA, the digital music data in SPI format can be converted into I2S format. It can make up the shortcoming of EMSK, which has no I2S peripheral on the Kit but a SPI peripheral. Now the I2S format music data can be converted from SPI and recognized by audio D/A convertor directly, and can finally be converted into analog signal.

Through an amplifier outside, the music can be played out from two big speakers. And it's a really good feeling!

With the keys on the Kit, users can control the system. While pressing the keys, users can view the performance data, the song playing at present, the playlist or the next song need to play through the OLED.

A number of experiment has been made to compare the performance in different optimize state, and different operate state. In this way, we can analyze the reason why the performance is so different, and where the choke point of download speed is. After that we can know mp3 decode algorithm speed has increased by 43%, because of the applying of optimizer; EMSK can reach in 305KB/s when use ESP8266 to download file in theory. And ARC EM Starter Kit has Huge potential for us to explore.

Keywords: DSP Instruction; optimized; mp3 decode; peripheral extension

目　录

第一章 方案论证

1.1 项目概述

近年来，物联网、人工智能技术快速发展，智能音箱成为各大互联网厂商争夺市场的热门产品——天猫精灵，亚马逊 echo，小爱同学等产品，让消费者应接不暇。它们的主要功能，是连接在线音乐平台——喜马拉雅 FM，蜻蜓 FM，百度 FM 等，获取音频数据，进行本地解码和播放。然而，高昂的产品价格和不成熟且有限的功能，往往限制了消费者的购买欲望。

究其价格居高不下的原因，是使用了价值不菲的硬件。当前几款主流智能音箱的处理器参数如下：

表 1-1：主流智能音箱处理器参数

小豹音响	Cortex-A7，4 核，1.5GHz
亚马逊 echo	DM3725(TI)：OMAP 3 + Cortex-A8 + DSP + 加速器
天猫精灵 M1	Coretx-A35，64 位，1.2GHZ
小爱同学	Cortex-A53，64 位，4 核，1.2GHz
小爱 mini	Cortex-A7，4 核，1.2GHz

iRhythm 网络音箱，则是基于 ARC EM Starter Kit 开发板，使用 ARC em7d 内核的无线网络音箱设计原型。它能够完整地实现抓取、下载在线网络音乐；解码 Mp3 格式的音乐；播放解压缩后的音乐文件；用户控制，OLED 显示等功能。这些功能，是当前市场上主流智能音箱最主要，最常用的功能。与此同时，iRhythm 因所需处理器的主频不高（低于 25MHz），导致成本极低，在市场竞争中具有绝佳的价格优势！

iRhythm 网络音箱，使用 Github 上针对 Cortex-M4 的 Mp3 软件解码开源的库（Walkgeek），对 Mp3 音乐文件进行软件解码。使用 DSP 指令加速，代码深

度优化，合理分配内存使用，利用 CCM 配合，任务调度规划等方法，弥补 ARC EM Starter Kit 2.3 主频只有 25MHz 的劣势，从而满足解码播放实时性的要求。

iRhythm 网络音箱，使用“GET”、“POST”两种 HTTP 方法，通过模拟浏览器访问百度 FM，蜻蜓 FM，喜马拉雅等在线音乐网站，获取分析音乐信息，得到下载链接，下载音乐提供本地解码播放。通过不断增加对在线音频网站的访问接口，iRhythm 网络音箱可以获得越来越多的在线音频资源，拥有更多的频道。

用户对系统的控制不可缺少的一环，在 iRhythm 网络音箱中，这一环也没有缺失：用户可以通过开发板上的按键和安装的 OLED 显示屏，查看性能数据，查看和控制音乐的播放。

通过外部 SPI-I2S 协议转换接口，数字-模拟转换，模拟信号放大等电路，音乐可以通过两个大扬声器播放，获得很好的听觉效果。

在项目实施后期，将尝试利用剩余的处理器资源，增加语音识别的功能，将语音识别作为用户控制又一个渠道，进一步发掘 ARC EM Starter Kit 的潜力，进一步增加 iRhythm 的功能，使我们的作品成为真正的智能音箱。

1.2 资源评估

软件需求：

（1） 网络环境

iRhythm 需要从网络上获取音频资源，因此需要连接网络，通过将自身模拟成为一个浏览器，请求网络在线音频提供商的数据，获取音频数据和信息。

（2） 音频解码

为了减小文件体积，从网络上获取的音频文件都是高度压缩的，不能直接送至音频输出接口播放。需要通过音频解码，解压缩为可以供音频数字-模拟转换器识别的 PCM 数据才能进行播放。

（3） 音频播放

为了在数字音频信号输出时减小处理器的负担，需要采用具有 DMA 的外设

进行输出，在 ARC EM Starter Kit 开发板中，可以选择 SPI 输出数字音频信号。但是音频 D/A 是无法识别 SPI 格式的音频数据的，因此需要外部格式转换电路，把 SPI 格式转换为 I2S 格式提供给 D/A 识别。

（4） 人机交互

为了让 iRhythm 能够被用户操作，GUI 界面和按键操作也是不可缺少的一环。

硬件需求：

（1） 数字部分

表 1-2：数字硬件需求

模块名称或功能	选型
Wifi 模块	ESP8266
SD 卡	Kingston SD HC 16GB
显示屏	12864 OLED
音频输出协议转换（SPI 转 I2S）	小型的 FPGA

（2） 模拟部分

表 1-3：模拟硬件需求

模块名称或功能	选型
D/A 模块	Pmod I2S （DIGILENT）
功率放大	TPA3137 （TI）
扬声器	4 欧姆，5 瓦喇叭
电源	LM2596（TI）

1.3 预期结果

预期实现的功能主要有：

1) 本地音频播放：对本地 SD 卡中的 MP3 音乐文件音频信号，iRhythm 网络音箱能够实现解码及播放。
2) 在线音频播放：通过抓取网络 FM 平台的数据，在联网状态下 iRhythm 网络音箱能够实现播放海量的在线音频。
3) 人机交互：用户可以通过 ARC EM Starter Kit 板载按键，来实现对网络音箱的手动控制。

1.4 项目实施评估

项目按照规划是分为两期来实现：

第一期开发周期是 3 月 1 号至 5 月 1 号，包括实现基本的用户界面，屏幕显示，按键功能，网络资源的获取，音频数据解码。主要目的是测试各模块底层驱动的稳定性，测试联网工作模式下音频的播放效果，完成 FPGA 上的 SPI 转 I2S 协议转换电路，为二期的改进工作提供宝贵的经验。

第二期的开发时间是 5 月 1 号至 6 月初。二期开发主要是在一期开发的基础上，完善各功能模块，对解码代码进行优化，对多个任务进行联调，推出一款效果更好的功能实现版本。

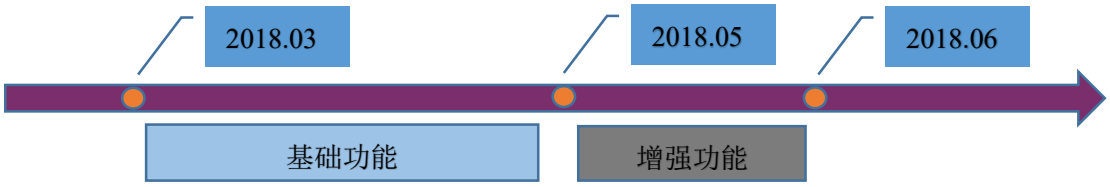

图 1-1：项目计划图

1.5 补充说明

部分队员参与了去年的 Synopsys ARC 杯电子设计竞赛，在 ilighting 项目中获得了二等奖的成绩。对 embarc 和 ARC EM Starter Kit 的特性较为熟悉，也参加过本科智能车，电子设计等全国大学生赛事，对嵌入式开发有充足的经验。

第二章 作品难点与创新

2.1 作品难点分析

2.1.1 网络音频的获取

由于版权加密等原因，从网络上获取音乐及其它音频下载资源有一定的难度，需要对各大网络音频提供商的在线播放的数据进行大量的分析，提取音频资源。

网络的响应具有不确定性：包括响应的数据量多少不确定，响应的正确性也无法保证。系统需要具有一定的容错性，在数据量超过接收上限时自动终止接收，对接收到错误的响应能够识别并跳过。防止因为网络的不稳定影响到整个系统的稳定。

iRhythm 采用 ESP8266 的方案，本质是与另一块只有有限 API 的网络处理器进行交互。由于缺乏对网络处理器足够的控制权限，就需要为网络处理耗费更多的资源，尽力满足网络处理的需求。

2.1.2 音频数据解码加速

从网络上获取的音频主要为 m4a 格式或者 mp3 格式，是高度压缩的数据。对其进行解析需要耗费大量的处理器运算资源。本设计中使用的解码库是为主频 100MHz 以上的 Cortex-M4 系列准备的，ARC EM Starter Kit 2.3 的主频最高为 25MHz，如果解码运算无法及时完成，不仅会影响其它任务的调度，更会导致音乐播放卡顿，无法连续播放，影响听觉效果。

对此就需要使用 ARC 的 Dsp 指令进行加速，同时尽力优化代码，提高运行效率，使得每帧解码运算尽快完成。

但提供的 GNU 工具链不支持 Arc 的 Dsp 库，直接使用 Arc 的 Dsp 汇编指令成为唯一选择，这也将成为开发过程中的一大难点。

2.1.3 自制音频输出数字协议转换模块

ARC EM Starter Kit 2.3 没有专用 I2S 音频输出模块。只能通过 DMA 传送

到 SPI 后发送。通过在开发板外进行扩展音频输出模块，使用 FPGA 进行数字协议转换，将 EMSK 用 SPI 格式输入的音频数据转换成 I2S 格式输出。

2.2 创新性分析

2.2.1 使用 ARC 的 DSP 指令集

为了满足运算的需求，在本设计中，我们使用 DSP 指令对解码算法进行了加速。在 GNU 工具链下，使用内联汇编封装了解码的核心代码，深度优化了算法，使得运算速度得到大幅度提升。

2.2.2 海量网络音乐库

本设计可以连接网络，抓取网络上的音频文件，当前很多网络电台都可以为 iRhythm 网络音箱提供音频源，相当于拥有了海量的播放资源。

2.2.3 功放放大输出音乐，听觉效果好

最终的音乐通过音频功率放大电路，输出于两个 5 瓦的大扬声器，可以通过调整功放的放大倍率，可以调整输出音量大小，获得更佳的听觉感受。

2.3 小结

运用 ARC 的 DSP 指令，使用低成本，低主频处理器完成音频数据的解码运算，FPGA 扩展外设是本作品的难点与创新。赛程的大部分时间都将使用在这里。同时，获得在线音频网站的接口，将使得 iRhythm 成为真正的网络音箱。音频功放的使用将使听觉效果得到提升。

第三章 系统结构与硬件实现

3.1 系统原理分析

iRhythm 网络音箱系统主要由网络音乐抓取下载，文件系统，音乐数据解码，音频数据输出这几个部分组成。虽然 ARC EM Starter Kit 2.3 只有 25MHz 的主频，但通过软件的合理分配和调度依然可以完美地实现网络音箱的功能。其中，最消耗资源的是音乐数据解码部分，需要进行合理的算法优化和 DSP 加速。网络音乐抓取下载部分虽然也要耗费大量时间，但其主要原因是网络延迟，而且主要由 ESP8266 模块负责，因此可以使用操作系统进行合理调度。文件的写入和读出速度也是一个瓶颈，但 ARC EM Starter Kit 2.3 开发板板载有 128MB 的 DDR2 内存块，可以通过合理的存储策略进行协调配合。音乐输出的大数据量搬运则可以交给 DMA 来完成。利用处理器，DMA 外设，ESP8266 的合理配合，使任务尽量并行化，流水线化，使得任务效率最大化。

3.2 系统结构

iRhythm 的音乐文件既可以来自本地的 SD 卡，也可以通过 ESP8266 Wifi 模块从网络上获取。完成解码后，通过 SPI 发送到外部的 FPGA 中，进行 SPI-I2S 协议转换，获得标准的 I2S 格式音频数据，交由播放电路进行数字-模拟转换，模拟信号放大及播放。

整个系统的结构图如下：

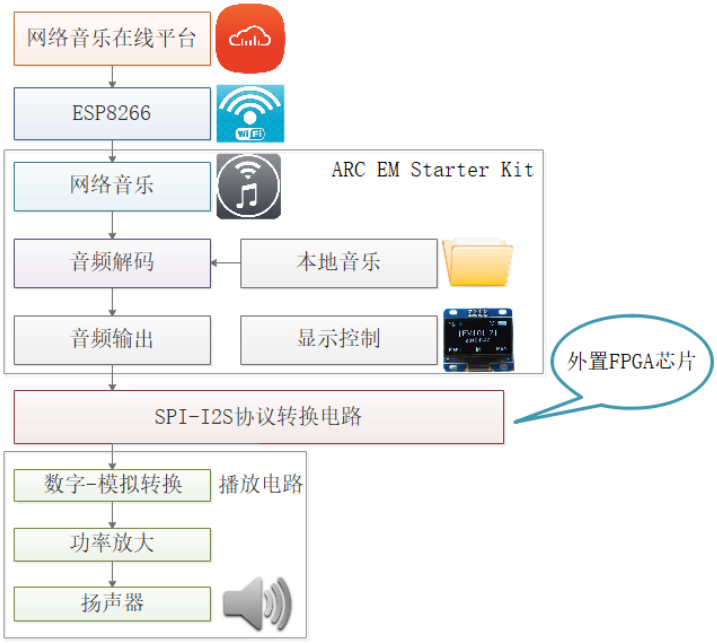

图 3-1：系统结构图

图 3-2：作品图（控制部分）

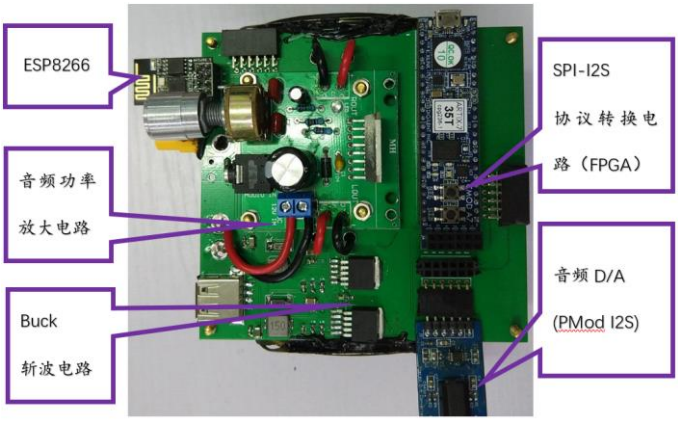

图 3-3：作品图（外设及信号处理部分）

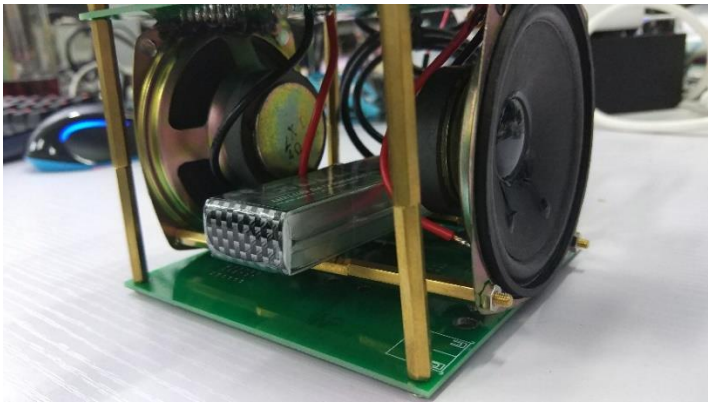

图 3-4：作品图（音乐输出部分）

3.3 硬件实现

在本项目中，需要通过硬件实现的内容主要包括音频输出数字协议的转换电路，音频数字-模拟转换电路，模拟信号功率放大电路、模拟音频输出电路，以及液晶屏显示，网络接口，电源等。

3.3.1 SPI 转 I2S 电路

由于我们的 EMSK 开发板的版本为 2.3，其不具有用于音频数据输出的 I2S 的外设接口，因此，我们需要利用 DMA 的辅助，通过 SPI Master 外设将解码后产生的大量音频信号输出，再通过一块外置的 FPGA 芯片，将 SPI 数据转换成为音频 D/A 芯片能够识别的 I2S 数据。

由于 I2S 为固定传输速度的串行协议，并考虑解码的速度和 ARC EM Starter Kit 的 DMA 外设传输的特点，SPI 的传输速度不能做到绝对稳定。因此，协议转换电路的输入和输出的速度是无法做到完全匹配的，当 SPI 输入的速度快于 I2S 的输出速度，部分数据将会丢失；当 SPI 的输入速度慢于 I2S 的输出速度，音乐将会出现卡顿。为此，在两个协议之间需要加入一个异步 FIFO 作为缓冲，并输出 FIFO 即将装满和即将耗空的标志信号，用于终止和恢复 SPI 发送，并保证 SPI 的传输速度在大部分情况下是快于 I2S 的。这样，才能保证系统稳定工作。

考虑到 FPGA 开发板作为外置模块存在，电路板面积需要足够小，我们选用了手上现有的，由 Digilent 推出的 Cmod A7 开发板来实现这个功能。

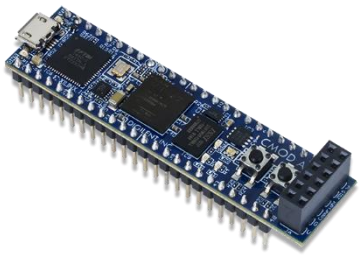

图 3-2：Cmod A7 FPGA 开发板

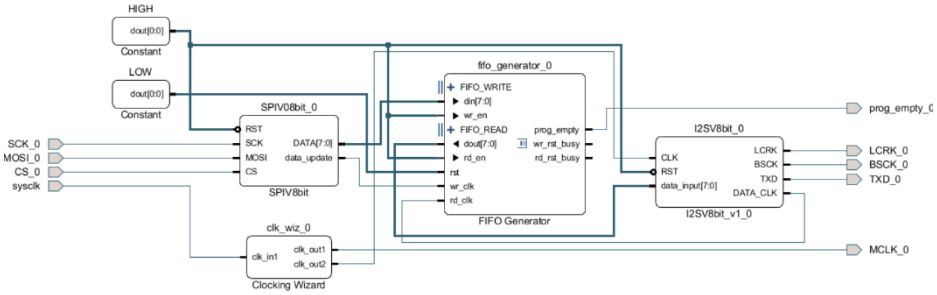

图 3-3：SPI-I2S 协议转换电路设计图

3.3.2 音频数字-模拟转换及功率放大电路

由于此部分不是设计的重点，为了加快原型的设计，这里使用了采购的模块进行实现。

为了将 I2S 数字音频数据转换成为模拟音频数据，本设计中采用 Digilent 提供的 Pmod I2S 模块。

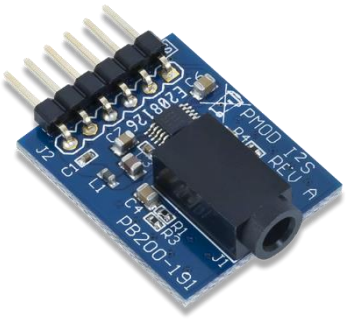

图 3-4:Pmod I2S 音频 D/A 模块

从 Pmod I2S 中输出的音乐可以直接使用耳机收听到，但为了增强效果，使用音频线将信号输入到功率放大电路中，使用扬声器播放。

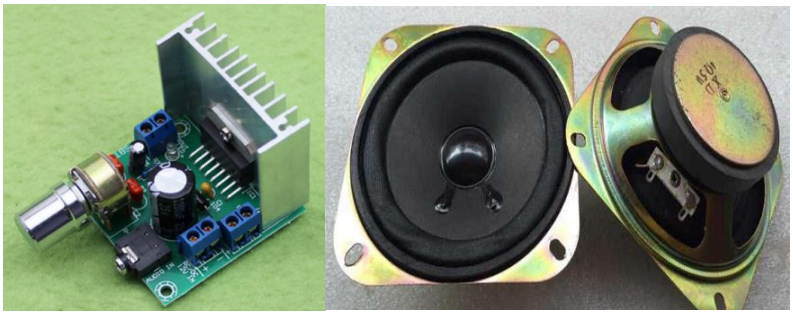

图 3-5：功放和扬声器

其中音频输出部分首先使用一块外置的 FPGA 将使用 SPI 输出的音频格式转换成为 I2S 输出，再由使用 I2S 协议的音频 D/A 转换芯片转换为模拟信号。最后通过搭建功率放大电路，将音频使用扬声器播放出来。

3.3.3 网络接口

为了让 ARC EM Starter Kit 具有连上无线网络的功能，本设计中使用 ESP8266 充当网络接口。通过使用配置过的最高波特率串口协议，以最高速度与 ESP8266 通讯，以达到最高的网络通讯速度。

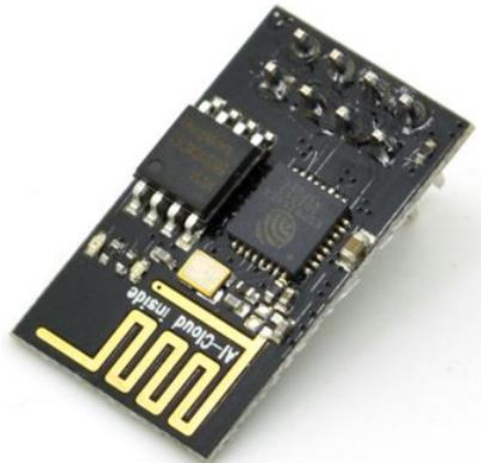

图 3-6：ESP8266 网络模块

3.3.4 人机交互模块

为了显示正在播放的音乐，解码速度，网络速度，待选音乐等信息。需要使用 OLED 打印信息。按键则直接采用 ARC EM Starter Kit 板载的轻触开关。

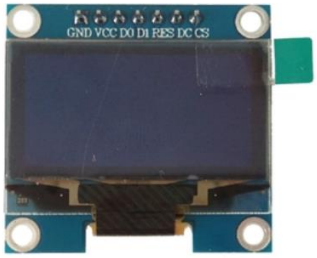

图 3-7：12864 OLED

3.3.5 电源处理

除外置模块的 3.3V 供电可以来源于开发板外，本作品中电源需求主要包括 ARC EM Starter Kit 开发板的 5V 供电，Cmod A7 开发板的 5V 供电，功率放大电路 7V-12V 的供电。

因此，本作品采用一块 7.4V 的锂电池作为系统的总电源，在共地连接的情况下，采用两块 LM2596-ADJ 进行 Buck 斩波变换。获得 5V 的电源和 7.0V 的稳压电源。

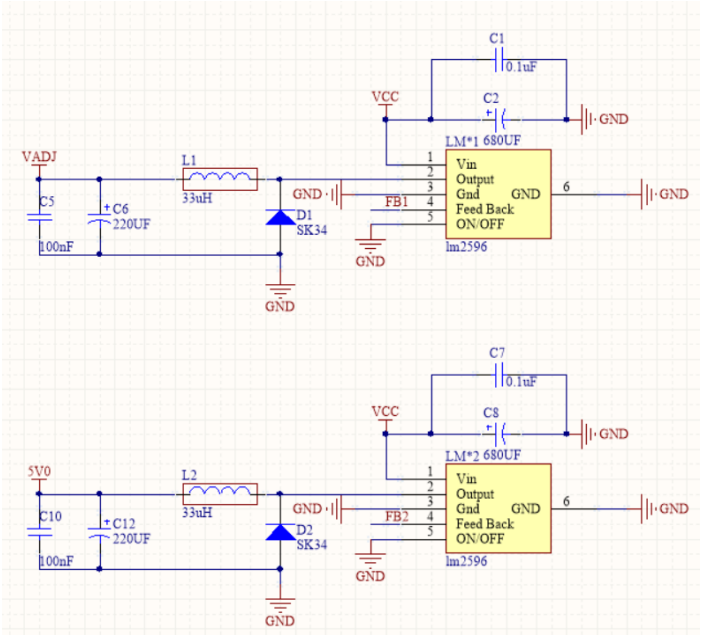

图 3-8：斩波电源原理图

3.4 小结

本作品的硬件工作量主要集中在数字协议转换电路和模拟音频信号处理方面。其中 SPI-I2S 协议转换电路的输入来自于 ARC EM Starter Kit 的 SPI 输出，其数据格式很大程度上决定于软件的配置，需要软件和硬件的密切配合才能完

成。而 SPI-I2S 协议转换电路的输出是音频 D/A 的输入，其协议格式决定于 D/A 芯片的参数。

第四章 软件设计流程及实现

iRhythm 软件部分主要需要完成音乐数据的解码、SPI 输出；网络音乐下载；用户控制这三项任务。

这三项主要任务对实时性都有较高要求：音乐解码输出不及时，将导致音乐播放卡顿，引入噪声，影响播放质量和听觉感受。网络接收数据不及时，将导致数据丢失，使接收的音乐文件不完整，影响正常解码。而用户控制如果无法得到及时响应，将大大影响用户体验。

4.1 软件设计流程

本设计在 Freertos 下开启 3 个主要任务，分别是负责人机交互的 Gui Task，负责解码 mp3 文件并通过 SPI 输出的 Music Task，负责连接网络，获取在线音乐信息并下载的 Net Task。

Gui Task 流程图：

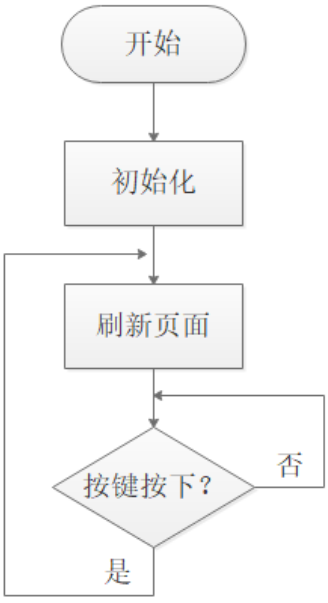

图 4-1：Gui Task 流程图

Music Task 流程图：

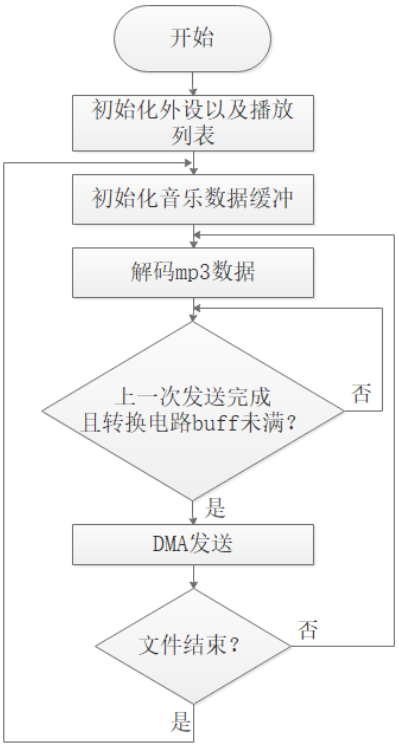

图 4-2：Music Task 流程图

Net Task 流程图：

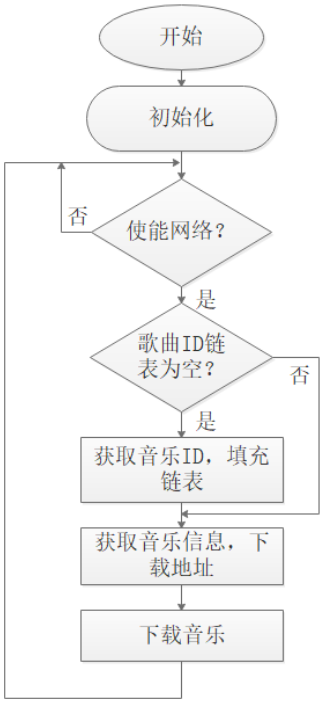

图 4-3：Net Task 流程图

4.2 软件实现

4.2.1 音乐解码播放任务的实现

4.2.1.1 音乐数据的获取

音乐数据可以来自于离线 SD 卡中预存的 Mp3 文件，也可以来自于网络中下载的在线歌曲。考虑到系统启动后从网上下载音乐需要等待一定的时间，因此系统外设初始化完成后将读取 SD 卡中文件数据，以链表形式建立播放列表，读取链表头的音乐文件信息首先进行播放。播放结束后将该歌曲信息从链表中删除。

由于解码速度快于播放速度，在解码的空暇时刻将进行在线音乐下载，每一次下载完成，将该歌曲信息插入到链表中用于下一首的播放。

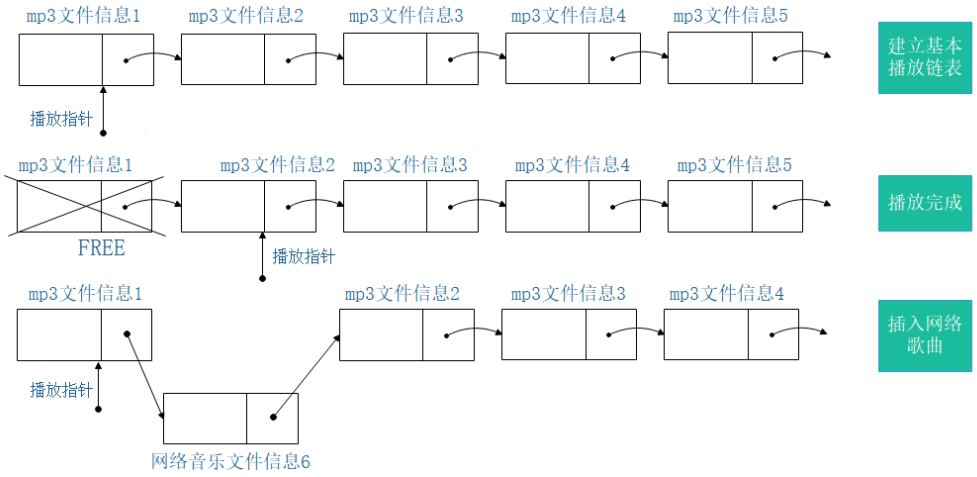

图 4-4：播放列表操作方式图

考虑到读取 SD 卡文件的速度不够快且会对 SPI 外设发送音频数据进行占用，再考虑到 ARC EM Starter Kit 有足够大的 DDR2，因此，无论是离线音乐还是在线音乐，都是将整首歌曲读入内存后再启动解码的。

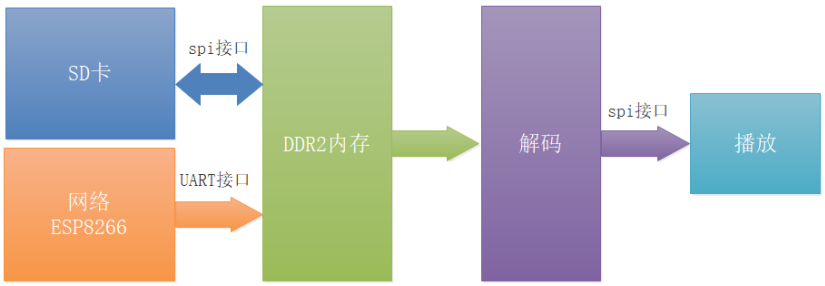

图 4-5：内存使用策略图

4.2.1.2 MP3 软件解码的实现

本设计中使用的 Mp3 解码库，是由 Github 上使用 BSD 协议开源的 Walkgeek mp3 软件解码库修改裁剪而来。开发者需要向解码库输入接口提供 mp3 压缩音频数据缓冲区的指针，解码完成后，将可以得到下次解码起始点的偏移量，以及一帧以“左声道+右声道”格式放置的，可以直接送至音频 D/A 播放 26ms 的原始音频数据。

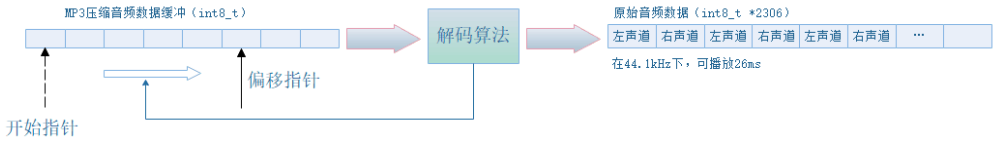

图 4-6：MP3 解码算法示意图

4.2.1.3 音频数据的 DMA 传输

由软件解码获得的原始音频数据需要送至 SPI 外设，从而发送到 SPI-I2S 数据协议转换电路。如果这部分工作由处理器完成，将会耗费大量的处理器资源。因此使用 DMA 代替处理器进行数据搬运将会是一个很好的选择。

通过申请两个相同大小的解压音频数据缓冲区，我们可以使用 Ping-pong 方法交替完成解码和发送。即在发送 A 缓冲区中的数据的同时进行下一帧的解码，将解码得到的数据放入 B 缓冲区中。下次则反过来，发送 B 缓冲区的数据，解码数据放入 A 缓冲区。

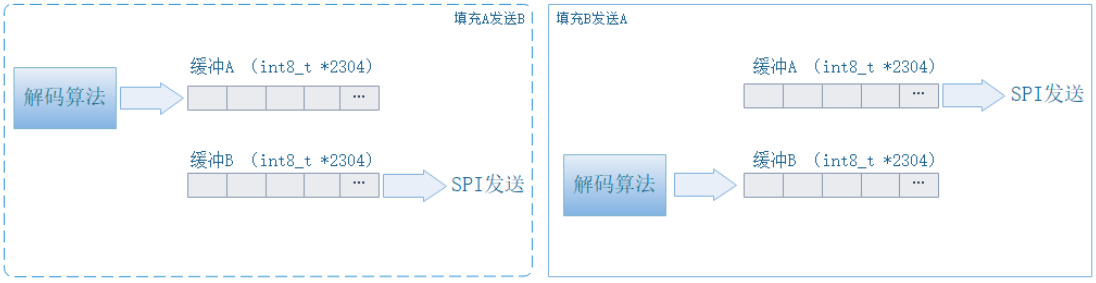

图 4-7：Ping-pong 方式示意图

数据在一次解码完成后，使能发送，使能 DMA 发送完成中断，并开始下一帧的解码，在下一帧解码完成准备发送时，查看上一次 DMA 发送完成中断

是否已经触发，如果已经完成，则继续发生和解码。否则挂起任务直到 DMA 发送完成。

4.2.2 MP3 软件解码加速的实现

由于 Walkgeek mp3 软件解码库是针对 Cortex-M4 系列的应用发布的，其主频较高，运算速度快，而 ARC EM Starter Kit 2.3 版本的主频最高为 25MHz。直接使用该库解码一帧能提供设备播放 26ms，而如果直接使用该解码库，解码一帧需要 32ms，远远不能满足音乐连续播放和其它任务调度的要求。因此，我们需要对该库进行裁剪和优化，并针对 ARC 的特点进行加速。

4.2.2.1 使用 DSP 指令加速运算

对解码运算中各部分函数运行时间的分析可知，大部分时间耗费在 32 位乘及累加运算中：原函数为了保证解码精度，防止运算过程中的溢出，采用了将 32 位扩展为 64 位数据进行运算，再将最后结果取高 32 位，进行最后的裁剪和输出的运算方法。

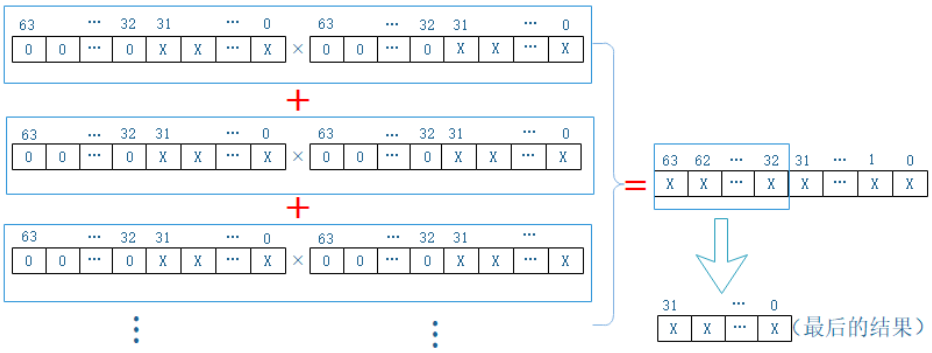

图 4-8：无优化运算示意图

对此，可以将 DSP 汇编指令 ***MAC*** 封装后替换原有的乘加函数接口，使得整个运算过程直接进行 32 位的乘和累加，将结果以 64 位形式保存在 ACC 高低寄存器中。在所有累加都完成后，通过读 ACC 寄存器的高位将最后读出并进行裁剪。

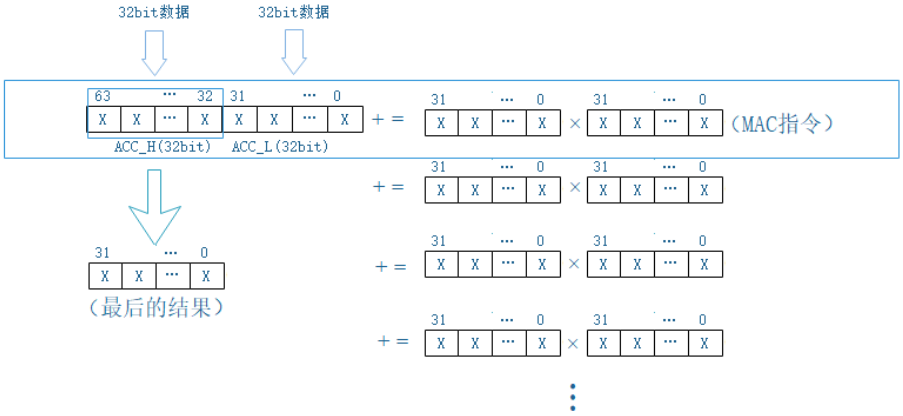

图 4-9：DSP 指令加速后运算示意图

使用此方法，单任务运行时解码一帧所需时间从之前的约 32ms，降低到约 19 ms。加速效果比较明显，而且所得的计算精度相比之前没有任何损失。已经可以单任务实现音乐的连续播放，但资源消耗仍然比较大。在多任务运行，特别是开始下载在线音乐文件时，由于需要实时接收数据，依然会出现解码不及时音乐卡顿的现象。

4.2.2.2 使用内联汇编深度优化核心解码运算

通过查看进行 DSP 加速后的反汇编代码，发现寄存器使用效率并不理想，在单组运算中，部分数据在内存和寄存器之间多次读写。

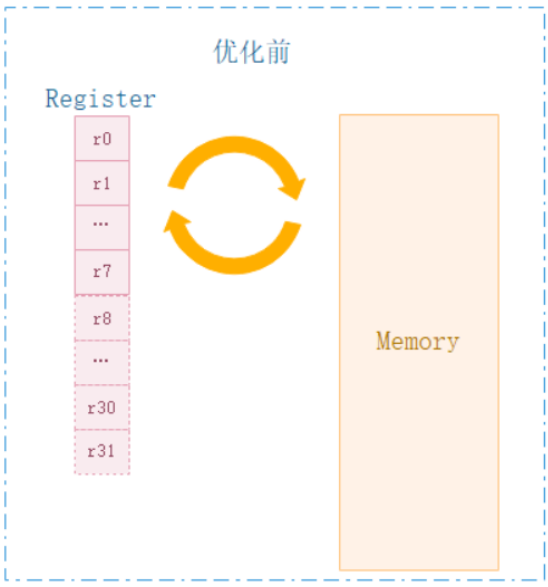

图 4-10：未优化内存读写

为此，在本设计中，Mp3 解码核心部分的代码完全使用内联汇编进行实现，从加载内存，计算结果，到写回内存，通过对寄存器的分配，使用进行了规划和优化。确保所有变量只从内存加载一次，在所有计算都完成后，再写回内存。大大提高了代码的效率。

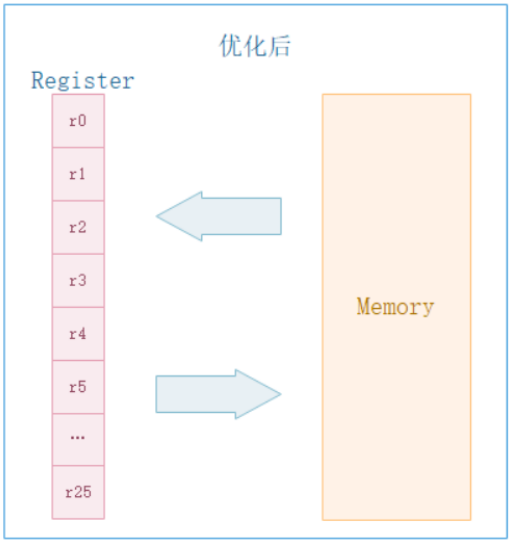

图 4-11：优化内存读写

使用此方法，MP3 解码耗费时间再减少了约 1ms，在单任务情况下需要约 18ms 完成。

4.2.2.3 使用 CCM 减少存储时间

通过查看链接脚本文件可知，启动时，代码和变量都是加载到以 0x10000000 为起始地址的 DDR 中。

本设计选用的是 ARC em7d 内核，具有 256KB 的 ICCM 和 128KB 的 DCCM。通过修改链接脚本文件，将所有代码放入 ICCM 中，将只读数据和解码所用到的全局变量放入到 DCCM 中。作为处理器内核私有的局部存储器，处理器对 CCM 的访问不产生任何总线通信，其访问时间是确定的，可大大提高性能，加快存取的速度。

通过这个方法，单任务 Mp3 解码耗费的时间减少到了约 14ms。而由于将解码的数据从 DDR 中分离，在多任务状态，特别是在 Net Task 开始接收在线音乐数据后，数据存储的压力得到明显缓解，多任务状态解码时间则有非常明显的降低。至此，ARC EM Starter Kit 已经完全可以实现流畅解码的同时执行其它任务。

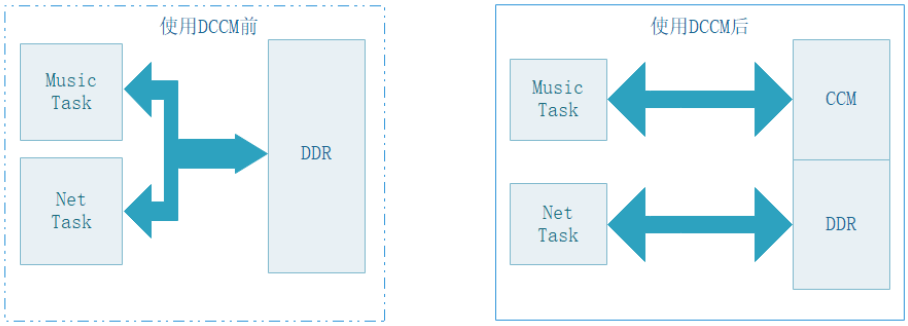

图 4-12：使用 DCCM 前后状态对比图

4.2.2.4 利用外部缓冲和 DMA 的配合减少跳转

本设计中，DMA 发送速度较快，正常情况下 DMA 传输将在解码完成之前完成。Music Task 一般情况下不会因为 DMA 未发送完成而挂起。

由于 MP3 解码速度需要高于实际播放所需速度才能保证音乐不卡顿连续播放，因此，只要时间足够长，外部 SPI-I2S 协议接口电路的缓冲区必将会填满。在缓冲区即将溢出时，SPI-I2S 协议接口电路会将控制 IO 电平置低，此时将挂起 Music Task，中止 SPI 传输和解码，开始执行 Net Task。

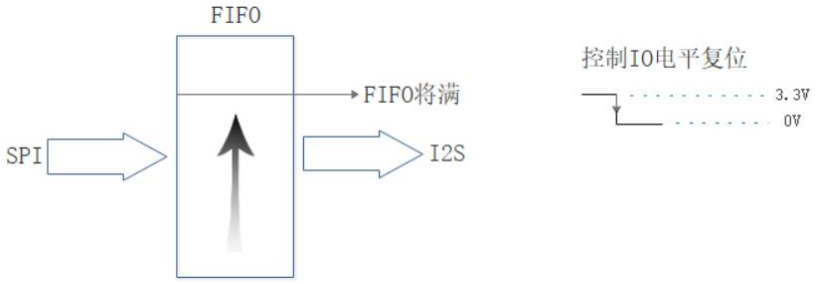

图 4-13：外部 FIFO 将满

当缓冲区即将耗尽，协议接口会将控制 IO 置高，此时将触发 IO 上升沿中断，恢复 Music Task 的执行。

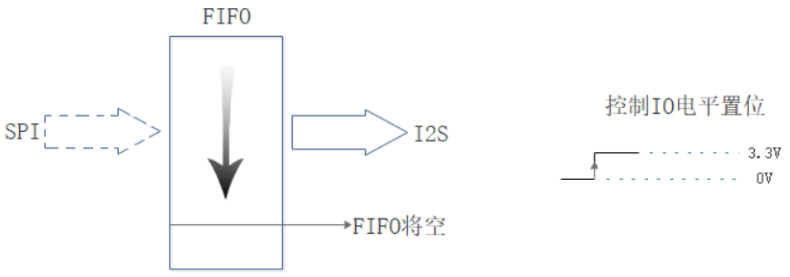

图 4-14：外部 FIFO 将空

利用这种类似施密特触发器（迟滞触发器）的方式，可以为网络下载任务提供足够的发送数据的时间。通过在 SPI-I2S 协议接口电路部分配置 FIFO，在

保证不会影响性能的情况下把将满-将空区间尽可能地设置大，将有效地提高任务执行的连续性，大幅度减少任务的调度频率。

4.2.3 在线音频数据获取的实现

4.2.3.1 连接网络

ESP8266 可以把连接 Wifi 的配置通过 AT 指令写入到模块的 Flash 中，但为了灵活配置，在系统初始化时，依旧依据程序的配置，连接指定的 Wifi 从而接入互联网。如果代码中指定的 Wifi 无法连接上，系统的初始化将无法完成。

4.2.3.2 使用 GET 方法模拟浏览器获取歌曲 ID

通过使用浏览器网络分析功能，对百度 FM（*fm.baidu.com*）进行分析。可知该在线音乐网站将歌曲分为包括“热歌”，“轻音乐”，“民谣”等的 39 大类。每首歌有各自的歌曲 ID 号。

图 4-15：百度 FM 音乐信息分析数据

首先需要通过歌曲的分类，请求获得歌曲的 ID。为此，本设计在连接了网络主机之后，仿照浏览器中的 GET 请求格式，构造并发送

GET http://fm.baidu.com/dev/api/?tn=playlist&id=public_tuijian_rege&hashcode=&_=1519727783752

HTTP/1.1\r\nHost: fm.baidu.com\r\nConnection: keep-alive\r\n\r\n

以获取热歌分类中歌曲的 ID。

在发送完成之后，接收到的是带有相应头的以 json 格式传递的多个歌曲 ID 信息，由于包含太多冗余信息，使用字符串查找的方式识别截取获得其中的音乐 ID 号。由于每个 ID 只会使用一次，因此建立单向链表来保存这些信息，每次用完即刻删除。

4.2.3.3 使用 POST 方法模拟浏览器获取歌曲信息

获取得到歌曲 ID 号之后，需要使用 POST 方法发送歌曲的 ID 号，向服务器请求获得歌曲包括歌名、歌手名、歌词下载链接、音乐下载链接在内的详细信息。

相关 POST 请求可以类似构造如下：

POST http://fm.baidu.com/data/music/songlink HTTP/1.1\r\nHost: fm.baidu.com\r\nConnection: keep-alive\r\nContent-Length: 17 \r\n\r\nsongIds=266942077

其中歌曲 ID 来自之前获取的音乐信息链表中，***Content-Length*** 的参数需要根据 ID 的长度计算获得。

请求之后将会返回音乐信息，其中包括音乐的下载链接：

songList: [...]
0: {...}
queryId: 266942077
status: 0
songId: 266942077
songName: 我好像在哪见过你
artistId: 88
artistName: 薛之谦
albumId: 241838068
albumName: 初学者
lrcLink: http://qukufile2.qianqian.com/data2/lrc/f63a5782603c444ad7c66b2d67bd534f/591587601/591587601.lrc
time: 279
linkCode: 22000
linkinfo: null
version: 影视原声
copyType: 1
enhancement: 2.430000
songLink: http://zhangmenshiting.qianqian.com/data2/music/af1918924adc3621b27a4da83e3bd9e4/584554521/584554521.mp3?xcode=de864c976ebfd76059dc68219d22c373
showLink: http://zhangmenshiting.qianqian.com/data2/music/af1918924adc3621b27a4da83e3bd9e4/584554521/584554521.mp3?xcode=de864c976ebfd76059dc68219d22c373
format: mp3
rate: 128
size: 4468707
format_arr: {...}
mp3: {...}
songLink: http://zhangmenshiting.qianqian.com/data2/music/af1918924adc3621b27a4da83e3bd9e4/584554521/584554521.mp3?xcode=de864c976ebfd76059dc68219d22c373
showLink: http://zhangmenshiting.qianqian.com/data2/music/af1918924adc3621b27a4da83e3bd9e4/584554521/584554521.mp3?xcode=de864c976ebfd76059dc68219d22c373
format: mp3
rate: 128

图 4-16：百度 FM 音乐详细信息分析数据

通过字符串计算，解析返回的信息即可获得下载链接。

4.2.3.4 下载歌曲到内存

连接百度 FM 的文件服务器：

211.91.125.36：80

使用 GET 方法，使用之前获得的下载链接请求歌曲文件：

GET

http://zhangmenshiting.qianqian.com/data2/music/af1918924adc3621b27a4da83e3bd9e4/584554521/584554521.mp3?xcode=de864c976ebfd76059dc68219d22c373 HTTP/1.1\r\nHost: zhangmenshiting.qianqian.com\r\nConnection: keep-alive\r\n\r\n

开启 ESP8266 透传，文件数据将通过串口接收中断填入预设缓冲中。

由于音乐解码函数能够自动识别或者跳过错误的文件信息，鉴于开发时间有限，简化了解析响应头的步骤，将网络响应头也一并放入文件缓冲中，这对后面音乐播放并没有影响。同时因为没有通过查看响应头判断文件的大小，本设计是通过定时查看串口接收的计数是否还在增加来判断文件接收是否完成的。

当一首歌完成下载后将歌曲信息作为下一个节点插入到播放列表的链表中，作为下一首等待播放，同时将接收完成标志位置位，阻塞网络任务。

4.2.3.5 提高网络传输速度

根据 ESP8266 的用户手册，该模块使用串口模式时，最高波特率可以达到 *115200bps* 的 40 倍。

通过查看 ARC EM Starter Kit 开发板的底层配置可知，其串口波特率是由总线时钟（50MHz）分频后除以 16 获得。因此在硬件电路许可的情况下，要想获得最高的下载速度，即需要使用最高的波特率，即不分频直接获得波特率。因此最高波特率为：

50,000,000Hz / 16 = 3125000 bps

在此情况下，最高网速理论可以达到：

3125000 / 10 / 1024 = 305KB/s

在实际测试中，由于外网网速的限制，实际最高下载速度可以达到约 *105KB/s*。

4.2.4 人机交互的实现

4.2.4.1 合理调度

OLED 的控制使用 U8glib，该库功能强大，但运行效率较低。在本设计中测试，每次刷新屏幕耗时约 40ms。对 Mp3 解码影响极为严重。

为了在顺畅执行主任务的同时不影响操作体验，需要对任务的调度进行合理的安排：

在本设计中，将 Gui Task 放在最高优先级，负责根据控制结构体参数刷新 OLED，保证可以及时响应。在每次刷新屏幕之后，挂起任务。因此，Gui Task 每次的执行时间是可知的 40ms。在 Music Task 运行到外部 FIFO 填满，需要挂

起时，发送信号量请求刷新一次屏幕。

只要外部 FIFO 的“将满-将空区间”是确定的，那么 Music Task 安全挂起不影响播放质量的时间也是确定的。只要该时间远大于 40ms，那么 Gui Task 刷新屏幕就不会影响音乐的解码和播放。

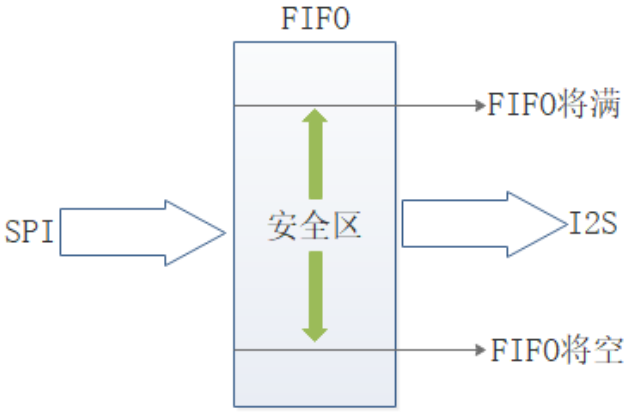

图 4-17：Music Task 在 FIFO 中的安全区示意图

4.2.4.2 打印性能数据

本设计的关键性能包括解码一帧所需时间和网络下载速度。

解码时间通过开启一个定时器，在启动解码前和解码完成后分别读取对比定时器计数器的值得到。

网络下载速度可以通过将两次调度网络任务时，网络文件缓冲区的数据量差除以两次调度的时间差获得。

4.2.4.3 选择播放歌曲

为了保证用户的按键控制足够及时，将按键控制放在 IO 中断中，以满足最高实时性的要求，在每次按键之后立即根据参数改变情况刷新 OLED 屏幕。

此时，因为 IO 中断的到来具有随机性，每次按键将打断音乐解码约 40ms 的时间。

最好的情况发生在 FIFO 将满，音乐任务刚刚挂起的时候，此时，FIFO 内有大量数据等待播放，因此不会对音乐播放产生影响。

最坏的情况发生在 FIFO 将空，音乐任务即将恢复执行时，此时，FIFO 内数据所剩不多，急待恢复 Music Task 进行解码来填充，此时 Gui Task 打断了解码任务，FIFO 需要多等待 40ms 才有新数据到来进行填充。因此外部 FIFO 从将空，到彻底变空需要坚持至少

$$40ms + t_{decode}$$

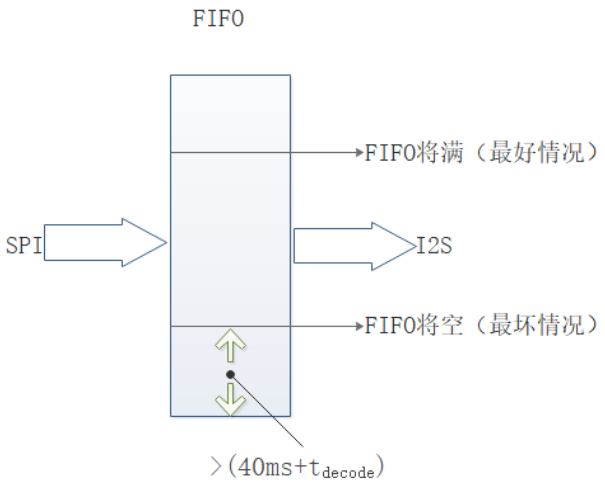

图 4-18：按键中断打断的最好情况和最坏情况示意图

4.3 小结

本设计中的核心任务是 Music Task，负责音乐的解码和输出播放，它的执行需要满足一定的速度才能保证音乐播放的流畅和连续，是需要首先保证的任务，但因为外部 SPI-I2S 协议转换电路中 FIFO 作为缓冲的存在，它对实时性需求被大大降低。

Net Task 的发送请求部分是最低优先级的任务，只要能得到调度，将请求发送出去即可。而网络接收的对实时性追求是极其高的，网络接收数据来源于 ESP8266 的串口发送，由于没有硬件流控，并且 ARC EM Starter Kit 的串口模块硬件 FIFO 很小，当数据到达时必须立马接收并搬运到文件缓冲区中，否则将会丢失数据。因此，网络接收数据部分是放在串口接收中断中执行的。

Gui Task 刷新屏幕是最高优先级的任务，在完成刷新后，长时间挂起任务等待下次刷新信号。使用这个方法，既可以保证刷新屏幕立马得到执行而不会有可察觉的滞后，也不会有多余的刷新占用太多资源。按键控制使用 IO 中断实现，使得用户命令得到及时的响应。

通过以上合理调度，多个任务得以流畅执行，配合很好。

第五章 系统测试与分析

5.1 系统测试指标

解码速度：Mp3 解码一帧，即执行函数

*MP3Decode(HMP3Decoder, unsigned char **, int *, char *, int);*

所需要耗费的时间。

下载速度：开始网络音乐下载后，音乐文件下载的平均速度。

5.2 测试环境

5.2.1 验证开发平台

运算内核：ARC EM Starter Kit 2.3 开发板平台 Arcem7d，内核频率 25MHz

网络环境：基于联通 4G 网络的手机 Wifi 热点

5.2.2 测试方案

解码速度测试：

分别在单独开启 Music Task 和同时开启 Music Task 和 Net Task 的情况下测试。

分别测试四种优化级别下完成 Mp3 解码函数 *MP3Decode()*的时间。

下载速度测试：

在音乐解码任务和网络下载任务同时启动的情况下，使用最高串口波特率：*3125000bps* 与 ESP8266 进行通讯。在启动音乐文件下载后，计算单位时间内（1 秒），网络数据的平均到达量。

5.3 测试结果

5.3.1 功能测试

进行完全的优化后，音乐解码播放，网络下载，人机交互操作三个任务可以很好地配合，互不干扰地运行。

音乐可以流畅地播放，没有失真，噪声。网络下载可以顺利进行，在本首歌播放完成之前，下一首歌可以完成下载，进入准备播放状态。GUI 控制流畅，操作没有延迟。

5.3.2 指标测试

解码速度测试：

表 5-1：优化效果对比

优化情况	单任务状态解码用时	多任务状态解码用时
无优化	32ms	任务无法调度
内联汇编封装 DSP 指令	19ms	任务无法调度
内联汇编封装核心代码（不使用 CCM）	18ms	26ms
内联汇编封装核心代码（使用 CCM）	14ms	18ms

网络速度测试：

表 5-2：网速测试表

次数	最低值	最高值	平均值
1	5kB/s	70kB/s	约 50kB/s
2	10kB/s	83kB/s	约 70kB/s
3	30kB/s	105kB/s	约 80kB/s
4	17kB/s	95kB/s	约 60kB/s
5	19kB/s	97kB/s	约 60kB/s

5.4 结果分析

5.4.1 解码速度分析

本设计中使用的是 ARC EM Starter Kit 2.3 arcem7d 内核。其内核频率只有 25MHz。而本设计中所使用的解码库，为了保证解码的精度并不受到损失，使用了大量 32 位乘法，甚至将 32 位的计算参数转换为 64 位进行运算。因此，在不使用任何优化的情况下，解码运算耗时较长，情况不理想。

在使用了 DSP 指令后，大量运算可以使用 DSP 加速，减少运算周期，同时利用部分 DSP 指令的特性，可以在不损失运算精度的情况下避免 64 位运算。因此计算效率得到大幅度提升。

将核心运算部分使用内联函数封装，最大程度地利用内核寄存器，可以有效降低存储器读写的频率。充分发挥流水线的优势。

由于网络接收是使用中断接收数据到缓冲区，该缓冲区较大，位于 DDR2 中。因此网速较快时，串口接收中断触发也较频繁，将会影响解码运算的资源，同时，两个任务都需要读写内存，读写速度将大幅度限制流水线性能的发挥。

在使用了 CCM 之后，将所有代码和解码所需要用到的核心数据分别放到 CCM 中，取指令和取数据所需要的时间大幅度减少，流水线优势得到发挥。同时网络下载任务主要写 DDR2，解码任务主要读写 DCCM，两者之间的干扰大幅度减小，因此性能得到了很大提升。

运算耗时在优化后较优化前减少了 43%，处理器最大占用仅为 70%，并且仍然有很大的优化空间。

5.4.2 网络下载速度分析

在本设计中使用了 ARC EM Starter Kit 开发板可以使用的最高波特率 *3125000bps* 与 ESP8266 通讯，因此理论最高网络下载速度可以达到 *305kB/s*。但实际最高下载速度只有 *105kB/s*。据此可知，下载速度受限于外网提供的带宽，测试没有达到硬件可以承受的最高带宽。

第六章 总结展望

本设计是一个网络音箱的设计原型，既可以播放本地 SD 卡中的音乐文件，也可以通过联网抓取网络上的在线音乐到本地进行播放。

通过一块 FPGA 开发板，给只有 SPI 接口的 ARC EM Starter Kit 开发板提供 SPI-I2S 的协议转换，从而扩展得到音频数字输出接口。

使用功率放大电路放大输出的音频信号，使用扬声器播放，从而获得更好的听觉感受。

本设计通过使用 DSP 加速，代码优化等方式，使用只有 25MHz 内核频率的 EMSK 开发板，完成了之前需要在 Cortex-M4，100MHz 内核频率下完成的 Mp3 解码播放任务。通过合理的任务调度，在音乐解码播放期间，可以从网络上抓取下载在线音乐提供播放；可以操作按键控制界面选择音乐。

通过这次比赛设计，使用软件的方式对 Mp3 进行解码，使我们对计算机的原理有了一个更加全面的认识；对工程上使用处理器的优势弥补其劣势的方法有了更深刻的理解。通过这次比赛，学习到了很多。

下一步，我们将会在此基础上，尝试边下边播的控制流程，同时，尝试移植开源的 Faad 解码库，解码当前网络上更流行的 m4a 音频格式。该格式压缩率更高，解码所需内存极大，操作难度更大，但更加普遍地应用在当前的网络音箱中。

考虑到处理器资源剩余较多，语音识别也将作为我们尝试一个“小目标”，我们将为此而努力。

最后感谢 Synopsys 举办了这场比赛，我们有机会通过比赛快速地学习知识，获得实践。我们通过这次比赛受益匪浅。

参考文献

[1] 吴小锋,张西宁,马博.基于 ARM 的 MP3 播放器的研究与实现[J].电子设计工程,2018,(1):25-28.

[2] SYNOPSYS. ARC_V2_ProgrammersReference

[3] SYNOPSYS. GNU_Toolchain_for_ARC