信号处理实验与设计

课程设计 基于 CVSD 编码的语音信号网络传输
——MCVSD 语音信号编解码器设计

无 84 吴文昊 2008011146

同组人：无 84 王梦娣 2008011134

无 84 张舸 2008011133

1 设计原理

作为一种自适应增量调制（ADM）算法，连续可变斜率增量调制（Continuously Variable Slope DM-CVSD）的基本原理是根据信号的平均斜率，自适应地连续调整量阶，用1 bit对差分信号进行量化。对量阶的自适应调整有利于克服增量调制中的过载失真和颗粒噪声，其公式如下。

$$\delta(n)=\delta(n-1)+k(n)\delta_0$$

其中$k(n)$为三阶斜率检测输出：

$$k(n)=\begin{cases}1, & b(n)=b(n-1)=b(n-2)\\ 0, & otherwise\end{cases}$$

CVSD具有较强的抗噪声和抗误码性能，可以提供质量较好的语音质量，实现也比较简单，但对输入信号的码率要求较高。在本次实验中，由于DSP开发板的AD采样率可以达到48kHz以上，量化位数在16bit以上，完全满足CVSD算法对码率的要求，因此我们在实验中选用CVSD作为基本的语音编解码算法。

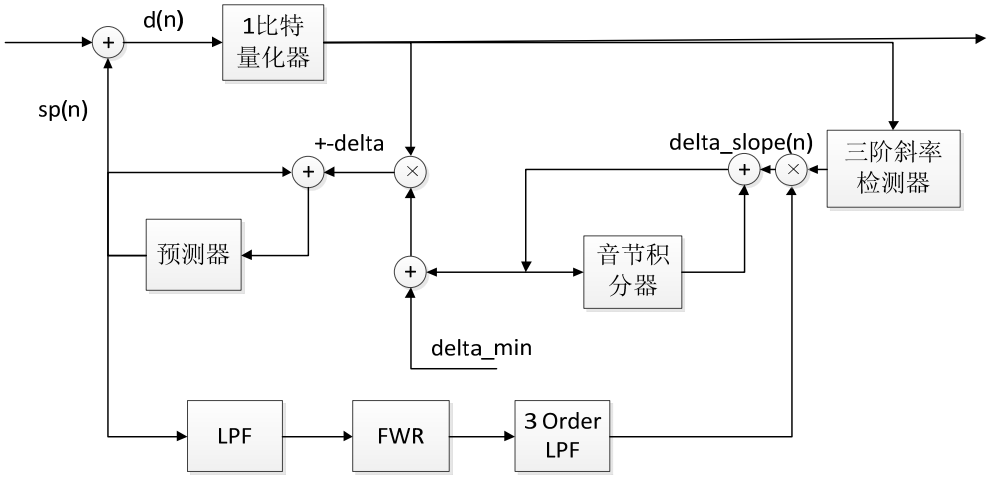
然而，在先期的算法仿真中，我们发现基本的CVSD算法还具有动态范围小的不足。在输入信号幅度变化较大的情况下，CVSD算法仍然会产生较大的过载失真和颗粒噪声，导致其不能忠实地再现输入波形。MYTRI等人提出了一种改进的CVSD算法（MCVSD）。采用根据重建信号的包络调整量阶的变化步长的方法，将动态范围提高了15-20dB。量阶的计算公式变为：

$$\delta(n)=\delta(n-1)+k(n)\delta_0(n)$$

其中

$$\delta_0(n)=p(n-1)\delta_0$$

其中$p(n-1)$为将重建信号经过低通滤波器-全波整流-低通滤波器后得到的瞬时包络。我们在MCVSD算法的基础上，针对DSP实际应用中的一些特点，如稳定性、定点DSP的有限字长效应及可能产生的溢出等问题，对编解码器进行了重新设计。编码器框图如下：



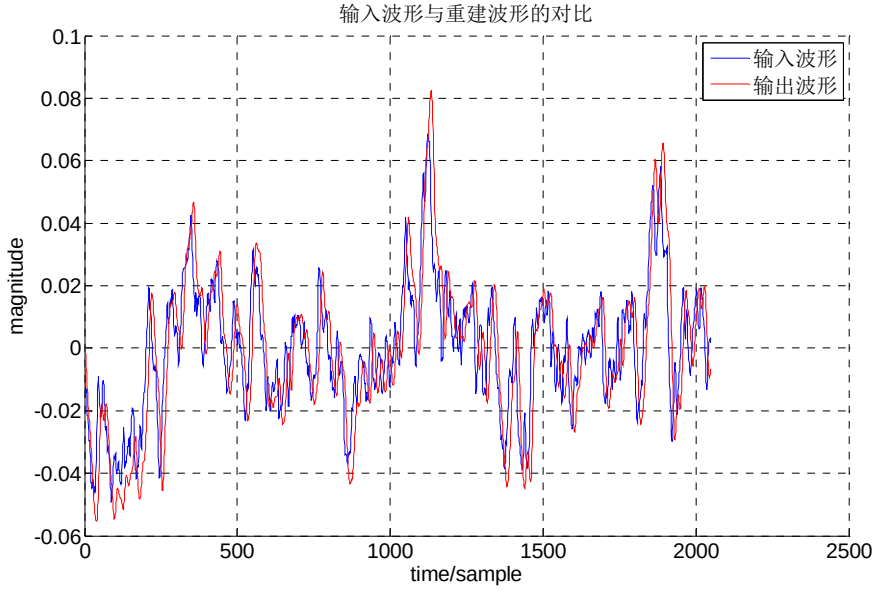
解码器基本是编码器的逆系统，只是对重建波形sp(n)进行了一次低通滤波进行输出以减少量化噪声，其框图从略。

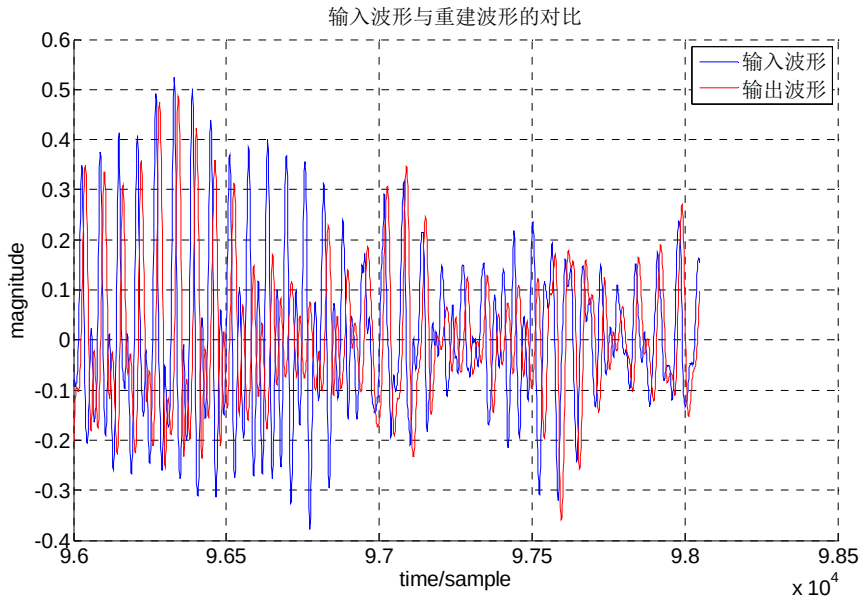
2 理论分析和设计计算

为了对MCVSD编解码算法进行功能验证，并根据实际语音样本的编解码输出进行参数优化，我们首先使用MATLAB对其进行了仿真。根据仿真结果，我们主要确定了MCVSD编解码器中各个滤波器的长度及系数，如下表。

alpha (1阶音节积分器系数)	0.995
beta（1阶预测器系数）	0.75
delta0	0.0035
delta_min	0.001
包络检测支路的LPF	长度为18的Hamming窗
包络检测支路的3-order LPF	长度为4的Hamming窗
解码器去量化噪声LPF	长度为30的Hamming窗

同时，仿真结果也证明MCVSD能够在很大的动态范围内较好地重建输入波形，如下图。





由于 CVSD 编解码过程中的主要运算量来自于各个滤波器，由前述设计可知，这些滤波器的总阶数大致为 50 左右。取帧长为 1024，则在一次子程序调用中，外层循环需进行 1024 次，

循环内部的运算量大致相当于进行 50 次乘累加操作。故对 1024 个样点进行一次编码或一次解码操作的时间开销大致为 1024×50×(3+1)=204800 (cycle)。而在 Process_data()函数中，由于左右声道需要分别进行编解码，故以上开销还需要乘以 4（程序结构见下文）。在仿真中，实际对 1024 个样点进行一次编码和一次解码所用的开销为 502284 (cycle)，与以上分析基本相符。由于 BF561 的时钟频率是 580Mcycles/s，可 Process_data()函数单次运行时间大致为 1.73ms。而采样采满一个缓存区的周期为 1024/48kHz=21.3ms，故 DSP 可以完成实时处理。

本程序对内存使用要求不高，使用片上缓存就能满足设计需要。

3 设计及其验证——编解码器的汇编实现

为了提高程序的运行速度，我们采用ADI Blackfin Processor的汇编语言实现了MCVSD的编解码器。编解码器的函数原型分别定义如下：

```
void _code_mcvsd(const fract16 x[],fract16 y[],int n, const fract16 delta0, const fract16 delta_min,
const fract16 alpha, const fract16 beta, fir_state_fr16 *s1, fir_state_fr16 *s2, const fract16 *sr, const
fract16 *delta, const fract32 *d, const int step);
void _decode_mcvsd(const fract16 y[],fract16 x[],int n, const fract16 delta0, const fract16 delta_min,
const fract16 alpha, const fract16 beta, fir_state_fr16 *s1, fir_state_fr16 *s2, const fract16 *sr, const
fract16 *delta, const fract32 *d, const int step, fir_state_fr16 *s3);
#endif
```

各参数说明如下：

_code_mcvsd	
参数	说明
x[]	输入音频采样序列，长度为n
y[]	输出音频编码序列，长度为n/16
n	每一帧输入音频序列的采样点数
delta0	量阶的基本变化步长
delta_min	最小量阶
alpha	音节积分器（1阶IIR）系数
beta	预测器（1阶IIR）系数
s1	定义包络检测支路输入端LPF的滤波器结构体
s2	定义包络检测支路输出端3阶LPF的滤波器结构体
sr	预测器反馈延时单元（长度为1）
delta	音节积分器反馈延时单元（长度为1）
d	记录编码结果历史的移位寄存器
step	输入采样序列的步长（双声道处理用）
_decode_mcvsd（其他参数的意义与code_mcvsd相同）	
参数	说明
y[]	输如音频编码序列，长度为n/16
x[]	输出音频采样序列，长度为n
s3	定义解码器输出端去量化噪声LPF的滤波器结构体

编解码汇编程序具有如下特点：

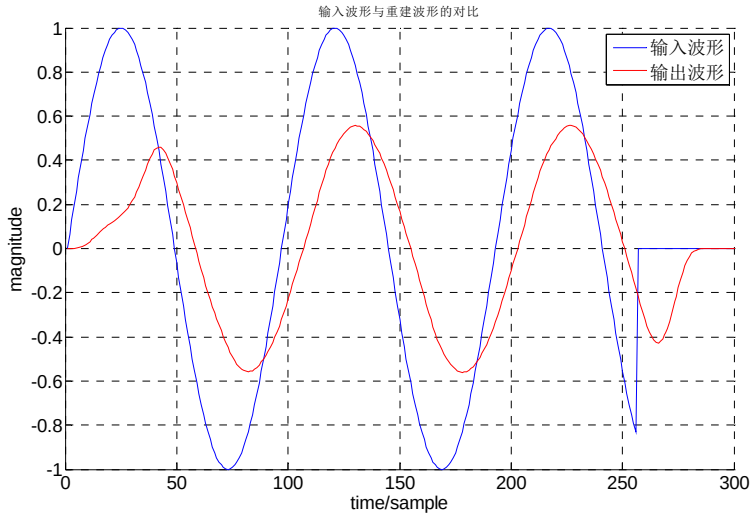
- 由于程序参数很多，在编解码其中使用了LINK和UNLINK指令组织堆栈以传递参数。
- 由于编解码程序中有多个FIR滤波模块，因此在堆栈基址指针FP后开出了一段本地变量空间，分别存储各个FIR滤波模块的DAG寄存器（延迟线读、写，滤波器系数读）的值。
- 程序的基本结构为两层硬件循环。每一次外层循环完成一次编码或一次解码，并多次使

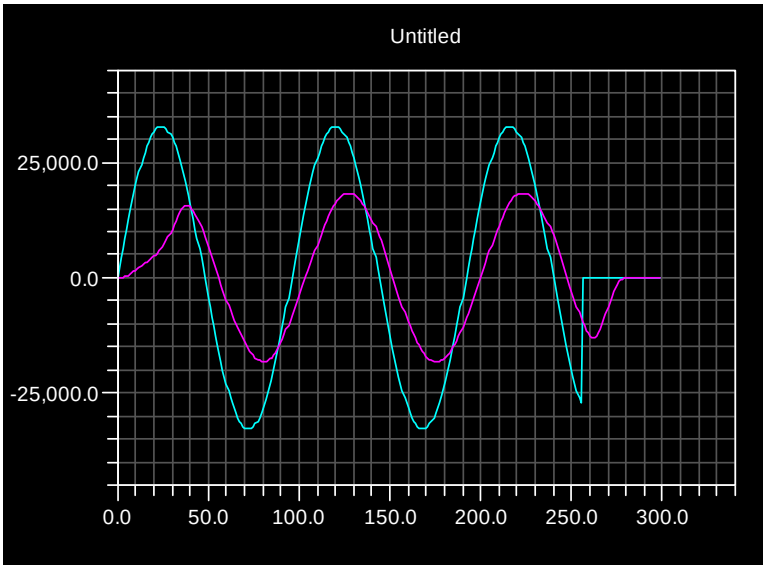
用内层硬件循环实现各个FIR滤波器。在编码器中，每一次外层循环读入一个采样值，编码结果通过移位依次存入数据寄存器中；每完成16个输入采样的编码，则将数据寄存器中16位编码结果以一个word的格式存入内存中。在解码器中，每完成16个编码结果的解码，则从内存中读一个word作为16位编码输入到寄存器中，并通过移位依次读出每一个编码结果并进行解码。

- 对各个FIR滤波器的系数进行了幅度加权，以避免小数的乘累加运算造成不必要的溢出或饱和。
- 合理地使用了各个寄存器，将最常用的参数保存在寄存器中，避免频繁访存。
- 为了克服多周期指令对流水线的不利影响，对各个访存指令进行了调度。

4 设计成果展示

我们首先参照第三次实验建立了一个简单的工程，对编解码器的基本功能进行了软件仿真验证。通过对一段500Hz的正弦波进行编解码，并将结果与MATLAB所得结果进行比较，可以看出汇编程序的功能是正确的。

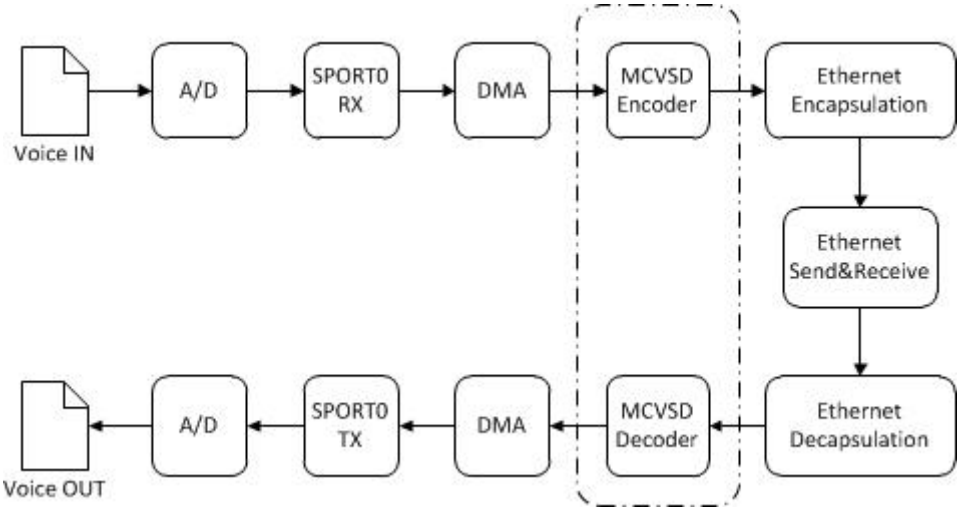




接下来我们参照第五次实验建立了一个工程并在开发板上进行了试运行。我们在实验五的基础上，将Process_data函数中对fir的调用换成了对编解码函数的调用，从而实现了对语音信号实时的编解码。实验结果表明，编解码函数在开发板上实现了预期的效果。不仅可以得到可懂的

语音编解码输出，而且即使在一些波形非常不规则的音频输入条件下（选用了芬兰哥特金属乐队Nightwish的单曲《Sacrament of Wilderness》），依然能够得到较好的音质。如果输入是波形比较规则的乐音（选用了Vivaldi的《双吉他协奏曲》中的行板），在没有和弦的情况下几乎觉察不到量化噪声，而在有和弦的情况下（类似于双音多频）也只有轻微的噪声。

最后我们将MCVSD编解码器集成到我们的整体设计中。其在整体结构中所处位置如下图表示



5 设计总结

我们在 DSP 上成功实现了 MCVSD 的实时编解码。运行结果充分表明了 MCVSD 算法的高动态范围、低量化噪声的特点；在 MATLAB 仿真阶段所进行的参数优化保证了最终得到满意的运行结果；而通过对指令执行周期数的观察和分析，可以发现汇编程序具有很高的效率。

MCVSD 编解码部分的设计难点主要在于汇编程序的编写。在开始编写代码之前，需要合理分配 DSP 的内部资源，规划好各个数据寄存器、指针寄存器的用途；同时也要对程序流程有一个清晰的认识，在编写代码的过程中采取循序渐进的方式，先按照 MATLAB 仿真程序进行移植，实现基本功能后再对本地内存和寄存器的使用进行精简、对指令进行优化调度；编写过程中还要随时注释或作笔记，以提高汇编程序的可读性，为后续的 debug 提供便利。

总而言之，通过这次实验，我对数字信号处理在实际应用中的一些问题有了更加深入的了解，对于 Blackfin Processor 的结构和使用也有了进一步的认识。通过实际的代码编写和调试，我也极大地锻炼了自己的编程技巧。而在与同学的合作中，在各个函数、功能模块间的接口的协商以及对总体程序的综合设计中，我的团队合作意识也得到了加强。

6 对整个课程的意见和建议

半学期的《信号处理设计与实验》给我带来了很大的收获。然而，由于数字信号处理涉及的知识基础和应用场景非常广，很难在半学期的课程内涵盖大量的课程内容。而为了保证对 DSP 的深入理解，课程中很大一部分涉及 DSP 硬件结构的内容又是必不可少的。这就可能导致同学们把过多的精力放在对 IDDE、处理器内核、开发板的了解上，而不是培养利用 DSP 去解决实际问题的能力。我的建议是在课程中保留对处理器本身知识的讲解，而在实验中增加有实际背景的 DSP 应用程序开发的内容所占比重，鼓励同学们在平时多读例程、查阅帮助系统、多写代码，在实际使用 DSP 的过程中加深对处理器结构和使用方法的理解。

7 参考文献

[1]. 肖熙，第六章 语音信号的波形编码，清华大学电子系《语音信号处理》课程讲义。

[2]. 韩纪庆， 语音信号处理，清华大学出版社
[3]. V. D. Mytri, A. P. Shivaprasad, “Improving the dynamic range of a CVSD coder,” Electronic Letters, vol. 22, no. 8, pp. 429-230, Feb. 1986.

附：代码文件目录

MCVSD MATLAB仿真程序

- “MCVSD_test.m”：MCVSD算法的MATLAB仿真程序代码。
- “voice_48kHz_16bit.wav”：输入音频样例（节选自DG的《Mozart：Great Opera Moments II》中的第一曲：歌剧《Mitridate, re di Ponto》中的咏叹调《Al destin, che la minaccia》，演唱Natalie Dessay），采样率为48kHz，16比特量化，单声道。

MCVSD基本功能测试程序

- “code_mcvsd.asm”：MCVSD编码函数
- “decode_mcvsd.asm”：MCVSD解码函数
- “test.c”：主函数所在程序，定义了各种全局变量
- “coder.h”：滤波器结构定义及编解码函数声明
- “fir_coeff.h”：各滤波器系数
- “voice_input.h”：输入声音样例数组的定义
- “mds_def.h”

MCVSD单板实时编解码程序

- “code_mcvsd.asm”：MCVSD编码函数
- “decode_mcvsd.asm”：MCVSD解码函数
- “Initialize.c”
- “ISR.c”
- “main.c”：主函数所在程序，定义了各种全局变量
- “Process_data.c”：中断函数所调用的处理函数，其中调用了MCVSD编解码函数
- “coder.h”：滤波器结构定义及编解码函数声明
- “fir_coeff.h”：各滤波器系数
- “mds_def.h”
- “Talkthrough.h”