南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

DSP 应用技术实验

FIR 滤波器实现实验报告

作　　者：	周鹏	学　　号：	9181040G0740
同 组 人：	杨霄宇	学　　号：	9181040G0736
同 组 人：	许昕荣	学　　号：	9181040G1038
学　　院：	电子工程与光电技术学院		
专　　业：	电子信息工程		
班　　级：	电信 3 班		
组　　号：	第二组 B6		
题　　目：	DSP 应用技术实验		
	FIR 滤波器实现实验报告		
指 导 者：	李彧晟		

2021 年 4 月

目录

1 实验目的

1.巩固数字 FIR 滤波器的概念

2.理解定点 DSP 中数的定标、有限字长、溢出等概念

3.理解算法实现中实时的概念

4.掌握 DSP 开发过程以及基本调试方法

2 实验仪器

2.1 实验仪器清单

计算机，TMS320F28335 DSP 教学实验箱，XDS510 USB 仿真器，示波器

2.2 硬件连接示意图

实验硬件连接大致如图 2. 1 所示。

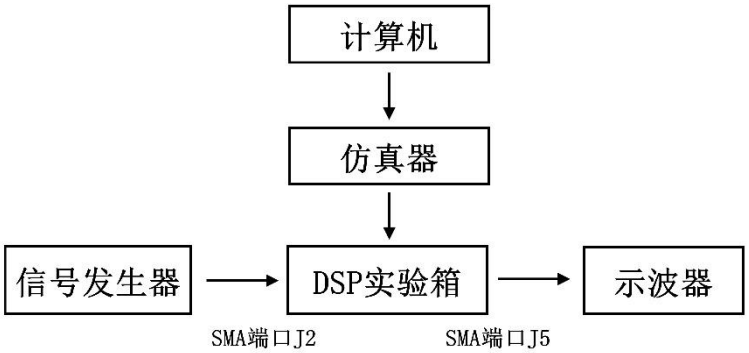

图 2. 1 硬件连接示意图

3 实验步骤及现象

3.1 算法实现流程图

结合实验要求，实验之前首先必须对 FIR 滤波器的设计、实现算法有所了解，必要时通过计算机算法仿真，理解 FIR 滤波器特性。

根据 FIR 滤波器算法，编写 C 源程序，实现算法功能。并验证 DSP 实现时算法的正确性以及精度的要求。这种算法功能上的仿真可以利用 CCS 集成开发环境中数据 IO 来模拟信号的输入，完成验证算法精度与功能的正确。

验证了算法的功能正确之后，可以将程序下载到 DSP 上运行，观察现象。更为重要的是在硬件平台上验证系统的实时性，以及评估资源的使用情况。若满足实时性要求，则测试各项指标，应该与原理设计相吻合。如果实现与理论不一致，则首先检查算法的实时性，以及资源使用是否冲突等原因，对程序进行用是否冲突等原因，对程序进行用是否冲突等原因，对程序进行优化后再次编译链接，重新验证直至正确。算法的优化有时会贯穿于整个设计之中。于是，程序流程图如图 3. 1 所示。

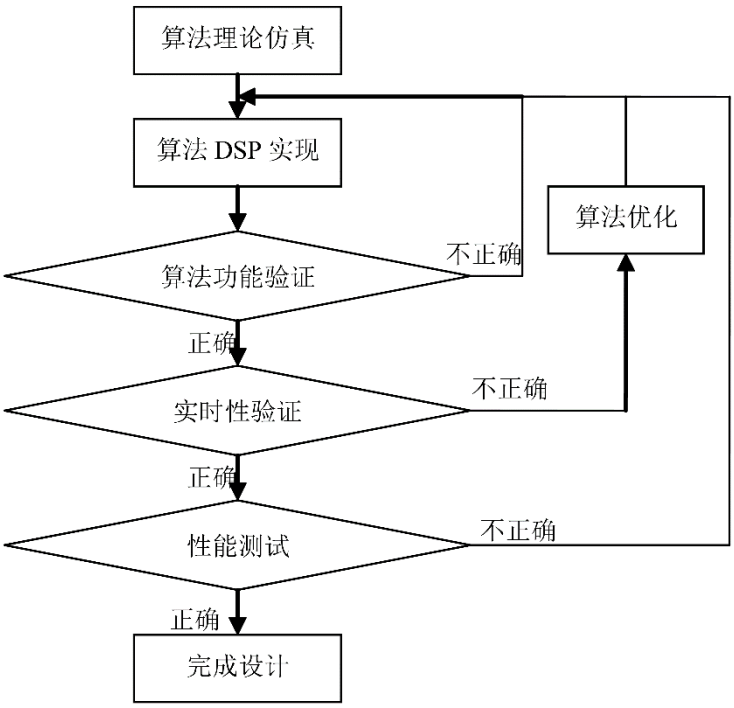

图 3.1 算法实现流程图

3.2 程序流程图

FIR 滤波器算法属于典型的数据流处理方式，每到达一个新数据，就必须进行一次计算，更新输出。因此，和 DSP 的数据采集实验类似，用 DSP 实现实时的 FIR 信号处理算法必须依赖于 ADC、DSP 以及 DAC 三大基本部件。充分利用 DSP 片上 ADC 外设，实现模拟信号的采样，并由 DSP 完成 FIR 核心算法，由实验箱中 DAC（AD9747）来完成数字到模拟的还原。在数据采集实验基础上，我们对程序流程稍加改动，就可实现完整数字 FIR 滤波器功能。程序流程如图 3.2 所示。

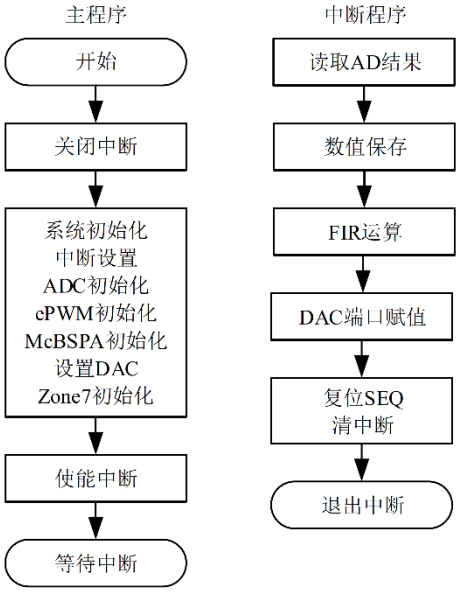

图 3.2 程序流程图

3.3 系统设计

3.3.1 利用设计滤波器系数

本次实验是在实验十一的基础上，对采集到的数据进行滤波处理。首先用 MATLAB 软件设计滤波器系数。利用 fdatool 命令行打开滤波器设计窗口，设计一阶数为 48 阶，截止频率为 800Hz 的高通滤波器，幅频相频特性曲线如图 3.3 所示

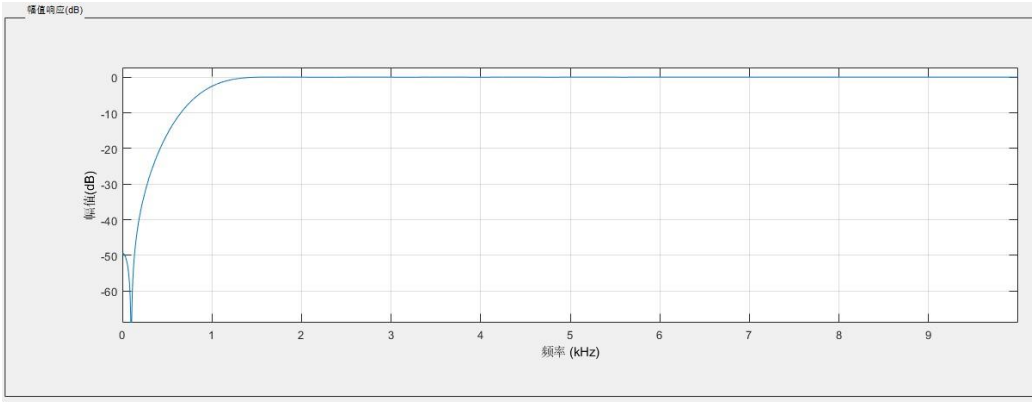

图 3.3 MATLAB 设计的滤波器的幅频特性曲线

滤波器各阶系数如图 3.4 所示

```
float coef[N]={-0.00991583305584109, -0.00435162063473924, -0.00492403469383765, -0.00522926375568833, -0.00516901123404181, -0.00464653932569515, -0.00357759145622985, -0.00188784817344092,
    0.00047737299500870 9, 0.00354552394963629, 0.00732058841294396, 0.0117689130755393, 0.0168326980481745, 0.0224069297205905, 0.0283871067424299, 0.0345922808733176, 0.0408669000838189,
    0.0470253839878939, 0.0528653339998438, 0.0581949259066030, 0.0628409806596799, 0.0666418683194186, 0.0694539000563188, 0.0711823293169716, 0.0717635876284712, 0.0711823293169716,
    0.0694539000563188, 0.0666418683194186, 0.0628409806596799, 0.0581949259066030, 0.0528653339998438, 0.0470253839878939, 0.0408669000838189, 0.0345922808733176, 0.0283871067424299,
    0.0224069297205905, 0.0168326980481745, 0.0117689130755393, 0.00732058841294396, 0.00354552394963629, 0.000477372995008709, -0.00188784817344092, -0.00357759145622985, -0.00464653932569515,
    -0.00516901123404181, -0.00522926375568833, -0.00492403469383765, -0.00435162063473924, -0.00991583305584109};
```

3.4 设备检查并启动集成开发环境

检查仿真器、C2000 DSP 实验箱、计算机之间的连接是否正确，确认无误后，打开计算机和实验箱电源，并进入集成开发环境 CCS。

3.5 编写 FIR 算法模块

3.5.1 FIR 滤波器编写原理

有限长的单位冲击响应滤波器（FIR）差分方程可表示为：

$$y[n]=\sum_{k=0}^{N}h[k]x[n-k]$$

其中，h是滤波器系数，x为输入的数字信号，y为 FIR 滤波器计算输出。N为滤波器阶数。由此可得，一个 N 阶的滤波器计算，需要 N+1 个滤波器系数，N+1 个数字输入，每得到一个y值，需要 N+1 次乘法以及 N 次加法。另外，N 阶滤波器需要保存当前的 N+1 个输入信号数值，以及事先设计的 N+1 个滤波器系数。

3.5.2 输入信号的更新

为实现滤波器输出的实时更新，每一个采样信号到来时，需要得知这个信号之前的 48 个信号，共 49 个信号。因此需要对输入信号进行前移，并将最新采样的数据存到最后，使得 49 个数据始终为按时间顺序排列的最新数据，实现的代码如下：

```
366     x= (AdcRegs.ADCRESULT1 & 0xFFF0);
367     int k;
368     for(ConvCount=0;ConvCount<48;ConvCount++)
369     {
370         xn[ConvCount]=xn[ConvCount+1];
371     }
372     xn[48]=x;
```

通过这段代码，前 48 个数据循环前移，并将最新的数据放入数组的最后一个位置。

3.5.4 FIR 滤波器代码

阅读程序，可以看到，数据采集和数据滤波都在中断函数“epwm1_timer_dac_isr”中实现。根据 FIR 滤波器差分方程，考虑数据右移等情况，反复测试调整之后，最后编写得到的 CCS 程序如图所示

```
interrupt void epwm1_timer_adc_isr(void)     //中断函数
{
    ConvCount = 0;
    adcInput0[ConvCount]=AdcRegs.ADCRESULT1>>4;

    FirRslt = 0;
    for(i=0;i<N;i++)
        FirRslt += (adcInput0[i])*coef[i];

    FIRout[III]=FirRslt;
    III++;
    if(III==49)
        III=0;

    tt=FirRslt;
    * Da_out=tt;

    for(i=N-1;i>=0;i--)
        adcInput0[i+1] = adcInput0[i];
```

3.6 建立工程并运行、调试程序

连接信号发生器至教学实验箱 SMA 输入端口 J2、教学实验箱 SMA 输出端口 J5 至示波器，编译链接工程并进入调试调试界面，运行程序后，查看输出是否正确，验证算法正确性。

3.7 算法实时性验证

测量采样频率与 FIR 算法的核心执行时间，判断该系统是否实时。若非实时，则优化程序甚至修改算法，直至满足实时要求。

4 实验结果及思考题回答

4.1 设计 FIR 滤波器

当输入信号为正弦信号时，改变正弦信号频率，观察示波器，记录各频点对应的幅度，并描点作图，与理论设计的幅频曲线比对，做误差分析，实际测量幅频曲线与理论曲线均需附在实验报告中，指出 FIR 滤波器系数的射极参数指标。

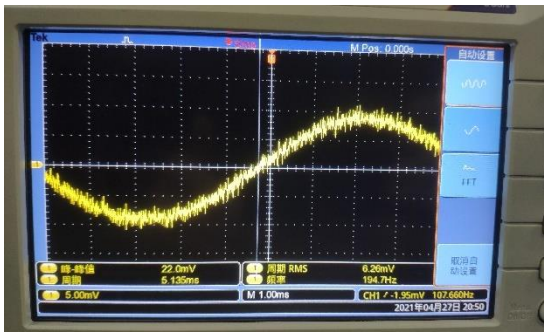

图 4. 1 100Hz 时的输出波形

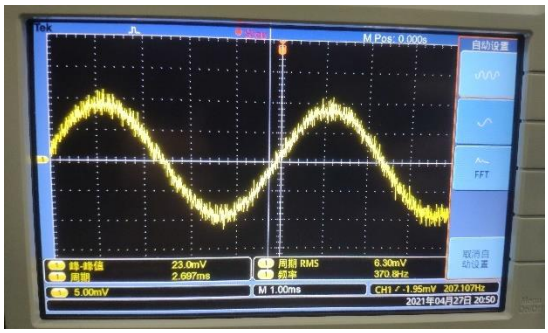

图 4.2 200Hz 时的输出波形

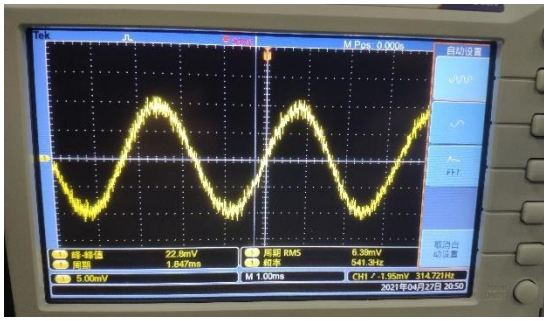

图 4. 3 300Hz 时的输出波形

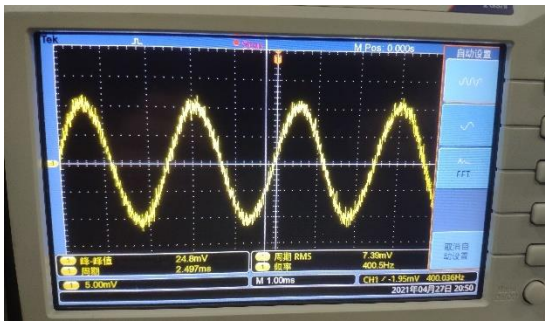

图 4. 4 400Hz 时的输出波形

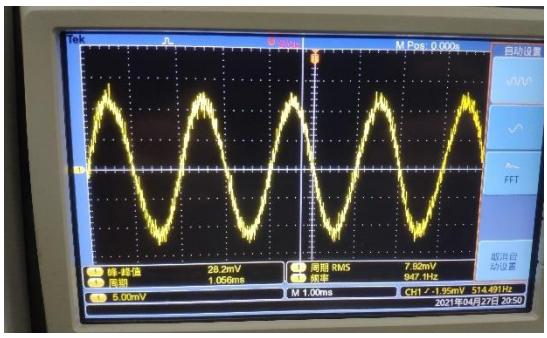

图 4. 5 500Hz 时的输出波形

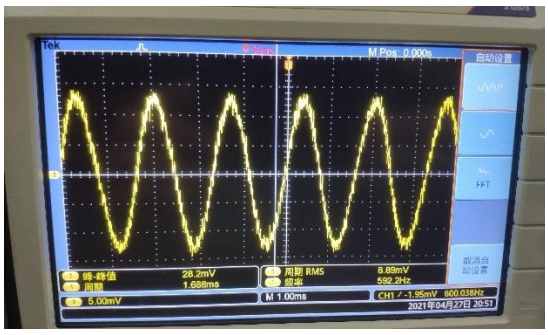

图 4. 6 600Hz 时的输出波形

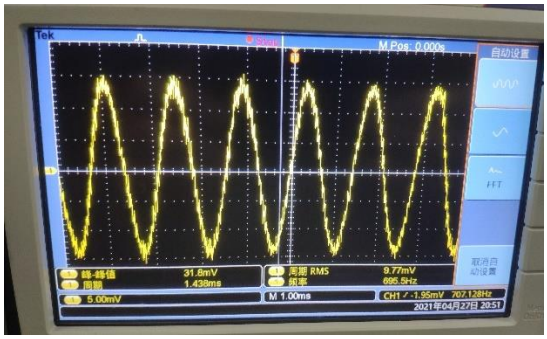

图 4. 7 700Hz 时的输出波形

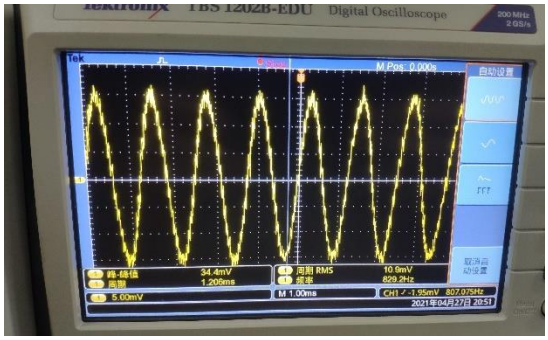

图 4. 8 800Hz 时的输出波形

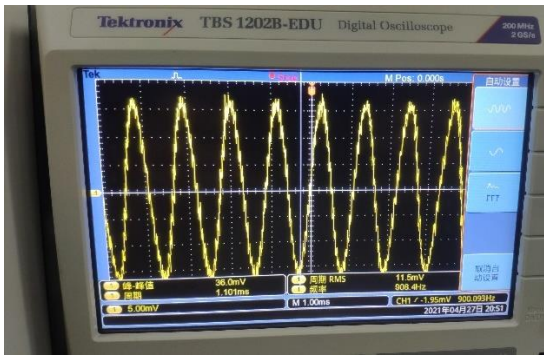

图 4. 9 900Hz 时的输出波形

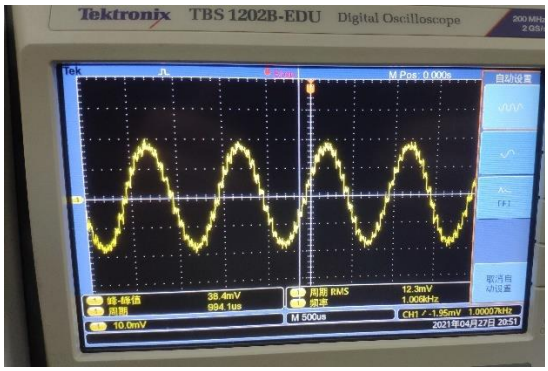

图 4. 10 1000Hz 时的输出波形

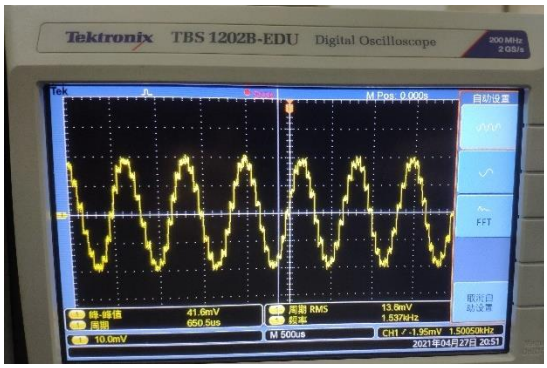

图 4. 11 1500Hz 时的输出波形

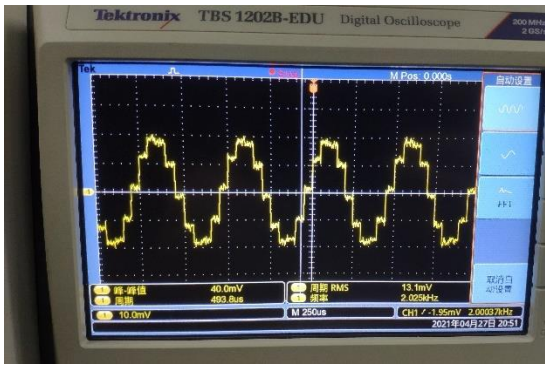

图 4. 12 2000Hz 时的输出波形

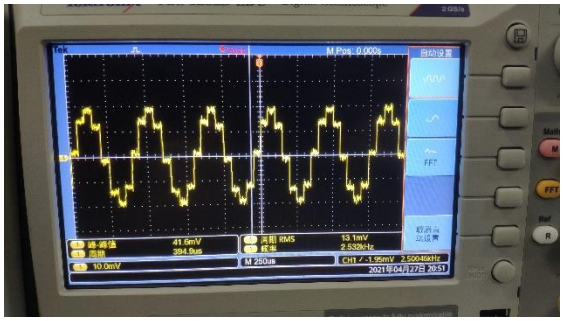

图 4. 13 2500Hz 时的输出波形

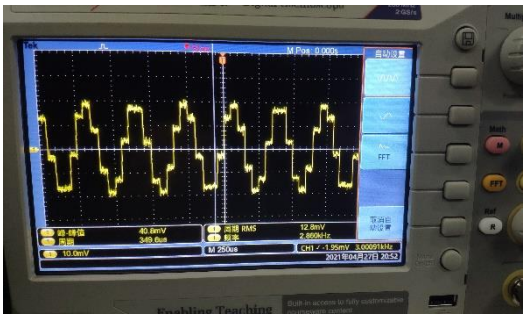

图 4. 14 3000Hz 时的输出波形

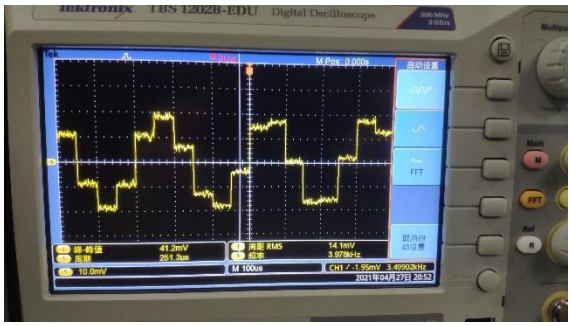

图 4. 15 3500Hz 时的输出波形

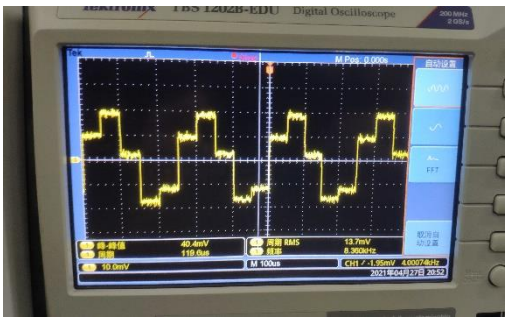

图 4. 16 4000Hz 时的输出波形

如图 4. 到**错误!未找到引用源。**所示，整理可得到**表** 4. 1，绘图，得到实际的幅频特性曲线如图 4. 所示。

表 4. 1 实际幅频情况记录表

频率(Hz)	幅度(mv)
100	20.6
200	21.2
300	22.4
400	23.8
500	26.2
600	26.4
700	28.6
800	33.6
900	37.5
1000	38.0
1500	40.2
2000	40.1
2500	39.8
3000	40.8
3500	39.4
4000	40.4

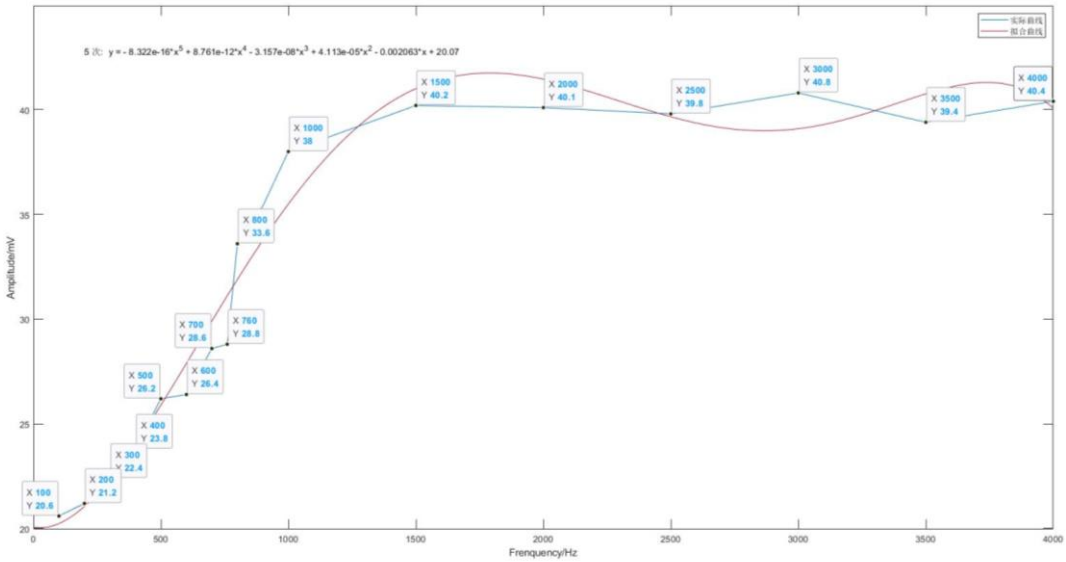

图 4.12 实际幅频特性曲线

4.2 验证算法实时性

记录 FIR 核心算法程序执行时间，以及采样时间，判断该系统是否实时。

在进入中断时给 DA 高电平，中断结束时给 DA 低电平，代码如下：

```
unsigned int property=10000
interrupt void epwm1_timer_adc_isr(void)     //中断函数
{
    * Da_out=property;
    AdIndex = 0;
    adcInput0[AdIndex]=AdcRegs.ADCRESULT1>>4;

    FirRslt = 0;
    for(i=0;i<N;i++)
        FirRslt += (adcInput0[i])*coef[i];

    FIRout[III]=FirRslt;
    III++;
    if(III==49)
        III=0;

//   tt=FirRslt;
//    * Da_out=tt;

    for(i=N-1;i>=0;i--)
        adcInput0[i+1] = adcInput0[i];

// Reinitialize for the next ADC Sequence
// Reset SEQ1
   AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;        //复位SEQ1
// Clear INT SEQ1 bit
  //EPwm1Regs.ETCLR.bit.INT = 1;            //清除中断标志位
   // 清除SEQ1中断标志位
     AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
// Acknowledge interrupt to PIE
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //PIEACK-PIE ackonwledge register   //中断应答
   * Da_out=10000-property;
   return;
}
```

可得到图 4. ，可知算法可以实时性。进一步用光标测量各时间，如图 4. 到**错误!未找到引用源。**所示，可知 FIR 运行所需时间为 20 微秒，采样时间为 50 微秒，算法时间小于采样时间且有一定裕余，满足实时性要求。

图 4. 13 验证算法实时性

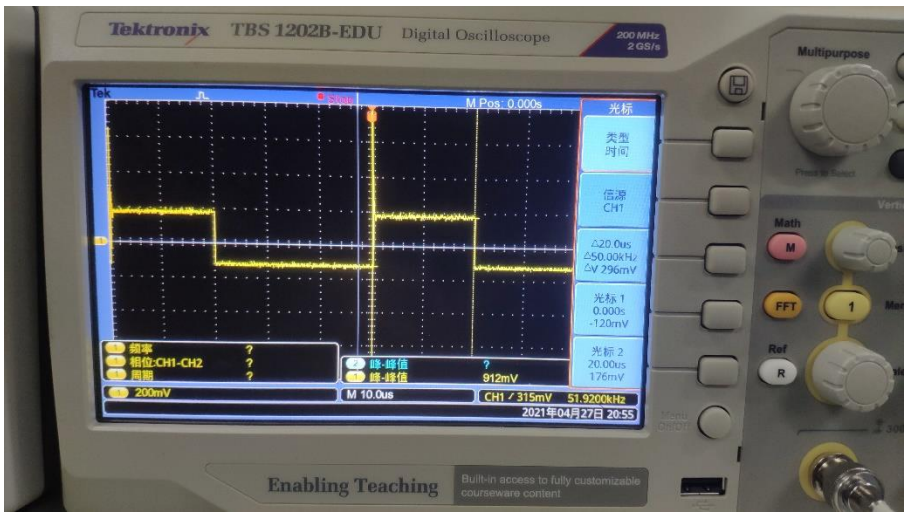

图 4. 14 测算采样时间

4.3 观察各种输入信号通过数字滤波器的波形

由于本滤波器是高通滤波器，因此方波，三角波等波形在通过本滤波器后，会出现不同程度的失真。这是由于这些波是由各种不同频率的正弦波叠加而成，因此在滤波后，只剩下了高频部分，低频部分被滤除。

4.4 加载由汇编语言编写的 FIR 滤波器的程序

由于一开始未能找到汇编语言的 FIR 滤波器程序，因此寻求老师帮助后得知，由汇编语言编写的程序，效率更高，程序运行时间会更短，实时性会更好。同时老师打开了 CCS 自动编译的汇编程序，发现由 C 语言编译转换为汇编语言后，会出现很多冗余成分，进一步佐证了汇编语言的高效。

4.5 总结分析系统实时性的取决因素

1、算法的高效，简洁

2、硬件的运行效率，如芯片速度、AD、DA 的采样时间

3、数据率的要求

5 实验总结

5.1 实验中遇到的问题及解决方法

1.程序运行后示波器无输出

在程序编译后示波器始终只有杂波输出，为了验证是否为硬件的问题将实验十一的程序烧入发现显示波形，因此怀疑是软件问题。排查程序的过程中经询问得知，对于 h[k]中的采样值而言，需要将里面的数值均右移四位才能得到想要的波形，这和滤波器采样区间有关，在添加了右移代码后重新编译还是显示不出波形，重新检查程序也发现可能有错误的地方，一度陷入困境。后来再次请教同学后得知可能是实验十二的工作环境已经损坏，将实验十二的主函数代码覆盖到到实验十一的主函数后编译实验十一，示波器最终显示出波形。

2. 实际截止频率与理论截止频率相差过大

在滤波器设计过程中我们发现并非所有参数的滤波器都得将误差控制在 5%

以内，这和电路结构有关，在放大信号的过程中会加大这种差异，而且由于示波器光标调节精度不够导致读数误差很大，很容易导致所设计出来的滤波器完全达不到理想效果，在反复实验调整参数后最终得到出来的实际截止频率大概为760Hz，与理论值 800Hz 之间相差 40Hz，误差为 4%。

5.2 实验心得体会

本次实验由于没有示例程序，因此需要自己对实验三的程序进行修改，以实现功能要求。因此难度相比前面的实验高了很多。

在本次实验中，我深刻感受到了从理论到实践的艰难历程。本次实验的要求很简单，只是设计一个高通滤波器。这些原理和理论知识都是之前熟练掌握的。但是由于经验的缺失，让我们完成的速度远远没有预想的那么快。而对于 DSP 采样仍一知半解，导致我们对采样数据的处理不够熟练。我们花了很久的时间才想明白为什么数据应该右移 4 位，这是由于采样是 12 位采样并保存在高 12 位中，因此 16 位的存储空间需要将数据整体右移四位。在搞明白了这一点后，我们才最终在示波器上完成了 DFT 算法的波形。

在这次实验中，由于既要验证采样频率、又要验证算法实时性，因此采用了与上次实验不同的验证方案，即在进入中断程序的一开始给 DA 高电平，中断中的程序照常执行、但不赋值给 DA，中断程序的最后一条语句给 DA 低电平。在这种方案下，产生的方波周期就是采样频率的周期；而通过查看高电平的持续时间是否足够短，可以验证系统实时性。

总之，经过本次实验，对 DSP 的认识更深了一步，同时我意识到了只懂理论是万万不行的，一定要能够将理论应用到实践中才算是真知。

最后，也很感谢老师对我们一次次的帮助，老师的帮助每次都让我们受益匪浅。也很感谢组员的努力付出，是我们一起实现了整个算法，大家一起做到了第一个完成任务！