PSI 2432 - Projeto e Implementação de Filtros Digitais

Implementação de Filtros Digitais em Aritmética de Ponto Fixo II

Vítor H. Nascimento

21 de outubro de 2005

1 Introdução

Nesta experiência completa-se o trabalho iniciado na anterior, estudando um pouco mais sobre a implementação de filtros digitais nas placas doadas pela Analog Devices. Nesta segunda parte são estudados alguns problemas da implementação de filtros recursivos (IIR) em aritmética de ponto fixo, e também como utilizar os conversores A/D e D/A da placa.

Os objetivos da experiência são:

1. Aprender algumas estratégias para implementação de filtros recursivos e não-recursivos em processadores digitais de sinais;
2. Identificar problemas em implementações de filtros recursivos, em particular ciclos-limite granulares e de estouro;
3. Implementar um filtro simples de maneira completa (usando os conversores A/D e D/A). Levantar a resposta em freqüência do filtro, levando em conta os filtros anti-recobrimento e de reconstrução.

2 Ciclos-limite em filtros recursivos

O uso de aritmética de precisão finita (e particularmente aritmética de ponto fixo) para a implementação de filtros recursivos traz alguns problemas adicionais, que não aparecem quando se usam filtros não recursivos. Devido às realimentações existentes nos filtros recursivos, existem casos em que a saída pode ser consistentemente não-nula mesmo que a entrada seja mantida em zero por muito tempo, ou seja, o filtro pode operar como um oscilador.[1] Em algumas situações essas características são usadas para gerar um sinal senoidal de maneira eficiente; no entanto, para a implementação de filtros, a existência de uma saída não-nula para uma entrada nula é altamente indesejável.

[1] A expressão *ciclo-limite* vem da teoria de sistemas dinâmicos, e se refere ao fato da oscilação ser uma condição de operação natural do filtro, consideradas as não-linearidades decorrentes da aritmética em ponto fixo.

Há dois mecanismos que podem dar origem a uma oscilação persistente na saída de um filtro digital mesmo com entrada nula: o primeiro é devido a erros de quantização (arredondamento), que dá origem a oscilações de baixa amplitude; e o segundo é devido a estouro (overflow), e dá origem a oscilações de grande amplitude que degradam completamente o funcionamento do sistema. Não se pretende dar nesta apostila um tratamento muito aprofundado de ciclos-limite, apenas mostrar como surgem, e verificar sua existência.

2.1 Ciclos-limite glanulares

O primeiro mecanismo para aparecimento de oscilações é o devido a erros de quantização, que dá origem a oscilações chamadas *ciclos-limite glanulares* ou mais simplesmente *de quantização* (a palavra "glanular" aqui também se refere à quantização). A melhor forma de entender o que acontece é ver um exemplo simples. Considere um filtro de primeira ordem,

$$H(z) = \frac{1}{1 - 0{,}75z^{-1}}, \tag{1}$$

implementado com precisão de 2 bits mais sinal, com entrada $x(0) = 0{,}5$, $x(1) = 0{,}25, x(2) = x(3) = \cdots = 0$. A equação de diferenças correspondente é

$$y(n) = 0{,}75y(n-1) + x(n).$$

Considerando que o filtro tem condição inicial nula, a saída será (já em binário)

$$\begin{aligned}
&y(0) = (0{,}10)_2 = (0{,}5)_{10};\\
&y(1) = \mathrm{fx}(0{,}11 \cdot 0{,}10 + 0{,}01) = \mathrm{fx}(0{,}1010) = (0{,}11)_2 = (0{,}75)_{10}\\
&y(2) = \mathrm{fx}(0{,}11 \cdot 0{,}11 + 0{,}00) = \mathrm{fx}(0{,}1001) = (0{,}10)_2 = (0{,}5)_{10}\\
&y(3) = \mathrm{fx}(0{,}11 \cdot 0{,}10 + 0{,}00) = \mathrm{fx}(0{,}0110) = (0{,}10)_2 = (0{,}5)_{10}\\
&y(4) = \mathrm{fx}(0{,}11 \cdot 0{,}10 + 0{,}00) = \mathrm{fx}(0{,}0110) = (0{,}10)_2 = (0{,}5)_{10}\\
&\quad\vdots
\end{aligned}$$

Trocando agora para um filtro

$$G(z) = \frac{1}{1 + 0{,}75z^{-1}},$$

com a mesma entrada teremos[2]

$$\begin{aligned}
&y(0) = (0{,}10)_2 = (0{,}5)_{10};\\
&y(1) = \mathrm{fx}(1{,}01 \cdot 0{,}10 + 0{,}01) = \mathrm{fx}(1{,}1110) = (1{,}11)_2 = (-0{,}25)_{10}\\
&y(2) = \mathrm{fx}(1{,}01 \cdot 1{,}11 + 0{,}00) = \mathrm{fx}(0{,}0011) = (0{,}01)_2 = (0{,}25)_{10}\\
&y(3) = \mathrm{fx}(1{,}01 \cdot 0{,}01 + 0{,}00) = \mathrm{fx}(1{,}1101) = (1{,}11)_2 = (-0{,}25)_{10}\\
&y(4) = \mathrm{fx}(1{,}01 \cdot 1{,}11 + 0{,}00) = \mathrm{fx}(0{,}0011) = (0{,}01)_2 = (0{,}25)_{10}\\
&\quad\vdots
\end{aligned}$$

[2]Aqui estamos assumindo que o arredondamento de números negativos é feito sempre de modo a afastar o resultado de 0 — isso é feito para evitar que o erro de arredondamento tenha uma pequena média não-nula. O ADSP-2189M implementa arredondamento desta forma por padrão.

No primeiro caso, um filtro passa-baixas acaba com uma saída constante apesar da entrada ser nula a partir da terceira amostra. No segundo caso, um filtro passa-altas apresenta uma oscilação de freqüência π rad/amostra, apesar de novamente a entrada ser nula a partir da terceira amostra. O valor não-nulo na saída é causado pelos arredondamentos. Mesmo os pólos dos filtros sendo perfeitamente representáveis com o tamanho de palavra utilizado, o filtro se comporta como se tivesse um pólo na circunferência unitária. Para entender melhor o que acontece, considere um filtro genérico de primeira ordem:

$$y(n) = -\alpha y(n-1) + x(n),$$

com função de rede

$$F(z) = \frac{1}{1+\alpha z^{-1}},$$

implementado em ponto fixo com B bits mais sinal. A saída do filtro se manterá mesmo sem entrada, se valer para alguma condição de operação do filtro, que

$$|y(n-1)| = |\,\mathrm{fx}(\alpha y(n-1))|, \tag{2}$$

o que caracteriza um pólo "na prática" na circunferência unitária. Para a equação (2) ser satisfeita, é necessário que

$$\big||y(n-1)| - |\alpha y(n-1)|\big| < 2^{-B-1},$$

ou seja, pode haver oscilações se

$$|y(n-1)| < \frac{2^{-B-1}}{1-|\alpha|}. \tag{3}$$

A equação acima indica a faixa de valores em que a saída pode ficar oscilando mesmo sem entrada. Para $|\alpha|$ pequeno a amplitude de oscilação é pequena (e em particular, para $|\alpha| < 0{,}5$ o filtro não oscila), mas à medida que α se aproxima de 1 a amplitude das oscilações vai aumentando, podendo ficar consideravelmente grande para um pólo próximo da circunferência unitária.

Para o caso de filtros de primeira ordem, as oscilações só podem ocorrer em DC e na freqüência máxima (π rad/amostra). Para filtros de ordem mais elevada, podem ocorrer oscilações mais complicadas. O método de análise é semelhante ao descrito aqui, mas fica mais complicado para filtros de ordem mais elevada. Para filtros de segunda ordem com função de rede dada por

$$F(z) = \frac{N(z)}{1+a_1 z^{-1}+a_2 z^{-2}},$$

podem ocorrer ciclos-limite com amplitude limitada por [1]

$$|y(n)| < 2^{-B+1}\left\lfloor \frac{0{,}5}{1-|a_2|} \right\rfloor,$$

em que $\lfloor x \rfloor$ representa o maior inteiro menor ou igual a x. Repare que novamente podem ocorrer oscilações de amplitude elevada apenas se os pólos estiverem próximos da circunferência

unitária (ou seja, quando $|a_2| \approx 1$). Mais informações podem ser encontradas também em [2, 3].

Pode ser demonstrado (veja [3]) que uma condição para que não haja oscilações é que o filtro seja estável (desconsiderados os efeitos de quantização) e se todas as quantizações forem feitas de forma que $\text{fx}(x) \leq x$, o que é satisfeito se for usado truncamento da resposta. Assim, truncamento tem desvantagens importantes (erro médio não-nulo e variância do erro maior), mas pode ser usado para evitar ciclos-limite granulares em filtros recursivos. Outra observação útil é que ciclos-limite são mais raros de ocorrer em sistemas de segunda ordem quando é usado um acumulador de precisão dupla (como o MAC do ADSP-2189M) [2].

Observe finalmente que ciclos-limite podem ocorrer quando a entrada fica **constante**, não apenas quando a entrada se anula.

2.2 Ciclos-limite por estouro

Outro mecanismo para o aparecimento de ciclos-limite é por estouro. Esse tipo de ciclo-limite é potencialmente mais prejudicial que o anterior, pois terá sempre oscilações de grande amplitude. Pode ser evitado usando saturação do sinal de saída. Para ver como pode ocorrer um ciclo-limite por overflow, considere o filtro[3]

$$y(n) = 0{,}75y(n-1) - 0{,}75y(n-2) + x(n),$$

com $x(n) = 0$ para todo n e $y(0) = -0{,}75$ e $y(1) = 0{,}75$. Considere também que a saída não é saturada, e que as contas são realizadas com 3 bits mais sinal, e que as multiplicações e adições são efetuadas em precisão simples. A saída será

$$\begin{aligned}
y(2) &= \text{fx}(0{,}110 \cdot 0{,}110 + 1{,}010 \cdot 1{,}010) = \text{fx}(\text{fx}(0{,}100100) + \text{fx}(0{,}100100)) = \\
&= \text{fx}(0{,}101 + 0{,}101) = \text{fx}(1{,}010) = (-0{,}75)_{10} \\
y(3) &= \text{fx}(0{,}11 \cdot 1{,}01 + 1{,}01 \cdot 0{,}11) = \text{fx}(\text{fx}(1{,}011100) + \text{fx}(1{,}011100)) = \text{fx}(1{,}011 + 1{,}011) = \\
&= (0{,}110)_2 = (0{,}75)_{10} \\
&\vdots
\end{aligned}$$

3 Parte experimental

Nesta experiência você deve observar a ocorrência de ciclos-limite em situações simples, e aprender a usar os conversores A/D. Os programas para esta experiência estão no diretório `Exp2` do diretório desta disciplina no servidor da sala C1-10.

3.1 Observação de ciclos-limite granulares

Neste item você vai utilizar um filtro recursivo de primeira ordem, e observar tanto o seu funcionamento quanto a existência de ciclos-limite granulares. No diretório `Exp2-item1` você

[3] Exemplo retirado de [2].

encontra o programa `ciclo_peq.dsp`, que implementa um filtro recursivo de primeira ordem

$$H(z) = \frac{b}{1 + az^{-1}}.$$

1. Escolha adequadamente o valor de b para que não haja estouro, considerando que a entrada satisfaz $|x(n)| \leq 1$ (use os métodos vistos na apostila anterior [4]).

2. Para $a = -0{,}9$, levante no Matlab a resposta em freqüência do filtro, com o valor escolhido de b.

3. Crie no Matlab, usando a função `escrevehex`, um arquivo com o sinal

$$s(n) = \frac{1}{4}\left\{\cos\left(\frac{2\pi}{512}5n\right) + \cos\left(\frac{2\pi}{512}80n\right) + \cos\left(\frac{2\pi}{512}100n\right)\right\},$$

   em que $n = 0 \ldots 511$. Use esse arquivo como entrada para o filtro da placa, calcule a saída, leia-a no Matlab, e, calculando a TDF da entrada e da saída, calcule a resposta em freqüência do filtro para as três freqüências indicadas. Compare com o valor teórico obtido no item anterior.

4. Use agora o sinal `ciclopeq.dat` como entrada para o filtro, e troque o valor de b para 0,75. Desenhe o gráfico do sinal de entrada no Matlab, e verifique que a entrada é nula a partir de um certo ponto. Ache a saída calculada pela placa. Para que valor tende a saída?

5. Repita o item anterior, agora usando $a = -0{,}95, -0{,}99, +0{,}9, +0{,}95, +0{,}99$. Anote o valor para onde tende a saída do filtro, e verifique se a expressão (3) é obedecida.

3.2 Uso dos conversores A/D e D/A

A placa possui um CI codificador/decodificador (CODEC) com dois conversores A/D e dois conversores D/A. Como explicado na apostila sobre assembly [5], a comunicação entre o DSP e o Codec é feita através de uma porta serial, e o aviso de que uma amostra está pronta para ser lida é enviado a uma das interrupções do DSP. O programa `codec.dsp` ajusta os parâmetros do Codec para operação com taxa de amostragem de 8kHz, e lê amostras dos conversores A/D e apenas escreve as mesmas amostras na saída no próximo ciclo.

Atenção: a comunicação entre o DSP e o Codec às vezes apresenta erros se as interrupções provocadas pelo programa monitor estiverem ativas. Para evitar esses erros, os programas a seguir desabilitam o programa monitor, desabilitando a interrupção do `timer`. Assim que você mandar rodar um dos programas abaixo, o computador perderá comunicação com a placa. Você deve fechar o `VisualDSP++`, fazer as medidas pedidas, e depois abrir o `VisualDSP++` novamente.

3.2.1 Resposta dos filtros anti-rebatimento e de reconstrução

1. Para começar, como não usaremos pontas de prova nesta experiência, ligue o osciloscópio diretamente ao gerador de sinais com um cabo BNC, e verifique que o ganho de entrada não varia muito na faixa de freqüências de 0 a 4kHz.

2. O programa `codec.dsp`, no diretório `Exp2-item2` apenas copia a entrada na saída.

   Rode esse programa, e observe as duas saídas do osciloscópio. Com uma freqüência de 100Hz, ajuste a amplitude do sinal do gerador para que a saída tenha uma amplitude razoável sem saturação. Observe depois também o sinal de entrada no osciloscópio, e anote a sua amplitude (não varie a amplitude do gerador depois disso).

3. Meça a amplitude do sinal de saída para a entrada com freqüências de 100, 500, 1.000 e 2.000 Hz. Qual é o ganho da placa? O que você esperava?

3.2.2 Filtro não-recursivo

1. Passe agora para o diretório `Exp2-item3`. Estude o programa `fir.dsp`. Qual é a função implementada para cada canal?

2. O arquivo `filtro.dat` contém os coeficientes de um filtro passa-banda. Levante no Matlab a resposta em freqüência desse filtro.

3. Com o programa `fir.dsp`, implemente o filtro acima na placa. Levante sua resposta em freqüência (em 4 freqüências relevantes diferentes — escolha adequadamente as freqüências que devem ser medidas). Compare com os valores teóricos. Como você poderia fazer para que o filtro implementado na placa tivesse uma resposta mais próxima da teórica?

4. Em uma freqüência em que o ganho do filtro é alto, aumente a amplitude do sinal de entrada, e compare o que acontece com as duas saídas da placa. Explique o que você observa.

3.2.3 Implementação de seção de segunda ordem de filtro recursivo

1. Modifique o programa do item anterior para implementar um filtro recursivo de 2ª ordem.

2. Implemente um filtro passa-banda com ganho máximo em 800 Hz e banda-passante de 200 Hz.

3. Levante a resposta em freqüência do filtro, implementado na placa.

Referências

[1] PARKS, T. W.; BURRUS, C. S. *Digital Filter Design*. [S.l.]: Wiley-Interscience, 1987. (Topics in Digital Signal Processing Series).

[2] OPPENHEIM, A. V.; SCHAFER, R. W.; BUCK, J. R. *Discrete-Time Signal Processing*. 2nd. ed. [S.l.]: Prentice-hall, 1999.

[3] ANTONIOU, A. *Digital Filters: Analysis, Design, and Applications*. 2nd. ed. [S.l.]: McGraw-Hill, 1993.

[4] NASCIMENTO, V. H. *Implementação de Filtros Digitais em Aritmética de Ponto Fixo.* 2005.

[5] NASCIMENTO, V. H. *Programação em Assembly no ADSP-2189M.* 2005.