**《数字逻辑电路与片上系统》**

**实验报告**

**The experimental report of**

**Digital Logic & System on Chip**

|  |  |
| --- | --- |
| 学 号 | 20049200453 |
| 姓 名 | 刘海旭 |
| 任课教师 | 康槿 |
| 实验教师 | 康槿 |
| 实验名称 | 面向通信专业的最小化片上系统设计 |
| 时 间 | 2022-2022第1学期（2021.10-2021.11） |
| 单 位 | 西安电子科技大学通信工程学院 |
| 组 号 |  |
| 组 成 员 |  |

说明

（1）本实验报告为西安电子大学通信工程学院本科卓越工程师班数字逻辑电路与片上系统教学过程使用，包括实验前预习、实验思路整理、实验记录，还包括实验报告内容。

（2）严格遵守本模板格式，不能新增排版格式。

（3）本文档针对第四次实验，将作为实验成绩依据计入期末总评。

目 录

[第1部分 学习备忘录 1](#_Toc85211595)

[2021年10月 1](#_Toc85211596)

[第7周（2021年10月11日-10月17日） 1](#_Toc85211597)

[第8周（2021年 月 日- 月 日） 1](#_Toc85211598)

[2021年11月 1](#_Toc85211599)

[第2部分 项目设计 2](#_Toc85211600)

[1 项目计划 2](#_Toc85211601)

[2 项目需求 2](#_Toc85211602)

[3 系统概述 2](#_Toc85211603)

[第3部分 项目实现 3](#_Toc85211604)

[1.系统设计 3](#_Toc85211605)

[2.模块设计 3](#_Toc85211606)

[3.电路实现及功能仿真 3](#_Toc85211607)

[第4部分 项目测试 4](#_Toc85211608)

[1.系统调试 4](#_Toc85211609)

[2.功能验证 4](#_Toc85211610)

[3.性能分析 4](#_Toc85211611)

[第5部分 优化与思考 5](#_Toc85211612)

[1.模块优化 5](#_Toc85211613)

[2.系统优化 5](#_Toc85211614)

[3.问题与思考 5](#_Toc85211615)

[4.总结心得 5](#_Toc85211616)

# 第1部分 学习备忘录

## 2021年10月

（主要记录用于本实验的学习时间，包括课上课下）

### 第7周（2021年10月11日-10月17日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 10.16 | 步骤一、步骤二探究 | 2h |
|  |  |  |

### 第8周（2021年 月 日- 月 日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
|  |  |  |
|  |  |  |

## 2021年11月

# 第2部分 项目设计

## 1 项目计划

设计FFT运算系统进行数字信号处理

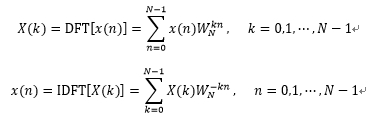
## 2 项目需求

## 3 系统概述

FFT是一种DFT的高效算法，称为快速傅立叶变换（fast Fourier transform）。傅里叶变换是时域—频域变换分析中最基本的方法之一。在数字处理领域应用的离散傅里叶变换(DFT：Discrete Fourier Transform)是许多数字信号处理方法的基础。

FFT是一种DFT的高效算法，称为快速傅立叶变换（fast Fourier transform）。FFT算法可分为按时间抽取算法和按频率抽取算法，先简要介绍FFT的基本原理。从DFT运算开始，说明FFT的基本原理。

DFT的运算为：

[](https://baike.baidu.com/pic/FFT%E5%8E%9F%E7%90%86/8966333/0/5366d0160924ab1832d4ecb237fae6cd7a890b70?fr=lemma&ct=single)

Wn=exp(-j2π/N)

C++实现代码：（来源与网络）

#include<bits/stdc++.h>

using namespace std;

//complex是stl自带的定义复数的容器

typedef complex<double> cp;

#define N 2097153

//pie表示圆周率π

const double pie=acos(-1);

int n;

cp a[N],b[N];

int rev[N],ans[N];

char s1[N],s2[N];

//读入优化

int read(){

int sum=0,f=1;

char ch=getchar();

while(ch>'9'||ch<'0'){if(ch=='-')f=-1;ch=getchar();}

while(ch>='0'&&ch<='9'){sum=(sum<<3)+(sum<<1)+ch-'0';ch=getchar();}

return sum\*f;

}

//初始化每个位置最终到达的位置

{

int len=1<<k;

for(int i=0;i<len;i++)

rev[i]=(rev[i>>1]>>1)|((i&1)<<(k-1));

}

//a表示要操作的系数，n表示序列长度

//若flag为1，则表示FFT，为-1则为IFFT(需要求倒数）

void fft(cp \*a,int n,int flag){

for(int i=0;i<n;i++)

{

//i小于rev[i]时才交换，防止同一个元素交换两次，回到它原来的位置。

if(i<rev[i])swap(a[i],a[rev[i]]);

}

for(int h=1;h<n;h\*=2)//h是准备合并序列的长度的二分之一

{

cp wn=exp(cp(0,flag\*pie/h));//求单位根w\_n^1

for(int j=0;j<n;j+=h\*2)//j表示合并到了哪一位

{

cp w(1,0);

for(int k=j;k<j+h;k++)//只扫左半部分，得到右半部分的答案

{

cp x=a[k];

cp y=w\*a[k+h];

a[k]=x+y; //这两步是蝴蝶变换

a[k+h]=x-y;

w\*=wn; //求w\_n^k

}

}

}

//判断是否是FFT还是IFFT

if(flag==-1)

for(int i=0;i<n;i++)

a[i]/=n;

}

int main(){

n=read();

scanf("%s%s",s1,s2);

//读入的数的每一位看成多项式的一项，保存在复数的实部

for(int i=0;i<n;i++)a[i]=(double)(s1[n-i-1]-'0');

for(int i=0;i<n;i++)b[i]=(double)(s2[n-i-1]-'0');

//k表示转化成二进制的位数

int k=1,s=2;

while((1<<k)<2\*n-1)k++,s<<=1;

init(k);

//FFT 把a的系数表示转化为点值表示

fft(a,s,1);

//FFT 把b的系数表示转化为点值表示

fft(b,s,1);

//FFT 两个多项式的点值表示相乘

for(int i=0;i<s;i++)

a[i]\*=b[i];

//IFFT 把这个点值表示转化为系数表示

fft(a,s,-1);

//保存答案的每一位(注意进位）

for(int i=0;i<s;i++)

{

//取实数四舍五入，此时虚数部分应当为0或由于浮点误差接近0

ans[i]+=(int)(a[i].real()+0.5);

ans[i+1]+=ans[i]/10;

ans[i]%=10;

}

while(!ans[s]&&s>-1)s--;

if(s==-1)printf("0");

else

for(int i=s;i>=0;i--)

printf("%d",ans[i]);

return 0;

}

指令集设计：



以上是参考指令集

其他指令需要进一步分析

# 第3部分 项目实现

## 1.系统设计

## 2.模块设计

## 3.电路实现及功能仿真

# 第4部分 项目测试

## 1.系统调试

## 2.功能验证

## 3.性能分析

# 第5部分 优化与思考

## 1.模块优化

## 2.系统优化

## 3.问题与思考

## 4.总结心得