**实验四 定点 DSP 程序设计优化**

**王亭午**

**无210**

**2012011018**

1. **实验目的**

**1.1 掌握有限字长效应的解决方案,提高系统信噪比,保证系统稳定性**

**1.2 学习基于 DSP 结构的程序设计的速度优化方法**

**1.3 验证滤波器结构对系统性能的影响( IIR )**

**1.4 学习 FFT 的汇编编程方法**

1. **实验材料**

**2.1 集成开发环境 IDDE —— ADI 公司的 VisualDSP + + 4.5**

**2.2 实验用源程序**

FIR 滤波例程\Analog Devices\VisualDSP 4.5\Blackfin\Examples\Tutorial \fir; IIR 滤波例程:网络堂课程文件中提供的 IIR\_df1\_cascad; FFT 例程:网络学堂课程文件中提供的 FFT\_8; 通过例程中的引用关系,从库函数中找到调用的 IIR 和 FFT 源程序

**2. 3 阅读内容**

a) 第四讲课件——《程序设计优化》

b) 本实验指导书及其 Blackfin 库函数说明

c) 请充分利用软件集成环境的帮助系统

1. **实验内容**

**3.1 任务一：基于 DSP 结构的 FIR 滤波程序优化**

**3.1.1实验步骤**

(1) 新建一个工程目录,将第二、三次实验中都使用过的 fir\_test. c 和 fir\_Smac\_asm. asm或myfir\_Smac\_asm. asm 导入。

(2) 参考\Analog Device\VisualDSP 4.5\Blackfin\Example\Tutorial\fir目录中fir\_test.c和fir.asm.

(3) 重新改写 fir 汇编程序,命名为 myfir\_Dmac\_asm.asm

(4) 保证改写的程序正确后,进行性能对比

(5) 重新改写实验二中自己编写的 myfir.c ,命名为 myDfir.c。对比改写前后的性能,选择编译器选项。

答案：我们改写的编myDfir.c和myfir\_Dmac\_asm.asm代码分别如下:

|  |
| --- |
| **LSETUP (FIR\_Loop\_START, FIR\_Loop\_END) LC0 = P0>>1;**  **output\_loop\_start: R1.H = W[I0++];**  **W[I3] = R1.H;**  **A0 = 0 ; A1=0;**  **LSETUP (DMAC\_Loop\_START, DMAC\_Loop\_END) LC1 = P3;**  **D\_inner\_Loop\_start: R0.H = W[I2++]; nop; R0.L=W[I2];**  **R1.H = W[I3--];**  **D\_inner\_Loop\_end: A0 += R1.H \* R0.H , A1 += R1.H \* R0.L;**  **// simultaneously caculate two cofficient**  **R0.H = W[I2++];nop;R0.L=W[I2];**  **R1.H = W[I3--];**  **A0 += R1.H \* R0.H;**  **R1.L = A0 ;**  **W[I1++] = R1.L;**    **R1.H = W[I0++];**  **I3 += 2;**  **A1 += R1.H \* R0.L;**  **R0.H = A1;**  **W[I1++] = R0.H;**  **W[I3] = R1.H;**    **output\_loop\_end: I3 += 2;** |

|  |
| --- |
| **void \_myDfir(const fract16 \*IN, fract16 \*OUT, int N, fir\_state\_fr16 \*s) {**  **int i, j, k, temp1, temp2, tapL;**  **for ( i =0; i < N; i++) {**  **tapL = (s->k > i) ? (i + 1) : (s->k);**  **temp1 = 0;**  **temp2 = 0;**  **for( j = 0; j < tapL; j++) {**  **temp1 += IN[i - j] \* (s->h)[j];**  **k = (j + 1) % 8;**  **temp2 += IN[i-j] \* (s->h)[k];**  **}**  **OUT[i] = (temp1 + 0x3FFF) >> 15;**  **i++;**  **OUT[i] = (temp2 + 0x3FFF) >> 15;**  **}**  **}** |

核心思想是在一个循环中同时进行多个运算，这样就可以节省循环的数目，降低运算复杂度。这点对于可以同时进行两个MAC运算的DSP非常有利。

**3.1.2 实验问题**

**问题 1.1 对比 fir.asm, myfir\_Smac\_asm.asm 和 myfir\_Dmac\_asm.asm 的输出结果。**

**答案:**输出的结果如下，可以看到我们的结果是一模一样的，通过后面的作差比较，我们知道结果是一模一样的。

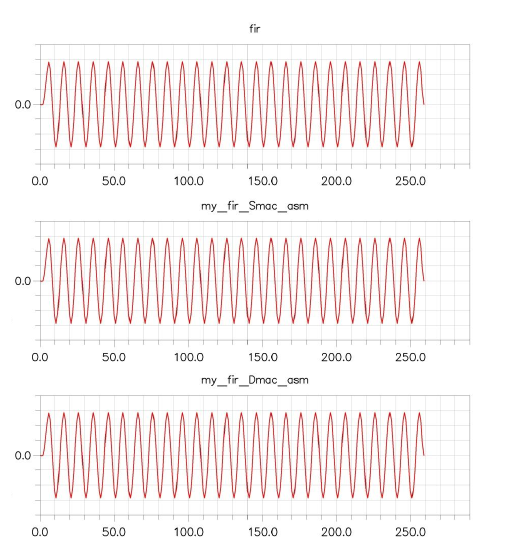


图1，输出结果图

**问题 1.2 对比 fir.asm，myfir\_Smac\_asm.asm 和 myfir\_Dmac\_asm.asm的执行效率。**

**答案：**这三个函数的执行结果如下，图片比价小，但是可以放大。

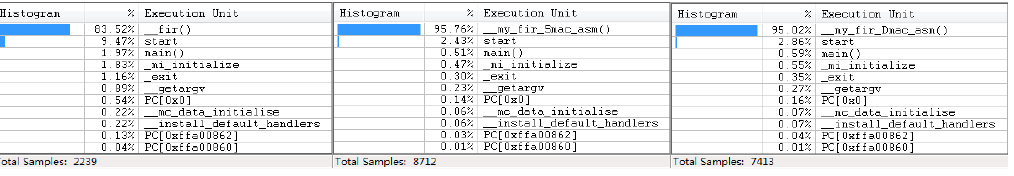


图2，执行效率图

可以看出，运行效率上自带的fir. asm效率最高，myfir\_Dmac\_asm.asm次之，myfir\_Smac\_asm.asm最差。这和我们的预期相符。

**问题 1.3 描述你的改写思路和评价。**

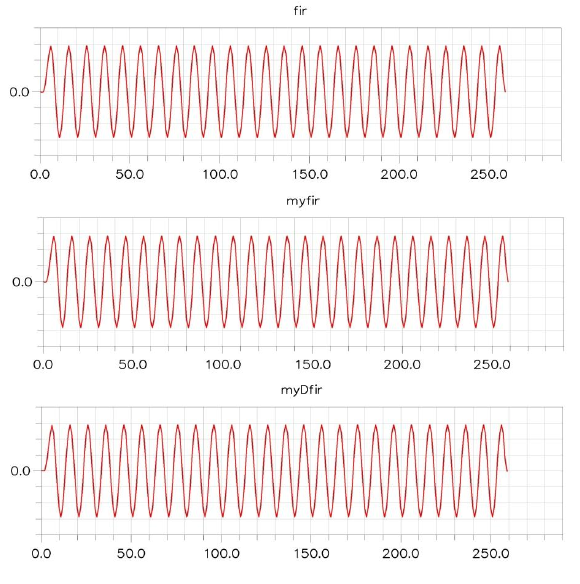
**答案：**改写方法很简单，就是使用并行的MAC。关键代码是：A0 += R1.H \* R0.H , A1 += R1.H \* R0.L;

不过根据我们的执行效率图，改写后并没有数量级的差别，不过也挺高了很多。

**问题 1.4 对比 fir.asm，myfir.c 和 myDfir.c的输出结果,误差各是多少？**

**答案：**结果图如下，通过作图可以看到结果。经过作差，我们发现误差为0。

图3，fir.asm，myfir.c 和 myDfir.c的输出结果



**问题 1.5 对比 fir.asm，myfir.c 和 myDfir.c 的执行效率。**

**答案：**几个函数的执行效率如下：

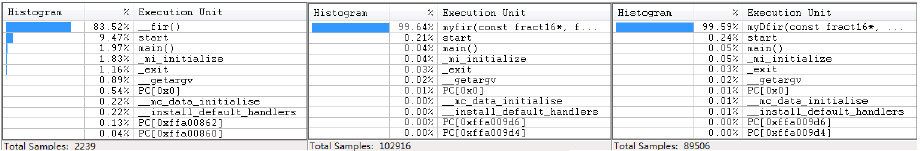


图4，fir.asm，myfir.c 和 myDfir.c 的执行效率。

通过对比可以看到，fir.asm最好，改写后的代码稍微比原来好一点。

**问题 1.6 选择打开编译器优化和关闭编译器优化，对比两种情况下 myfir.c 和 myDfir.c 的执行效率。**

**答案：**结果如下：

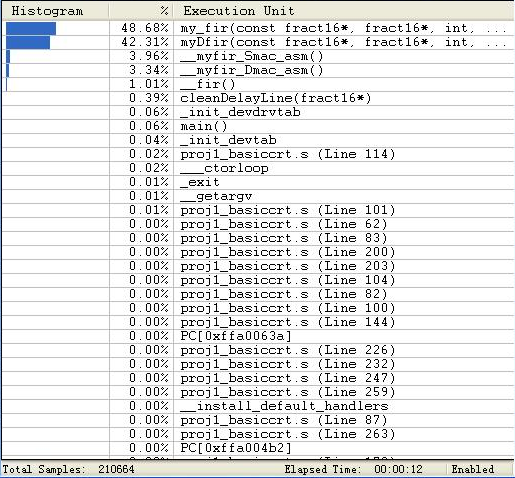


图5，打开编译优化前的执行效率。

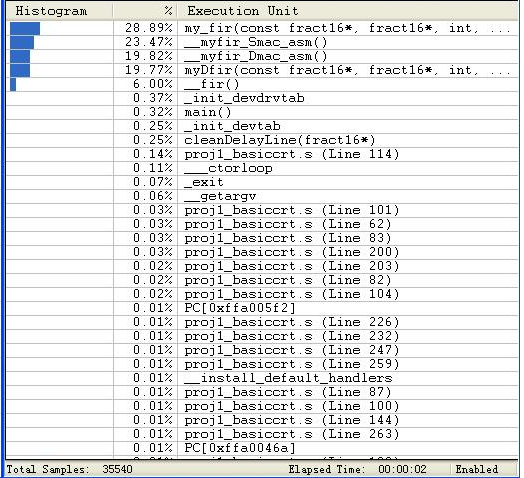


图6，打开编译优化后的执行效率。

在上面的编译中，我们都同时把几个函数全部编译进去方便对比。可以看到，打开编译优化后，程序的运行时间有了数量级的变化。同时我们的改写后的c代码相比老的c代码得到了更加多的优化。

**问题 1.7 描述你的改写思路和评价。**

**答案：**改写的方法和汇编一样，就是在一个循环中同时计算两个参数。不过显然，因为C中不存在所谓的并行MAC概念，提升的效果并不是特别明显。

**3.2 任务二：设计 IIR 滤波应用程序对比定点计算精度**

**3.2.1实验步骤**

(1) 下载网络学堂提供的 IIR\_df1\_cascad,是采用级联形式结构实现的滤波器。

(2) 建立 IIR\_df1\_cascade 工程,并记录运行结果和系统性能。

**3.2.2实验问题**

**问题 2.1 用双音多频( DTMF )信号做为输入,执行程序并分析这个滤波器对 DTMF 滤波**

**输出的精度(和 Matlab 计算结果对比)。**

**答案：**

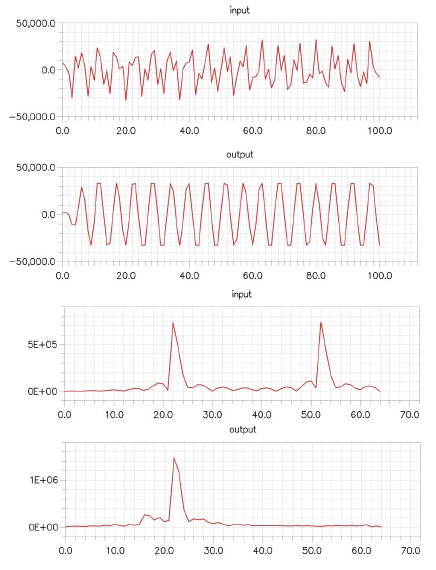


图7，输出结果图。

可以看到，我们分别看到两个不同信号输出的时域和频域结果。通过频域的对比，我们应该可以知道，这是一个带通滤波器（或者低通滤波器，单从这个波形还看不出来）。

接下来我们把输入输出的数据dump出来放到matlab上进行分析。

关键代码如下：

|  |
| --- |
| **% these are the coefficients for the filters**  **a1 = [1,-0.6645,0.8076];**  **a2 = [1,-0.8839,0.6449];**  **a3 = [1,-1.2399,0.8438];**  **b1 = [1,0,-1]/1.7544;**  **b2 = [1,0.322,1]/1.7544;**  **b3 = [1,-1.7544,1]/1.7544;**  **w1 = filter(b1,a1,x);**  **w2 = filter(b2,a2,w1);**  **matlab\_results = filter(b3,a3,w2);**  **% calculate the error**  **delta=(dsp\_results - matlab\_results) ./ matlab\_results;**  **plot(delta);** |

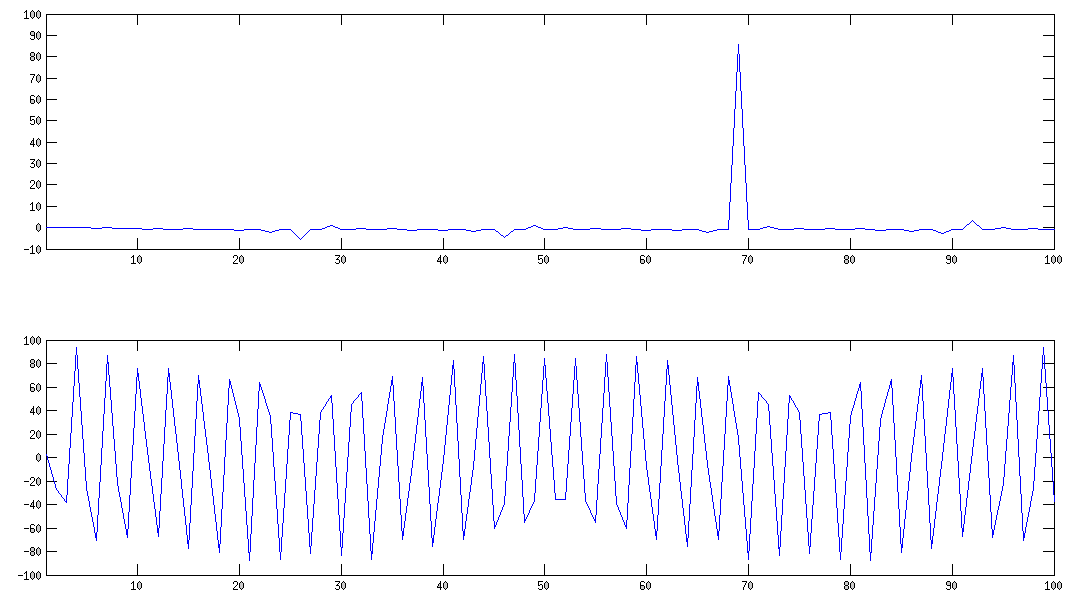


图８，输出结果图

可以看到，这个结果图显示出来，滤波器实际上是并没有这个功能的。考虑到我们的matlab函数实际上是不可能会随便出错的，我们应该想到是dsp程序出现了大的问题。我们注意到dsp的结果中存在大量的32767和-32786，我们猜测，dsp程序出现了大量的切顶现象！

因此我们在dsp实验中重新运行，不过这个时候我们把代码中的数量先除以20，尝试用降低精度的方法来解决切顶的现象，如下：

|  |
| --- |
| **/\*To avoid the overflow of 32767 and -32786\*/**  **int i = 0;**  **for(i = 0; i<= 260; i++) {**  **IN[i] = IN[I] / 20;**  **}** |

在结果处我们把输出再进行复原：

|  |
| --- |
| **/\*To avoid the overflow of 32767 and -32786\*/**  **int i = 0;**  **for(i = 0; i<= 260; i++) {**  **OUT[i] = OUT[I] \* 20;**  **}** |

这个时候我们再次运行程序，我们的结果和我们的matlab中就是几乎一样的了。

**问题 2.2 结合先修课程学过的滤波器结构和有限字长效应两部分内容,分析并解释这个滤波程序的问题,试着提出修改建议。**

**答案：**在我们的DSP程序中，有限字长效应非常的明显，因为我们的几个系数在进入到dsp程序之后都出现了精度的截断。这个时候如果我们不使用级联结构反而会更加好一点，因为级联的话，精度上的误差会更加的大。不过这里的程序出问题更加主要的原因是因为切顶。

为了让我们的程序能够正常运转，我们应该尝试把输入按照系数放小，在输出再恢复。这个我们已经在上一问中完成了。

**3.3 任务三 利用汇编完成 8 点 FFT**

从网络学堂中下载例程 FFT\_8,执行并理解以下问题

**3.3.1 实验问题**

**问题 3.1 复习 FFT 计算中的倒位序问题,请对比解释 DSP 对 FFT 计算的优势**

**答案：**FFT中最重要的两个功能是反向计算地址和进行乘法累加。我们使用的 DSP 处理器专门有反位序间接寻址，同时可以在一个指令周期内完成乘和累加的运算。这两点使得它特别适合进行FFT的计算

**问题 3.2 实现倒位序取址的功能单元在 DSP 结构中的什么位置?**

**答案：**是我们使用的M寄存器。

**问题 3.3 FFT 的三层嵌套循环是怎样实现的?**

**答案：**这个就写在实验报告书里面了。上面是这么说的：

根据以上特点,所以整个 L 级递推过程由三个嵌套循环组成。外层的循环控制 L级的顺序运算;内层的两个循环控制同一级的各个碟形运算,其中最内层控制同一种碟形运算,中间一层控制不同种碟形运算。

**问题 3.4 分析 FFT 计算的精度、速度和动态调整方法——用 FFT 汇编程序结果与 Matlab结果对比,分析是否有误差，如果有,采用分级调整的方法进行优化。**

**答案：**误差的原因是精度使用的不同。其中Matlab在进行fft的时候可以使用double型数据（32位）或者single（16位）。而我们的dsp则是使用的8位的fft。为了分级调整，可以在计算结果中逐级进行放缩。

我使用了一份学长提供的测试数据，得到的结果如下：

|  |
| --- |
| Out\_MATLAB = [  0.0062, 0.0187, 0.0375, 0.0625, 0.0875, 0.1125, 0.1375, 0.1625,  0.1875, 0.2124, 0.2374, 0.2624, 0.2874, 0.3124, 0.3374, 0.3624, 0.2812, 0.1937, 0.1000, 0.000, 0.000, 0.000, 0.000] |

|  |
| --- |
| OUT\_DSP = [  0.0059, 0.0195, 0.0391, 0.0645, 0.0859, 0.1113, 0.1348, 0.1602,  0.1855, 0.2109, 0.2363, 0.2617, 0.2852, 0.3105, 0.3359, 0.3594, 0.2813, 0.1914, 0.0917, -0.020, 0.000, 0.000, 0.000] |

可以看到，整体的误差是比较小的。

1. **实验总结**

**4.1 总结实验中遇到的问题和解决办法**

**答案：**最大的问题是在实验二中，因为一开始做出来的结果肯定是错误的。而这个时候我完全没有意识到这个错误的结果才是正常的，因此花了很长的时间寻找matlab上面可能的程序错误。事实上这里就是需要分析错误的原因，因此我浪费了很多时间。其次的问题在于汇编代码，个人感觉这次实验中的汇编代码难读都比较高，花了很长的时间来理解和消化。尤其是最后的fft算法的汇编代码，不花很长的时间是很难理解的。个人认为，这段代码是面向已经学习好了这种汇编语言的人的，而不是面向初学者的，因此注释晦涩难懂。

另外一个比较蠢的问题是，在使用plot的时候选错了地址，因此有一段时间，我的OUT的结果都是0。

**4.2 本次实验体会与建议**

**答案：**本次实验最大的体会就是，当程序的结果和预想不一样的时候，有可能这个不一样才是正常的情况。这次实验，我们大家都卡在了这个地方，都以外是出现了代码编写上的错误。实际上则是我们课件没有理解好，因此没有注意到这个问题就是要考验我们对dsp运算中非常重要的切顶和有限字长的理解。

其次就是，混合编程上面的问题。虽然这一次的matlab和dsp的混合并不严格，实际上也是分割的两个部分，不过还是锻炼了我们不同平台上运用代码的分析的能力。

最后就是，做实验的时候应该多思考，不要盲目的进行debug。这样可以节省我们非常多的时间。