

# 数字逻辑设计报告 Music3D 组

潘传宇 任一

June 17, 2020

# Contents

<b>1</b>	<b>实验目的</b>	<b>3</b>
<b>2</b>	<b>实验完成情况与任务分工</b>	<b>3</b>
2.1	完成情况 . . . . .	3
2.2	任务分工 . . . . .	3
<b>3</b>	<b>实验演示说明</b>	<b>4</b>
<b>4</b>	<b>文件说明</b>	<b>4</b>
<b>5</b>	<b>总体设计</b>	<b>5</b>
5.1	整体框架 . . . . .	5
5.2	FPGA 设计概要 . . . . .	5
5.3	光立方工作原理 . . . . .	6
<b>6</b>	<b>关键技术分析</b>	<b>7</b>
6.1	Fast Fourier Transformation . . . . .	7
6.2	串口协议通信 . . . . .	8
6.3	灯光信息整合输出 . . . . .	8
<b>7</b>	<b>程序注释</b>	<b>8</b>
<b>8</b>	<b>波形仿真结果说明</b>	<b>8</b>
<b>9</b>	<b>遇到问题与解决方法</b>	<b>8</b>
9.1	FPGA 逻辑资源不足 . . . . .	8
<b>10</b>	<b>实验总结</b>	<b>9</b>

## 1 实验目的

本实验的最终目的是希望实现一个能够接收音频信号，并将音乐的律动以“立体”的方式显示出来的音频可视化系统。其中我们使用 AN831 模块作为音频的输入模块，并使用光立方模块作为“立体显示”模块。具体目标有以下几点：

1. 实现音频的数字信号输入及处理；
2. 借助 FFT 算法实现音频的时域信号向频域信号转换；
3. 实现频域信号向光立方所需的灯光信号的转换；
4. 实现灯光信号的“队列式”存储与移位；
5. 实现灯光信号的“整合打包”与串口协议输出；

## 2 实验完成情况与任务分工

### 2.1 完成情况

时间	任务
第九、十周	确定主题以及大致设计框架，购买外设，熟悉 FPGA 板使用
第十一周	确定设计框架，开始编写音频处理部分代码，调试外设
第十二周	完成音频输入、串口输出部分的代码编写，进一步完善设计框架
第十三周	完成音频 FFT 部分的代码编写，上板调试音频部分
第十四周	完成灯光信号处理部分的代码编写，上板进行整体调试
第十五周	尝试多种模式设计，优化效果
第十六周	准备课堂展示

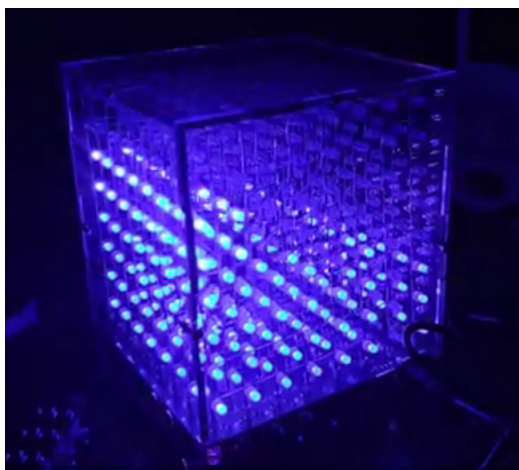
### 2.2 任务分工

- 潘传宇：调试板子及外设，编写灯光信号处理部分的代码，编写光立方通信协议部分的代码；
- 任一：选购音频模块 AN831，编写音频输入及 FFT 处理部分的代码，协助调试硬件；

### 3 实验演示说明

实验演示按照如下步骤进行：

1. 按照引脚分配将 AN831 模块和光立方模块接入 FPGA 板；
2. 将板子接入电源，并将程序烧写进入板子；
3. 启动光立方，通过设置单片机工作模式进入“串口接收模式”；
4. 烧写完成后，板子立即进入工作状态，此时通过音频线将音频接入 AN831 模块后，即可看到光立方上随音乐变化的显示效果；
5. 默认模式为“舒缓模式”，按住实验板上的 reset 键可进入“动感模式”；



展示效果图

展示视频链接如下：

### 4 文件说明

本项目文件说明如下表：<sup>1</sup>

---

<sup>1</sup>完成者中带 \* 对应的文件，参考了<https://github.com/Ugon/fpga-fft-equalizer>或直接使用 Altera 中的 PLL，并非完全独立实现。

文件名	功能	完成者
AN831.vhd	读入 AN831 模块的输出，进行 FFT	任一
audio_processor.vhd	由时域数字信号做 FFT 得到频域数字信号	任一 *
dsp_slave_reader.vhd	将串行的 ADCDAT 以样本点为单位转为并行	任一 *
fft_dif.vhd	递归实现 FFT	任一 *
fft_input_deserializer.vhd	为 FFT 准备时域信号	任一 *
fft_twiddle_factors_64.vhd	FFT 计算中用到的旋转因子	任一 *
fft_utils.vhd	FFT 计算中用到的函数	任一 *
i2c_clk_prescaler.vhd	时钟分频，由 50MHz 转换为 100kHz	任一 *
i2c_master_writer.vhd	配置 WM8731 芯片寄存器	任一 *
mTransmitter.vhd		潘传宇
mTXD.vhd		潘传宇
mw8731_controller.vhd	处理 ADCDAT，配置 WM8731 芯片寄存器	任一 *
mypll.vhd	时钟分频，由 100MHz 转换为 50MHz	任一 *
top.vhd	顶层模块，连接音频处理和灯光信息传输模块	任一
trans_pkg.vhd		潘传宇

Table 1: 项目文件说明

## 5 总体设计

### 5.1 整体框架

本项目设计的整体框架如 Figure(5.1)所示。整个工程的输入为 3.5mm 的音频线，输入的是模拟信号。经过 AN831 模块中的 WM8731 芯片的模数转换，得到 24 位的数字信号。数字信号输入 FPGA 后，由 FPGA 做 FFT 和数据的打包处理，由 RX/TX 串口将灯光信号发送给光立方，光立方产生显示效果。

### 5.2 FPGA 设计概要

FPGA 设计概要如 Figure(5.2)所示。在 FPGA 中，我们首先接收了 AN831 模块输出的时域数字串行信号，并将其转为并行，之后进行 FFT，得到频域信号，由频域信号再转为灯光信息。接着将灯光信息进行存储和整合，最后由 UART 串口输出给光立方。

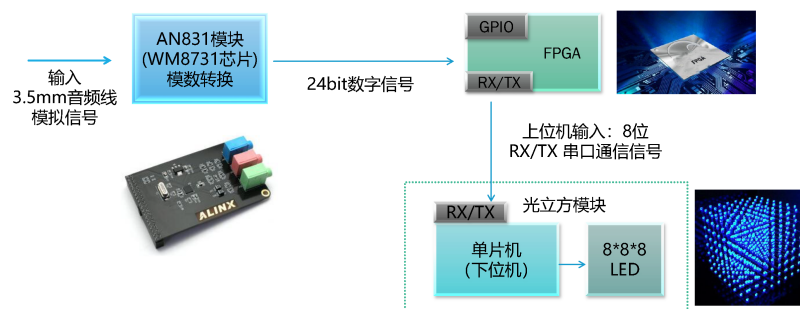


Figure 1: 整体框架图

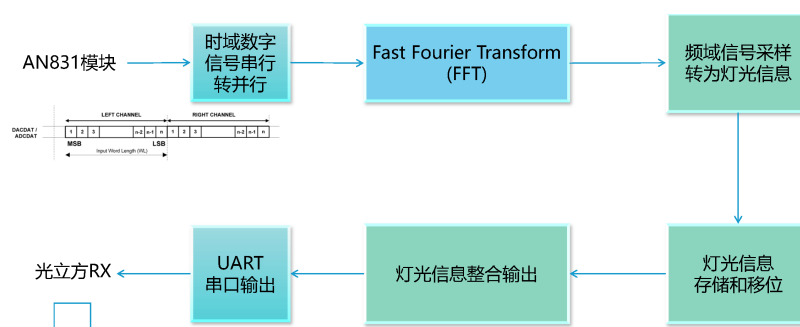


Figure 2: FPGA 设计概要图

### 5.3 光立方工作原理

光立方的工作原理示意如 Figure(5.3)所示。光立方是三维的显示，三个维度分别是时间、频率和强度。随着时间的推移，光立方的显示沿着时间轴队列式逐层后移。从频率-强度平面来看，光立方的 8 列分别代表 8 个频率分量的强度。亮灯多的列对应的频率分量强度大，亮灯少的列对应的频率分量强度小。同时，音量的大小也与强度有一定正相关关系，音量大则强度大，音量小则强度小。

## Music3D 工作示意

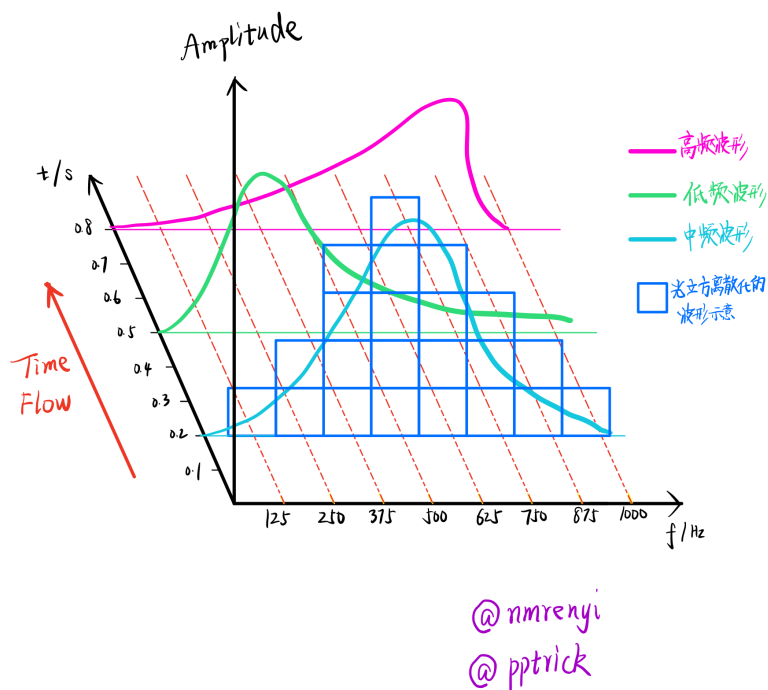


Figure 3: 光立方工作原理示意图

## 6 关键技术分析

### 6.1 Fast Fourier Transformation

为了从时域信号中提取频域信息，我们需要使用快速傅里叶变换 (FFT)。在本代码中，我们配置 WM8731 芯片的采样频率为 8KHz，进行 64 点的 FFT，得到了 125Hz, 250Hz, 375Hz, 500Hz, 625Hz, 750Hz, 875Hz, 1000Hz 这 8 个频率处的频谱分量，正好对应光立方每一帧的 8 列灯光信息。为了节约计算成本，我们在进行 FFT 时，通过定点数的方式，提前储存了旋转因子。

## 6.2 串口协议通信

通过 UART 串口传送串行信号，可以将设计好的灯光信息传给光立方模块上的单片机（相当于下位机），并由单片机控制灯的亮灭。每一时刻的灯光信号由 65 个 8 位（并行）逻辑向量构成，需要将这些 8 位并行数据转化为串行的 01 输出，因此需要编写串口协议通信模块。该模块基于状态机实现，通过不同状态表示当前输出处于起始位、数据位、校验位还是停止位，并通过计数方法使之适应相应的波特率。

## 6.3 灯光信息整合输出

灯光信息采用“队列”结构存储，即每一时刻由采样模块收集频域信息并转化为灯光信息，将该灯光信息传入“队尾”，然后将“队顶”信息弹出。

每次传给光立方的数据是 65 个 8 位逻辑向量，它是由 1 个“起始位”和 64 个“数据位”构成的；在传输时需要依次将这些 8 位逻辑向量传给“串口通信模块”。实现这一功能的难点在于与频域采样频率和串口波特率之间相适配。此模块同样基于状态机实现，用不同状态判断当前输出处于起始位、数据位还是停止位；同时由于采样频率远小于波特率，可以通过计数方法将灯光信息输送频率调整到一个较为恰当的值。

# 7 程序注释

# 8 波形仿真结果说明

# 9 遇到问题与解决方法

## 9.1 FPGA 逻辑资源不足

在最初的项目调试中，我们希望做到的是采样频率 48kHz 的 1024 点 FFT。这样做存在的问题是，由于 FFT 涉及到大量的乘法，导致板子的逻辑资源严重不足。

因此我们从两方面进行改进，一方面是降低 FFT 点数，从 1024 点降低到 64 点，使得计算复杂度降低，同时我们也降采样频率由 48kHz 降低到 8kHz，从而保证 FFT 得到的每两点之间的频率差不会过大。另一方面，我们降低了旋转因子定点数的位数，从原项目的 16bit 旋转因子降低到 14bit



旋转因子，进一步降低了计算的复杂度。通过这两方面的改进，我们成功实现了采样频率为 8kHz 下，64 点的 FFT。

其他可能的改进思路还有降低采样深度 (既 bits per sample)。这样做的原理与降低旋转因子的位数相似，都能够降低 FFT 中乘法的资源使用。但由于本项目使用的是 WM8731 的 DSP 模式传输数据，在 DSP 模式下 WM8731 模块仅支持 24bit 的采样深度，因此我们最终没有采用这种优化方案。此外，也可以考虑从 FFT 的算法本身进行改进，例如利用移位代替乘除法等方法，降低硬件逻辑资源使用率。

## 10 实验总结