UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

Edgard Luciano Oliveira da Silva

IMPLEMENTAÇÃO DO SINTETIZADOR DE FORMANTES DE KLATT EM PONTO-FIXO UTILIZANDO O PROCESSADOR TMS320C25

Este exemplar corresponde à redação final da tese defendida por Edgard Luciano Oliveira da Silva e aprovada pela Comissão Julgadora em 04/10/96.

José Gudo Cugnasca
Orientador

Edgard Luciano Oliveira da Silva

IMPLEMENTAÇÃO DO SINTETIZADOR DE FORMANTES DE KLATT EM PONTO-FIXO UTILIZANDO O PROCESSADOR TMS320C25

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA.

Orientador: Prof Dr. José Geraldo Chiquito

Campinas - SP

Outubro 1996

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Si38i

Silva, Edgard Luciano Oliveira da
Implementação do sintetizador de formantes de Klatt em ponto-fixo utilizando o processador TMS320C25 / Edgard Luciano Oliveira da Silva.--Campinas, SP: [s.n.], 1996.

Orientador: José Geraldo Chiquito.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas de processamento da fala. 2. Síntese da voz. 3. Codificador de voz. 4. Processamento de sinais - Técnicas digitais. I. Chiquito, José Geraldo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétricia e de Computação. III. Título.

Agradeço...

... ao Prof. Dr. José Geraldo Chiquito pela valiosa orientação dada a este trabalho e por sugerir este tema;

... ao Eng.º Edson Nagle, CPqD/Telebrás, pelo seu auxílio nos primeiros passos de síntese, pela paciente revisão e sugestões neste trabalho e pelo material cedido sobre o TMS320C25;

... aos amigos do Laboratório de Processamento de Sinais e da UNICAMP pelas sugestões, contribuições e por terem pacientemente ouvido os resultados;

... aos Prof. Dr. Laércio Caldeira, Prof. Paulo César Crepaldi e Prof Luís Edival da Escola Federal de Engenharia de Itajubá - EFEI por terem me recomendado ao mestrado;

... à todos os colegas e amigos que, através de comentários ou de uma simples palavra de incentivo, contribuíram de uma forma ou outra para a realização deste trabalho;

... à minha noiva, Alessandra, pela compreensão, cooperação e incentivo;

... à instituição financeira de pesquisa CAPES pelo apoio financeiro.

À meus pais, a quem devo tudo que sou.

RESUMO

O presente trabalho trata da implementação do software do sintetizador de formantes cascata/paralelo de Klatt em ponto-fixo no TMS320C25. Neste trabalho, os efeitos da aritmética de ponto-fixo, a qual requer uma série de cuidados que por vezes desprezamos na implementação em ponto-flutuante, assim como o sintetizador de formantes de Klatt e o CI-TMS320C25, são estudados em detalhes. Uma base teórica sobre o processo de produção da fala e suas características são apresentados nos capítulos iniciais. Modificações no diagrama de blocos do sintetizador são feitas com o objetivo de se alcançar um melhor desempenho computacional. As características de voz do autor são apresentadas e um exemplo de síntese é realizado. A análise dos resultados obtidos é feita a partir de espectrogramas de banda larga e através da análise LPC.

CONTEÚDO

FIGURAS

TABELAS

EQUAÇÕES

1. Introdução

O objetivo deste trabalho é o desenvolvimento de um software para a implementação prática de um sintetizador de fala utilizando o circuito integrado dedicado para processamento digital de sinais, TMS320C25 da Texas Instruments, a partir do modelo de sintetizador de formantes em cascata/paralelo de Klatt.

Circuitos integrados dedicados para processamento digital de sinais (DSP-ICs) estão cada vez mais presentes em nossas vidas. A família TMS320 surgiu nos anos 80 e se tornou um sucesso como DSP-IC's de uso geral. A possibilidade de se implementar as funções de processamento digital de sinais como filtragem, modulação, codificação dos sinais de fala, FFT e processamento de imagem por meios numéricos já era estudada há muito tempo. Um DSP-IC de uso geral pode realizar todas estas operações. Já o TMS320C25 é um microprocessador digital de sinais da segunda geração da família TMS.

Algumas das aplicações mais importantes das técnicas de processamento digital de sinais têm sido na área de processamento de sinais da fala. De fato, uma grande percentagem do background teórico do processamento digital de sinais tem sido derivado do estudo de sinais da fala pelos pesquisadores.

Um DSP-IC pode ser usado para a implementação de muitos dos algoritmos necessários não apenas para analisar mas também para sintetizar os sinais de fala. Ao analisarmos os sinais de fala, podemos extrair um conjunto de parâmetros digitais que representam o sinal original. Os métodos incluem técnicas matemáticas como a transformada de Fourier, análise LPC, análise homomórfica, e também algoritmos de detecção de pitch. A síntese de sinais da fala consiste na recuperação do sinal de fala original a partir de um conjunto de parâmetros que variam continuamente no tempo.

O maior interesse nos sistemas de síntese de sinais de fala é estimulado pela necessidade de um armazenamento digital econômico de sinais de fala para sistemas computadorizados de resposta de voz. Um sistema computadorizado de resposta vocal é basicamente um serviço automático de informações todo digital, que pode ser questionado por uma pessoa a partir de um terminal e no qual a resposta contendo a informação desejada é um sinal de fala. Uma vez que até um simples telefone pode ser usado como um terminal para tal sistema, a capacidade de tal serviço automático de informação pode ser feito universalmente disponível para qualquer tipo de telefone sem a necessidade de qualquer equipamento especializado.

Sistemas de síntese de fala também desempenham um papel fundamental no desenvolvimento de modelos de produção da fala.

Uma outra aplicação é no auxilio de deficientes. Esta aplicação consiste no processamento de sinais para tornar a informação disponível de uma forma que é normalmente melhor assimilada por um deficiente. Por exemplo, "ler'" um livro para um cego, permitindo que ele "escute" a informação escrita.

A escolha do TMS320C25 como DSP-IC repousa no fato dele ser amplamente utilizado nos dias de hoje. Para termos uma idéia disto, podemos citar o equipamento da KAY Elemetrics, o CSL (*Computer Speech Lab*) Model 4300B, que é um equipamento dedicado a análise de voz e que possui internamente um TMS30C25. Este equipamento foi utilizado pela equipe do Doutor Badan Palhares na análise das gravações da conversa da torre do aeroporto de Guarulhos com o avião prefixo PT-LSD momentos antes do acidente que matou o conjunto Mamonas Assassinas e mais recentemente na análise das gravações da conversa telefônica de Susana Marcolina no caso do assassinato de PC Farias em Maceió - AL. O CSL é também usado na caracterização fonética de sinais da fala no Laboratório de Fonética Acústica e Psicolinguística Experimental do IEL (LAFAPE-IEL) e no CPqD.

O Laboratório de Processamento de Sinais da UNICAMP além de possuir este equipamento da KAY, o qual foi amplamente utilizado durante este trabalho,

também possui um módulo de desenvolvimento de hardware/software do TMS320C25, o XDS (*Extended Development Suport*). Outro fator importante na escolha do TMS320C25 é a disponibilidade no mercado de uma série de ferramentas para desenvolvimento de aplicações.

O modelo escolhido para síntese repousa no fato de sintetizadores de formantes produzirem um sinal de fala a partir de um conjunto de parâmetros formuladas no domínio acústico. Formas de ondas glotais e fontes de ruído são utilizados para estimular dois conjuntos de filtros: um em cascata responsável pelos sons sonoros e outro em paralelo responsável pelos sons surdos. Teoricamente ambas as funções podem ser implementadas tanto pelo conjunto em paralelo quanto pelo conjunto em cascata. Mas na prática é mais fácil de implementarmos na configuração cascata/paralela.

Além disso, o bloco básico de construção de um sintetizador de formantes cascata/paralelo de Klatt é um filtro digital de segunda ordem. As vantagens de um filtro digital são muito significativas sobre os filtros analógicos: Alta confiabilidade, alta precisão, não há o efeito dos componentes (drift de voltagem, drift de temperatura, problemas de ruído) afetando a performance do sistema. Uma outra importante vantagem dos filtros digitais é a mudança fácil dos parâmetros para modificar as características do filtro.

2. Elementos da Fala

2.1 O Processo da Produção dos Sinais de Fala

No estudo do processo de produção da fala, é útil separarmos as características importantes do sistema físico de maneira a termos um modelo realístico. O sinal da fala é apenas uma onda acústica que é irradiada deste sistema. A Figura 2-1 coloca em evidência as características do sistema de produção da fala humana.

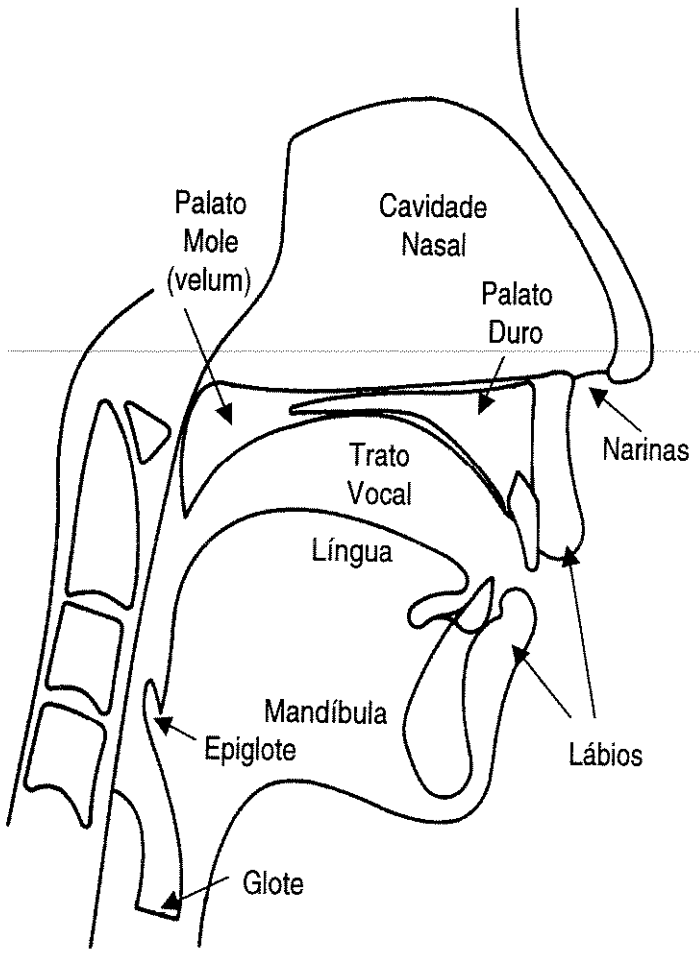


Figura 2-1 Vista Transversal do Mecanismo Vocal

O trato vocal, inicia-se na abertura entre as cordas vocais ou glote e termina nos lábios. O trato vocal assim consiste da faringe (a conexão do esôfago com

a boca) e a boca ou cavidade oral. O trato nasal inicia no velum e termina nas narinas. Quando o velum é abaixado, o trato nasal é acusticamente acoplado ao trato vocal para produzir os sons nasais da fala. Temos outros elementos presentes na produção da fala: língua, lábios, mandíbula e velum. Estes elementos são chamados de articuladores da fala.

Podemos considerar o trato vocal e o trato nasal como sendo um tubo acústico de área transversal não uniforme. A área de seção transversal não uniforme do trato vocal depende fortemente da posição dos articuladores (língua, lábios, mandíbula e velum) e varia de 0 cm^2 (completamente fechado) até 20 cm^2.

O trato vocal humano, que se estende da glote aos lábios, tem em média cerca de 17 cm de comprimento em um adulto masculino e conseqüentemente sua primeira ressonância de quarto de onda ocorre na freqüência de:

$$F_1 = \frac{1}{4}\frac{c}{l} \cong \frac{1}{4}\frac{34.000cm/seg}{17cm} \cong 500Hz$$

sendo l = comprimento do tubo e c a velocidade do som.

2.2 Características da Fala

A Figura 2-2 mostra um diagrama esquemático do mecanismo de produção da fala humana. O sistema sub-glotal é composto pelos pulmões, brônquios e traquéia. O sistema sub-glotal serve como fonte de energia no processo de produção da fala quando o ar é expelido dos pulmões. Na produção da fala, a cavidade peitoral se expande e contrai para forçar o ar para fora dos pulmões, através da traquéia passando pela glote. Ao escapar pela laringe a corrente de ar é modulada pelas cordas vocais. As cordas vocais são na realidade, uma válvula constituída por membranas, músculos e ligamentos. Para produzir voz, basta um movimento muscular voluntário. Se as cordas vocais são tensionadas, como para sons sonoros, como por exemplo as vogais, elas irão vibrar em um modo de oscilação amortecida, modulando o ar através de sopros discretos (pulsos). À freqüência destes pulsos daremos o nome de freqüência fundamental (F0) ou freqüência de pitch. Assim, F0 é a freqüência

fundamental de vibração das cordas vocais; ou seja a freqüência com que estes sopros de ar ocorrem. Se no entanto as cordas vocais são relaxadas, o fluxo de ar passa através da glote e não é afetado. É essa corrente de ar modulada que constitui a fonte sonora. A este fenômeno damos o nome de fonação. As pequenas lufadas (sopros) de ar que escapam pela laringe durante a fonação constituem a mais importante fonte de energia da fala: aquilo que costumamos chamar de voz.

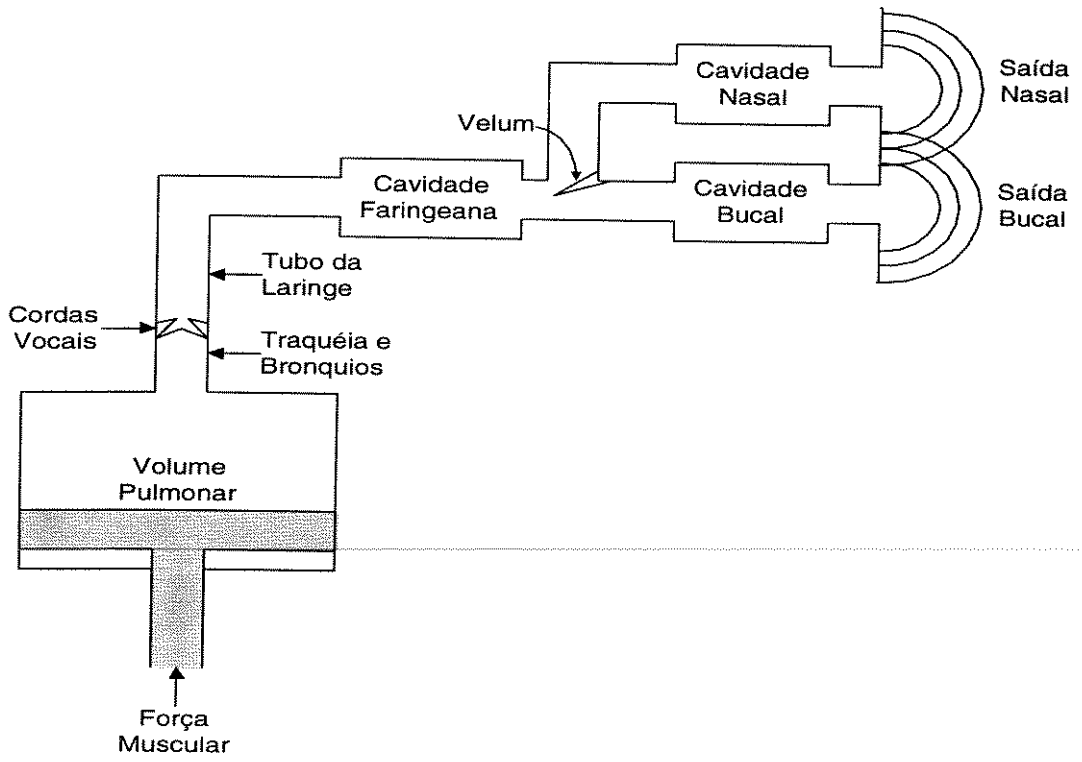


Figura 2-2 Diagrama Esquemático do Mecanismo de Produção da Fala Humana (Após Flanagan)

O som gerado, discutido anteriormente, propaga-se através destes tubos. O espectro de freqüência é modelado pela seletividade do tubo. Este efeito é muito similar ao efeito de ressonância de um órgão de tubos ou instrumento de sopro. O trato vocal tem certos modos normais de ressonância denominados formantes, que dependem fortemente da posição exata dos articuladores. Uma vez que estes articuladores modificam o formato e as dimensões do trato vocal, cada formato é caracterizado por um conjunto de freqüências de ressonância. Sons diferentes são formados variando-se o formato do trato vocal. Assim, as propriedades espectrais dos sinais da fala variam com o

tempo e dependem do formato do trato vocal. Como os vários articuladores (isto é, lábios, língua, mandíbula e velum) mudam continuamente de posição durante a fala, o formato das várias cavidades mudam drasticamente.

A Tabela 2-1 apresenta valores típicos dos quatro primeiros formantes (F1 para a freqüência do primeiro formante, F2 para o segundo e assim por diante) em Hz e suas respectivas larguras de banda (B1 para a largura de banda do primeiro formante, B2 para o segundo e assim por diante) em Hz .

Vogal	F1	F2	F3	F4	B1	B2	B3	B4
[a]	649	1276	2465	3588	148	58	57	97
[é]	470	1972	2637	3442	72	88	83	321
[ê]	325	2077	2660	3394	45	97	87	271
[i]	219	2177	2845	3661	49	42	263	159
[ó]	556	997	2589	3404	64	39	171	131
[ô]	342	824	2555	3135	85	55	228	186
[u]	353	732	2574	3215	96	147	156	204

Tabela 2-1 Valores típicos das vogais pronunciadas pelo autor

A Figura 2-3 mostra as características de transmissão da freqüência de algumas vogais. A natureza das ressonâncias é claramente vista. É útil notarmos que somente os três primeiros formantes são os mais importantes na determinação do som que é ouvido, embora os formantes superiores sejam necessários para produzir sons de uma qualidade aceitável. Este fato é a base para vários sistemas de compressão de sinais de fala para o armazenamento ou transmisão a baixas taxas.

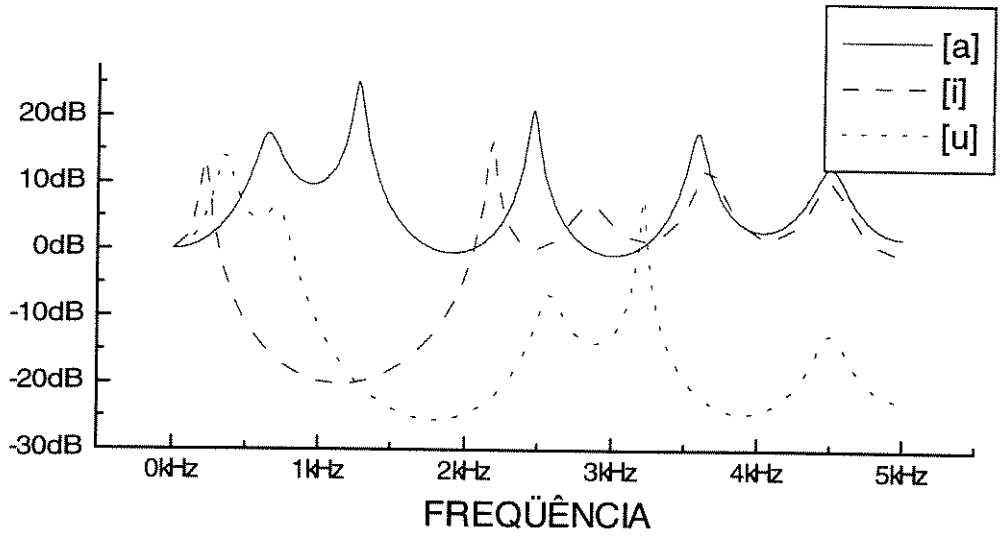


Figura 2-3 Posição das formantes das vogais [a], [i], [u]

2.3 Classificação dos Sons

Os sons da fala podem ser classificados em 3 categorias de acordo com o seu modo de excitação. Sons Vozeados (sonoros) são produzidos forçando o ar através da glote com a tensão das cordas vocais ajustadas; assim elas vibram como uma oscilação amortecida, produzindo pulsos quase periódicos de ar, os quais excitam o trato vocal. Sons fricativos são gerados formando-se uma constrição em algum ponto do trato vocal (usualmente em direção ao final da boca), e forçando o ar através da restrição em uma velocidade suficiente para produzir turbulência. Isto cria uma fonte de ruído de espectro largo para excitar o trato vocal. A fonte está no ponto de constrição e gera um fluxo de ar quase aleatório. Sons plosivos resultam de se fazer um completo fechamento (oclusão), constituindo em um aumento da pressão do ar atrás do fechamento, e liberando-o abruptamente. Aqui, a fonte está no ponto de fechamento e consiste na rápida liberação da pressão do ar construída atrás da oclusão total. Conclui-se que a voz é a principal fonte de energia na fala, embora não seja a única pois tem-se ainda a fonte fricativa e a fonte plosiva.

3. Elementos Básicos para Síntese de Sinais de Fala

3.1 Sintetizadores de Formantes dos Sinais de Fala

Sintetizadores de formantes derivam de uma aproximação da forma de onda da fala por um conjunto simples de regras formuladas no domínio acústico. O projeto do sintetizador é baseada na teoria acústica da produção da fala apresentado por Fant (1960) e sumarizado na Figura 3-1.

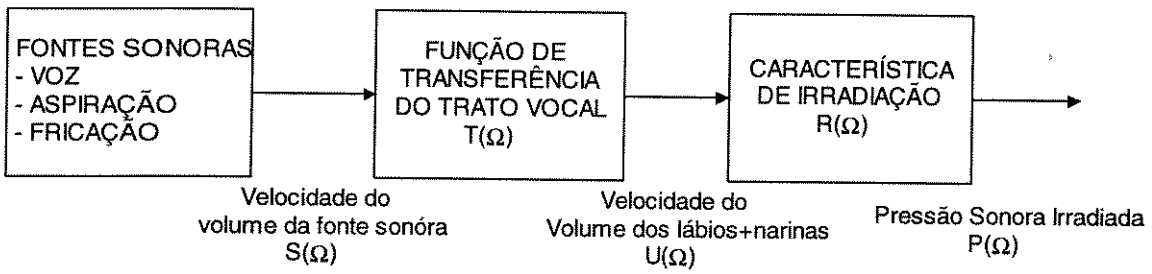


Figura 3-1 O espectro de saída de um som da fala

Na Figura 3-1, P(Ω), pode ser representado no domínio da freqüência como um produto do espectro da fonte sonora S(Ω), a função de transferência do trato vocal T(Ω), e a característica de irradiação, R(Ω). Assim P(Ω)=S(Ω)T(Ω)R(Ω).

Tratando cada fonte separadamente, nós podemos caracterizá-las no domínio da freqüência por um espectro da fonte de entrada, S(Ω), onde Ω é a freqüência. Cada fonte de som que excita o trato vocal funciona como um sistema ressonador análogo ao órgão de tubos. Uma vez que o trato vocal pode ser modelado como um sistema linear, este pode ser caracterizado no domínio da freqüência por uma função de transferência linear, T(Ω) que nada mais é que a razão da velocidade do volume de ar que passa pelos lábios e nariz, U(Ω), pela fonte de entrada S(Ω). O espectro de pressão sonora que

poderia ser captado a alguma distância dos lábios do locutor, P(Ω), é relacionada pelo produto da velocidade do volume de ar dos lábios mais nariz, U(Ω), com a característica de irradiação R(Ω) que descreve os efeitos da diretividade de propagação do som a partir da cabeça.

3.2 Sintetizador Básico

A premissa básica de quase todo sistema de processamento de síntese de sinais de fala é a de que a fonte da excitação e o sistema do trato vocal são independentes. As fontes de excitação são: um gerador de impulsos (controlado externamente por um sinal de período do pitch) e um gerador de números aleatórios. O gerador de impulsos produz um impulso (correspondente a inicialização do sopro de ar) a cada N_o amostras. Esta duração é referida como o período de pitch e é o recíproco da freqüência fundamental ou freqüência de oscilação das cordas vocais. A saída do gerador de números aleatórios simula a turbulência quase aleatória e a forma de onda de sons surdos.

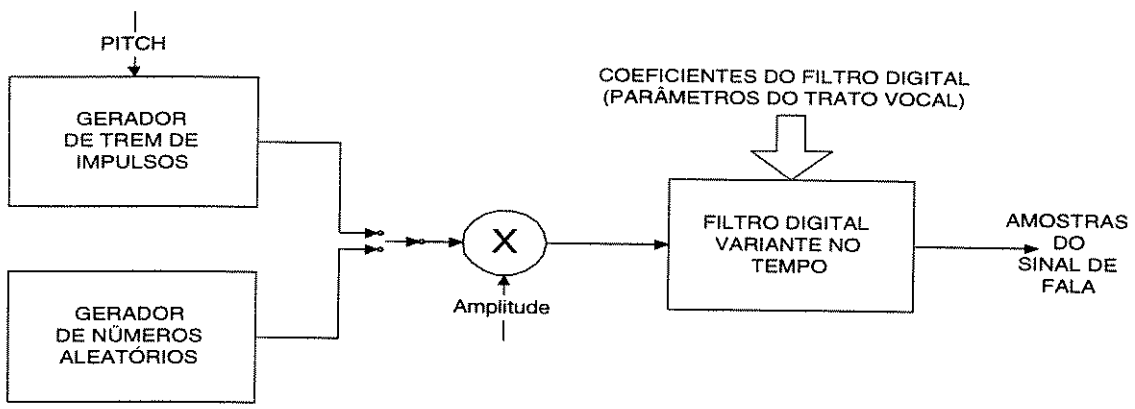


Figura 3-2 Modelo digital da produção da fala (Após Schafer)

Ambas as fontes podem ser aplicadas à entrada de um filtro linear digital variante no tempo. Este filtro simula o sistema de trato vocal e assim, os coeficientes do filtro especificam de alguma maneira o trato vocal+fonte glotal como uma função do tempo durante a fala contínua. Os valores dos parâmetros de controle são atualizados a cada 5 ms, indicando assim uma

nova configuração do trato vocal. Isto é freqüentemente suficiente para imitar a nova configuração do trato vocal, mesmo nas transições de formantes mais rápidas. Os parâmetros podem ser atualizados, se desejado, somente a cada 10 ms com uma pequena perda na qualidade de saída.

O controle de ganho entre a fonte e o sistema permite uma certa flexibilidade do nível acústico da saída. O formato de onda digital da saída do filtro corresponde a saída final do sinal de fala, amostrada na taxa apropriada.

O controle do modelo acima requer um conhecimento dos parâmetros apropriados (período de pitch, amplitude e coeficientes do filtro) como função do tempo. O objetivo da maioria dos sistemas de síntese de fala é usar estes parâmetros, obtendo um sinal sintético de fala.

3.3 Função de Transferência do Trato Vocal

3.3.1 Ressonadores Digitais

O Trato Vocal pode ser simulado por um conjunto de ressonadores digitais. O ressonador digital é o bloco básico de construção de um sintetizador de formantes. Dois parâmetros são usados para especificar as características de saída x entrada de um ressonador: a freqüência de ressonância do ressonador *F* e a largura de banda de ressonância *BW*. Amostras da saída de um ressonador digital, *y[nT]* são computados a partir de uma seqüência de entrada, *x[nT]*, pela equação:

$$y[nT] = Ax[nT] + By[nT - T] + Cy[nT - 2T] \qquad \textbf{Equação 3-1}$$

onde *y[nT-T]* e *y[nT-2T]* são os dois valores prévios da seqüência de saída *y[nT]*. As constantes *A*, *B* e *C* são relacionadas com a freqüência de ressonância *F* e a largura de banda *BW* de um ressonador pela transformação invariante ao impulso. (Rabiner e Gold - 1975) (Klatt - 1980)

$$r = \exp(-\pi\, BW\, T)$$

$$C = -r^2$$

$$B = 2\cos(-2\pi F T)$$

$$A = 1-2r\cos(-2\pi F T)+r^2$$

onde:

T = período de amostragem = $1/F_A$

F_A = Freqüência de amostragem

F = Freqüência de ressonância do filtro.

Se assumirmos que as condições iniciais $y[-1]=0$ e $y[-2]=0$, então a resposta ao impulso será:

$$h(n) = \alpha_1 r^n \operatorname{sen}[\theta\,(n+1)] \quad \text{onde} \quad \theta=-2\pi F T$$

A resposta impulsiva, $h(n)$, representa um sistema de segunda ordem no qual a resposta ao impulso é uma senóide amortecida sendo:

$$\alpha_1 = \frac{1-2r(\cos\theta)+r^2}{\operatorname{sen}(\theta)}$$

A resposta em freqüência associada à resposta impulsiva pode ser escrita como:

$$H\,(e^{j\omega}) = \frac{1-2r(\cos\theta)+r^2}{1-2r(\cos\theta)e^{-j\omega}+r^2e^{-2j\omega}} \qquad \textbf{Equação 3-2}$$

A resposta impulsiva e o módulo da resposta em freqüência de um sistema de segunda ordem correspondente a um valor fixo de $\theta=(\pi/4)$ e variando BW são mostrados nas Figura 3-3 e Figura 3-4 respectivamente. Pode-se observar que um sistema de segunda ordem representa um simples ressonador digital.

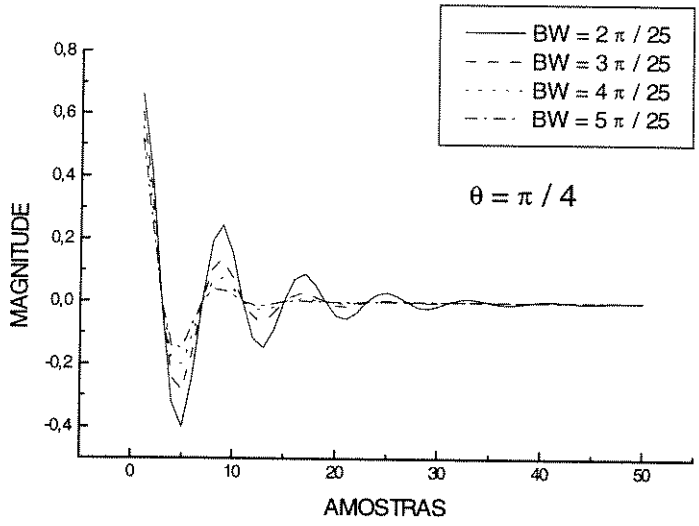


Figura 3-3 Resposta ao impulso de um sistema de segunda ordem

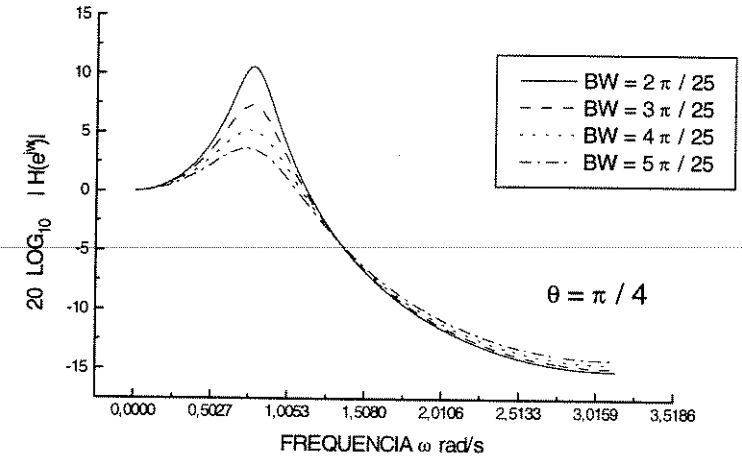


Figura 3-4 Resposta em freqüência de um sistema de segunda ordem (Ressonador)

Os valores dos parâmetros de controle do ressonador *F* e *BW* são atualizados a cada 5 ms, causando a mudança discreta das constantes da Equação 3-1 diferença em pequenos passos. Mudanças bruscas nestas constantes podem introduzir estalitos na saída de um sintetizador. Felizmente a teoria acústica indica que as freqüências dos formantes devem sempre mudar vagarosamente e continuamente relativo aos intervalo de atualização dos parâmetros de controle de 5 ms.

3.3.2 Anti-ressonadores Digitais

Embora um ressonador digital seja o bloco básico de um sintetizador, um anti-ressonador (também chamado de anti-formante ou par de zeros da função de transferência) torna-se necessário para dar um melhor formato ao espectro da fonte de voz e/ou para simular os efeitos de nasalização na função de transferência do trato vocal em algumas configurações. Um anti-ressonador possui uma função inversa ao ressonador. Assim, supondo um ressonador e um anti-ressonador acoplados em cascata com a freqüência de ressonância e largura de banda idênticas teremos:

$$H'(z) = \frac{1}{H(z)}$$

$$H'(z) = \frac{1 - Bz^{-1} - Cz^{-2}}{A}$$

$$H'(z) = A' + B'z^{-1} + C'z^{-2}$$

e a equação diferença será dada por:

$$y[n] = A'x[n] + B'x[n-1] + C'x[n-2]$$

onde:

$$A' = 1 / A$$
$$B' = -B / A$$
$$C' = -C / A$$

Observamos também que temos um filtro FIR pois a saída só depende das entradas passadas. A resposta ao impulso unitário será:

$$h[n] = A'\delta[n] + B'\delta[n-1] + C'\delta[n-2]$$

A resposta de freqüência de um anti-ressonador é uma imagem espelhada da resposta apresentada na Figura 3-4 e é apresentada na Figura 3-5, ou seja:

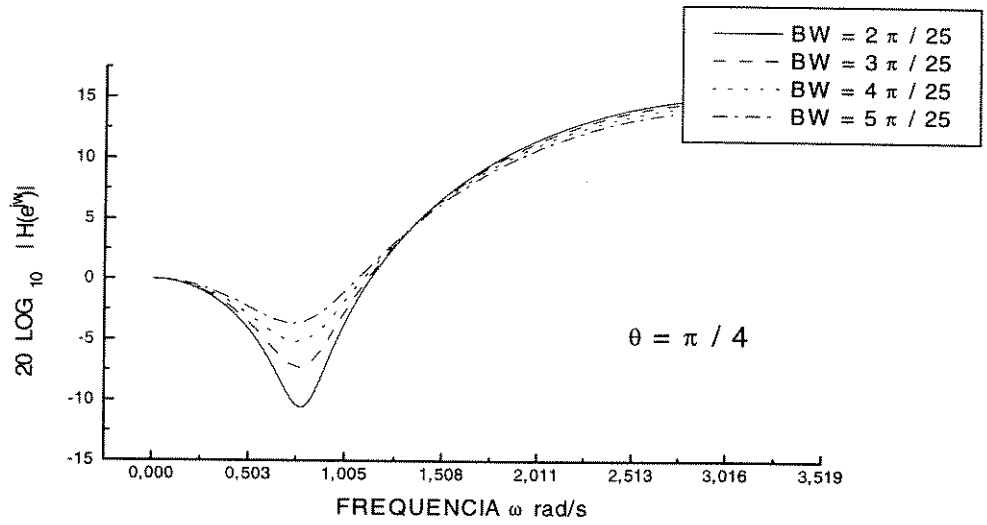


Figura 3-5 Resposta em freqüência de um sistema de segunda ordem (Anti-ressonador)

Um anti-ressonador pode então ser realizado por uma simples mudança no ressonador.

3.3.3 Ressonador Passa-Baixa

Em um caso especial, a freqüência *F* de um ressonador pode ser feita igual a zero, produzindo então um filtro passa-baixa com atenuação nominal de -12 dB por oitava e freqüência de corte igual a *BW*/2. Acoplando dois ressonadores passa-baixa teremos um filtro com atenuação nominal de -24 dB por oitava.

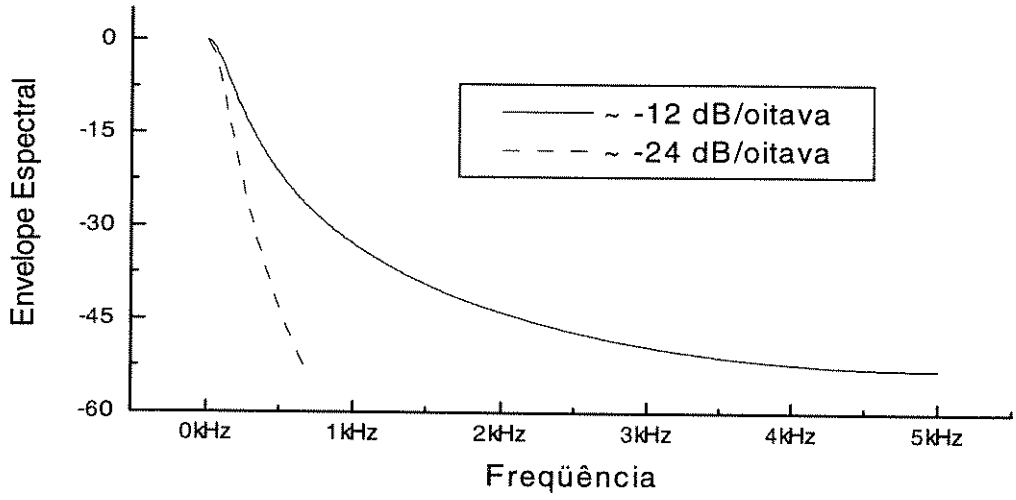


Figura 3-6 Resposta de ressonadores com F=0.

3.4 Configuração dos Ressonadores do Trato Vocal

Diferentes configurações são empregadas para se tentar alcançar o mesmo resultado, uma aproximação de alta qualidade da fala humana. Duas configurações são comuns aos melhores sintetizadores de formantes.

Em um tipo de configuração chamada sintetizador de formantes em paralelo, os ressonadores de formantes que simulam a função de transferência do trato vocal são conectados em paralelo como mostrados na Figura 3-7. Cada ressonador (R1, R2,...) é precedido por um controle de amplitude (A1, A2, ...) que determina a amplitude relativa de um pico espectral (formante) no espectro de saída para os sons sonoros e não-sonoros (surdos).

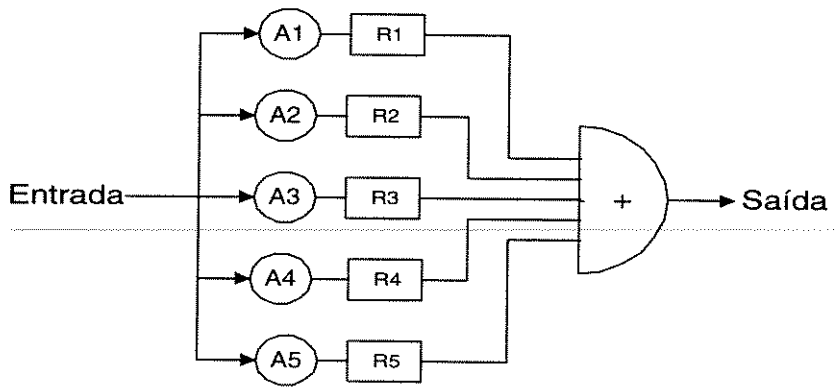


Figura 3-7 Função de transferência do trato vocal simulada por ressonadores digitais em paralelo

No segundo tipo de configuração, chamado sintetizador de formantes em cascata, os sons são sintetizados usando um conjunto de ressonadores de formante conectados em cascata como visto na Figura 3-8.

Figura 3-8 Função de transferência do trato vocal simulada por ressonadores digitais em cascata

A vantagem da conecção em cascata é que a amplitude relativa dos picos dos formantes para a vogal são próximos dos reais e se ajustam automaticamente, sem a necessidade de controle de amplitudes individuais para cada formante.

A desvantagem é que ainda é necessário uma configuração em paralelo para a geração de sons fricativos e plosivos; (a função de transferência do trato vocal não pode ser modelada adequadamente por cinco ressonadores em cascata, quando a fonte de som esta acima da laringe) assim o sintetizador em cascata é geralmente mais complexo em uma estrutura geral.

Uma segunda vantagem da configuração em cascata é a sua maior precisão em modelar a função de transferência do trato vocal durante a produção de sonoras não nasais. Será mostrada que a função de transferência de certas vogais são difíceis de se conseguir usando um sintetizador de formantes em paralelo. Embora não sendo o ideal, um sintetizador em paralelo é particularmente útil para a geração de estímulos que violam a relação normal de amplitudes entre formantes, ou se, deseja-se gerar, por exemplo, modelos únicos para formantes.

Pode ser observado que foram utilizados em ambas as configurações cinco ressonadores. Isto se deve ao fato de que a maioria da energia sonora da fala está contida nas freqüências entre 80 e 8000 Hz. Todavia, testes de inteligibilidade de filtros passa-faixa indicam que a inteligibilidade não muda se as freqüências acima de 5.000 Hz forem removidas. A fala filtrada por um filtro passa-baixa soa perfeitamente natural mas perde-se qualidade nas fricativas devido as altas freqüências. Como o trato vocal possui em média 17 cm de comprimento, a primeira ressonância de quarto de onda se encontra por volta de 500 Hz. Supondo um trato vocal uniforme, as próximas ressonâncias ocorrerão aproximadamente em 1500, 2500, 3500, 4500.... Assim, até 5000 Hz, teremos 5 formantes.

Poderemos a partir do que foi observado, construir um hardware analógico para implementar o sintetizador, utilizando circuitos ressonadores analógicos. No entanto as vantagens da implementação em software, são substanciais. O sintetizador não precisa de repetidas calibrações, ele é estável e a relação sinal ruído no processamento pode ser feito tão grande quanto desejada. A configuração pode ser facilmente mudada quando novas idéias forem propostas. Por exemplo, vozes de mulheres e crianças podem ser sintetizadas

facilmente com qualidade ruim com modificações apropriadas no número de formantes (Klatt e Klatt - 1990).

4. Sintetizador de Formantes de Klatt

4.1 Introdução

O projeto do sintetizador de formantes de Klatt é baseado na teoria acústica da produção da fala apresentado por Fant (1960) e sumarizada no capítulo 3.

O sintetizador de Klatt utiliza a configuração cascata/paralela para a simulação da função de transferência do trato vocal T(Ω). Pode ser configurado para um uso híbrido de sintetizador cascata/paralelo como na Figura 4-1, ou para aplicações especiais como um sintetizador estritamente paralelo como na Figura 4-2, embora na prática somente a configuração cascata/paralela seja usada. O experimentador deve decidir qual configuração irá ser empregada. A mudança na configuração depende do estado de uma simples chave e o programa evita operações computacionais desnecessárias para os ressonadores que não serão usados.

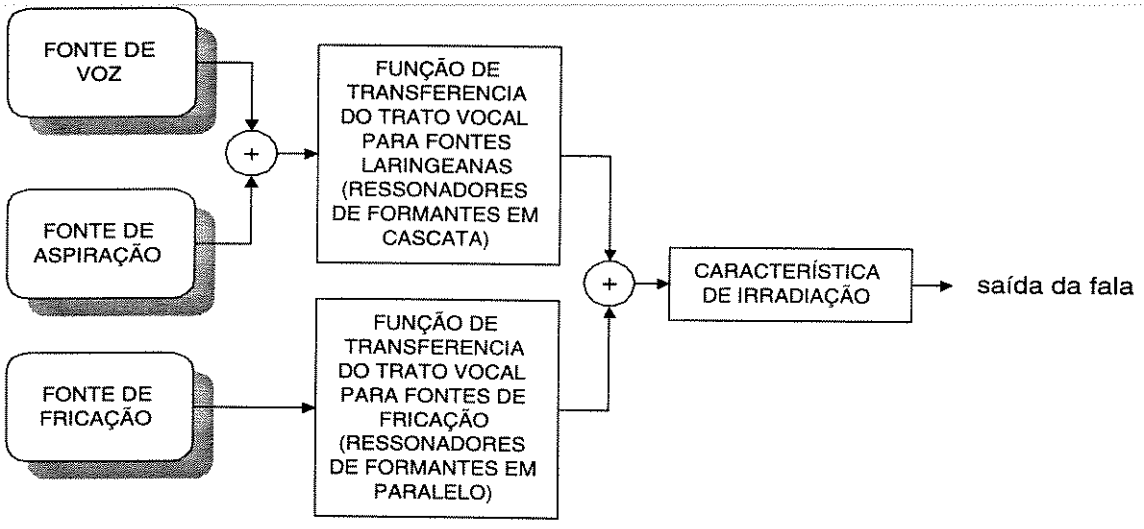


Figura 4-1 Configuração dos formantes cascata/paralela

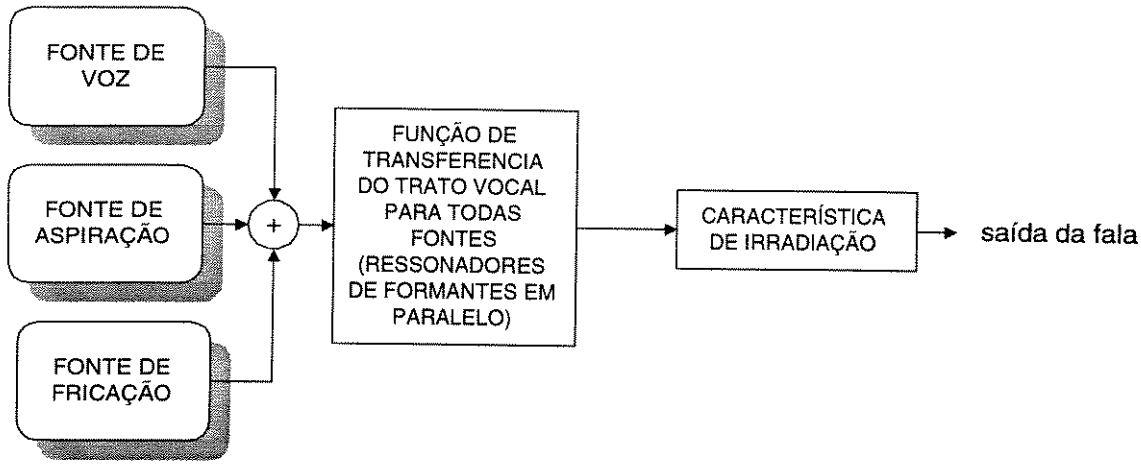


Figura 4-2 Configuração dos formantes toda em paralela para propósitos especiais

4.2 *Diagrama de Blocos do Sintetizador*

Um diagrama de blocos do sintetizador é mostrado na Figura 4-3. Há 39 parâmetros de controle que determinam as características da saída. O nome e a faixa de valores de cada parâmetro são dados na Tabela 4-1. Como pode ser visto da Tabela 4-1, pode-se variar 20 dos 39 parâmetros durante a síntese dos sinais de fala para se tentar obter uma determinada elocução. Aos parâmetros constantes na Tabela 4-1 têm sido dados valores apropriados para uma voz masculina particular e estes parâmetros podem ser ajustados suavemente para aproximar à voz de outro locutor, masculino ou feminino. A lista das variáveis dos parâmetros de controle é longa comparada com a de alguns sintetizadores mas, a ênfase aqui, é para definir estratégias para a síntese de fala de alta qualidade. Nós não estamos preocupados em minimizar o conteúdo de informação na especificação dos parâmetros de controle.

Na tabela 4-1, a segunda coluna indica quando o parâmetro é normalmente constante (C) ou variável (V) durante a síntese. A Tabela 4-1 apresenta também a faixa dos valores permitidos para cada parâmetro, e o seu valor inicial.

Nº	V/C	Símbol	Nome do Parâmetro de Controle	Min	Máx	Inicial

		o				
1	V	AV	Amplitude da Voz (dB)	0	80	0
2	V	AF	Amplitude da Fricação (dB)	0	80	0
3	V	AH	Amplitude da Aspiração (dB)	0	80	0
4	V	AVS	Amplitude de Voz Quase Senoidal (dB)	0	80	0
5	V	F0	Freqüência Fundamental da Voz (Hz)	0	500	0
6	V	F1	Freqüência do Primeiro Formante (Hz)	150	900	450
7	V	F2	Freqüência do Segundo Formante (Hz)	500	2500	1450
8	V	F3	Freqüência do Terceiro Formante (Hz)	1300	3500	2450
9	V	F4	Freqüência do Quarto Formante (Hz)	2500	4500	3300
10	V	FNZ	Freqüência do Zero Nasal (Hz)	200	700	250
11	C	AN	Amplitude do Formante Nasal (dB)	0	80	0
12	C	A1	Amplitude do Primeiro Formante (dB)	0	80	0
13	V	A2	Amplitude do Segundo Formante (dB)	0	80	0
14	V	A3	Amplitude do Terceiro Formante (dB)	0	80	0
15	V	A4	Amplitude do Quarto Formante (dB)	0	80	0
16	V	A5	Amplitude do Quinto Formante (dB)	0	80	0
17	V	A6	Amplitude do Sexto Formante (dB)	0	80	0
18	V	AB	Amplitude do Caminho de By-Pass (dB)	0	80	0
19	V	B1	Largura de Banda do Primeiro Formante (Hz)	40	1000	50
20	V	B2	Largura de Banda do Segundo Formante (Hz)	40	1500	70
21	V	B3	Largura de Banda do Terceiro Formante (Hz)	40	2000	110
22	C	SW	Chave Cascata (0) / Paralelo (1)	0	1	0
23	C	FGP	Freqüência do 1° Ressonador Glotal (Hz)	0	600	0
24	C	BGP	Largura de Banda do 1° Ressonador Glotal (Hz)	100	2000	100
25	C	FGZ	Freqüência do Zero Glotal (Hz)	0	5000	1500
26	C	BGZ	Largura de Banda do Zero Glotal (Hz)	100	10000	6000
27	C	B4	Largura de Banda do Quarto Formante (Hz)	100	3000	250
28	V	F5	Freqüência do Quinto Formante	3500	4900	3750
29	C	B5	Largura de Banda do Quinto Formante (Hz)	150	4000	200
30	C	F6	Freqüência do Sexto Formante (Hz)	4000	4999	4900
31	C	B6	Largura de Banda do Sexto Formante (Hz)	200	2000	1000
32	C	FNP	Freqüência do pólo Nasal (Hz)	200	500	250
33	C	BNP	Largura de Banda do Pólo Nasal (Hz)	50	500	100

34	C	BNZ	Largura de Banda do Zero Nasal (Hz)	50	500	100
35	C	BGS	Largura de Banda do 2º Ressonador Glotal (Hz)	100	1000	200
36	C	SR	Taxa de Amostragem - (*Sample Rate)*	5000	20000	10000
37	C	NWS	Nº de amostras da forma de onda por intervalo de sintese	1	200	50
38	C	G0	Controle de Ganho Geral (dB)	0	80	47
39	C	NFC	Número dos ressonadores em Cascata	4	6	5

Tabela 4-1 Lista dos parâmetros de controle para o software de sintetizador de formantes de Klatt.

Na Figura 4-3, ressonadores digitais são indicados pelo prefixo R e os controles de amplitude por um prefixo A. Cada ressonador R_n tem um parâmetro de controle da freqüência de ressonância F_n e um parâmetro de controle de largura de banda BW_n

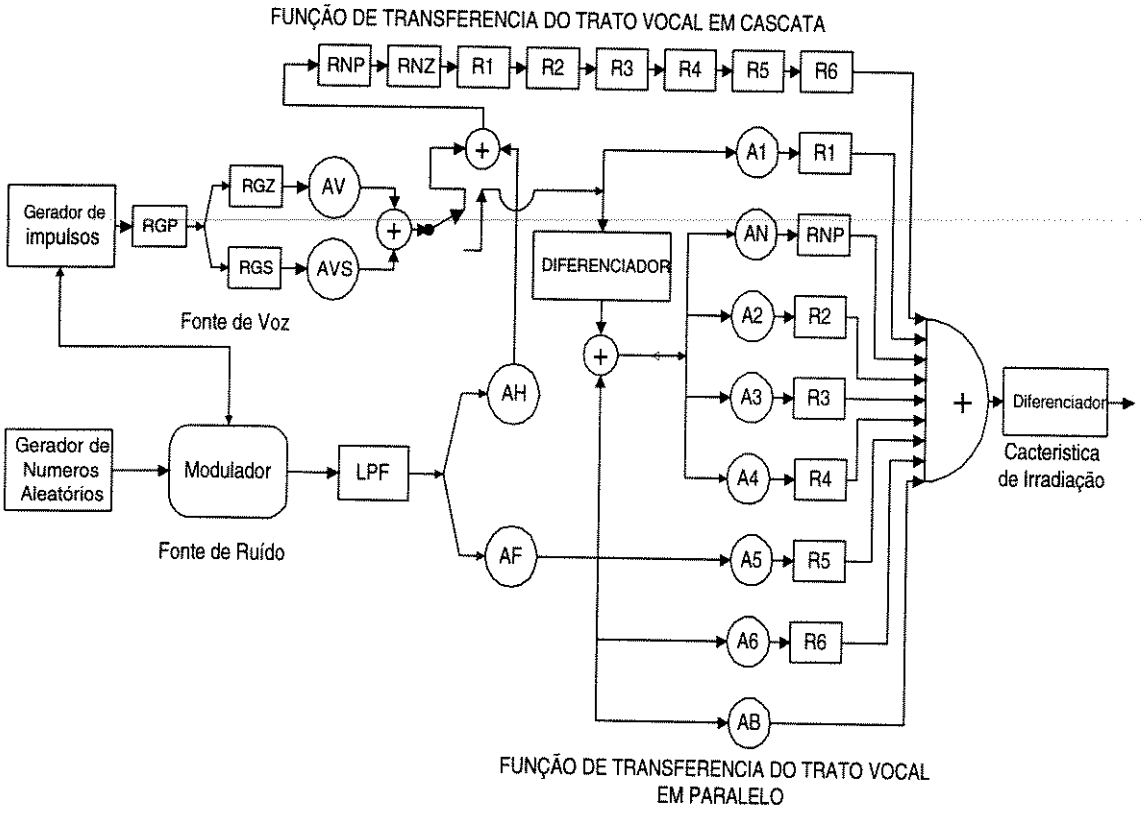


Figura 4-3 Diagrama de blocos do sintetizador de formantes cascata/paralelo.

4.3 Fontes de Som

Há dois tipos de fontes sonoras que podem ser ativadas durante a produção da fala: a primeira envolve vibrações quase periódicas das cordas vocais. A segunda envolve a geração de turbulências pelo rápido fluxo de ar passando por uma constrição (estreitamento do trato vocal). O ruído resultante é chamado aspiração, se a constrição ocorrer nas proximidades das cordas vocais. Se a constrição for localizada acima da laringe como por exemplo durante a produção de sons como [s], o ruído resultante é chamado de fricação. O som [s] representa s, ss ou ç em senso, passo e açúcar.

Quando voz e geração de ruído turbulento coexistem como em uma fricativa sonora como [z] em *casa*, o ruído é modulado em amplitude periodicamente pelas vibrações das cordas vocais. Além disso, as cordas vocais podem vibrar sem se encontrarem na linha central. Neste tipo de fala, a amplitude dos harmônicos de alta freqüência do espectro da fonte sonora é significantemente reduzida e a forma se aproxima de uma onda senoidal. Portanto o sintetizador deve ser capaz de gerar pelo menos dois tipos de forma de onda para a fonte sonora (ou fonte de voz): voz normal e voz quase senoidal. Deverá ser capaz também de gerar dois tipos de forma de onda fricativas: fricativas normais e fricativas moduladas em amplitude. E ainda dois tipos de aspiração: aspiração normal e aspiração modulada em amplitude.

4.3.1 Fontes Sonoras

A estrutura de fontes sonoras é mostrada no topo esquerdo da Figura 4-3. Parâmetros de controle variável são usados para especificar a Freqüência Fundamental (F0), Amplitude de Voz normal (AV) e a Amplitude de Voz quase Senoidal (AVS).

Um trem de impulsos é gerado quando F0 é maior que zero. A amplitude de cada impulso é modulada por AV e AVS, as amplitude de voz normal e quase senoidal, em dB. A faixa de AV e AVS é de 60 dB numa vogal forte e 0 dB

quando a fonte de voz é desligada. A freqüência fundamental é especificada em Hz, um valor de F0=100 deverá produzir um trem de impulsos, com os impulsos espaçados de 10 ms. O número de amostras entre impulsos T0 é determinado por SR/F0, isto é, para uma taxa de amostragem de 10.000 amostras/segundo e freqüência fundamental de 200 Hz, um impulso será gerado a cada 50º amostra.

4.3.1.1 Voz Normal

Ignorando por um momento os efeitos do anti-ressonador glotal (RGZ), vemos que o trem de impulsos é enviado através de um filtro passa-baixa, o ressonador glotal primário (RGP), para produzir um formato de onda suavizado que se parece com uma forma de onda glotal típica. A freqüência de ressonância (FGP) varia de 0 a 100 Hz. Os impulsos filtrados têm espectros que decaem com aproximadamente 12 dB por oitava acima de 50 Hz. A forma de onda que é gerada não tem o mesmo espectro de fase como um típico pulso glotal; ou não contém os zeros espectrais que geralmente aparecem na fala natural. Estas diferenças são importantes para, por exemplo, aumentar a naturalidade da voz gerada. Klatt e Klatt (1990) introduziram vários melhoramentos no modelo das fontes de voz, com o objetivo de melhorar a qualidade da síntese, principalmente das vozes femininas. A obtenção dos parâmetros de controle deste novo modelo, entretanto, é muito mais difícil de ser estimado do que o que é adotado neste trabalho.

O anti-resonador RGZ é usado para modificar mais precisamente a forma do espectro da fonte de voz para um determinado indivíduo, usando somente um filtro passa-baixa. Os valores escolhidos para FGZ e BGZ na Tabela 4-1 são tais que tendem a criar um espectro geral de voz que de certo modo possui as características da voz de Klatt. A forma de onda produzida pelo envio do trem de impulsos através de RGP e RGZ é mostrado na Figura 4-4.

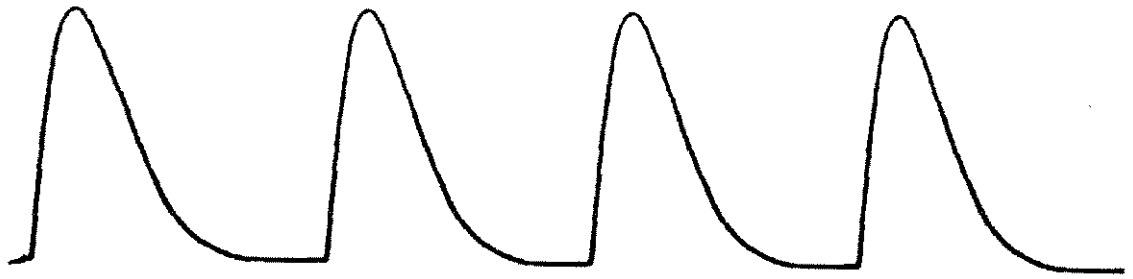


Figura 4-4 Forma de onda da voz normal

4.3.1.2 Voz quase senoidal

O parâmetro de controle da amplitude AVS determina o grau de suavização da onda glotal o som as fricativas sonoras, aspiradas sonoras e murmúrio presente nas oclusivas sonoras. Uma forma de onda apropriada para uma voz quase senoidal é obtida pela filtragem passa-baixa de um impulso por um ressonador digital RGP e RGS. O controle de freqüência RGS é setado a zero para produzir um filtro passa-baixa, e BGS=200 determina a freqüência de corte a partir da qual os harmônicos são fortemente atenuados.

Figura 4-5 Forma de onda suavizada

Uma forma de onda produzida pela fonte de voz quase senoidal é mostrado na Figura 4-5. Após os efeitos da função de transferência do trato vocal e características de irradiação serem impostas no espectro da fonte sonora, a saída da forma de onda quase senoidal contém energia significativa somente nas primeiras harmônicas da freqüência fundamental. A faixa de AVS vai 60 dB para plosivas sonoras ou fricativas sonoras fortes a 0 dB se não está presente uma voz quase senoidal. Um pouco de voz quase senoidal pode ser somado a uma fonte normal de voz em combinação com ruído de aspiração para produzir uma qualidade de voz aspirada.

4.3.2 Fonte de Fricação

A fonte de ruído é simulada no sintetizador por um gerador de números pseudo aleatórios, um modulador, um controle de amplitude AF e um filtro digital passa-baixa com -6 dB por oitava, como mostrado previamente na Figura 4-3. O espectro de uma fonte fricativa deverá ser aproximadamente plano, e a distribuição das amplitudes deverá ser gaussiana. Sinais produzidos por um gerador de números aleatórios tem um espectro plano, mas este tem uma distribuição uniforme de amplitudes entre os limites determinados pelo parâmetro de controle de amplitude AF. Uma distribuição de amplitude quase gaussiana é obtida no sintetizador somando-se 16 dos números produzidos pelo gerador de números aleatórios.

Na teoria, a fonte de ruído é uma fonte ideal de pressão sonora. A velocidade do volume de ar de um ruído de fricação depende da impedância vista pela fonte sonora. Uma vez que a função de transferência do trato vocal, $T(\Omega)$ relaciona a velocidade do volume de ar da fonte com a velocidade de saída do volume de ar nos lábios, deve-se estimar a velocidade do volume do ruído para determinar a velocidade de saída do volume de ar nos lábios. No caso geral, isto é um cálculo complexo, mas nós iremos assumir que a velocidade do volume da fonte é proporcional à integral da pressão da fonte (uma excelente aproximação para a fonte fricativa nos lábios porque a impedância de irradiação é altamente indutiva, mas somente uma aproximação para outras localização das fontes)(Klatt-1980). A integral é aproximada por um filtro digital passa-baixa de primeira ordem, que é mostrado na Figura 4-3. Amostras de saída para este filtro $y[nT]$ são relacionados com a seqüência de entrada $x[nT]$ pela equação.

$$y[nT]=x[nT]+y[nT-T]$$

Como será visto na seção 4.5, a característica de irradiação é um filtro digital passa-alta que cancela exatamente os efeitos do filtro passa-baixa (Para uma eficiência computacional, a característica de irradiação pode ser movida dentro de ambas as fontes, sonora e ruído, então a combinação da característica de

irradiação e o filtro passa baixa LPF pode ser removida da fonte de ruído)(Klatt - 1980).

A saída do gerador de números aleatórios é modulada em amplitude pelo componente chamado Modulador na Figura 4-3, quando a freqüência fundamental (F0) e a amplitude da voz (AV) são ambas maiores que zero. Sons surdos (AV=0) não são modulados em amplitude porque as cordas vocais são afrouxadas e não vibram para modular o fluxo de ar. O grau de modulação em amplitude é fixado em 50% no sintetizador. A envoltória da modulação é uma onda quadrada com período igual ao período de pitch. Experiências tem mostrado que não é necessário variar o grau de modulação em amplitude durante o curso da sentença, mas somente assegurar a sua presença nos sons fricativos sonoros e nos sons aspirados (Klatt -1980).

A amplitude do ruído de fricação é determinado por AF, em dB. Um valor de 60 dB irá gerar um ruído de fricação forte, enquanto um valor zero efetivamente desliga a fonte fricativa.

4.3.3 Fonte de Aspiração

O ruído de aspiração é essencialmente o mesmo ruído das fricativas, exceto, que é gerado na laringe. Em um modelo estritamente paralelo do trato vocal, AF pode ser usado para gerar ambos os ruídos de fricação e de aspiração. Contudo na configuração em cascata do sintetizador, o ruído de aspiração é enviado ao modelo de trato vocal em cascata (uma vez que a configuração em cascata seja especialmente projetada para modelar as características do trato vocal para fontes sonoras laringeanas) enquanto as fricativas requerem uma configuração do trato vocal em paralelo. Além disso, controles separados de amplitude são necessários para fricação e aspiração na configuração cascata/paralelo. A amplitude do ruído de aspiração que é enviada ao modelo de trato vocal é determinado por AH que é dado em dB. Um valor de 60 dB irá gerar uma forte aspiração, enquanto um valor zero efetivamente desliga a fonte de aspiração.

4.3.4 Controle de Amplitude das Fontes

Os valores dos parâmetros especificando a amplitude das fontes AV, AVS, AF e AH são ajustados pelo usuário para novos valores a cada 5 ms. Contudo, AV e AVS somente têm efeito na forma de onda sintética quando um impulso glotal é emitido. A razão para ajustar a amplitude da voz descontinuamente na aproximação de cada período glotal é para prevenir a criação de estalitos pela descontinuidade introduzida na forma de onda, por uma repentina mudança no controle de amplitude no meio de um período de voz.

A amplitude do ruído AF e AH são usadas para interpolar a intensidade da fonte do ruído linearmente sobre o intervalo de 5 ms (50 Amostras). Assim há um atraso de 5 ms na obtenção do novo valor de amplitude por uma fonte de ruído.

A interpolação permite uma aproximação gradual para uma fricativa, que poderia de outro modo ser impossível. Há contudo uma exceção para esta estratégia de controle. Uma oclusiva envolve uma aproximação mais rápida da fonte, então pode ser alcançada por 5 ms numa interpolação linear. Além disso, se AF crescer mais 50 dB de um valor especificado nos 5 ms do segmento anterior, então AF é instantânea e automaticamente mudado para seu novo valor.

4.3.5 Controle da Freqüência Fundamental

Algumas vezes é desejável especificar precisamente o tempo do primeiro pulso glotal (aproximação da voz) relativamente para uma oclusiva. Usualmente um pulso glotal é produzido no sintetizador em um tempo especificado pelo valor do parâmetro de controle do período da freqüência fundamental existente quando o último pulso glotal foi produzido. Contudo, se AV ou F0 forem zerados, nenhum pulso glotal será produzido durante estes 5 ms de intervalo

de tempo. De fato nenhum pulso glotal será produzido até o momento no qual ambos os parâmetros de controle AV e F0 se tornarem diferentes de zero.

4.4 Função de Transferência do Trato Vocal

As características do trato vocal são determinadas por sua área de seção transversal em função da distância da laringe aos lábios. O trato vocal pode ser modelado como uma linha de transmissão não uniforme na qual o comportamento pode ser determinado para freqüências abaixo de 5 kHz resolvendo a equação de onda unidimensional (Fant 1960) (Acima de 5 kHz, temos de considerar os modos de ressonância tridimensionais). Soluções para as equações de onda resultam na função de transferência que relaciona as amostras da velocidade do volume da fonte glotal à velocidade do volume de saída de ar nos lábios.

A configuração do sintetizador na Figura 4-3 inclui componentes para realizar dois tipos de função de transferência do trato vocal. A primeira, uma configuração em cascata de ressonadores digitais, modela as propriedades ressonantes do trato vocal quando a fonte de som está na laringe. A segunda, uma configuração em paralelo de ressonadores digitais e controle de amplitude, modela as propriedades do trato vocal durante a produção de ruído de fricação. A configuração paralela pode também ser usada para modelar as características do trato vocal para fontes laringeanas de som, embora a aproximação não seja tão boa como a do modelo em cascata.

4.4.1 Modelo em Cascata do Trato Vocal

A função de transferência do trato vocal pode ser representada no domínio da freqüência por um produto de pólos e zeros e contém somente cinco pares de pólos complexos e nenhum zero na faixa de freqüência abaixo de 5 kHz se não houver nasalização e a fonte de som se localizar na laringe (Fant 1960). Neste

caso, a função de transferência é equivalente ao modelo somente com pólos, pois não há ramificação lateral ou múltiplos caminhos sonoros.

Cinco ressonadores são suficientes para simular um trato vocal masculino com um comprimento típico de cerca de 17 cm, pois a média das distâncias entre formantes é igual a velocidade do som dividida pela metade do comprimento de onda, e gira em torno de 1000 Hz. Um trato vocal feminino típico é 15% a 20% menor, sugerindo que somente quatro ressonadores de formantes sejam usados para representar uma voz feminina em uma simulação de 5 kHz (ou que a simulação deva ser estendida para algo em torno de 6 kHz para síntese com 5 formantes). É sugerido para aproximação das vozes femininas e infantis que o parâmetro de controle NFC seja igual a 4, assim removendo o quinto formante do ramo em cascata do diagrama de bloco visto na Figura 4-3. Para um locutor masculino com trato vocal muito longo, pode ser necessário somar um sexto ressonador para o modelo em cascata. Como correntemente programado, NFC pode ser setado para quatro, cinco ou seis formantes no ramo em cascata. Qualquer mudança em NFC, implica uma mudança no comprimento efetivo do trato vocal. NFC não pode ser usado simplesmente para remover um formante já presente abaixo de 5 kHz pois o espectro de som resultante é quebrado de uma maneira imprópria. Estímulos com um reduzido número de formantes devem ser gerados usando uma configuração toda em paralela (Klatt -1980).

Ignorando por um momento o pólo nasal do ressonador RNP, o zero nasal do anti-ressonador RNZ e o pólo do ressonador R6, o modelo em cascata da Figura 4-3, consistindo de cinco ressonadores, tem uma função de transferência da velocidade do volume que pode ser representada no domínio da freqüência como um produto da função de transferência idêntica à Equação 3-2 (Gold e Rabiner 1968), isto é:

$$T(f) = \prod_{n=1}^{5} \frac{A(n)}{1 - B(n)z^{-1} - C(n)z^{-2}} \qquad \textbf{Equação 4-1}$$

As constantes *A(n)*, *B(n)* e *C(n)* são determinadas pelos valores das n-ésimas freqüências de formantes *F(n)* e pela n-ésima largura de banda *BW(n)*, pelas relações dadas anteriormente na Equação 3-1. As constantes *A(n)* no numerador da Equação 4-1, asseguram que a função de transferência tem um valor unitário para freqüência nula, isto é, o fluxo de ar é desimpedido.

4.4.2 Freqüência dos Formantes

Cada ressonador introduz um pico no espectro de magnitude mostrado na Figura 2-3. A freqüência central do formante "n" é determinada pelo parâmetro de controle da freqüência do formante *F*n. A amplitude do pico depende não somente de *F*n e do parâmetro de controle da largura de banda *BW*, mas também das freqüências de outros formantes, como será visto posteriormente.

Os valores das freqüências dos formantes são determinados pelo formato detalhado do trato vocal. As freqüências dos três primeiros formantes variam substancialmente com mudanças na articulação (por exemplo, a faixa observada para F1 é de 180 a 750 Hz, de F2 é de 600 a 2300 Hz e de F3 de 1300 a 3100 para um típico locutor masculino). As freqüências e larguras de banda do 4º e 5º ressonadores de formantes não variam muito e podem ser mantidas constantes com um pequeno decremento na qualidade de som na saída. Estes ressonadores de alta freqüência ajudam a dar forma geral ao espectro, mas pouco contribuem para a inteligibilidade das vogais. Os valores particulares escolhidos para as freqüências do quarto e do quinto formantes (Tabela 4-1) produzem uma concentração de energia em torno de 3 a 3,5 kHz e uma rápida queda no espectro de energia em torno de 4 kHz que é um modelo típico de muitos locutores.

4.4.3 Largura de Banda dos Formantes

A largura de banda dos formantes é função das perdas de energia devido a condução de calor, viscosidade, movimento das cavidades das paredes, irradiação do som a partir dos lábios e a parte real da impedância glotal. As

larguras de banda são difíceis de serem estimadas a partir da análise do sinal de fala natural, devido à irregularidade do espectro da fonte glotal. As larguras de banda têm sido estimadas por outras técnicas como o uso de fonte de varredura de tons sonoros (Fujimura e Lindquist, 1971). Resultados indicam que a largura de banda varia por uma fator de 2 ou mais vezes em função do segmento fonético particular sendo falado. O efeito primário da percepção da mudança de largura de banda é o incremento ou decremento na intensidade efetiva da concentração da energia do formante (Figura 4-7). As variações da largura de banda são suficientemente pequenas para que todas as larguras de banda dos formantes sejam mantidas constantes em algumas aplicações, podendo ser neste caso variados somente F1, F2 e F3 para simular a função de transferência do trato vocal para uma vogal não nasalizada e consoantes sonoras.

4.4.4 Nasais e Nasalização de Vogais

Não é possível a produção satisfatória de murmúrios nasais e a nasalização de vogais que são adjacentes às nasais utilizando-se um sistema cascata de apenas cinco ressonadores. Mais que cinco formantes estão geralmente presentes neste sons, e a amplitude dos formantes não converge para a relação inerente da configuração em cascata, devido à presença de zeros na função de transferência (Fujimura 1961, 1962).

A nasalização de vogais adjacentes é um elemento importante na síntese de consoantes nasais. A percepção mais importante da mudança associada com a nasalização de uma vogal é a redução na amplitude do primeiro formante, trazida pela presença de um par de pólos e um par de zeros de baixa freqüência próximos. A freqüência do primeiro formante também tende a se deslocar para cima na maioria das vogais nasalizadas.(Klatt -1980)

Murmúrios nasais e nasalização de vogais são aproximados pela inserção de um ressonador RNP adicional e um anti-ressonador RNZ no modelo em

cascata do trato vocal. A freqüência do pólo nasal FNP pode ser estabelecida para um valor fixo de 270 Hz durante todo o tempo. A freqüência de zero nasal FNZ deverá também ser estabelecida para um valor de 270 Hz durante um som não nasalizado, mas freqüências de um zero nasal deverão ser incrementadas durante a produção de nasais e nasalização. O par RNP e RNZ é efetivamente removido do circuito em cascata durante a síntese dos sons da fala não nasalizados se FNP=FNZ.

4.4.5 Modelo em Paralelo do Trato Vocal para Fontes Fricativas

Os pólos da função de tranferência do trato vocal são freqüências naturais de ressonância de toda a configuração do trato vocal e não dependem da localização da fonte de excitação sonora. Dependem apenas das freqüências de ressonâncias naturais do trato, que por sua vez variam apenas com a posição dos articuladores. Entretanto, a constrição que caracteriza cada som fricativo separa o trato vocal em duas cavidades: uma da constrição até a boca e a outra da constrição à glote. A reflexão de ondas sonoras que se propagam inicialmente da constrição para a glote produz o reforço de algumas freqüências da função de transferência do trato vocal e o cancelamento de outras, dependendo da localização da fonte sonora (ou seja, da constrição). Disso resulta o aparecimento de zeros na função de transferência do trato vocal e a modificação da amplitude dos formantes. O efeito perceptual de zeros no espectro de um sinal é mascarado pela energia nas frequencias das regiões adjacentes (Nagle, Chiquito - 1993).

Uma aproximação satisfatória na função de transferência do trato vocal para excitação fricativas pode ser alcançada com um conjunto paralelo de ressonadores tendo controle de amplitude e sem a necessidade de utilizar anti-ressonadores (Klatt -1980).

Um caminho direto é também incluído no modelo de trato vocal em paralelo. O caminho direto com controle de amplitude AB está presente pois a função de transferência para [f,v,p,b] não contém nenhum pico proeminente. Neste caso

o ruído não passa por nenhum ressonador e produz uma função de transferência plana.

Durante a produção de fricativas sonoras, há duas atividades da fonte de som; uma localizada na glote (som) e outra na constrição do trato vocal (fricação). A saída da fonte de voz quase senoidal é enviada através do modelo em cascata do trato vocal, enquanto a fonte fricativa excita o ramo paralelo para gerar uma fricativa sonora.

4.4.6 Simulação da Configuração em Cascata pela Configuração em Paralelo

A função de transferência da excitação laringeana do trato vocal pode, também, ser aproximada por cinco ressonadores conectados em paralelo. Os mesmos ressonadores que formam o ramo paralelo para excitação da função da fricação podem ser usados para sintetizar qualquer som sonoro, se valores apropriados forem escolhidos para as amplitudes dos formantes.

As seguintes regras resumem o que acontece com a amplitude dos formantes na função de transferência T(Ω) no modelo em cascata quando as freqüências e as larguras de banda dos cinco primeiros formantes são mudadas. Estas relações seguem diretamente da Equação 3-1 supondo que cada freqüência do formante, F(n) seja pelo menos 5 a 10 vezes maior que sua largura de banda BW(n) (Klatt - 1980).

1 - Os picos de formante na função de transferência são iguais para o caso em que a freqüências dos formantes são estabelecidas para 500,1500, 2500, 3500 e 4500 Hz e a largura de banda é igual a 200 Hz. Isto corresponde a um trato vocal com uma área de seção transversal uniforme, uma glote fechada, lábios abertos (e um conjunto de valores de largura de banda não realísta), como mostrado na Figura 4-6.

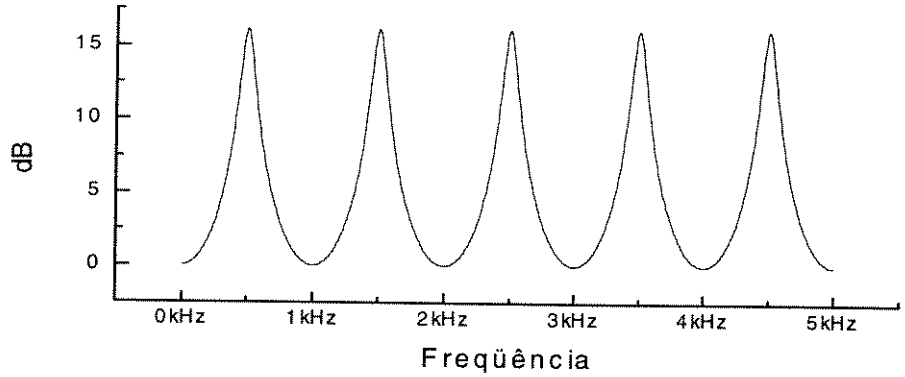


Figura 4-6 Função de transferência de um trato vocal uniforme

2 - A amplitude do pico do formante é inversamente proporcional à sua largura de banda. Se a largura de banda de um formante for dobrada, seu pico será reduzido em 6 dB. Se a largura de banda cair pela metade, o pico será incrementado em 6 dB, como mostrado na Figura 4-2.

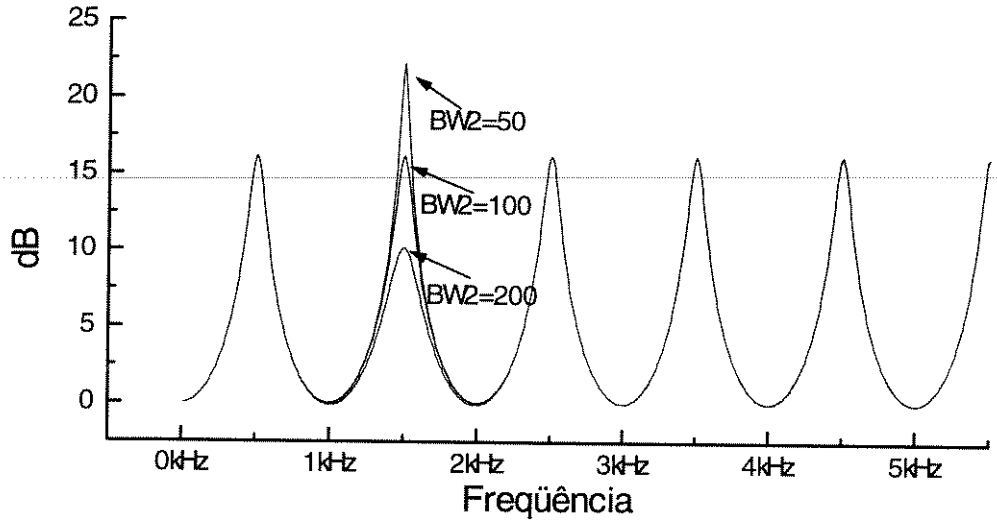


Figura 4-7 Influência da mudança da largura de banda

3 - A amplitude de um pico do formante é proporcional à freqüência do formante. Se a freqüência do formante for dobrada, o pico desta formante crescerá em 6 dB, como mostrada na Figura 4-8. Isto é verdade para T(Ω) mas não o é para o espectro resultante da fala de saída. Uma vez que o espectro da fonte glotal decai com 12 dB/oitava e as características de irradiação impõe + 6 dB/oitava para a inclinação espectral, resultando em mudanças na amplitude dos formantes de +6 dB - 12 dB + 6 dB = 0 dB.

4 - Mudanças nas freqüências dos formantes também afetam a amplitude dos picos dos formantes de freqüências mais altas por um fator proporcional ao quadrado da freqüência. Por exemplo, se a freqüência de um formante for dividida por dois, as amplitudes de todos os formantes superiores serão decrescidas de 12 dB, como mostrado Figura 4-8.

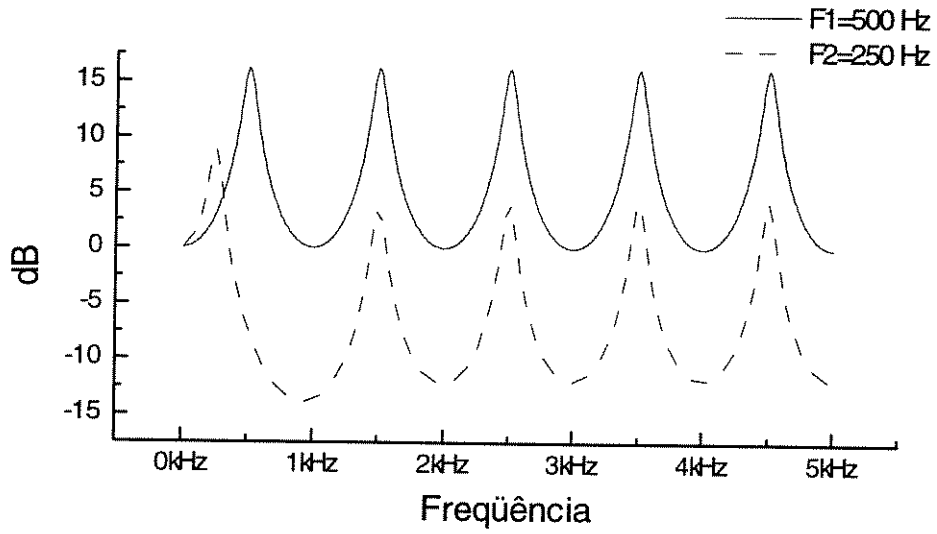


Figura 4-8 Mudanças na amplitude dos formantes causadas pelos deslocamento da freqüência do menor formante

5 - As freqüências de dois formantes adjacentes não podem estar mais próximas que uma distância de 200 Hz (largura de banda do formante). Contudo, se dois formantes estiverem separados um do outro por algo ao redor deste valor, cada pico dos formantes serão incrementados pela adição de 3 a 6 dB, como mostrado Figura 4-9.

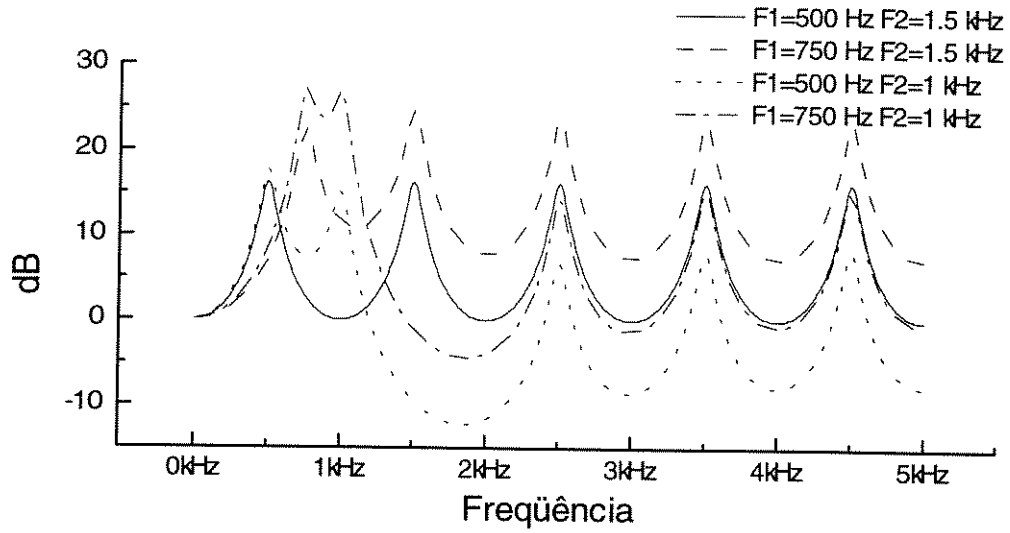


Figura 4-9 Aumento da amplitude dos formantes quando a freqüência de dois formantes estão próximas

Se as amplitudes (A1 a A5) forem todos colocados em 60 dB por exemplo, a função de transferência irá se aproximar àquele que foi encontrado para o modelo do trato vocal em cascata.

A saída dos ressonadores em paralelo são combinados com sinais alternados. A degradação perceptiva é menor no caso de sinais alternados, porque os nós espectrais são menos perceptíveis do que a energia preenchida no vale espectral entre dois formantes.

O ressonador nasal (RNP) está presente no ramo paralelo para ajudar na aproximação de murmúrios nasais durante a síntese de vogais. Nem o ressonador nasal paralelo, nem o primeiro ressonador paralelo do primeiro formante são necessários na configuração normal do sintetizador cascata/paralelo (SW=0). No entanto, eles são necessários para a simulação da nasalização quando a configuração toda paralela for utilizada (SW=1).

4.5 Características de Irradiação

O bloco rotulado na Figura 4-3 de característica de irradiação modela o efeito de diretividade de irradiação do som a partir da cabeça. A pressão de som medida diretamente em frente dos lábios a uma distância de aproximadamente

1 metro é proporcional à derivada temporal da velocidade do volume de ar dos lábios+nariz e inversamente proporcional a distância a partir dos lábios (Fant, 1960). A transformação é simulada no sintetizador tomando-se a diferença da velocidade do volume de ar dos lábios-nariz.

$$p[nT] = u[nT]-u[nT-T]$$

A característica de irradiação produz um aumento gradual no espectro geral como mostrado na Figura 4-10.

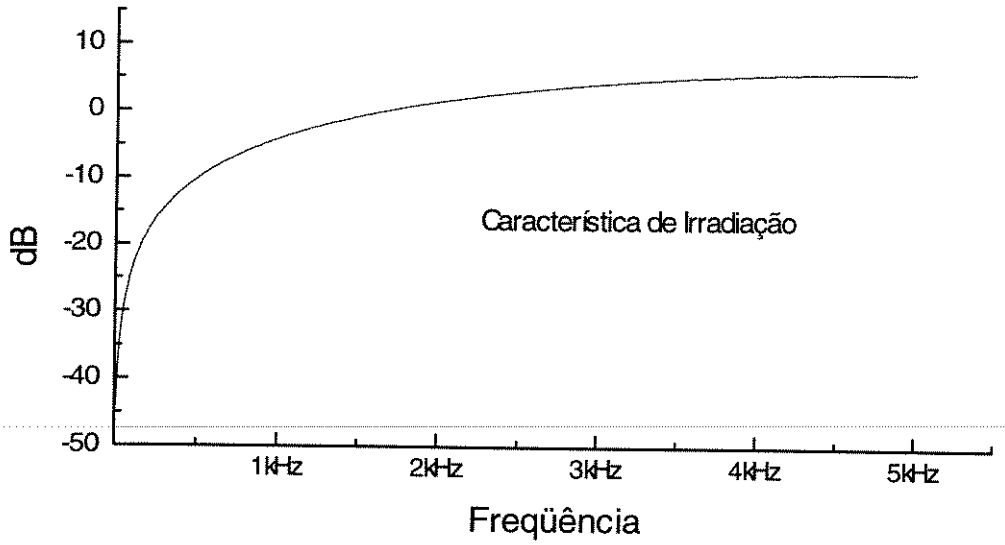


Figura 4-10 Característica de irradiação

5. O Micro Processador TMS320C25

5.1 Introdução

O início dos anos 80 marcaram o aparecimento dos microprocessadores dedicados ao processamento digital de sinais. Primeiro foi o INTEL 2920 seguido pelo NEC μPD7720 e depois o TMS32010 da Texas Instruments em 1982. Como segunda geração do TMS320, em 1985 foi lançado o TMS32020 e em 1986 o TMS320C25. Os microprocessadores digitais de sinais são caracterizados por instruções de multiplicação rápidas, um conjunto reduzido de instruções e instruções especializadas para permitir que os algoritmos de DSP sejam executados rápida e eficientemente. Neste capítulo estudaremos as características do TMS320C25.

5.2 Arquitetura

O TMS320C2x (segunda geração da família TMS320) tem uma arquitetura que valoriza a velocidade geral do sistema, comunicações, e flexibilidade na configuração do processador. Instruções e sinais de controle proporcionam transferência de blocos da memória, comunicações com dispositivos lentos e implementação de multiprocessamento. Voltado para aplicações em Processamento Digital de Sinal, a família TMS320C2x é caracterizada por instruções de multiplicar/acumular em um único-ciclo, de dois grandes blocos de RAM interna, oito registradores auxiliares com uma unidade aritmética dedicada, uma porta serial, um timer interno no hardware, rápida I/O (Entrada/Saída) para processamento intensivo de dados e outras características.

5.2.1 Histórico da Arquitetura do TMS320C25

A família TMS320 usa a arquitetura Harvard modificada. A arquitetura Harvard recebe este nome devido ao computador Harvard Mark-I, projetado por Howard Aiken, físico de Harvard, em 1937 e construído pela IBM em 1939. Este computador possuía espaços separados para os programas e para os dados. Era um computador mecânico cujo programa era rodado a partir de uma fita de papel perfurado. Os dados eram armazenados em rodas engrenadas, e realizava uma multiplicação de 10 dígitos por volta de 3 segundos. Os dados eram carregados inicializando-se as rodas engrenadas no número desejado.

O ENIAC (1946), outro computador, também tinha um espaço de dados separado do espaço de programa. O ENIAC, contudo, era um computador eletrônico. Seu espaço de dados consistia de 20 acumuladores os quais eram baseados em válvulas tríodo. No seu projeto, contadores em anéis com válvulas substituíram as rodas engrenadas do Harvard Mark-I e os dados eram carregados manualmente, setando-se as chaves. O ENIAC realizava uma multiplicação de 10 dígitos em aproximadamente de 3 milisegundos.

Estes dois computadores são uma extensão da Máquina Diferencial de Babbage (1823) e da Máquina Analítica (1834) os quais também tinham espaços separados de programas e dados. Babbage usava registros perfurados para seu programa.

A maioria dos computadores nos dias de hoje são computadores com programa armazenados. A arquitetura deste tipo de computador é chamada de arquitetura von Neumann após seu inventor Jonh von Neumann (1903-1957). A maior contribuição da arquitetura von Neumann, e a razão de sua popularidade hoje, é que ela dá ao programa a capacidade de mudar seu próprio conteúdo do programa na memória. (O computador de von Neumann não usou cartões ou fitas de papel para manter os programa mas no local usou memória eletrônica)

A arquitetura von Neumann ainda tem um aspecto da arquitetura Harvard. Tem a forma de um conjunto de registradores, os quais atuam como um buffer de dados para a memória principal e são usados para armazenar resultados e/ou dados intermediários. Muitos algoritmos podem ser implementados eficientemente com menos de oito registradores. Isto significa que um programa típico não requer muitos acessos diretos a localizações de memória de dados; basta apenas trocar os dados necessários, para dentro e para fora da memória de programa para as sucessivas rotinas. Como se sabe, este não é o caso dos algoritmos típicos de processamento digital de sinais. Estes algoritmos tipicamente requerem grandes quantidades de memória. Enquanto, a maioria dos microprocessadores e microcomputadores, os quais são microprocessadores com memória interna, possuem menos que 32 registradores, a primeira geração dos TMS320 poderia ser considerada como tendo um conjunto de 256 registradores.

Na arquitetura von Neumann o programa tem a habilidade de mudar seu próprio conteúdo da memória de programa. Quando enriquecido com o conjunto próprio de instruções, um programa também tem a capacidade de salvar dinamicamente mudança nos valores dos dados.

A família TMS320 usa arquitetura Harvard modificada, pois consiste basicamente da arquitetura Harvard estrita, possuindo ainda características da arquitetura de von Neumann.

Estas características, no TMS320, incluem a capacidade de inicializar a memória de dados a partir da memória de programas, e a habilidade para transferir da memória de dados para a memória de programa. Esta capacidade permite ao TMS320 a multiplexação por divisão de tempo em suas memórias entre tarefas bem como inicializar a memória de dados com constantes (por exemplo, um coeficiente) armazenadas na ROM. Isto minimiza o custo do sistema por eliminar a necessidade de uma ROM de dados e maximiza a utilização da memória de dados por permitir uma redefinição de suas funções.

A principal razão para o TMS320 ser baseado na arquitetura Harvard é a velocidade. Espaços separados de dados e programas permitem ciclos de busca simultaneamente de instruções de programa e dados. Isto possibilita uma maior eficiência do conjunto.

A combinação da arquitetura Harvard, onde temos os barramentos de dados e programas separados, e seu conjunto de instruções especiais para processamento digital de sinais, provê velocidade e flexibilidade para produzir uma família de microprocessadores capazes de executar 10 MIPS (milhões de instruções por segundo). A família TMS320 otimiza velocidade através da implementação de funções em hardware que outros microprocessadores implementam através de software ou microcódigo. Este tipo de hardware provê ao engenheiro projetista poderes não disponíveis anteriormente em um único chip.

5.2.2 Detalhes da arquitetura do TMS320C25

A segunda geração da família TMS320 usa um barramento de dados e um barramento de instruções de programa de 16 bits cada. Além disso, existe um barramento de endereços de 16 bits para a memória de dados e outro de 16 bits para a memória de programa. O montante de memória de dados interno ao chip, que é endereçável por um barramento de endereços de 16 bits, é 64k palavras, das quais 544 estão dentro do chip, e o restante pode ser endereçado de acordo com a necessidade do programa

O processador TMS320C25 possui um conjunto de oito registradores auxiliares denominados AR0-*Auxiliary Register Zero* até AR7-*Auxiliary Register Seven*. Cada registrador é usado para armazenar valores de 16 bits de comprimento.

Além da memória interna, o processador TMS320C25 é equipado com o hardware necessário para realizar operações lógicas e aritméticas. Possui um acumulador de 32 bits, com as funções de acumular e de buffer. O registrador T de 16 bits é usado para armazenar o multiplicando nas operações de

multiplicação. Finalmente o registrador P, ou registrador de produto, é de 32 bits e é usado para armazenar o resultado da multiplicação ou produto.

5.2.3 Características Chave do TMS320C25

O dispositivo TMS320C25 pertence à segunda geração da famíla TMS320 e possui as seguites características chave:

- Ciclo de tempo por instruções de 100 ns;
- Programação de 544 palavras na RAM interna de dados;
- Conjunto de 133 instruções;
- 4k-palavras na ROM interna de programa;
- 64k palavras endereçaveis para programa;
- 64k palavras endereçaveis para dados;
- Acumulador/ALU de 32 bits;
- Multiplicador paralelo de 16x16 bits com um produto de 32 bits;
- Instruções de um único ciclo para multiplicar/acumular;
- Instruções de repetição para um eficiente uso do espaço de programa e execução enriquecida;
- Movimentação e blocos para gerenciamento de programa/dados;
- *Timer* interno ao chip para controle das operações;
- Oito registradores auxiliares com unidade de aritmética dedicada;
- Oito níveis de stack por hardware;
- Dezesseis canais de entrada e saída;
- *Shifter* paralelo de 16 bits;
- *Wait states* para comunicação com memórias/periféricos mais lentos fora do chip;
- Porta serial para interface de codificação direta;
- Entrada de sincronismo para configuração de multiprocessamento;
- DMA concorrente usando uma extensa operação de hold;
- Instruções para implementação de filtro adaptativo, FFT's, e aritmética de precisão extendida;
- Modo de endereçamento indexado Bit-reversed;

- Gerador interno de clock;
- Alimentação simples de 5 V;
- Encapsulamento: 68 - pinos PGA ou 68 - PLCC;
- Tecnologia CMOS;

O processador TMS320C25 é equipado com memória ROM interna de 4k, programável pelo fabricante. Para endereçá-la, um barramento de endereços de 16 bits é utilizado. Contudo, com 16 bits de endereçamento é possível endereçar até 64k posições de memória de programas.

O processador TMS320C25 pode ser interfaceado com uma variedade de recursos: 64k de programa de memória externa, 63k de memória de dados externa, e uma matriz de 16 conversores A/D e D/A. Todos estes recursos podem ser acessados pelo conjunto de 16 linhas de endereços, 16 linhas de dados bidirecionais, e várias linhas de controle. Estes recursos somam-se aos recursos internos do chip: ROM de 4k para armazenamento de programas e RAM de 544 palavras.

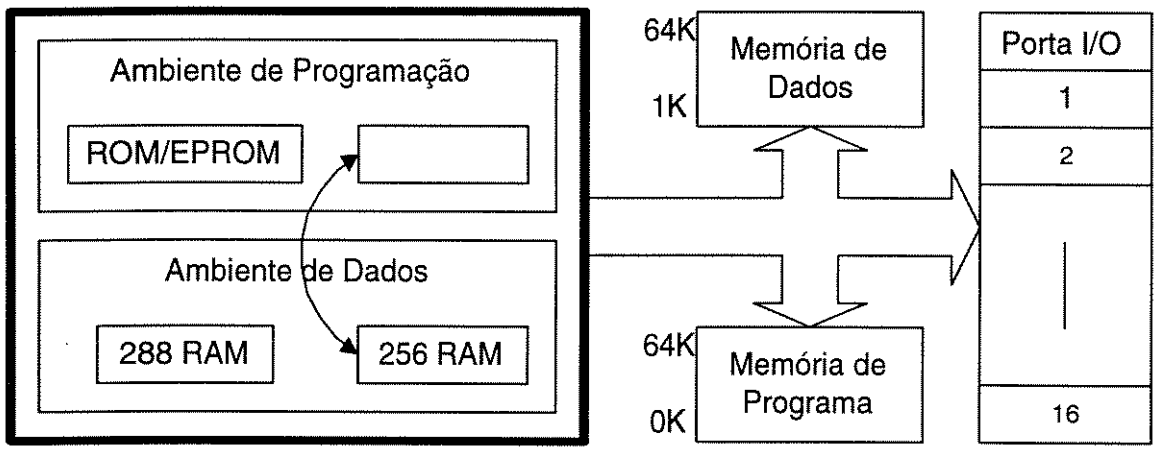


Figura 5-1 Diagrama de blocos simplificado de endereçamento do TMS320C25

5.3 Aplicações Típicas

A família TMS320 possui tamanha versatilidade e desempenho em tempo real que oferece flexibilidade numa variedade de aplicações. Além disso, o

processador TMS320C25 pode simultaneamente prover múltiplas funções geralmente requeridas em aplicações complexas.

Algumas das características do processador TMS320C2X, tais como a instrução multiplicar/acumular em um único ciclo, unidade aritmética de 32 bits, um conjunto de registradores auxiliares com uma unidade separada de aritmética, e abundante memórias RAM e ROM, fazem este dispositivo particularmente projetado para aplicações em sistemas de processamento digital de sinais. Ao mesmo tempo, aplicações de uso geral são realçadas pelo amplo espaço de endereçamento, timer interno, porta serial, múltipla estrutura de interrupção, provisão para wait-state externo, e capacidade para interface para multiprocessamento e acesso direto a memória.

O processador TMS320C2x possui flexibilidade para ser configurado de modo a satisfazer uma larga faixa de requisitos em sistemas. Isto permite ao dispositivo ser aplicado em sistemas correntes usando caros processadores Bit-Slice ou CI's comuns. Alguns das configurações de sistemas são:

- Sistemas *stand-alone* usando a memória interna;
- Sistemas de multiprocessamento paralelo usando memória global de dados compartilhados, ou;
- Coprocessamento usando sinais de controle de interface;

5.4 Diagrama de Blocos Funcional

O diagrama de blocos funcional mostradado na Figura 5-2 esboça os blocos principais e os caminhos dos dados. Maiores detalhes dos blocos funcionais serão dados nas próximas seções. O diagrama de bloco também mostra os pinos de interface do TMS320C25.

A arquitetura do TMS320C25 é construída ao redor de dois barramentos: o barramento de programa e o barramento de dados. No barramento de programa circulam o codigo da instrução e os operandos imediatos da memória de programa. O barramento de dados interconecta vários elementos, como a unidade central lógica e aritmética (CALU) e o conjunto de

registradores auxiliares à RAM de dados. Juntos, os barramentos de dados e programas podem transportar dados da memória interna RAM e memória interna e externa de programas para o multiplicador em uma operação de multiplicar/acumular em um único ciclo.

O TMS320C25 tem um alto grau de paralelismo, isto é, enquanto um dado está sendo operado pela CALU, operações aritméticas podem também ser implementadas na Unidade Aritmética dos Registradores Auxiliares (ARAU). Tal paralelismo proporciona um poderoso conjunto de operações aritméticas, lógicas e de manipulação de bits que podem ser realizadas em um único ciclo de máquina.

5.5 Visão Geral da Arquitetura do Microprocessador

O processador de sinais digitais de alto desempenho TMS320C2x, como os dispositivos TMS320C1x, implementam a arquitetura Harvard que maximiza a potência de processamento utilizando duas estruturas de barramentos de memórias separadas, uma para dados e outra para programa, para a velocidade total de execução. Instruções são incluídas para prover transferência de dados entre os dois espaços. Externamente, as memórias de programa e de dados podem ser multiplexadas sobre o mesmo barramento para maximizar a faixa de endereço para ambos os espaços, enquanto minimiza o número de pinos do dispositivo.

O aumento da flexibilidade em projeto de sistemas é provido por dois blocos de RAM de dados (num total de 544 palavras de 16 bits), um dos quais é configurável tanto como programa ou memória de dados. Um espaço de memória externa de 64k palavras pode ser diretamente endereçada para facilitar a implementação dos algoritmos em DSP.

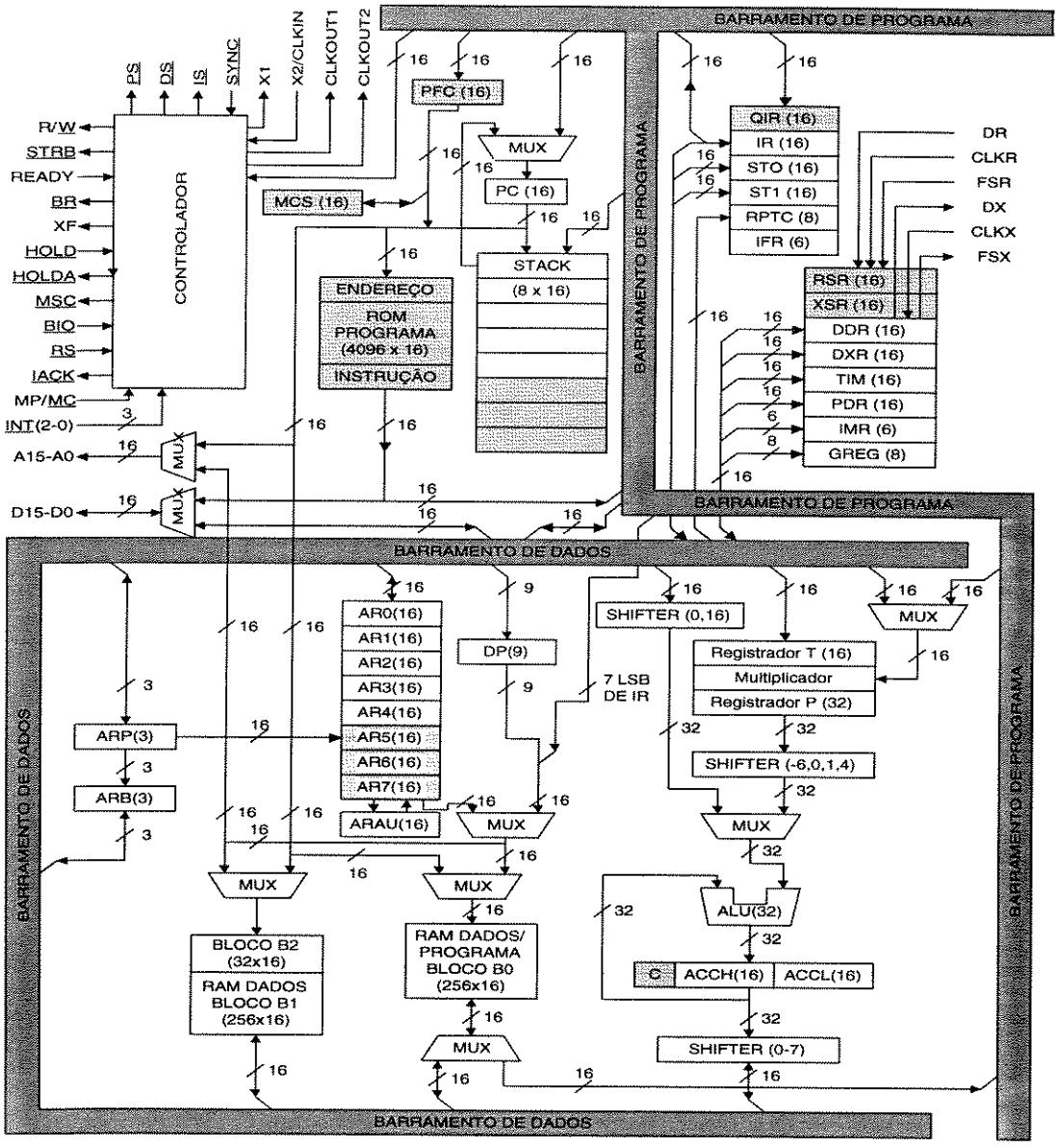


Figura 5-2 Diagrama de blocos completo do TMS320C2x

A memória ROM de 4k palavras no TMS320C25 pode ser usada para reduzir o custo de sistemas, provendo assim uma verdadeira solução de DSP em um único chip. O espaço de memória restante dos 64k palavras são alocados externamente. Amplos programas podem ser executados na velocidade máxima nesta área de programa. Programas também podem ser carregados da memória externa para a interna para operações em velocidade máxima.

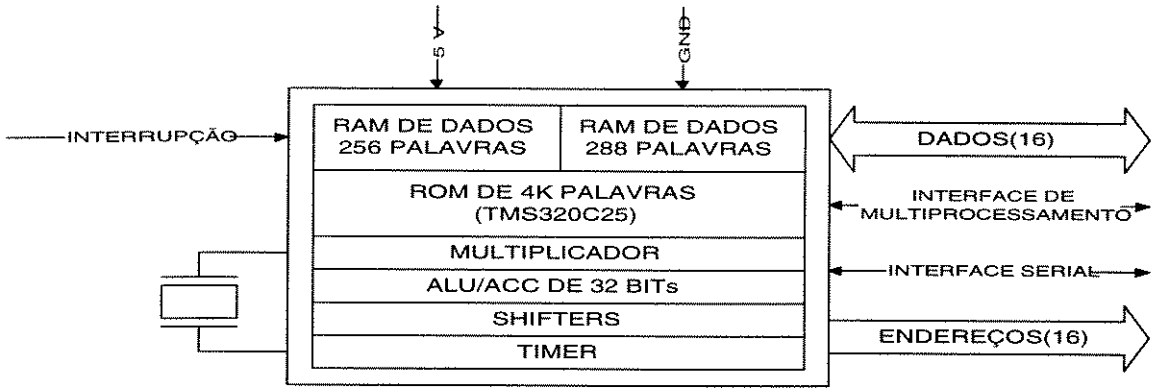


Figura 5-3 Diagrama de bloco simplificado do TMS320C2x

O processador TMS320C2X realiza operações aritméticas em complemento de dois usando a Unidade Lógica e Aritmética (ALU) e o acumulador (ACC). A ALU é uma unidade aritmética de uso geral que opera usando palavras de 16 bits provenientes da RAM de dados ou vindo de instruções imediatas, ou ainda usando o resultado de 32 bits do registrador de produto de multiplicação (PR). Além disso, para instruções de aritmética usual, a ALU pode realizar operações Booleanas, tendo a capacidade de manipulação de bits requeridas em controles de alta velocidade. O acumulador de 32 bits de comprimento é dividido em palavra alta (ACCH) (bits 31 a 16) e palavra baixa (ACCL) (bit 15 a 0). Há instruções para armazenamento da parte alta (SACH) e parte baixa (SACL) do acumulador na memória.

O multiplicador realiza uma operação de 16x16 bits em complemento de dois com um resultado de 32 bits em um simples ciclo de instrução. A multiplicação consiste de três elementos: o registrador T, o registrador P e uma matriz de multiplicação. O registrador T armazena temporariamente o multiplicando, o registrador P armazena o produto de 32 bits. Valores a serem multiplicados vêm tanto da memória de dados, quanto da memória de programa, quando são usados as instruções MAC/MACD, ou os dados são processados imediatamente pela instrução MPYK (*multiply immediate*). O multiplicador rápido interno permite realizar eficientemente operações fundamentais de DSP tais como convolução, correlação e filtragem.

O *Shifter* de escalonamento do TMS320C2x tem uma entrada de 16 bits conectada ao barramento de dados e uma saída de 32 bits conectada a ALU. O shifter produz um deslocamento a esquerda dos bits de 0 a16 dos dados de entrada, como programada na instrução. Os LSBs da saída são preenchidos com zeros, e os MSBs podem ser tanto preenchidos com zeros ou com sinal extendido, dependendo do estado do bit de modo de extensão no registrador ST1. O shifter tem a capacidade adicional de habilitar o processador a fazer escalonamento numérico, extração de bits, aritmética extendida e prevenção de overflow.

A interface interna de memória consiste de um barramento paralelo a 16 bits (D15-D0), um barramento de endereços de 16 bits (A15-A0).

O processador TMS320C25 possui oito níveis de stack por hardware para salvar o conteúdo do contador de programa (PC - program conter) durante interrupções e chamadas a subrotinas. Instruções são disponíveis para salvar completamente o contexto do dispositivo. Instruções de PUSH e POP permitem um nível de aninhamento restrito apenas ao montante de memória RAM disponível. As interrupções usadas neste dispositivo são mascaráveis.

Operações de controle são suportadas no TMS320C2x, mapeando-se em memória um timer de 16 bits, um contador de repetição, três interrupções externas mascaráveis e interrupções internas geradas por operações na porta serial ou pelo timer. Um mecanismo interno protege daquelas instruções que são repetidas ou requerem ciclos múltiplos para apresentar o sinal READY e de holds e interrupções.

Um porta serial interna full-duplex provê comunicação direta com dispositivos seriais tais como codecs, conversores A/D seriais e outros sistemas. Os sinais de interface são compatíveis com codecs e com muitos outros dispositivos, usando um mínimo de hardware externo. As duas portas seriais com registradores mapeados em memória (registro de dado transmitido/recebido) podem ser operadas nos modos de 8 e 16 bits. Cada registrador tem um sinal

externo de entrada de clock, uma entrada de sicronizador de quadro e registradores de shifter associados.

Comunicações seriais podem ser usadas entre processadores em aplicação de multiprocessamento. O TMS320C2x tem a capacidade de alocação global do espaço de memória de dados e comunicação com este espaço via os sinais de controle BR (*bus request* - requisição de barramento) e READY. O registrador de alocação global de memória de 8 bits mapeado em memória (GREG) especifica 32k palavras do espaço de memória global de dados do TMS320C2x. Se a instrução corrente endereça um operando dentro deste espaço, BR é declarado à requisitar o controle do barramento. A extensão do ciclo de memória é controlado pela linha de READY.

O TMS320C2x suporta as operações de acesso direto a memória (DMA - Direct Memory Access) para sua memória externa de dados/programa usando os sinais de HOLD e HOLDA. Outro processador pode tomar completamente o controle da memória externa do TMS320C2x colocando a linha HOLD em estado baixo. Isto faz com que o TMS320C2x coloque seu barramento de endereços, dados e linhas de controle no estado de alta impedância. As sinalização entre o processador externo e o TMS320C2x pode ser feito, usando interrupções. O TMS320C25 tem dois modos disponíveis: um, como no TMS32020, no qual a execução é suspensa enquanto a linha HOLD é colocado em nível baixo, e o modo DMA concorrente, no qual o TMS320C25 continua a executar o programa enquanto opera com a RAM e ROM internas, aumentando enormemente a capacidade de saída em aplicações intensivas.

5.6 Sumário do Hardware Interno

O hardware interno do TMS320C25 implementa funções que os outros microprocessadores realizam por software ou microcódigo. Por exemplo, o dispositivo contém um hardware para multiplicação de 16x16 bits em único ciclo, deslocamento de dados e manipulação de endereços. Esta característica

provê uma potência computacional anteriormente não disponível em um simples chip.

A Tabela 5-1 apresenta um sumário do hardware interno, registradores e barramento do TMS320C2x. Está em ordem alfabética por símbolo. Todos os símbolos usados nesta tabela correspondem aos símbolos usados no restante deste documento.

SÍMBOLO	FUNÇÃO
ACC(31-0) ACCH(31-16) ACCL(15-0)	Acumulador de 32 bits dividido em duas metades: ACCH(acumulador alto) e ACCL(acumulador baixo).Usado para armazenar a saída do ALU
ALU	Unidade lógica e aritmética de 32 bit's em complemento de dois tendo duas portas de entrada de 32 bit's e uma porta de saída de 32 bit's alimentando o acumulador
ARAU	Unidade Aritmética de 16 bit's sem sinals usados para realizar operações no registrador de dados auxiliares
AR0-AR7 (15-0)	Registradores Auxiliares de 16 bits usados para endereçamento da memória de dados, armazenamento temporário ou processamento de aritmética inteira através da ARAU
ARP(2-0)	Apontador dos Registradores Auxiliares: Um registrador de 3 bits usado para selecionar um dos oito registradores auxiliares
ARB(2-0)	Buffer do Apontador dos Registradores Auxiliares: Um registrador de 3 bits usado como buffer do ARP. Cada vez que o ARP é carregado, o seu valor antigo é escrito no ARB, exceto durante uma instrução LST (*load status register*). Quando o ARB é carregado com uma instrução LST1, o mesmo valor é copiado para o ARP.
CALU	Unidade Central de Aritmética e Lógica O conjunto de ALU, Multiplicador, Acumulador e Shifter
DP(8-0)	Apontador de Página de Memória de Dados: Um registrador de 9 bit's apontando o endereço de página corrente. Páginas de dados são formadas por 128 palavras cada, resultando em 512 páginas de espaço de memória endereçável (algumas alocações são reservados)
GREG(7-0)	Registrador de Alocação Global de Memória de 8 bits mapeado em memória para alocação global do espaço global de memória
IR(15-0)	Registrador de Instrução de 16 bits usado para armazenar a instrução corrente
IFR(5-0)	Registrador de Flag de Interrupção de 6 bits usado como latch das interrupções externas INT(2-0) e as interrupções internas XINT/RINT (transmissão /

	recepção da porta serial) e interrupção TINT(timer). IFR não é acessível por software.
IMR(5-0)	Registrador de Interrupção Mascaráveis: Registrador mapeado em memória usado para mascarar as interrupções
MULT	Multiplicador paralelo de 16x16 bits
PRD(15-0)	Registrador de Período de 16 bits mapeado em memória para recarregar o Timer
PR(31-0)	Registrador de Produto de 32 bits usado para manter o produto de uma multiplicação
PC(15-0)	Contador de Programa com 16 bits usados para endereçar a memória de programas. O PC sempre contém o endereço da próxima instrução a ser executada. O conteúdo do PC é atualizado após a operação de decodificação de cada instrução.
ST0, ST1 (15-0)	Registradores de Status: Dois registradores de 16 bits que contém os bits de status e controle
TIM(15-0)	Timer: Um timer de 16 bits mapeados em memória para controle do timer

Tabela 5-1 Hardware interno do TMS320C2x

5.7 Organização da Memória

O processador TMS320C25, como visto na Figura 5-1, possui um total de 544 palavras de 16 bits na memória interna de dados RAM, dos quais 288 palavras são sempre memória de dados e o restante das 256 palavras são sempre configuradas como memória de dados ou programa. O TMS320C25 também posui 4k palavras na memória ROM de programas. Nesta seção explicaremos o gerenciamento das memórias interna de dados e programas, mapeamento da memória, registradores mapeados em memória, registradores auxiliares, modos de endereçamento da memória e movimentos de memória para memória.

5.7.1 Memória de Dados

As 544 palavras internas na RAM de dados são divididas em três blocos: B0, B1 e B2. As 256 palavras do Bloco B0 são configuráveis como memória de programa ou memória de dados pelas instruções para este propósito; as 288 palavras Blocos B1 e B2 são sempre memórias de dados. Uma memória de

dados com 544 palavras permite ao TMS320C25 guardar um vetor de 512 palavras (256 palavras se o Bloco B0 estiver configurado como memória de programa), enquanto ainda mantém 32 localizações para armazenamento intermediário.

O processador TMS320C25 pode endereçar um total de 64k palavras de memória de dados. A memória de dados interna e as localizações internas reservadas são mapeadas no espaço de memória mais baixo de 1k palavras. Pode-se expandir diretamente a memória de dados para 64k palavras mantendo-se a velocidade máxima de operação. O TMS320C25 pode ser interfaceado com memórias mais lentas, e mais baratas, como por exemplo DRAM.

5.7.2 Memória de Programa

Programas armazenados na ROM e RAM interna, ou memórias de programa de alta velocidade podem ser usadas na velocidade máxima sem "wait states". Alternativamente, pode-se interfacear o TMS320C25 com memórias mais lentas. Um total de 64k palavras de espaço de memória é disponível. O bloco de memória interna B0 pode ser configurado como memória de programa usando instruções para este propósito.

Além disto, o TMS320C25 possui uma ROM interna de 4k palavras. A ROM interna permite a execução do programa a velocidade máxima sem a necessidade de memórias externas de alta velocidade. O uso desta memória também permite que o barramento externo de dados possa ser liberado para acessar a memória de dados externa.

5.7.3 Mapa de Memória

Vimos que o TMS320C25 tem duas estruturas internas de memória: uma para instruções de programa, usualmente uma ROM, e o outra para dados, usualmente uma RAM. Isto permite uma implementação mais eficiente de algoritmos que trabalham intensamente com dados.

O TMS320C25 tem uma capacidade de endereçamento de 64k palavras para dados e 64k palavras para programas. Internamente há 4k posições de memória para programa em ROM ou EPROM e 544 palavras de 16 bits na memória RAM interna de dados, sendo 256 posições selecionáveis como recursos de dados ou programas. O restante das 288 posições na RAM são apenas para dados.

O TMS320C25 pode trabalhar tanto como microcomputador, quanto microcontrolador, utilizando a ROM interna.

A memória externa pode ser ampliada se desejado pelo usuário. Ambos os espaços de dados e programas podem ser uma combinação de RAM e ROM. Adicionalmente, as memórias externas podem ter diferentes velocidades, pois o C25 pode operar com *"wait-state"*. Desta forma memórias rápidas (e caras) e memórias lentas (geralmente baratas) podem ser utilizadas de acordo com as necessidades do projeto.

A memória RAM interna é dividida em dois blocos de 256 palavras de 16 bits (bloco B0 e bloco B1) e o bloco B2 de 32 palavras de 16 bits.

Na Figura 5-4 e Figura 5-5, observamos que o bloco B0, pode estar no espaço de dados ou no espaço de programa. Desde que há somente um bloco B0 na RAM, este bloco pode estar somente em uma dada localização em um determinado tempo ou seja, ou no espaço de memória de dados, ou no espaço de memória de programas. No entanto pode ser chaveado (com seu conteúdo intacto) a qualquer tempo usando os comandos próprios para isto.

Somente a versão com ROM/EPROM é mostrado na Figura 5-4 e Figura 5-5. A versão com memória externa (modo de microprocessador) será a mesma, com a área de 0 a 1000h passando a ser uma memória externa, com as localizações de 0 a 20h reservada para os vetores de interrupção.

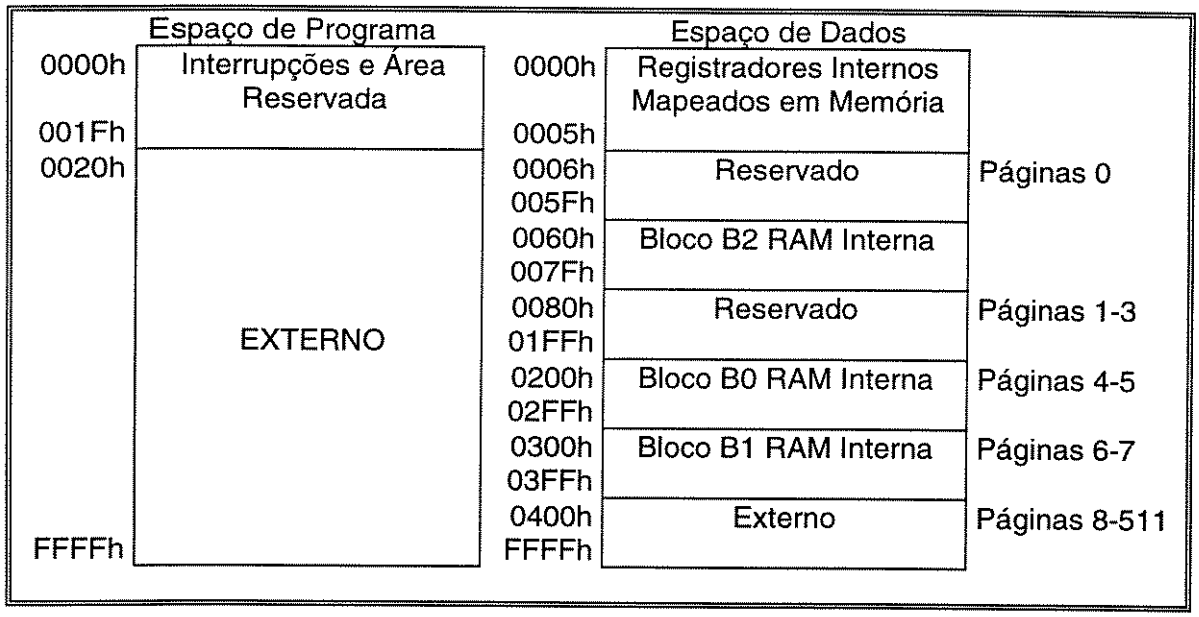


Figura 5-4 Mapa de memória do TMS320C25 - Bloco B0 configurado como área de dados

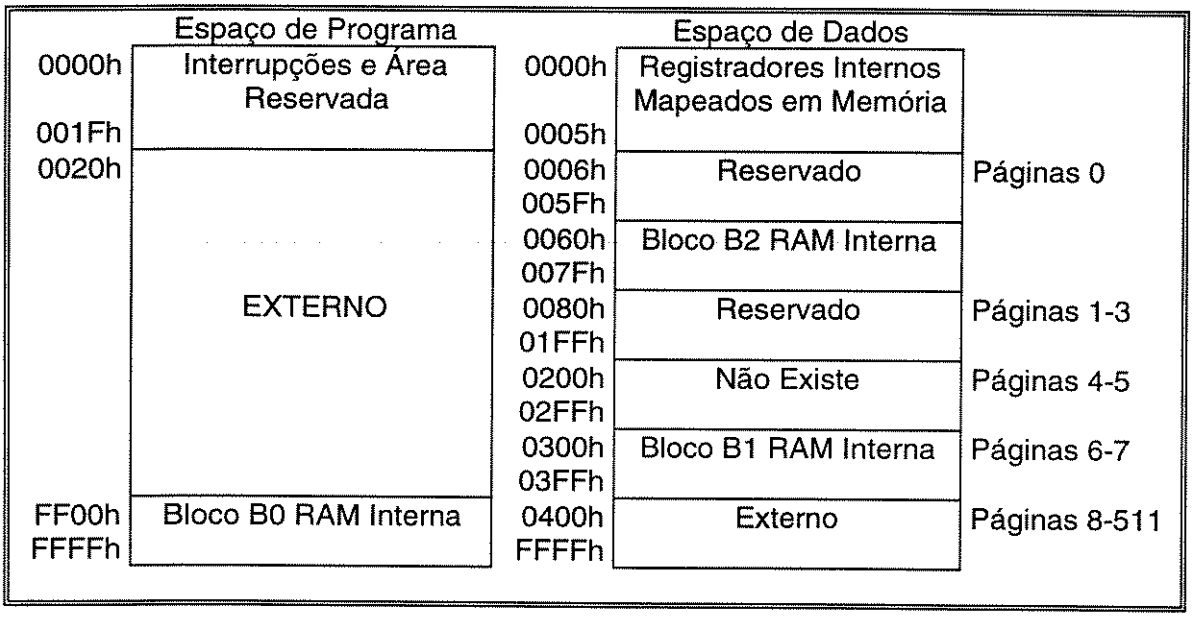


Figura 5-5 Mapa de memória do TMS320C25 - Bloco B0 configurado como área de programa

5.8 Registradores Mapeados na Memória

O TMS320C25 possui seis registradores mapeados no espaço de memória de dados, os quais são listado na Tabela 5-2 e são mostrados Figura 5-3.

Nome do	Endereço de	Definição

registrador	localização	
DRR(15-0)	0	Registro de dado recebido pela porta serial
DXR(15-0)	1	Registro de dado transmitido pela porta serial
TIM(15-0)	2	Registro TIMER
PRD(15-0)	3	Registro de período
IMR(5-0)	4	Registro de máscara de interrupção
GREG(7-0)	5	Registro de alocação global de memória

Tabela 5-2 Registradores mapeados na memória de dados

Os registradores mapeados na memória podem ser acessados da mesma maneira que qualquer posição na memória de dados, com a exceção de que os movimentos de blocos usando a instrução BLKD (block move from data memory to data memory) não podem ser realizados a partir dos registradores mapeados em memória.

Os registradores de interrupção são também mapeados em memória.

Nome da Interrupção	Localização de Memória	Prioridade	Função
R3	0	1	Sinal Externo de Reset
INT0	2	2	Interrupção Externa 0
INT1	4	3	Interrupção Externa 1
INT2	6	4	Interrupção Externa 2
	8-17		Área Reservada
TINT	18	5	Interrupção Interna do Timer
RINT	1A	6	Interrupção de recepção da porta serial
XINT	1C	7	Interrupção de transmissão da porta serial
TRAP	1E	N/A	Endereço da instrução TRAP

Tabela 5-3 Interrupções

Obs.: Prioridade 1 é a mais importante

Localização na Memória de Programa

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMR	Reservado										XINT	RINT	TINT	INT2	INT1	INT0

Tabela 5-4 Registrador de Máscara de Interrupção

5.9 Registradores Auxiliares

O TMS320C25 possui oito registradores auxiliares, que podem ser usados para o endereçamento indireto da memória de dados ou para armazenamento temporário de dados. O endereçamento indireto com registrador auxiliar permite que um endereço de um operando na memória de dados seja colocado em um dos registradores auxiliares. Estes registradores são apontados por um apontador de registrador auxiliar de três bits (ARP) que é carregado com um valor de 0 a 7 designando AR0 a AR7, respectivamente. Os registradores auxiliares e o ARP podem ser carregados a partir da memória de dados ou por um operando imediato definido na instrução. O conteúdo destes registradores também podem ser armazenados na memória de dados.

O arquivo de registradores auxiliares é conectado à Unidade Aritmética dos Registradores Auxiliares (ARAU). A ARAU pode auto-indexar o registrador auxiliar corrente, enquanto a localização da memória de dados está sendo endereçada. Como resultado disto, acessar tabelas de informações não requer o uso da Unidade Central de Lógica e Aritmética (CALU) para a manipulação de endereço, liberando-a assim para outras operações.

O registrador auxiliar AR0 ou os oito bits menos significativos (LSB's) do registrador de instrução pode ser conectado a uma das entradas da ARAU. A outra entrada da ARAU é alimentada pelo AR corrente (sendo apontada pelo ARP). AR(ARP) refere-se ao conteúdo do AR corrente apontado pelo ARP. O ARAU realiza as seguintes funções:

AR(ARP)+AR0→AR(ARP)	Indexa o AR corrente pela adição do conteúdo de AR0
AR(ARP)-AR0→AR(ARP)	Indexa o AR corrente pela subtração do conteúdo de AR0
AR(ARP)+1→AR(ARP)	Incrementa o AR corrente de 1
AR(ARP)-1→AR(ARP)	Decrementa o AR corrente de 1
AR(ARP)→AR(ARP)	Não modifica o AR corrente
AR(ARP)+IR(7-0)→AR(ARP)	Soma um valor de 8 bits imediato ao AR corrente

AR(ARP)-IR(7-0)→AR(ARP)	Subtrai um valor de 8 bits imediato ao AR corrente
AR(ARP)+rcAR0→AR(ARP)	Indexação Bit-Reversed, soma AR0 com propagação reversa do carry (rc) ao AR corrente
AR(ARP)-rcAR0→AR(ARP)	Indexação Bit-Reversed, subtrai AR0 com propagação reversa do carry (rc) ao AR corrente

Embora a ARAU seja útil para a manipulação de endereços em paralelo com outras operações, ela pode servir como uma unidade aritmética de uso geral, uma vez que o conjunto de registradores auxiliares pode se comunicar diretamente com a memória de dados. A ARAU implementa uma aritmética em 16 bits, e a CALU implementa uma aritmética em complemento de dois de 32 bits. A instrução BANZ permite que os registradores auxiliares sejam também usados como contadores de loop. A comparação do Registrador Auxiliar apontado pelo ARP com o registrador AR0 permite um desvio na execução do programa. Isto é feito com a instrução CMPR <constante CM> (*Compare Auxiliary Register with Auxiliary Register AR0*).

CM	Valor
00	AR(ARP)= AR0
01	AR(ARP)<AR0
10	AR(ARP)>AR0

Tabela 5-5 Valores para comparação

5.10 Registradores de Status

O TMS320C25 possui dois registradores de status: ST0 e ST1. Os registradores de status contém os bits de controle e de status do processador. Os registradores de status podem ser armazenados/carregados na/da memória de dados, permitindo assim que o status da máquina possa ser armazenado/carregado em interrupções e subrotinas. O registrador ST0 armazena dois apontadores: o apontador de registros auxiliares (ARP) e o apontador de página de memória (DP). O ARP seleciona um dos oito registradores auxiliares e o DP seleciona a página de memória de dados ativa.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST0	ARP			OV	OVM	1	INTM	DP								
ST1	ARB			CNF	TC	SXM	C	1	1	HM	FSM	XF	FO	TXM	PM	

Tabela 5-6 Registradores de status

ARP(2-0)	Apontador dos Registradores Auxiliares: Um registrador de 3 bits usado para selecionar um dos oito registradores auxiliares
ARB(2-0)	Buffer do Apontador dos Registradores Auxiliares: Um registrador de 3 bits usado como buffer do ARP. Cada vez que o ARP é carregado, o seu valor antigo é escrito no ARB, exceto durante uma instrução LST (*load status register*). Quando o ARB é carregado com uma instrução LST1, o mesmo valor é copiado para o ARP.
C	Bit de carry. Este bit é setado para 1, se o resultado de uma adição gera um carry ou resetado para 0 se o resultado de uma subtração gera um borrow. Caso contrario ele é restado após uma adição ou setado após uma subtração, exceto nas instruções ADDH ou SUBH. ADDH pode somente setar o bit de carry e SUBH pode somente inicializá-lo, mas não o afetam caso contrário. As instruções de Shift e rotate também afetam este bit, bem como as instruções SC, RC e LST1. Duas instruções de desvio condicionais, BC e BNC, são utilizadas para desvio em função de C. C é setado para 1 no reset.
CNF	Configuração da RAM interna. Se setado para 0, B0 é configurado como memória de dados, caso contrário B0 é memória de programa.. CNF pode ser modificado por CNFP, CNFD e LST1
DP	Apontador da memória de dados. Os 9 bits do registrador DP são contatenados com os 7 LSBs da palavra de instrução para formar um endereço de 16 bits. DP pode ser modificado pelas instruções LST, LDP e LDPK.
INTM	Bit do modo de interrupção. Quando setados para 0, todas as interrupções não mascaráveis são habilitadas, caso contrário são desabilitadas. É setado/resetado pelas instruções DINT/EINT. Não é afetado pela instrução LST.
OV	Bit de Flag de Overflow. Funcionando como uma trava do sinal de overflow, é setado para um quando um overflow ocorre na ALU. OV permanece em 1 até um reset ou uma instrução BV, BNV ou LST reseta-lo.

OVM	Bit de modo de overflow. Quando setado para zero, overflow ocorrem no acumulador sem afetar o resultado. Quando setado para um no entanto, se um overflow ocorre o acumulador satura no valor mais positivo ou mais negativo dependendo do sentido do overflow. As instruções SOVM/ROVM setam/resetam o bits OVM respectivamente
PM	Modo do deslocamento do produto. Este registrador é explicado na seção Multiplicador, registradores T e P.
SXM	Modo da extensão de sinal. SXM=1 produz a extensão de sinal no dado quando ele é passado para dentro do acumulador através do scaling shifter. SXM=0 suspende a extensão de sinal.
TC	Bit de teste/Controle. TC é setado para um se um bit testado por BIT ou BITT for um 1, se uma condição de comparação testada por CMPR existente entre AR0 e outro AR apontado por ARP ou se uma função e ou-exclusivo dos dois MSBs do acumulador for verdadeira quando testado pela instrução NORM. Duas instruções de desvio, BBZ e BBNZ, permitem desvios em função do status de TC.
FO	Bit de formato. Quando setado para o, os registradores da porta serial são configurados como registradores de 16 bits. Quando setados para 1, os registradores da porta são configurados para receber e transmitir oito bits. FO pode ser modificado pelas instruções FORT e LST1. No reset, FO=0
FSM	Frame Syncronization Mode Bit
HM	Hold Mode Bit
TXM	Bit do Modo de Transmissão
XF	Bit de Status do pino XF

Tabela 5-7 Definição dos campos dos registradores de status

5.11 Modos de Endereçamento da Memória

O processador TMS320C25 pode endereçar um total de 64k palavras da memória de programa e 64k palavras da memória de dados. A memória de dados interna é mapeada no espaço de memória de dados de 64k. A ROM interna do TMS320C25 é mapeada dentro do espaço de memória de programa

quando no modo de microcomputador. O mapa de memória, que muda com a configuração do Bloco B0, foi descrito na seção anterior.

O conjunto de instruções do TMS320C2x proporciona o endereçamento da memória de três modos:

- Modo de endereçamento direto usando o barramento de endereço direto (DRB - *Data Direct Bus*);
- Modo de endereçamento indireto usando o barramento do arquivo de registradores auxiliares (AFB - *Auxialiary Register File Bus*);
- Modo de endereçamento imediato, onde os operandos são também endereçados pelo conteúdo do contador de programa (PC - *Program Counter*).

5.11.1 Modo de Endereçamento Direto

O endereçamento Direto é um meio simples e comum de retirar e colocar dados na memória. Acessando cada localização de memória de dados através de seu endereço de 0 a 64k, nós podemos usar cada localização de memória de forma independente. Para identificar uma localização nos 64k precisamos de 16 bits.

No modo de endereçamento direto, os 9 bits do apontador de página da memória de dados (DP - *Data memory Page Pointer*), apontam para uma das 512 páginas, cada página consistindo de 128 palavras. Estes 9 bits são inicializados com o valor desejado de página, e devem ser mudados se outra página for requerida. Isto representa um pequeno gasto computacional, assumindo que o programa localiza os operandos para uma dada rotina na mesma página. Com 128 localizações na mesma página, isto, usualmente, não é difícil de ser feito. Este esquema de endereçamento por páginas significa que com o endereçamento direto, o programador refere-se ao endereço 00h-7Fh nas páginas 000h-1FFh, no lugar dos endereços 0000h-FFFFh.

O endereço da memória de dados (dma - data memory address), especificado pelos 7 LSB's do Registrador de Instrução (IR), aponta para a palavra desejada

naquela página. O endereço no barramento de endereço direto (DRB - Data Direct Bus) é formado concatenando-se os 9 bits de DP com os 7 bits do dma.

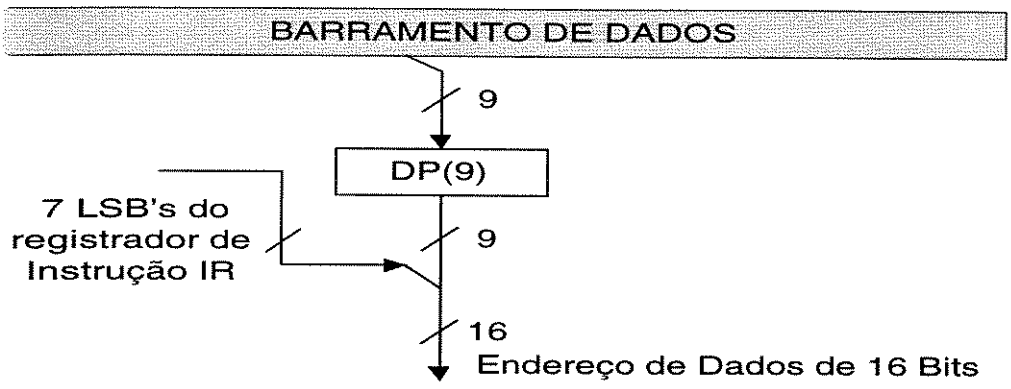


Figura 5-6 Diagrama de blocos do modo de endereçamento direto

5.11.2 Modo de Endereçamento Indireto

O endereçamento indireto é a mais poderosa maneira de se trabalhar com vetores de dados. Diferentemente do endereçamento direto, o endereço não é expresso na palavra da instrução, mas é localizado no Registrador Auxiliar (AR - Auxiliary Register). O uso do AR provê alguns benefícios. O AR é um registrador de 16 bits, assim não é necessário o uso de endereçamento por páginas. Mais importante ainda é o fato de que o AR pode ser automaticamente incrementado após um operando ter sido achado. O uso desta característica torna rápido e fácil o processamento interativos de matrizes.

No modo de endereçamento indireto, o registrador auxiliar de 16 bits selecionado (AR(ARP)) endereça a memória de dados através do barramento de arquivo de registradores auxiliares (AFB - Auxialiary register File Bus). Enquanto o registrador auxiliar selecionado provê o endereço na memória de dados, e o dado é manipulado na CALU, o conteúdo do registrador auxiliar pode ser manipulado através da ARAU.

Há oito Registradores Auxiliares no TMS320C25. Para o endereçamento indireto ser usado, o AR deve ser inicializado e selecionado prioritariamente

para ser usado. Os AR's são inicializados com a instrução LARK (*Load Auxiliary Register with Constant*). Seleciona-se o AR por dois métodos:

(1) Durante a inicialização, a instrução LARP (*Load Auxiliary Register Pointer*) é usada;

(2) Nas instruções subsequentes, a instrução LARP pode ser embutida em alguma instrução usando um endereçamento indireto.

Os 7 bits menos significativos da instrução (que mantinham o endereço no endereçamento direto) são usados para especificar as opções para endereçamento indireto. Assim uma simples instrução é capaz de achar o operando, incrementando o AR corrente, e selecionando o próximo AR.

Há três campos principais nas instruções de endereçamento indireto que afetam o acumulador e dois campos de operando para a maioria das outras instruções. A opção de Shift nas operações do acumulador é feita por um operando extra.

5.11.2.1 Ordem de execução do Endereçamento Indireto

Exemplo: `ADD *+,1,0`

1. `*` ARP seleciona o AR para manter o operando do dma
2. `1` O operando é deslocado a esquerda 1 bit - 0 é o valor opcional default
3. `ADD` Realiza a operação com o operando já deslocado
4. `+` Modifica o valor de AR (opcional)
5. `0` Novo valor de AP (opcional)

Os Registradores Auxiliares permitem vários tipos de opções de modificações para suportar o enorme número de aplicações do TMS320C25. O AR pode não ser modificado, incrementado ou decremento (por um) após ter sido usado. Desta maneira, muitas aplicações são possíveis, pois uma aplicação comum é o processo de movimentação de um elemento por vez dentro de um vetor. Há aplicações onde necessitamos incrementar/decrementar o AR com um valor de passo diferente de um. Para estas aplicações, o programador necessita incrementar/decrementar o valor de AR0, usando qualquer outro registrador

auxiliar como um ponteiro de dados, e selecionar a opção de endereçamento "0+"ou "0-". Em um único ciclo, o endereço corrente é acessado e o AR é modificado após a operação corrente ser modificada.

OP *(,Novo ARP)	Nenhuma mudança no AR corrente
OP *+(,Novo ARP)	O AR corrente é incrementado
OP *-(,Novo ARP)	O AR corrente é incrementado
OP *0+(,Novo ARP)	AR0 é somado ao AR corrente
OP *0-(,Novo ARP)	AR0 é decrementado do AR corrente
OP *BR0+(,Novo ARP)	AR0 é somado ao AR corrente, com propagação reversa do carry
OP *BR0-(,Novo ARP)	AR0 é decrementado do AR corrente, com propagação reversa do carry

Tabela 5-8 Opções de endereçamento indireto no TMS320C25

5.11.3 Modo de Endereçamento Imediato

No modo de endereçamento imediato, a instrução contêm o valor do operando imediato. O TMS320 tem instruções imediatas curtas de uma única palavra (constantes de 8 e 13 bits) e instruções imediatas longas de palavra dupla (constantes de 16 bits). O operando imediato está contido dentro da palavra de instrução nas instruções imediatas curtas.

Nas instruções imediatas longas, a palavra seguinte ao opcode da instrução é usada como operando imediato. O endereçamento imediato permite o uso de constantes e são na maioria das vezes usados para inicialização de posicões de memória.

As seguintes instruções curtas imediatas contém o operando imediato na palavra de instrução e são executadas em um único ciclo de instrução. O comprimento do operando na palavra é dependente da instrução.

ADDK	Somar ao Acumulador (Constante absoluta de 8 bits)
ADRK	Somar ao Registrador Auxiliar (Constante absoluta de 8 bits)

LACK	Carregar o Acumulador (Constante absoluta de 8 bits)
LARK	Carregar o Registrador Auxiliar (Constante absoluta de 8 bits)
LARP	Carregar o Apontador de Registros (ARP) (Constante de 3 bits)
LDPK	Carregar o Apontador de Página de Dados (Constante de 9 bits)
MPYK	Multiplique (Constante de 13 bits em complemento de dois)
RPTK	Instrução de repetição como especificado pelo valor imediato (Constante de 8 bits)
SBRK	Subtrair do Registrador Auxiliar (Constante absoluta de 8 bits)
SUBK	Subtrair do acumulador (Constante absoluta de 8 bits)
ZAC	Zera o acumulador

Tabela 5-9 Instruções imediatas curtas

5.11.3.1 Exemplo do Formato de Endereçamento Curto:

`RPTK 99` Executa a instrução seguinte 100 vezes

Note que a área de dados não é envolvida neste tipo de instrução. Uma verdadeira máquina de Harvard trabalha somente com o espaço de dados. Esta capacidade de passar operandos da área de programa é uma das razões do TMS320C25 ser chamado de arquitetura Harvard Modificada.

5.11.4 Resumo dos Modos de Endereçamento

Esta seção descreve e compara cada método de endereçamaneto. Os vários modos tem aparência similar e podem ser facilmente confundidos.

5.11.4.1 Endereçamento Direto

- 512 "páginas" de 128 palavras
- Usado geralmente por longas seqüências de linha de código

DP		Da instrução		Endereço do operando
9 bits (MSB)	+	7 bits (LSB)	=	16 bits

5.11.4.2 Endereçamento Indireto

- Oito registro auxiliares
- Usado geralmente em loops de programas com a opção incrementar / decrementar

5.11.4.3 Endereçamento Imediato

- Constantes longas e curtas
- Exemplo:

Instrução	Memória de Programa	Tipo
ADDK 5	ADDK 5	Instrução Curta
ADLK 1325h	ADLK 1325h	Instrução Longa

5.11.5 Modo de Endereçamento Bit-Reversed

A última forma para endereçamento no TMS320C25 é chamada de Bit-Reversed. É um meio muito eficiente para endereçamento no cálculo da Transformada Rápida de Fourier. A FFT é um processo para conversão da informação no domínio do tempo para a informação no domínio da freqüência, e se baseia na regra da "Borboleta" como núcleo do processamento matemático.

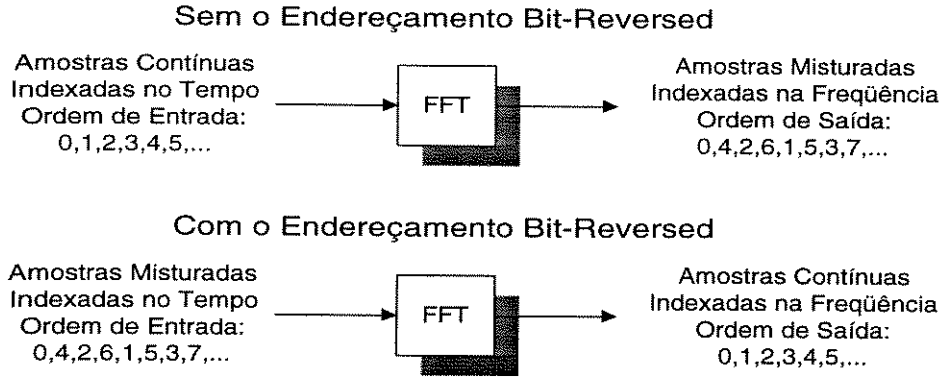


Figura 5-7 Modo de endereçamento Bit-Reversed

Note pela Figura 5-7 que sem o endereçamento Bit-Reversed, o processo faz com que a saída da FFT esteja fora de seqüência. Reordenar a matriz pode requerer um longo tempo de processamento , ou o uso de lógica externa. O TMS320C25 tem uma forma de endereçamento, o qual elimina a necessidade de corrigir o endereço, colocando os resultados fora de ordem na sua posição correta de memória durante o processo de FFT. Assim, nenhum tempo extra é necessário para se reordenar a matriz, permitindo uma implementação de FFT muito mais eficiente. Para entender melhor este modo de endereçamento, observe a ordem de saída com seus índices em forma binária. A princípio, parecem números aleatórios, mas há uma sequencia crescente se os números são lidos de trás para frente ou seja, do bit menos significativo para o bit mais significativo, na ordem reversa dos bits: Esta operação de endereçamento, algumas vezes chamada de propagação reversa de carry é muito mais que uma imagem no espelho da ordem normal de endereçamento. O endereçamento Bit-Reversed como usado no TMS320C25 atua como se o endereço reverso gerado pela FFT fosse colocado em um espelho (obtendo assim uma imagem invertida), e aí é lido normalmente. A implementação desta imagem espelhada duas vezes requer que um valor de contagem do vetor seja colocado em AR0, e que qualquer outro AR seja usado como um registrador de endereçamento, com o modo BR+ ou BR- selecionado. O valor de contagem do vetor é simplesmente um 1 na posição correta para incrementar o processo. O endereçamento Bit-Revesed acessa as amostras em uma ordem que assegura uma ordem contínua das amostras em freqüência.

5.12 Movimento de Memória para Memória

O TMS320C2x possuí instruções para movimento de blocos dos dados e programas e funções de movimentos dos dados que utilizam eficazmente a RAM interna configurável.

A instrução BLKD move um bloco na memória de dados, e a instrução BLKP move um bloco de memória de programa para a memória de dados. Quando usada com a instrução (RPT/RPTK), as instruções BLKP/BLKD realizam eficazmente movimento de blocos para memórias internas e externas do chip.

A função DMOV (*data move*), a qual só endereça a memória RAM interna, permite que uma palavra seja copiada do endereço corrente da memória de dados interna para a próxima posição mais alta, enquanto o dado da localização endereçada é operada sobre o mesmo ciclo (e.g., pela CALU). Uma operação na ARAU pode também ser realizada no mesmo ciclo quando utilizada uma instrução de endereçamento indireto. A função DMOV é útil para a implementação de algoritmos que utilizam a operação Z^{-1} de atraso, como nas convoluções e nos filtros digitais. A função de movimento de dado pode ser usada em qualquer lugar dos blocos B0, B1 e B2. Ela é contínua nos limites entre B0 e B1 mas não pode ser usada com a memória externa do chip. As instruções multiplicar/acumular com movimento do dado MACD (*multiply e accumulate with data move*) e carregar registrador T, acumular o produto anterior com movimento do dado - LTD (*load T register, accumulate previous product, and data move*)

As instruções TBLR/TBLW (*table read/write*) permite que palavras sejam transferidas entre os espaço de programa e espaço de dados. TBLR é usado para ler palavras da memória ROM interna ou da memória de programa ROM/RAM externa para a RAM de dados. TBLW é usada para escrever palavras da RAM de dados para escrever palavras da memória de dados RAM interna para a RAM de programa externa. Pode ainda ser usada para escrever a palavra no Bloco B0 quando este estiver configurado como RAM de programa.

5.13 Unidade Central de Lógica e Aritmética - CALU

A Unidade Central de Lógica e Aritmética doTMS320C25 contém um *"shifter"* escalonador de 16 bits, um multiplicador paralelo de 16 x16 bits, uma unidade de Lógica e Aritmética de 32 bits, um acumulador de 32 bits e shifters adicionais na saída do acumulador e do multiplicador.

Os seguintes passos ocorrem na implementação de uma típica instrução na ALU:

1) O dado é buscado da RAM no barramento de dados
2) O dado é passado através do shifter de escalonamento e da ALU onde a aritmética é realizada e
3) O resultado é movido para o acumulador.

Uma entrada da ALU é sempre provida pelo acumulador e a outra entrada pode ser transferida do Registrador de Produto (PR) do multiplicador ou do shifter de escalonamento o qual é carregado a partir da memória de dados.

5.13.1 O Multiplicador, Registradores T e P

O TMS320C25 utiliza um multiplicador de 16 x16 bits que é capaz de calcular um produto de 32 bits, com ou sem sinal em um único ciclo de máquina. Todas as instruções de multiplicação, exceto a instrução MPYU (*Multiply Unsigned*), realizam uma operação de multiplicação com sinal no multiplicando. Isto é, os dois números são tratados como números com sinal e o resultado também será um número de 32 bits com sinal. O multiplicador possui dois registradores associados a ele:

- Um registrador temporário (TR) de 16 bits que guarda um dos operandos a multiplicar,
- Um registrador de produto de 32 bits que guarda o produto.

A saída do registrador de produto pode ser deslocada de 1 a 4 bits. Isto é útil para implementarmos aritmética fracionária ou ajustar produtos fracionários. A

saída do registrador P também pode ser deslocada 6 bits a direita para permitir a execução de 128 operações de multiplicar/acumular sem a possibilidade de overflow.

Uma instrução LT(*load T register*) normalmente carrega o TR para prover um operando (do barramento de dados), e uma instrução MPY (*multiply*) provê o segundo operando (também do barramento de dados). Uma multiplicação pode ser realizada com um operando imediato usando a instrução MPYK. Em ambos os casos o produto pode ser obtido sempre em dois ciclos.

Duas instruções multiplicar/acumular (MAC e MACD) utilizam toda a capacidade computacional permitindo que ambas as operações, multiplicar e acumular, sejam realizadas simultaneamente. Os operandos podem estar em qualquer lugar da memória interna ou externa, ou podem ser transferidas para o multiplicador em um ciclo, via os barramentos de dados e programa. Isto permite multiplicar/acumular em um único ciclo, quando usado com as instruções (RPT/RPTK). Observe que a parte de deslocamento de dados da instrução MACD não funciona com a memória externa de dados. No TMS320C25 as instruções MAC e MACD podem ser usadas com os operandos da memória interna ou externa. As instruções SQRA (*square/add*) e SQRS (*square/subtract*) passam o mesmo valor para ambas as entradas do multiplicador para elevar ao quadrado um valor na memória de dados.

A instrução MPYU no TMS320C25 realiza um multiplicação sem sinal. O conteúdo sem sinal do registrador T é multiplicado pelo conteúdo sem sinal de uma localização na memória de dados, com o resultado armazenado no registrador P. Isto permite que números com mais de 16 bits sejam quebrados em palavras de 16 bits e processados separadamente para gerar produtos com mais de 32 bits.

Após a multiplicação dos dois números de 16 bits, o produto de 32 bits é armazenado no Registrador de Produto (PR) no TMS320C2x. O produto no registrador PR pode ser transferido para a ALU.

Quatro modos de deslocamento do produto (PM-*Product Shift Modes*) são disponíveis para a saída do registrador de produto (PR). Estes produtos são úteis quando realizamos operações de multiplicar/acumular, aritmética fracionária, ou ajuste de produtos fracionários. O campo PM do registrador de status ST1 especifica o deslocamento no modo PM.

Se PM é	Resultado é
00	Nenhum Shift
01	Shift a esquerda 1 Bit
10	Shift a esquerda 4 bits
11	Shift a direita 6 bits

Tabela 5-10 Valores do shift no registrador P

Deslocamentos a esquerda especificados pelo valor de PM são úteis para a implementação de aritmética fracionária ou ajuste de produtos fracionários. Por exemplo, o produto de dois números normalizados de 16 bits, números em complemento de dois ou dois números Q15 contém dois bits de sinais, um dos quais é redundante. O formato Q15, um dos vários tipos do formato Q, é uma representação numérica geralmente usada em operações com números não inteiros. O deslocamento de um bit para a esquerda elimina o bit redundante do produto quando este é transferido para o acumulador. Isto resulta em que o conteúdo do acumulador seja formatado da mesma maneira que os multiplicandos. Similarmente, o produto tanto de números normalizados, 16 bits, complemento de dois ou um número Q15 e uma constante de 13 bits com sinal em complemento de dois com 5 bits de sinal, quatro deles são redundantes. Este é o caso por exemplo, de quando usamos a instrução MPYK. Aqui o deslocamento de quatro bits alinha o resultado assim que este é transferido para o acumulador.

O uso do deslocamento de um bit à direita permite a execução de até 128 operações multiplicar/acumular consecutivas sem o perigo de um overflow aritmético, evitando a necessidade de gerenciamento de overflow. O deslocamento pode ser nulo no produto quando trabalhamos com operandos

inteiros ou de 32 bits de precisão. O deslocamento à direta é sempre com sinal independente do estado de SXM.

Os quatro bits menos significantivos do registrador T (TR) também permitem um deslocamento variável através do shifter escalonado para as instruções LACT/ADDT/SUBT (*load/add-to/subtract-from accumulator with shift specified by TR*). Estas instruções são úteis na aritmética de ponto flutuante onde um número precisa ser normalizado, isto é, conversão de ponto flutuante para ponto-fixo. A instrução BITT (*bit test*) permite o teste de um simples bit de uma palavra na memória de dados baseada no conteúdo dos quatro LSB's de TR.

5.14 Sistema de Controle

O sistema de controle no TMS320C25 é proporcionado pelo contador de programa, um stack interno no hadware, registradores de status, timer interno contador de repetição, interrupção e sinais externos.

5.14.1 Contador de Programa e Pilha

O TMS320C25 contém um contador de programas (PC - *Program Counter*) de 16 bits e uma pilha de 8 posições no hardware para armazenamento do PC. O contador de programas endereça a memória de programas interna/externa nas instruções de busca de instruções. A pilha é usada durante as interrupções e subrotinas.

O contador de programa endereça a memória de programa através do barramento de endereços de programa (PAB - *Program Adress Bus*). Através do PAB, uma instrução é buscada e carregada no Registrador de Instrução (IR - *Instruction Register*).

A memória de dados pode ser endereçada pelo contador de programa durante a instrução BLKD, que move blocos de dados de uma seção de memória para outra.

5.14.2 Controle do Programa

O TMS320 possui muitas instruções que controlam o fluxo do programa. Instruções de desvio permitem saltar para qualquer localização do mapa de memória. As instruções CALL e RET permitem saltar e voltar em subrotinas. Estas instruções podem ser condicionais - ocorrendo somente quando certas condições são satisfeitas, ou incondicionais - ocorrendo sempre. Quando as condições de desvio condicional ou CALL não são satisfeitas, a execução do programa continua em seqüência. Todas as instruções de desvio, exceto BACC, permitem modificar o Registrador Auxiliar se desejado.

5.14.3 Contador de Repetição

O contador de repetição (RPTC) é um contador de 8 bits, que, quando carregado com um número N, força a execução da próxima instrução simples N+1 vezes. O RPTC pode ser carregado com um número de 0 a 255 usando tanto a instrução RPT (*repeat*) como a instrução RPTK (*repeat immediate*). Isto resulta num máximo de 256 execuções da próxima instrução. RPTC é zerado com o reset do microprocessador.

A repetição pode ser usada com instruções como multiplicar/acumular (MAC/MACD), movimento de blocos (BLKD/BLKP), transferência de I/O (IN/OUT), e leitura/escrita de tabelas (TBLR/TBLK). Estas instruções, que são normalmente de múltiplos ciclos, são processadas em *"pipeline"* quando usada a característica de repetição, e se tornam instruções de um único ciclo. Por exemplo, a instrução de leitura de tabela pode levar três ou mais ciclos para ser realizada, mas quando no modo de repetição, uma localização de tabela pode ser lida a cada ciclo. Contudo, nem todas as instruções podem ser repetidas.

6. Implementação do Sintetizador de Formantes Utilizando Aritmética de Ponto Fixo

6.1 Introdução

O TMS320C25 é um microprocessador que executa operações de ponto fixo com operandos de 16/32 bits e não possui recursos explicitos de hardware para realizar operações em ponto flutuante. As operações de ponto flutuante são implementadas utilizando recursos de software (rotinas apropriadas). Estas operações são extremamente complexas e requerem uma série de operações matemáticas. Neste caso, o ganho é na precisão do resultado mas, o "preço a pagar" é a lentidão do processamento e o gasto de memória. Assim, por questão de velocidade/memória e arquitetura do sintetizador, este será implementado em ponto fixo. Contudo, uma série de inconvenientes ao se realizar uma implementação em ponto fixo surgem devido ao efeito do comprimento finito da palavra.

Antes de efetivamente implementarmos os elementos do Sintetizador de Klatt no TMS320C25, torna-se necessária uma pequena revisão sobre numeração binária, quantização e aritmética de ponto fixo.

6.2 Representação Numérica com o TMS320C25

Como visto no Capítulo 5, o TMS320C25 possui uma Unidade Central de Lógica e Aritmética (CALU) onde são realizadas as operações lógicas e matemáticas. Estas operações são realizadas em números binários, com ou sem sinal. A CALU contém um shifter escalonador de 16 bits, um multiplicador de 16 bits x 16 bits, uma unidade lógica aritmética de 32 bits,

um acumulador de 32 bits, e shifters adicionais na saída do multiplicador e do acumulador. Todos estes elementos tem um comprimento finito de bits (16 ou 32).

Um número binário inteiro (Ψ) possui a seguinte representação:

$$\Psi=b_0b_1b_2.b_3...b_{B-1}$$

onde B = número de bits, b_i são os valores iguais a 1 ou 0 do bit na posição i=0,1,...,N-1.

O processamento digital de sinais é uma técnica que geralmente é confundida, infelizmente, com o estudo de sistemas discretos ou de dados amostrados. Na análise teórica de sistemas discretos, filtros e transformações são definidos algebricamente. Geralmente assumimos que os valores das amostras dos sinais e dos coeficientes do sistema são representados em números reais. Quando implementamos sistemas de processamento digital de sinais, devemos representar sinais e coeficientes em algum sistema de representação numérica que sempre tem precisão finita. Um número real pode ser representado com precisão infinita no formato binário de complemento de dois, como:

$$x = X_m(-b_0 + \sum_{i=1}^{\infty} b_i 2^{-i}) \qquad \textbf{Equação 6-1}$$

onde X_m é um fator de escala arbitrário e representa a faixa dinâmica do valor x. Assim teremos, $-X_m \leq x \leq X_m$. A quantidade b_o é referida como bit de sinal. Se $b_o=0$ então $0 \leq x \leq X_m$ e se $b_o=1$, então $-X_m \leq x < 0$. Assim, qualquer número real cuja magnitude seja menor ou igual a X_m pode ser representado pela Equação 6-1. Um número real arbitrário x requer um número infinito de bits para uma representação binária exata. No entanto, como dispomos de um número B finito de bits, deveremos quantizar o valor de x ou seja, reduzir o número de bits para representá-lo. A Equação 6-1 deve ser modificada para:

$$\hat{x} = Q_B[x] = X_m(-b_0 + \sum_{i=1}^{B} b_i 2^{-i}) = X_m \hat{x}_B \qquad \textbf{Equação 6-2}$$

Devido ao efeito de quantização, os filtros digitais reais comportam-se diferentemente do esperado. A discordância entre a resposta de um filtro digital ideal e do modelo discreto é dita como *efeito do comprimento finito da palavra*. Os efeitos do comprimento finito da palavra são encontrados em muitas formas, mas podem ser caracterizados como:(Taylor)

1 Erros de Quantização;

2 Erros de Aritmética.

A menor diferença entre dois números quantizados é:

$$\Delta = X_m 2^{-B}$$

No caso em consideração, os números quantizados estão na faixa de $-X_m \le \hat{x} < X_m$ ou seja $-2^B \Delta \le \hat{x} \le (2^B - 1)\Delta$ e teremos 2^{B+1} níveis de quantização. A quantização é uma operação não linear, sem memória, cujo propósito é transformar um número binário de infinitos bits em um número binário com finitos bits. O formato Q_B representa o passo entre dois níveis consecutivos como 2^{-B}. Assim, no formato Q_3, por exemplo, $\Delta_3 = X_m 2^{-3}$.

A representação binária em complemento de dois é também conhecida como *aritmética circular*. Isto se deve ao fato de ao se somar Δ ao maior número positivo, $(2^B - 1)\Delta$, ou seja b_i=1 para i≠0, ocorrer um estouro da capacidade de representação e então teremos como resultado, o maior número negativo, ou seja, $-2^B \Delta$.

O comportamento circular pode ser representado por um anel binário. Por questão de simplicidade a Figura 6-1 apresenta um anel binário de 4 bits.

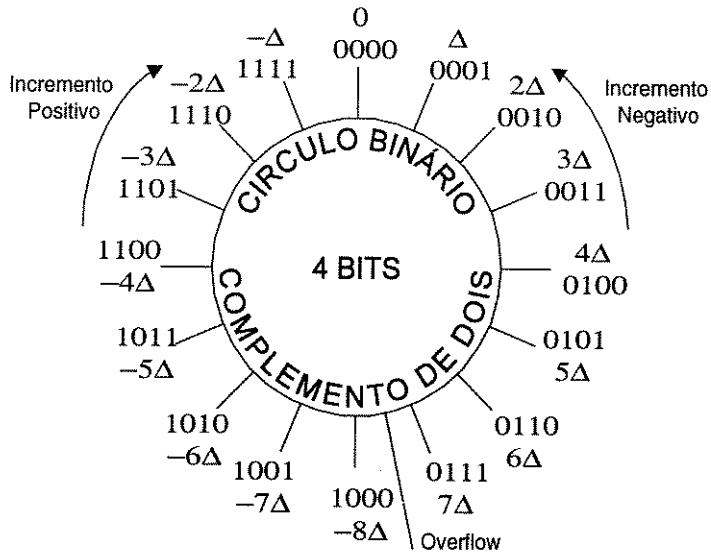


Figura 6-1 Círculo binário de 4 bits

Para operações de adição caminhamos no sentido horário e para operações de subtração caminhamos no sentido anti-horário. Ao passarmos de -Δ para 0, um *carry* (vai um) será gerado, e ao passarmos no sentido contrário, ou seja, de 0 para −Δ, um *borrow* (empresta um) será gerado. Somando-se Δ ao maior número positivo $(2^B-1)\Delta$, no lugar de $2^B\Delta$, teremos $-2^B\Delta$ como resultado, e um *overflow* será gerado. Um *overflow* nada mais é que o estouro da capacidade de representação de um número em B+1 bits, pois a faixa de representação como vimos é de $-2^B\Delta \le \hat{x} \le (2^B-1)\Delta$.

A vantagem em se usar o sistema de complemento de dois é que números podem ser subtraídos usando-se o mesmo método da adição. Assim, não é necessário nenhum hardware extra para se fazer a subtração. O complemento de dois é uma convenção do programador, não do hardware.

6.3 Quantização

A operação de quantização de um número x para (B+1) bits pode ser implementada por arredondamento ou por truncamento, mas em ambos os casos a quantização é uma operação não linear sem memória. Considerando-se o efeito da quantização, geralmente definimos o erro de quantização como:

$$e = Q_B[x] - x = \hat{x} - x$$

Truncamento é o processo de quantização em que somente os primeiros (B+1) bits (bits mais significativos, *MSB*) de x são usados para formar $\hat{x}$. Então a faixa do erro será:

$$-\Delta < e \leq 0$$

O processo de truncamento é interpretado graficamente no lado direito da Figura 6-2.

Outra forma de quantização usada é o arredondamento. Um versão de (B+1) bits de x corresponde a atribuir ao valor de $\hat{x}$ o nível de quantização mais próximo de x, conforme o lado esquerdo da Figura 6-2. Como resultado, o erro de arredondamento é limitado por $-\frac{\Delta}{2} < e \leq \frac{\Delta}{2}$. Isto é, o módulo do erro máximo de arredondamento vem a ser a metade do erro máximo de truncamento.

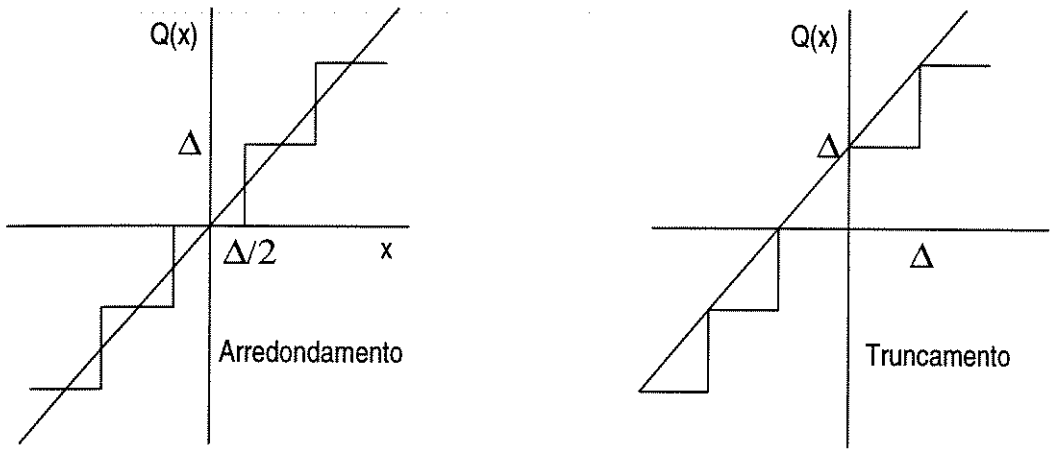


Figura 6-2 Interpretação gráfica das regras de arredondamento/truncamento

Uma variável discreta real x é representada por uma palavra digital Q[x]. O valor resultante da quantização pode ser dado como:

$$Q_B[x(n)] = x(n) - e(n)$$

A operação de quantização pode ser considerada como a soma de dois números reais para produzir um valor quantizado, isto é, definido sobre o anel finito. Na maioria das aplicações o valor de x é desconhecido a priori.

Shannon tem discutido que se:(Chassaing,Horning)(Taylor)(Oppenhein)

1. O erro{e(n)} do processo {x(n)} é estatisticamente estacionário;
2. {e(n)} é não correlacionado com {x(n)};
3. {e(n)} é um processo aleatório estacionário;

então a função densidade de probabilidade P(e) será uniformemente distribuído no intervalo de [-Δ/2,Δ/2] (Figura 6-3).

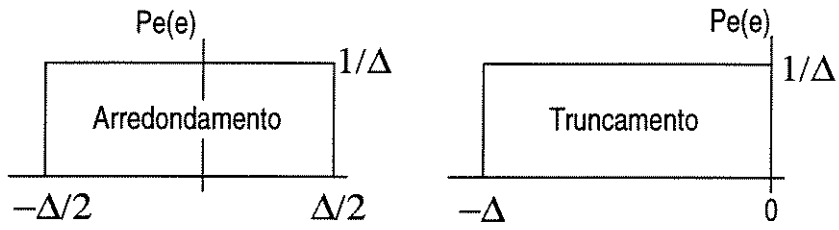


Figura 6-3 Representação estatística da função densidade de probalidade do arredondamento e truncamento

Macroscopicamente, estas suposições são geralmente válidas para processos dinâmicos e complexos. Contudo há exceções. Por exemplo, se x(n) é constante, o processo do erro de quantização é também uma constante. Um erro constante no processo não pode ser uniformemente distribuído.

Aceitando este modelo do erro de quantização, nos admitimos nossa incapacidade para modelar o processo de erro rigorosamente usando conceitos de álgebra finita.

Aceitando-se a hipótese de densidade uniforme do erro de quantização, teremos então o erro médio, $\overline{e}(n)$, para o arredondamento igual a zero e para o truncamento igual a -Δ/2.

A variância para ambos os casos será:

$$\sigma_e^2 = \frac{\Delta^2}{12}$$

O que dá um valor rms de:

$$e_{rms} = (\frac{\Delta^2}{12})^{1/2} = \frac{\Delta}{2\sqrt{3}}$$

Em dB, o erro será dado por:

$$10\log_{10}(\sigma_e^2) = K - 6.02B$$

onde K é uma constante. Isto é: a variância do erro pode ser reduzida de 6 dB para cada bit adicional somado a palavra de dados. No TMS320C25 a Relação Sinal Ruído (SNR) nos registradores com 16 bits é 96dB e nos registradores de 32 bits é 192 dB. Podemos então interpretar o erro como o ruído.

6.3.1 Efeitos da Quantização

Erros de quantização ocorrem nos filtros digitais, pois sistemas digitais devem representar amplitudes do sinal e valores de coeficientes dos filtros com precisão finita. A precisão é governada pelo número de bits para representar os valores. Assim, uma amostra de sinal ou um coeficiente de um filtro digital serão representados por palavras de comprimento finito. Como resultado, os coeficientes reais derivados nas seções do projeto de um filtro discreto irão ser realizados com um erro de quantização. Estes erros de aproximação podem afetar o desempenho de um filtro digital e em alguns casos a estabilidade do sistema.

Há dois tipos de erro de quantização em filtros digitais:

1 Quantização dos Coeficientes

2 Quantização do Sinal.

6.3.1.1 Erros da Quantização dos Coeficientes

Os erros de quantização dos coeficientes ocorrem porque o comprimento da palavra não é suficiente para representar o coeficiente do filtro com precisão. Para termos alguma idéia do efeito de quantização dos coeficientes no lugar dos pólos, consideraremos a função de transferência de um filtro IIR de um pólo:

$$\frac{X(z)}{Y(z)} \equiv H(z) = \frac{1}{1 - cz^{-1}}$$

O pólo p está localizado em z=*c*. O sistema deverá ser implementado com uma aritmética binária de (B+1) bits. O coeficiente c deverá ser representado com (B+1) bits de precisão.

O efeito de quantização dos coeficientes é geralmente determinado separadamente dos efeitos da quantização dos sinais. Isto é, os coeficientes ideais de uma função do sistema são substituídos por seus valores quantizados, e a resposta da função resultante é testada para ver se a quantização degrada o desempenho a níveis não aceitáveis. Se o número real *c* é quantizado com (B+1) bits, consideraremos a função de transferência do sistema como:

$$\frac{\hat{X}(z)}{\hat{Y}(z)} \equiv \hat{H}(z) = \frac{1}{1 - \hat{c}z^{-1}}$$

Uma vez que há somente 2^{B+1} diferentes números binários, o pólo de $\hat{H}(z)$ pode ocorrer somente em 2^{B+1} localizações do eixo real no plano Z. Como exemplo, se dispõe-se de apenas de dois bits fracionários (±x.xx), então *c* e p podem assumir somente os valores (0,±0.25,±0.50,±0.75) ao longo do eixo real no plano z e o sistema ainda permanecer estável. Pólos colocados em qualquer outro local, não são representáveis com 2 bits fracionários.

Para uma função de transferência de dois pólos temos:

$$H(z) = \frac{1}{1 + c_1 z^{-1} + c_2 z^{-2}} = \frac{1}{1 - 2r\cos\theta z^{-1} + r^2 z^{-2}} \equiv \frac{1}{(1 - re^{j\theta} z^{-1})(1 - re^{-j\theta} z^{-1})}$$

Equação 6-3

onde $z^2+c_1z+c_2=(z-p_1)(z-p_2)=0$ definem os pólos. Os coeficientes são reais e os pólos são complexos e conjugados, $p_1=p$ e $p_2=p^*$. Além disto, $c_1=-2\mathrm{Re}(p)$ e $c_2=p_1p_2=|p|^2$. O valor discreto c_1 contém a parte real do pólo, como no caso da função de transferência da função de um pólo, e valor discreto de c_2 contém o espaçamento radial dos pólos para valores discretos. Além disto, os pólos desta função de transferência podem somente estar na interseção de linhas verticais representando valores reais e círculos concêntricos representando valores radiais. Entretanto, é impossível evitar que essa quantização eventualmente movimente os pólos e zeros para uma posição diferente da original.

Referindo-se à Equação 6-3, faremos com que $\hat{c}_2 = Q_B(r^2)$ e $\hat{c}_1 = Q_B(2r\cos\theta)$ sejam respectivamente os valores quantizados de r^2 e $2r\cos\theta$. Os valores de r e θ que corresponderão à posição efetiva dos pólos do sistema serão:

$$r_{ef} = Q(r^2)^{1/2}$$

$$\theta_{ef} = \cos^{-1}(Q(2r\cos\theta)/2r_{ef})$$

Os coeficientes serão implementados com D bits após a vírgula. Então, tanto $Q(r^2)$ como $Q(2r\cos\theta)$ terão que ser múltiplos de 2^{-D}. Isso significa que as posições efetivas (r_{ef}, θ_{ef}) dos pólos só poderão estar num número finito de pontos discretos no plano z, dados por:

$$r_{ef} = (m^*2^{-B})^{1/2}$$

$$2r_{ef}\cos(\theta_{ef}) = (n^*2^{-B})$$

com m e n inteiros e m não-negativo.

Teremos então como nova função de transferência:

$$\hat{H}(z) = \frac{1}{1 + \hat{c}_1 z^{-1} + \hat{c}_2 z^{-2}} = \frac{1}{1 - 2r_{ef}\cos\theta_{ef} z^{-1} + r_{ef}{}^2 z^{-2}} \equiv \frac{1}{(1 - r_{ef} e^{j\theta_{ef}} z^{-1})(1 - r_{ef} e^{-j\theta_{ef}} z^{-1})}$$

Equação 6-4

As posições permitidas para os pólos formam uma constelação de pontos, mostrada na Figura 6-4 para o caso de dois bits fracionários e na Figura 6-5 para o caso de cinco bits fracionários e. O valor de $\hat{c}_2$ representa o radiano dentro do círculo de raio unitário. Ele são (0.25, 0.50, 0.75) e o correspondente raio (0.5, 0.707, 0.866). Os valores de $\hat{c}_1$, que darão partes reais dentro do primeiro quadrante do círculo de raio unitário, são (2.0, 1.75, 1.50, 1.25, 1.00, 0.75, 0.50, 0.25) e o valor correspondente da parte real de p são (1.0, 0.875, 0.75, 0.625, 0.5, 0.375, 0.25, 0.125). Pólos podem somente ser colocados na intersecçsão do círculo e uma linha radial. Examinando a Figura 6-4 e Figura 6-5, observamos que ao redor de z=±1 (0º e 180º) há menos posições possíveis dos pólos do que na área ao redor de ±j (90º e 270º). Isto indica que os pólos não podem ser colocados na região de ±1 (0º e 180º) com a mesma precisão de outras regiões do círculo.

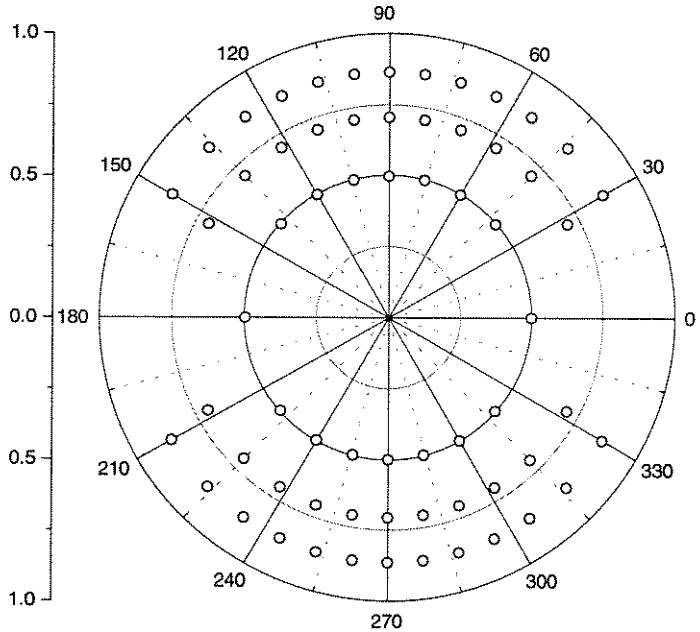


Figura 6-4 Posições possíveis para os pólos em 2 bits

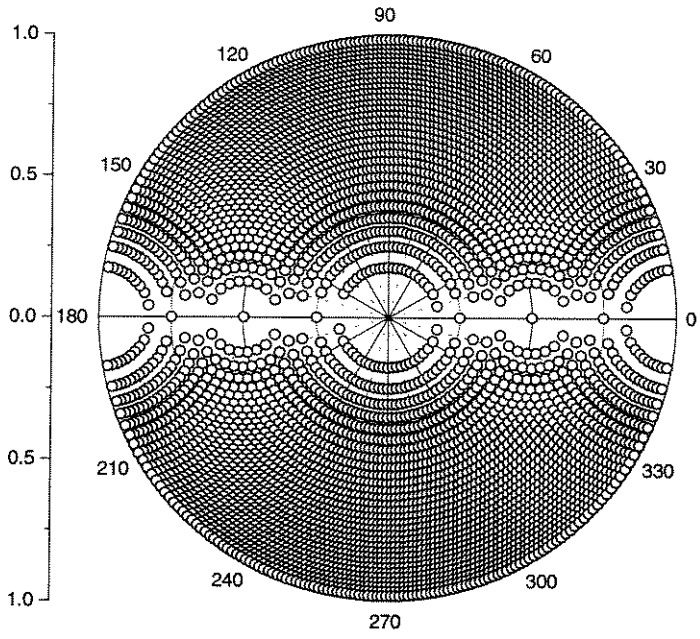


Figura 6-5 Posições possíveis para os pólos em 5 bits

O espaçamento dos pólos na região próxima a ±1 (0º e 180º) indica que quando os pólos forem colocados nesta região, eles irão ser mais sensíveis à quantização dos coeficientes. Isto significa que filtros passa-baixa ou passa-alta de banda estreita são sensíveis à quantização dos coeficientes pois seus pólos devem repousar nas regiões de +1 (0º) e -1 (180º), respectivamente.

A quantização de coeficientes é afetada pelo oversampling. Enquanto o oversampling tende a diminuir o aliasing e se aproximar mais do sinal, elé também tende a dirigir os pólos próximos para z=+1 (0º) e então causar grande sensibilidade dos coeficientes.

6.3.1.2 Erros de Quantização dos Sinais

Outra efeito da quantização encontrado em sistemas digitais é a redução da precisão dos sinais. Erros de quantização do sinal ocorrem quando um produto ou soma for truncado ou arredondado de 2B bits para B bits mais significativos (MSB). Se a amplitude do sinal é suficientemente grande e o espectro razoavelmente largo, estes erros podem usualmente ser modelados como sinais aleatórios não correlatos entre si e com o sinal. Assim a análise de ruído de quantização é tratada como um problema linear. Cada fonte de erro pode ser considerada estaticamente independente das outras.

Ruído de sinal e de quantização ocorrem tanto em sistema de ponto flutuante quanto em sistemas de ponto fixo. A quantização é usualmente um problema menor em representação de ponto flutuante porque apresenta uma capacidade de escalonamento dinâmica.

A representação em complemento de dois tem uma média zero no erro de quantização, se o sinal é arredondado, mas tem uma polarização se o sinal for truncado. Usualmente, isto não tem conseqüência, mas em algumas aplicações isto pode ser importante para não termos nível DC na saída.

Consideremos um sistema de 1º ordem representada pelo seguinte equação diferença:

$$y[n] = x[n] + cy[n-1] \qquad \textbf{Equação 6-5}$$

Ao quantizarmos os sinais *y[n], x[n]* e *y[n-1]* e o coeficiente *c* com (B+1) bits teremos:

$$\hat{y}[n] = \hat{x}[n] + \hat{c}\hat{y}[n-1] \qquad \textbf{Equação 6-6}$$

Em implementações de processamento de sinais é comum assumirmos que todas as variáveis e coeficientes são frações binárias. Assim, se nós multiplicarmos um sinal variável de (B+1) bits por um coeficiente de (B+1) bits, o resultado é uma fração com 2(B+1)-1 que pode ser convenientemente

reduzido para (B+1) por arredondamento ou truncando dos bits menos significativos. Com esta conversão, a quantidade X_m pode ser imaginada como um fator de escala que permite a representação de números que são maiores que 1 em magnitude. Por exemplo, em computação em ponto fixo é comum assumirmos que cada número binário tem um fator de escala $X_m=2^D$. Assim, um valor D=2 implica que o ponto é na verdade localizado entre b_2 e b_3 na palavra binária.

Ao efetuarmos o produto $\hat{c}\hat{y}[n-1]$ teremos como resultado um número com 2(B+1) - 1 bits. Pode-se transformar o resultado em um número de (B+1) para ser somado a *x[n].*, mas esta operação apresentará erro de quantização de aritmética. Para se somar o valor de *x[n]* ao produto, *x[n]* deverá ser transformado em um número de 2(B+1)-1 bits. Isto não acarretará qualquer erro. O resultado, y[n] deverá ser transformado novamente para (B+1) bits para ser armazenado em memória e posteriormente ser utilizado.

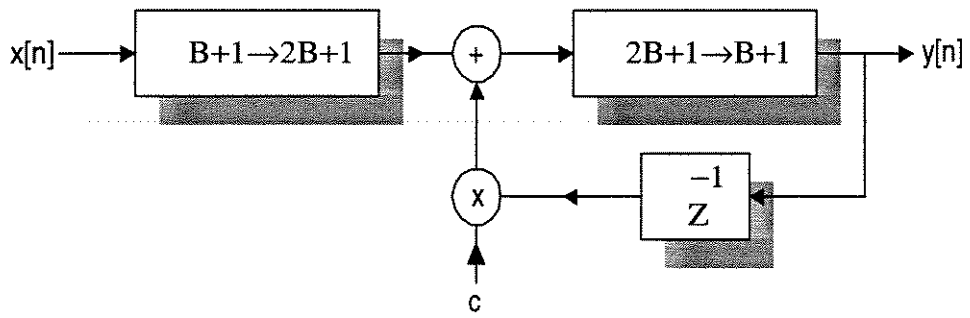


Figura 6-6 Implementação de um filtro digital simples

6.4 Erro de Aritmética

Vimos que o erro de truncamento é limitado ao intervalo $-\Delta < e \leq 0$ e que o erro de arredondamento é limitado por $-\Delta/2 < e \leq \Delta/2$. Contudo quando um overflow ocorre, o erro cresce enormemente sendo por vezes catastrófico. Por exemplo, considere um sistema de complemento de dois de quatro bits. Ao representarmos um número em complemento de dois com B+1 bits, teremos uma faixa de representação de $-2^B\Delta \leq \hat{x}_B < 2^B\Delta$. Para um sistema

de 4 bits teremos uma faixa de $-2^3\Delta \leq \hat{x}_3 < 2^3\Delta$. Considere o número 0111 que é equivalente a 7Δ. Se somarmos, 0001, a propagação do carry ira até o bit de sinal. Assim o resultado será igual a -8Δ e então o erro resultante será muito maior quando um overflow ocorrer. Isto vale também ao subtrairmos Δ de -8Δ. Temos na Figura 6-7 o efeito de um overflow natural/regular em complemento de dois.

Deveremos implementar algum método para minimizar o erro no resultado. Uma alternativa é a técnica de saturação, ou clipping, mostrada Figura 6-8. Com esta técnica o erro não aumenta abruptamente para a adição de complemento de dois quando um overflow ocorrer, contudo a desvantagem deste método de overflow é que impede o uso da seguinte interessante e útil propriedade da aritmética de complemento de dois: "*Dados vários números em complementos de dois cuja soma não causa um overflow, então o resultado da soma em complemento de dois destes números é correta mesmo se overflows intermediários puderem ocorrer*".

Para evitarmos que um overflow ocorra, gerando um erro muito grande e normalmente catastrófico durante uma operação de adição ou subtração, o TMS320C25 possui um modo de saturação de *overflow* ou *clipping*. Neste modo, o acumulador do TMS320C25, onde estamos realizando a operação de soma ou subtração, satura no valor mais positivo ou mais negativo dependendo do sentido do overflow. Desta forma, o tamanho do erro não incrementa abruptamente quando um *overflow* ocorre.

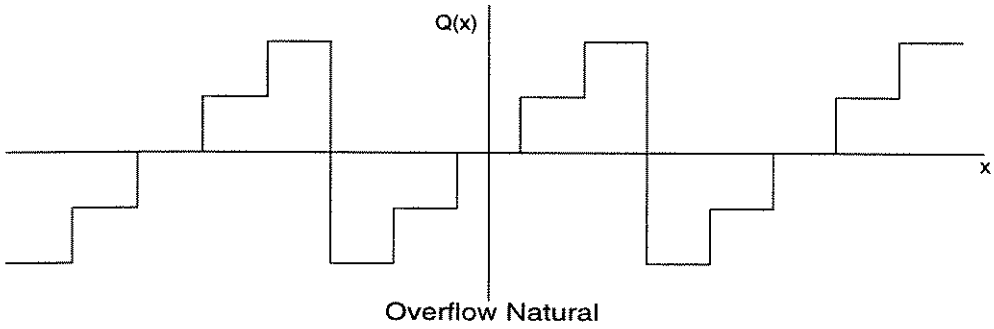


Figura 6-7 Overlow para uma operação com efeito de comprimento finito da palavra

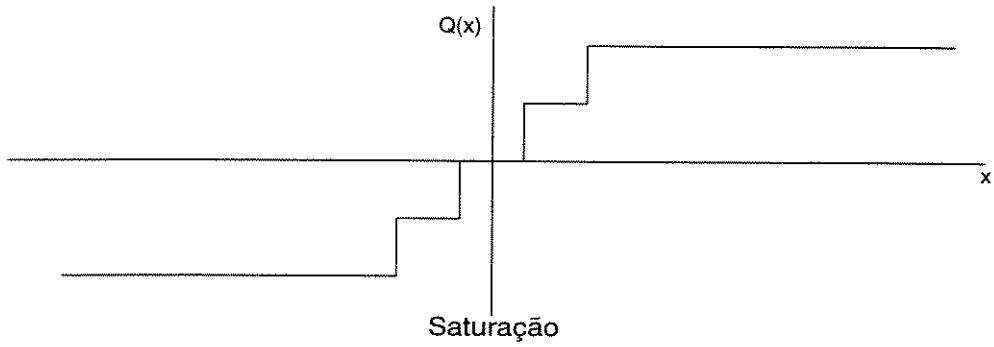


Figura 6-8 Saturação para uma operação com efeito de comprimento finito da palavra

Um overflow ocorre na representação de complemento de dois quando somamos ou subtraímos dois números com mesmo bit de sinal e obtivermos um número com sinal diferente. Devemos então observar o bit de sinal para detectarmos overflow. O TMS320C25 possui no registrador de status (ST0), um bit de overflow que indica quando um overflow ocorre. Possui também um bit para o carry no registrador ST1. O bit de carry é levado para 1 quando um carry ocorrer em uma adição ou é setado para 0 quando um borrow em uma subtração ocorrer.

O TMS320C25 pode trabalhar com números com sinal, representados em complemento de dois ou com números sem sinal. Com números com sinal temos uma faixa dinâmica de $0 \leq \hat{x}_B < 2^{B+1}\Delta$. Assim, o overflow ocorre em um oposição diferente, devido a mudança da faixa dinâmica. O TMS320C25 também possui um bit no registrador de Status ST1 que indica se as operações são com números com ou sem sinal (SXM - *Signal Extension Mode*). Observe que este bit serve para mudar a posição do overflow, pois muda a faixa dinâmica, além de ser útil nas operações de deslocamento e extensão de sinal que explicaremos mais tarde.

Para melhor compreendermos o efeito do overflow e do carry daremos um exemplo. Por questão de simplicidade usaremos novamente números de 4 bits em complemento de dois e realizaremos a seguinte operação: $6\Delta+(-2\Delta)$. Teremos como resposta 4Δ. Neste caso, um carry será gerado pois 6Δ em binário é 0110 e -2Δ é 1110. Somando 0110 com 1110 teremos 10100 sendo gerado um *carry* (vai um) no MSB, que será ignorado. O *carry* corresponde à

passagem por zero. A adição dos números 6Δ e −2Δ resulta na resposta correta por ignorarmos o carry out no MSB. Se efetuarmos -7Δ+(-8Δ) não teremos como resultado -15Δ. Ao invés disto teremos Δ como resposta. Neste caso, além do *carry* gerado, pois passamos pelo zero, teremos um *overflow*.

6.4.1 Ciclo limite de Entrada Zero

Erros de aritmética podem tomar a forma de ciclo limite de entrada zero, que é caracterizado pelo aparecimento de oscilações na saída do filtro mesmo para a entrada igual a zero. Este comportamento oscilatório é derivado das condições iniciais do filtro. Teoricamente, se o sistema é assintoticamente estável, esta energia do sinal de saída deve cair a zero. Contudo, em alguns casos, devido ao efeito de quantização, a saída não irá convergir para zero mas para um ciclo limitado. Algumas vezes estas oscilações tendem a desestabilizar um filtro. Mesmo sendo as oscilações de pequena amplitude, elas podem ser perturbadoras em aplicações de áudio e voz. O fenômeno de ciclo limite pode ser melhor entendido através de um exemplo de filtro de primeira ordem.

Exemplo: Considere um filtro dado por y(n)= x(n) - 0.75y(n-1). A resposta deste filtro para o estimulo $x(n)=(-1)^{n-1}$ é dada por {1,-1.75,2.3125,...,4}. A resposta deste filtro para y(0)=1 e x(n)=0 para todo n é dada por {-0.75,0.5625,-0.421875,...,0}. A estabilidade do filtro discreto é portanto aparente. Contudo, se o filtro for implementado usando-se uma palavra de 5 bits tendo a forma de ±x.xxx, então a resposta para uma entrada zero no filtro digital, com respeito ao arredondamento, é dada por:
y(i+1)=-3/4y(i);

i	-3/4*y(i-1)	Resultado	Binário	Arredondado	y(i)
0					1
1	-3/4* 1	-3/4	11.01000	11.010	-3/4

2	-3/4*-3/4	9/16	00.10010	00.101	5/8
3	-3/4*5/8	-15/32	11.10001	11.100	-1/2
4	-3/4*-1/2	3/8	00.01100	00.011	3/8
5	-3/4*3/8	-9/32	11.10111	11.110	-1/4
4	-3/4*-1/4	3/16	00.00110	00.010	1/4
5	-3/4*1/4	-3/16	11.11001	11.110	-1/4
6	-3/4*-1/4	3/16	00.00110	00.010	1/4

Tabela 6-1 Valores comparativos para ciclo-limite de entrada zero no arredondamento

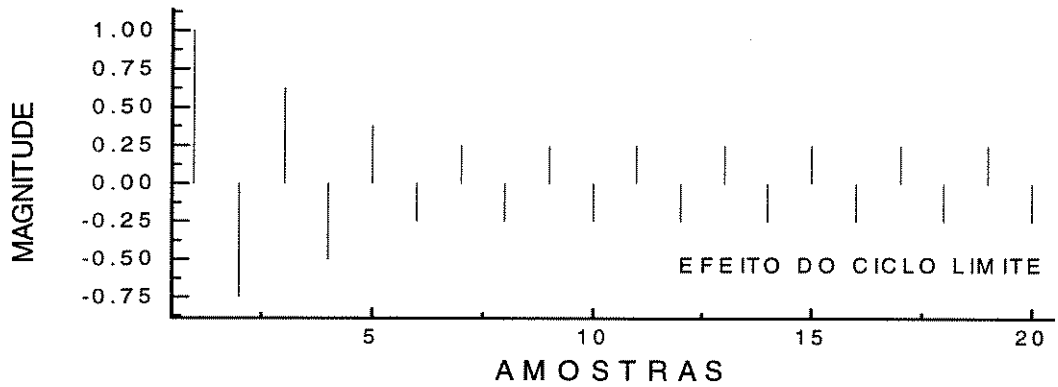


Figura 6-9 Resposta ao impulso de um filtro de primeira ordem exibindo ciclo limite

O efeito de palavra finita faz com que o LSB oscile com magnitude de 1/4 e período igual a 2T. A amplitude e o período destas oscilações, variam de projeto para projeto. Os efeitos e condições nos quais há ocorrência de ciclos limites para entrada zero podem ser previstas algebricamente.

6.4.2 Ciclo limite de Overflow.

Ciclo limite de overflow de larga escala é outro tipo de instabilidade. Enquanto os ciclos limites de entrada zero podem causar um transtorno, os ciclos limites de overflow de larga escala podem ser desastrosos. Eles podem ser provocados quando se excede a faixa dinâmica preestabelecida. A limitação da faixa dinâmica é estabelecida pela comprimento de palavra adotado e pelo sistema de numeração. Por exemplo, considere o filtro anterior implementado em sistema de complemento de dois. Uma palavra de

4 bits é usada, tendo a forma ±x.xx. A transição de 01.11=1.75 para seu sucessor em complemento de dois não é o decimal 2. No lugar disso, 10.00 tem um valor decimal de -2. Com respeito a esta limitação, a saída do filtro simples para uma entrada dada por $u(n)=(-1)^{n-1}$ para y(0)=0, torna-se:

i	$-\frac{3}{4}y(i-1)$	=	7bits	4bits	$+(-1)^{i-1}$	y(i)	Ideal	erro
1	-3/4* 0	0	00.00000	00.00	0+1	1	1	0
2	-3/4*1	-3/4	11.00100	11.01	-3/4-1	-7/4	-1.75	0
3	-3/4*(-7/4)	-21/16	10.10110	10.11	-5/4+1	-1/4	2.3125	2.5625
4	-3/4*(-1/4)	3/16	00.00110	00.01	1/4-1	-3/4	-2.734375	1.984375
5	-3/4*(-3/4)	9/16	00.10010	00.10	1/2+1	3/2	3.050781	1.55078

Tabela 6-2 Valores comparativos para ciclo-limite de overflow

Estes grandes erros são geralmente intoleráveis. Para superar estes efeitos vários métodos são possíveis. Uma medida é de aumentar o comprimento dos registradores do sistema de soma, acumuladores e assim em diante. Isto irá aumentar sua faixa dinâmica e assim evitar o problema de overflow.

Outra aproximação para a prevenção do problema do overflow é escalonamento. Este assunto será estudado na seção Escalonamento.onde o processo de entrada para um filtro é escalonado, assim nenhuma variável irá exceder a faixa dinâmica alocada. Esta aproximação é usada em computadores analógicos para prevenir a saturação dos amplificadores. Vamos, por exemplo, analisar o filtro de primeira ordem visto anteriormente.

O pior caso irá ocorrer quando a entrada for $u(n)=(-1)^{n-1}$, que produz uma saída(estado fixo)=4 ou -4. Adaptando o formato de dados para ±xxx.x, overflow irá possivelmente ocorrer para uma função arbitrária. Contudo, se a entrada for escalonada para 1/4, isto é $u'(n)=((-1)^{n-1})/4$, o overflow não poderá ocorrer mesmo nas condições de pior caso.

Este limite de escalonamento é difícil de ser determinado. Contudo, na Seção Escalonamento algumas técnicas poderosas de auxilio computacional serão apresentadas e irão simplificar o processo extremamente.

A técnica de saturação pode também ser usada para reduzir os efeitos de ciclos limites de overflow. A saturação irá geralmente eliminar ou suprimir grandes oscilações intoleráveis que podem ocorrer durante períodos de overflow em um sistema não saturado. No lugar de grandes erros, irão ocorrer pequenos erros no formato de não linearidade distorcendo o processo de saída (*clipping*).

6.5 Escalonamento

Devido aos efeitos dos erros de quantização dos coeficientes, é desejável se ter coeficientes tão grandes quanto possível para se manter a precisão. Fazendo-se isto, contudo, é possível produzir um sistema com ganho maior que um.

Nas discussões anteriores, nós determinamos que todos os operandos devem ser menores que a unidade com o objetivo de serem representados como fração. Como a entrada *x[n]* e a saída *y[n]* são limitados a $\pm$ 1, surge um problema ao termos um ganho maior que um. Para exemplificarmos, vamos supor um filtro com ganho máximo igual a 7: Assim:

$$X(z)*H(z)=Y(z)$$

onde:$|X|<1$ e $H_{max}=7$ e portanto: $Y_{max}=7$

Uma vez que o valor 7 não pode ser representado como uma fração, algo deve ser feito com o valor de X ou H para permitir que Y seja representado. Duas escolhas possíveis são possíveis:

$$Y=X*H/7 \text{ ou } Y=H*X/7$$

A primeira vista, pode parecer que qualquer uma das alternativas causa uma perda de precisão, uma vez que dividimos os termos *X* ou *H* por 7, perdendo quase 3 bits de precisão. Uma vez que a matriz *H* é a causa de overflow, um

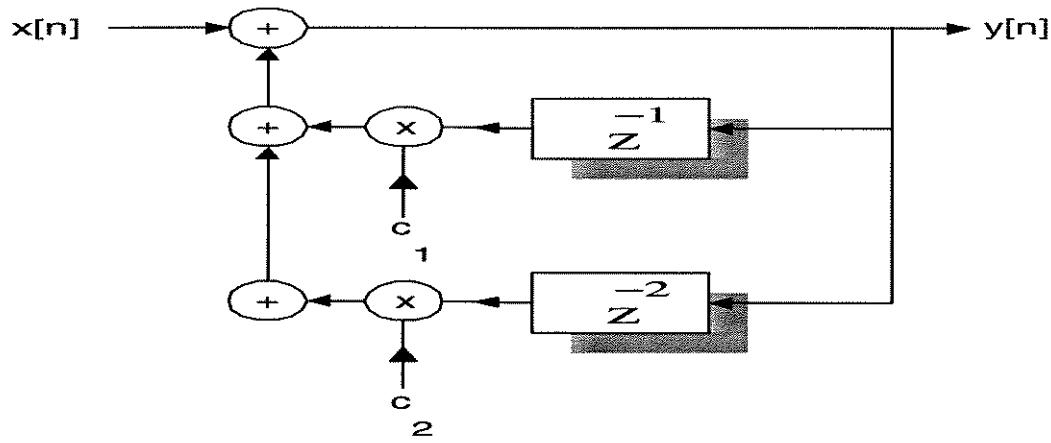
projetista geralmente irá reduzir o ganho do filtro. Fazendo-se assim, contudo, ele irá também reduzir a precisão de todos os coeficientes.

É possível escalonarmos a entrada sem perda de precisão no TMS320C25. Isto será vista nas próximas seções.

6.6 Erros na Implementação dos Ressonadores e Anti-ressonadores

Nesta seção investigaremos os Ressonadores (filtro IIR) e os Anti-ressonadores (filtro FIR) em maiores detalhes. O foco de atenção será as fontes de ruído e os modos de como minimizá-los. Muitos dos resultados apresentados aqui são aplicáveis ao anti-ressonador (filtro FIR), diferenciadores e geradores de ruído, mas são menos preocupantes devido a sua natureza não recursiva. Contudo os ressonadores (filtro IIIR) usam suas saídas como parte da próxima entrada, assim os erros podem ser misturados todo o tempo. Um ressonador, o qual é um filtro de segunda ordem possuí a função de transferência dada pela Equação 6-3.

Esquematicamente:



Para implementarmos um ressonador ou outro filtro qualquer necessitamos quantizar seus sinais de entrada x[n], saída y[n] e coeficientes c_i gerando erros. Assim, a Equação 6-3 torna-se:

$$\hat{H}(z) = \frac{1}{1 + \hat{c}_1 z^{-1} + \hat{c}_2 z^{-2}} = \frac{\hat{X}(z)}{\hat{Y}(z)}$$ **Equação 6-7**

Para assegurar o funcionamento correto do sistema, deveremos garantir certas condições. Para isto, as fontes de erro devem ser definidas, e a possibilidade de um erro catastrófico eliminada.

Para melhor caracterizar estes fenômenos de erro do filtro IIR, considere o diagrama na Figura 6-10.

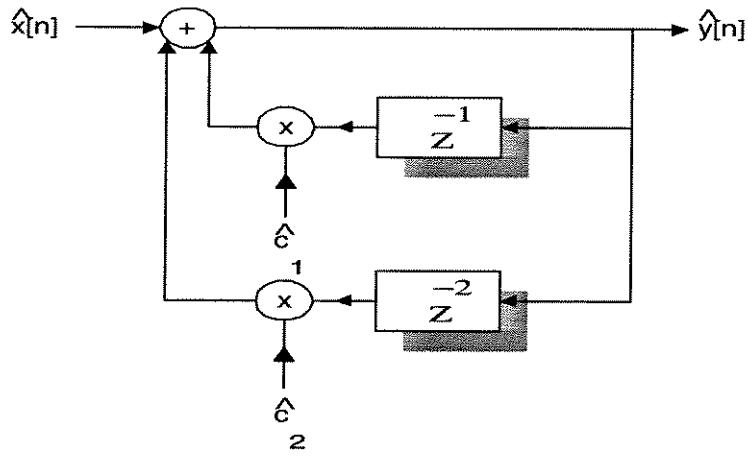


Figura 6-10 Filtro de segunda ordem com os coeficientes/sinais quantizados

onde:

$\hat{x}[n] = x[n] + e_{qx}$	Sinal de entrada + Erro de quantização do sinal de entrada
$\hat{y}[n] = y[n] + e_{qy}$	Sinal de saída + Erro de quantização do sinal de saída
$\hat{c}_1[n] = \hat{c}_1[n] + e_{qc1}$	Coeficiente c_1 + Erro de quantização do coeficiente c_1
$\hat{c}_2[n] = \hat{c}_2[n] + e_{qc2}$	Coeficiente c_2 + Erro de quantização do coeficiente c_2
e_{qc1} e e_{qc2}	Erro de quantização dos Coeficientes

O erro de quantização do coeficiente, e_{qci}, é o erro introduzido pela quantização do coeficiente. Como o coeficiente é arredondado somente uma vez, durante o projeto do filtro, este erro é constante. Após o projeto, o filtro pode ser simulado e reprojetado se o filtro não satisfazer as especificações. O e_{qy} denota a fonte de erro de arredondamento ou truncamento , o que é assumido ser branco e uniformemente distribuído. O ruído aditivo

corresponde à multiplicação com precisão finita. O efeito dos ruídos pode ser avaliado usando-se métodos teóricos de análise lineares ou através de simulação. Geralmente, o efeito do arredondamento ou truncamento da multiplicação ou adição é a redução da precisão. O efeito acumulativo deste erro pode ser controlado ajustando-se o comprimento da palavra do filtro e/ou escolhendo-se uma arquitetura para que seja mínima a sensibilidade aos erros de quantização.

Avaliar o erro total acumulado pode a princípio, parecer complexo se for considerados os erros de cálculo no produto e na soma. No entanto, estes erros são ou podem ser feitos, menores que o erro de quantização de I/O. Se o programador projetar o sistema com cuidado, o erro pode ser considerado inteiramente baseado nos erros de I/O e torna-se fácil para modelar. Isto é possivel no TMS320C25 nas operações de (multiplicação e acumulação) pois, o dispositivo permite uma faixa dinâmica de 192 dB para os resultados intermediários enquanto os operandos de I/O, com 16-bits, possuem uma faixa dinâmica de 96 dB.

Para se alcançar resultados ótimos para o filtro IIR, o programador deverá manter a máxima precisão possível.

6.6.1 Resumo

Como visto nas seções anteriores, cálculos numéricos apresentam vários fontes de erro incluindo:

1 Quantização do Sinal de Entrada/Saída: O número de bits de precisão disponíveis em um sinal na entrada x[n] e na saída y[n] determinam os erros que são atribuídos aos sinais. Ambos são dominantes pois o sinal de entrada x[n] e o sinal de saída y[n] e suas amostras atrasadas y[n-1] e y[n-2] circulam dentro e ao redor do filtro.

2 Quantização dos Coeficientes do Filtro. O resultado da representação imprecisa dos coeficientes é um desvio da resposta de freqüência desejada do sistema. Na maioria dos casos coeficientes de 16-bits são suficientes. Contudo, no caso de filtros com alto Q com pólos perto do círculo unitário, pequenos erros de coeficiente podem resultar em pólo fora do círculo de raio unitário. Esta condição permitiria um sistema instável.

3 Ruído de Arredondamento não correlato. Este ruído é o resultado de um sistema cujas saídas são muito pequenas para representar com suficiente precisão. Como o número de bits disponíveis diminui, o erro aleatório no LSB torna-se crescentemente significante.

4 Ruído de Arredondamento Correlato: A existência dos ciclos limites e o seu pior caso overflow, é atribuído ao efeito de erros acumulados. Estes fenômenos é difícil para modelar. É geralmente melhor eliminado com o uso de coeficientes de alta precisão e assegurando a disponibilidade de uma faixa dinâmica para representação do sinal de entrada e saída.

5 Restrição a faixa dinâmica. Os operandos de 16-bits usados nos dispositivos TMS320C25 permitem uma faixa dinâmica de 96 dB. A faixa dinâmica é extendida para 192 dB nas operações de (multiplicação e acumulação) para os resultados intermediários.

6.7 Escalonamento nos Filtros IIR

A implementação de um ressonador em ponto fixo requer que todos os coeficientes (c_1,c_2) e as variáveis (x,y) sejam representados em numeração de ponto fixo, escalonados ou ajustados para frações (na realidade números inteiros) quando estes forem transferidos para a memória.

O overflow é um dos maiores problemas na implementação de filtros IIR na aritmética de ponto fixo. Variáveis e coeficientes devem ser tão grandes

quanto possível para termos a maior faixa dinâmica mas não tão grandes a ponto de causar overflow. O overflow pode ocorrer em qualquer um dos nós de soma. A entrada x[n] ou os coeficientes (c_1,c_2) deverão ser escalonados para prevenir o overflow. Quando escalonamos, deveremos ter cuidado para não modificarmos a resposta original H(z). Utilizando a equação diferença e escalonando y[n] por um fator s teremos:

$$\frac{y(n)}{s} = \frac{c_0}{s_1}\frac{x(n)}{s_2} + \frac{c_1}{s_3}\frac{y(n-1)}{s_4} + \frac{c_2}{s_5}\frac{y(n-2)}{s_6} \qquad \textbf{Equação 6-8}$$

onde y(n-k)/s e x(n-k)/s_2, k=0,1,2,3... representam os valores após o escalonamento de y(n-k) e x(n-k), respectivamente, e $s=s_1s_2=s_3s_4=s_5s_6$. O fator de escalonamento s é distribuído entre os c_i e o sinal de entrada x[n] como visto na Equação 6-8. Esta equação representa a mesma relação das equações diferenças exceto que a saída está escalonada por s. Para achar os fator s de escalonamento temos três maneiras:

$$\sum_{n=0}^{N-1}|h(n)| \qquad \textbf{Equação 6-9}$$

$$[\sum_{n=0}^{N-1}h(n)^2]^{1/2} \qquad \textbf{Equação 6-10}$$

$$max|H(jf)| \qquad \textbf{Equação 6-11}$$

onde *H(jf)* é o domínio de freqüência da resposta ao impulso unitário *h(n)*. A Equação 6-9 é o limite superior na resposta e é a mais conservadora das três. A Equação 6-9 é a única que garante nenhum overflow. A Equação 6-10 não será estudada (Chassaing, Horning). A Equação 6-11 apenas garante que não teremos um overflow em um estado estável na resposta senoidal. Transientes podem ainda causar overflow.

Segue-se um guia geral para escalonamento de filtros IIR:

1 Estimar a gama de valores das variáveis;

2 Escalonar a entrada ou os coeficientes pelo maior estimador para prevenir o overflow. As variáveis e coeficientes devem ser tão grandes quanto possível para maximizar a precisão e a faixa dinâmica da função de transferência;

3 Coeficientes maiores que 1 podem ser transformados pela divisão por uma potência de dois e quando usado, deverá ser multiplicado por um fator de dois para produzir o resultado correto. Adição de ±1 pode também ser usado;

4 Overflows intermediários não irão afetar a resposta final correta desde que as seguintes condições sejam satisfeitas:

- O modo de saturação estiver inativo;
- O resultado final da soma estiver na faixa entre +1 a -1.

A simulação das operações de um filtro é um passo importante para o sucesso de sua implementação, pois é muito mais fácil detectarmos um overflow na simulação que em uma operação em tempo real. A resposta ao impulso unitário e a resposta em freqüência fornecem uma estimativa da magnitude dos sinais nos nós de uma estrutura e serão usadas para escalonar cada filtro no sintetizador de Klatt.

7. Implementando o Sintetizador com TMS320C25

7.1 Introdução

O sintetizador de formantes de Klatt em cascata/paralelo será implementado com TMS320C25. Entretanto, os parâmetros para o funcionamento do sintetizador serão gerados externamente. Os dados gerados pelo programa do TMS320C25 serão analisados em um computador compatível com o IBM PC como mostrado na Figura 7-1. Os parâmetros de controle do sintetizador, como o deslocamento da freqüência do primeiro formante em função do tempo, são especificados pelo experimentador, usando o programa de controle do sintetizador HANDSY.FOR (Klatt) ou um outro módulo conversor (Gomes, Nagle,Chiquito), que for capaz de gerar os parâmetros do sintetizador de Klatt. Mais de 20 parâmetros de controle podem ser variados em função do tempo, servindo como entrada para a subrotina de conversão dos parâmetros em coeficientes KLEPAR1.FOR. O arquivo de saída desta rotina, ENTRADA.HEX será utilizado pelo TMS320C25 para a geração de forma de onda sintetizada. As amostras de saída da forma de onda são computadas e armazenadas em memória, quando utilizado o XDS, ou em disco quando simulamos o TMS320C25 em um PC. Estas amostras em formato hexadecimal podem ser convertidas para formato decimal para visualizarmos e analisarmos a forma de onda com o programa MATLAB, ou convertidas para um formato adequado para serem reproduzidas e analisadas no CSL (*Computer Speech Lab*), ou ainda convertidas para o formato .WAV para serem reproduzidas por um PC com placa de som e ambiente WINDOWS e/ou OS/2.

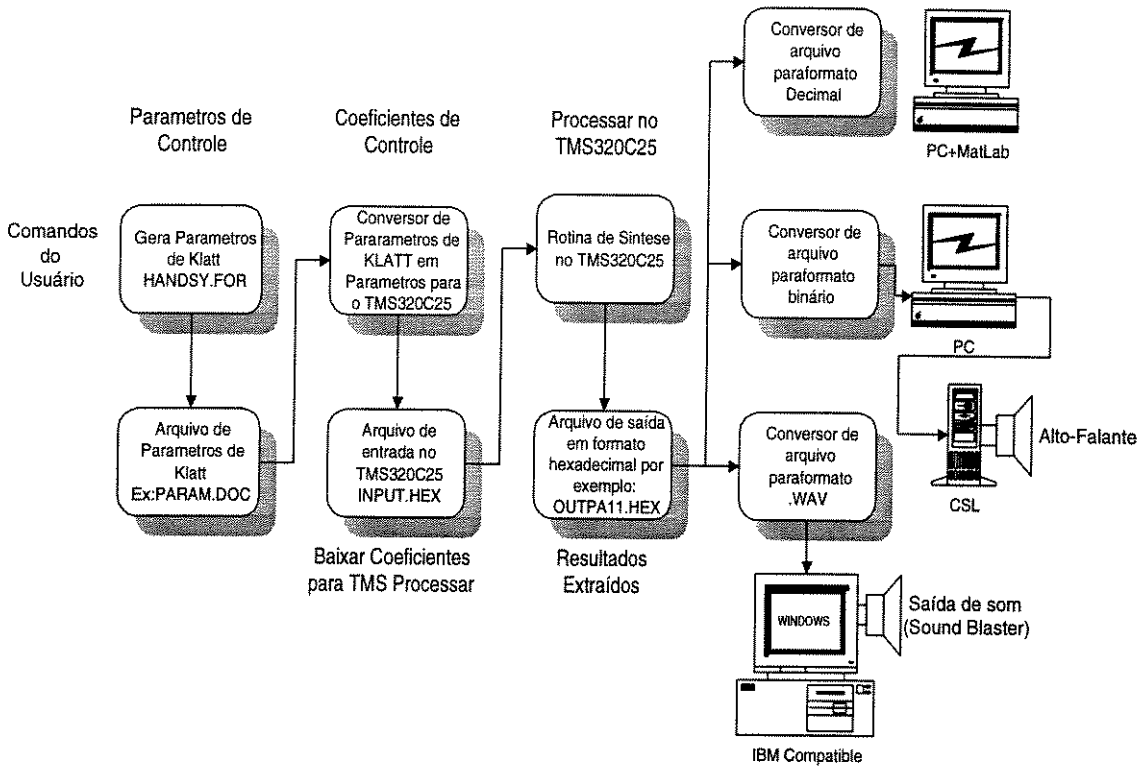


Figura 7-1 Relação dos componentes software/hardware no sintetizador

Ao analisarmos a forma de onda de saída, podemos utilizar o espectrograma, com o objetivo de se fazer comparações entre os espectros de forma de onda natural e sintético.

7.2 Modificando o Algorítmo de Klatt

Como vimos no capítulo 3, a característica de irradiação, sendo um filtro diferenciador com característica de +6 dB/oitava, pode ser movida para antes dos blocos da função de transferência do trato vocal em cascata/paralelo e destes paras as fontes de voz e ruído. Desta maneira o filtro passa-baixa, com característica de -6 dB/oitava, localizado na fonte de ruído pode ser eliminado. É interessante notarmos que o diferenciador localizado no bloco da função de transferência do trato vocal em paralelo não será modificado.

Para se evitar transientes inoportunos ao implementarmos o sintetizador em ponto fixo, embora sendo um sistema linear variante no tempo, certas mudanças na ordem dos ressonadores da função de transferência do trato

vocal serão feitas. Estas mudanças não afetarão o processamento, pois os coeficientes dos ressonadores variam muito lentamente com o tempo, podendo considerarmos o sistema como invariante no tempo.

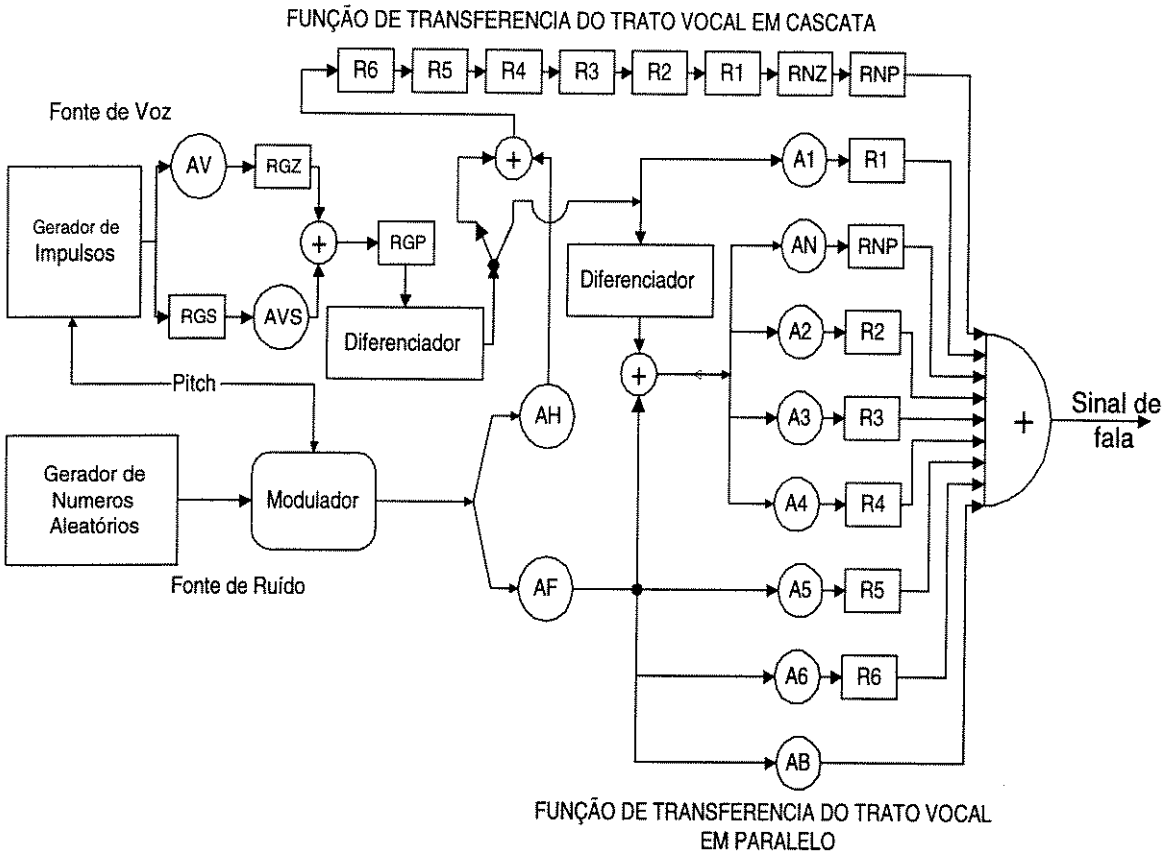


Figura 7-2 Diagrama do sintetizador de formantes cascata/paralelo modificado

Outra mudança a ser feita será na fonte de voz. Ao aproximarmos a característica de irradiação da fonte de voz, os fatores multiplicativos AV e AVS não poderão estar imediatamente antes ao diferenciador pois AV e AVS sendo atualizados em intervalos de 5 ms (default) causarão picos indesejáveis no sinal de saída do diferenciador. O fator multiplicativo AV será colocado após o gerador de impulsos. Contudo o fator multiplicativo AVS só poderá ser colocado após o ressonador RGS pois caso contrário teremos problemas de ciclos limites de entrada nula. Todos estes problemas, que são associados a aritmética em ponto fixo serão explicados ainda neste capítulo.

Assim teremos então o seguinte diagrama de blocos apresentado na Figura 7-2 do Sintetizador de Klatt.

7.3 Implementação do Ressonador com TMS320C25

Vimos no capítulo 3 que equação de um ressonador digital é:

$$y[n]=Ax[n]+By[n-1]+Cy[n-2]$$

Do capítulo 6, Equação 6-4, a quantização dos coeficientes de um ressonador digital será:

$$C \equiv \hat{c}_2 = Q_B(r^2)$$

$$B \equiv \hat{c}_1 = Q_B(2r\cos\theta)$$

$$A = 1 - B - C$$

$$r = \exp(-\pi * BW / FA)$$

$$\theta = 2 * \pi * F / FA$$

onde:

FA = freqüência de amostragem

F = freqüência de ressonância do filtro

BW = largura de Banda

então a faixa dinâmica será:

$r = exp(-\pi\ BW/FA)$	$0 \geq r > 1$
$C = -r^2$	$-1 \leq C < 0$
$B = 2r\ cos(-2\pi\ f/FA)$	$-2 \leq B \leq 2$
$A = 1-B-C$	$-1 < A \leq 4$

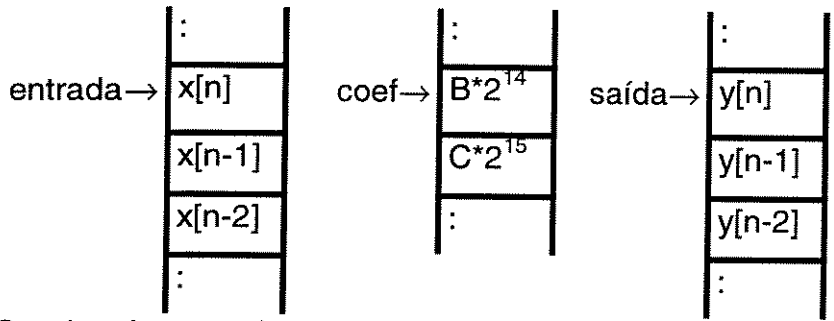
Para representarmos números fracionários, no TMS320C25, será necessário normalizarmos estes números. Para normalizarmos C, o multiplicaremos por 2^{15} e teremos uma faixa de $-2^{15} < C*2^{15} < 0$ e B será normalizado ao multiplicarmo-lo por 2^{14} e teremos uma faixa de $-2^{15} < B*2^{15} < 2^{15}$.

Visando a implementação no TMS320C25 consideremos a seguinte equação do filtro $y[n]=((x[n]*2^{15}+C*2^{15}*y[n-2])/2+B*2^{14}*y[n-1])/2^{14}$. Pode parecer a princípio que modificamos a equação do ressonador, mas se simplificarmos a equação diferença voltaremos a equação original exceto pelo fator *A*. A

seqüência de saída estará dividida por *A*. Para escalonarmos a entrada, para evitar que *y[n]* seja maior que ±1, basta multiplicarmos *x[n]* por um fator menor que 2^{15}. Assim, a saída y[n] será também dividida por este fator de escalonamento.

7.4 Implementação da Equação do Ressonador Digital

Estudaremos a implementação da equação TMS320C25 a partir da forma da organização da memória.



O valor de entrada aponta para a posição de memória cujo conteúdo é x[n], coef para B e saída para y[n]

`LAC entrada,15`	;Carrega o acumulador com o valor*2^{15} apontado por entrada.
`LT saida+2`	;Carrega o registrador T com o conteúdo da posição de memória apontado por saida+2
`MPY coef+1`	;Multiplica o registrador T pelo conteúdo da posição de memória apontado por coef+1. O produto fica no registrado P
`LTD saída + 1`	;Carrega o registrador T com o conteúdo da posição de memória apontado por saída + 1. Copia o conteúdo da posição de memória apontado por saída + 1 para a posição seguinte ou seja para a posição de memória apontada por saída + 1
`MPYA coef`	;Soma o conteúdo do registrador P ao acumulador. Multiplica o registrador T pelo conteúdo da posição de memória apontado por coef. O produto fica no registrado P
`SFR`	;Divide o conteúdo do acumulador por 2
`APAC`	;Soma o conteúdo do registrador P ao acumulador
`SACH saída, 2`	;Armazena o conteúdo do acumulador/2^{16-2} na posição de memória apontada por saída

Podemos observar a simplicidade de implementação com poucas instruções, e utilização dos recursos característicos do TMS320C25.

7.5 Implementação do Anti-ressonador

Para a implementação do anti-ressonador deveremos proceder de maneira semelhante, preparando previamente os coeficientes a serem utilizados, como fizemos no caso do ressonador:

$$y[n]=A'x[n]+B'x[n-1]+C'x[n-2]$$

e

$$r = \exp(-\pi\, BW\, T)$$

$$C = -r^2$$

$$B = 2\cos(-2\pi\, F\, T)$$

$$A = 1-2r\cos(-2\pi\, F\, T)+r^2$$

$$A' = 1 / A$$

$$B' = -B / A$$

$$C' = -C / A$$

Assim a faixa dinâmica será:

$r=exp(-\pi*BW*T)$	$0 \geq r > 1$
$C=-r^2$	$-1 \leq C < 0$
$B=2*cos(-2*\pi*F*T)*r$	$-2 \leq B \leq 2$
$A=1-B-C$	$-1 < A < 4$
$A'=1/A$	$-\infty < A' < \infty$
$B'=-A'*B$	$-\infty < B' < \infty$
$C'=-A'*C$	$-\infty < C' < \infty$

FA = período de amostragem

F = freqüência do filtro

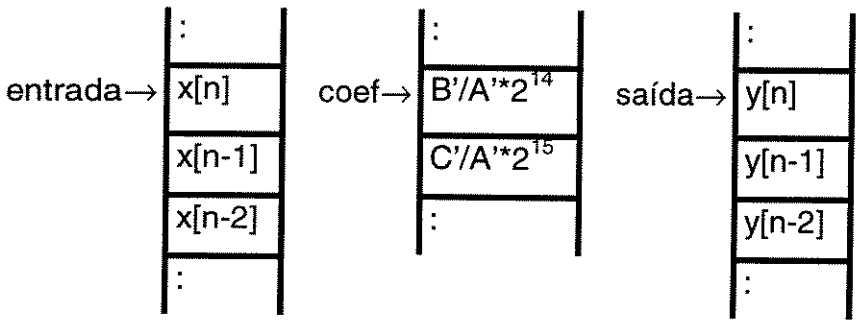
BW = largura de Banda do ressonador

O valor do coeficiente *A'* é problemático na implementação do anti-ressonador, pois os coeficientes escalonados deverão estar na faixa de ±1. Pode ser observado que um anti-ressonador é um filtro FIR pois a saída só depende das

entradas passadas. Portanto dividiremos ambos os termos por A', tendo como resultado:

$$y[n]/A'=x[n]+[B'/A']^* x[n-1]+[C'/A']x[n-2]$$

Utilizando-se o mesmo raciocínio anterior, multiplicaremos $C'/A'*2^{15}$ e $B'/A'*2^{14}$ Como na implementação do ressonador, a implementação do anti-ressonador será feita a partir da forma de organização dos dados na memória:



O valor de entrada aponta para o valor de x[n], coef para $B'/A'*2^{14}$ e saída para y[n]

`LAC entrada,15`	;Carrega o acumulador com o conteúdo*2^15 da posição de memória entrada.
`LT entrada+2`	;Carrega o registrador T com o conteúdo da posição de memória entrada+2
`MPY coef+1`	;Multiplica o registrador T pelo conteúdo da posição de memória apontado por coef+1. O produto fica no registrado P
`LTA entrada+1`	;Soma o conteúdo do registrador P ao acumulador Carrega o registrador T com o conteúdo da posição de memória apontado por entrada+1.
`MPY coef`	;. Multiplica o registrador T pelo conteúdo da posição de memória apontado por coef. O produto fica no registrado P
`SFR`	;Divide o conteúdo do acumulador por 2
`APAC`	;Soma o conteúdo do registrador P ao acumulador
`DMOV saida+1`	;Copia o conteúdo da posição de memória saida+1 para a posição seguinte ou seja para a posição de memória saida+1+1
`DMOV saída`	;Copia o conteúdo da posição de memória saída para a posição seguinte ou seja para a posição de memória saida+1
`SACH saida,2`	;Armazena o conteúdo do acumulador/2^[16-2] na posição de memória apontada por saída

Com isto o valor da saída do anti-ressonador estará dividida por um fator A'.

7.6 Implementação do Gerador de Números Aleatórios

O sintetizador de formantes de Klatt possui um gerador de números aleatórios. Entretanto, qualquer programa computacional, irá produzir um resultado que é inteiramente previsível. Assim, um programa não conseguirá gerar um número verdadeiramente aleatório. Chamaremos a este resultado de números pseudo aleatórios. Deve ser lembrando ainda que o que pode ser "suficientemente aleatório" para uma aplicação, pode não ser para outra.

Um gerador de números pseudo aleatórios pode ser facilmente utilizado em uma linguagem de alto nível como C++ ou FORTRAN, pois estas já possuem estas funções implementadas. A Linguagem Assembly do TMS320C25 não possui nenhuma função para a geração de números pseudo aleatórios.

7.6.1 Geração de Números Pseudo Aleatório com Distribuição Uniforme

Embora um grande números de técnicas para geração de uma seqüência uniformemente distribuída de números pseudo aleatórios seja descrita na literatura, apresentaremos uma das mais velhas e mais populares técnicas para geração de números pseudo aleatórios chamada algoritmo de Lehmer (Papoullis) modificado ou método congruential (*Numerical Recipies in Fortran*), onde o número aleatório $x(n)$ é gerado a partir do número aleatório precedente $x(n-1)$ pela relação recursiva:

$$x(n) = (A * x(n-1) + C) \bmod(B) \qquad \textbf{Equação 7-1}$$

onde B é o número principal, muito grande. As constantes A e C são positivas e inteiras cuidadosamente escolhidas chamados de multiplicador e incremento respectivamente. A seqüência gerada pela Equação 7-1 irá eventualmente se repetir com um período que obviamente será menor que *B*. Para valores apropriados de *A*, *B* e *C*, esta relação recursiva irá gerar inteiros na faixa de 1 a B-1 em seqüência quase aleatória com período *B*. As vantagens deste método são a simplicidade e a pequena quantidade de memória requerida. É

um método muito rápido requerendo somente poucas operações a cada chamada, daí o seu uso quase universal. As desvantagens são: existem outros métodos mais rápidos, pois deverá ser realizada uma multiplicação e uma divisão a cada interação e o método é extremamente sensível ao valores de *A*, *B* e *C* e ainda os valores não estão livres de correlação nas sucessivas chamadas.

7.6.2 Técnica para Geração de Números Pseudo Aleatórios com Distribuição Gaussiana

O número pseudo aleatório, gerado pela técnica pela descrita na seção 7.6.1, possui uma distribuição uniforme. É importante a geração de uma seqüência de números pseudo aleatórios com distribuição gaussiana. Uma maneira simples para isto é utilizar o Teorema do Limite Central, que diz que no limite de N tendendo a infinito, a soma de N variáveis aleatórias independentes, identicamente distribuídas tendem a uma variável com distribuição Gaussiana. Assim, para se converter uma seqüência de números {x(n)} independentes, uniformemente distribuídas em uma seqüência de números com distribuição gaussiana {y(n)} devemos:

$$y(n) = \frac{1}{N}\sum_{i=0}^{N-1} x(nN - i)$$ **Equação 7-2**

onde N é suficientemente grande. Nos casos práticos, um valor de N na ordem de 10 permite uma aproximação razoavelmente boa para uma distribuição gaussiana.

7.6.3 Implementação do Gerador de Números Aleatórios com TMS320C25

O gerador de Números Aleatórios, simula a fonte de ruído de aspiração e fricação, gerando um ruído pseudo aleatório, com distribuição uniforme entre 0 e 1, e armazena os valores calculados em um vetor de 16+1 elementos. O valor de IA utilizado foi 32749 e IC foi 32717. O valor do número pseudo aleatório será calculado multiplicando-se a "semente" por IA, somando-se IC e

armazenado-se o resultado no LSB do acumulador. Este valor será usado como "semente" para o próximo número. A primeira "semente" utilizada será 1. Os valores de IA e IC foram escolhidos ao acaso gerando-se uma seqüência de números primos de 15 bits e escolhendo-se os dois maiores. Não foi encontrada na literatura nenhuma referencia sobre estes método.

Após calcular os 17 elementos, o programa soma os primeiros 16 elementos e divide o resultado por 16 (executando quatro shift a direita). Ao executarmos esta soma teremos um resultado com distribuição aproximadamente gaussiana entre (0 a 1). Depois disto subtraímos um off-set e teremos uma distribuição entre (-0.5 e 0.5).

Após calcularmos a soma dos 16 primeiros elementos, o 17º valor será copiado para a 1º posição e utilizado como semente para o cálculo do próximo número que será armazenado na 2º posição. Repete-se o método.

7.7 Implementação dos Diferenciadores

Os diferenciadores serão implementados com 32 bits de precisão. Embora coeficientes e sinais sejam números representados com 16 bits, a diferenciação deverá ter uma precisão maior pois como os sinais variam lentamente, 16 bits não são suficientes para a representação da diferenca entre dois resultados no acumulador. Assim a equação efetiva implantada será:

$$u[n] = (x[n]-x[n-1])*2^7$$

sendo $x[n]$ e $u[n]$ números de 32 bits. O fator 2^7 é um fator de escala determinado por simulação. Como o sinal $u[n]$ será utilizado como entrada de ressonador é interessante termos os 16 MSB's com valores significativos. Por facilidade teremos uma estrutura de memória desta maneira:

	:
ALTO	u[n]
	u[n-1]
BAIXO	u[n]
	u[n-1]

O modo de saturação (Clipping) fica ativo durante o cálculo da diferença e logo após é desativado para não afetar outras partes do programa.

7.8 Implementação dos Integradores

Os integradores serão implementados com 32 bits de precisão. São utilizados dois integradores. O primeiro calcula o valor de AH e o segundo de AF. O integrador deverá ter um precisão maior pois como os sinais variam lentamente, 16 bits não são suficientes para a representação precisa da integral. O valor do parâmetro passado para o TMS320 será o passo de integração.

O passo de integração para o integrador de AH é dado por:

passoAH[n]=(AH[n]-AH[n-1]) / (tamanho do trecho a sintetizar)

e o passo de integração para o integrador de AF é dado por:

passoAF[n]=(AF[n]-AF[n-1]) / (tamanho do trecho a sintetizar)

estes valores são calculados externamente e depois transmitidos ao TMS320C25.

A equação de integração implementada no TMS320C25 será:

$$x[n] = x[n-1]+passo$$

sendo x[n] e passo números de 32 bits. Para o cálculo do ruído de aspiração e fricação utilizaremos a parte alta de AH e AF respectivamente. Novamente, teremos uma estrutura de memória desta maneira:

	:
ALTO	x[n]
	x[n-1]
BAIXO	x[n]
	x[n-1]

O modo de saturação (Clipping) fica ativo também durante o cálculo da integral sendo desligado logo após sua operação.

8. Exemplos de Síntese de Sinais da Fala e Análise dos Resultados

8.1 Estratégias de Síntese

Estratégias gerais para a síntese de sílabas estão fora do escopo deste material, mas os seguintes parágrafos pretendem provem os valores típicos para um som estático desejado. Os valores apresentados na Tabela 8-1 e Tabela 8-3 podem constituir um bom ponto de partida para a síntese de uma expressão se os procedimentos listados abaixo forem adotados. Os passos usados na síntese de uma expressão no TMS320C25 em nossos laboratórios são descritos e um exemplo simples é apresentado.

8.2 Síntese de Vogais

Os parâmetros de controle que são usualmente variados para gerar uma vogal isolada são a amplitude da voz AV, a freqüência fundamental das vibrações das cordas vocais, F0, os três primeiros formantes F1, F2, e F3 e as larguras de banda B1, B2 e B3. A freqüência do 4º e 5º formantes podem ser variadas para simular detalhes espectrais, mas isto não é essencial para uma alta inteligibilidade. Para criar uma terminação natural de uma vogal aspirada, a amplitude de aspiração AH e a amplitude de voz quase senoidal AVS podem ser ativados (Klatt).

Vogal	F1	F2	F3	F4	B1	B2	B3	B4
[a]	649	1276	2465	3588	148	58	57	97
[é]	470	1972	2637	3442	72	88	83	321
[ê]	325	2077	2660	3394	45	97	87	271
[i]	219	2177	2845	3661	49	42	263	159
[ó]	556	997	2589	3404	64	39	171	131
[ô]	342	824	2555	3135	85	55	228	186
[u]	353	732	2574	3215	96	147	156	204

Tabela 8-1 Valores de formantes pronunciadas pelo autor.

A Tabela 8-1 incluí sugestões para o controle dos parâmetros variáveis que são usados para diferenciar vogais. A freqüência dos formantes e a largura de banda são obtidas utilizando-se o CSL (*Computer Speech Lab*) sendo os dados de entrada, o conjunto de vogais pronunciadas pelo autor. Valores da largura de banda e freqüência são ligeiramente diferentes dos valores obtidos por Klatt e por Rabiner e Gold.

	F1	F2	F3	
Autor	581	1276	2409	[a]
Nagle	700	1350	2600	[a]
Rabiner e Gold	520	1190	2390	Λ
Klatt	620	1220	2550	Λ

Tabela 8-2 Valores comparativos típicos de formantes. A vogal Λ esta tipicamente presente na palavra *but*. Já a vogal [a] foi medida sozinha pela autor.

A amplitude da fonte de voz, AV, é ajustada para algo ao redor de 60 dB para uma vogal forte e cai gradualmente para uns poucos dB perto do fim da sílaba. O contorno da freqüência fundamental para uma vogal isolada pode ser aproximado por uma queda linear de F0, por exemplo de 130 para 100 Hz.

8.2.1 Sintetizando Vogais

Como uma aplicação inicial, iremos sintetizar as vogais presentes na Tabela 8-1. As vogais serão sintetizadas com uma duração de 100 ms. As amplitudes finais serão AVS=45, AH=50 e AV = 55 dB, exceto nas vogais [ô] e [u], sendo que nos primeiros 50 ms as amplitudes (AV, AH, AVS) partem de zero até o seu respectivo valor final da síntese. Nos próximos 50 ms a amplitude se mantêm constante e após este período cai a zero. A freqüência fundamental F0 se mantêm constante e igual a 120 Hz. As respectivas Largura de Banda também variam ao longo do tempo.

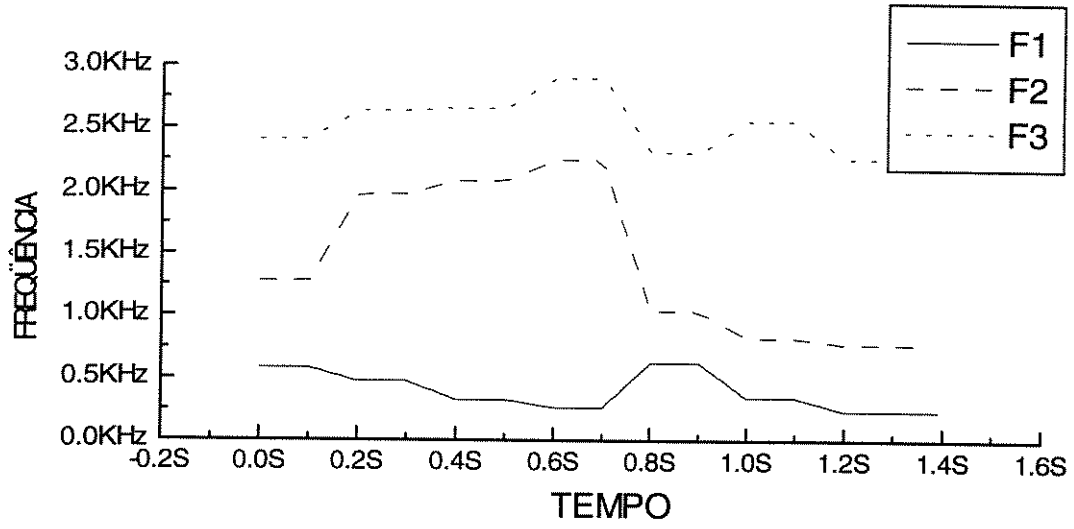


Figura 8-1 Variação da freqüência dos três primeiros formantes no exemplo de síntese de vogais

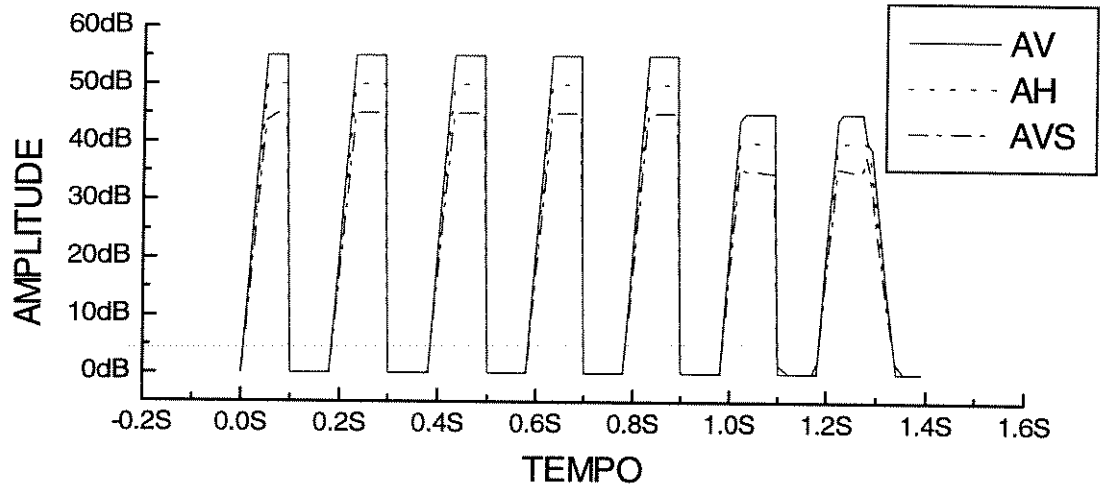


Figura 8-2 Variação das amplitudes AV, AVS e AH no exemplo de síntese de vogais

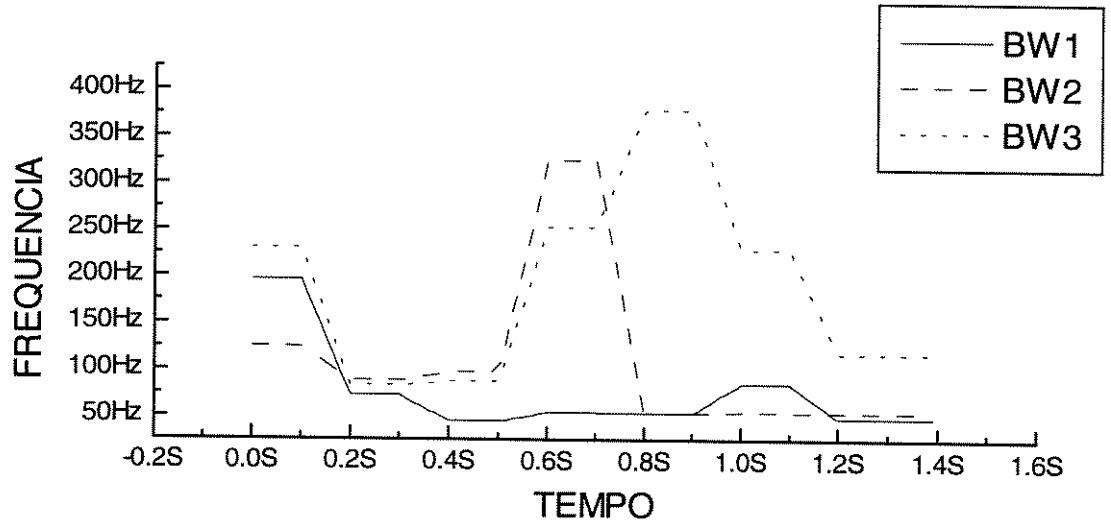


Figura 8-3 Variação da largura de banda no exemplo de síntese de vogais

8.2.2 Análise das Vogais Sintetizadas

A ferramenta básica para análise é o espectrograma de banda larga e a análise LPC, para estimarmos o valor da freqüência e sua largura de banda. Análise de pitch e análise de fourier também são ferramentas extremamente importantes.

0 espectrograma caracteriza-se por um padrão de claro-escuro onde se alternam manchas, lacunas e estriações. O espectrograma difere dos espectros lineares por levar em conta o fator tempo. O tempo é representado na abcissa e a freqüência na ordenada. A amplitude é representada por tons de cinza: As componentes mais intensas são mais escuras, as menos intensas são claras, e a ausência de energia acústica expressa-se por uma área branca. Obviamente, esse método não é muito preciso, mas o suficiente para captar as variações de amplitude mais importantes da fala. 0 espectrograma de banda-estreita, possuí uma melhor resolução em freqüência, já o espectrograma de banda-larga, possui uma melhor resolução temporal proporcionando uma melhor visualização das formantes, pois apresenta manchas escuras bem nítidas na região destes, embora não se possam distinguir os harmônicos. Tais manchas significam, que há uma concentração de energia acústica em dadas áreas de freqüência.

Uma comparação simples pode ser feita para as vogais sintetizadas. Para tanto iremos utilizaremos os espectrogramas de faixa larga das vogais pronunciadas pelo autor, sintetizadas pelo TMS320C25 e sintetizadas por um sintetizador de formantes de Klatt em ponto flutuante, implementada em um microcomputador do tipo IBM-PC.

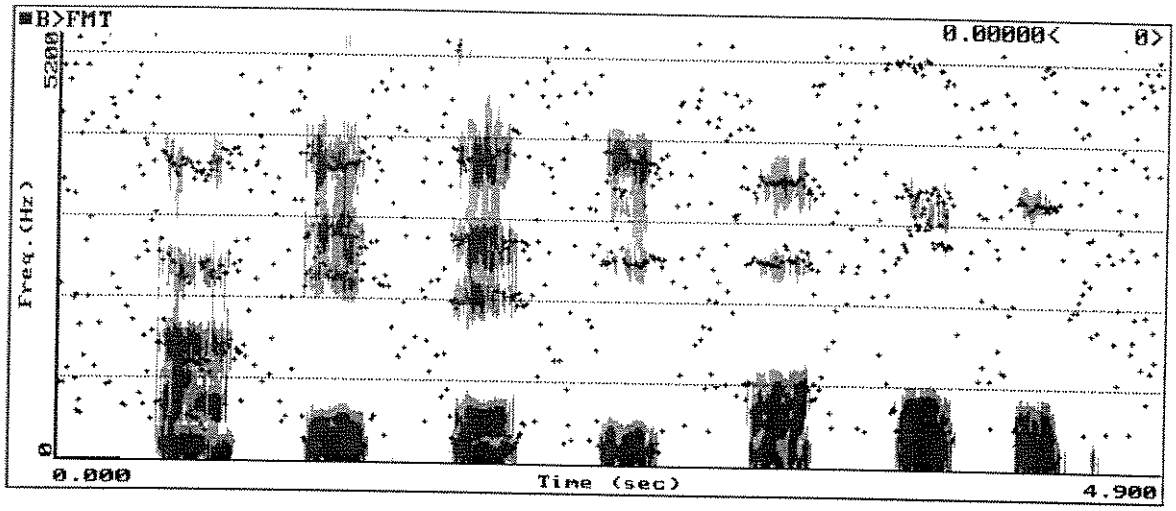


Figura 8-4 Espectrograma de banda larga das vogais a, é, ê, i, ó, ô ,u pronunciadas pela autor

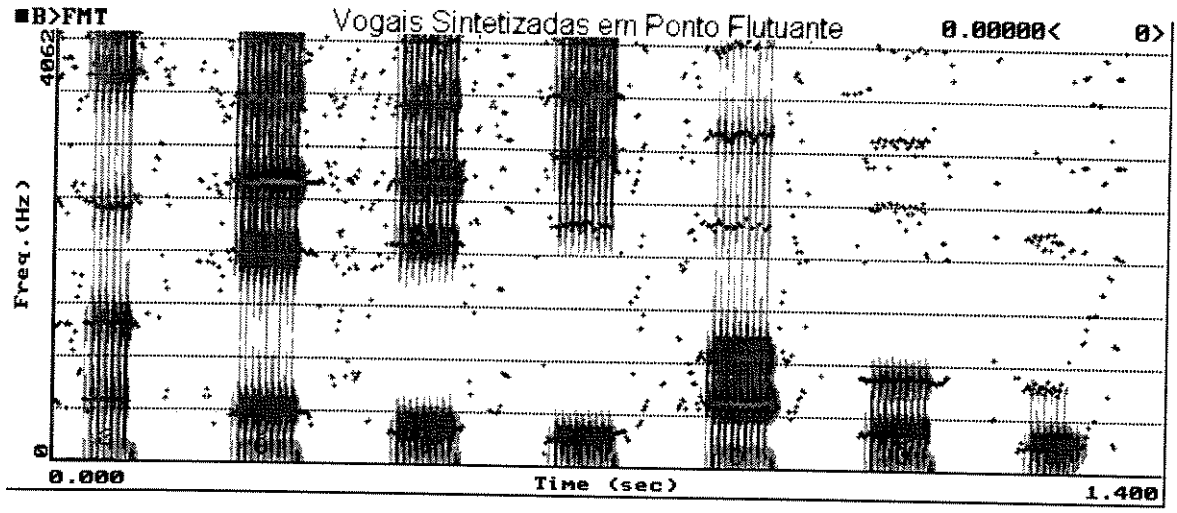


Figura 8-5 Espectrograma de banda larga das vogais a, é, ê, i, ó, ô ,u sintetizadas em ponto-flutuante

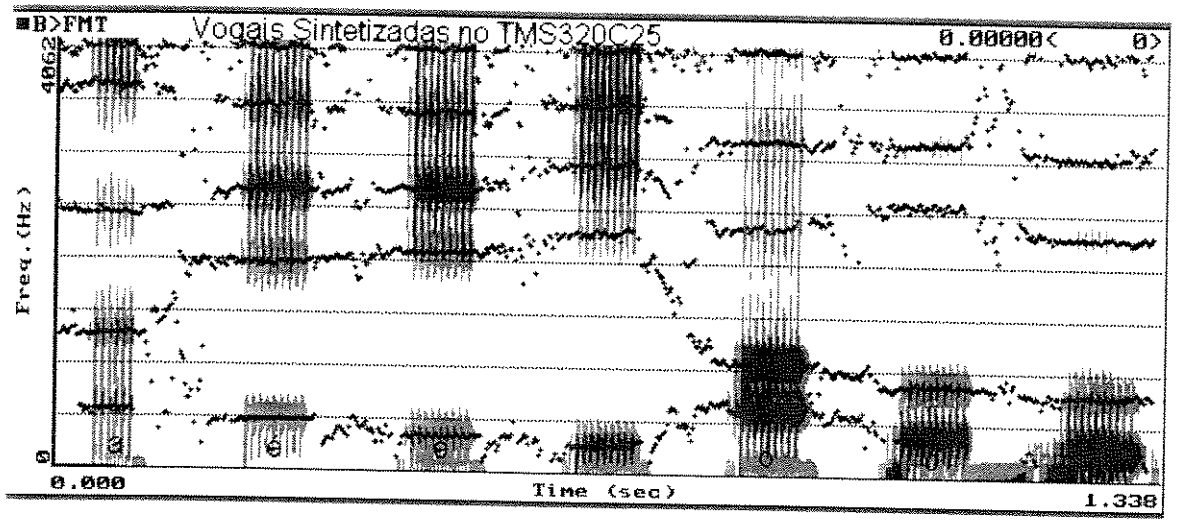


Figura 8-6 Espectrograma de banda larga das vogais a, é, ê, i, ó, ô ,u sintetizadas com TMS320C25.

Comparando-se a Figura 8-4 e Figura 8-5 e a Figura 8-6, nota-se que as áreas escuras representam intensidades levemente diferentes mas posições muito próximas. Para se extrair com precisão o valor de freqüência e largura de banda utiliza-se a análise LPC de 12 pólos. Para cada espectrograma foi feita uma análise LPC de tempo-curto. Assim é possível visualizarmos melhor o deslocamento das formantes durante o tempo.

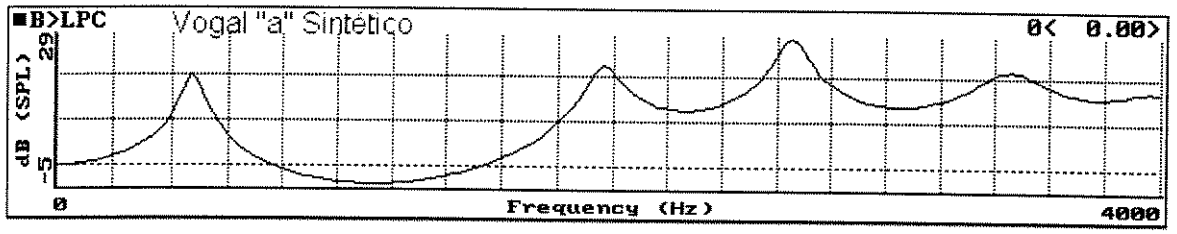


Figura 8-7 Análise LPC da vogal [a] sintetizada com TMS320C25

Na Figuta 8-8 é possível observarmos o deslocamento das formantes através da posição das linhas contínuas (Análise LPC).

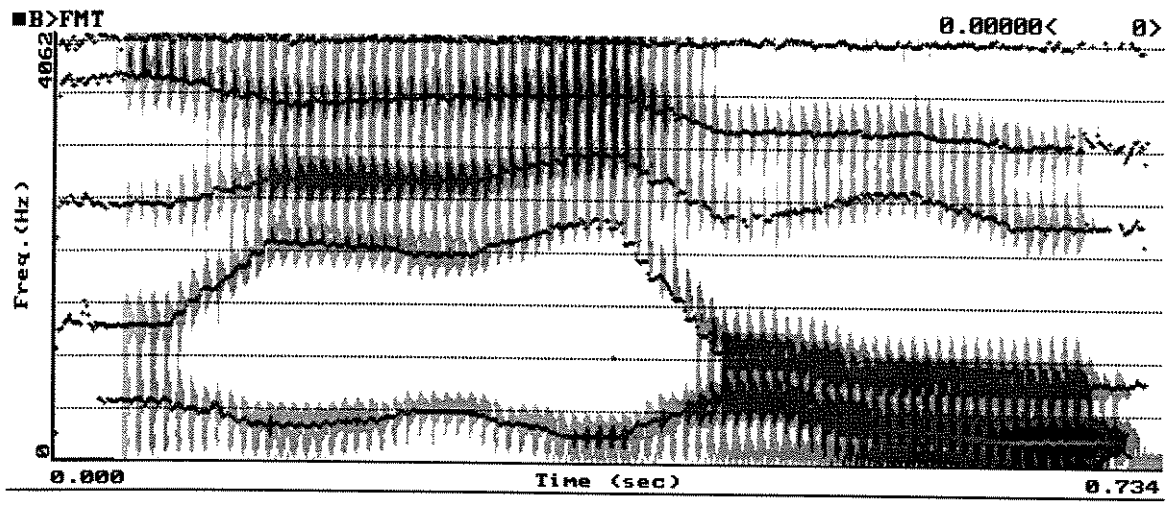


Figura 8-8 Vogais sintetizadas com TMS320C25 sem pausa entre elas.

8.2.3 Variações da Freqüência Fundamental

A variação da freqüência fundamental não altera o valor da posição das formantes. Entretanto, a variação de F0 altera a amplitude do sinal de fala sintetizado. Como exemplo, foi sintetizada a vogal [a] variando-se o pitch durante a síntese.

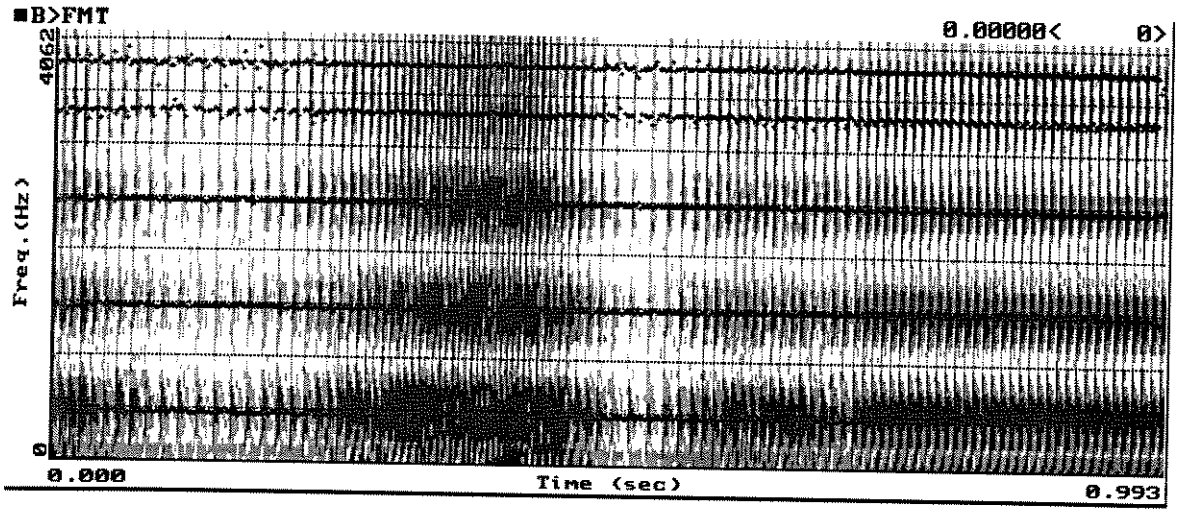


Figura 8-9 Espectrograma da vogal [a] sintetizada com variações de pitch.

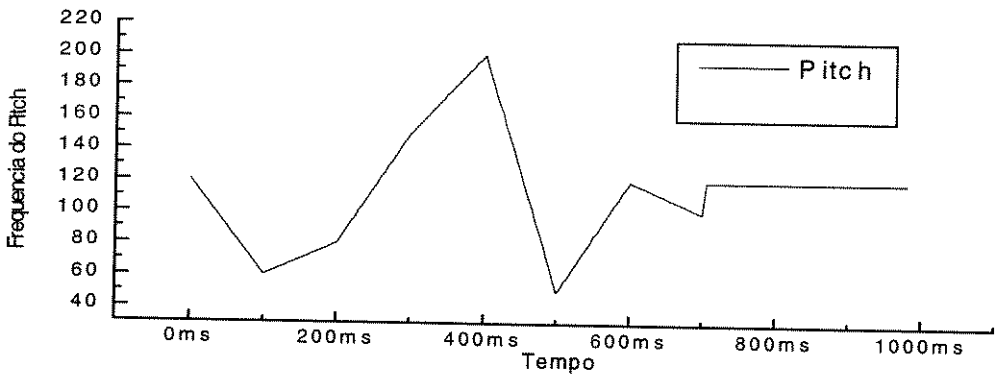


Figura 8-10 Variação da freqüência fundamental F0, vogal [a] na Figura 8-9

Pode-se observar que a intensidade do sinal produzido aumenta proprorcionalmente ao valor do pitch. Isto está de acordo com o que foi programado uma vez que o gerador de impulsos possui uma modulação relacionada a freqüência de pitch. Entretanto como deveria ser esperado a freqüência dos formantes não varia ao longo do tempo.

8.3 Síntese de Consoantes

A determinação dos parâmetros de controle para consoantes em qualquer ambiente fonético está fora do escopo deste material, mas valores apropriados podem sem encontrados, por tentativa e erro, para equivalerem à fala natural. Os valores adotados para o teste do sintetizador foram os sugeridos por Nagle na Tabela 8-3, e por Klatt nas Tabela 8-4 e Tabela 8-5. Se uma vogal for precedida por uma consoante, controle adicionais dos parâmetros podem ter que ser variados. A Tabela 8-4 incluí valores marcados para os parâmetros de controle variáveis que são usados para a síntese de consoantes da língua inglesa (espectro de rompimento para oclusivas, e murmúrios nasais).

Fricativa	[s]	[z]	[x]	[j]	[f]	[v]
AV=AVS=AH	0	40	0	40	0	0-40
AF	60	50	60	50	40-60	40-50
AB	0	0	0	0	50	50
A2	0	0	30	30	0	0
A3	0	0	40	40	0	0
A4	30	30	50	50	0	0
A5	50	50	40	40	0	0
A6	50	40	30	30	0	0
F1	250	250	450	350	300	300
F2	1250	1250	2000	1650	1000	1000
F3	2700	2700	2700	2600	2000	2000
F4	4000	4000	3400	3400	3400	3400
F5	5600	5600	5600	5600	5600	5600
F6	6700	6700	6700	6700	6700	6700
B1	200	70	120	70	200	60
B2	80	60	100	80	120	90
B3	200	180	250	150	150	120
B4	250	250	250	250	250	250
B5	600	600	600	600	600	600

B6	500	500	500	500	500	500

Tabela 8-3 Estimativas dos valores dos parâmetros de controle do sintetizador de formantes de Klatt ativados na síntese de consoantes fricativas do português (Nagle - agosto 1991)

Não fricativa	F1	F2	F3	B1	B2	B3	A2	A3	A4	A5	A6	AB
[TSH]*	350	1800	2820	200	90	300	0	44	60	53	53	
[DZH]*	260	1800	2820	60	80	270	0	44	60	53	53	0
Oclusiva												
[p]	400	1100	2150	300	150	220	0	0	0	0	0	63
[b]	200	1100	2150	60	110	130	0	0	0	0	0	63
[t]	400	1600	2600	300	120	250	0	30	45	57	63	0
[d]	200	1600	2600	60	100	170	0	47	60	62	60	0
[k]	300	1990	2850	250	160	330	0	53	43	0	45	0
[g]	200	1990	2850	60	150	280	0	53	43	45	45	0

Tabela 8-4 Valores dos parâmetros de síntese de consoantes em inglês sugeridos por Klatt

Nasal	FNP	FNZ	F1	F2	F3	B1	B2	B3
[m]	270	450	480	1270	2130	40	200	200
[n]	270	450	480	1340	2470	40	300	300

Tabela 8-5 Valores dos parâmetros para síntese de consoantes nasais em inglês sugeridos por Klatt

Os parâmetros que são usados para gerar um murmúrio nasal incluem a freqüência de pólo e de zeros nasais FNP e FNZ. O pólo e zero nasais são usados primeiramente para aproximar a nasalização de vogais na versão nasal pela divisão de F1 em um pólo-zero-pólo complexos. A nasalização das vogais tem sido utilizado para ser perceptivelmente importante. Uma vogal nasalizada é gerado pelo incremento de F1 em algo em torno de 100 Hz e por setar uma freqüência de zero nasal para ser a média deste novo valor de F1 e 270 Hz (a freqüência fixada de um pólo nasal).

8.4 Síntese de uma nova expressão

O primeiro passo na preparação de uma nova expressão é obter o modelo natural. A disponibilidade de uma expressão falada é importante pois experiências tem mostrado que nem todos os valores dos parâmetros de controle da síntese podem ser deduzidos a partir de considerações teóricas, e uma expressão sintética soará não natural sendo pouco inteligível se geralmente repousar inteiramente na teoria disponível.

A expressão falada é filtrada em um filtro passa-baixa de 5 KHz, digitalizada a 10.000 amostras por segundo, e salva em um arquivo para comparações subsequentes diretas com a imitação sintética que foi criada.

Será utilizado como expressão falada a frase: - *O vaso cheio já se foi.* A escolha dos elementos fonéticos nesta frase não é aleatória. Observando a Tabela 8-3 nota-se que todos os sons fricativos do português estão presentes. Re-escrevendo a frase à partir dos sons teremos: - O [v]a[z]o [x]eio [j]á [s]e [f]oi.

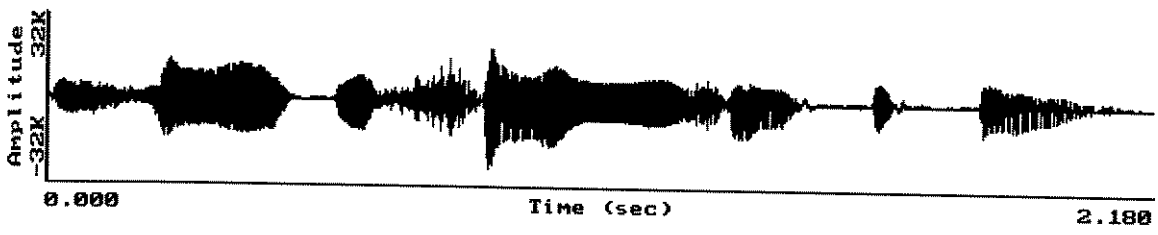


Figura 8-11 Expressão falada, *O vaso cheio já se foi*

Um espectrograma de faixa larga de uma expressão natural pronunciada é então produzida com o objetivo de visualizar a característica geral acústica da sentença e determinar a duração aproximada de seus componentes. Análise computacional descrita abaixo pode prover muita da informação redundante, mas é mais fácil visualizar as relações de freqüência, tempo e intensidade nas gravações se o espectrograma for disponível.

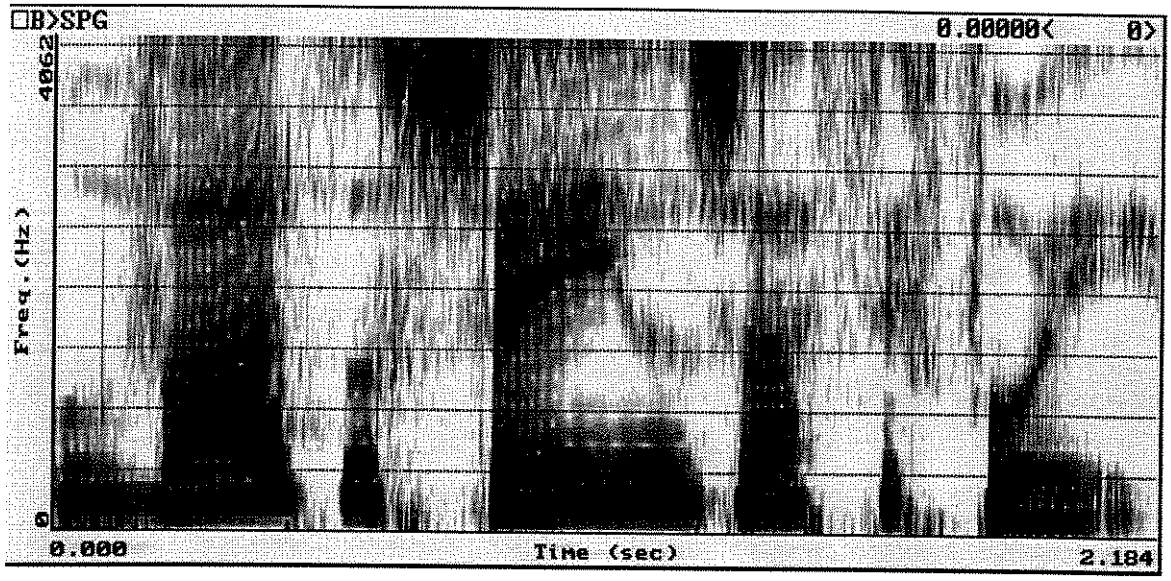


Figura 8-12 Espectrograma da expressão natural *O vaso cheio já se foi*

A expressão a ser sintetizada deverá ter a mesma intensidade do modelo, embora não seja fácil de se deduzir valores para os vários controles de amplitude que resultariam neste contorno numa primeira tentativa.

A porção de voz na palavra a ser sintetizada será analisada profundamente por rotinas nos computadores que extraírão uma estimativa da freqüência fundamental da voz (Análise de extração do Pitch) e as trajetórias das freqüências dos formantes (Análise LPC). O movimento dos formantes podem ser usados diretamente para especificar parâmetros de controle da freqüência dos formantes F1 a F5 durante as porções sonoras da expressão (somente quatro formantes são vistos abaixo de 5 KHz para muitos locutores femininos, no qual o parâmetro de controle NFC que controla o número de formantes no ramo em cascata do sintetizador deverá ser setado para 4).

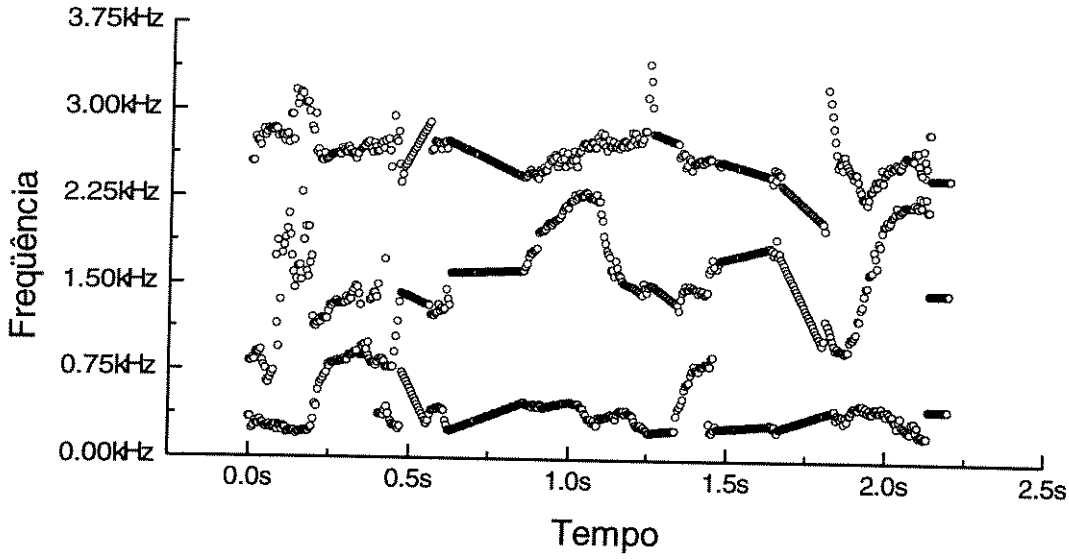


Figura 8-13 Freqüência dos três primeiros formantes extraídos da expressão natural *O vaso cheio já se foi*

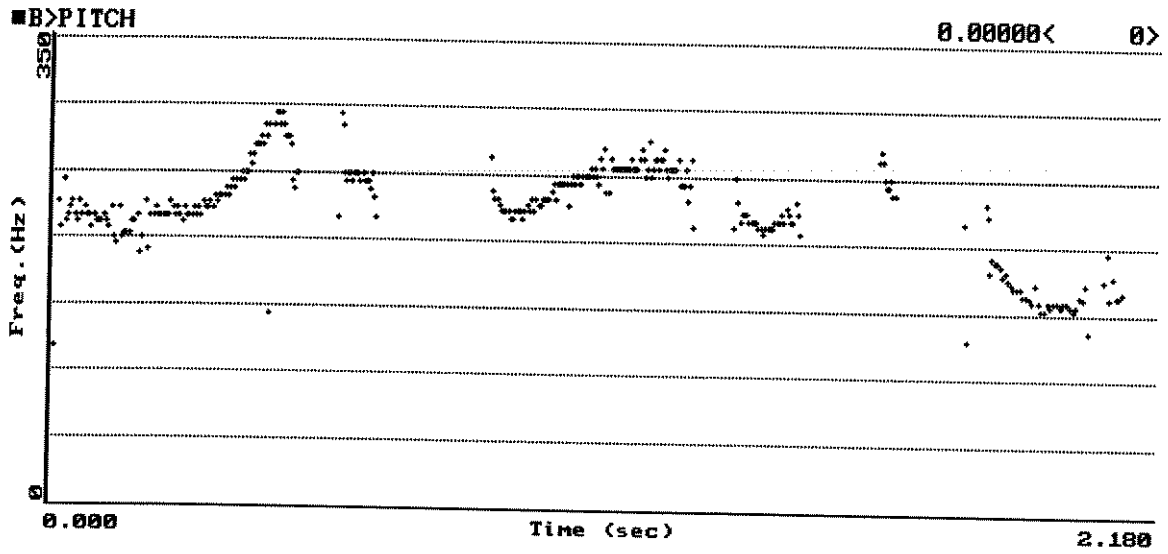


Figura 8-14 Freqüência fundamental extraída da expressão natural *O vaso cheio já se foi*

Amostras do espectro de predição linear em vários tempos da expressão natural são utilizados. O espectro médio para ruídos fricativos [s, por exemplo] poderá ser obtido usando-se uma janela ponderada tendo uma duração efetiva de 40 ms. A janela de longa duração prove uma melhor estimação do espectro estacionários.

Uma expressão sintética inteligível será uma imitação conveniente do locutor original.

Exames detalhados do espectro de Fourier dos intervalos sonoros podem indicam a presença de algum ruído de aspiração para este locutor, isto é, o espectro será menos que harmônico em altas freqüências. Além disto, AH, poderá ser ajustado para seguir o mesmo contorno de AV, mas com um valor de 3 dB abaixo. Este produzirá um pouco de qualidade de voz respirada, típica de muitos locutores. Sons sonoros podem se tornar desconfortáveis sons mecânicos, mas um pequeno ruído de aspiração quebra a regularidade da estrutura harmônica.

Por exemplo, para sintetizar o espectro observado de [s] é necessário produzir um forte pico espectral próximos a 5 KHz. Para fazer isto, F6 pode ser posicionado em 4,9 KHz e é ligado no momento apropriado. Valores default são escolhidos para as freqüências dos formantes e larguras de banda são ajustados na base de tentativa e erro com o objetivo de repetir o espectro de [s].

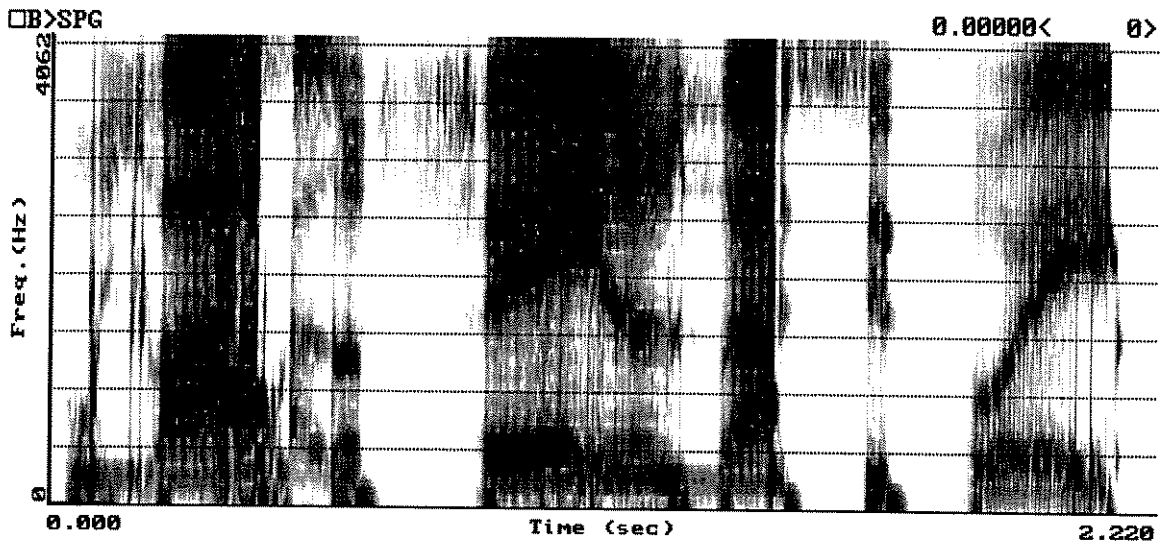


Figura 8-15 Espectrograma da expressão sintetizada *O vaso cheio já se foi*

9. Conclusões

Na implementação do sintetizador de formantes de Klatt, conceitos básicos do processamento digital de sinais foram estudados e revisados. Esta revisão foi útil pois em qualquer aplicação futura do TMS320C25 ela será necessária.

A implementação completa do sintetizador de Klatt, por ser um modelo muito genérico, apresentou alguns problemas que poderiam facilmente ser eliminados se uma implementação mais específica fosse feita. Por exemplo, levantando os possíveis parâmetros para a língua portuguesa e otimizando os ressonadores para uma faixa limitada. O fato dos ressonadores trabalharem com uma faixa muito grande de freqüências e larguras de banda fazem com que a simulação para o possível valor de escalonamento possa por vezes ser muito grande comparado com o valor de escalonamento ideal para um dado idioma.

O software implementado em ponto fixo no TMS320C25 poderá ser implementado em qualquer equipamento desde que as devidas modificações na linguagem sejam feitas.

A síntese de vogais é extremamente simples mas a síntese de componentes não sonoras merece um estudo mais profundo. O sintetizador foi implementado utilizando o modelo completo de Klatt.

O TMS320C25 possui um ciclo de instrução de 100 ns. Para uma implementação em tempo real, uma síntese de 10.000 amostras por segundo deveria ser implementada com 100 μs por amostra. Assim deveríamos poder calcular uma amostra com no máximo 1000 ciclos de instrução. Se não considerarmos as instruções de entrada/saída de dados, e utilizarmos um modelo simplificado do sintetizador, poderemos provavelmente implementar a síntese em tempo real. Para garantirmos isto, algumas instruções de comparação no sintetizador podem ser eliminadas, como por exemplo a

determinação do número de formantes da configuração utilizada, e a eliminação do 6º ressonador do ramo em cascata. O sintetizador seria alimentado somente com o valor de pitch (F0), o valor de freqüência (Fn) e largura de Banda dos três primeiros formantes (Bwn). A freqüência do pólo nasal (FNZ), também será alimentada no sintetizador juntamente com as amplitudes AV, AVS e AH. AF seria somado as amplitudes A2..A6 e a amplitude AB.

As análise do sinal da fala natural foram efetuadas com o auxilio do CSL. Estudos poderão ser feitos sobre a viabilidade de se implementar um sistema de análise LPC de 12 pólos, um extrator de pitch, e transformada de Fourier utilizando o TMS320C25. Assim, talvez seja possível implementar um sistema de análise/síntese totalmente baseado no TMS320C25.

A síntese em ponto fixo com TMS320C25 é auditivamente muito parecida com a síntese em ponto flutuante realizada em um computador de uso-geral, embora apresente uma concentração de energia maior em certas freqüências. A correção deste problema é a redução das amplitudes de determinados formantes seguindo os critérios apresentados no capítulo 3 e por Klatt.

Um estudo mais profundo da fonética acústica torna-se necessário para a operação otimizada do sintetizador. Este trabalho pode ser comparado, com as devidas restrições, com a construção de um piano ou de um carro de fórmula um. O projeto foi feito; e as condições básicas para o controle foram dadas. Uma operação mais avançada no entanto exigiria um bom pianista ou um excelente piloto de corridas.

Com a exposição deste trabalho chegamos às condições básicas para que possamos desenvolver futuros projetos utilizando o sintetizador de maneira mais ampla e enriquecida. Poderemos por exemplo, montar um coral eletrônico, passando por exemplo os parâmetros de dois cantores e o próprio TMS320 organizaria as vozes, modificando por exemplo as oitavas, ou como foi dito montar um sistema de codificação/decodificação de voz totalmente baseado no TMS320C25.

A. Apêndice - Desenvolvimento do Programa

A.1 Processo de Desenvolvimento do Software

O Processo geral de desenvolvimento segue o fluxograma mostrado na Figura A-1

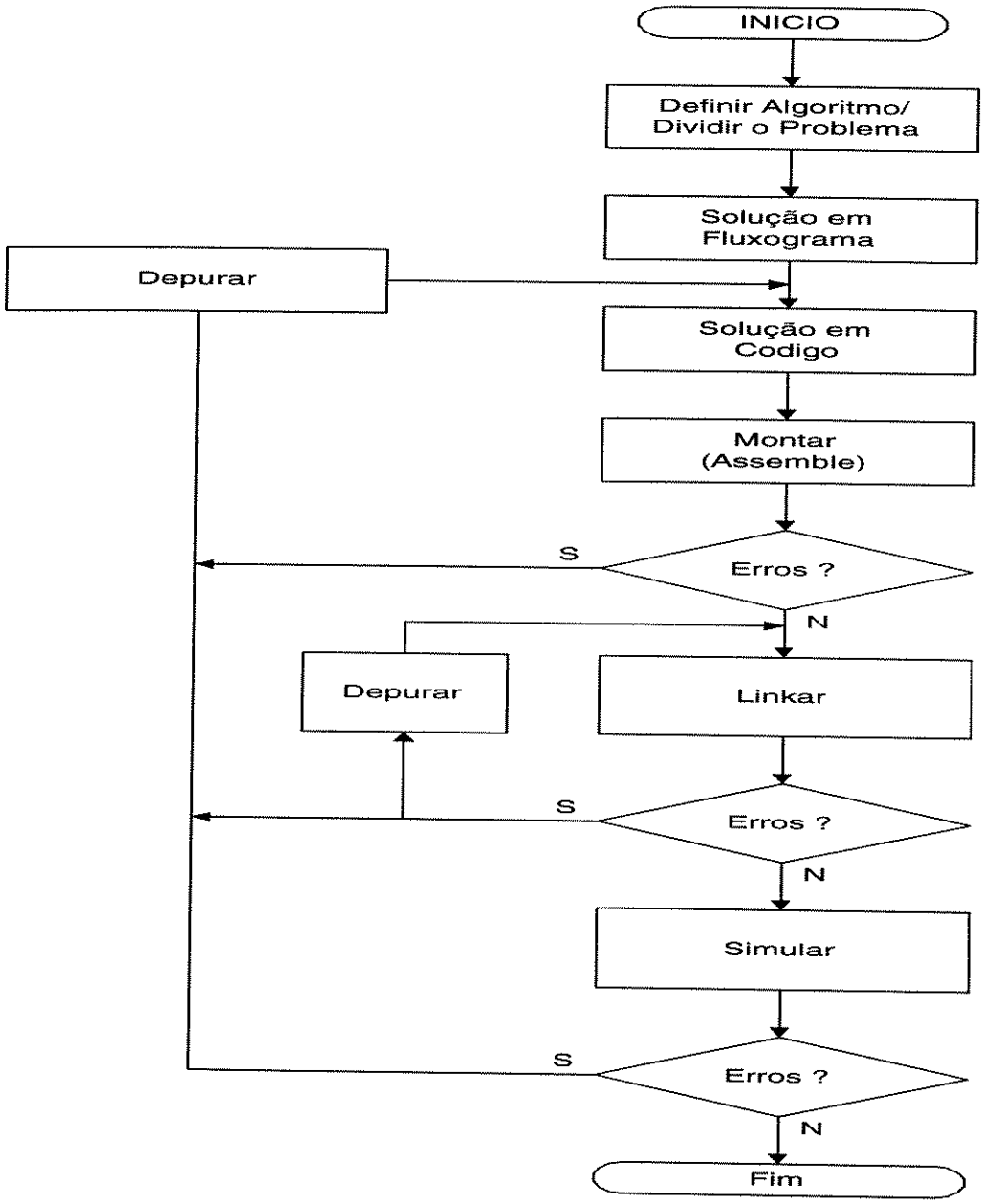


Figura A-1 Algorítmo para Desenvolvimento do software

A.2 Ambiente COFF

Num esforço para normalizar o processo de desenvolvimento do software, TI tem empregado o Formato Comum do Arquivo Objeto (COFF - Common Object File Format). COFF tem várias características nas quais fazem dela uma estrutura útil e poderosa no desenvolvimento do software. É muito útil quando a tarefa de desenvolvimento é dividido entre vários programas. O processo de desenvolvimento consiste de várias tarefas Figura A-2

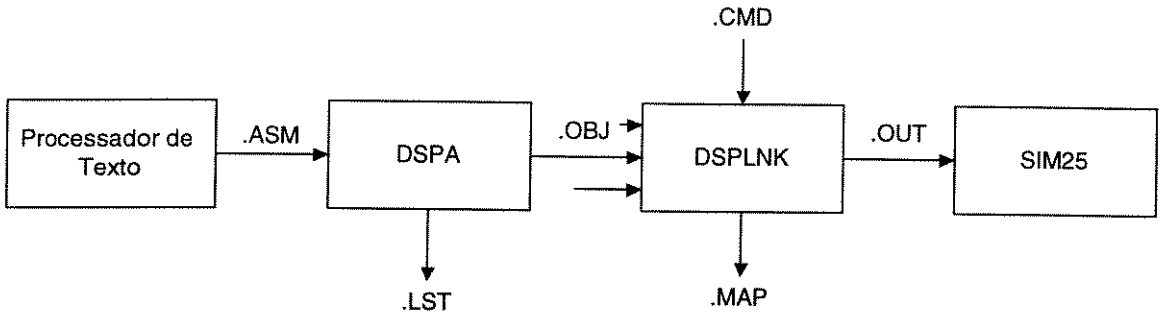


Figura A-2 Processo de desenvolvimento do software no ambiente COFF

Cada seção de código, chamado módulo, pode ser escrita independentemente, incluindo a especificação de todos os recursos necessários para uma operação correta do módulo. Módulos são escritas em nível de mnemônicos assembler usando-se qualquer editor de texto capaz de prover uma saída de arquivo ASCII. A extensão esperada do arquivo fonte é .ASM que se entende por assembly.

Depois disto, cada um destes módulos é assemblado para traduzir o código a nível de mnemônico para uma representação binária a qual é reconhecida pelo TMS320. Uma listagem é também produzida para documentar os resultados assemblados.

Numerosos módulos podem ser juntados para formar um programa completo. O Linker é uma ferramenta capaz de eficientemente alocar os recursos disponíveis no TMS320 para cada módulo no sistema. O Linker é

capaz de consultar um arquivo de comandos (.CMD) o qual identifica todos os arquivos de entrada e saída, os recursos disponíveis no TMS320, e onde irão as várias seções de cada modulo. Saídas do processo de linkagem podem incluir o arquivo linked-object (.OUT), o qual roda no TMS320 e um arquivo .MAP o qual identifica onde cada seção foi locada

O alto nível de modularidade e portabilidade resultantes deste sistema simplifica o processo de verificação, correção e manutenção. O processo de desenvolvimento COFF é apresentado em detalhes nos seguintes parágrafos.

A.3 Diretivas e Sintaxe Assembler

O formato para uma instrução ou diretiva de linguagem assembly é:

[<label>[:]] <mnemônico> [lista de operandos] [;comentários]

O <label> é opcional, mas se usado, deverá começar na coluna 1. Os dois pontos (:) seguintes ao<label> é opcional; o assembler irá ignora-lo.

O <mnemônico> é um mnemônico da instrução de linguagem assembly, uma diretiva assembler, ou um nome de uma macro.

Um <comentário> pode aparecer em qualquer lugar de uma linha se este for precedido por ponto e virgula (;). Além disso, se um asterisco (*) estiver na coluna 1, a linha inteira é tratada como um comentário

A.4 Constantes

O assembler suporta as seguintes constantes:

TIPO	EXEMPLO
Inteiros Decimais	4321 ou +4321/-4321 (default é +)

Inteiros Hexadecimais 34ACh ou 37ACH ou 0x37AC. Uma constante Hexadecimal não pode começar com uma letra, comece com 0 antes; por exemplo use 0ACh no lugar de Ach. O formato 0x37AC é o mesmo suportado na linguagem de programação C.

Alem destes formatos temos: Inteiros Binários, Inteiros Octais, Constantes de ponto-flutuante, string de caracteres e de constantes.

A.5 Símbolos

Símbolos são string com até 32 caracteres (A-Z,a-z,0-9,$e_). Um símbolo não pode ser iniciado com um numeral 0 a 9. o sinal de dólar ($) representa o valor do contador de localização da seção corrente.

O assembler tem predefinido ou reservado o seguinte:
Nome dos registradores: AR0 a AR7
Endereços das portas: PA0 a PA15

A.6 Algumas diretivas Básicas do Assembler

A.6.1 . title

Esta diretiva permite a você ter um título contendo até 50 caracteres impressos no cabeçalho de cada pagina da listagem. O título deverá entre aspas duplas. Por exemplo:

.title "Programa de Klatt"

A.6.2 .option

Esta diretiva permite habilitar opções para a listagem de saída do assembler. Por exemplo:

.option x

irá produzir uma referencia cruzada dos símbolos no final da listagem. Outras opções são disponíveis. Veja Assembly Language Tools User's Guide para detalhes.

A.6.3 .set

Use esta diretiva para definir uma constante assembly. O valor da constante irá para o campo do valor do operando. O label na diretiva .set pode ser usado em qualquer lugar onde uma constante é válida. A diretiva .set também pode ser usada para definir um novo nome para um registrador. Por exemplo:

```
IDADE       .set   20h    ;define uma constante "IDADE" igual a 20h
CONTADOR .set   AR0   ;define "CONTADOR" para ser o ;mesmo que AR0
```

A.6.4 .global

Esta diretiva declara que um símbolo é para ser definido como um símbolo global. O símbolo é definido no modulo, ou faz referencia no modulo e definido fora. O linker encontra todas as referencias para símbolos globais. Por exemplo:

```
        .global INICIO      ;faz INICIO um símbolo global
INICIO .set $               ;INICIO tem o valor da localização do contador
                            ;da seção corrente
```

A.6.5 .end

Esta diretiva deve ser a ultima diretiva em um arquivo fonte de linguagem assembly

A.7 Diretivas para Definir Seções

O menor bloco relocável de código no arquivo fonte é chamada de seção. Todo código e dados no arquivo fonte são assemblados dentro das seções como uma diretiva assembler. Em geral, seções podem aparecer em qualquer ordem e podem ser repetidas se necessário.

Estas diretivas permitem você a:

Montar o código e dados na especifica seção inicializada.

Reservar espaço na memória para variáveis não inicializadas.

A.8 Seções sem nome

A.8.1 .text

Esta é a seção defaut, que usualmente contem código fonte executável. A localização física do programa na memória é decidido durante o processo de linkagem como especificado no arquivo de comandos do linker. Como a localização é especificada irá ser explicado mais tarde quando discutiremos o arquivo de comandos do linker.

A.8.2 .data

Esta seção usualmente contém variáveis de dados inicializados dentro da memória do sistema. A diretiva .data diz ao montador que os dados seguintes são para residir na memória de programa na seção .data como definido no arquivo de comando do linker. Por isso, a diretiva .data é usada para dizer ao montador que os valores dos dados seguintes a diretivas são para pre-inicializar valores. Como a localização de memória para o inicio da seção .data é especificada será discutida mais tarde.

A.8.3 .bss

Esta seção é usualmente usada para reservar espaço na memória do sistema para variáveis não inicializados nas quais, sobre controle do programa, são eventualmente inicializados. A sintaxe para a diretiva .bss é:

.bss <símbolo>,<tamanho>

O <símbolo> aponta para a primeira localização reservada pela invocação da diretiva .bss; isto é, ele corresponde ao nome da variável para o qual você

esta reservando espaço. O <tamanho> é um numero de palavras a reservar. A seção .bss é referida por apenas um ponto inicial pelo linker. Isto significa que o local na memória de dados é seqüencial e relativo para somente um endereço como especificado no arquivo de comandos do linker. Ao contrario desta diretiva temos a diretiva .usect.

A.9 Nomes de seções definidas pelo usuário

A.9.1 .sect

Esta diretiva é usada para definir um nome de seção na memória inicializada. Uma vez um nome definido para uma seção, a seção pode ser linkada para iniciar em uma especifica localização de memória. A sintaxe é:

[label] .sect "<nome da seção">

O <nome da seção> poderá ter no máximo 8 caracteres entre aspas ("). Pode-se definir um nome de seção usando .sect no meio de qualquer código fonte para as seções .text ou .data. Se um label aparece, seu valor é o endereço de inicial da seção.

A.9.2 .usect

Esta diretiva é usada para definir uma seção na memória não inicializada, usualmente para manter valores de dados na RAM. A sintaxe é:

<label> .usect "<nome da seção>",<tamnho>

O <nome da seção> define o nome da seção; é limitado a 8 caracteres. O <tamanho> aloca um numero de palavras. O <label., o qual é requerido pela diretiva .usect, é determinado no montador como o endereço do primeiro valor na tabela.

A diretiva .usect é similar a .bss em que é usada para reservar espaço de memória não inicializada no TMS320. Contudo, a diretiva .usect contem um

nome de seção cuja localização é determinada pelo arquivo de comando do linker.

A.9.3 .asect

Esta diretiva é usada para definir uma seção absoluta. Sua sintaxe é;

[label] .asect "<nome da seção>'", <endereço>

O <label> e o <nome da seção> tem o mesmo significado que em .sect e .usect. O <endereço> é usado para dizer ao montador o endereço em qual endereço iniciar a montagem. A diretiva é usada para montar código que é para ser executado a partir de uma localização conhecida. O endereço dos labels em uma seção .asect são absolutos com respeito ao endereço inicial da seção.(Eles não podem ser relocados). O linker pode alocar o inicio de uma seção .asect em qualquer lugar da memória de acordo com a diretivas das seções.

Um típico uso de .asect é montar código que é para ser executado fora da RAM interna, mas armazenado na ROM externa. O código deve ser copiado sobre o controle do programa ou DMA antes da execução do código.

A.10 Reservando área de memória RAM

Para se reservar espaço de memória na área de memória RAM (não inicializada), use a diretiva .bss. Por exemplo:

```
bss     matriz, 10*3
```

reserva 30 palavras de memória não inicializada e acessa o símbolo matriz como a primeira localização da tabela. O linker faz a referencia a localização específica de memória para matriz (Esta ação é chamada união [BINDING]).

A.11 Definindo valores para tabelas na ROM

Para definir uma tabela de dados residente na ROM, use a diretiva .data para dizer ao assembler para colocar valores na seção .data. Recorde-se que .data é geralmente usado para inicializar-se a memória.

Para montar valores inteiros de 16 bits na memória (sem considerar a seção), use a diretiva . word. Por exemplo:

```
.word 1,2,3
```

coloca valores de 16 bits para as constantes 1,2,3 dentro de três localizações consecutivas de memória.

Alternativamente, você pode reservar bits e inicializar uma tabela toda com zeros usando a diretiva .space. Por exemplo:

```
.space 30*16
```

é equivalente a 30 diretivas .word.

A.12 Arquivo de Comando do Linker

O arquivo de comando do linker (*.CMD) pode conter informações para o linker

- Arquivos Objetos a linkar
- Nome a ser dado ao arquivo .map, o qual a lista os endereços de todas as seções de entrada e saída, bem como símbolos e labels usados no link.
- Nome a ser dado ao arquivo de saída (.OUT), o qual contém código objeto executável pelo TMS320.(Se nenhum nome for dado ao arquivo .OUT, o nome a.out será dado)
- Descrição de memória do sistema destino
- União de áreas lógicas assembly para alocar o destino das localizações de memória

O arquivo de comando do linker pode realizar outras tarefas, mas somente as tarefas listadas acima são essenciais para a maioria das aplicações encontradas.

Um arquivo de comandos pode ser usado, com simples modificações, na maioria dos sistemas, bastando para isto,. redefinir o nome dos arquivos, especificar de que maneira você quer que as seções sejam linkadas, e/ou configurar a memória de maneira diferente. Alem disto, você poderá usar o modelo apresentado no apêndice que irá prover versatilidade e um mínimo de esforço para criar um ambiente desejado para linkagem.

A.13 Usando as ferramentas COFF

Agora que estamos familiarizados COFF, iremos examinar cada um dos componentes do sistema e seu uso em grandes detalhes.

A.13.1 Utilitário Conversor de Programa Fonte

Embora durante este trabalho, tenhamos trabalhado com a nova versão (5.0 ou superior) do montador, do linker etc., alguns detalhes no programa fonte foram escritos para versão anterior (3.0 ou anterior) uma vez que muito for baseado no livro SECOND-GENERATION TMS320 User's Guide. A nova versão usa o formato COFF enquanto a antiga usa o formato TI-TAG. O novo formato tem a muitas vantagens, por exemplo:

-prove grande facilidade de uso;
-faz a programação modular mais fácil;

Arquivos objetos COFF contem separadamente, blocos, ou seções de código ou dados independentes e relocáveis.

Para invocar o conversor utilizamos:

dspcv [arquivos fonte de entrada]

O utilitário conversor assume que os arquivos fonte de entrada tenham a extensão .ASM. A extensão de saída é .Ann sendo nn variando de 00 a 99 e o nome é o mesmo do arquivo de entrada.

Exemplos de conversão:

Arquivo Original	Arquivo convertido
BYTE constante	.byte constante
AORG 123	;>>>WARNING, NO ABSOLUTE CODE ;>>>AORG 123 .sect "AORG0"
>1234	01234H
?0101	0101B
sym EQU 10 comentario	sym .equ 10 ;comentario

A diretiva COPY ARQUIVO.ASM também é convertida para .copy "ARQUIVO.ASM" mas o arquivo incluído não será convertido.

Para maiores informações consulte Assembly Language Tools User's Guide, Chapter 10

A.13.2 O montador cruzado: DSPA

O montador é um simples processo o qual é invocado pelo comando:

DSPA <arquivo de entrada> [<arquivo objeto> [<arquivo de listagem>]] [-<opções)]

O extensão default para o <arquivo de entrada> é .ASM. O defaut para o nome do <arquivo objeto> e <arquivo de listagem> é o mesmo do <arquivo de entrada> com as extensões .OBJ. e .lst respectivamente.

Algumas das <opções> disponíveis são dadas na seguinte tabela:

Opção	Ação
-v10	para 32010
-v25	para 320C25
-l	listar
-x	Referencia Cruzada
-c	Ignora Maiúsculas/minúsculas
-s	Todos os símbolos no Arquivo Objeto
-q	roda quito

uma maneira usual para se invocar o montador pode ser:

dspa teste -v25,l,,x,c

o qual irá assemblar o arquivo teste.asm e produzir os arquivos teste.obj e teste.lst. As opções -l,x especificam que o arquivo de listagem irá ser criado e que teremos uma referencia cruzada no final deste arquivo montar Note que, sem a opção -c, os símbolos "valor", "Valor" e "VALOR" não terão o mesmo significado, um detalhe que pode ser uma fonte de problemas se não for considerado.

Maiores detalhes do montador está em Assembly Language Tools User's Guide (SPRU018), Section 4.

A.13.3 O liker: DSPLNK

O linker é um poderoso, embora simples, programa de processo que é invocado pelo comando:

DSPLNK <opções> <arquivo1 ... arquivoN >

onde:

arquivo1 ... arquivoN é uma lista de todos os arquivos de entrada.

Algumas das opções do linker são descritas na seguinte tabela. Para mais informações, veja Assembly Language Tools User's Guide:

Opção	Ação
-f<valor>	Preenche áreas não usadas com <valor>
-m<nome do arquivo>	<nome do arquivo> de mapa
-o	<nome do aruivo> de saída
-q	roda sem ecoar (quiet mode)

As seguintes extensões são recomendadas:

Extensão	Tipo
.cmd	arquivo de comandos do linker
.obj	arquivo objeto assemblado
.map	arquivo de mapa da linkagem
.out	arquivo de saída da linkagem

Uma possível chamada do linker é:

```
dsplnk teste.cmd
```

É possível não se usar um arquivo de comandos para se especificar todas as entradas e saídas na chamada do linker, mas um arquivo de comandos prove vários benefícios. Usando este método permite grande controle do processo de linkagem, economiza tempo de desenvolvimento, e também prove uma melhor documentação e repetibilidade. Claro que é necessário ter-se previamente desenvolvido um arquivo de comandos para linkagem, mas isto é uma maneira simples se utilizarmos o arquivo mostrado no apêndice.

A.13.4 Utilitário de conversão: DSPROM

O XDS, o qual foi desenvolvido para uma norma anterior de programa chamada TI-TAG, não reconhece diretamente arquivos COFF. Para isto, TI desenvolveu um programa chamado DSPROM.EXE para converter arquivos COFF no formato TAG. Este conversor também pode converter para os formator EXTENDED TECKTRONIC HEX OBJECT, formato INTEL HEX OBJECT. A conversão é feita chamando:

dsprom [-opções] [Arquivo de entrada COFF [arquivo de saída1 [arquivo de saída2]]]

Opção	Formato de Saída
-x	Tecktronix hex
-i	Intel hex
-w	Intel word
-t	TI-TAG

Se nenhuma opção for dada, a saída produzida será no formato Tectronicx hex.

Se não for especificado arquivo de entrada , o conversor pedirá um.

Extensão	Tipo
.obj	Arquivo de Entrada
.tag	Formato TI-TAG
.hi	Formato Tecktronix hex ou Intel hex
.hex	Formato Inter word

Quando o utilitário termina a conversão do arquivo de entrada, é impressa a mensagem:

Translation Complete

B. Apêndice - Comandos do TMS320C25

B.1 Instruções do TMS320C25

O TMS320C25 possui um conjunto de 133 instruções. Cada instrução tem seu significado. A Tabela B-2 apresenta os comandos por ordem alfabética e as Tabela B-3, Tabela B-4, Tabela B-5, Tabela B-6 e Tabela B-7 apresentam as instruções agrupadas por funções.

SÍMBOLO	SIGNIFICADO
A	Port address
ACC	Accumulator
ARB	Auxiliary register pointer buffer
ARn	Auxiliary register n (AR0, AR1 assembler symbols equal to 0 or 1)
ARP	Auxiliary register pointer
B	4-bit field specifying a bit code
BIO	Branch control input
C	Carry bit
CM	2-bit field specifying compare mode
CNF	On-chip RAM configuration control bit
D	Data memory address field
DATn	Label assigned to data memory location n
dma	Data memory address
DP	Data page pointer
FO	Format status bit
FSM	Frame synchronization mode bit
HM	Hold mode bit
I	Addressing mode bit
INTM	Interrupt mode flag bit
K	Immediate operand field
MCS	Microcall stack
>nn	nn = hexadecimal number (others are decimal values)
OV	Overflow mode flag bit
OVM	Overflow mode bit
P	Product register
PA	Port address (PA0- PAI5 assembler symbols equal to 0 through 15)
PC	Program counter
PFC	Prefetch counter
PM	2-bit field specifying P register output shift code
pma	Program memory address
PRGN	Label assigned to program memory location n
R	3-bit operand field specifying auxiliary register
RPTC	Repeat counter
S	4-bit left-shift code
STn	Status register n (STO or ST1
SXM	Sign-extension mode bit
T	Temporary register
TC	Test control bit
TOS	Top of stack
TXM	Transmit mode bit
X	3-bit accumulator left-shift field
XF	XF pin status bit
→	Is assigned to
| |	An absolute value
< >	User-defined items
[ ]	Optional items
( )	Contents of
{ }	Alternative items, one of which must be entered
	Blanks or spaces must be entered where shown.

Tabela B-1 Símbolos das instruções

Mnemonic and Description		W	16 Bit Opcode			
			MSB			LSB
ABS	Absolute value of accumulator	1	1100	1110	0001	1011
ADD	Add to accumulator with shift	1	0000	SSSS	IDDD	DDDD
ADDC	Add to accumulator with carry	1	0100	0011	IDDD	DDDD
ADDH	Add to high accumulator	1	0100	1000	IDDD	DDDD
ADDK	Add to accumulator short immediate	1	1100	1100	KKKK	KKKK
ADDS	Add to low accumulator with sign-extension suppressed	1	0100	1001	IDDD	DDDD
ADDT	Add to accurnufator with shift specified by T register	1	0100	1010	IDDD	DDDD
ADLKT	Add to accumulator long immediate with shift	2	1101	SSSS	0000	0010
AND	AND with accumulator	1	0100	1110	IDDD	DDDD
ANDK	AND immediate with accumulator with shift	2	1101	SSSS	0000	0100
CMPL	Complement accumulator	1	1100	1110	0010	0111
LAC	Load accumulator with shift	1	0010	SSSS	IDDD	DDDD
LACK	Load accumulator short immediate	1	1100	1010	KKKK	KKKK
LACT	Load accumulator with shift specified by T register	1	0100	0010	IDDD	DDDD
LALK	Load accumulator long immediate	2	1101	SSSS	0000	0001
NEG	Negate accumulator	1	1100	1110	0010	0011
NORM	Normalize contents of accumulator	1	1100	1110	1010	0010
OR	OR with accumulator	1	0100	1101	IDDD	DDDD
ORK	OR immediate with accumulator with shift	2	1101	SSSS	0000	0101
ROL	Rotate accumulator left	1	1100	1110	0011	0100
ROR	Rotate accumulator right	1	1100	1110	0011	0101
SACH	Store high accumulator with shift	1	0110	1XXX	IDDD	DDDD
SACL	Store low accumulator with shift	1	0110	0XXX	IDDD	DDDD
SBLKT	Subtract from accumulator long immediate with shift	2	1101	SSSS	0000	0011
SFLT	Shift accumulator left	1	1100	1110	0001	1000
SFRT	Shift accumulator right	1	1100	1110	0001	1001
SUB	Subtract from accumulator with shift	1	0001	SSSS	IDDD	DDDD
SUBB	Ssubtract from accumulator with borrow	1	0100	1111	IDDD	DDDD
SUBC	Conditional subtract	1	0100	0111	IDDD	DDDD
SUBH	Subtract from high accumulator	1	0100	0100	IDDD	DDOD
SUBK	Subtract from accumulator short immediate	1	1100	1101	KKKK	KKKK
SUBS	Subtract from low accumulator with sign extension suppressed	1	0100	0101	IDDD	DDDD
SUBT	Subtract from accumulator with shift specified by T register	1	0100	01l0	IDDD	DDDD
XOR	Exclusive-OR with accumulator	1	0100	1100	IDDD	DDDD
XORK	Exclusive-OR immediate with accumulator with shift	2	1101	SSSS	0000	0110
ZAC	Zero accumulator	1	1100	1010	0000	0000
ZALH	Zero low accumulator and load high accumulator	1	0100	0000	IDDD	DDDD
ZALR	Zero low accumulator and load high accumulator with rounding	1	0111	1011	IDDD	DDDD
ZALS	Zero accumulator and load low accumulator with sign extension suppressed	1	0100	0001	IDDD	DDDD

Tabela B-2 Sumário do conjunto de instruções do acumulador e memória no TMS320C25

Mnemonic and Description		W	16 Bit Opcode			
			MSB			LSB
ADRK	Add to auxiliary register short immediate	1	0111	1110	KKKK	KKKK
CMPR	Compare auxiliary register with auxiliary register AR0	1	1100	1110	0101	OOKK
LAR	Load auxiliary register	1	0011	0RRR	IDDD	DDDD
LARK	Load auxiliary register short immediate	1	1100	0RRR	KKKK	KKKK
LARP	Load auxiliary register pointer	1	0101	0101	1000	1RRR
LDP	Load data memory page pointer	1	0101	0010	IDDD	DDDD
LDPK	Load data memory page pointer immediate	1	1100	100K	KKKK	KKKK
LRLK	Load auxiliary register long immediate	2	1101	0RRR	0000	0000
MAR	Modify auxiliary register	1	0101	0101	IDDD	DDDD
SAR	Store auxiliary register	1	0111	0RRR	IDDD	DDDD
SBRK	Subtract from auxiliary register short immediate	1	0111	1111	KKKK	KKKK

Tabela B-3 Instruções do apontador da página de dados e dos registradores auxiliares

Mnemonic and Description		W	16-Bit Opcode			
			MSB			LSB
APAC	Add P register to accumulator	1	1100	1110	0001	0101
LPHT	Load high P register	1	0101	0011	IDDD	DDDD
LT	Load T register	1	0011	1100	IDDD	DDDD
LTA	Load T register and accumulate previous product	1	0011	1101	IDDD	DDDD
LTD	Load T register, accumulate previous product, and move data	1	0011	1111	IDDD	DDDD
LTP	Load T register and store P register in accumulator	1	0011	1110	IDDD	DDDD
LTS	Load T register and subtract previous product	1	0101	1011	IDDD	DDDD
MACT	Multiply and accumulate	2	0101	1101	IDDD	DDDD
MACD	Multiply and accumulate with data move	2	0101	1100	IDDD	DDDD
MPY	Multiply (with T register, store product in P register)	1	0011	1000	IDDD	DDDD
MPYA	Multiply and accumulate previous product	1	0011	1010	IDDD	DDDD
MPYK	Multiply immediate	1	101K	KKKK	KKKK	KKKK
MPYS	Multiply and subtract previous product	1	0011	1011	IDDD	DDDD
MPYU	Multiply unsigned	1	1100	1111	IDDD	DDDD
PAC	Load accumulator with P register	1	1100	1110	0001	0100
SPAC	Subtract P register from accumulator	1	1100	1110	0001	0110
SPH	Store high P register	1	0111	1101	IDDD	DDDD
SPLT	Store low P register	1	0111	1100	IDDD	DDDD
SPMT	Set P register output shift mode	1	1100	1110	0000	10KK
SQRA	Square and accumulate	1	0011	1001	IDDD	DDDD
SORS	Square and subtract previous product	1	0101	1010	IDDD	DDDD

Tabela B-4 Registrador T, registrador P e instruções de multiplicação

Mnemonic and Description		W	16-Bit Opcode			
			MSB			LSB
B	Branch unconditiona11y	2	1111	1111	IDDD	DDDD
BACC	Branch to address specified by accumulator	1	1100	1110	0010	0101
BANZ	Branch on auxiliary register not zero	2	1111	1011	IDDD	DDDD
BBNZ	Branch if TC bit ≠ 0	2	1111	1001	IDDD	DDDD
BBZT	Branch if TC bit = 0	2	1111	1000	IDDD	DDDD
BC	Branch an carry	2	0101	1110	IDDD	DDDD
BGEZ	Branch if accumulator >= 0	2	1111	0100	IDDD	DDDD
BGZ	Branch if accumulator > 0	2	1111	0001	IDDD	DDDD
BIOZ	Branch on I/O status = 0	2	1111	1010	IDDD	DDDD
BLEZ	Branch if accumulator <=0	2	1111	0010	IDDD	DDDD
BLZ	Branch if accumulator < 0	2	1111	0011	IDDD	DDDD
BNC	Branch on no carry	2	0101	1111	IDDD	DDDD
BNV	Branch if no overflow	2	1111	0111	IDDD	DDDD
BNZ	Branch if accumulator ≠ 0	2	1111	0101	IDDD	DDDD
BV	Branch on overflow	2	1111	0000	IDDD	DDDD
BZ	Branch if accumulator = 0	2	1111	0110	IDDD	DDDD
CALA	Call subroutine indirect	1	1100	1110	0010	0100
CALL	Call subroutine	2	1111	1110	IDDD	DDDD
RET	Return from subroutine	1	1100	1110	0010	0110
TRAPT	Software interrupt	1	1100	1110	0001	1110

Tabela B-5 Instruções de branch e call

Mnemonic and Description		Words	16-Bit Opcode			
			MSB			LSB
BLKD	Block move from data memory to data memory	2	1111	1101	IDDD	DDDD
BLKP	Block move from program memory to data memory	2	1111	1100	IDDD	DDDD
DMOV	Data move in data memory	1	0101	0110	IDDD	DDDD
FORT	Format serial port registers	1	1100	1110	0000	111K
IN	Input data from port	1	1000	AAAA	IDDD	DDDD
OUT	Output data to port	1	1110	AAAA	IDDD	DDDD
RFSM	Reset serial port frame synchronization mode	1	1100	1110	0011	0110
RTXM	Reset serial port transmit mode	1	1100	1110	0010	0000
RXF	Reset external flag	1	1100	1110	0000	1100
SFSM	Set serial port frame synchronization mode	1	1100	1110	0011	0111
STXM	Set serial port transmit mode	1	1100	1110	0010	0001
SXFT	Set external flag	1	1100	1110	0000	1101
TBLR	Table read	1	0101	1000	IDDD	DDDD
TBLW	Table write	1	0101	1001	IDDD	DDDD

Tabela B-6 Operações na memória de dados e i/o

Mnemonic and Description		W	16-Bit Opcode			
			MSB			LSB
BIT	Test bit	1	1001	BBBB	IDDD	DDDD
BITT	Test bit specified by T register	1	0101	0111	IDDD	DDDD
CNFD	Configure block as data memory	1	1100	1110	0000	0100
CNFP	Configure block as program memory	1	1100	1110	0000	0101
DINT	Disable interrupt	1	1100	1110	0000	0001
EINT	Enable interrupt	1	1100	1110	0000	0000
IDLE	Idle until interrupt	1	1100	1110	0001	1111
LST	Load status register ST0	1	0101	0000	IDDD	DDDD

LST1	Load status register ST1	1	0101	0001	IDDD	DDDD
NOP	No operation	1	0101	0101	0000	0000
POP	Pop top of stack to low accumulator	1	1100	1110	0001	1101
POPD	Pop top of stack to data memory	1	0111	1010	IDDD	DDDD
PSHD	Push data memory value onto stack	1	0101	0100	IDDD	DDDD
PUSH	Push low accumulator onto stack	1	1100	1110	0001	1100
RC	Reset carry bit	1	1100	1110	0011	0000
RHM	Reset hold mode	1	1100	1110	0011	1000
ROVM	Reset overflow mode	1	1100	1110	0000	0010
RPT	Repeat instruction as specified by data memory value	1	0100	1011	IDDD	DDDD
RPTK	Repeat instruction as specified by immediate value	1	1100	1011	KKKK	KKKK
RSXM	Reset sign-extension mode	1	1100	1110	0000	0110
RTC	Reset test/control flag	1	1100	1110	0011	0010
SC	Set carry bit	1	1100	1110	0011	0001
SHM	Set hold mode	1	1100	1110	0011	1001
SOVM	Set overflow mode	1	1100	1110	0000	0011
SST	Store status register ST0	1	0111	1000	IDDD	DDDD
SST1	Store status register ST1	1	0111	1001	IDDD	DDDD
SSXM	Set sign-extension mode	1	1100	1110	0000	0111
STC	Set test/control flag	1	1100	1110	0011	0011

Tabela B-7 Instruções de controle

BIBLIOGRAFIA

1. CHASSAING, Rulph; HORNING, Darrell W., *Digital Signal Processing with the TMS320C25*, John Wiley & Sons, Inc., New York, 1990.

2. CROWELL, Charles *Floating-Point Arithmetic with the TMS32020*, Digital Signal Processing Applications with the TMS320 Family, pp 245-268, Texas Instruments Inc, 1986

3. GARCIA, Domino *Precision Digital Sine-Wave Generation with the TMS32010*, Digital Signal Processing Applications with the TMS320 Family, pp 269-290, Texas Instruments Inc, 1986

4. GOMES, Leandro de C.T.; NAGLE, Edson José; CHIQUITO José Geraldo *Interface entre Processamento de Texto e de Sinal para a Síntese de Fala por Regras*, Anais do VII Simpósio Brasileiro de microondas e optoeletrônica e XVI Simpósio Brasileiro de Telecomunicações, Curitiba, PR, Vol.1, pp 355-360, junho 1996.

5. KLATT, Dennis H. *Review of text-to-speech conversion for English*, J.Acoustical Society of America, Vol.82, 737-793, September 1987.

6. KLATT, Dennis H. *Software for a Cascade/Parallel Formant Synthesizer*, J.Acoustical Society of America, Vol.67, 971-995, March 1980.

7. KLATT, Dennis H.; KLATT, Laura C. *Analysis, synthesis, and perception of voice quality variations among female and male talkers*, J.Acoustical Society of America, Vol.87, 820-857, February 1990.

8. MAIA, Eleonora A.Mota *No reino da fala: a linguagem e seus sons*, Editora Ática S.A., São Paulo, 1985.

9. NAGLE, Edson José *Síntese Inicial de Vogais e Fricativas* , Relatório final de IA350 - Estudos Especiais III, agosto 1991.

10. NAGLE, Edson José; CHIQUITO, José Geraldo *Síntese de Sinais de Fala Usando o Sintetizador de Formantes de Klatt*, 11º Simpósio Brasileiro de Telecomunicações, Natal, RN, Vol.2, pp 718-723, setembro 1993.

11. NOLL, Peter; JAYANT N.S. *Digital Coding of Waveforms Principles and Application to Speech and Video*, Prentice-Hall, Englewood Cliffs, N.J., 1984.

12. OPPENHEIM, Alan V., Editor *Applications of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

13. OPPENHEIM, Alan V.; Schafer, Ronald W., *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

14. PAPOULLIS, Athanasios *Probability, Random Variables, and Stochastic Processes.* Mc Graw-Hill, Inc., Third Edition, 1991.

15. RABINER, Lawrence; GOLD, Bernard, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewwod Cliffs, N.J., 1975.

16. RABINER, Lawrence; R.; SCHAFER, Ronald W., *Digital Processing of Speech Signals*, Prentice-Hall, Englewwod Cliffs, N.J., 1978.

17. *Second-Generation TMS320 User's Guide*, Texas Instruments Inc., Dallas, Tex., 1988.

18.SILMAR, Ray; LOVRICH, Al *Implementation of FIR/IIR Filters with the TMS32010/TMS32020*, Digital Signal Processing Applications with the TMS320 Family, pp 27-68, Texas Instruments Inc, 1986

19.TAYLOR, Fred J., *Digital Filter Design Handbook*, Marcel Dekker, Inc. New York and Basel, 1983.

20.*TMS320 Fixed-Point DSP Assembly Language Tools User's Guide*, Texas Instruments Inc., Dallas, Tex., 1990.

21.*TMS320C25 DSP Desing Workshop, Student Guide*, Texas Instruments Inc., Dallas, Tex., 1989.

22.WALDMAN, Hélio *Processamento Digital de Sinais*, I Escola Brasileiro-Argentina de Informática (I EBAI), Edição Preliminar, Editoria Kapelusz S.A. Buenos Aires, febrero, 1987.

23.*XDS/22 TMS320C2x Emulator User's Guide*, Texas Instruments Inc, Dallas, Tex, 1988.