CENTRO FEDERAL DE EDUCACAO TECNOLOGICA DE SANTA CATARINA (CEFET)

JOÃO PAULO BODANESE

IMPLEMENTAÇÃO DE UM EQUALIZADOR DE ÁUDIO EM DSP

Florianópolis, SC

2008

DEDICATÓRIA

Aos meus pais e meu irmão, pelo constante apoio e incentivo às minhas idéias.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

RESUMO

Com o advento e a atual expansão dos sistemas multimídias de áudio, o uso de equalizadores digitais se tornou um item fundamental no desenvolvimento destes. O presente trabalho relata a síntese do desenvolvimento de um equalizador gráfico digital, composto de 10 filtros do tipo IIR, implementados em freqüências divididas simetricamente, ao longo do espectro auditivo em um processador digital de sinal (DSP), no sistema operacional uClinux. Baseado nesta síntese foi proposto um protótipo apto a realizar equalizações em tempo real, utilizando a interface de comunicação serial do computador para o controle dos parâmetros de equalização no DSP. Neste trabalho será abordada a teoria sobre processamento digital de sinais, a arquitetura dos processadores DSPs, teoria básica sobre som, os principais tipos de filtros utilizados por equalizadores profissionais e a estratégia utilizada para implementação deste trabalho. E por fim serão apresentados os resultados de simulação obtidos com o Matlab e o simulador do DSP, assim como resultados obtidos em tempo real.

Palavras Chaves: Equalizador digital, processamento digital de sinal, DSP, uClinux.

ABSTRACT

With the advent and the current audio multimedia fever, the use of digital equalizers has became a fundamental item in it's development. This work makes a brief synthesis of the development of a digital graphic equalizer with 10 IIR filters, implemented in frequencies symmetrically divided along the auditive spectrum in a digital signal processor (DSP), an uClinux operation system. Based on this synthesis, it was proposed a prototype able to make equalizations in real time, using a serial communication interface of the computer for the control of the parameters of equalization in the DSP. In this work, it will be explain the theory about digital signal processing, the architecture of DSP processors, the basic theory of the sound, the main kinds of filters used by professional equalizers and the strategy used for implementation of this work. Finally, it will be presented the results acquired by the simulations with the Matlab and the DSP simulator as well as the results acquired in real time.

Key Words: Digital Equalizer, Digital signal processor, DSP, uClinux.

1 INTRODUÇÃO

A tecnologia faz parte do mundo atual. O cotidiano de todo cidadão está intimamente ligado ao prazer e ao bem estar que a tecnologia oferece. O desenvolvimento progressivo de novos produtos alavanca o crescimento de mercado, o que realimenta a vontade das empresas em fornecer programas e equipamentos, que tornem a vida cada vez mais confortável. Pensando nisso, elas têm feito pesados investimentos em tecnologia nos últimos anos, o que faz surgir a cada ano uma nova descoberta promissora.

A década de 80 é marcada pelo fim do reinado dos circuitos analógicos, e início de uma nova era baseada nos circuitos digitais. Hoje se nota um aumento considerável de um público que possui televisores, computadores e sofisticados aparelhos de som em suas residências. Com o aparecimento dos CDs, DVDs, MP3 e outros sistemas digitais, surgiu a necessidade de se desenvolver novas tecnologias para implementar essas aplicações. No ano de 1982, a empresa *Texas Instruments* deu os primeiros passos e desenvolveu o DSP (processador digital de sinais), que mais tarde tornou-se uma ferramenta indispensável para o desenvolvimento dos sistemas digitais citados acima. Os DSPs são microprocessadores especializados em processamento digital de sinal, em aplicações que requerem alta performance de processamento. Como exemplo podemos citar aplicações de áudio e vídeo operando em tempo real.

Um sinal é um conjunto de informações que podem ser armazenados, transmitidos ou utilizados para tomar decisões. Sinais típicos que encontramos em nossas vidas diárias são voz, música e imagem. Um sinal é uma função de uma ou mais variáveis como tempo, distância, posição, temperatura ou pressão. Um sinal carrega informação, e o objetivo do processamento de sinais é extrair essa informação e processá-la adequadamente. O processamento digital de sinais lida com a representação matemática do sinal e com as operações algorítmicas executadas para a extração da informação desejada (SOUZA, 2005).

Uma vez digitalizado o sinal, o DSP poderá executar diferentes algoritmos, como por exemplo, equalizar o sinal, eliminar ruídos, adicionar efeitos sonoros (em áudio), comprimir ou descomprimir o sinal, entre outras aplicações.

Equalizadores são circuitos (digitais ou analógicos) que modificam a resposta de freqüência de um sinal de saída, de acordo as necessidades. Por exemplo, caso seja necessário obter um sinal com a faixa dos graves mais reforçado, deve-se amplificar as faixas de freqüências mais baixas, que correspondem ao som grave. O mesmo pode ser feito para obter os sinais mais

agudos, ou qualquer outra freqüência dentro da banda de (20Hz a 20KHz), que é percebida pela audição do ser humano. Exemplos de equalizadores “populares” são encontrados em diversos aparelhos de som domésticos, aparelhos de CDs para carros e entre outros. Já em aplicações profissional, pode-se citar: compensar as deficiências acústicas de um local, obter o timbre desejado para um determinado instrumento, ou atender a um padrão como o RIAA (Recording Industry Association of America) (NEIVA, 2005). Um equalizador digital de alta qualidade, deve operar com a maior precisão possível e não introduzir distorções nos sinais.

Neste trabalho será estudada a viabilidade e desenvolvimento de um equalizador digital, que permitirá equalização de 10 faixas de freqüências divididas simetricamente ao longo do espectro auditivo. A implementação será desenvolvida utilizando o DSP Blackfin BF537 da Analog Devices, que através de uma interface de comunicação serial do computador ao DSP, será possível alterar em tempo real os ganhos de equalização. O kit de desenvolvimento é chamado de STAMP board e tem como sistema operacional o uClinux.

1.1 PROBLEMA

Diante do exposto, surge o seguinte questionamento: Como desenvolver um equalizador digital em um DSP, utilizando filtros digitais e que opere em tempo real?

Pretende-se através de um estudo sobre filtros digitais dos tipos IIR e FIR, escolher a estratégia mais adequada para implementar em DSP e executar em tempo real um equalizador digital.

A escolha do filtro a ser utilizado depende de alguns fatores como, por exemplo a quantidade de coeficientes utilizados para a sua síntese. Para filtros do tipo FIR, que utilizam muitos coeficientes, exigindo maior tempo de processamento, dependendo da velocidade do DSP utilizado, este pode não conseguir processar uma amostra do sinal de áudio a tempo antes que uma próxima amostra chegue.

O uso filtros do tipo IIR, tem-se a vantagem da utilização de poucos coeficientes, assim economizando espaço de memória e conseqüentemente diminuindo o tempo de processamento entre uma amostra e outra. Por outro lado, os filtros IIR têm a desvantagem de poderem ser instáveis. Estes e outros detalhes serão abordados nos capítulos a seguir.

Existe no mercado, especificação de filtros digitais que são de específico uso para equalizadores de áudio, entretanto, estes filtros possuem funções de transferência muito

complexas e de difícil acesso, já que as empresas desenvolvedoras não disponibilizam estas especificações para o domínio público.

Para implementação do filtro, também devem ser levados em consideração problemas decorrentes ao uso de processadores DSPs, que utilizam a notação de ponto fixo. Já que a faixa de representação dos valores é limitada, fica-se mais propenso ao problema de saturação do sinal.

O equalizador digital proposto nesse projeto consiste em um grupo de 10 filtros digitais passa-faixa em paralelo, centrados em diferentes freqüências, a fim de cobrir todo o espectro audível do ser humano. Em cada filtro, será aplicado um ganho para controlar os níveis de equalização. A figura 1 ilustra uma representação pretendida da distribuição dos filtros. Nota-se que a representação abaixo está na escala logarítmica, já que os níveis de audição são representados dessa maneira.

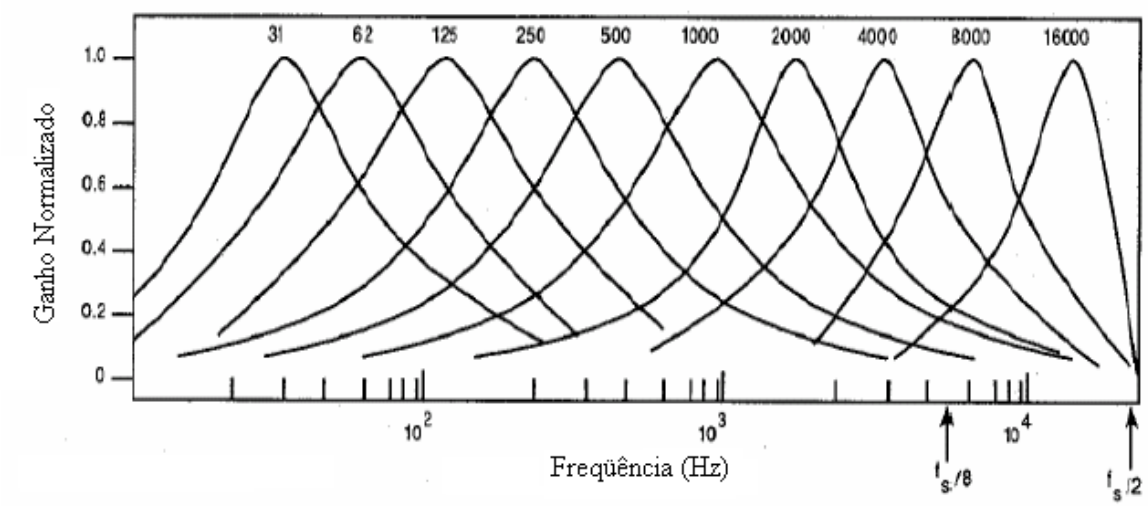

Figura 1: Curva de resposta dos filtros.

Fonte: MICEA e STRATULAT, 2001.

Assumindo que este equalizador irá trabalhar com uma fonte sonora com freqüência de amostragem *Fs* igual a 48000 Hz (qualidade de CD), as freqüências centrais para os filtros compreenderão a faixa de 0 Hz a *Fs*/2 como mostrado na figura 1.

A cada período de amostragem, uma amostra do canal direito e e uma do esquerdo (estéreo) passarão pelos 10 filtros. Após cada filtro selecionar a sua freqüência correspondente, a saída é modificada pelo ganho particular de cada filtro. E finalmente, a de cada filtro será somadoa com as demais. A figura 2 mostra o banco de filtros dispostos em paralelo.

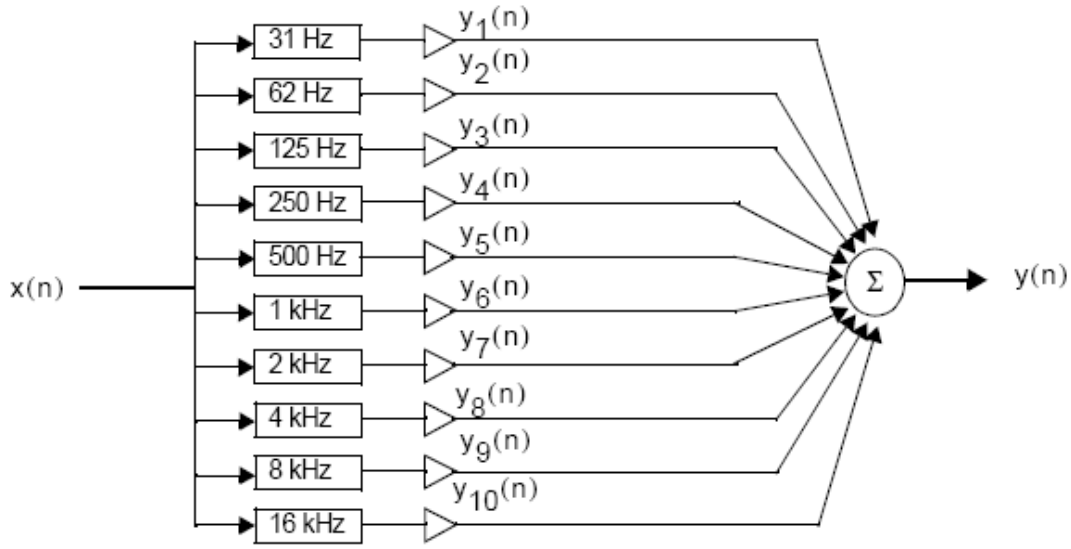

Figura 2: Diagrama de blocos representando um banco de filtro em paralelo.

Fonte: MONTGOMERY, 2001.

Como validação do projeto, o equalizador digital deverá operar em tempo real, ou seja, os parâmetros (ganhos) de cada um dos 10 filtros serão modificados em tempo de execução. Como proposta, os controles farão parte de uma interface gráfica em um computador conforme ilustrado na figura 3.

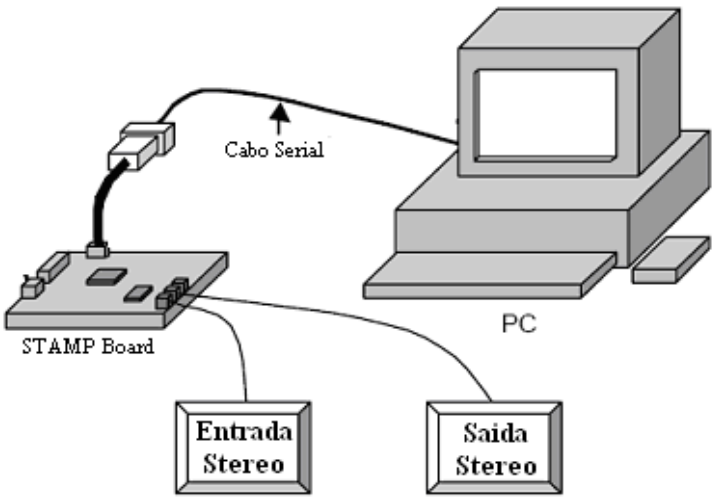

Figura 3: Interface gráfica em um PC interagindo com o DSP.

Fonte: Elaboração própria (Adaptado de MONTGOMERY, 2001).

A escolha do sistema operacional uClinux e do processador DSP BF537 veio da familiarização no desenvolvimento nessa plataforma. O uso de um DSP e S.O. mais simples poderiam ser utilizados para realizar a mesma tarefa.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Implementar um protótipo para equalizar um sinal de áudio estéreo, assim como a variação dos parâmetros em tempo real das faixas de freqüências.

1.2.2 Objetivos Específicos

- Estudar equalizadores digitais;
- Estudar técnicas de DSP usados em equalizadores;
- Estudar filtros digitais;
- Estudar especificações de equalizadores comerciais;
- Implementar em Matlab o algoritmo para o equalizador;
- Implementar em DSP o algoritmo escolhido;
- Desenvolver a interface gráfica em Matlab;
- Executar testes de validação.

1.2.3 Escopo e delimitação do trabalho

Este trabalho consistirá no desenvolvimento de um protótipo de um equipamento para equalização de sinais de áudio, utilizando filtros digitais. Os algoritmos serão inicialmente implementados e simulados no Matlab, sendo posteriormente desenvolvidos e validados em tempo real no kit de desenvolvimento.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados todos os conceitos relacionados com o desenvolvimento do trabalho de conclusão de curso.

2.1 SINAIS ANALÓGICOS E SINAIS DIGITAIS

Os sinais realizam um papel importante em nosso cotidiano. Um sinal é uma função de variáveis independentes como distância, posição, temperatura e pressão. Uma música ou um sinal de voz, por exemplo, representam a pressão do ar como função do tempo propagando=se no espaço. Estes sinais são gerados de modo natural, sendo chamados de sinais analógicos, os quais são representados por um número infinito de valores de representação. Isto ocorre porque o domínio dos sinais analógicos é contínuo no tempo.

Um sinal de áudio pode ser representado por uma tensão elétrica. Quando essa tensão elétrica é aplicada a um alto falante, provoca a vibração do cone flexível, produzindo, dessa forma, as ondas de pressão que são interpretadas como sons (BARBOSA, 2005). A unidade de medida da intensidade do som (nível de pressão acústica) mais utilizada é o *decibel* (dB).

Os sinais digitais são formados por um conjunto de *bits,* onde cada bit pode assumir dois valores distintos: alto (1) e baixo (0). Por exemplo, existem 256 possíveis representações em uma seqüência de 8 bits sem sinal. Considerando-se esses valores como inteiros os mesmos correspondem a uma faixa de 0 até 255. Já numa representação de inteiro com sinal, os valores correspondem a uma faixa de -128 até +127.

Os sinais digitais não são representados em um domínio de tempo contínuo, mas sim num domínio de tempo discreto. Se necessária a conversão do sinal analógico para o domínio digital deve ser feita a sua conversão através dos processos conhecidos como “amostragem” e “quantização”.

2.1.1 Processo de amostragem

A amostragem consiste na retenção de um conjunto de valores discretos, a partir da variação contínua de valores assumidos pelo sinal analógico. A figura 4 ilustra um exemplo de como se pode proceder a amostragem de um sinal analógico qualquer. A cada **período de**

amostragem (*Ta*), retira-se uma amostra do sinal. Ao inverso de *Ta* dá-se o nome de **freqüência de amostragem** (*Fa*), que representa o número de amostras tomadas por unidade de tempo (MENDONCA e ZELENOVSKI, 2004).

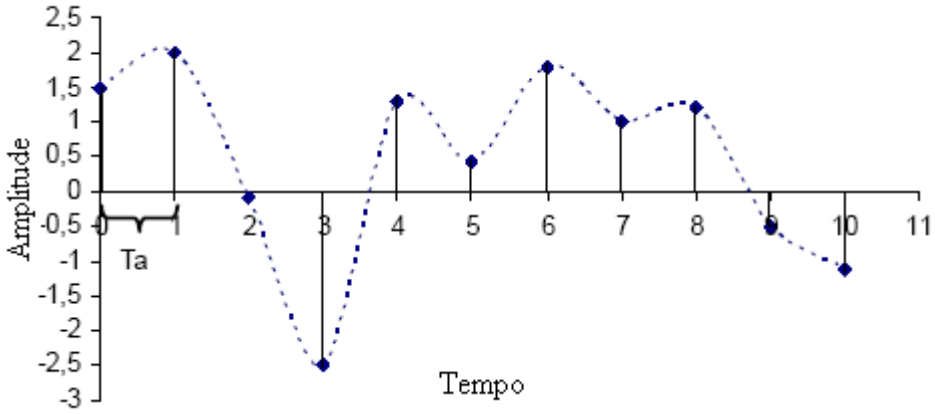

Figura 4: Exemplo de um sinal amostrado.
Fonte: YNOGUTI, 2005.

Na discretização de um sinal, é desejável que o processo de amostragem preserve o máximo possível das informações contidas no sinal original, assim o período de amostragem deve ser selecionado baseado em algum critério. O teorema de *Nyquist* garante que qualquer sinal amostrado pode ser reproduzido fielmente por um sistema digital, desde que o sinal seja amostrado a uma freqüência mínima que corresponde a duas vezes o valor da maior componente de freqüência presente no sinal, como é apresentado equação 1 (YNOGUTI, 2005):

$$f_a \geq 2 \times f_{sinal} \qquad \text{Eq. 1}$$

Por exemplo, se tivemos um sinal original com a maior componente de freqüência de 4 Khz, a taxa de amostragem deve ser, no mínimo, de 8 KHz. Outro exemplo que podemos citar, no processo gravação dos sistemas baseados em Compact Disc Áudio (CD), no qual a taxa de amostragem é de 44,1 kHz, visto que a freqüência máxima que um ouvido humano pode captar é de cerca de 20 KHz. A figura 5 mostra a amostragem de um sinal abaixo da taxa de Nyquist, o que introduz componentes de freqüências adicionais ao sinal, causando distorção definitiva no sinal original. Esse processo é conhecido como *aliasing*.

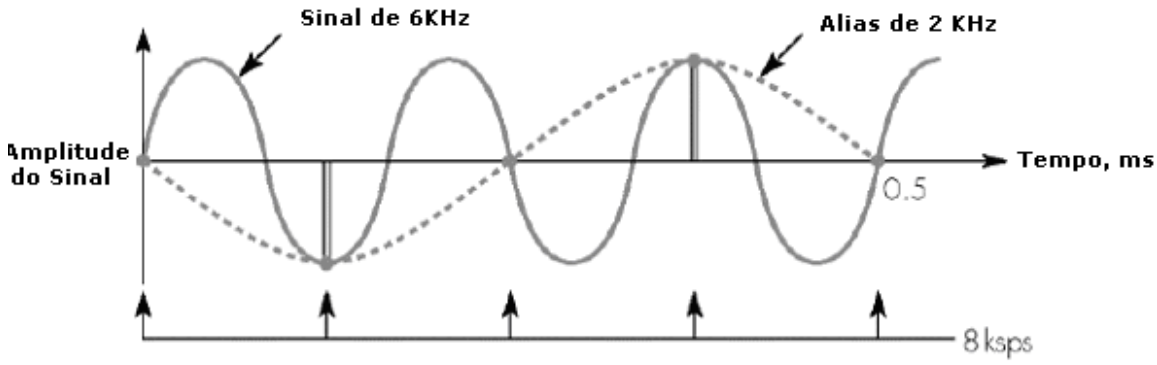

Figura 5: Sinal reconstruído com a taxa de amostragem abaixo da taxa de Nyquist.

Fonte: OLIVEIRA, 2003.

2.1.2 Processo de quantização

Outro fator que limita a qualidade do sinal a ser discretizado é o processo de quantização do sinal. A quantização representa a resolução de uma amostra no domínio da tensão. Se Vmax é o valor máximo representável e *n* é o número de bits usados por amostra, o passo de quantização *q* é dado pela equação 2 *(OLIVEIRA, 2003):*

$$q = \frac{V_{\max}}{2^n} \qquad \text{Eq. 2}$$

Por exemplo, o processo de reconstrução do sinal da figura 6 que transforma sinais de -8V a +8V em um número digital de 3 (011) a -4 (100).

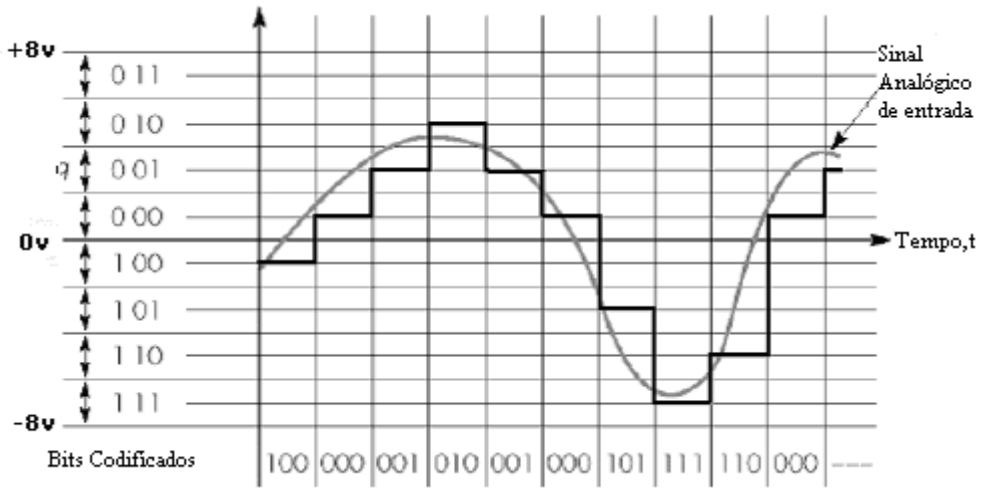

Figura 6: Erro de quantização no sinal reconstruído.
Fonte: Elaboração própria (Adaptado de OLIVEIRA, 2003).

Do exemplo apresentado na figura 6 pode-se concluir que o intervalo de quantização é de 2V. Demonstra que os valores tomados do tempo de amostragem terão passos na escala de 2V. Nota-se que não é possível representar, por exemplo, um valor de 1V.

Erro de quantização é um erro inerente ao processo de conversão, na qual os valores que não podem ser representados com a precisão requerida são arredondados. A ocorrência sucessiva deste tipo de erro gera o chamado ruído de quantização (GONÇALVES, 2005). Este erro é minimizado com o aumento da resolução do número de bits usados. Para obtermos uma resolução equivalente à de um sistema CD de áudio, por exemplo, são necessários 16 bits, o que significa que temos 65536 combinações numéricas possíveis.

a) Clipping

Um conceito muito importante quando se trabalha com sinais de áudio, é o chamado *clipping*. Uma vez que a resolução do áudio digital é determinada pelo número de bits utilizados, não é possível representar valores acima de um determinado limite. O valor mais alto que pode ser representado geralmente é expresso como sendo 0 dB. Se a amplitude da onda ultrapassa esse valor, ocorre um corte (*clipping*) da crista da onda, mudando sua forma original e ocasionando uma distorção do som, como é representado pela figura 7 (IAZZETTA, 2005).

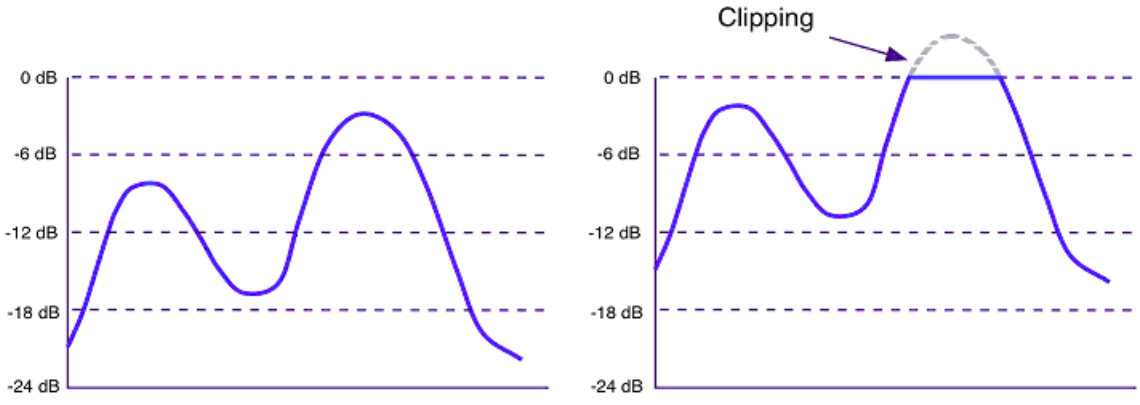

Figura 7: Clipagem num instante do sinal.

Fonte: IAZZETTA, 2005.

2.2 Conversores A/D e D/A

Os sinais analógicos devem ser digitalizados para posteriormente serem interpretados e processados pelos DSPs. Como citado anteriormente, a digitalização é realizada através dos processos de amostragem e quantização. Existem componentes eletrônicos que realizam esta conversão do sinal analógico para um número digital binário, que são chamados de conversores analógico-digital (A/D). O inverso também pode ser obtido pelos chamados conversores digital-analógico (D/A) (ALMEIDA, 2005).

Uma das aplicações mais importantes dos conversores A/D e D/A é o uso no processo de digitalização do sinal para o processamento digital. Basicamente, ele é composto por 3 fases descritas a seguir (Figura 8) (MENDONÇA e ZELENOVSKI, 2004):

- **Aquisição**: Nesta fase, os diversos sensores (de pressão, temperatura, microfones, etc.) disponibilizam informações analógicas que, ao passarem por um conversor A/D, são traduzidas para informações digitais;
- **Processamento**: Nesta fase, as informações digitalizadas provenientes dos sensores são processadas segundo algum critério ou algoritmo específico pelo DSP. Por exemplo, pode-se atenuar os sons agudos de um sinal de voz através do uso de filtros digitais;
- **Geração**: Nesta fase, os sinais que resultam do processamento de digital passam por um conversor D/A e alimentam os atuadores. Exemplo de um atuador é um auto-falante emitindo uma onda acústica.

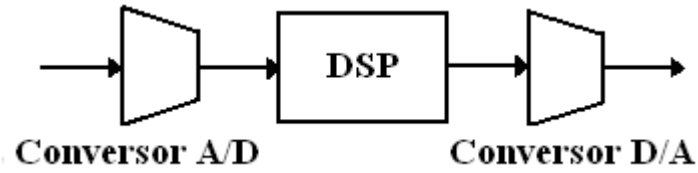

Figura 8: Aplicações dos conversores A/D e D/A

Fonte: Elaboração própria.

Deve-se frisar, que a freqüência máxima do processo de amostragem, está ligado ao tempo que o hardware do conversor leva entre processar uma amostra e outra. Já a resolução do processo de quantização é relacionada ao número de bits que o conversor possui.

A resolução do conversor do kit de desenvolvimento a ser utilizado é de 16 bits, sendo suficiente para aplicações de áudio mais simples. Porém deve-se ressaltar que conversores de 24 e 32 bits também são utilizados em processamento de aplicação de áudio profissional, que exigem alta fidelidade.

2.3 FILTROS

Um filtro é um sistema que transforma o sinal de entrada, através da aplicação de algum algoritmo, em um sinal de saída. Um filtro tem como função selecionar, rejeitar ou equalizar uma ou mais freqüências de um sinal de acordo com algum critério. Os filtros constituem uma das aplicações mais comuns da eletrônica, sendo amplamente utilizados na aquisição e processamento de sinais, como áudio e vídeo.

A figura 9 ilustra um diagrama de bloco ilustrando o funcionamento básico de um filtro. O bloco processador indicado na figura executa algum algoritmo de filtragem, já os blocos ADC e DAC são responsáveis pelas conversões analógicas e digitais do sinal.

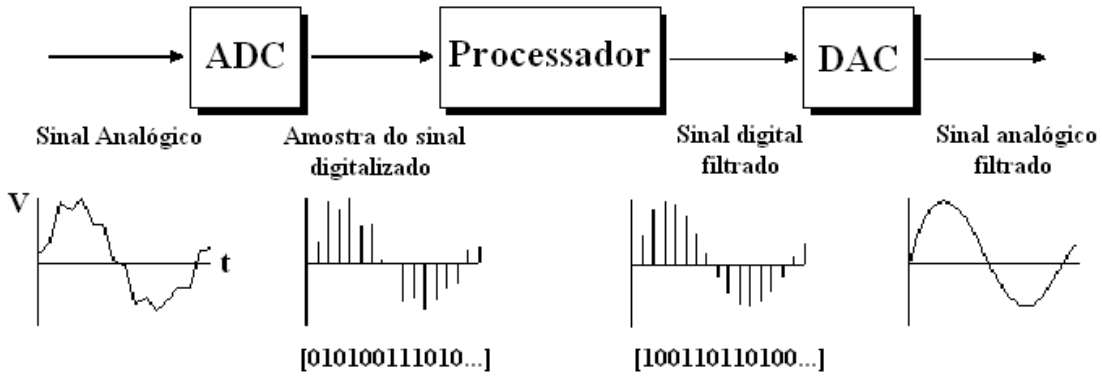

Figura 9: Filtragem de um sinal

Fonte: Adaptado de ROBIM, 2004

2.3.1 Filtros seletivos de freqüência

Os tipos mais comuns de filtros seletivos de freqüência são os filtros passa-baixa, passa-alta, passa-faixa e rejeita-faixa. As funções de transferência destes filtros caracterizam-se por permitirem, ou não, a passagem de determinados valores de freqüência do sinal original (SMITH, 1999).

a) Filtro Passa-Baixas

São filtros que permitem a passagem apenas de freqüências abaixo da freqüência de corte *Fc*. Eles atuam eliminando as componentes de alta freqüência do sinal de entrada (figura 10).

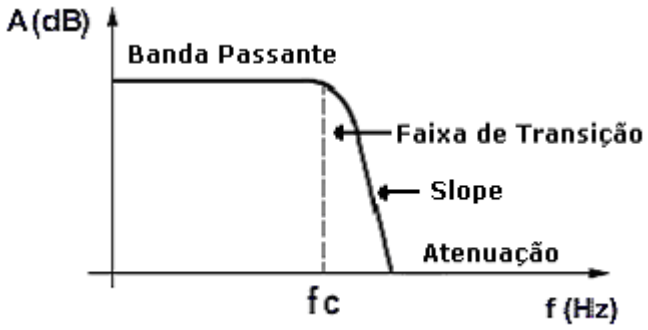

Figura 10: Filtro passa-baixas.

Fonte: Adaptado de BARBOSA, 1999

Em equalizadores de áudio, os filtros passa-baixas permitem a passagem das tonalidades mais graves do som, atenuando ou eliminando as outras.

Na figura 10, é possível observar que a transição da banda passante para a atenuação não ocorre de forma abrupta, mas sim progressivamente a partir da freqüência de corte. A área formada pela curva a partir de *Fc*, é chamada de "faixa de transição". O quão abrupto é esse corte, é definido por um parâmetro referido como *slope*, relacionado com a inclinação da reta a partir da freqüência de corte. (IAZZETTA, 2005).

b) Filtro Passa-Altas

São filtros que permitem a passagem apenas de freqüências acima da freqüência de corte *Fc*. O funcionamento dos filtros passa-altas é similar aos passa-baixas só que eles atuam eliminando as componentes de baixa freqüência do sinal de entrada (figura 11).

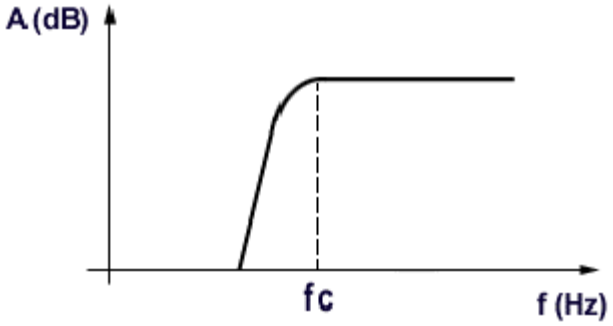

Figura 11: Filtro passa-altas

Fonte: BARBOSA, 1999.

Em equalizadores de áudio, os filtros passa-altas permitem a passagem das tonalidades mais agudas do som, atenuando ou eliminando as outras.

c) Filtro Passa Faixa

São filtros que permitem a passagem das componentes de freqüência dentro de uma faixa de valores (**Q)** que define o tamanho da largura da banda passante. Eles eliminam ou atenuam os componentes de freqüência fora desta faixa (figura 12).

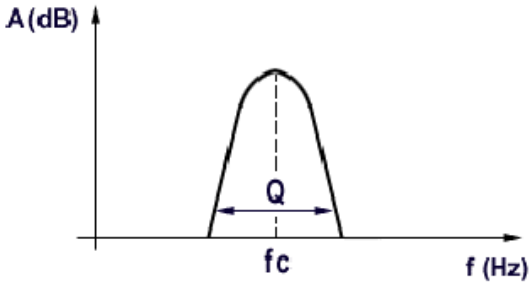

Figura 12: Filtro passa-faixas.

Fonte: BARBOSA, 1999.

Este filtro é útil quando defini-se quais componentes de freqüência é preciso manter no sinal, como por exemplo, conhecendo a faixa de freqüência de determinado instrumento musical de uma banda, é possível atenuar todos os outros instrumentos e manter apenas a freqüência selecionada.

d) Filtro Rejeita Faixa

Corresponde ao inverso do filtro passa banda, impedindo a passagem de componentes de freqüência definidos por *fc1* e *fc2***,** que definem o início e fim da banda rejeitada (figura 13).

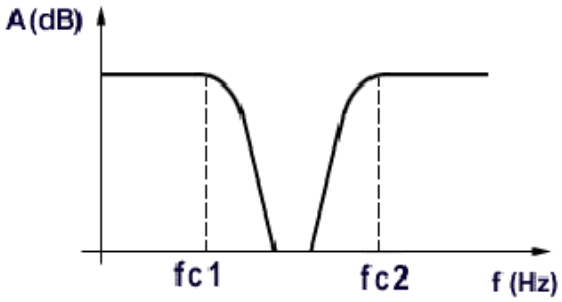

Figura 13: Filtro rejeita-faixas.

Fonte: BARBOSA, 1999.

A exemplo do anterior, pode-se definir quais componentes de freqüência atenuar, como por exemplo, conhecendo a faixa de freqüência de determinado instrumento musical de uma banda, pode-se eliminar o som produzido por este instrumento.

2.3.2 Filtros analógicos

Filtros analógicos fazem o uso de circuitos eletrônicos feitos basicamente com resistores, capacitores e amplificadores operacionais para produzir o efeito desejado. São largamente empregados em aplicações como redução de ruído, em sinais de vídeo, áudio, e em muitas outras áreas.

2.3.3 Filtros digitais

Filtros digitais utilizam processadores digitais para executar cálculos numéricos em valores amostrados de um sinal qualquer. Existem muitos softwares que podem ser utilizados para projetar filtros digitais. Os algoritmos utilizados no processamento de filtros digitais, geralmente requerem grande poder de processamento, assim, normalmente é necessário o uso de processadores especializados, como os DSPs.

a) Vantagens da utilização de filtros digitais

Segundo Wiley (2005) algumas das principais vantagens da utilização dos filtros digitais sobre os filtros analógicos são:

- Um filtro digital é reprogramável, pelo fato do seu algoritmo estar armazenado na memória de um processador. Isso significa que, a especificação de um filtro digital pode ser alterada sem a necessidade de se alterar o circuito;
- Os filtros digitais são facilmente projetados, testados e implementados em um simples computador;
- Os componentes dos filtros analógicos são sujeitos à alteração de suas características devido à temperatura, ajuste dos componentes e envelhecimento. Estes fatores não influenciam os filtros digitais;
- Os filtros digitais podem trabalhar com sinais de baixa freqüência com precisão. Com o aumento da velocidade dos DSPs, os filtros digitais estão sendo aplicados também a altas freqüências, na faixa das ondas RF, o no passado era tecnologia exclusiva dos circuitos analógicos;
- Os DSPs podem executar complexas combinações de filtros em paralelo ou série, fazendo os requisitos de hardware relativamente simples e pequenos quando comparados a um circuito analógico equivalente.

Basicamente os filtros digitais se dividem em dois grupos, os filtros de resposta ao impulso finita FIR (*Finite Impulse Response*) e os filtros de resposta ao impulso infinita IIR (*Infinite Inpulse Response).*

b) Filtros de Resposta Finita ao Impulso (FIR)

Se as saídas do sistema dependem somente da entrada presente e de um número finito de entradas passadas, então o filtro tem uma resposta impulsiva finita. O funcionamento básico de um filtro FIR é demonstrado na figura 14. Os blocos indicando Z^{-1} representam operadores de atraso unitário, ou seja, sua saída é igual à entrada apenas defasada em uma amostra. Para armazenar uma série desses elementos de atraso é utilizado um vetor de memória conhecido como linha de atraso. As amostras de entradas estão representadas por x(n-i), sendo que x(n)

representa a amostra atual do sinal de entrada. O número total de amostras de entradas utilizadas para o cálculo de cada saída é representado por N, as amostras N-1 mais recentes estão na linha de atraso (SCHWANKE, 2000).

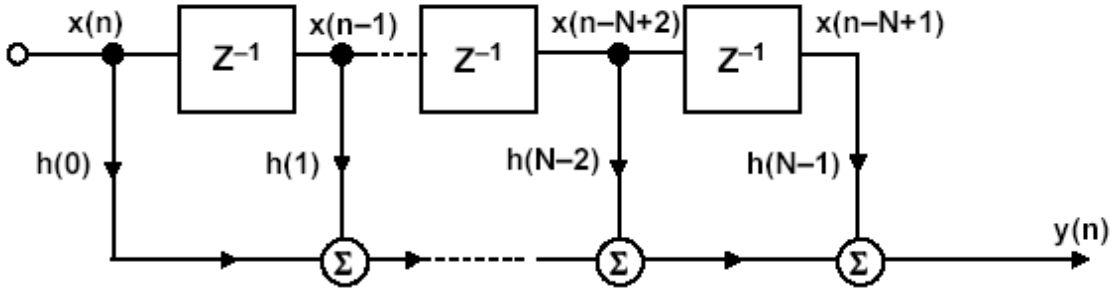

Figura 14: Estrutura de um filtro FIR.

Fonte: KESTER, 2001.

A cada amostra de entrada recebida, ocorre o deslocamento de todo o vetor da linha de atraso, ou seja, a amostra mais antiga é excluída e a atual é inserida no início da linha de atraso (SCHWANKE, 2000).

A equação que representa a saída de um filtro FIR é (SCHWANKE, 2000):

$$y(n) = \sum_{i=0}^{n-1} h(i).x(n-i) \qquad \text{Eq. 3}$$

onde **h(i)** representa os coeficientes do filtro e **y(n),** a saída atual do filtro.

c) Filtros de Resposta Infinita ao Impulso (IIR)

A resposta de um filtro de resposta infinita ao impulso é função dos sinais de entrada presentes e passados, e dos sinais de saída passados. A dependência das saídas passadas (recursividade) faz com que a duração da resposta seja infinita, mesmo quando cessarem os sinais de entrada (SCHWANKE, 2000).

Devido ao fato que nestes filtros os sinais de saída calculados farão parte no cálculo dos sinais de saída ainda por calcular, estes filtros também são chamados de filtros recursivos (figura 15).

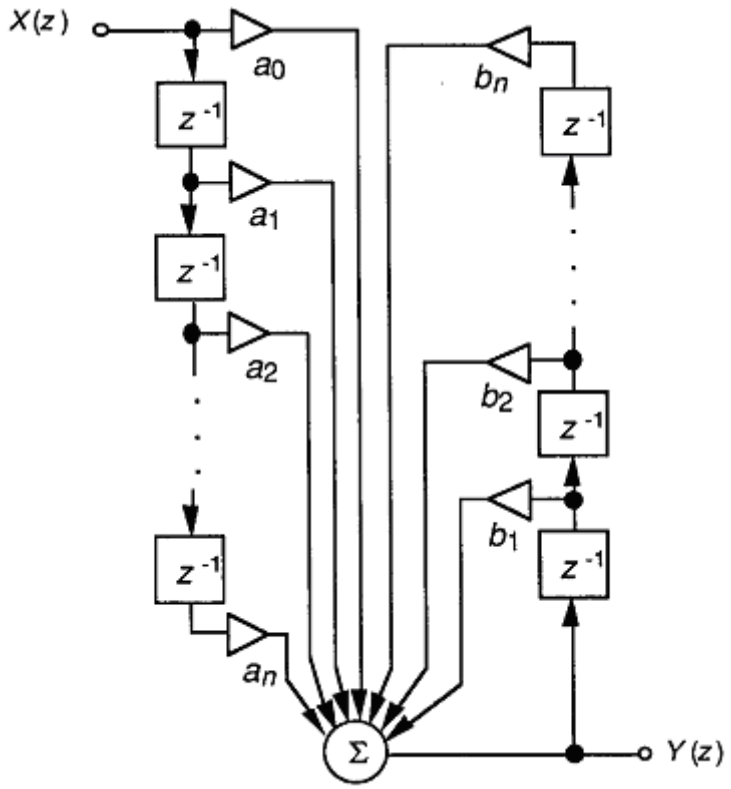

Figura 15: Estrutura para implementação de um filtro IIR.

Fonte: ROCHA, 2005.

A equação que representa a saída de um filtro IIR é (ANALOG, 2003):

$$y(n)=\sum_{k=0}^{n-1} a_k\, x(n-k)+\sum_{k=1}^{n-1} b_k\, y(n-k) \qquad \text{Eq. 4}$$

Onde $\mathbf{a_i}$ e $\mathbf{b_i}$ representa os coeficientes do filtro, **x(n)** é a entrada e **y(n)** a saída atual do filtro.

2.3.4 Comparação entre filtros IIR e FIR

A tabela a seguir cita as principais diferenças entre os filtros FIR e IIR.

Tabela 1: Comparativo entre um filtro IIR e FIR

Filtros IIR	Filtros FIR
Mais eficiente	Menos eficiente
Equivalência analógica	Sem equivalência analógica
Podem ser instáveis	Sempre estáveis
Resposta de fase não linear	Resposta de fase linear
Ordem mais baixa	Ordem alta
Pacotes de desenvolvimento disponíveis	Pacotes de desenvolvimento disponíveis

Fonte: KESTER, 2001.

2.4 SOM

Segundo Barbosa (2005), o som pode ser entendido como uma variação de pressão muito rápida, que se propaga na forma de ondas em um meio elástico. Em geral, o som é causado por uma vibração de um corpo elástico, o qual gera uma variação de pressão corresponde no meio à sua volta. Qualquer corpo elástico capaz de vibrar rapidamente pode produzir som e, nesse caso, recebe o nome de “fonte sonora”.

Para que se possa perceber o som é necessário que as variações de pressão que chegam aos ouvidos estejam dentro de certos limites de rapidez e intensidade. Se essas variações ocorrem entre 20 e 20.000 vezes por segundo esse som é potencialmente audível, ainda que a variação de pressão seja de alguns milionésimos de pascal. A figura 16 mostra uma onda senoidal representando um som puro:

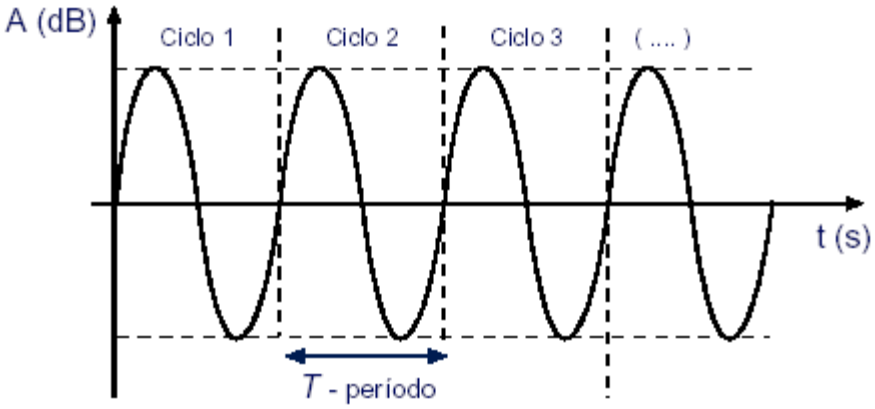

Figura 16: Onda senoidal

Fonte: BARBOSA, 2005.

Por definição, pode-se calcular matematicamente o valor da freqüência de um movimento oscilatório periódico, a partir do inverso do seu período.

Uma forma de onda correspondente a um som real, como exemplo a voz humana, nunca é pura e simples como a onda senoidal apresentada anteriormente. As formas de onda sonoras são normalmente sinais complexos, cuja variação de amplitude e freqüência ao longo do tempo não obedece a uma definição matemática simples, como apresentado na figura 17.

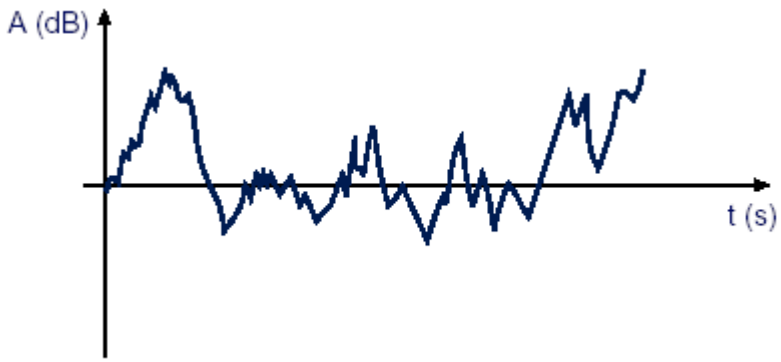

Figura 17: Sinal sonoro qualquer.

Fonte: Barbosa, 2005.

2.4.1 Componentes de um som

A seguir serão descritas as componentes de uma onda sonora.

a) Freqüência

A freqüência do som é a quantidade de vezes com que ele vibra por segundo. A unidade de medida da freqüência é o Hertz (Hz). Um som que vibra uma vez por segundo oscila a 1 Hz. As freqüências são descritas normalmente em quilohertz (kHz), a unidade que representa 1.000 Hz. O ser humano saudável pode perceber sons na faixa de aproximadamente 20 a 20.000 Hz (PEIL, 1998).

b) Amplitude

A amplitude de uma onda representa sua intensidade de energia. A unidade de medida da intensidade do som é o decibel, Abreviado com dB. A sensibilidade do ouvido humano é extraordinária, podendo perceber sons de intensidade muito alta e logo em seguida perceber um som muito baixo (PEIL, 1998)..

c) Timbre e Tonalidade

De acordo com Ratton (2005), todos os sons naturais, como as notas produzidas pelos instrumentos musicais acústicos, a voz humana, etc, possuem uma natureza oscilatória bastante complexa. Ou seja, as características de suas oscilações não são tão simples como a que foi apresentado na figura 16 (que só é obtida por sinais eletrônicos).

Tomando como exemplo a vibração de uma corda de violão (figura 18), sabe-se que seu movimento é na verdade, o resultado da superposição de vários movimentos simultâneos

O primeiro modo de vibração, que seria a oscilação do comprimento da corda inteira, é chamado de modo fundamental. Os demais modos, chamados de harmônicos, ocorrem em comprimentos que são subdivisões inteiras da corda (1/2, 1/3, ¼, etc), ou seja, os movimentos oscilatórios dos harmônicos ocorrem em freqüências múltiplas da fundamental.

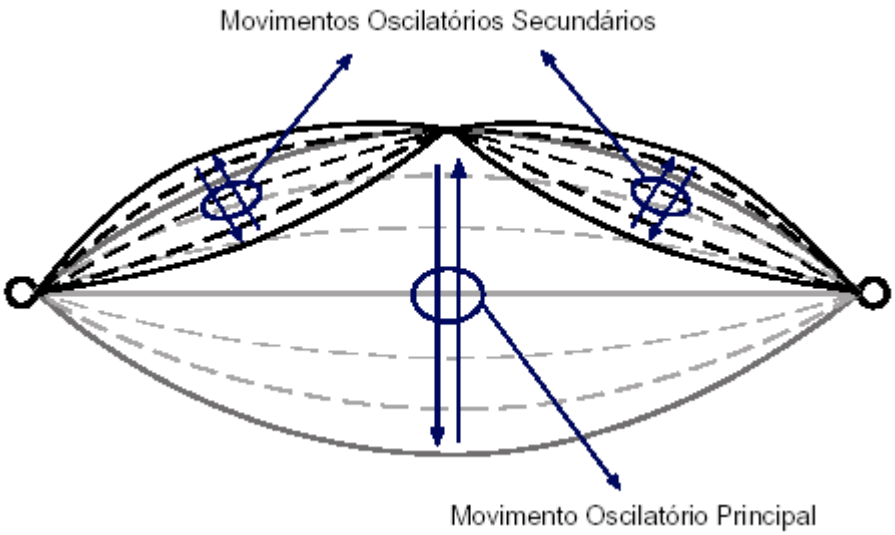

Figura 18: Oscilação numa corda de violão
Fonte: BARBOSA, 2005.

O valor da freqüência fundamental corresponde à tonalidade do som (nota musical). O conjunto dos valores de freqüência relativas às oscilações secundárias na fonte sonora, designa-se "conteúdo harmônico". O conteúdo harmônico corresponde ao timbre do som, característica que permite distinguir o suporte físico da fonte sonora (RATTON, 2005).

2.4.1 Decibel

No que se refere à intensidade, o ouvido humano pode perceber uma gama muito grande de intensidades sonoras, isto é, pode ouvir desde oscilações com muito pouca variação de pressão do ar até oscilações de pressão muito fortes. A diferença entre a pressão sonora mais fraca que podemos perceber (limiar da audição) e a pressão mais forte que podemos suportar (limiar da dor) é de mais de 1 milhão de vezes. Além disso, o ouvido não reage de forma linear às variações de nível, pois para se ter a sensação de que o nível sonoro dobrou é necessário aumentar a potência sonora em dez vezes (RATTON, 2005).

Qualquer unidade de medida de pressão (ex: N/m^2), que se queira usar para representar todos os níveis de pressão sonora que podem ser detectados pelo ouvido humano, teria uma escala tão grande que seria pouco prática (uma escala de 1 milhão de valores). Para contornar esse inconveniente, criou-se uma escala logarítmica baseada em uma medida relativa, chamada **decibel** (dB).

A medida em decibel é sempre feita em relação a um valor de referência. Por definição, o valor de uma medida de nível em decibel é calculado pela formula dB= 10.log (L / Lo), onde *L* é o nível que se está medindo, *Lo* é o nível de referência, e *log* é o logaritmo decimal.

Por exemplo, qual seria o aumento, em dB, que ocorre quando se dobra o nível de referência sonora? Pela fórmula, tem que: dB= 10log(L / Lo); como L= 2Lo, então dB= 10.log 2. Como o logaritmo de 2 é 0,30, temos então dB= 10 x 0,30. Ou seja, a cada aumento de 3 dB temos o dobro do nível de pressão sonora.

Outras variações da equação para a representação em volts e watts são: $dB_{volts}= 20\log(V_0/V_i)$ e $dB_{watts}= 10\log(P_0,P_i)$ respectivamente.

2.5 PROCESSAMENTO DIGITAL DE SINAL

2.5.1 Histórico

Os primeiros protótipos de processadores destinados para aplicações de processamento digital de sinais datam das décadas de 60 e 70. Devido aos custos elevados essas aplicações foram limitadas a somente algumas aplicações críticas. Esforços pioneiros foram feitos em quatro áreas: radar e sonar, onde a segurança nacional estava em risco; exploração de petróleo onde poderiam ser feitas grandes fortunas; exploração espacial, onde os dados devem ter uma grande precisão; e no processamento de imagens na área médica onde poderiam ser salvas vidas (DUQUE, 2004).

A revolução dos meios digitais nos anos 80 causou uma explosão de novas aplicações que necessitavam do processamento digital de sinal, levando ao desenvolvimento do primeiro DSP comercialmente viável pela empresa *Texas Instruments,* no ano de 1982. Em lugar de estar incentivado por razões militares e governamentais, o DSP foi dirigido para o mercado comercial e rapidamente, definiu seus produtos alvos como: telefones móveis, CD players, modems, etc. em uma escala muito elevada (DUQUE, 2004).

2.5.2 Introdução ao DSP

Os DSPs são microprocessadores cujo hardware, software e conjunto de instrução são otimizados para processamento de alto desempenho em aplicações numéricas. Os DSPs são utilizados em processamento digital de sinal, como por exemplo, em aplicação de áudio e vídeo, que exijam processamento em tempo real.

Os DSPs tornaram-se um componente chave no desenvolvimento de muitos produtos nas áreas comerciais, de comunicação, médica e industrial, pois eles podem realizar tarefas complexas, que seriam difíceis ou impossíveis de serem realizadas usando processadores convencionais (HAYS, 2005).

Hoje os DSPs representam o segmento que mais cresce no mercado de semicondutores e são capazes de atender a crescente demanda por processamento rápido de informações (TI, 2005).

A diferença entre os DSPs e outros tipos de processadores, está na arquitetura utilizada para combinação dos diversos elementos de projeto: controle de acesso à memória, operandos aritméticos, conjunto de instruções, paralelismo, endereçamento de dados. A importância desta arquitetura pode ser melhor assimilada a partir do entendimento da relação entre sinais de tempo real e a velocidade de cálculo do DSP. O fluxo de dados de um sinal chega ao DSP, vindo do conversor analógico/digital (A/D) em amostras individuais. Para realizar uma filtragem em tempo real, por exemplo, o DSP deve ser capaz de completar todos cálculos e operações necessárias para o processamento de cada amostra, antes que a próxima chegue, num processo que normalmente envolve ainda vários dados anteriores (SCHWANKE, 2000).

2.5.3 Arquitetura

A maioria das aplicações de processamento digital de sinal tem exigência computacional muito elevada, se comparado a outros tipos de aplicações. Os processadores DSPs possuem em sua arquitetura várias unidades de execução independentes, que são capazes de processar dados em paralelo. Por exemplo, em adição a unidade MAC (unidade de multiplicação e acumulação) geralmente há uma unidade lógico-aritmética (ALU) e uma unidade de deslocamento (*shifter*), conforme é mostrado na figura 19.

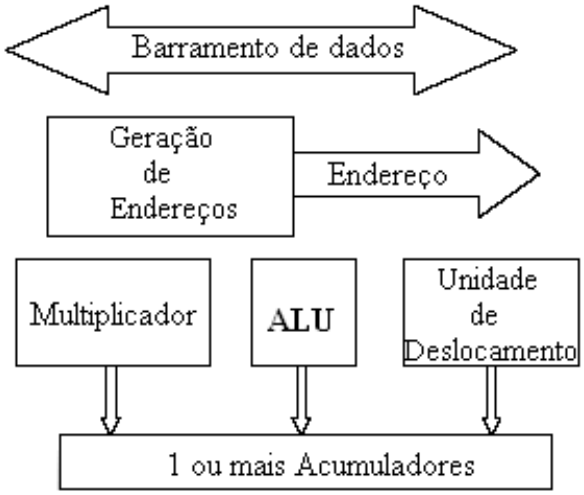

Figura 19: Diagrama de bloco de um DSP convencional.

Fonte: Elaboração própria. (Adaptado de ANALOG DEVICES, 2005).

A seguir serão descritas as funcionalidades dos principais blocos que compõem um DSP convencional.

a) Unidade de multiplicação e acumulação (MAC)

A unidade MAC é responsável pelas operações de multiplicação e acumulação. Por exemplo, na equação $Y= A_0.X(n) + B_0.X(n-1)$, a unidade MAC irá buscar os operandos A_0 e $X(n)$, efetuar a multiplicação, e acumular o resultado, para que posteriormente seja feito o mesmo procedimento para a segunda parte da equação. Inicialmente, os primeiros microprocessadores implementavam multiplicações por uma série de deslocamentos e soma, sendo que cada uma dessas instruções exigia um ou mais ciclos de clock. O primeiro processador DSP lançado no mercado (TMS32010), incorporava em seu hardware habilidade de executar multiplicações em um único ciclo. Como a multiplicação (combinada freqüentemente com a acumulação dos dados) é uma das operações mais comuns executadas pelo processador digital de sinais, e era de esperar que a maioria dos DSPs modernos incluíssem em sua arquitetura pelo menos uma unidade dedicada MAC, com ciclo único.

b) Unidade lógica aritmética (ALU)

A ALU executa operações lógicas e aritméticas tais como soma, subtração, e complemento. Seguindo a filosofia da unidade MAC, a ALU realiza as operações em um único ciclo, e geralmente um DSP possui mais de uma unidade.

c) Unidade de deslocamento (Shifter)

A unidade de deslocamento tem como objetivo deslocar um determinado número de bits de uma palavra, a fim de efetuar uma divisão (shift à direita) ou multiplicação (shift à esquerda). Nos DSPs da família AD-21xx, a unidade de shift é constituído pela soma de dois registradores de 16 bits (parte alta e parte baixa), totalizando um registrador 32 bits para poder comportar o tamanho dos dados. Nestes processadores, estão disponíveis os modos shift lógico e shift aritmético, no qual o primeiro não leva em conta o digito sinalizado, já o segundo sim.

d) Unidade de geração de endereços (AGU)

A elevada exigência de velocidade da memória, é alcançada através de um hardware dedicado para o cálculo dos endereços de memória, chamados de unidade de geração do endereço (AGU).

Esta unidade de geração do endereço, opera em paralelo com as principais unidades de execução do DSP (MAC, ALU e Shifter), permitindo assim alcançar dados em posições novas na memória sem pausas para calcular o novo endereço. Existem duas modalidades de acesso a memória nos DSPs, o acesso seqüencial e acesso circular.

O mais comum destas modalidades é acesso seqüencial, que é usado para incrementar automaticamente o ponteiro de endereços. O acesso seqüencial é usado nos algoritmos onde as rotinas são executadas de forma repetitivas e uma série de dados é armazenada seqüencialmente na memória. Sem esta característica, o programador precisaria fazer uso das instruções que incrementam explicitamente o ponteiro do endereço, gerando *overhead* (EYRE e BIER, 2000).

Muitos DSPs suportam também o endereçamento circular, que permite ao processador acessar um bloco de dados seqüencialmente, e então quando chegar ao fim do bloco, voltar automaticamente para o endereço inicial.

e) Acesso eficiente à memória

A arquitetura dos DSPs, provêm de técnicas que permite a execução de uma operação de multiplicação e acumulação (MAC) em apenas um ciclo de clock. Executar uma operação MAC significa: buscar da instrução, efetuar a operação desejada e atualizar os ponteiros da memória de endereço e da memória de dados (RIBEIRO, 2005).

Para executar as operações relacionadas acima com um bom desempenho, o DSP exige que o acesso à memória seja bastante eficiente. A solução adotada foi o desenvolvimento de uma arquitetura que a memória pudesse suportar múltiplos acessos em apenas um ciclo. A arquitetura Harward é a solução mais adotada. Na arquitetura Harvard, há barramentos; de endereços e de dados independentes. Enquanto um barramento serve para a leitura de instruções de um programa, o outro serve para a leitura e escrita de dados. Com isso, é possível operar simultaneamente em uma instrução e um byte de dados. Isso garante maior velocidade de processamento (MORELLATO, 2005).

Muitos algoritmos implementados em DSP (tais como filtros) consomem dois operandos por instrução, uma otimização geralmente utilizada, inclui um banco pequeno de memória *cache* perto do núcleo do processador. Quando um grupo pequeno de instruções for executado repetidamente (em *loop*), a *cache* é carregada com aquelas instruções, livrando o barramento de instruções de ser usado repetidas vezes sem necessidade. Esta arquitetura é chamada de Super-Harvard (HARCH). A figura 20 faz uma distinção entre a arquitetura dos processadores convencionais (Von Neumann) com a arquitetura dos processadores Harvard e Super-Harvard.

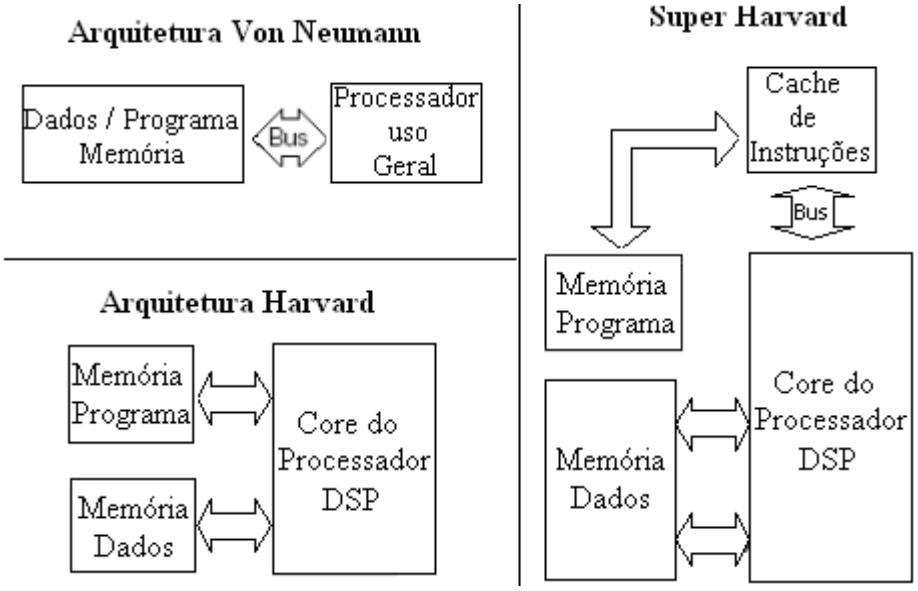

Figura 20: Distinção entre as arquiteturas Von Neumann, Harvard e Super Harvard.

Fonte: Elaboração própria. (Adaptado de SMITH, 2005).

f) Laços de repetição sem *overhead*

Os algoritmos de processamento digital de sinal geralmente gastam a maioria do seu tempo em pequenas seções do software, que são executadas repetidamente. Como exemplo nos laços de repetição. Por isso, a maioria dos DSPs dispõem em sua arquitetura suporte especial para os laços de repetição, permitindo que o programador implemente esses sem gastar nenhum ciclo de *clock* tanto para atualizar e testar o contador, quanto para desviar para o início do laço (EYRE e BIER, 2000).

g) Dispositivos de E/S

Muitos modelos de DSPs oferecem serviços de entrada e saída de dados de alta performance e de baixo custo, incorporam uma ou mais portas seriais ou paralelas. Estas portas, através de mecanismos especializados de tratamento de E/S, tais como atendimento de interrupções com baixa sobrecarga e acesso direto à memória (DMA – Direct Memory Access), possibilitam a transferência de dados com mínima ou sem intervenção direta do processador (EYRE e BIER, 2000).

h) Formato dos dados

Existem dois tipos de representação de dados na arquitetura DSP. Aqueles que realizam operações de ponto fixo e aqueles que realizam operações de ponto flutuante.

No formato ponto fixo, os números são representados por números decimais correspondendo ao intervalo entre –1.0 e +1.0. DSPs com essa arquitetura, possuem um custo mais baixo e gastam menos energia do que os de ponto flutuante, porém oferecem menor alcance de representação. Para compensar essa falta, os desenvolvedores precisam fazer um grande esforço para tratar da precisão dos números em suas aplicações.

Alguns processadores DSP, no entanto, fazem uso da aritmética em ponto flutuante, onde os valores são representados por uma mantissa e por um expoente (*mantissa* x $2^{expoente}$), sendo a mantissa um valor fracionário no intervalo entre –1.0 e +1.0, enquanto o expoente é um inteiro que especifica o lugar do ponto binário, análogo ao ponto decimal nos números em base 10. (RIBEIRO, 2005). Dados em ponto flutuante permitem uma faixa maior de representação, e eliminam o perigo da fidelidade numérica (evitam o *overflow).* Em contra partida essa arquitetura exige hardware mais complexo, ocasionando um preço mais elevado e maior consumo de energia.

Desde que a precisão numérica é mais fácil de ser mantida usando o formato de ponto flutuante, pode parecer surpresa que a maioria dos processadores DSP utiliza ponto fixo. Em muitas aplicações, no entanto, os processadores DSPs se defrontam com algumas restrições como: baixo custo e baixo consumo de energia.

O tamanho da palavra de dados também influencia nos parâmetros de custo e consumo, já que quanto maior o número de bits, mais complexo é o hardware. Os processadores DSP tendem a usar o menor tamanho da palavra de dados possível que irá fornecer uma precisão adequada para atingir os resultados da aplicação. A maioria dos processadores de ponto fixo utiliza palavra de dados de 16 bit, pois já é suficiente para a maioria das aplicações. Poucos processadores de ponto fixo utilizam 20, 24 ou até mesmo 32 bits para possibilitar melhor precisão em aplicações que exigem maior fidelidade.

Processadores com arquitetura de ponto fixo utilizam hardware especializado para manter a fidelidade numérica. Por exemplo, a maioria dos DSPs possuem um ou mais acumuladores para guardar os resultados de soma e multiplicação. Os acumuladores geralmente são maiores do que os outros registradores, e oferecem bits extra (*guard bits)* com o objetivo de estender a

representação numérica, evitando o *overflow*. Os DSPs incluem bom suporte para saturações aritméticas, arredondamentos e deslocamento, todos quais são úteis para manter a precisão numérica. (EYRE e BIER, 2000).

i) Conjunto de instruções especializadas

O conjunto de instruções do DSP foi projetado com dois objetivos. O primeiro objetivo é utilizar ao máximo os recursos do hardware, aumentando assim a eficiência. O segundo objetivo é minimizar a quantidade de dados armazenados em memória, desde que a relação custo-benefício das aplicações seja atendida (EYRE e BIER, 2000).

Para atender o primeiro objetivo, o conjunto de instruções permite que o programador especifique diversas operações para executar em paralelo, em uma única instrução. Geralmente é incluído uma ou duas buscas dos dados na memória (junto com a atualização do ponteiro de endereço) com a operação aritmética principal.

A tabela 2 apresenta como exemplo, um trecho de código em linguagem assembler para DSP da família ADSP21xx da Analog Devices que serve para ilustrar as características citadas até aqui.

Tabela 2: Algoritmo FIR

```
i2 = ^DADOS;
i4 = ^COEF;
m2 = 1;
m4 = 1;
mr = 0, mx0 = dm(i2,m2), my0 = pm(i4,m4);
cntr = 10;
do laco until ce;
   laco: mr = mr + mx0 * my0, mx0 = dm(i2,m2), my0 = pm(i4,m4);
```

Fonte: SCHWANKE, 2000.

O segmento de código inicia com a configuração das unidades de endereçamento, carregando-se os registradores de endereçamento indireto *i2 e i4* com os endereços dos vetores DADOS e COEF e inicializando os registradores de modificação m2 e m4 com o valor 1. A seguir, uma

instrução múltipla de único ciclo zera o acumulador *mr* com o conteúdo do primeiro elemento do vetor DADOS e o registrado *my0* com o conteúdo do primeiro elemento do vetor COEF. Além disso, os ponteiros *i2* e *i4* são auto-incrementados em *m2* e *m4* posições, respectivamente, ainda no mesmo ciclo. A última linha é um laço que será executado *cntr* vezes (10, neste exemplo), realizando a cada iteração uma operação MAC e dois acessos à memória para atualizar a cada iteração uma operação MAC e dois acessos à memória para atualização dos próximos operandos da multiplicação. Ao final de toda operação, o registrador *mr* deverá conter o resultado do produto vetorial dos dez primeiros elementos de cada vetor (SCHWANKE, 2000).

Para atender o segundo objetivo, os DSPs fazem uso de pequenas instruções (assim usando menos memória de programa), restringindo os registradores que podem ser usados com as operações, e restringindo as operações possam ser combinadas em uma mesma instrução. Para reduzir ainda mais o número de bits requeridos para codificar as instruções, os DSPs oferecem poucos registradores em comparação a outros tipos de processadores (EYRE e BIER, 2000).

Como resultado dessas otimizações, é de esperar que os processadores DSPs tendem a ser altamente especializados com conjunto de instruções irregulares e complicados. Estas características são vistas como um inconveniente significativo para estes processadores, porque complica a tarefa de criar um software eficiente. Por esse motivo, os programadores são forçados a escreverem seus programas na linguagem assembler para obter o máximo aproveitamento do processador. Desenvolver programas em linguagem de alto nível como C/C++ implica no não aproveitamento das características do DSP, devido ao fato que os compiladores não são hábeis o suficiente para gerar um código otimizado devido à complexidade do hardware. Isso não acarreta que os programas não possam ser escritos em alto nível, porém é necessária otimização dos trechos críticos em linguagem assembler.

2.5.4 A importância da faixa dinâmica do DSP em processamento de áudio

Uma das considerações muito importantes quando se projeta um sistema de áudio, é determinar uma qualidade aceitável para a aplicação no qual irá desenvolver. Quando é referenciada a frase “qualidade de CD” na maioria dos produtos comerciais, está se referindo a faixa dinâmica (*dynamic range*) do sistema, que é subentendida como a qualidade do sinal (TOMARAKOS e LEDGER, 1998).

A tabela abaixo relaciona comparação da faixa dinâmica entre alguns dispositivos e aplicações.

Tabela 3: Algumas comparações entre faixas dinâmicas

Aplicativo/Sistema de Áudio	Qualidade do Sinal
Rádio AM	48 dB
Rádio FM	70 dB
TV Analógica	60 dB
Fita de Vídeo	75 dB
Conversores de 16 Bits	90 a 95 dB
Tv Digital	85 dB
Mini-Disk Player	90 dB
CD Player	92 a 96 dB
Conversores de 18 Bits	104 dB
Conversores de 20 Bits	110 dB
Conversores de 24 Bits	110 a 120 dB
Microfone Analógico	120 dB

Fonte: TOMARAKOS e LEDGER, 1998.

Os níveis são medidos em dB, considerando a equação (TOMARAKOS e LEDGER, 1998):

$$DynamicRange(dB) = 6.02n + 1.76dB \simeq 6\text{n}$$

Onde **n** é o número de bits usados na quantização do conversor e a constante 1.76dB é baseada em estatísticas na forma de onda senoidal.

A faixa dinâmica é determinada pela diferença entre o nível mais alto de sinal que o sistema pode operar e o nível de ruído existente (*noise floor*) conforme mostrado na figura 21.

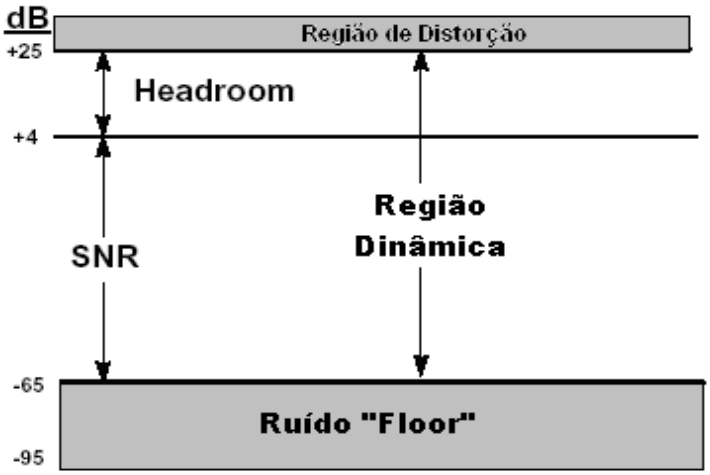

Figura 21 : Relação entre a faixa dinâmica, SNR e *headroom*.

Fonte: Elaboração própria (Adaptado de TOMARAKOS e LEDGER, 1998).

Como geralmente o sinal de áudio não possui uma amplitude constante, pois há momentos em que a sinal fica mais forte e outros em que a sinal fica mais suave, o nível nominal de operação deve ser tal que permita aumentos de sinal sem que haja saturação (clipping). A essa "folga" do nível de sinal dá-se o nome de *headroom* (RATTON, 2005).

O nível de sinal/ruído (SNR) é a diferença entre o nível de operação *(nominal eletronic line level*) e o ruído existente.

Levando em conta as considerações feitas acima, a escolha do DSP a ser utilizado deve ser feita de maneira que a aplicação de áudio a ser implementada, não sofra perda de qualidade em função da faixa dinâmica.

2.5.5 Conclusão

A utilização de processadores DSP, cuja arquitetura de hardware e software é otimizada para o tratamento digital de sinais, tem se mostrado uma alternativa promissora em relação ao uso de processadores genéricos. O desempenho de tais processadores DSP em aplicações numéricas intensivas é superior e a relação de custo vem se reduzindo favoravelmente.

Atualmente, além das empresas que desenvolvedoras dos DSPs estarem concentradas em reduzir os custos e aumentar a performance dos processadores, elas tem se voltado de forma intensiva no desenvolvimento de novas tecnologias que permitam os programadores a utilizarem linguagens de alto nível com eficiência.

2.7 EQUALIZADOR

2.7.1 Introdução

O termo equalização tem origem nos antigos sistemas telefônicos, no qual eram usados para compensar as deficiências de tonalidade causadas pelos precários sistemas. Um equalizador (EQ) é um sistema designado para alterar a tonalidade do som (SOUND ON SOUND, 2001). O equalizador é formado por um número de filtros, que são capazes de aplicar ganhos ao sinal de áudio, dentro de uma freqüência específica.

Uma das primeiras e mais rudimentar forma de equilíbrio de freqüências em sistemas sonoros é o controle de "tone" encontrado em muitos aparelhos domésticos simples, porém não muito eficiente, pois conseguem no máximo dar um ganho nos graves ou agudos (TERAHATA, 2003).

No mercado podemos encontrar três principais tipos de equalizadores: os equalizadores gráficos, os paramétricos e os de controle de tonalidade. As características de cada estão apresentadas a seguir.

2.7.2 Equalizador gráfico

O equalizador gráfico é constituído por vários filtros do tipo passa banda que cobrem todo o espectro auditivo, porém com suas freqüências centrais e largura de banda (Q) são fixas.

O usuário tem à disposição aquele conjunto de freqüências, para manipular uma a uma, o que é útil quando várias freqüências precisam ser manipuladas ao mesmo tempo (TERAHATA, 2003).

A largura de banda (Q) será determinada pela quantidade de bandas que o equalizador tiver. A largura de banda normalmente é classificada em 1/3, 1/2, 2/3 ou 1/1 oitava, que correspondem a 31, 20, 15 ou 10 filtros peak respectivamente e igualmente espaçados entre si para cobrir o todo o espectro de áudio (TERAHATA, 2003). A figura 29 ilustra a curva de um equalizador com 5 bandas.

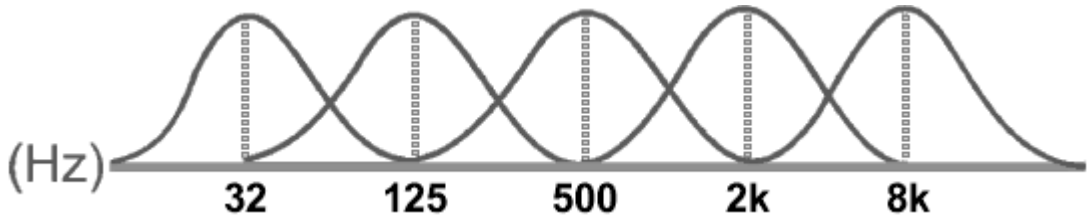

Figura 29: Filtros peak cobrindo todo o espectro de áudio.

Fonte: TERAHATA, 2003.

Para definir o espaçamento das bandas, é necessário determinar a razão entre as freqüências dos pontos de -3 dB (BW_{-3}) dos filtros do equalizador, dependendo da classificação da largura da banda (1/3, 1/2, 2/3 ou 1/1 oitava) escolhida. Como por exemplo, a razão será igual a: $2^{1/3} = 1,26$ ou $2^{1/2} = 1,41$ ou $2^1 = 2$ conforme a classificação escolhida (NEIVA, 2002). Os filtros usados nos equalizadores são simétricos em relação à sua freqüência central fc, a qual será sempre a média geométrica de f_1 e f_2.

Por exemplo, um filtro com f_c = 1 kHz e BW_{-3} = 1 oitava teria f_1 e f_2 em 707 e 1414 Hz aproximadamente, de forma que $f_2/f_1 = 2$ (1 oitava). A largura da faixa de passagem (em Hz) será de 707 Hz.

Para um filtro com f_c = 2000 Hz, os pontos seriam em 1414 e 2828 Hz mantendo a mesma porcentagem de f_c, mas com largura em Hz de 1414 Hz.

Devido a rigidez de suas bandas, em relação às freqüências, este equalizador torna-se menos eficiente na correção de timbres, sendo o mais indicado para esta tarefa o equalizador paramétrico (TERAHATA, 2003).

A figura 30 mostra os controles de um canal do equalizador gráfico CGE2101S da empresa CICLOTRON. Este equalizador gráfico possui 2 canais, 10 bandas de 1 oitava centradas na norma ISO, com filtros de Q-constante. As montagens dos circuitos são feitas exclusivamente de componentes analógicos.

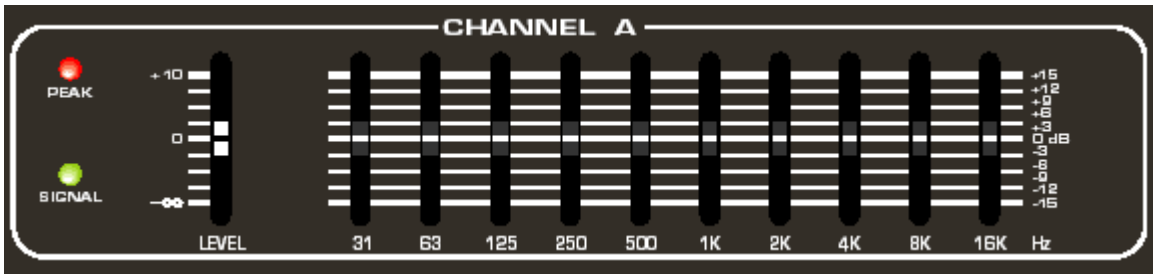

Figura 30: Equalizador gráfico da empresa Ciclotron.

Fonte: CICLOTRON, 2005.

2.7.3 Equalizador paramétrico

Os equalizadores paramétricos apresentam menores quantidades de faixas (em geral de uma a quatro bandas), mas são os que têm mais controles. Este tipo de equalizador possui seções com controles que atuam de forma independente sobre os três parâmetros principais de um filtro que são (IZECKSOHN, 1998):

- Freqüência central;
- Largura de banda passante (Q);
- Ganho em dB.

O usuário escolhe exatamente a freqüência central que deseja manipular em cada banda, a largura da banda, ou quantas freqüências vizinhas, e a amplitude dessa banda. Um controle determina a freqüência central, o outro controla a largura da banda e um terceiro controle reforça ou atenua o nível dessa faixa de freqüências. É útil quando se quer mexer em algumas bandas, mas com precisão, para só afetar as freqüências que realmente precisam de equalização (IZECKSOHN, 1998).

A figura 31 esboça parte de um equalizador paramétrico profissional da marca Behringher de 4 bandas. A banda 1 apresentada, o operador pode selecionar a freqüência central entre 20Hz a 400Hz. O valor de Q varia entre 0.03 e 2, e já o ganho é de –15 a +15 dB.

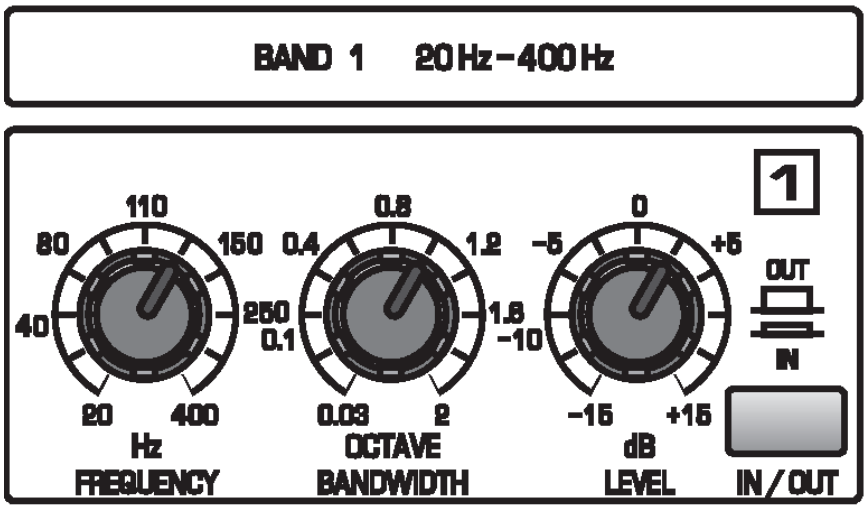

Figura 31: Ilustração do equalizador paramétrico da empresa Behringher.

Fonte: BEHRINGHER, 2005.

a) Implementação

Os equalizadores paramétricos são compostos por um número finito de filtros que os usuários podem manipular. Cada filtro é implementado utilizando filtros de resposta infinita ao impulso (IIR) de segunda ordem, no qual os coeficientes são calculados em tempo de execução, dependendo dos parâmetros que são dados como entrada. O cálculo em tempo real é necessário devido à impossibilidade de armazenar previamente todos os coeficientes para as diferentes combinações de valores de entrada.

2.7.4 Equalizadores de controle de tonalidade

É a configuração mais simples de equalização, e podem ter apenas dois controles, um para graves (baixas freqüências), atuando em torno de 100 Hz, e outro para agudos (altas freqüências) atuando em torno de 10 kHz, cada controle permitindo uma variação de ganho entre + 15 e –15 dB nas freqüências nominais e acima ou abaixo destas, com uma queda suave de 4dB/oitava. Nesta configuração, podem ser usados apenas um filtro *hi-shelf* e um *low-shelf*.

3 PROJETO DO FILTRO DIGITAL

Neste capítulo será apresentada a síntese dos filtros IIR utilizados pelo equalizador gráfico de 10 bandas.

3.1 SÍNTESE DO FILTRO IIR

A estratégia escolhida para o desenvolvimento dos filtro apresentada, é encontrada em (MONTGOMERY, 2001). O equalizador gráfico de 10 bandas é projetado a partir do uso de 10 filtros digital IIR passa banda em paralelo para cada canal de áudio (estéreo). O *codec* do kit amostra o áudio de entrada a uma freqüência de 48.000 Hz e 16 bits. As freqüências centrais destes filtros variam de 0 Hz a fs/2 (onde fs é a freqüência de amostragem de 48.000 Hz). A figura 32 apresenta o circuito RLC passivo que forma um filtro passa banda analógico.

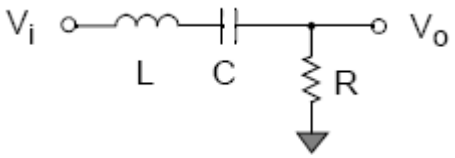

Figura 32: Circuito RLC passivo.
Fonte: MONTGOMERY, 2001.

O filtro digital IIR, discutido anteriormente, é baseado neste circuito. Partindo deste, pode-se determinar a função de transferência de segunda ordem que é apresentado na equação abaixo.

$$H(s) = \frac{V_o}{V_i} = \frac{Rs}{Rs + Ls^2 + \frac{1}{C}} \qquad \text{Eq. 5}$$

Aplicando a transformada de Laplace na função de transferência acima, tem-se a representação no plano S, que é mostrado na **Equação 6**.

$$\frac{V_o}{V_i} = \frac{R}{R + j(2\pi f)L + \frac{1}{j(2\pi f)C}} \qquad \text{Eq. 6}$$

onde $s = j(2\pi f)$.

Aplicando a transformada bilinear entre o plano S e o plano Z (**Equação 7**) temos:

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \qquad \text{Eq. 7}$$

Onde $z = e^{j\theta}$, $\theta = \omega T = (2\pi f)(1/f_s)$, e T é o período de amostragem $(1/f_s)$.

Usando a **Equação 2**, é encontrada a função de transferência do plano Z a partir da **Equação 6** (OGATA, 1998).

$$H(z) = \frac{\alpha(1 - z^{-2})}{\frac{1}{2} - \gamma z^{-1} + \beta z^{-2}} \qquad \text{Eq. 8}$$

Com a **Equação 8** os coeficientes para cada filtro são calculados usando as três seguintes equações:

$$\beta = \frac{1}{2}\frac{1 - \tan\left(\frac{\theta_o}{2Q}\right)}{1 + \tan\left(\frac{\theta_o}{2Q}\right)} \qquad \text{Eq. 9}$$

$$\gamma = \left(\frac{1}{2} + \beta\right)\cos\theta_o \qquad \text{Eq. 10}$$

$$\alpha = \left(\frac{1}{2} - \beta\right)/2 \qquad \text{Eq. 11}$$

Onde $Q = f_o/(f_2 - f_1)$ e $\theta_o = 2\pi(f_o/f_s)$. O valor f_o é freqüência central do filtro passa banda, f_1 e f_2 são os pontos abaixo da freqüência central (onde o ganho é igual a $1/(\sqrt{2})$).

Para implementar a função de transferência da **Equação 8** como um filtro IIR digital, é necessária a transformação da equação para o domínio do tempo, cujo é mostrada na **Equação 7**.

$$y(n) = 2\{\alpha[x(n) - x(n-2)] + \gamma y(n-1) - \beta y(n-2)\} \qquad \text{Eq. 12}$$

Pode-se fazer a representação na forma de diagrama de blocos da equação acima, conforme a figura 33.

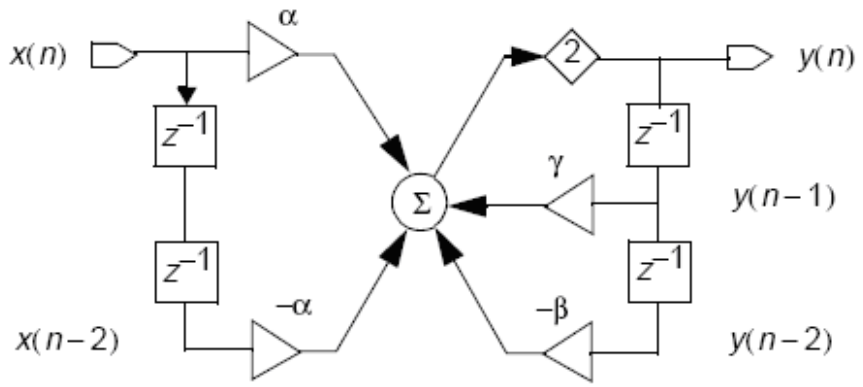

Figura 33: Diagrama de blocos da Equação 12.

Fonte: MONTGOMERY, 2001.

A tabela 3 apresenta os coeficientes das 10 freqüências centrais escolhidas para este trabalho.

Tabela 3: Coeficientes α, ß e γ para cada freqüência central.

Freqüência Central	α	ß	γ
31 Hz	0.000723575	0.49855285	0.998544628
62 Hz	0.001445062	0.497109876	0.997077038
125 Hz	0.002904926	0.494190149	0.994057064
250 Hz	0.005776487	0.488447026	0.987917799
500 Hz	0.011422552	0.477154897	0.975062733
1000 Hz	0.02234653	0.455306941	0.947134157
2000 Hz	0.04286684	0.414266319	0.88311345
4000 Hz	0.079552886	0.340894228	0.728235763
8000 Hz	0.1199464	0.2601072	0.3176087
16000 Hz	0.159603	0.1800994	-0.4435172

A cada período de amostragem, uma amostra do canal direito e esquerdo passarão pelos mesmos 10 filtros. Após cada filtro selecionar a sua freqüência correspondente, a saída é modificada pelo ganho em particular de cada filtro. E finalmente, o resultado da saída de cada filtro será somado. A figura 34 mostra o banco de filtros dispostos em paralelo.

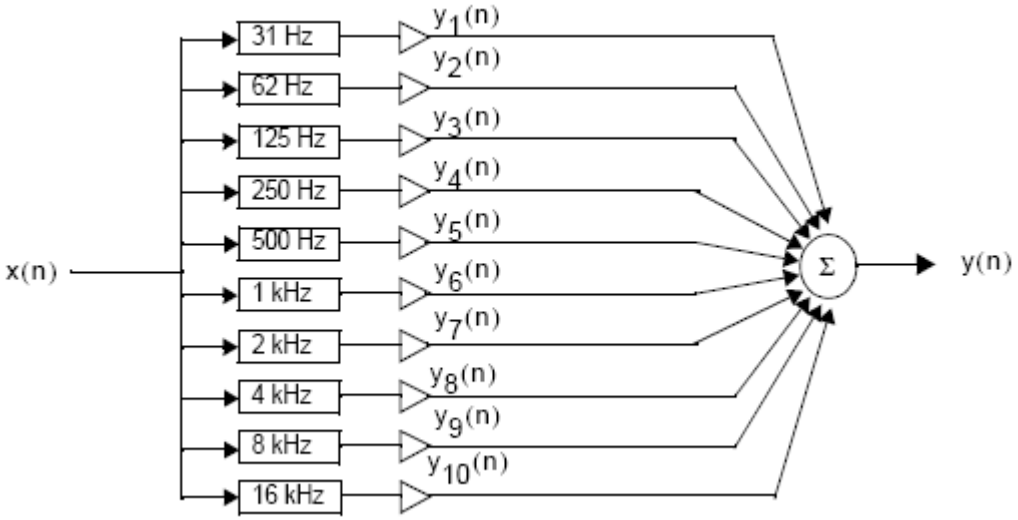

Figura 34: Diagrama de blocos representando um banco de filtro em paralelo.

Fonte: MONTGOMERY, 2001.

4 RESULTADOS DE SIMULAÇÕES

Neste capítulo serão apresentados os resultados de simulações com o Matlab e com o simulador do DSP.

4.1 RESULTADOS OBTIDOS COM O MATLAB

Inicialmente, foi validada a curva de resposta para cada freqüência central mostrados na tabela 3. Para isso, o Matlab dispõe de um comando chamado *freqz(B,A,N,F)*, que verifica a resposta em freqüência da transformada Z de um filtro digital. Toda função de transferência segue o seguinte modelo:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}}{a(1) + a(2)z^{-1} + \cdots + a(m+1)z^{-m}} \qquad \text{Eq. 13}$$

Adaptando a equação acima com a **Equação 8**, pode-se extrair os coeficientes B e A que formam o numerador e denominador respectivamente. N é o número de pontos em que se quer avaliar a resposta e F é a freqüência de amostragem. Para o filtro de 16 kHz temos:

```
A= [0.159603; 0; -0.159603];
B= [0.5; 0.4435172; 0.1800994];
N= 4096;
F= 48000;
freqz(A,B,N,F);
```

Com esses comandos, o Matlab apresentará o gráfico da figura 35, com a curva de ganho e fase para o filtro de 16 Khz.

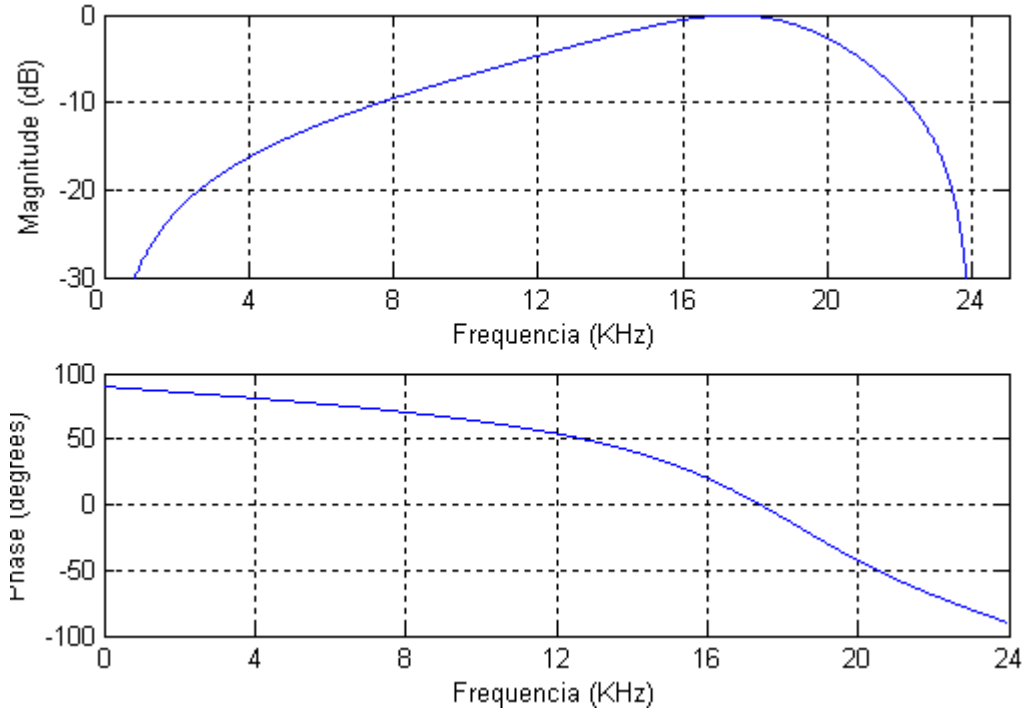

Figura 35: Curva de resposta de magnitude e a resposta de fase para o filtro de 16 Khz.

Fonte: Elaboração própria.

A figura 36 mostra a curva de resposta e fase do filtro de 2 Khz.

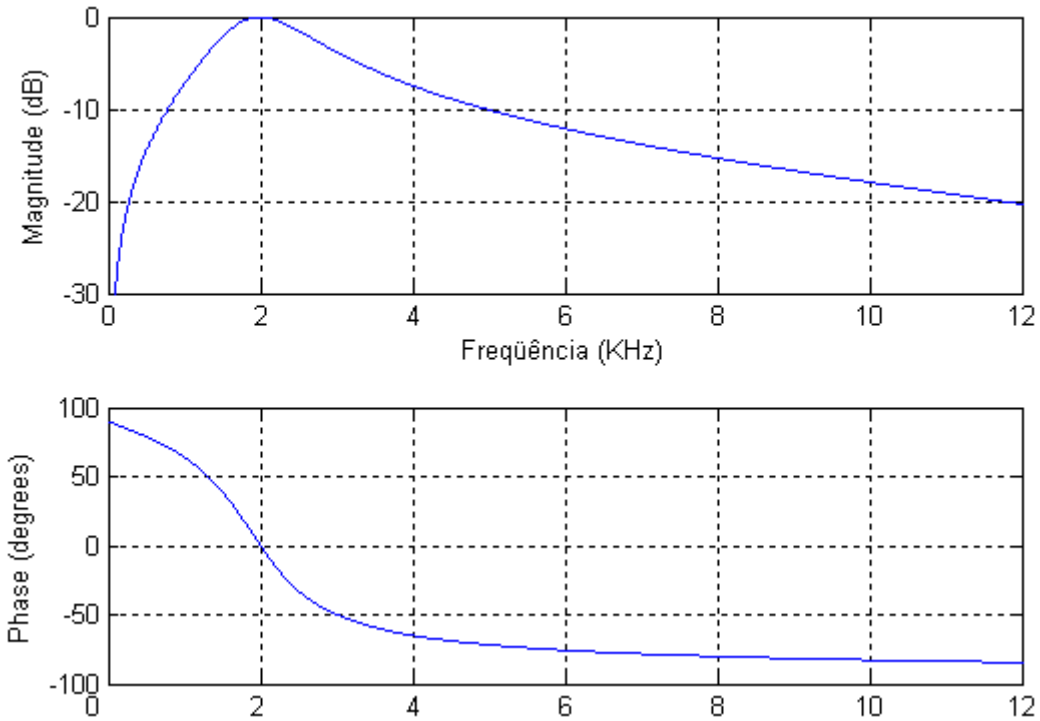

Figura 36: Curva de resposta e a resposta de fase para o filtro de 16 Khz.

Fonte: Elaboração própria.

Observa-se que a curva de resposta de cada filtro, atinge o valor de zero dB de amplificação no seu respectivo valor de freqüência. Ou seja, a intensidade de uma componente de entrada igual a freqüência do respectivo filtro não sofrerá alteração na saída na magnitude, apenas na fase.

Após todos os filtros serem validados individualmente, foram plotados a união deles, formando a figura 37.

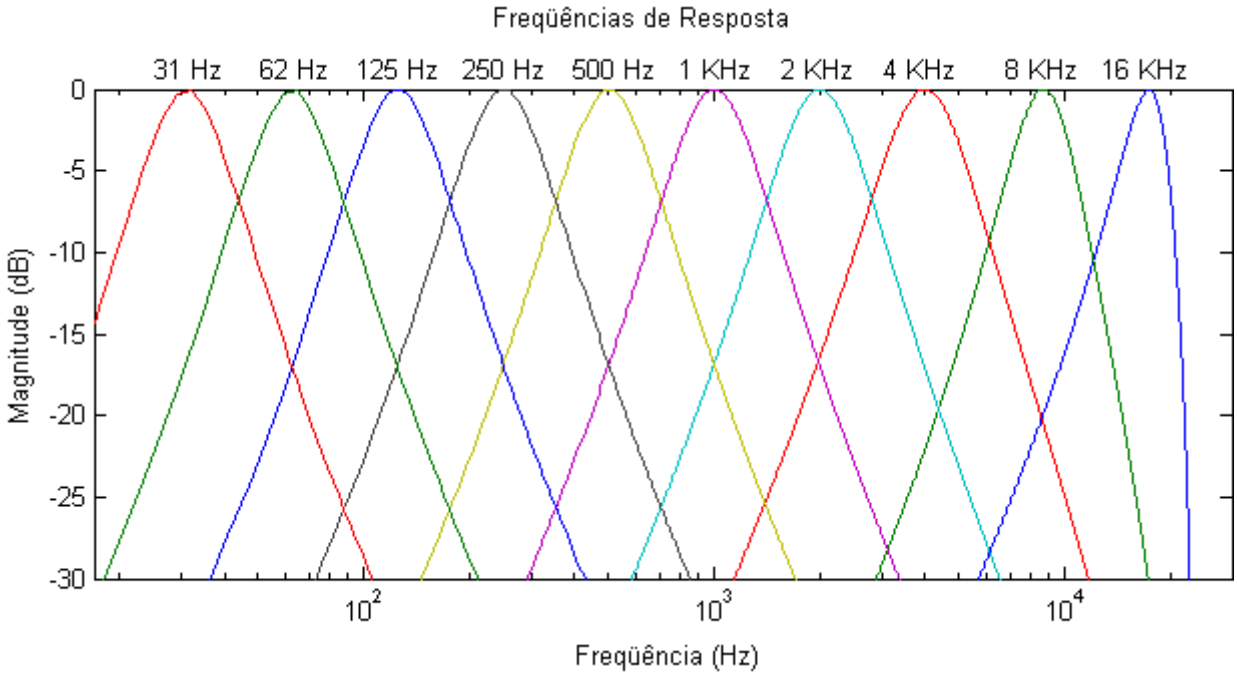

Figura 37: Curva de resposta de todos os filtros.

Fonte: Elaboração própria.

Foi desenvolvido na ferramenta gráfica do Matlab (GUI), um programa que contivesse todas as freqüências do filtro e que pudessem ser simuladas de modo rápido e de fácil visualização. A figura 38 apresenta a tela do programa.

No menu esquerdo, existem 2 opções: a primeira é onde seleciona a freqüência do filtro a ser testado e a segunda é o ganho a ser aplicado na saída do filtro. Neste exemplo, foi selecionado o filtro de 16 Khz. O programa fornece como entrada a soma de várias freqüências ao longo do espectro auditivo, como pode ser observado no primeiro gráfico da figura. O segundo

gráfico da figura mostra o sinal já filtrado pelo filtro correspondente. Já, o terceiro gráfico, apresenta a curva de resposta deste filtro.

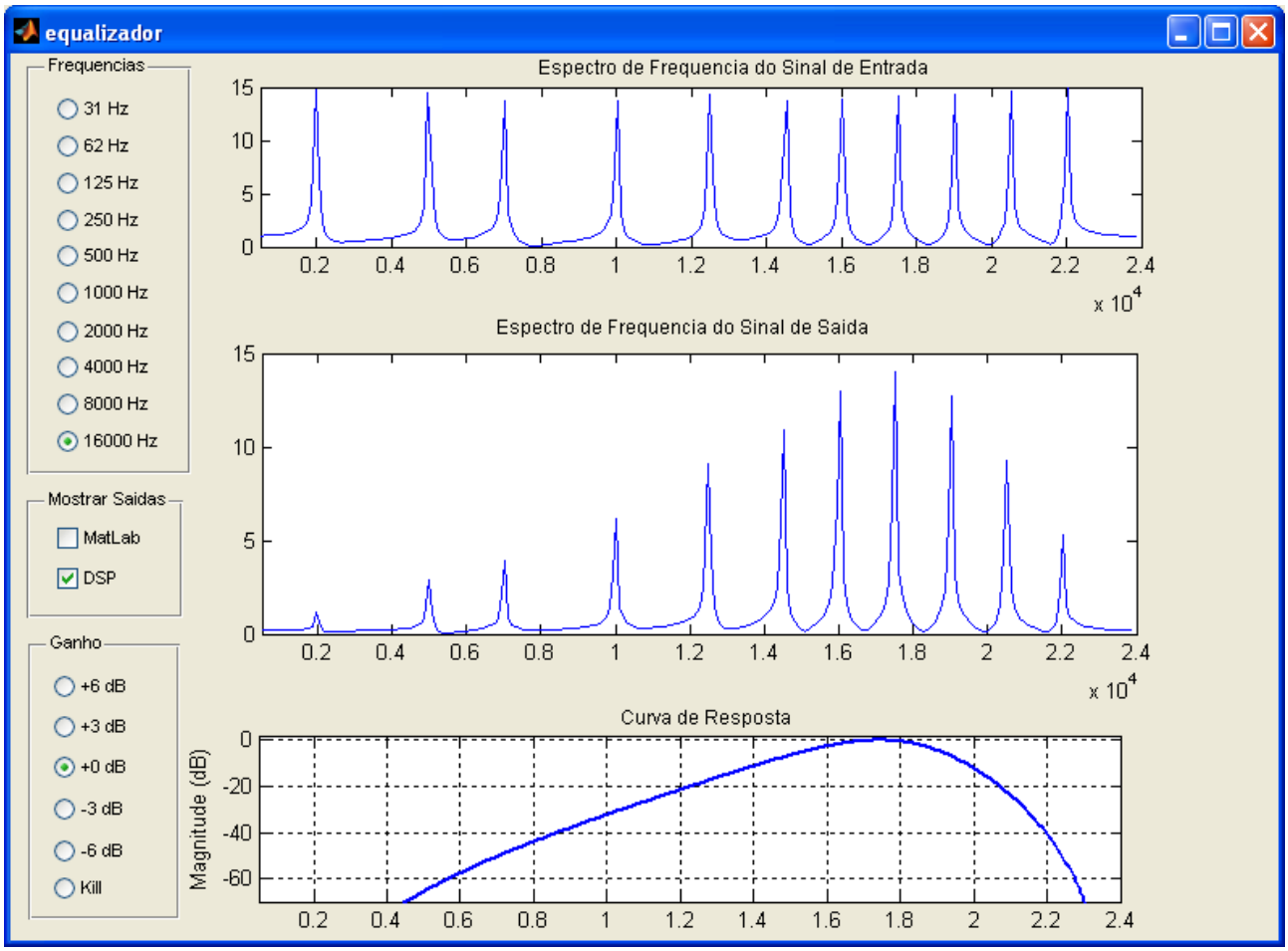

Figura 38: Interface gráfica para teste individual dos filtros.

Fonte: Elaboração própria.

Observa-se na figura 38 que a intensidade do espectro de freqüência do sinal de saída acompanha a linha da curva de resposta do filtro.

Após a validação do funcionamento individual de cada filtro, foi criada uma nova interface gráfica no Matlab, que fizesse a soma de todos os 10 filtros, como foi visto na figura 34. A interface apresentada na figura 39 tem como objetivo controlar o ganho de cada filtro individualmente.

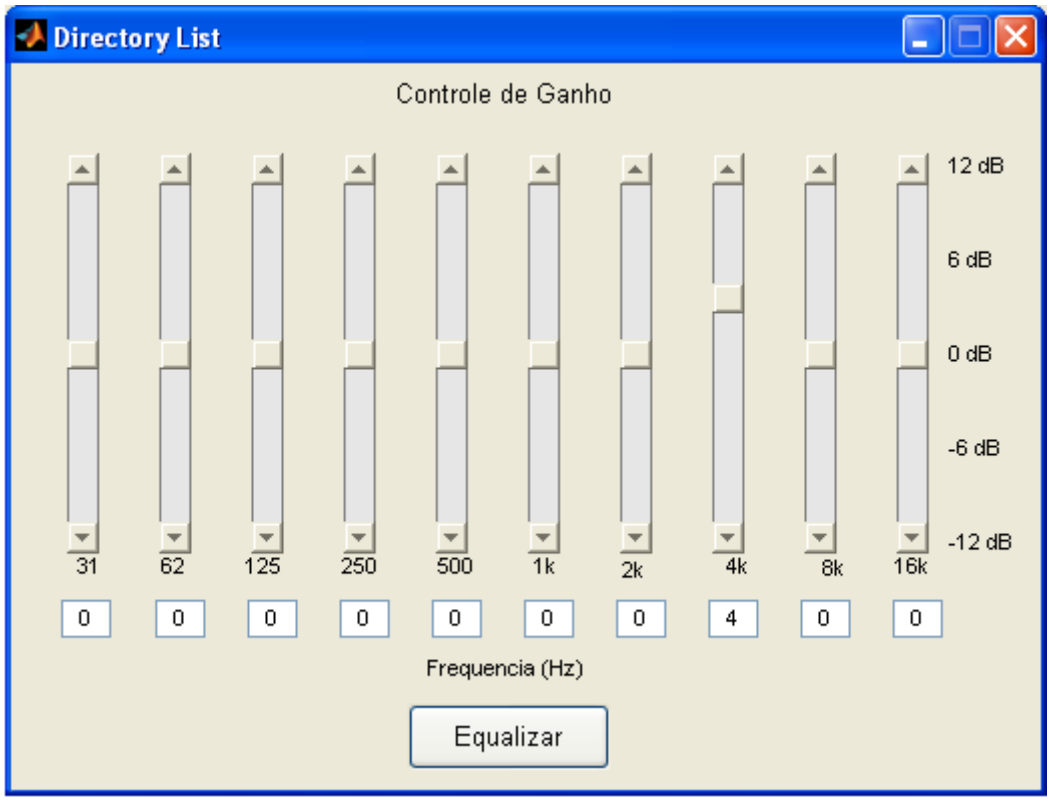

Figura 39: Interface gráfica para controle de ganho dos filtros.

Fonte: Elaboração própria.

Para o programa efetuar a equalização, deve-se criar qualquer sinal de entrada. Por exemplo, na figura 40 foi gerado apenas uma componente de 4 Khz e aplicado o ganho de 4 dB (observado na figura acima). Ainda na mesma figura, pode-se observar a entrada e a saída do filtro, além da intensidade do sinal de saída com um ganho de 4 dB.

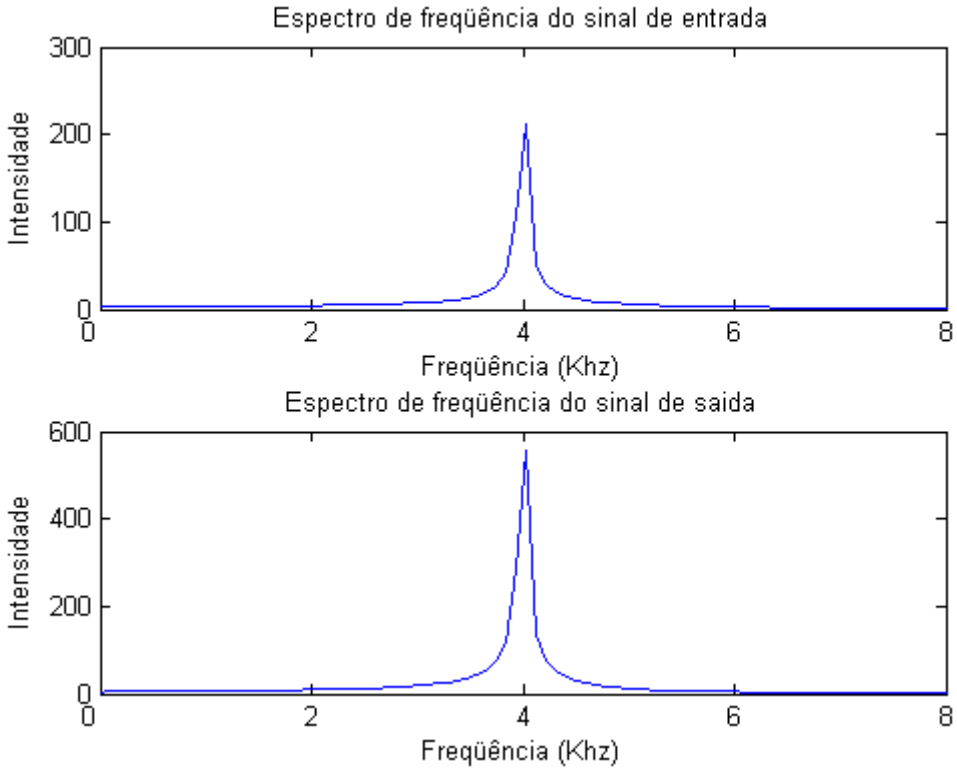

Figura 40: Sinal de entrada de 4 Khz e saída amplificada em 4 dB.

Fonte: Elaboração própria.

No segundo exemplo representado pela figura 41, foi gerado um sinal com cinco componentes: 1 Khz, 2 Khz, 4 Khz, 8 Khz e 16 Khz. O programa foi configurado da seguinte maneira:

Tabela 4: Tabela de ganho para o exemplo da figura 41.

Freqüência (Hz)	Ganho (dB)
31	0
62	0
125	0
250	0
500	0
1000	4.8
2000	-2.4
4000	8.8
8000	-9.6
16000	7.2

No primeiro gráfico da figura 41, é mostrado o espectro de freqüência do sinal de entrada, que corresponde aos citados acima. No segundo gráfico da figura, obtem-se o espectro de freqüência do sinal de saída. Se comparado este último gráfico com a tabela de ganhos acima, conclui-se o correto funcionamento dos filtros.

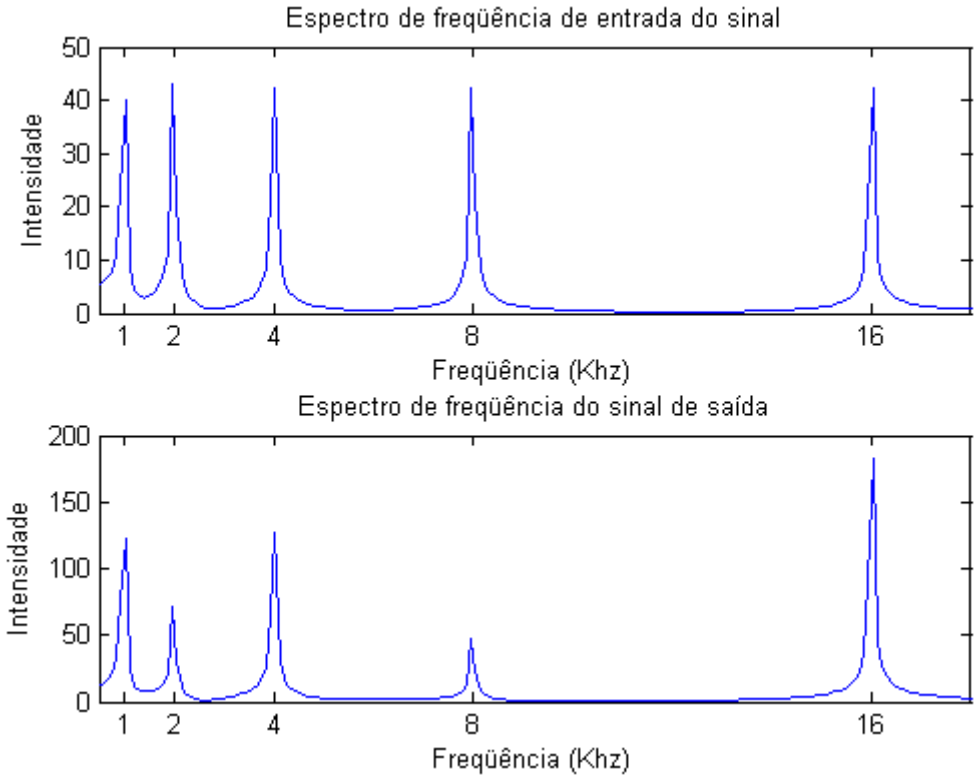

Figura 41: Espectro de freqüência do sinal de entrada e saída.

Fonte: Elaboração própria.

Neste outro exemplo da figura 42 abaixo, é gerado um sinal de entrada com duas componentes: uma de 2 Khz e outra 4 Khz. Os ganhos de todas as freqüências centrais foram configurados no programa em 0 dB.

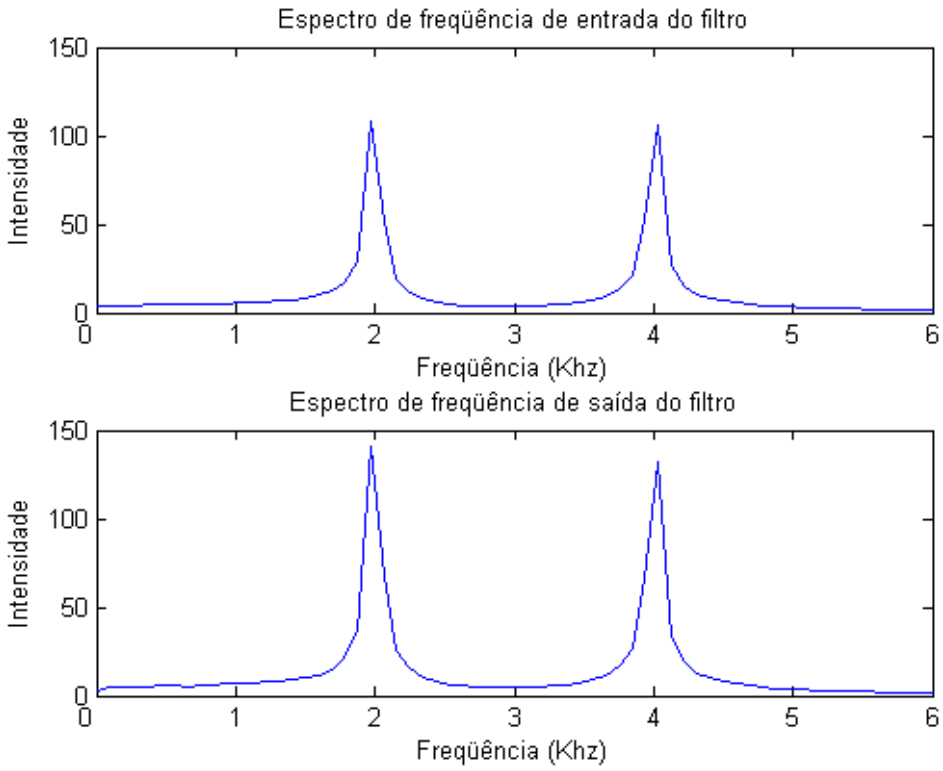

Figura 42: Espectro de freqüência do sinal de entrada e saída.

Fonte: Elaboração própria.

No gráfico que apresenta o espectro de freqüência de saída do filtro, era esperado um sinal com intensidade igual ao de sinal de entrada. O motivo do sinal de saída ter intensidade maior é mostrado na figura 43. Como pode ser observado, existe uma região na curva de resposta dos filtros que é comum entre eles, representada no gráfico abaixo por F3. Portanto, a saída do filtro será a soma das componentes (F1 + F3) para filtro de 2 Khz e (F2 + F3) para o filtro de 4 Khz, resultando na saída a soma das componentes F1 + F2 + 2F3.

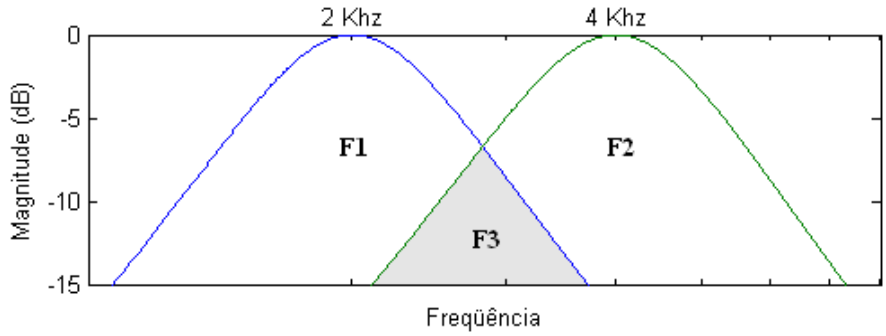

Figura 43: Problema relativo a sobreposição de freqüências.

Fonte: Elaboração própria.

4.2 RESULTADOS OBTIDOS COM O SIMULADOR

O procedimento adotado para realizar os testes no simulador DSP é idêntico aos realizados no Matlab. Inicialmente cada filtro deve ser testado individualmente, para que depois de validados, sejam integrados. A figura 44 mostra o resultado de várias componentes passando pelo filtro de 4 KHz.

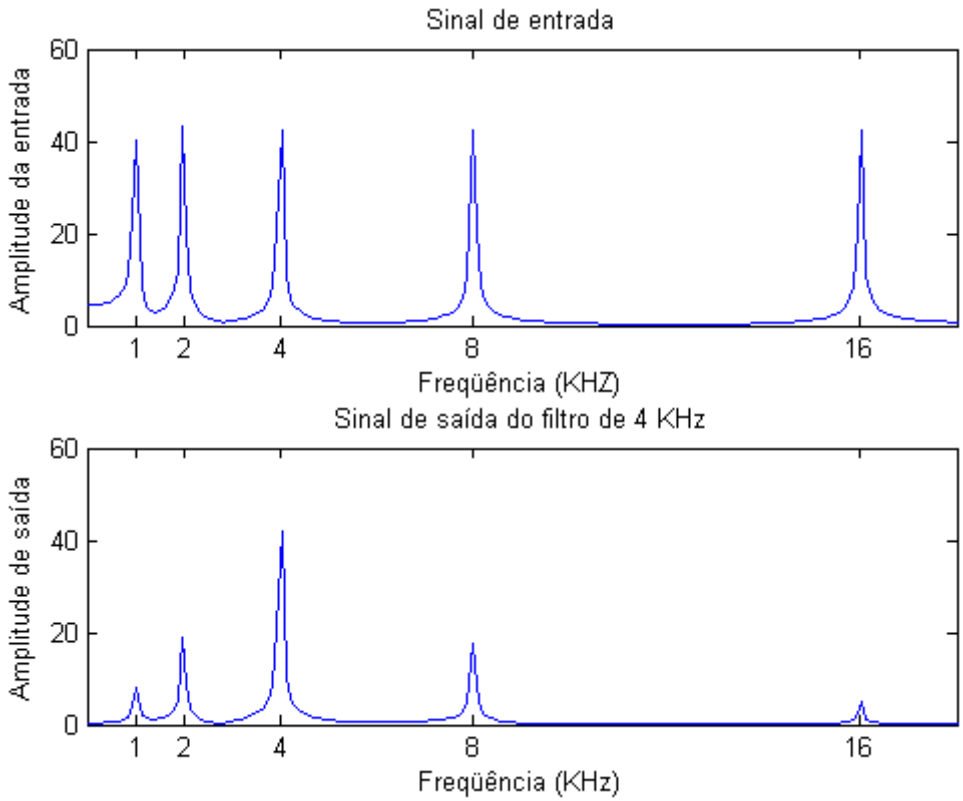

Figura 44: Espectro de freqüência do sinal de entrada e saída no simulador.

Fonte: Elaboração própria.

Foram realizados testes para todas as freqüências centrais. Com exceção dos filtros de 31 Hz e 62 Hz, todos os outros foram validados satisfatoriamente com o simulador do DSP.

4.2.1 Erro de quantização para os filtros de 31 Hz e 62 Hz

A figura 45A mostra um sinal de entrada qualquer, passando pelo filtro de 62 Hz do simulador (figura 45B) e pelo filtro de 62 Hz do Matlab (figura 45C). A saída gerada pelo DSP não corresponde à saída gerada pelo Matlab. Após exaustivos testes, foi concluído que o erro resultante é devido à quantização do DSP, já que o mesmo utiliza níveis de quantização de 16 Bits.

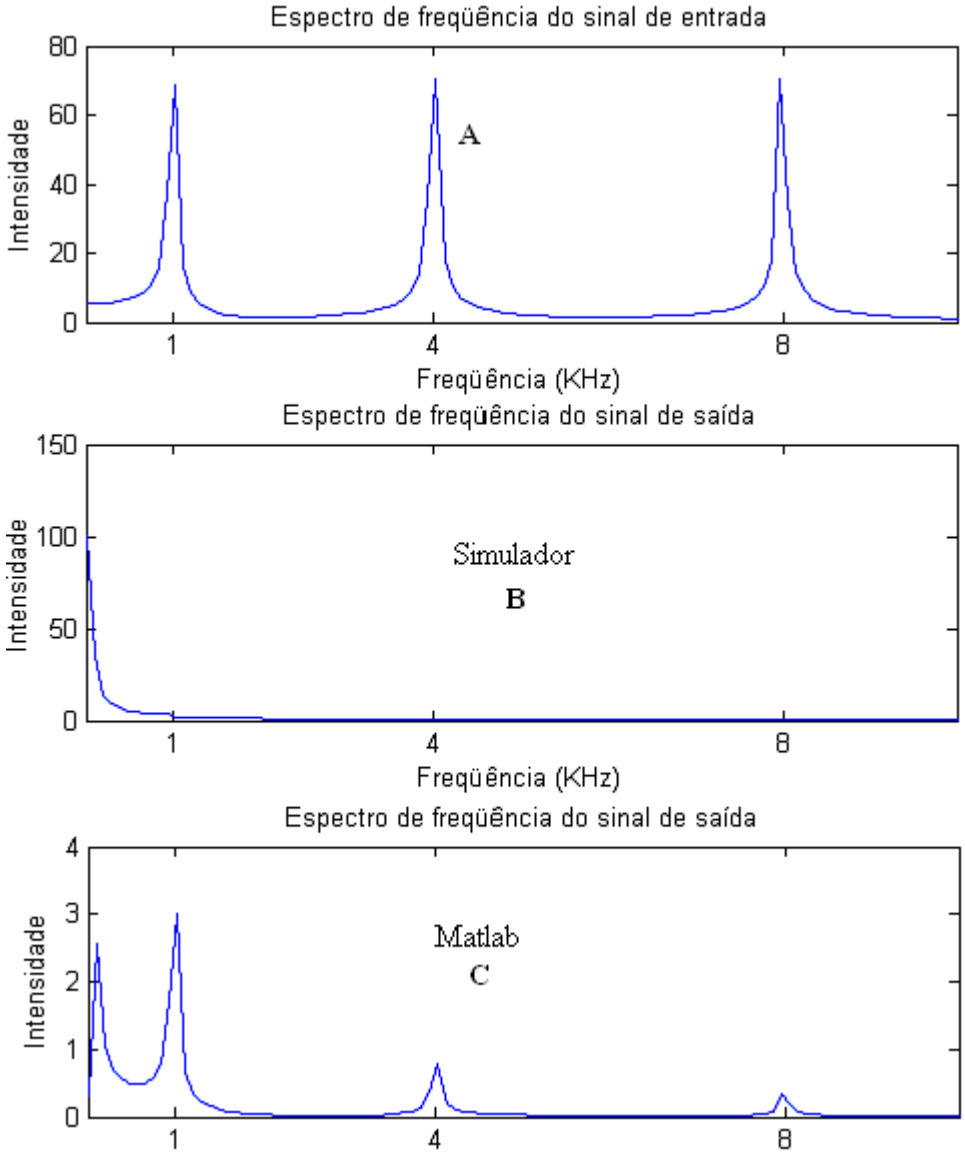

Figura 45: Comparação da saída entre o Matlab e o simulador para o filtro de 62 Hz.

Fonte: Elaboração própria.

Problemas relativos à quantização podem ser observados na tabela 5, que faz um comparativo entre os valores de saída das cinco primeiras iterações entre o Matlab e o simulador.

Tabela 5: Comparativo dos valores de saídas do Matlab e do simulador.

Iteração	Matlab	Simulador
1	0.000000000000000	0.000000
2	0.000607432960255	0.000580
3	0.002294961559587	0.002167
4	0.004697194331718	0.004425
5	0.005685743930584	0.005043

Na segunda iteração, o Matlab apresenta o valor de saída 0.000607432960255. Se esse número for quantizado em 16 Bits, irá resultar no valor 0.000580, que é idêntico à saída do simulador. Isso demonstra a perda de precisão relativa à quantização. Como o coeficiente α dos filtros de 31 Hz e 62 Hz está muito próximo a zero, onde ocorre multiplicações usando esse coeficiente, são gerados valores próximo a zero e devido ao problema de quantização, ocorre perda de precisão. Deve-se ressaltar, que estes valores de saída são utilizados como entrada nas próximas iterações do filtro, ocorrendo propagação do erro até o momento em que o filtro se torne instável.

Analisando a estabilidade, a figura 46 mostra o circulo unitário com os zeros e pólos da função de transferência para o filtro de 62 Hz. Pode-se observar na figura, os dois pólos representados pelo marcador "X". Tais pólos estão muito próximos do limite do círculo unitário, cujo qualquer erro de quantização poderá resultar na saída do círculo e entrada na região de instabilidade, pois segundo (OGATA, 1998) a condição de estabilidade de um sistema discreto é que a magnitude dos pólos seja menor que um.

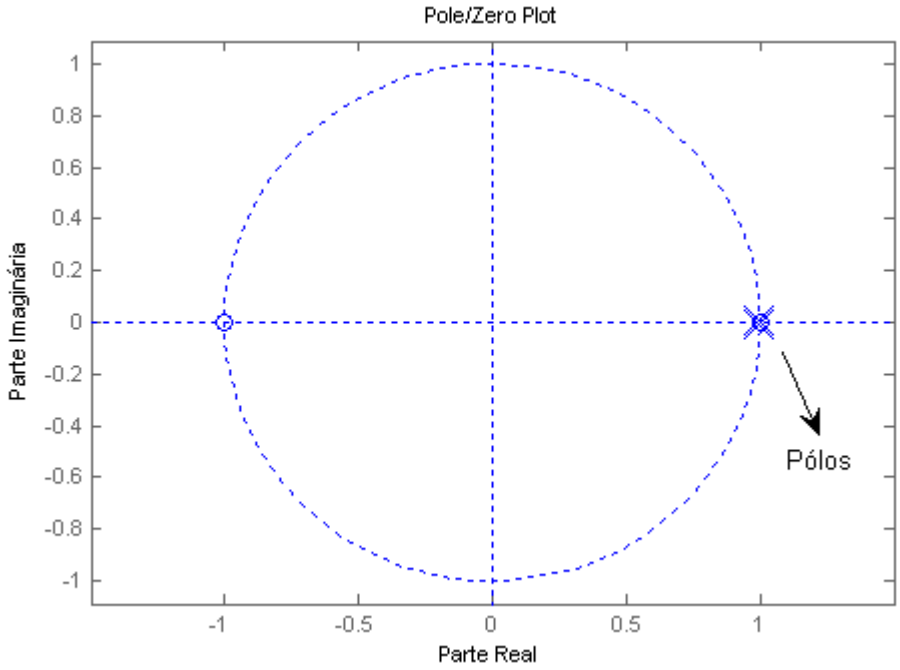

Figura 46: Pólos e zeros do filtro de 62 Hz
Fonte: Elaboração própria.

Já para os filtros de maior freqüência, os pólos estão situados na região interna do círculo unitário garantindo sua estabilidade. Como exemplo observa-se na figura 47 a disposição dos pólos e zeros do filtro de 8 Khz.

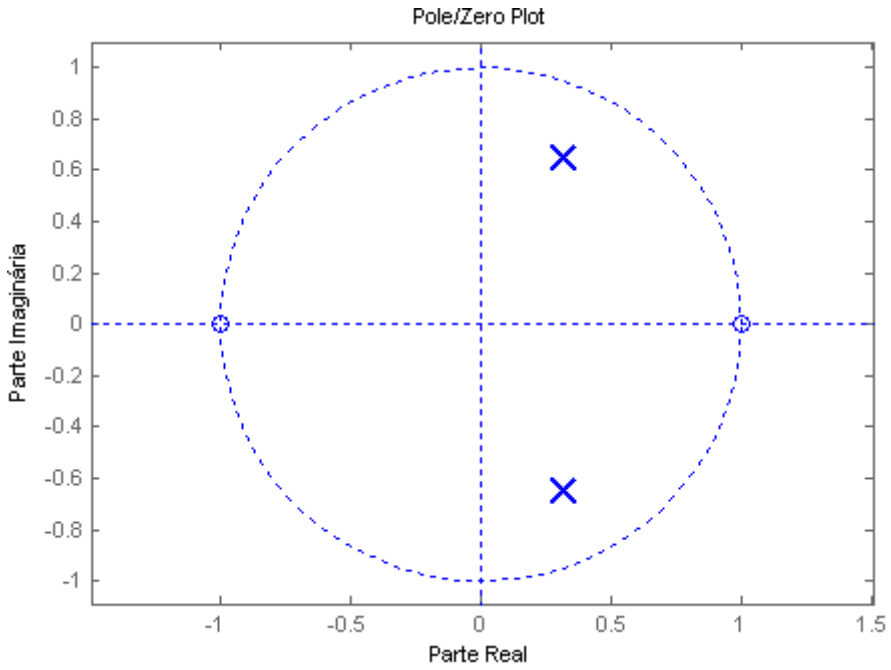

Figura 47: Pólos e zeros do filtro de 8 KHz

Fonte: Elaboração própria.

Para suprir a falta dos dois filtros, foram projetados no Matlab outros dois semelhantes. Entretanto, devido à ordem do filtro ser elevada (pela característica de baixa freqüência e banda passante), foi decidido retirar a implementação destes do projeto.

4.3 CONCLUSÃO

Todos os resultados obtidos na simulação com o Matlab são consistentes. Logo no simulador, as freqüências de 32 e 62 Hz apresentaram problema de instabilidade devido a erros de quantização. Após a validação dos oito filtros, estes foram integrados e realizados diversos testes. Os resultados obtidos são consistentes com os obtidos do Matlab. A implementação dos 8 filtros será detalhada no capítulo a seguir.

5 RESULTADOS EXPERIMENTAIS

Neste capítulo serão apresentadas as características do kit de desenvolvimento utilizado, a comunicação serial e a implementação dos 8 filtros digitais no kit de desenvolvimento.

5.1 KIT DE DESENVOLVIMENTO

O kit de desenvolvimento BF537 STAMP Board é uma plataforma de desenvolvimento de baixo custo com o processador ADSP-BF537. A placa STAMP é parte da comunidade *open source* (código aberto) de desenvolvimento Blackfin/uClinux. Todos os fontes e esquemáticos são encontrados no site da comunidade http://blackfin.uclinux.org. O kit também é disponível comercialmente e pode ser encontrado na Digikey.com.

A STAMP é designada para ser usada em conjunto com a *Toolchain* da GNU para testar as capacidades do processador. A Toolchain da GNU torna possível o desenvolvimento de aplicações avançadas e debug como: (ANALOG DEVICES, 2007)

- Criar, compilar, montar e linkar aplicações escritos em C++, C, e assembler do BF537.
- Carregar, rodar, passo-a-passo, halt e setar breakpoints em programas.
- Ler e escrever dados e programa de memória.
- Ler e escrever nos registradores de periféricos e core.

Acesso ao processador usando o PC através:

- Ethernet
- Serial
- JTAG

Overview das características do kit STAMP:

- ADSP-BF537 Blackfin device com interface JTAG
- 500MHz core clock
- 133MHz system clock
- 32M x 16bit external SDRAM (64MBytes)
- 2M x 16bit external flash (4MBytes)
- 10/100 Mbps Ethernet Interface (via on chip MAC, conectado via DMA)

- CAN TJA1041 transceiver com dois conectadores modulares
- RS-232 UART interface com conector serial DB9
- JTAG ICE 14-pin header
- Seis LEDs e quatro push-buttons de uso geral

5.2 A FAMÍLIA BLACKFIN

A família de DSP's Blackfin da Analog Devices são processadores embarcados de 16-32-bits, desenvolvidos especialmente para computação de alta performance e restrições de consumo (0.8V), como áudio, vídeo e aplicações de comunicação. Baseado na arquitetura Micro Signal (MSA) desenvolvido juntamente com a Intel Corporation, o Blackfin combina um processador RISC com duas unidades de multiplicação e acumulação (MAC) de 16 bits. Essa combinação de atributos qualifica a família Blackfin a desempenhar funções tanto de processamento de sinais quanto de controle e em muitos casos não requer o uso de outro processador auxiliar. Esta capacidade simplifica o custo de implementações de software e hardware. (ANALOG DEVICES, 2007)

Atualmente, os processadores Blackfin podem suportar velocidades de até 756 MHz em um único core. Novos processadores multi-simétricos podem dobrar a performance na mesma freqüência (dual-core). A combinação de alto desempenho e baixo consumo é essencial para as atuais necessidades para processamento de sinais incluindo aplicações *broadband wireless*, áudio/vídeo e dispositivos móveis.

Toda a família Blackfin fornece diversos benefícios para o desenvolvedor de sistemas que incluem:

- Processamento de sinal de alta performance e eficiênte controle de processamento, favorecem o desenvolvimento de uma gama de novos produtos e aplicativos;

- Gerenciamento dinâmico de consumo (DPM) facilita o desenvolvedor especificar seus requisitos e produto final.

- Fácil de misturar instruções de 16 e 32 bits e ferramenta de desenvolvimento que permite um tempo de desenvolvimento reduzido.

a) Core de alta performance

A arquitetura Blackfin suporta pipeline de 10 estágios com mistura do set de instruções de 16 e 32 bits desenvolvido para otimizar a densidade de código. A arquitetura é também totalmente SIMD e inclui instruções para aceleração de vídeo e processamento de imagem.

b) Instruções de vídeo

Suporte especial para dados em 8-bit, tamanho normalmente usado em muitos algoritmos de processamento de píxel. Inclui em sua arquitetura instruções especificas que otimizam o processamento de vídeo.

c) DMA de alta capacidade

Todos os processadores Blackfin possuem múltiplo e independente controlador de DMA que suporta transferência automática de dados com o mínimo de *overhead* do core do processador. A transferência DMS pode ocorrer entre a memória interna e qualquer outro periférico. Transferências também pode ocorrer entre periféricos e dispositivos externos conectados a interface de memória externa, incluindo o controlador da SDRAM e o controlador da memória assíncrona.

d) Eficiente Controle de Processos

A arquitetura conta com uma variedade de benefícios normalmente encontrados em processadores de controle de processo RISC. Estas características incluem uma poderosa e flexível arquitetura de memória hierárquica, superior densidade de código, e uma variedade de periféricos como 10/100 Ethernet MAC, UARTS, SPI, controlador CAN, Timers com suporte a PWM, Watchdog, Real-Time Clock, e um controlador de memória síncrona e assíncrona.

e) Memória Hierárquica

A família Blackfin fornece blocos de memória Level 1 (L1) e Level 2 (L2). A memória L1 é ligada diretamente ao core do processador, roda no clock máximo do sistema e oferece o máximo de performance para algoritimos de tarefas críticas. A memória L2 é maior mas oferece uma velocidade reduzida em comparação a L1, porém mais rápida que uma memória externa.

A estrutura da memória L1 foi implementada para fornecer a performance necessária para processamento de sinal enquanto oferece programação fácil encontrada nos processadores de uso geral. Isto é alcançado permitindo a memória L1 ser configurada como SRAM, cache, ou a combinação de ambas. Os desenvolvedores podem alocar dados de tarefas criticas que requerem altas taxas de transferência e baixa latência na SDRAM, enquanto armazenado dados de menor exigência temporal como controle/S.O na memória cache.

f) Serial Ports (SPORTs)

O processador incorpora duas portas seriais dual-channel (SPORT0 e SPORT1) para comunicação serial e comunicação multiprocessador. As portas serais possuem as seguintes características (figura 48):

- **Operação em I^2S**
- **Operação Bidirecional** – Cada SPORT tem 2 pinos diferentes de transmissão e recepção, possibilitando 8 canais de I^2S áudio estéreo.
- **Buffer** - Oito estágios na transmissão e recepção.
- **Clock** – Cada porta de transmissão e recepção pode usar clock externo ou gerar propriamente.
- **Word length** – Cada SPORT suporta dados de 3 até 32 bits.
- **Frames** - Cada porta de transmissão e recepção pode acomapanhar ou não um frame sincronizador.
- **DMA** – DMA com apenas um ciclo de overhead.
- **Interrupçao** – Cada porta de transmissão e recepção pode gerar uma interrupção no término da transferência dos dados
- **Capacidade de multicanal** – Cada SPORT suporta 128 canais.

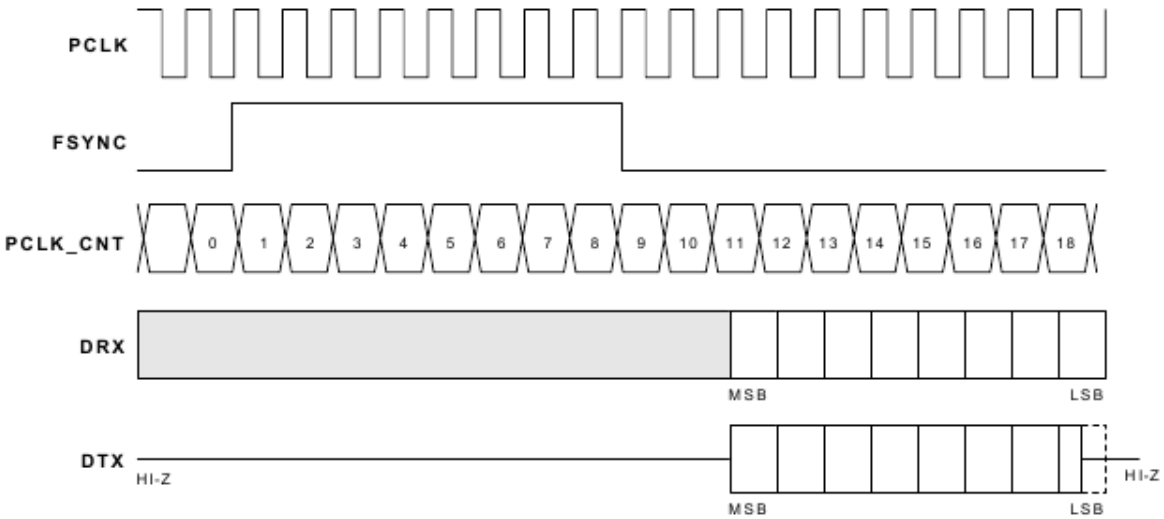

Figura 48: Transmissão serial (SPORT)

Fonte: Analog Devices

g) Serial Peripheral Interface (SPI) Port

O processador tem uma porta compatível com SPI que permite a comunicação com vários dispositivos SPI's. A transmissão SPI usa três pinos para transferência de dados: dois pinos de danos de um de clock.

Um chip select de entrada permite outros dispositivos SPI selecionar o processador, e sete chip select de saída permitem o processador selecionar outros dispositivos. Usando três pinos, a porta SPI fornece comunicação serial síncrona full-duplex, suportando modos master, slave e multi-mestre (figura 49).

Modelo de transmissão SPI

Fonte: Analog Devices

As portas SPI possuem baud rate e clock fase/polaridade configuráveis e possuem um controlador DMA integrado, configurado para suportar tanto transmissão quando recepção de dados (figura 50).

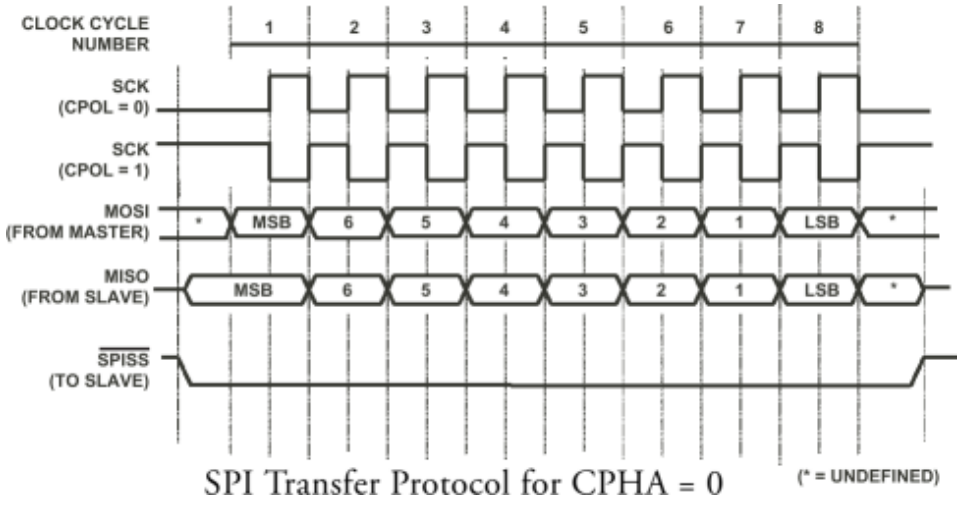

Figura 50: Transmissão SPI.

Fonte: Analog Devices

5.3 AD1836A Codec de Áudio

A placa de áudio usada para aquisição e emissão de áudio é baseada no chipset AD1836A da Analog Devices. A placa fornece três saídas e duas entradas estéreo com suporte para microfone. As saídas e entradas podem ser tanto digitais (S/P-DIF) quanto analógicas. A conexão com o kit STAMP é feita por uma portal serial de alta velocidade (SPORT0 ou SPORT1). Os registradores internos do codec são configurados usando o padrão de comunicação serial SPI. (ANALOG DEVICES, 2007)

O codec AD1836 dispõe de três canais de saída estéreo (total de 6), dois canais de entrada estéreo e entrada e saída de S/PDIF. A interface serial SPORT do processador é linkado com a entrada e saída estéreo dos pinos do codec AD1836. O processador transfere os dados para o codec de áudio utilizando a técnica de multiplexação por divisão de tempo (TDM), pode operar com a freqüência de amostragem de no máximo de 48 kHz com níveis de quantização de 16, 18, 20 e 24 bits.

5.4 O SISTEMA OPERACIONAL uClinux

Atualmente, é inegável a grande participação do sistema operacional Linux em vários segmentos. Inicialmente voltado para servidores, cada vez mais o Linux vem se popularizando como opção efetiva para desktops, quiosques multimídia, PDAs, DVDs, entre

outras. São vários os motivos de tanto sucesso: escalonabilidade, portabilidade, estabilidade, bom desempenho, excelente suporte de rede e, obviamente, por se tratar de um sistema de código aberto. A Revista do Linux número 26 (fevereiro de 2002) publicou um artigo interessante a respeito do mercado de dispositivos embarcados, mostrando a grande evolução do Linux neste segmento.

A popularização do Linux em sistemas embarcados está também ligada ao aparecimento de uma variante do kernel denominada de uClinux (pronuncia-se *you-see-linux*). A grande diferença para o Linux tradicional, é que o uClinux foi desenvolvido visando os processadores/microcontroladores que não possuíam uma Unidade de Gerenciamento de Memória (Memory Management Unit - MMU). Estes processadores são geralmente bem mais baratos e freqüentemente utilizados em sistemas embarcados com recursos, como memória e poder de processamento, mais escassos.

Processadores com MMU, como Pentium e PowerPC, têm embutido no seu microcódigo o suporte ao gerenciamento de memória, que permite que os processos rodem em um espaço de endereçamento virtual, deixando para o sistema operacional o controle e mapeamento entre os endereçamentos virtual e real do sistema. Isto gera um sistema mais seguro, em que um processo não consegue invadir a área de memória do outro e a comunicação entre eles só pode ser feita através de funções do sistema operacional. Já os processos baseados em uClinux compartilham o mesmo espaço real de endereçamento, uma vez que não existe este mapeamento entre endereçamento real e virtual, permitindo a utilização de microprocessadores sem MMU (chamados de NOMMU no kernel), gerando um sistema final com custo reduzido.

O uClinux possui basicamente quase toda funcionalidade do Linux, sendo encontrado nos dois ramos principais do kernel, 2.0 e 2.4. A parte ligada ao gerenciamento de memória foi reescrita, além de várias outras pequenas modificações no kernel. A Interface do Programa de Aplicação (do inglês Application Program Interface - API) fornecida pelo uClinux é praticamente idêntica à do Linux, permitindo que várias aplicações Linux rodem sem problema no uClinux. Foi necessário criar também uma nova biblioteca C (libc). A primeira foi denominada uC-libc, mas devido a sua falta de compatibilidade com outras implementações de bibliotecas C, acabou gerando um trabalho derivado que originou uma nova biblioteca, denominada de uClibc. A uClibc, além de manter a compatibilidade com

outras implementações de bibliotecas C, está também preparada para ser compilada para arquiteturas com ou sem MMU.

A primeira versão do uClinux foi gerada em janeiro de 1998, por Kenneth Albanowski e D. Jeff Dionne, baseada no kernel 2.0 e voltada para o microcontrolador Motorola DragonBall MC68328, usados naquela época, principalmente em PDAs. Atualmente, o uClinux está baseado no kernel 2.4 e disponível para várias plataformas, como Motorola (DragonBall, ColdFire), ARM (vários fabricantes, como NetSilicon, Atmel, Samsung, Intel e Texas), Hitachi (H8), Intel i960, a linha i386 e compatíveis, Axis (ETRAX 100LX) e até mesmo para processadores gerados em chips programáveis (FPGAs), como o NIOS (Altera) e o MicroBlaze (Xilinx). O suporte gráfico também está disponível através do ambiente MicroWindows/Nano-X, com uma API muito semelhante ao X-Windows.

O tamanho do kernel uClinux 2.4 compilado (footprint), é possível ter um sistema completo, com rede, interpretador de comandos (shell) e MicroWindows para o Palm com 548KB, sendo 453KB de código e 95KB de dados. Além disso, é geralmente necessário gerar um sistema de arquivos que fica disponível para o sistema depois de montado, ocupando cerca de 286KB. No total, um sistema uClinux completo com 834KB.

A STAMP board utilizada nesse projeto utiliza uma versão do uClinux que é mantida pela Analog Devices.

5.5 IMPLEMENTAÇÃO DO PROGRAMA EQUALIZADOR

A implementação do equalizador acontece em 13 estágios, conforme pode ser visto na figura 51.

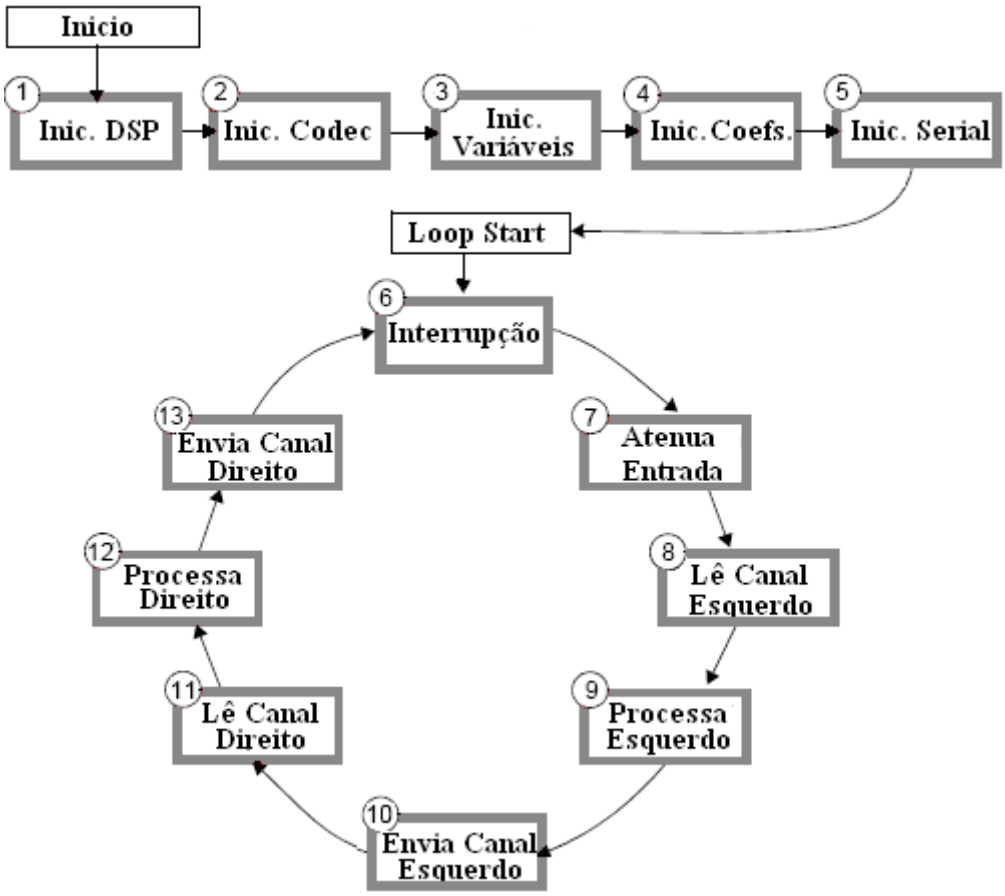

Figura 51: Estágios do programa equalizador

Fonte: Elaboração própria (Adaptado de MONTGOMERY, 2001),

Os cinco primeiros estágios inicializam o hardware e o software (driver) do DSP.

- Estágio 1: Inicializa as constantes de tempo e interrupção do DSP;
- Estágio 2: Inicializa o *codec* com freqüência de amostragem igual a 48 kHz e 16 bits;
- Estágio 3: Inicializa as variáveis do programa;
- Estágio 4: Carrega os coeficientes dos filtros numa variável na memória;
- Estágio 5: Inicializa a comunicação serial com a interface.

Os próximos oito estágios fazem parte de um laço infinito para receber, processar e enviar o dado pelo *codec*.

- Estágio 6: Acontece uma interrupção a cada 1/48k segundos;
- Estágio 7: O canal direito e esquerdo são lidos do *codec* e atenuados para evitar saturação;
- Estágio 8: Uma amostra do canal esquerdo é passado para o algoritmo equalizador;
- Estágio 9: Processa a amostra do canal esquerdo usando os 8 filtros equalizadores. Para cada filtro é aplicado um ganho na saída, passado pela serial;
- Estágio 10: A amostra do canal esquerdo é enviada para o *codec*;
- Estágio 11: Uma amostra do canal direito é passado para o algoritmo equalizador;
- Estágio 12: Processa a amostra do canal direito usando os 8 filtros equalizadores. Para cada filtro é aplicado um ganho na saída, passado pela serial;
- Estágio 13: A amostra do canal direito é enviada para o *codec*.

5.6 INTERFACE GRÁFICA (GUI)

Como pode ser visto na figura 52, foi desenvolvida uma interface gráfica para modificar o ganho de cada filtro, através da interface serial. Os ganhos definidos são: -12 dB, -6 dB, -3 dB, 0 dB, 3 dB, 6 dB, 9 dB e 12 dB. Como definido, os filtros de 31 e de 62 Hz foram retirados.

Figura 52: Interface gráfica do equalizador.

Fonte: Elaboração própria, 2006.

5.7 RESULTADOS OBTIDOS EM TEMPO REAL

Os testes em tempo real foram realizados com o auxílio do software de gravação chamado Sound Forge. Os testes se sucederam da seguinte maneira: A entrada de áudio do DSP foi conectada a saída de áudio do computador e a saída do DSP foi conectada na entrada de áudio do computador. Desta maneira, o som que era reproduzido no computador, depois de processado pelo DSP, retornava e era gravado pelo Sound Forge.

Como fonte de áudio, foram utilizados freqüências pré-estabelecidas e arquivos sonoros (músicas). O primeiro gráfico da figura 53, mostra um sinal de 4 Khz enviado para o DSP, no qual todos os ganhos dos filtros estão em 0 dB. O segundo gráfico, mostra o mesmo sinal, porém agora, com o filtro de 4 KHz com atenuação de –9 dB (e o restante dos filtros sem 0 dB).

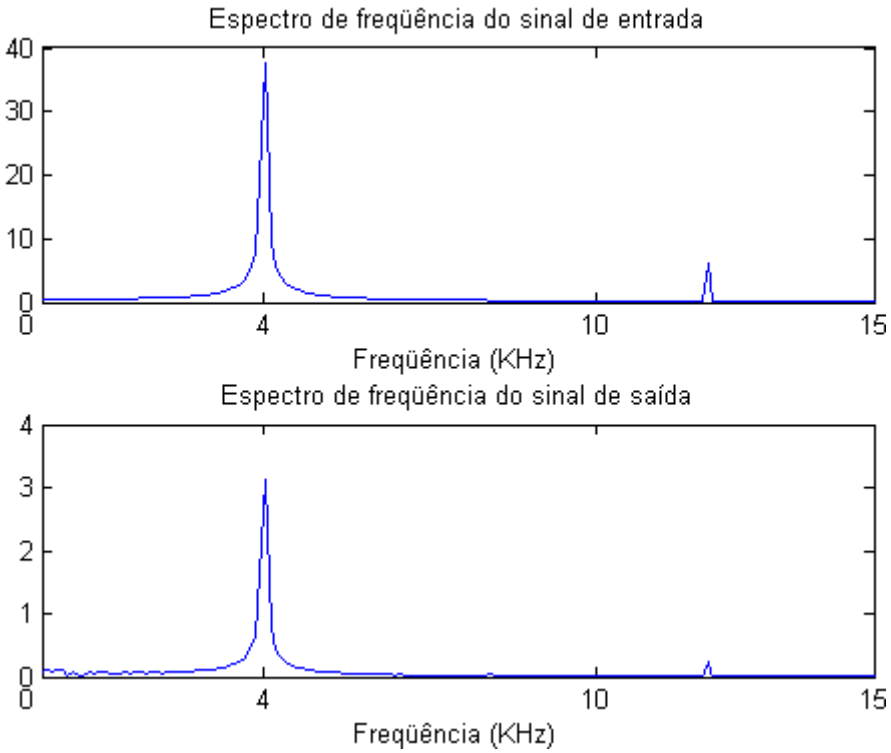

Figura 53: Resultados com o Kit.

Fonte: Elaboração própria, 2006.

O próximo teste foi realizado com um trecho de voz de uma pessoa. O gráfico da figura 54 mostra o trecho de voz sem equalização. Pode-se observar um ruído de alta freqüência e de baixa intensidade proveniente da própria placa de som durante a aquisição do sinal.

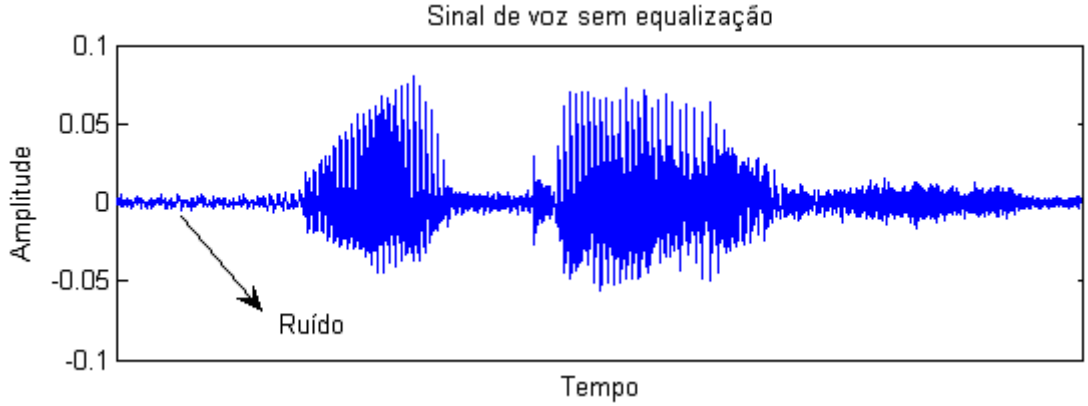

Figura 54: Sinal de voz original.

Fonte: Elaboração própria, 2006.

No gráfico da figura 55 os ganhos foram configurados de maneira que atenuasse as freqüências mais graves e amplificassem as freqüências mais agudas da voz.

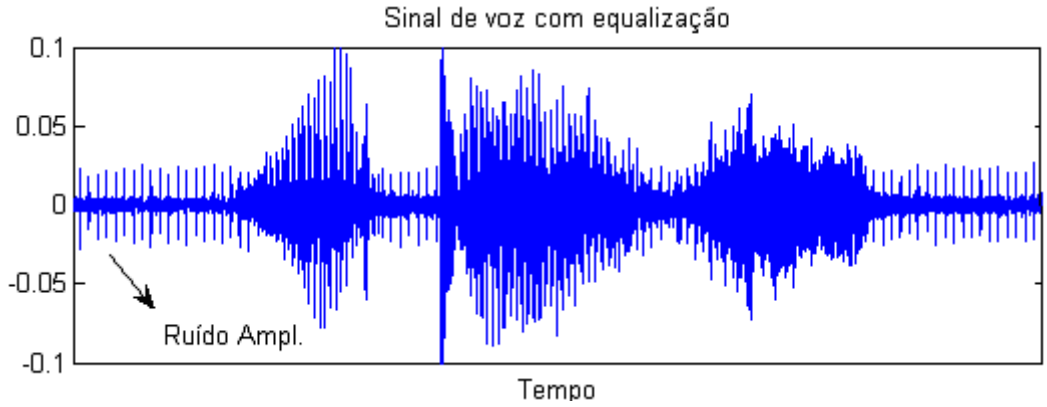

Figura 55: Sinal de voz com freqüências altas amplificadas e baixas atenuadas.

Fonte: Elaboração própria, 2006.

No gráfico da figura 56 os ganhos foram configurados de maneira que amplificasse as freqüências mais graves e atenuassem as freqüências mais agudas da voz. Como esperado, a intensidade do ruído de alta freqüência diminuiu.

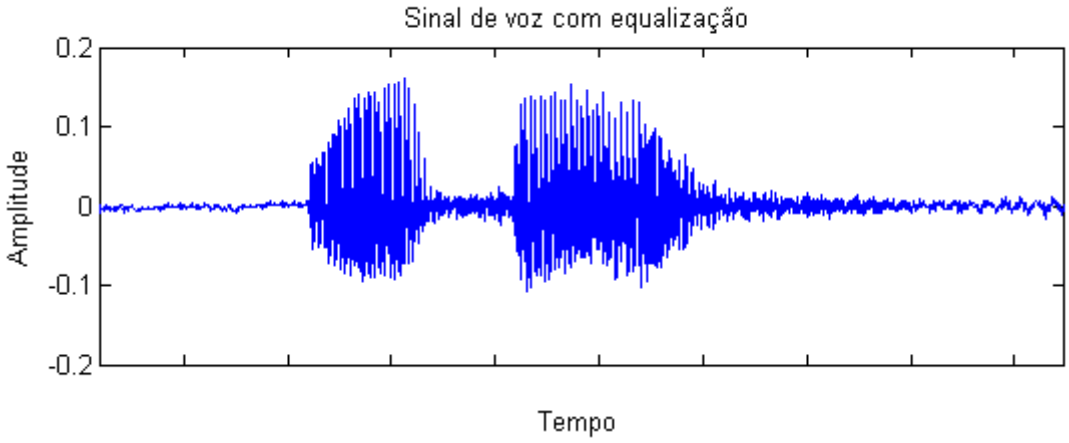

Figura 56: Sinal de voz com freqüências altas atenuadas e baixas amplificadas.

Fonte: Elaboração própria, 2006.

6 CONCLUSÃO

O presente trabalho fez uma síntese do desenvolvimento de um equalizador digital, composto de 10 filtros do tipo IIR implementados em um processador digital de sinal (DSP). Baseado nesta síntese foi proposto um protótipo apto a realizar equalizações em tempo real a partir de uma fonte sonora. Os trabalhos realizados e os resultados obtidos permitiram se chegar à conclusão básica de que a proposta apresentada demonstra ser correta, embora a observação de alguns detalhes se faça necessária.

A escolha do processador DSP se mostrou adequado aos requisitos de velocidade para implementação dos filtros digitais IIR. Entretanto, a escolha do DSP não atendeu devidamente às necessidades de precisão para os filtros de freqüência mais baixa (31 e 62 Hz), devido a erros de quantização. Para contornar esse problema, deve-se aumentar a precisão do codec para no mínimo 24 bits.

O protocolo de comunicação serial adotado também mostrou-se eficiente, uma vez que fornece a funcionalidade e o desempenho suficientes para o controle dos parâmetros de ganho em tempo real. O número reduzido de comandos contribuiu também para tornar a comunicação entre o módulo de recepção e de interface uma tarefa relativamente simples.

A construção do filtro IIR digital obtido a partir de um filtro RLC-passivo analógico mostrou-se adequado ao projeto de um equalizador digital. A estrutura proposta possibilita a adição de um número maior de bandas sem nenhuma modificação na síntese do filtro e com poucas modificações no algoritmo equalizador.

O uso ferramenta Matlab para testes e simulações pré-projeto foi essencial no desenvolvimento do equalizador digital. A familiarização com o Matlab levou a uma mudança do escopo do trabalho. A interface gráfica no qual seria desenvolvida com Delphi, por motivos de praticidade, principalmente com a comunicação serial, foi desenvolvida no próprio Matlab.

Com alguns dos conhecimentos obtidos na realização deste projeto, uma série de sugestões para trabalhos futuros pode ser enumerada. Não somente envolvendo equalizadores, mas

outros abrangendo o processamento digital de áudio tais como: processadores de efeitos, compressores, limitadores, equalizadores paramétricos e etc.

O trabalho de monografia oportunizou a extensão dos conhecimentos teóricos e práticos adquiridos nas várias áreas do curso de especialização. A motivação pela pesquisa impulsionou a busca de novos conhecimentos e aprimoramentos importantes para o futuro profissional. A oportunidade de desenvolver o trabalho com uso de um processador digital de sinal (DSP), que representa o segmento que mais cresce no mercado de semicondutores (TEXAS INSTRUMENTS, 2005), contribuiu para o conhecimento técnico abrindo portas para futuras oportunidades acadêmicas e para o mercado de trabalho.

7 REFERÊNCIAS

1) FERREIRA, Adriano. **Medidor de energia residencial,uma abordagem usando DSP**. 2003. 105p. Trabalho de conclusão de curso (Graduação em Engenharia da Computação) – Universidade do Vale do Itajaí, São José, 2003.

2) ANALOG DEVICES**. DSP architecture.** 2002. Disponível em: <http://www.analog.com>. Acesso em: 5 jan. 2008.

3) ALMEIDA, Ailson R. **Conversores digital-analógico (DAC) e analógico-digital (ADC)**. 2005. Disponível em: <http://www2.ele.ufes.br/~ailson/digital2/SDprograma2003_1.html>. Acesso em: 11 jan. 2008.

4) ARBORETUM. **Hyperprism filter processes**. 2005. Disponível em: <http://www.sfu.ca/sca/Manuals/Hyperprism2.5/hppc_proc_filters.html>. Acesso em: 31 out. 2007.

5) BARBOSA, Álvaro M. **Edição digital de som:** Uma abordagem aos fundamentos da escultura sonora orientada para criadores**.** Escola das artes som e imagem - Universidade Católica Portuguesa. 1999. Disponível em: <www.abarbosa.org/docs/edicao_digital_som.pdf> Acesso em: 03 out. 2007.

6) BEHRINGHER. **Equalisadores paramétricos**. 2005. Disponível em: <http://www.behringher.com.br>. Acesso em: 15 mar. 2008.

7) CANZIAN, Edmur. **CNZ Engenharia e Informática Ltda. Minicurso de Comunicação Serial - RS232.** Disponível em: <http://www.cnz.com.br>. Acesso em: 12 dez 2007.

8) CICLOTRON. **Equalisadores gráficos**. 2005. Disponível em: <http://www.ciclotron.com.br>. Acesso em: 10 nov. 2007.

9) DUQUE, A. D. **O processador digital de sinais família Tms320f2xx.** 2004. Disponível em: <http://www.mestradoeletrica.ufjf.br/professores/duque/microp_cap1.pdf>. Acesso em: 27 out. 2007.

10) EYRE, J. e BIER J. **The evolution of DSP processors**. Disponível em: <www.bdti.com/articles/evolution.pdf>. Acesso em: 05 set. 2007.

11) GONÇALVES, L. V. Conceitos básicos sobre comunicação. 2005. Disponível em: <http://pessoal.cefetpr.br/vagnerg/siscom/basico_com.pdf>. Acesso em: 01 out. 2007.

12) HAYS, W. P. **DSPs**: **Back to the future** - What is a DSP?. 2005. Disponível em: <http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=127&page=2> Acesso em: 25 jan. 2008.

13) IAZZETTA, F. **Filtros.** 2005. Disponível em: <http://www.eca.usp.br/prof/iazzetta/tutor/audio/filtros/filtros.html> Acesso em: 11 jan. 2008

14) IZECKSOHN, S. **Os equalizadores**. 1998. Disponível em: <http://www.homestudio.com.br/Artigos/Art016.htm>. Acesso em: 31 fev. 2008.

15) KESTER, W. **Mixed-signal and DSP design techniques, digital filters**. 2001. Disponível em: <http://www.analog.com>Acesso em: 13 mar. 2008.

16) MENDONÇA, A; ZELENOVSKY, R. **Eletrônica digital.** 1. ed. Rio de Janeiro: MZ, 2004.

17) MERCER, C. **Audio equalisation filter & parametric filtering**. 2005. Disponível em: <http://www.prosig.com/signalprocessing/AudioEqualisationFilter.pdf>. Acesso em: 31 mar. 2008.

18) MONTGOMERY, James M. **Implementing a 10-Band Stereo Equalizer on the DSP56311 EVM Board.** 2001. Disponível em: <http>//www.mot.com/SPS/DSP/documentation>. Acesso em: 20/03/2008.

19) MORELLATO, V. **Microprocessadores**. 2005. Disponível em: <http://www.stoinacio.com.br/~morellato/>. Acesso em: 15 out. 2007.

20) NEIVA, Álvaro C. A. **Filtros e equalizadores**. 2002. Disponível em: <http://geocities.yahoo.com.br/alvaroneiva/filteq2.htm>. Acesso em: 22 mar. 2008.

21) OGATA, Katsuhiko. **Engenharia de controle moderno.** 3 ed. Rio de Janeiro: Prentice Hall do Brasil, c1998. 813p.

22) OLIVEIRA, C. de Oliveira. **Sistemas multimídia**. 2003. Disponível em: <www.lncc.br/~jauvane/SMM/2002/SMM-A02.pdf>. Acesso em: 09 fev.2008.

23) PAYAN, R. **Parametric equalization on TMS320C6000 DSP** – Catalog DSP (Texas Instruments). 2002. Disponível em: <http://www.ti.com>. Acesso em: 31 out. 2007.

24) PEIL, Norberto de Castro. **Reconhecimento de voz**: uma abordagem utilizando lógica difusa. Ciência de Computação, Tese de Mestrado – UFSC. Florianópolis, 1998. 55p.

25) PHOTONICS SYSTENS. **FIR and IIR digital filers**: department of electrical and computer engineering of McGill University. 2005. Disponível em: <www.photonics.ece.mcgill.ca/DigitalFilters.pdf> Acesso em: 13 set. 2007.

26) RATTON, M. **Fundamentos de áudio.**1. ed. Rio de Janeiro: Music Center, 2005.

27) RIBEIRO, M. V. **Técnicas de reconstrução de pacotes aplicadas à codificadores de forma de onda para VoIP** – Implementação em tempo real. 2005. Disponível em: <http://ti.com/sc/brasil/noticias/2004/pack_reconst_part_ii.pdf>. Acesso em: 26 jan. 2008.

28) ROBIN, Iain A. **Digital filters:** An introduction. 2002. Disponível em: <http://www.techonline.com/community/tech_group/dsp/20365>. Acesso em: 09 jan. 2008.

29) ROCHA, A. F. **Filtros IIR**. 2005. Disponível em: <http://www.ene.unb.br/~adson/FiltrosIIR2.doc> Acesso em: 05 mar. 2008.

30) SCANDELARI, L. **Filtros digitais.** 2005. Disponível em: <www.pessoal.cefetpr.br/luciano/filtros.pdf> Acesso em: 10 mar. 2008.

31) SCHWANKE, D. **Exame de potênciais evocados auditivos utilizando processador digital de sinal - DSPEA**. 2000. 105p. Dissertação (Mestrado em Ciência da Computação) - Faculdade de Ciência da Computação - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.

32) SMITH, Steven W. **The scientist and engineer's guide to digital signal processing.** 1999. Disponível em: <http://dspguide.com/filters.htm> Acesso em: 07 agosto 2007.

34) SOUND ON SOUND. **Equalisers explained** - Music Recording Magazine**.** 2001 July. Disponível em: <http://www.soundonsound.com/sos/jul01/articles/equalisers1.asp>. Acesso em: 31/10/2007.

35) SOUZA, A. **Estudo de técnicas para processamento de sinais digitais.** 2001. Disponível em: <http://buscatextual.cnpq.br/buscatextual/visualizacv.jsp?id=K4784943A0>. Acesso em: 31 fev. 2008.

36) TERAHATA, T. **Equilibrando freqüências.** 2003. Disponível em**:** **<**http://territorio.terra.com.br/canais/rockonline**>**. Acesso em: 31 out. 2007.

37) TOMARAKOS, J; LEDGER, D. **Using the low-cost, high performance ADSP-21065l digital signal processor for digital audio applications.** 1998. Disponível em: <http://dspguide.com/audio/daag.pdf>. Acesso em: 15 nov. 2007

38) WILEY, D. **Digital filters:** An Introduction2005. Disponível em:

<http://www.dsptutor.freeuk.com/dfilt1.htm> Acesso em: 10 set. 2007

39) YNOGUTI, C. Alberto. **Processamento digital de sinais***:* Conversão A/D e D/A. 2005. Disponível em: <http://www.inatel.br/docentes/ynoguti/e724/Digitalizacao.pdf>. Acesso em: 13 fev. 2008

.

.

8 ANEXOS

Os códigos fonte dos programas se encontram em anexo no CD.