Ferramenta para Síntese e Implementação de Filtros em DSPs

João Gonçalves[1], Libânia Marques[1], José Salvado[1], Leonel Sousa[2]
[1] Escola Superior de Tecnologia de Castelo Branco, Av. do Empresário, 6000-767 Castelo Branco, Portugal, {joão.santos,libgfc,jsalvado}@est.ipcb.pt
[2] Instituto Superior Técnico (DEEC) e Instituto de Engenharia de Sistemas e Computadores (SIPS), R. Alves Redol, 9, 1000-029 Lisboa, Portugal, las@inesc-id.pt

Neste artigo apresenta-se uma ferramenta computacional para síntese automática de filtros digitais, com interface do tipo Windows®, desenvolvida em MATLAB™. Os filtros sintetizados podem ser posteriormente implementados em DSPs da família TMS320 da Texas Instruments, nomeadamente, nos DSPs TMS320C50 e TMS320C6201. Os filtros foram testados num sistema baseado no DSP C6201, recorrendo a sinais áudio digitais armazenados em ficheiro (.wav) enquanto no sistema com o DSP C50 se efectua a filtragem de sinais analógicos. Através da interface gráfica, o utilizador define o tipo de filtro (FIR ou IIR) e as suas especificações (com base na resposta em frequência), sendo a geração do código e a implementação realizadas de forma totalmente automática.*

1. Introdução

Os processadores digitais vocacionados para o processamento de sinais (DSPs) têm registado um número crescente de aplicações nos últimos anos. Este facto está ligado à evolução do desempenho deste tipo de processadores e à evolução das ferramentas computacionais de apoio ao desenvolvimento de sistemas de processamento de sinal. Entre os sistemas de processamento digital de sinais mais comuns, inclui-se os sistemas de filtragem, que visam alterar as características dos sinais em função da frequência.

A implementação prática dos filtros digitais corresponde à realização de operações numéricas de multiplicação e adição, envolvendo os coeficientes do filtro e amostras do sinal a processar. Estas operações tanto podem ser realizadas por *hardware* dedicado como por programação, baseando-se na operação de multiplicação e acumulação (MAC). Porém, a alteração das características dos filtros implica o projecto e a realização de novos circuitos ou programas, com custos elevados de tempo de desenvolvimento.

Visando sobretudo ultrapassar esta dificuldade, foi desenvolvida uma ferramenta computacional para apoio à síntese automática de filtros digitais, com base nas especificações dos filtros no domínio da frequência [1,2], e a subsequente implementação em processadores digitais de sinal, nomeadamente nos DSPs TMS320C50 e TMS320C6201 da Texas Instruments™ [3,4]. A aplicação foi desenvolvida em ambiente MATLAB 6.0™, dispondo de uma interface gráfica com o utilizador (GUI), e explorando o conjunto de funções específicas para o processamento de sinais disponíveis no MATLAB [5,6].

Na fase de implementação dos filtros, estabelece-se a ligação entre o ambiente MATLAB e as ferramentas de desenvolvimento da Texas Instruments [7]. Desde a fase de especificação à fase de implementação, não é requerida qualquer intervenção por parte do utilizador, pelo que este não necessita de ter conhecimentos sobre os métodos de síntese de filtros digitais, nem sobre linguagens de programação ou sobre arquitectura dos DSPs.

Este artigo está organizado da seguinte forma. Na secção seguinte referem-se os aspectos fundamentais relativamente aos filtros digitais. Na 3ª secção apresentam-se as principais características das arquitecturas dos DSPs ‘C50 e ‘C6201. Os métodos usados para a síntese e implementação de filtros, na aplicação desenvolvida, são referidos na secção 4, apresentando-se alguns resultados experimentais na secção 5. Finalmente, na última secção apresentam-se as conclusões.

2. Filtros Digitais to Tipo FIR e IIR

Os filtros digitais são geralmente classificados como sistemas de resposta impulsiva finita (FIR) ou infinita (IIR). Os filtros IIR correspondem sempre a sistemas recursivos, podendo ser sistemas instáveis. Os filtros FIR correspondem a sistemas não recursivos e são sempre estáveis, podendo apresentar características de fase linear. Porém, para as mesmas especificações, os filtros FIR tem uma ordem superior aos filtros IIR, requerendo, portanto, um maior número de operações aritméticas. Por outro lado, o efeito dos erros de quantificação é, em geral, mais significativo nos filtros IIR do que nos filtros FIR.

Os filtros IIR podem ser descritos pela sua equação às diferenças, ou através da sua função de sistema (aplicando a transformada Z):

$$y[n]=\sum_{k=0}^{M} a_k \cdot x[n-k]+\sum_{k=1}^{N} b_k \cdot y[n-k] \;\Leftrightarrow\; T(z)=\frac{\sum_{k=0}^{M} a_k \cdot z^{-k}}{1-\sum_{k=1}^{N} b_k \cdot z^{-k}} \tag{1}$$

O mesmo se aplica aos filtros FIR:

$$y[n]=\sum_{k=0}^{M} b_k \cdot x[n-k] \;\Leftrightarrow\; T(z)=\sum_{k=0}^{M} a_k \cdot z^{-k} \tag{2}$$

O comportamento de qualquer sistema contínuo, linear e invariante no tempo, pode ser aproximado pela equação diferencial (no dominio do tempo):

$$y^{N}(t)+\sum_{k=0}^{N-1} a_i \cdot y^{i}(t) \;=\sum_{k=0}^{M} b_i \cdot x^{i}(t) \tag{3}$$

A resposta no domínio da frequência pode ser obtida calculando a transformada de Laplace e fazendo $s = j\omega$:

$$H(s)=\frac{Y(s)}{X(s)}=\int_{t=0}^{+\infty} h(t)e^{-st}dt \tag{4}$$

A representação discreta do sistema contínuo, linear e invariante no tempo, origina um sistema discreto linear e invariante, caracterizado pela equação às diferenças:

$$y[n]+\sum_{i=1}^{N} a_i y[n-i]=\sum_{i=0}^{M} b_i x[n-i] \tag{5}$$

A função de transferência do sistema discreto apresenta uma relação com a função de transferência do sistema contínuo através de:

$$H_d(z)=H(s)\Big|_{s=(1/T)\ln z} \tag{6}$$

É comum fazer-se a aproximação da expressão $\left(1/T\right)\ln z$, donde resulta uma relação de transposição do plano Z no plano S, designada por transformação bilinear, definida por:

$$s=\frac{2}{T}\frac{z-1}{z+1} \tag{7}$$

E a transposição do plano S no plano Z, definida pela equação:

$$\omega=\frac{2}{T}\frac{1}{j}\frac{(e^{j\Omega}-1)}{(e^{j\Omega}+1)}=\frac{2}{T}\frac{(1/j2)(e^{j(\Omega/2)}-e^{-j(\Omega/2)})}{(1/2)(e^{j(\Omega/2)}+e^{-j(\Omega/2)})}=\frac{2}{T}\tan\frac{\Omega}{2} \tag{8}$$

Esta equação utiliza-se, normalmente, para compensar o desvio de frequências existentes entre o plano S e o plano Z (*pre-warping*).

Síntese de Filtros Digitais

A síntese de filtros digitais faz-se usando vários métodos, sendo estes diferentes para filtros IIR e FIR.

Os coeficientes dos filtros IIR podem ser obtidos através das características dos filtros analógicos correspondentes segundo vários métodos, nomeadamente, a conservação da resposta impulsiva, a transformação bilinear e pela localização de pólos e zeros da função de transferência. Porém, embora simples, este último método não é indicado para filtros mais complexos, devido aos efeitos dos erros numéricos na localização de pólos e zeros. O método mais usual é o método da transformação bilinear.

Os filtros FIR não podem ser obtidos pelas aproximações utilizadas nos sistemas contínuos, dado que não existem sistemas analógicos com resposta impulsiva finita. A resposta impulsiva dos filtros FIR é definida em (2). Para a síntese de filtros FIR usam-se, geralmente, o método dos minimos quadrados (*least squares*) ou o método das janelas (*windows method*), para truncatura da resposta impulsiva.

3. Arquitectura dos DSPs TMS320C50 e TMS320C6201

Os DSPs são microprocessadores com arquitectura especializada para o processamento de sinal. Os algoritmos de processamento de sinal são computacionalmente intensivos, mas as operações de base são operações aritméticas simples de adição e multiplicação. Assim, os DSPs possuem pelo menos um multiplicador e um acumulador implementados em *hardware*, permitindo que num único ciclo de relógio se realizem em simultâneo uma multiplicação e uma adição. Adoptam ainda, em geral, arquitecturas do tipo Harvard modificadas, com barramentos separados para dados e programa, com um funcionamento em cascata (*pipelining*). Arquitecturas mais avançadas, do tipo VLIW (*Very Long Instruction Word*), permitem que cada instrução seja composta por várias instruções, que por sua vez são executadas por diferentes unidades funcionais.

O DSP TMS320C50 é um processador em vírgula fixa, com arquitectura *Harvard* do tipo acumulador, a funcionar a uma frequência máxima de 40 MHz, com um multiplicador e uma ALU de 16 *bits*, sendo as instruções quase todas executadas numa cascata com 4 andares. A taxa máxima de execução de instruções é de 40 MIPS (milhões de instruções por segundo) [8].

O DSP TMS320C6201 baseia-se numa arquitectura de elevado desempenho do tipo VLIW, denominada VelociTI™ [9], dispondo de 8 unidades de processamento agrupadas em dois blocos de processamento independentes (cada um com 4 unidades), aos quais estão associados bancos de 16 registos de 32 bits. A cascata do DSP C6201 tem 11 andares, e na situação de máximo aproveitamento dos recursos permite executar até 8 instruções num único ciclo de relógio. Com frequências de relógio de 167 ou 200 MHz, este DSP permite taxas máximas de execução de instruções de 1336 ou 1600 MIPS.

4. Síntese e Implementação de Filtros com Base na Aplicação DFDT

A ferramenta computacional para síntese e implementação de filtros em DSPs, DFDT (*Digital Filter Design Toolkit*), foi desenvolvida de modo a realizar, automaticamente, a síntese e a implementação de filtros com resposta impulsiva FIR e IIR em DSPs. Apresentam-se em seguida os métodos usados para a síntese dos filtros, as janelas para introdução das respectivas especificações

Síntese de Filtros IIR

Com base na relação entre o plano S e o plano Z, utilizam-se os seguintes passos para síntese de filtros digitais IIR:

- Compensação das especificações na frequência do filtro digital, usando a equação (8);
- Obtenção do filtro passa-baixo equivalente;
- Obtenção da função de transferência no plano S, através de uma aproximação clássica para filtros analógicos (Butterworth, Chebyshev, Inversa de Chebyshev e Elíptica);

- Desnormalização e eventual transformação de frequências para o filtro desejado;
- Obtenção a função de transferência no plano Z, através da equação (7).

Síntese de Filtros FIR

Para a síntese de filtros FIR usa-se o método da truncatura da resposta impulsiva infinita, através de janelas que definem a função de truncatura:

$$h(n) = \begin{cases} h_d(n), & 0 \le n \le N-1 \\ 0, & \text{caso contrario} \end{cases} \tag{10}$$

Em que a resposta impulsiva ideal $h_d(n)$ é obtida através da transformada de Fourier inversa $H_D(\omega)$, definida por:

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_D(\omega) e^{jwn} d\omega \tag{11}$$

sendo $h(n)$ geralmente representada como o produto entre a resposta impulsiva desejada e a janela $w(n)$ de duração finita:

$$h(n) = h_d(n) \times w(n) \tag{12}$$

Nesta aplicação consideram-se a janela rectangular, de Hanning, de Hamming, de Blackman e de Kaiser, como se mostra na figura 1, em que a janela de Kaiser é variável (não apresentada), cujos valores aproximados de atenuação (para filtros passa-baixo) se apresentam na tabela 1.

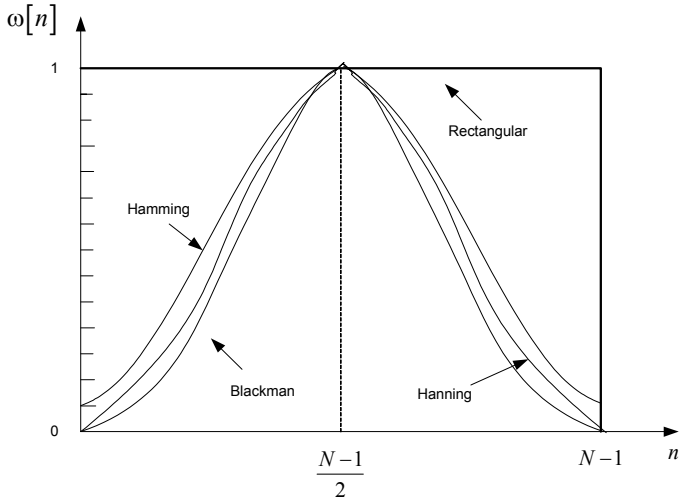

Figura 1. Janelas utilizadas para a truncatura da resposta impulsiva, nos filtros FIR.

Tabela 1. Valores aproximados de atenuação para filtros passa-baixo, para as diferentes janelas consideradas.

Tipo de Janela	Atenuação no 1º lóbulo (dB)	Atenuação mín. na banda de atenuação (dB)
Rectangular	-13	21
Hanning	-31	44
Hamming	-41	53
Blackman	-57	74
Kaiser	--	90

As especificações dos filtros pretendidos são estabelecidas pelo utilizador, para os diferentes tipos de filtros, através da interface apresentada nas figuras 2 e 3.

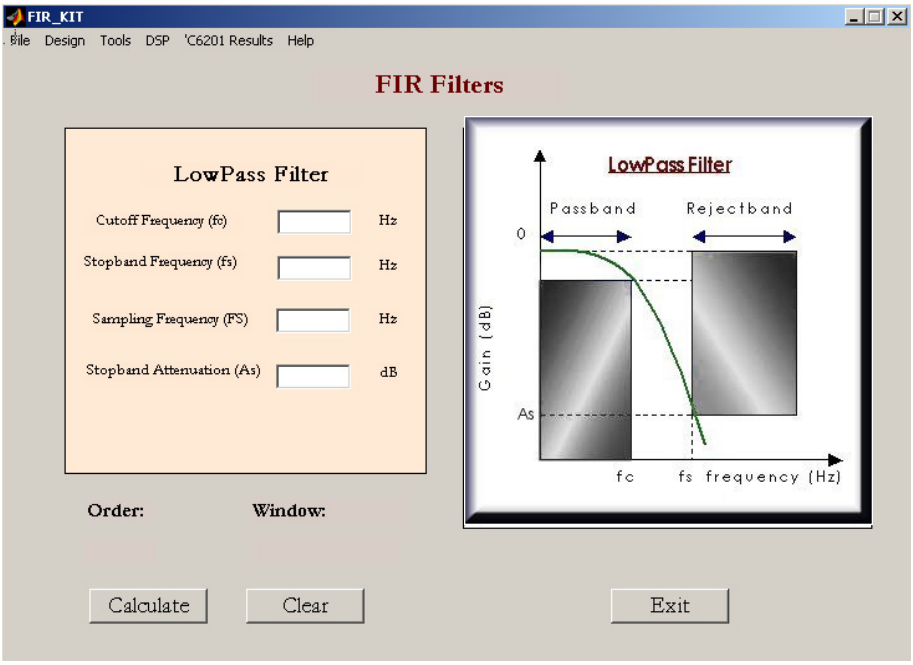

Figura 2. Aspecto da interface com o utilizador para especificação de filtros FIR.

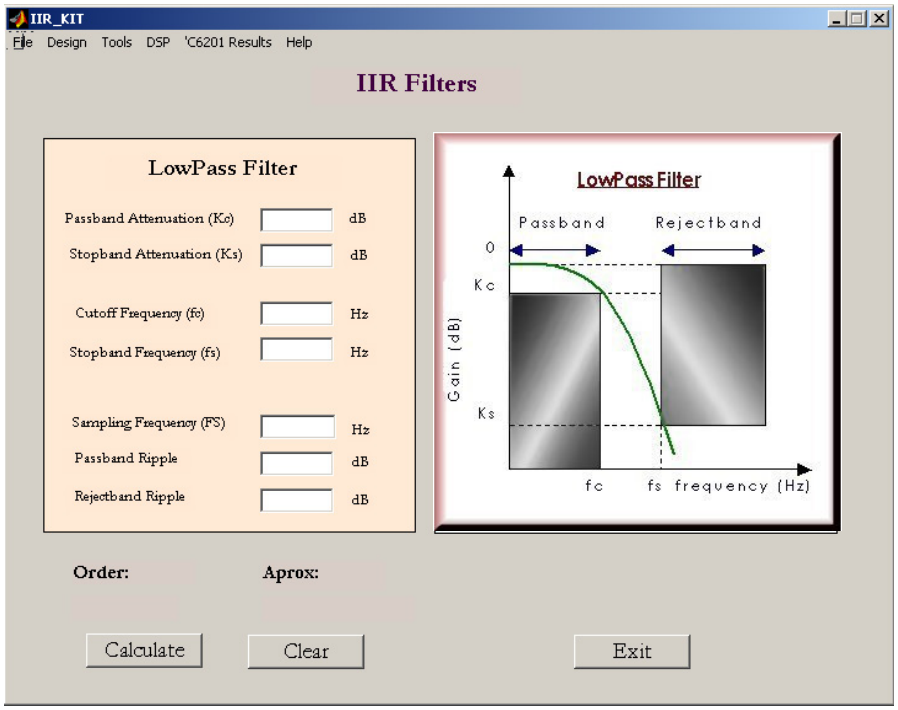

Figura 3. Aspecto da interface com o utilizador para especificação de filtros IIR.

Para a implementação dos filtros usam-se os módulos de avaliação da Texas Instruments, para cada um dos DSPs considerados: o módulo TMS320C6201 EVM (ver figura 4), com frequência de relógio 167 MHz, e o módulo C50 DSP Starter Kit (figura 5) a 40 MHz. O módulo TMS320C6201 é realizado numa placa com barramento PCI para PC, que pode ser programado utilizando linguagens de alto nível, em C/C++ [10], ou em *assembly* [11], a partir das ferramentas de desenvolvimento da Texas Instruments [12]. O módulo C50 DSP Starter Kit é um módulo externo, que é programado em *assembly*.

Sendo a capacidade dos DSPs limitada ao nível da precisão, torna-se necessário representar o valor dos coeficientes e das amostras em palavras de dimensão finita. Visando evitar os erros devido à quantificação, efectua-se o varrimento do sistema (representado por secções bi-quadráticas) de modo a determinar o valor máximo, obtendo-se a notação Q mais adequada à representação dos coeficientes e das amostras, consoante o valor máximo seja positivo (eq. 13) ou negativo (eq. 14).

$$Q = \lfloor 15 - \log_2 |\max| \rfloor \quad (13)$$

$$Q = \lceil 14 - \log_2 |\max| \rceil \quad (14)$$

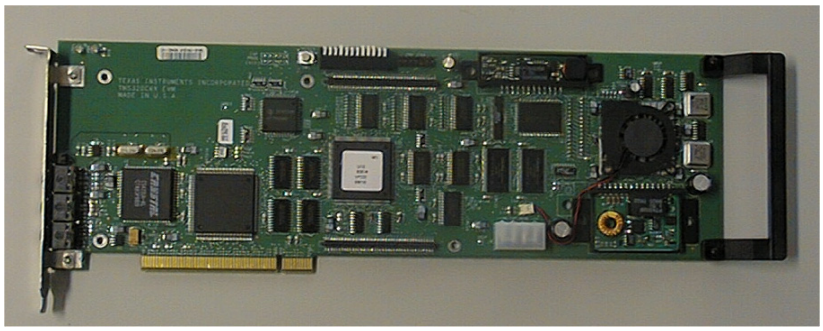

Figura 4. Aspecto do módulo de avaliação TMS320C6201 EVM (frequência de relógio 167 MHz).

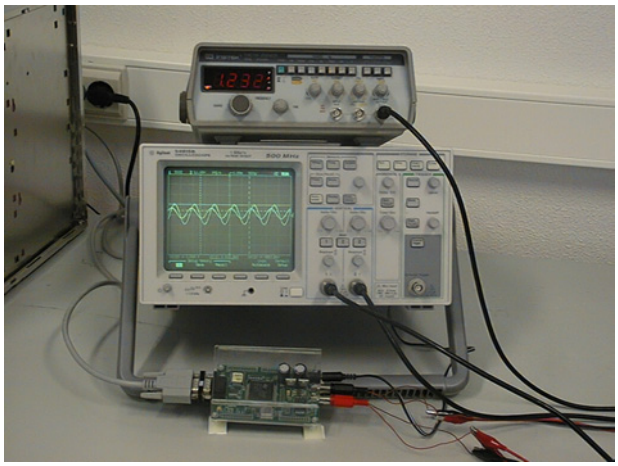

Figura 5. Aspecto de uma aplicação de filtragem em tempo real com o módulo TMS320C50 DSK.

Apresentam-se em seguida exemplos do código gerado para implementação do filtros, código C para a implementação no DSP C6201 (ver fig. 6), e em assembly, para a implementação no DSP C50 (fig. 7).

```
for(j=0;j<M;j++)       /*Ciclo relativo às secções */
{
  for(i=0;i<DIM;i++){ /*Ciclo relativo ao cálculo das amostras filtradas */
    y0=1L << (Q-1);
    if(i==0)           /* Cálculo da amostra do sinal de saída */
      y0 = num[sec]*x[i];
    else if(i==1)
      y0 = num[sec]*x[i]+num[sec+1]*x[i-1]-den[sec+1]*y[i-1];
    else
      y0 = num[sec]*x[i]+num[sec+1]*x[i-1]+num[sec+2]*x[i-2]-
           den[sec+1]*y[i-1]-den[sec+2]*y[i-2];

    y0=y0<<(16-Q);   /*Elimina bits redundantes de sinal */
    y[i]=(short)(y0>>16);  /*Cast do resultado para 16 bit */
  }
  if(j<M){  /*Caso não seja a última bi-quadrátca o sinal de saída de
              uma secção passa a ser o sinal de entrada da seguinte */
  for(i=0;i<DIM;i++)
    x[i]=y[i];
  }
  sec+=3;  /*Avança para os próximos coeficientes */
}
```

Figura 6. Exemplo de código em "C" gerado através da aplicação DFDT, para filtros IIR no DSP C6201.

```
RX:     setc    INTM          ;Desactiva interrupts
        lamm    DRR           ;Lê amostra de entrada (ADC)
        and     #0FFFCh
        ldp     #XN           ;Amostra do sinal de entrada reservada
        sacl    XN            ;numa posição de memória
        lar     AR0,#YNLAST0  ;Carrega registo AR0 com o valor da saída
                              ;anterior
        zap                   ;Limpa acumulador e registo p
        mar     *,AR0         ;Coloca no acumulador valor do registo AR0
        rpt     #TAPS         ;Repete instrução seguinte nº de coef.-1
        macd    #hLAST,*-     ;Multiplica, acumula e desloca para a
                              ;próxima posição
        apac                  ;Acumula resultado final (produto final)
        ldp     #0
        sach    OUTPUT,2      ;Elimina bits redundantes de sinal (16-Q)
        lacc    OUTPUT        ;Resultado na variável OUTPUT
        and     #0FFFCh
        samm    DXR           ;Envia valor calculado para o DAC
        clrc    INTM          ;Activa interrupts
        rete                  ;Retorna ao programa principal
```

Figura 7. Exemplo de código em *Assembly* gerado através da aplicação DFDT, para filtros FIR no DSP C50.

5. Resultados Experimentais

Nesta secção apresentam-se alguns resultados experimentais obtidos relativamente à resposta em frequência dos filtros sintetizados. Na implementação no DSP C6201, os resultados obtidos podem ser visualizados directamente, a partir da própria aplicação, nomeadamente, os diagramas de amplitude e de fase (ver figura 8), as respostas ideal (analítica) e real (prática) e a localização de pólos e zeros (ver figura 9). É ainda possível visualizar a resposta ao impluso unitário e ao degrau unitário, e o atraso de grupo.

Na implementação no DSP C50, como este opera em tempo real, os resultados da resposta em frequência obtêm-se através de medições externas, procedendo-se posteriormente à comparação entre a resposta ideal e real.

6. Conclusões

Neste artigo apresentou-se uma aplicação desenvolvida em ambiente MATLAB™, para a síntese e implementação de filtros em DSPs. Através de uma interface gráfica, o utilizador define o tipo de filtro e especifica as características da resposta em frequência, e o DSP a considerar para a implementação. O processo é realizado de forma totalmente automática, sendo a síntese, a geração do código e a implementação no DSP realizadas sem qualquer intervenção do utilizador.

Os filtros IIR são sintetizados a partir das aproximações clássicas para filtros analógicos (Butterworth, Chebyshev, Inversa de Chebyshev e Elíptica), utilizando a transformação bilinear. Na síntese dos filtros FIR utiliza-se a truncatura da resposta impulsiva pelo método das janelas, considerando a janela rectangular, de Hanning, de Hamming, de Blackman e de Kaiser.

Relativamente aos filtros sintetizados, a aplicação desenvolvida permite visualizar, a resposta em frequência (amplitude e fase) e a estabilidade dos filtros, nomeadamente, pela localização de pólos e zeros, e pela resposta ao degrau (apenas nos filtros IIR). O utilizador pode guardar os dados relativos aos filtros sintetizados em ficheiro, sem quantificação, para posterior consulta, e/ou implementação nos DSPs.

Para implementação no DSP ‘C6201, usam-se ficheiros do tipo (***.wav**), sendo possível visualizar e comparar o sinal original e o sinal filtrado. Na implementação no DSP ‘C50, os sinais são filtrados em tempo real.

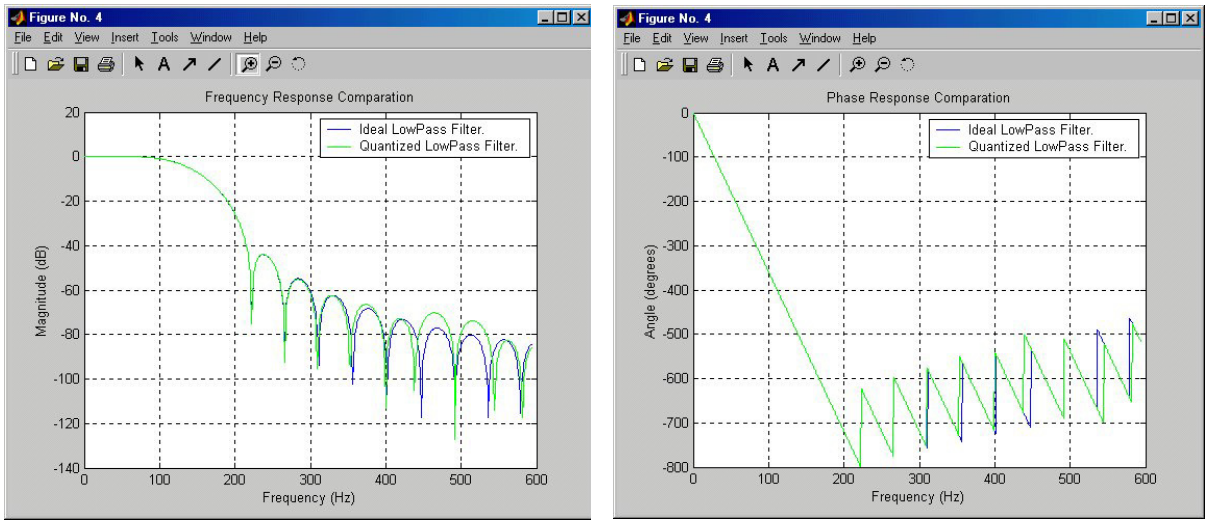

Figura 8. Aspecto dos resultados obtidos para a resposta de filtros: amplitude (esq.) e fase (dir).

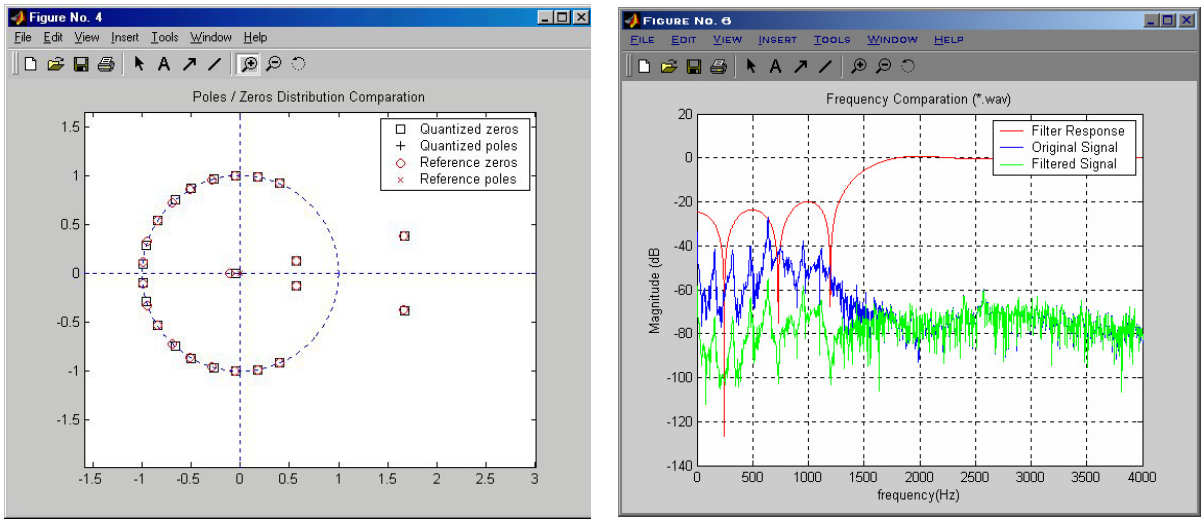

Figura 9. Aspecto dos resultados para análise dos filtros implementados: estabilidade (esq.) e comparação (dir)..

Referências

[1] A. Oppenheim, R. Schaffer, "Digital Signal Processing", *Prentice-Hall*, 1975.

[2] L. Huelsman, "Active and Passive Analog Filters Design", McGraw-Hill, 1992.

[3] "TMS320C5x DSP User's Guide", *Texas Instruments,* SPRU056-C, 1997.

[4] "TMS320C6000 Programmer's Guide", *Texas Instruments*, SPRU198-C, 1999.

[5] "Matlab: Signal Processing Toolbox", *The MathWorks*, 2000.

[6] "Matlab: Creating Graphical User Interfaces", *The MathWorks*, 1999.

[7] "Code Composer Studio User's Guide", *Texas Instruments*, SPRU328, 1999.

[8] E. Ifeachor, B. Jervis, "Digital Signal Processing: A Practical Approach", *Addison-Wesley*, 1991.

[9] "TMS320C6000 CPU and Instruction Set reference Guide", *Texas Instruments*, SPRU189, 1998.

[10] "TMS320C6000 Optimizing C Compiler User's Guide", *Texas Instruments*, SPRU187, 1999.

[11] "TMS320C6000 Assembly Language Tools User´s Guide", *Texas Instruments*, SPRU186, 1999.

[12] "TMS320C50 Applications Guide", *Texas Instruments*, BPRA063, 1997.