

unesp

**UNIVERSIDADE ESTADUAL PAULISTA
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

***DISSERTAÇÃO DE MESTRADO
EM
ENGENHARIA ELÉTRICA***

1210001150



Nº 55

**REDUÇÃO DE RUÍDO EM SINAIS DE
VOZ EM TEMPO REAL USANDO O
DSP TMS320C31**

**ALUNO: LUIZ ANTONIO PEREZI MARÇAL
ORIENTADOR: JOZUÉ VIEIRA FILHO**

Te.1150



**Universidade Estadual Paulista Júlio de Mesquita Filho
Faculdade de Engenharia de Ilha Solteira
Pós-Graduação em Engenharia Elétrica**

***Redução de Ruído em Sinais de Voz em
Tempo Real usando o DSP TMS320C31***

“Dissertação apresentada à Faculdade de Engenharia de Ilha Solteira – UNESP, como parte dos requisitos necessários para obtenção do título de Mestre em Engenharia Elétrica (ME)”.

Candidato: Luiz Antônio Perezi Marçal

Orientador: Prof. Dr. Jozué Vieira Filho

Ilha Solteira, Maio de 2000.



"Redução de Ruído em Sinais de Voz em Tempo Real Usando o DSP TMS320C31"

Luiz Antonio Perezi Marçal

DISSERTAÇÃO SUBMETIDA À FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA – UNESP – COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA ELÉTRICA (ME).




Prof. Dr. Carlos Roberto Minussi – Coordenador

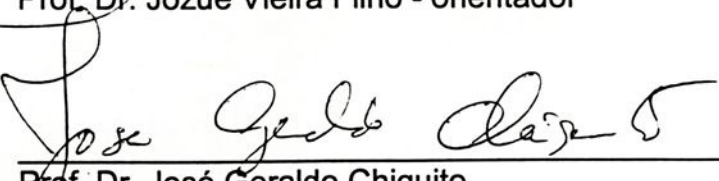
COMISSÃO EXAMINADORA:

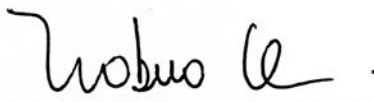
Doc. 017/2000 - N.º 095 - GISA

UNESP - "CAMPUS"	ILHA SOLTEIRA
SERVIÇO	EXAMINAÇÃO
DATA DE CRIAÇÃO	13.09.00
DATA DE ENCERRAMENTO	29.09.00
RE	aula
AC	Te. 1150
Assinatura	M313n

3040000 ENGA. ELÉTRICA


Prof. Dr. Jozué Vieira Filho - orientador


Prof. Dr. José Geraldo Chiquito


Prof. Dr. Nobuo Oki

Ilha Solteira – SP, maio de 2000

BCpIS - FEIS - UNESP

unesp

Trabalho realizado com o apoio da FAPESP (processo 98/04257-4) e da CAPES (bolsa de estudos CAPES/DS)



*À Jeová Deus, o projetista da vida
À minha esposa Adriana e às minhas filhas Driellen e Lorena*

AGRADECIMENTOS

- *A Deus que, na sua grandiosidade infinita, ouviu minhas orações, em todos os momentos em que Dele precisei;*
- *Ao Prof. Dr. Jozué Vieira Filho pela excelente orientação durante a realização deste trabalho e por ter confiado a mim este grandioso projeto;*
- *A todos os professores, funcionários e colegas da Pós-graduação em Engenharia Elétrica da FEIS-UNESP que, indiretamente, auxiliaram, de uma forma ou de outra, na realização deste projeto;*
- *Ao meu irmão, Eng. Lindsay Perezi Marçal, que me recebeu em Ilha Solteira, no início do curso de mestrado, e também esteve comigo na conclusão deste trabalho;*
- *A minha super mãe Dalva, a quem devo tudo que sou;*
- *Agradecimentos especiais a minha esposa Adriana, quem eu amo, a minhas filhas Driellen e Lorena e aos nossos familiares, pelo apoio durante a curso de mestrado.*



RESUMO

Este trabalho consiste na implementação de um algoritmo de redução de ruído em sinais de voz para operação em tempo real usando o processador de sinais digitais TMS320C31. O filtro redutor de ruído foi desenvolvido visando, principalmente, uma futura aplicação em telefonia. O algoritmo implementado é baseado na técnica de Subtração Espectral (SE), com algumas modificações para eliminar o ruído musical, presente na SE clássica. A motivação para o uso desse algoritmo foi que ele apresenta uma baixa carga computacional, o que viabilizou a operação em tempo real usando o TMS320C31. Além disso, um novo método de detecção de ruído em sinais de voz foi proposto. Este método é baseado na variância da relação sinal/ruído e integra o filtro redutor de ruído implementado. Este filtro foi desenvolvido e executado numa plataforma composta de uma placa TIGER 31/PC adicionada a um microcomputador (PC). A TIGER 31/PC é uma placa de aquisição e processamento de sinais, que contém um processador de sinais digitais TMS320C31. O software utilizado para o desenvolvimento do filtro foi o Hypersignal RIDE, que permitiu a elaboração do aplicativo em ambiente gráfico. Os resultados do processamento, avaliados por meio de testes e análise de resultados, mostraram eficácia na redução de ruído e, além disto, o sinal processado apresentou baixas distorções e boa qualidade. Com relação ao uso de recursos da placa de processamento digital de sinais utilizada, observou-se que os recursos disponíveis, incluindo memória e processador, foram bem aproveitados. O novo método de detecção de ruído em sinais de voz também foi testado e avaliado durante o processamento no TMS320C31, obtendo-se resultados superiores aos obtidos com outros métodos convencionais.



ABSTRACT

This work consists in the implementation of a speech enhancement algorithm using a TMS320C31 Digital Signal Processor (DSP), which will be used to reduce noise in an analog telephone system in real time. The algorithm is based on the Spectral Subtraction, with several modifications to minimize the “musical noise”. The main advantage of this algorithm is its low computational load, making possible real-time applications. Furthermore, a new noise detection method for speech signals based on the variance of the signal-to-noise ratio was proposed and implemented. The implementation of the filter was based on a DSP/PC board, which incorporates a TMS320C31 running at 60 MHz, and a Real-time Integrated Development Environment (RIDE), which allows developing applications using a graphical interface. Available DSP’s resources were well explored and the results of processing, evaluated through objective and subjective tests, show good noise reduction and low distortions in the processed signal. The implemented system may be used to reduce noise in speech signals sampled up to 19 kHz.



SIMBOLOGIA E ABREVIACÕES

- A/D-D/A** – Analógico/Digital – Digital/Analógico
- ALU** – Unidade Lógica e Aritmética ("Arithmetic and Logic Unit")
- ARAU** – Unidade Aritmética de Registrador Auxiliar ("Auxiliar Register Arithmetic Unit")
- COFF** – Formato Comum de Arquivo de Objeto ("Common Object File Format")
- CPU** – Unidade Central de Processamento ("Central Processing Unit")
- DIS** – Detecção de Intervalos de Silêncio
- DISS** – DIS usando a Variância da Relação Sinal/Ruído
- DLL** – Biblioteca de Conexão Dinâmica ("Dynamic Link Library")
- DMA** – Acesso Direto a Memória ("Direct Memory Access")
- DSP** – Processador de Sinais Digitais ("Digital Signal Processor")
- E{.}** – Indica Valor Estimado
- E/S** – Entrada/Saída
- EPROM** – ROM programável e reprogramável ("Erasable Programmable ROM")
- FFT** – Transformada Rápida de Fourier ("Fast Fourier Transform")
- HFT** – Telefones Mãos Livres ("Hands Free Telephones")
- IFFT** – Transformada Rápida de Fourier Inversa ("Inverse Fast Fourier Transform")
- ISA** – "Industry Standard Architecture"
- ISTFT** – Transformada Inversa de Fourier de Curto Prazo ("Inverse Short Term Fourier Transform")
- MEM** – Método de Ephaim & Malah
- PC** – Computador Pessoal ("Personal Computer")
- RAM** – Memória de Acesso Aleatório ("Random Access Memory")
- RIDE** – Ambiente de Desenvolvimento Integrado de Tempo-Real
("Real-Time Integrated Development Environment")
- ROM** – Memória Somente para Leitura ("Read Only Memory")
- RT** – Tempo-Real ("Real-Time")
- SE** – Subtração Espectral
- SEM** – Subtração Espectral Modificada
- SECP** – Subtração Espectral de Curto Prazo
- SNR** – Relação Sinal/Ruído ("Signal/Noise Relation")
- SNR_{post}** – Relação Sinal/Ruído a posteriori
- SNR_{prio}** – Relação Sinal/Ruído a priori
- STFT** – Transformada de Fourier de Curto Prazo ("Short Term Fourier Transform")

CONTEÚDO

<i>Introdução</i>	1
-------------------------	---

CAPÍTULO I - INTRODUÇÃO AOS PROCESSADORES DE SINAIS DIGITAIS

<i>I.1 – Introdução</i>	4
<i>I.2 – O processador TMS320C31</i>	6
<i>I.2.1 – Características básicas</i>	7
<i>I.2.2 – Arquitetura</i>	9
<i>I.3 – Ferramentas de desenvolvimento de software para o TMS320C31</i>	14
<i>I.4 – Conclusões</i>	17

CAPÍTULO II - REDUÇÃO DE RUÍDO USANDO A SUBTRAÇÃO ESPECTRAL MODIFICADA (SEM)

<i>II.1 – Introdução</i>	18
<i>II.2 – Subtração Espectral (SE) Clássica</i>	19
<i>II.3 – Subtração Espectral Modificada (SEM)</i>	22
<i>II.4 – Metodologia de Implementação da SEM</i>	25
<i>II.4.1 – Janelamento</i>	27
<i>II.4.2 – Transformada Rápida de Fourier de Curto Prazo (STFT)</i>	27
<i>II.4.3 – Estimação do Ruído</i>	27
<i>II.4.4 – Estimação da Relação Sinal/Ruído a posteriori (SNR_{post})</i>	29
<i>II.4.5 – Estimação da Relação Sinal/Ruído a priori (SNR_{prio})</i>	29
<i>II.4.6 – Filtragem usando a SEM</i>	30
<i>II.4.7 – Transformada Inversa de Fourier de Curto Prazo (ISTFT)</i>	31
<i>II.4.8 – Reconstrução do sinal no tempo</i>	31
<i>II.5 – Conclusões</i>	31



CAPÍTULO III - DETECÇÃO DE INTERVALOS DE SILÊNCIO EM SINAIS DE VOZ USANDO A VARIÂNCIA DA RELAÇÃO SINAL/RUÍDO (DISS)

<i>III.1 – Introdução</i>	<i>33</i>
<i>III.2 – Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS).....</i>	<i>33</i>
<i>III.3 – Resultados Obtidos.....</i>	<i>38</i>
<i>III.4 – Conclusões</i>	<i>43</i>

CAPÍTULO IV - IMPLEMENTAÇÃO DO FILTRO REDUTOR DE RUÍDO EM SINAIS DE VOZ EM TEMPO REAL USANDO O TMS320C31.

<i>IV.1 – Introdução</i>	<i>45</i>
<i>IV.2 – Plataforma de desenvolvimento e operação</i>	<i>46</i>
<i>IV.2.1 – A placa TIGER 31/PC</i>	<i>47</i>
<i>IV.2.1.1 – Características do hardware</i>	<i>47</i>
<i>IV.2.1.2 – Arquitetura.....</i>	<i>48</i>
<i>IV.2.1.3 – Software</i>	<i>51</i>
<i>IV.2.2 – O software Hypersignal RIDE.....</i>	<i>52</i>
<i>IV.2.2.1 – Características básicas.....</i>	<i>52</i>
<i>IV.2.2.2 – Suporte às aplicações em tempo real</i>	<i>54</i>
<i>IV.3 – Desenvolvimento do filtro no RIDE</i>	<i>56</i>
<i>IV.3.1 – Transformação Tempo-Frequência.....</i>	<i>58</i>
<i>IV.3.2 – DIS e Estimção do Ruído.....</i>	<i>60</i>
<i>IV.3.3 – Cálculo da Função de Transferência do Filtro ($H(w)$).....</i>	<i>64</i>
<i>IV.3.4 – Filtragem</i>	<i>66</i>
<i>IV.3.5 – Transformação Frequência-Tempo.....</i>	<i>68</i>
<i>IV.4 – Conclusões.....</i>	<i>70</i>

CAPÍTULO V - RESULTADOS OBTIDOS E AVALIAÇÕES

<i>V.1 – Introdução.....</i>	<i>72</i>
<i>V.2 – Estatísticas do filtro implementado no RIDE</i>	<i>73</i>
<i>V.2.1 – Memória.....</i>	<i>73</i>
<i>V.2.2 – Tempo de processamento.....</i>	<i>75</i>
<i>V.3 – Avaliação da filtragem realizada pelo filtro implementado.....</i>	<i>79</i>
<i>V.3.1 – Formas de Onda dos sinais temporais.....</i>	<i>80</i>
<i>V.3.2 – Detecção de Intervalos de Silêncio (DIS).....</i>	<i>84</i>
<i>V.3.3 – Relações Sinal/Ruído e Redução de Ruído obtida na filtragem</i>	<i>86</i>
<i>V.3.4 – Espectrogramas</i>	<i>92</i>
<i>V.3.5 – Testes subjetivos.....</i>	<i>96</i>
<i>V.4 – Conclusões.....</i>	<i>98</i>

CAPÍTULO VI - COMENTÁRIOS FINAIS.....	99
--	-----------

REFERÊNCIAS BIBLIOGRÁFICAS.....	101
--	------------

INTRODUÇÃO

O desenvolvimento e a implementação para operação em tempo real de novos algoritmos de redução de ruído em sinais de voz são atualmente alvo de muitas pesquisas científicas, devido ao crescente desenvolvimento das telecomunicações neste final de milênio. Dentro desse contexto, a redução de ruído em sinais de voz pode ser aplicada em várias atividades, como por exemplo, em redes telefônicas convencionais, telefonia celular, telefones viva-voz ("Hands-free") e até em serviços de informação baseados em sistemas com síntese/reconhecimento de voz. Qualquer avanço objetivando o melhoramento de voz dessa natureza pode ser considerado como uma grande contribuição à pesquisa científica em telecomunicações.

Na maioria das aplicações práticas um algoritmo de redução de ruído eficiente deverá ser capaz de melhorar o sinal ruidoso nos aspectos físico (recuperação da forma de onda original) e auditivo (inteligibilidade), além de possuir baixa carga computacional, viabilizando uma implementação em tempo real.

Para satisfazer a estes requisitos fundamentais uma nova técnica de redução de ruído, baseada na Subtração Espectral (SE), desenvolvida em [1], foi implementada para operação em tempo real. Esta técnica, aqui denominada *Subtração Espectral Modificada (SEM)*, apresenta um melhoramento significativo do sinal ruidoso, mantendo a baixa carga computacional da SE Clássica. A vantagem é, principalmente, a minimização do ruído musical, característico da SE Clássica.

Um ponto importante deste trabalho, foi o desenvolvimento e a implementação de um novo método de detecção de ruído em sinais de voz ruidosos. Este método, baseado na relação sinal/ruído, usa um sinal de variância desta relação para comparação com um limiar de decisão voz/ruído constante, e foi denominado *Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS)*. A DISS integra o algoritmo redutor de ruído implementado.

A definição de uma plataforma apropriada para o desenvolvimento e execução do filtro foi fundamental para o sucesso deste projeto, considerando principalmente a quantidade de memória disponível para o armazenamento de instruções e dados e o processador que executa



a aplicação. Na plataforma escolhida, o hardware é composto basicamente de uma placa de processamento digital de sinais TIGER 31/PC [12] acoplada a um microcomputador (PC). A placa TIGER 31/PC escolhida possui 128 KB de memória RAM estática (SRAM) e incorpora um processador de sinais digitais TMS320C31-60 [17] operando a 60 MHz. O software de desenvolvimento é o Hypersignal RIDE [11], que opera sobre o Windows e é utilizado a partir dos consoles do PC. O RIDE possui um controlador (“driver”) que realiza a interface entre o PC e a TIGER 31/PC. Este software permitiu que o filtro fosse elaborado graficamente, através da criação de diagramas de blocos. O uso desta plataforma foi uma tarefa que requereu um estudo das características do processador TMS320C31, da placa TIGER 31/PC e do software Hypersignal RIDE. Por outro lado, obter tempo-real, nas condições desejadas, não foi tarefa das mais fáceis, pois demandou, durante o processo de elaboração do filtro no RIDE, otimização de funções, envolvendo constantes alterações no diagrama de blocos do filtro e avaliações do processamento.

Para atingir o objetivo proposto nesta dissertação, com sucesso, várias etapas de estudos e desenvolvimento foram realizadas. Assim, o texto foi estruturado como segue:

- No Capítulo I é apresentada uma introdução aos processadores de sinais digitais (DSPs). Descrevem-se as características básicas e a arquitetura do DSP TMS320C31, bem como as ferramentas básicas de desenvolvimento de software para este processador.
- No Capítulo II são descritas as técnicas de redução de ruído baseadas na Subtração Espectral (SE). A partir da SE clássica e da definição de novos parâmetros, identificados como a relação sinal/ruído a posteriori (SNR_{post}) e a relação sinal/ruído a priori (SNR_{prio}), uma nova técnica é apresentada, a Subtração Espectral Modificada (SEM), bem como a metodologia de implementação desta técnica e das várias etapas envolvidas nesta implementação.
- No Capítulo III um novo método de detecção de ruído em sinais de voz ruidosos, usando a variância da relação sinal/ruído, é proposto. Os resultados da nova detecção também são apresentados.



- No Capítulo IV é feita uma descrição detalhada da implementação do filtro redutor de ruído com o software de desenvolvimento Hypersignal RIDE, para armazenamento e execução na placa TIGER 31/PC, em tempo real. Neste contexto, apresenta-se a plataforma utilizada no projeto do filtro, incluindo as características básicas da TIGER 31/PC e do ambiente de desenvolvimento RIDE, bem como as suas capacidades com relação a operações em tempo real. O filtro desenvolvido no RIDE, incluindo seus subsistemas, são explicados.
- No Capítulo V são apresentados e avaliados os resultados de testes realizados com o filtro implementado no RIDE. Estes testes incluem estatísticas do DSP, como memória utilizada e tempo de processamento, bem como características da filtragem, incluindo redução de ruído obtida e análises da qualidade do sinal filtrado processado.
- Finalmente, no Capítulo VI, comentam-se os resultados obtidos.



CAPÍTULO I

INTRODUÇÃO AOS PROCESSADORES DE SINAIS DIGITAIS

I.1 – Introdução

Os microprocessadores dedicados ao processamento de sinais digitais surgiram no início dos anos 80 com o INTEL 2920, seguido pelo NEC μ PD7720 e pela primeira geração da família TMS320 da Texas Instruments, com o TMS320C10, em 1982. Desde a introdução do primeiro processador da família TMS320, a Texas Instruments estabeleceu um padrão para a indústria de processadores de sinais digitais (DSPs).

Os DSPs foram desenvolvidos para aplicações que requerem uma alta velocidade de processamento, principalmente para a execução de algoritmos em tempo real. A principal vantagem dos DSPs em relação aos microprocessadores comuns está no fato de os DSPs utilizarem um conjunto de instruções baseadas em operações típicas de processamento digital. Por exemplo, uma operação típica de processamento digital combina operações de soma, multiplicação e armazenamento (tipo $A=BC + D$) e pode ser realizada em um único ciclo de instrução de um DSP, enquanto que em microprocessadores comuns levam mais de quatro ciclos. Funções de multiplicação levam mais tempo, especialmente quando envolvem números tais como π . Um microprocessador convencional pode levar várias centenas de ciclos de *clock* para implementar este tipo de cálculo. Os DSPs têm unidades de hardware dedicadas a tais operações, de modo a realizá-las em um único ciclo de *clock*.

De um modo geral, os DSPs consistem em vários módulos básicos, a saber: unidade de processamento, unidade de memória para armazenamento de programas, unidade de memória de dados e unidades de interfaceamento. Como qualquer microprocessador de uso geral, o DSP busca, decodifica e executa uma instrução, que pode ou não envolver dados armazenados na memória. Entretanto, o modo como isto é realizado não é o mesmo para todos os processadores. Duas arquiteturas podem ser identificadas: *Von Neumann* e *Harvard*. Os diagramas em blocos referentes a estas arquiteturas são mostrados nas figuras I.1 e I.2. A aplicação, além de memória e configuração periférica, normalmente define o tipo de arquitetura empregada.



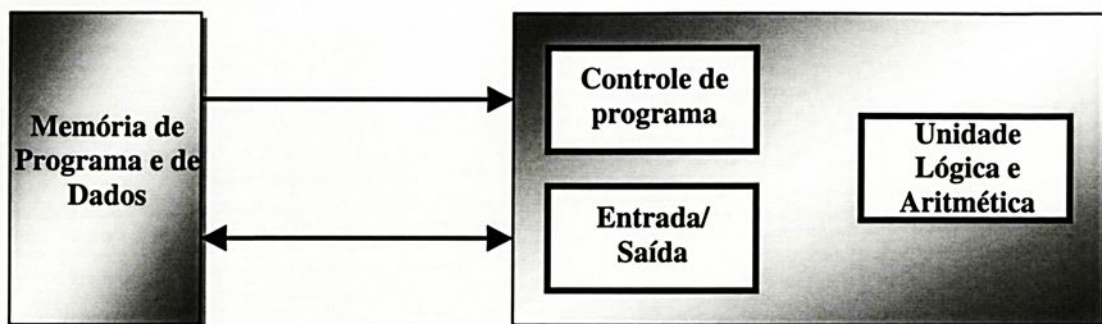


Fig. I.1 – Arquitetura Von Neumann

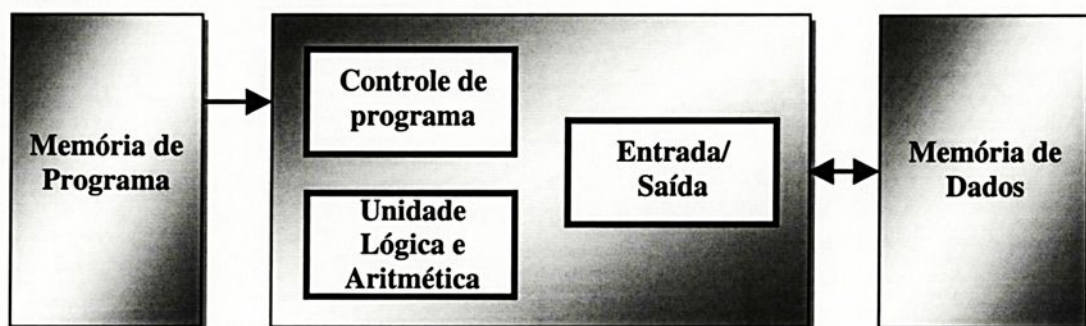


Fig. I.2 – Arquitetura Harvard

A arquitetura Von Neumann estabeleceu um padrão para o desenvolvimento de computadores durante os últimos 40 anos. Nessa arquitetura, que é usada na maioria dos microprocessadores, programa e dados podem residir no mesmo espaço de memória. A desvantagem com esta arquitetura é que há somente um barramento para dados e endereços, o que impede o acesso simultâneo a códigos e dados.

Para aplicações em que é necessária uma manipulação rápida de dados, acessar as memórias de dados e programa em um único ciclo é vantajoso. Isto acontece com a arquitetura Harvard, que separa os espaços de memória de dados e de programa e viabiliza o acesso simultâneo através de dois barramentos independentes. O aumento na velocidade de processamento é óbvio. Contudo, há um custo adicional, que é a complexidade do hardware. Uma solução intermediária foi encontrada com a arquitetura *Harvard Modificada*, que tem somente um barramento externo, mantendo no entanto dois barramentos internos.

A arquitetura de um sistema típico baseado em DSP é apresentada na figura I.3, que inclui:

- DSP;
- Memória, para armazenar instruções de programa e dados;
- Circuitos de conversão analógico para digital e vice-versa (conversores A/D e D/A), visto que a maioria das aplicações envolve sinais analógicos.

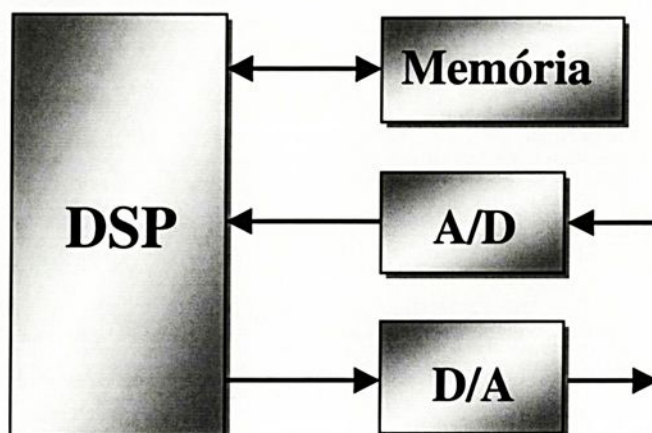


Fig. I.3 – Sistema típico usando DSP

Devido à sua arquitetura projetada especificamente para o processamento de sinais digitais, os DSPs geralmente têm um poder de processamento superior ao dos processadores convencionais. Para aplicações em tempo real, onde a manipulação rápida de dados é necessária, o uso dos DSPs se torna praticamente indispensável. As aplicações dos DSPs atingem atualmente todas as áreas tecnológicas, como telecomunicações, áudio, vídeo, controle, etc.

I.2 – O processador TMS320C31

O TMS320C31 é um processador de sinais digitais de ponto flutuante da terceira geração de processadores produzidos pela “Texas Instruments” [15], denominado aqui de ‘C31. Sua arquitetura é parecida com a Von Neumann e sua alta velocidade de processamento é viabilizada com a implementação de funções em hardware que outros processadores

convencionais implementariam através de software ou microcódigos. O 'C31 pode realizar paralelamente e em um único ciclo, multiplicação e operações na ALU (Unidade Lógica e Aritmética) sobre dados ponto-flutuante. O 'C31 também possui vários registradores, uma memória "cache" de programa, unidades aritméticas especiais (ARAUs), memórias internas de acesso simultâneo, canal de DMA e alta velocidade de processamento. Há também alguns dispositivos auxiliares, como temporizadores, interface serial e controladores de interrupção.

A CPU tem um multiplicador e uma ALU independentes para oferecer até 60 milhões de operações por segundo em ponto flutuante (MFLOPS). O 'C31 com capacidade para operar a 60 MFLOPS é denominado TMS320C31-60.

O controlador de DMA tem seu próprio barramento de dados e opera em paralelo com a CPU. Com uma arquitetura de barramentos internos independentes, o acesso a dispositivos de entrada/saída (E/S) via DMA e à memória externa ou interna, pode ocorrer mesmo quando a CPU está realizando alguma operação. O espaço total de memória do 'C31 é de 16M palavras de 32 bits (64 Mbytes), incluindo acesso à E/S. Internamente há duas unidades de memória RAM de 1K palavras de 32 bits (8 Kbytes), que podem ser acessadas simultaneamente pela CPU em um único ciclo. Para manter o alto desempenho usando pouca memória externa, o 'C31 possui uma memória "cache" de 64 palavras (256 bytes).

Para programação do 'C31, existem ferramentas de desenvolvimento da Texas Instruments que incluem um simulador por software, um compilador C, incluindo um "linker" e um "Assembler", um emulador de hardware, um módulo de avaliação e um "kit" inicializador do DSP. Adicionalmente, há outras ferramentas e/ou ambientes de desenvolvimento, como o software Hypersignal RIDE, que permite o desenvolvimento de aplicações em ambiente gráfico.

1.2.1 – Características básicas

O processador TMS320C31 é produzido em um encapsulamento quadrado de 132 pinos, usando tecnologia EPIC ("Enhanced Performance Implanted CMOS) de 0,6 μ m e possui as seguintes características básicas:



CPU

- Tempo de ciclo de instrução de 33 ns (TMS320C31-60);
- Palavras de instrução e dados de 32 bits, e de endereço de 24 bits;
- Operações lógicas de 32 bits, inteiras de 24/32 bits e ponto-flutuante de 32/40 bits;
- Instruções de dois e três operandos;
- Instruções de processamento paralelo no Multiplicador e na ALU, em um único ciclo;
- Capacidade de repetição de bloco;
- Chamadas condicionais e retornos;
- Instruções para multiprocessamento;
- Dois geradores de endereços com oito registradores auxiliares e duas unidades aritméticas de registrador auxiliar (ARAUs);
- Oito registradores de precisão estendida (40 bits);
- Registrador de deslocamento de 32 bits;
- Registradores de controle de barramento que configuram a geração de estado de espera;

Periféricos (mapeados em memória):

- Um controlador de DMA;
- Uma porta serial;
- Dois temporizadores de 32 bits;

Memória

- “Cache” de instrução de 64 palavras de 32 bits (256 bytes);
- Dois blocos RAM de 1K palavras de 32 bits (8 Kbytes), de acesso simultâneo num único ciclo;
- espaço de endereçamento de 16M palavras (64 Mbytes);
- ROM de “boot” pré-programada;
- um barramento de memória externa;



1.2.2 – Arquitetura

O 'C31 tem uma arquitetura Von Neuman, que provê um espaço de programa e dados unificado. Entretanto, há um conjunto de barramentos internos, que podem operar de modo independente, e um grande número de periféricos internos, que aumentam sensivelmente seu desempenho. A figura I.4 apresenta o diagrama em blocos do processador TMS320C31, onde destacam-se:

CPU

A CPU do TMS320C31, apresentada na figura I.5, tem um multiplicador e uma ALU independentes e atinge até 60 MFLOPS (TMS320C31-60). Resultados são armazenados em um dos 8 registradores de precisão estendida. Estes registradores de 40 bits armazenam valores com 32 bits de mantissa e 8 bits de expoente e destinam-se a qualquer operação aritmética. Os registradores de precisão estendida são um recurso extremamente valioso para programação em linguagem Assembly ou C. Estes registradores permitem manter resultados intermediários sem armazenar dados na memória. Isto resulta em um código Assembly de alto desempenho e um compilador C eficiente.

Para atingir 60 MFLOPS, a CPU tem duas ARAUs (Unidades Aritméticas de Registradores Auxiliares). Estas ARAUs geram endereços de 24 bits que são acessados através de oito registradores auxiliares. As ARAUs podem executar qualquer uma das seguintes operações:

- Incremento ou decremento, com valores iguais ou diferentes de 1;
- Endereçamento circular para o uso de “buffers” circulares;
- Endereçamento “bit-reverso” para FFTs;

Além disso, a arquitetura baseada em registradores da CPU dispõe de um registrador de deslocamento de 32 bits, oito registradores de precisão estendida (R0-R7), dois registradores de índice (IR0 e IR1), oito registradores de endereçamento indireto (AR0-AR7) e doze registradores de sistema para controle de barramentos. Os dois registradores de índice (IR0 e IR1) são usados pelas ARAUS para indexar um endereço [16].



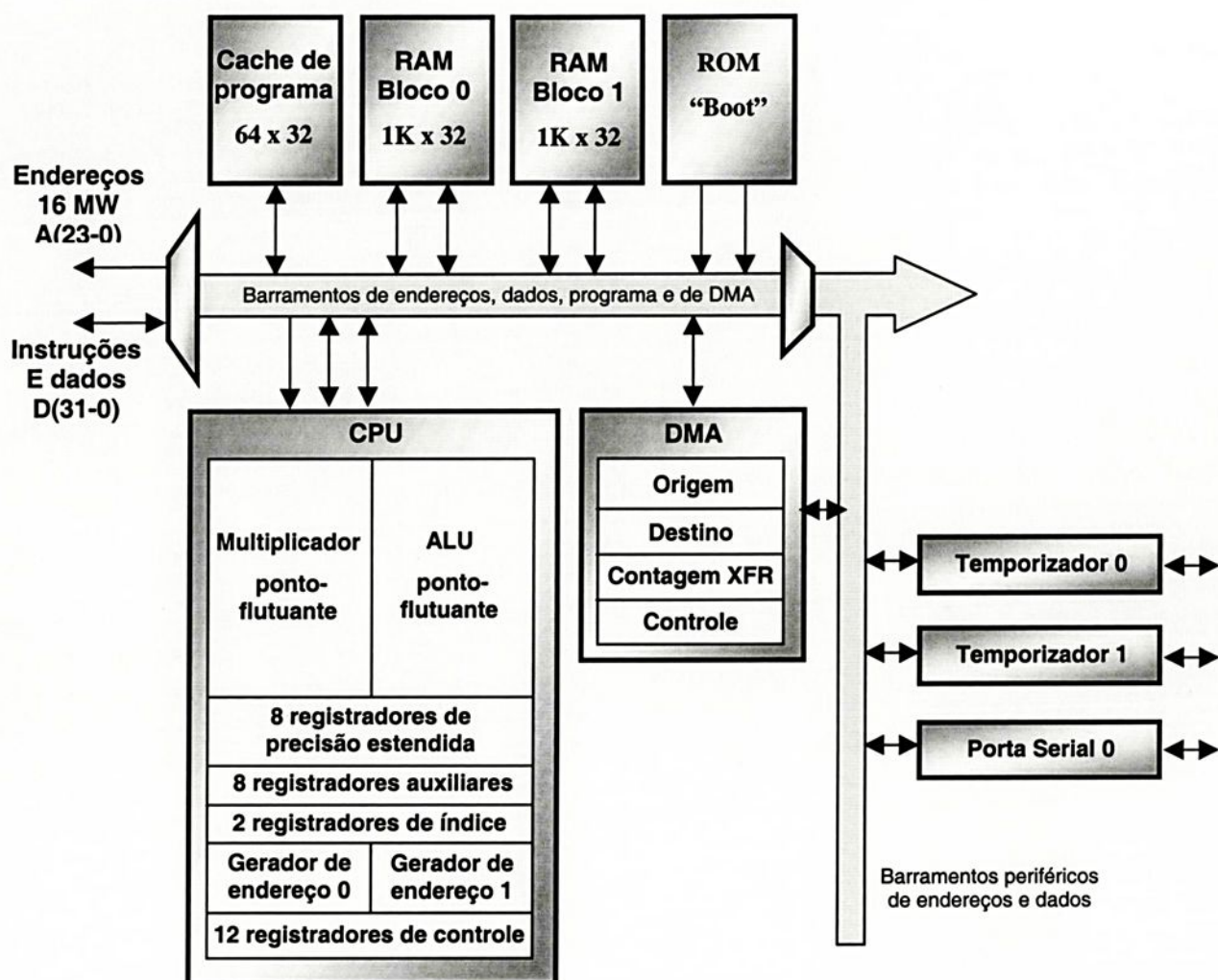


Figura I.4 – Diagrama em blocos do TMS320C31

Memória

Para obter um máximo aproveitamento, o processador 'C31 é projetado para acessar até quatro palavras por ciclo. Estas consistem de um código de instrução, dois operandos de dados (internos) e uma transferência de dado do controlador de DMA. Os barramentos internos podem transferir todas as quatro palavras em paralelo. O 'C31 usa sete barramentos para realizar as transferências internas. Estes barramentos são descritos como:

- *Dados/endereços para programa:* A CPU usa estes barramentos para buscar instruções em todos os ciclos;

- *Dados/endereços para dados:* Em qualquer ciclo, a CPU pode buscar dois operandos, devido aos dois barramentos de endereços de dados e um barramento de dado, que pode ser acessado duas vezes em um único ciclo;
- *Dados/endereços para DMA:* O DMA usa estes barramentos para realizar a transferência do controlador de DMA em paralelo com operações da CPU.

A arquitetura de barramentos independentes do 'C31 permite o uso de memórias interna e externa (de dados e de programa). Há internamente dois blocos RAM de 1 K palavras de 32 bits (8 Kbytes), uma memória *cache* e uma unidade de memória ROM.

A figura I.6 apresenta o diagrama em blocos da memória do processador 'C31.

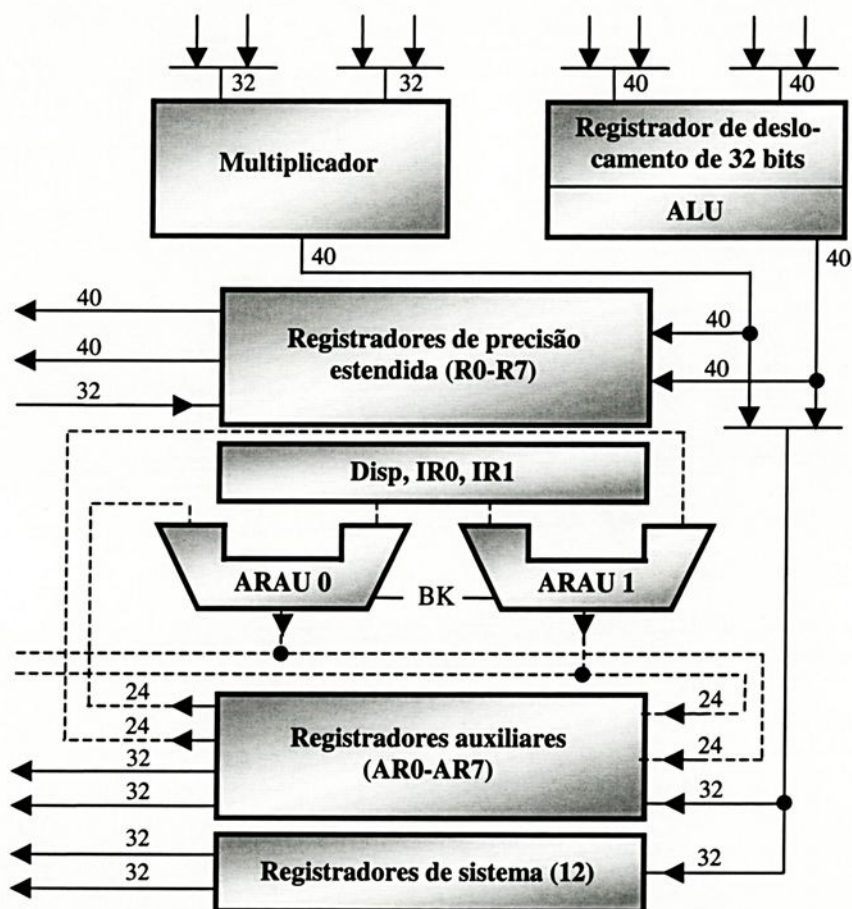


Figura I.5 – Diagrama em blocos da CPU do TMS320C31

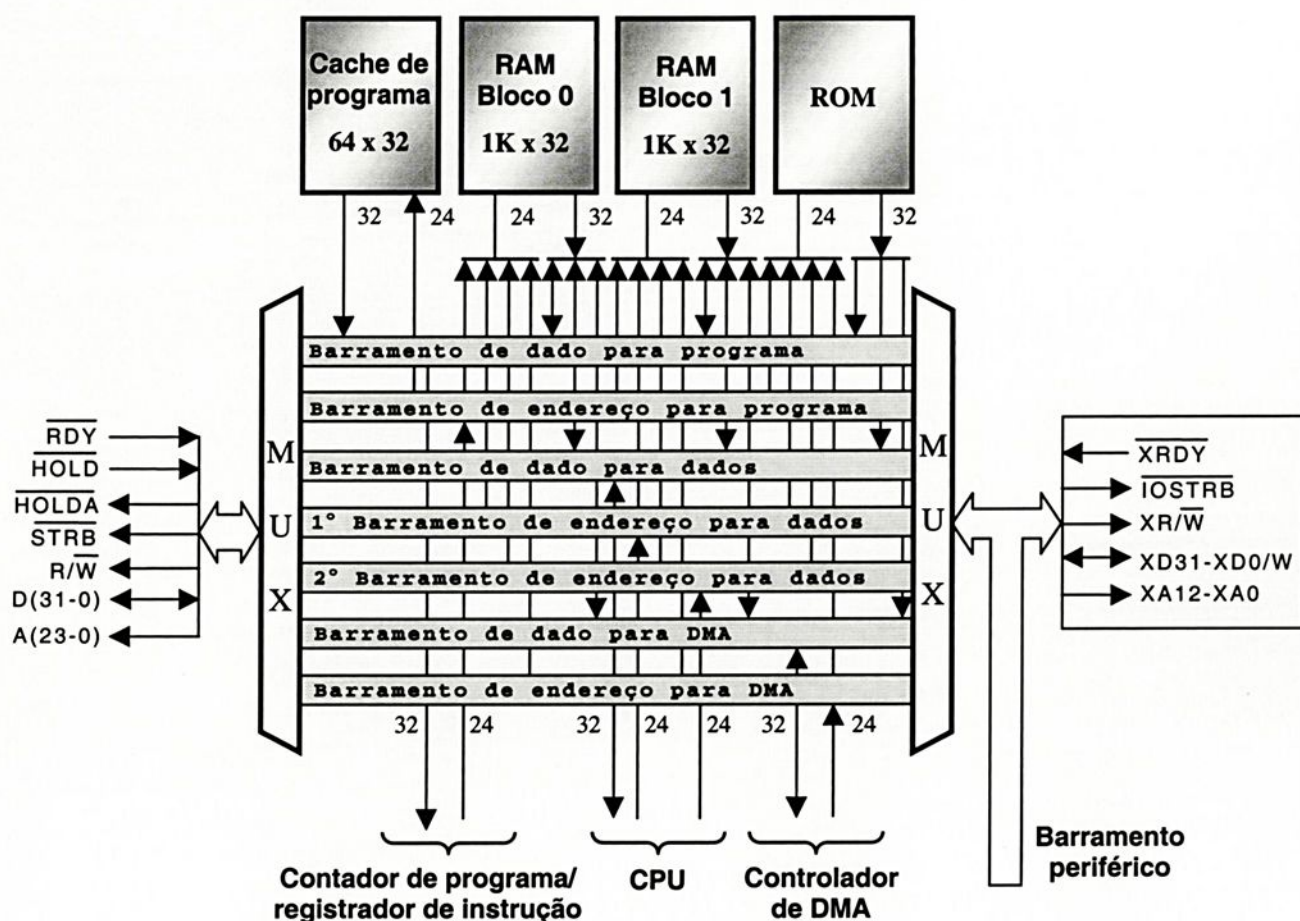
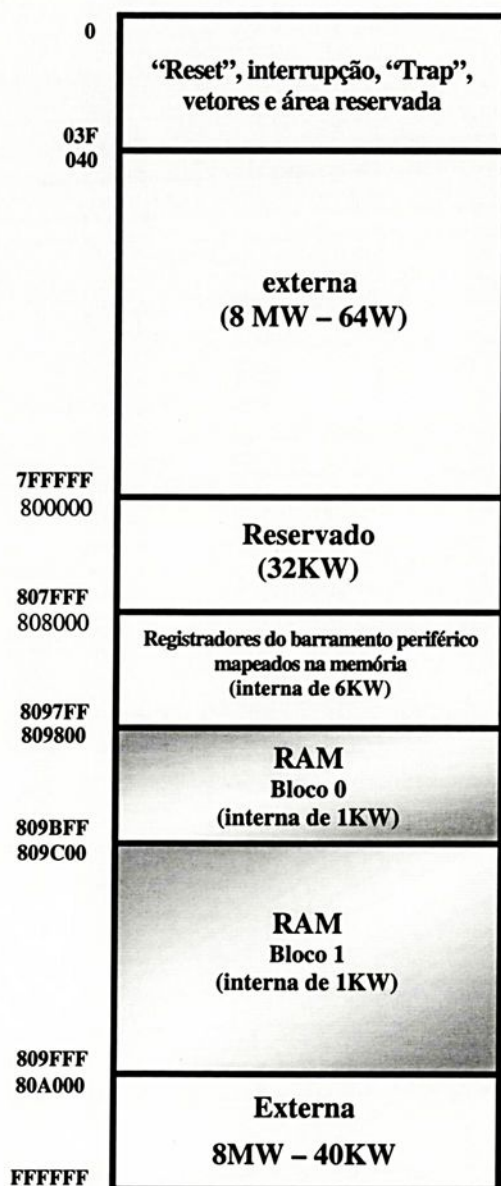


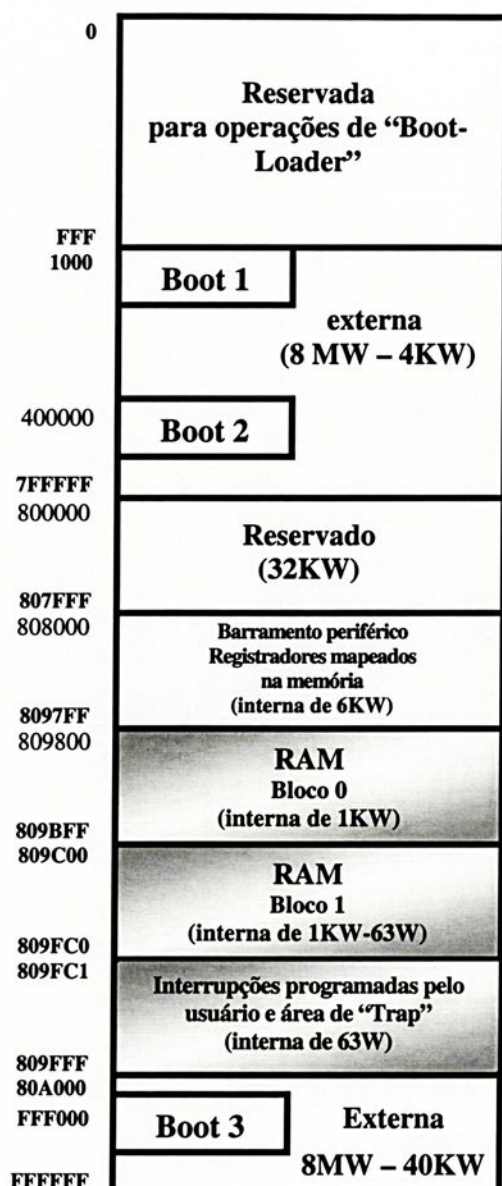
Figura I.6 – Diagrama em blocos da Memória do TMS320C31

O barramento primário do 'C31 está habilitado a acessar até 16M palavras (64 Mbytes), que define o limite máximo de memória que pode ser usada pelo processador. Os mapas de memória do 'C31 [17] são mostrados na figura I.7, onde os endereços estão em hexadecimal e o tamanho de memória é medido em palavras (W) de 32 bits.

Na figura I.7-a tem-se o mapa de memória do 'C31 no modo "microprocessador". Este é um modo de operação do processador em que a ROM interna ou a memória flash (se houver) está *desabilitada* e, neste caso, o processo de "boot" deve ser realizado a partir da memória externa. Na figura I.7-b tem-se o mapa de memória do modo "microcomputador", em que a ROM do 'C31 ou a memória *flash* está *habilitada* e, portanto, a inicialização do TMS320C31 é realizada a partir de informações contidas nesta ROM. [16]



(a) Modo microprocessador



(b) Modo microcomputador/"Boot-Loader"

Figura I.7 – Mapas de Memória do TMS320C31

DMA

A figura I.8 mostra o diagrama em blocos do controlador de DMA do processador 'C31. Para realizar uma transferência, o controlador de DMA lê o conteúdo do endereço apontado pelo registrador de endereço de origem e então escreve este conteúdo no endereço apontado pelo registrador de endereços de destino. Os endereços de origem e destino são incrementados ou

decrementados após cada transferência, dependendo do valor do registrador de controle. O controlador de DMA realiza transferências contínuas através do barramento de DMA até que o valor no registrador de contagem de transferência chegue a zero, ativando uma interrupção.

Por exemplo, uma aplicação poderia usar o DMA para transferir 512 palavras da memória externa lenta para a RAM interna, enquanto a CPU realiza algum processamento. Entretanto, caso a CPU necessite acessar à RAM interna, o controlador de DMA suspende imediatamente as transferências.

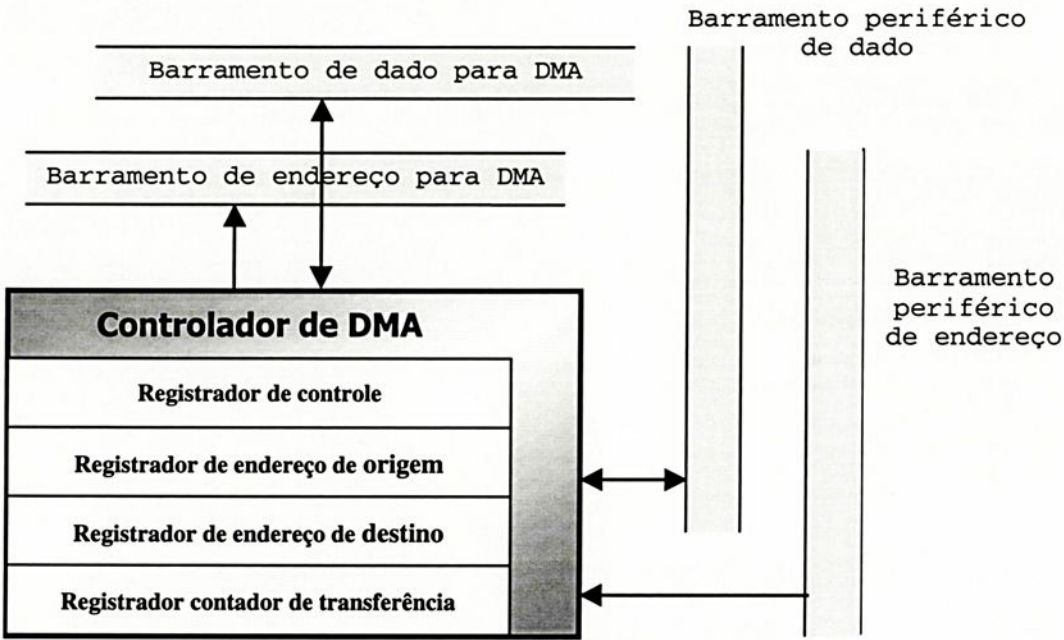


Figura I.8 – Diagrama em blocos do Controlador de DMA do TMS320C31

I.3 – Ferramentas de desenvolvimento de software para o TMS320C31

Para desenvolver uma aplicação para o TMS320C31 existe um conjunto básico de ferramentas de geração de código fornecido pela Texas Instruments, que inclui um compilador C, um “assembler”, um “linker” e geradores de Biblioteca. A figura I.9 apresenta um diagrama com a seqüência lógica padrão para o desenvolvimento de aplicações para o TMS320C31 [14]. Os blocos sombreados da figura realçam o caminho mais comum no desenvolvimento de software; os outros blocos são opcionais.



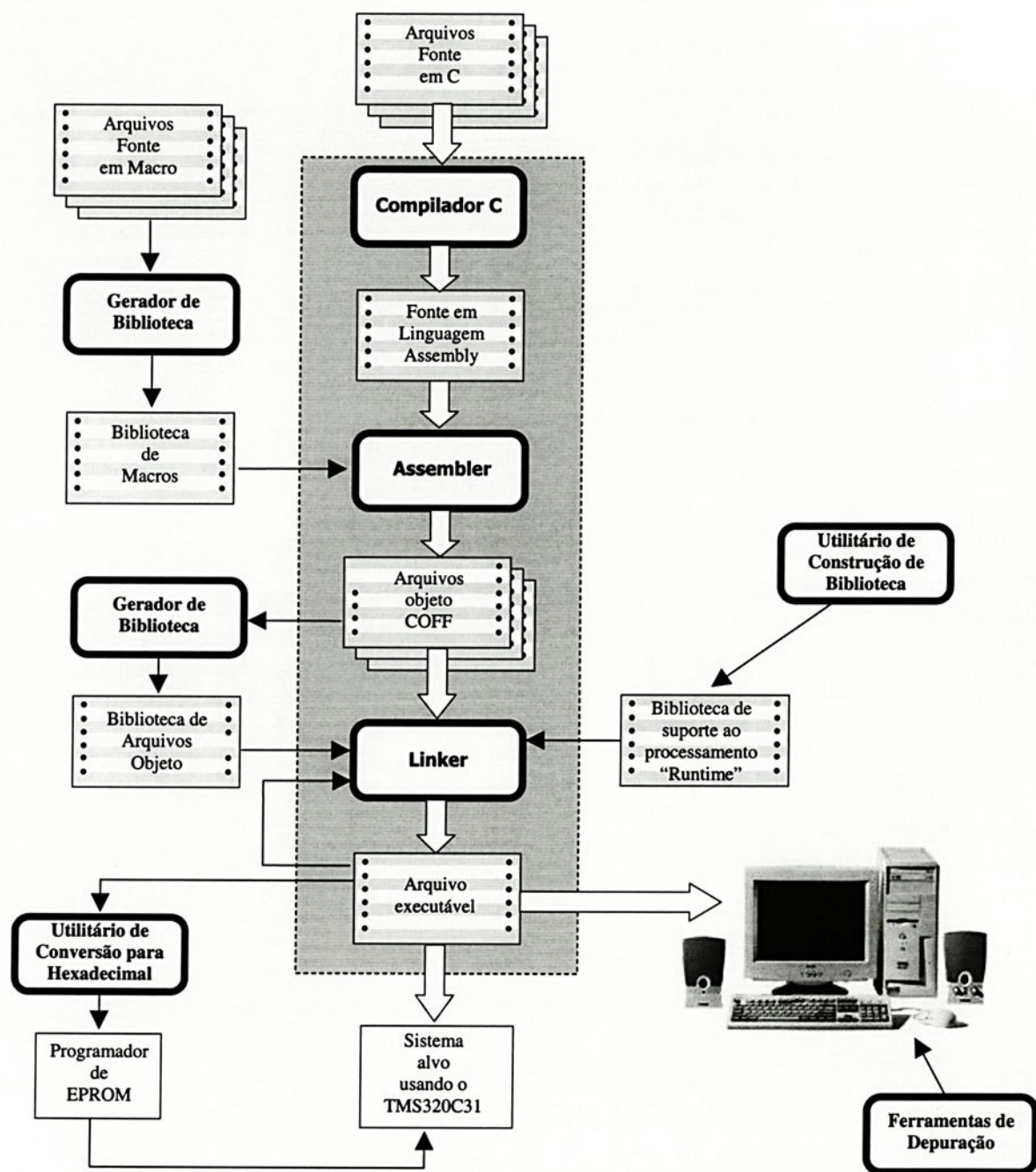


Figura I.9 – Diagrama de desenvolvimento de software para o TMS320C31

Compilador C

O desenvolvimento de uma aplicação em linguagem “assembly” de qualquer DSP é tarefa difícil e, em muitos casos, impraticável. Assim, há um compilador C ANSI que produz o código fonte em linguagem “assembly” do TMS320C31. O compilador fornecido pela Texas Instruments inclui também um otimizador que opera diretamente no código em “assembly”.

☐ **Assembler**

O “assembler” traduz arquivos fonte em linguagem “assembly” para arquivos objeto em linguagem de máquina. A linguagem de máquina é tipo COFF (“Common Object File Format” – Formato Comum de Arquivo Objeto). O formato COFF usa o conceito de programação modular, através da definição de seções (segmentos contínuos de memória).

☐ **Linker**

O “linker” combina arquivos objeto formando um único módulo objeto executável. O linker reconhece arquivos objeto COFF e bibliotecas de objeto como entrada.

☐ **Gerador de Biblioteca**

O Gerador de Biblioteca permite coletar um grupo de arquivos e formar uma biblioteca. Ele também permite ao usuário modificar uma biblioteca, apagando, substituindo, extraíndo ou adicionando membros. Uma das aplicações mais usuais do Gerador de Biblioteca é a construção de uma biblioteca de módulos objeto.

☐ **Bibliotecas de arquivos objeto**

Oito bibliotecas de objeto são fornecidas com o compilador C. Estas bibliotecas contêm as funções padrão de suporte ao processamento (“run-time”) e funções do compilador para o TMS320C31.

☐ **Utilitário de Construção de Biblioteca**

O Utilitário de Construção de Biblioteca deve ser utilizado para a construção de bibliotecas próprias, de suporte ao processamento (“run-time”). As funções padrões de suporte ao processamento são fornecidas em código fonte.

☐ **Utilitário de Conversão para Hexadecimal**

O Utilitário de Conversão para Hexadecimal converte um arquivo objeto COFF para o formato objeto ASCII-hex, Intel, Motorola-S, TI-target, ou Tektronix. O arquivo convertido pode ser carregado num programador de EPROM.



A sequência indicada na figura I.9 é necessária para a obtenção de um módulo que possa ser executado em um sistema usando o TMS320C31. Para auxiliar esta tarefa, o usuário pode usar ferramentas de “depuração” para refinar e corrigir o código antes de carregá-lo no sistema alvo. As plataformas de “depuração” oferecem uma interface que permite ao usuário visualizar informações em linguagem de baixo nível, inspecionar e modificar variáveis em C, visualizar o código fonte em C e monitorar a execução do programa.

I.4 – Conclusões

Devido à sua arquitetura projetada especificamente para o processamento de sinais digitais, os DSPs geralmente têm um poder de processamento superior ao dos processadores convencionais, viabilizando o desenvolvimento de sistemas para aplicações em tempo real. Este fato é comprovado pelas diversas áreas em que são aplicados os DSPs, como por exemplo, as telecomunicações, áudio e vídeo, e controle.

O processador TMS320C31 é um DSP de ponto-flutuante que possui um alto grau de paralelismo, com uma arquitetura composta de vários barramentos internos de dados e endereços, uma CPU com alta capacidade de processamento e diversos registradores. Destaca-se que a CPU do TMS320C31 tem um multiplicador e uma ALU independentes, possibilitando que até duas operações aritméticas sejam executadas em um único ciclo. O TMS320C31 tem duas unidades de memória RAM, memória “cache” e ROM de inicialização, que podem ser acessadas simultaneamente, em um único ciclo de *clock*, devido à estrutura de barramentos internos independentes. Este DSP possui também um controlador de DMA para a realização de operações concorrentes entre a E/S e a CPU, além de uma porta serial e dois temporizadores.

Com relação à programação do TMS320C31, existem várias ferramentas de desenvolvimento de software disponíveis. As ferramentas básicas incluem um compilador C, um “Assembler” e um “Linker”, além de várias bibliotecas oferecidas pela Texas Instruments e outras empresas de DSP.



CAPÍTULO II

REDUÇÃO DE RUÍDO USANDO A SUBTRAÇÃO ESPECTRAL MODIFICADA (SEM)

II.1 – Introdução

A redução de ruído em sinais de voz pode ser aplicada em várias áreas. Em telecomunicações, o ruído em sinais de voz ocorre em diversos ambientes como, por exemplo, em redes telefônicas convencionais, telefonia celular e em telefones viva-voz (“Hands-free”). Na maioria das aplicações um algoritmo de redução de ruído eficiente deverá ter duas características fundamentais:

- melhoria do sinal ruidoso nos aspectos físico (recuperação da forma de onda original) e auditivo (inteligibilidade);
- baixa carga computacional, viabilizando uma implementação em tempo real.

Satisfazer a estes requisitos fundamentais não é tarefa fácil, pois se por um lado as técnicas simples de redução de ruído não apresentam resultado satisfatório, por outro lado, técnicas mais eficientes possuem alta complexidade matemática, impossibilitando o processamento em tempo real. Como exemplo de técnicas de redução de ruído com características tão distintas têm-se a *Subtração Espectral (SE) Clássica* [1] e o *Método de Ephraim & Malah (MEM)* [3]. Baseada na SE Clássica, de simples implementação porém de resultados pouco eficientes, uma nova técnica foi desenvolvida [9]. Esta nova técnica, aqui denominada *Subtração Espectral Modificada (SEM)*, apresenta um melhoramento significativo do sinal ruidoso, mantendo a boa eficiência computacional da SE Clássica.

As técnicas de redução de ruído baseadas na Subtração Espectral (SE) [1] somente podem ser aplicadas para sinais ruidosos estacionários [7]. Para o sinal de voz, sabe-se a priori que este apresenta uma característica de não-estacionariedade extremamente forte. No entanto estudos mostram que para pequenos intervalos de tempo, normalmente até 40 ms [7], o sinal de voz pode ser considerado aproximadamente estacionário. Neste caso pode-se pensar na aplicação



das técnicas baseadas na Subtração Espectral (SE), usando-se técnicas de processamento de sinais. A principal é a STFT (Short Term Fourier Transform) que possibilita o processamento de trechos do sinal aproximadamente estacionários [8]. A aplicação da SE a pequenos períodos do sinal recebe o nome de Subtração Espectral de Curto Prazo (SECP), mas a título de simplificação será denominada apenas de SE.

II.2 – Subtração Espectral (SE) Clássica

Seja, por exemplo, um sinal de voz puro, $v(t)$, degradado por um ruído aditivo $r(t)$, formando um sinal ruidoso $y(t)$ como apresentado na equação II.1:

$$y(t) = v(t) + r(t) \quad (\text{II.1})$$

Após um processo de amostragem do sinal ruidoso e considerando apenas as amostras no intervalo Δt , pode-se reescrever a equação II.1 como segue:

$$y_{\Delta t}(n) = v_{\Delta t}(n) + r_{\Delta t}(n) \quad (\text{II.2})$$

Para o desenvolvimento da técnica de subtração espectral, considere Δt como sendo o quadro temporal analisado. Assim, no domínio da frequência, a equação II.2 torna-se:

$$Y_{\Delta t}(w) = V_{\Delta t}(w) + R_{\Delta t}(w) \quad (\text{II.3})$$

Tomando-se o quadrado na equação II.3 obtém-se:

$$|Y_{\Delta t}(w)|^2 = |V_{\Delta t}(w)|^2 + V_{\Delta t}(w)^* \times R_{\Delta t}(w) + V_{\Delta t}(w) \times R_{\Delta t}(w)^* + |R_{\Delta t}(w)|^2 \quad (\text{II.4})$$



Considerando o sinal de voz e o ruído como processos aleatórios independentes, da equação II.4 tem-se:

$$|V_{\Delta}(w)|^2 = |Y_{\Delta}(w)|^2 - |R_{\Delta}(w)|^2 \quad (\text{II.5})$$

A equação II.5 mostra que é possível recuperar o espectro de amplitude do sinal original, desde que se conheça o espectro do ruído. Como não se dispõe isoladamente dessa informação, deve-se estimá-la. Além disso, observa-se que não existe uma recuperação da fase do sinal. Isto ocorre porque, além de não existir método que permita uma estimação da fase original, demonstra-se que o ouvido humano é pouco sensível às variações de fase nestas condições (sinal puro/sinal ruidoso) [5]. Portanto, a subtração espectral é aplicada somente para o espectro de potência do sinal, preservando-se a fase do sinal ruidoso.

Assim, estimando-se a potência do ruído e aplicando-se o valor estimado na equação II.5, obtém-se:

$$E\{|V_{\Delta}(w)|\} = \sqrt{|Y_{\Delta}(w)|^2 - E\{|R_{\Delta}(w)|^2\}} \quad (\text{II.6})$$

Definindo-se a função de transferência do filtro redutor de ruído como sendo:

$$|H_{\Delta}(w)| = \frac{E\{|V_{\Delta}(w)|\}}{|Y_{\Delta}(w)|} \quad (\text{II.7})$$

obtém-se diretamente da equação II.7 um filtro redutor de ruído dado por:

$$|H_{\Delta}(w)| = \frac{\sqrt{|Y_{\Delta}(w)|^2 - E\{|R_{\Delta}(w)|^2\}}}{|Y_{\Delta}(w)|} \quad (\text{II.8})$$

Da equação II.8 verifica-se que o filtro obtido pela técnica da SE Clássica só é realizável se a potência do sinal ruidoso for maior ou igual à potência estimada do ruído. No entanto, como os sinais possuem fases aleatórias, isto não é garantido. Esta aleatoriedade na fase pode, no processo de adição dado na equação II.3, resultar na formação de um sinal ruidoso de potência inferior à do ruído, como indicado na figura II.1.

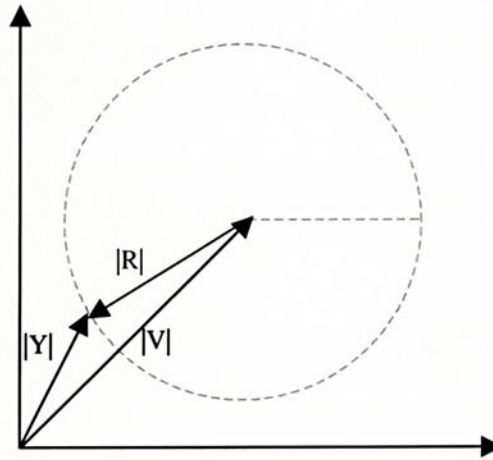


Figura II.1 – Mecanismo de formação do sinal ruidoso

Para a solução desse problema pode-se tomar o valor absoluto na equação II.5, definindo-se uma nova função de transferência, dada por:

$$|H_{1\Delta t}(w)| = \frac{\sqrt{|Y_{\Delta t}(w)|^2 - E\{|R_{\Delta t}(w)|^2\}}}{|Y_{\Delta t}(w)|} \quad (\text{II.9})$$

Outra solução é fazer uma “retificação de meia-onda”, que resultaria num filtro definido por:

$$|H_{2\Delta t}(w)| = \begin{cases} \frac{\sqrt{|Y_{\Delta t}(w)|^2 - E\{|R_{\Delta t}(w)|^2\}}}{|Y_{\Delta t}(w)|} & \text{se } |Y_{\Delta t}(w)|^2 \geq E\{|R_{\Delta t}(w)|^2\} \\ 0 & \text{caso contrário} \end{cases} \quad (\text{II.10})$$

No entanto, qualquer que seja a solução adotada, o resultado será uma alteração aleatória das amplitudes nas frequências onde isto ocorre, acarretando a geração de tons indesejáveis no sinal processado. Esses tons, que formam o conhecido *ruído musical*, são um dos maiores inconvenientes na aplicação da subtração espectral [2 e 4].

A figura II.2 ilustra o diagrama em blocos do algoritmo de redução de ruído em sinais de voz baseado somente na SE Clássica, destacando-se a re-utilização da fase do sinal ruidoso na reconstrução do sinal processado, como citado anteriormente.



Figura II.2 – Redução de ruído usando a SE Clássica

II.3 – Subtração Espectral Modificada (SEM)

Com o objetivo de eliminar o ruído musical presente na SE Clássica, mantendo-se a baixa carga computacional, um novo método foi proposto em [9]. Este método utiliza dois novos parâmetros denominados SNR_post e SNR_prio, definidos como:

$$SNR_post_{\Delta}(w) = \frac{|Y_{\Delta}(w)|^2}{|R_{\Delta}(w)|^2} \quad \text{e} \quad SNR_prio_{\Delta}(w) = \frac{|V_{\Delta}(w)|^2}{|R_{\Delta}(w)|^2} \quad (\text{II.11})$$

onde:

$SNR_post_{\Delta}(w)$ = Relação Sinal/ Ruído a posteriori para a janela analisada

$SNR_prio_{\Delta}(w)$ = Relação Sinal/ Ruído a priori para a janela analisada

Estes dois termos foram introduzidos inicialmente por McAulay & Malpass [6] e são os mais importantes do modelo redutor de ruído desenvolvido por Ephraim & Malah (MEM)¹ [3].

Na nova técnica usando a relação sinal/ruído, a função de transferência definida na SE Clássica é reescrita em função da Relação Sinal/Ruído a priori (SNR_prio).

¹ O MEM apresenta alta carga computacional, inviabilizando atualmente sua implementação em tempo real no TMS320C31. Por isso, sua teoria não foi discutida neste trabalho.

Das equações II.9 e II.11 tem-se:

$$|H_{\Delta}(w)| = \sqrt{1 - \frac{1}{E\{SNR_post_{\Delta}(w)\}}} \quad (II.12)$$

A equação II.12 mostra que a SE Clássica é função da Relação Sinal/Ruído a posteriori (SNR_post). De acordo com a figura II.1 e a equação II.11, o cálculo ou estimação da SNR_post pode levar a valores abaixo de 1, devidos também ao problema de fase. Isto exigirá uma correção na resposta do filtro estimado, o que, desta forma, não evitaria o ruído musical.

No entanto, mostra-se que existe uma relação entre a SNR_post e a SNR_prio . Assim, na equação II.5, dividindo-se todos os termos por $|R_{\Delta}(w)|^2$, que é a potência do ruído, obtém-se:

$$\frac{|Y_{\Delta}(w)|^2}{|R_{\Delta}(w)|^2} = \frac{|V_{\Delta}(w)|^2}{|R_{\Delta}(w)|^2} + 1 \quad (II.13)$$

De acordo com as definições da equação II.11, temos:

$$SNR_post_{\Delta}(w) = SNR_prio_{\Delta}(w) + 1 \quad (II.14)$$

Das equações II.12 e II.14, mostra-se facilmente que a subtração espectral pode ser redefinida, possibilitando a proposição de um novo filtro com uma resposta em amplitude dada por:

$$|H_{\Delta}(w)| = \sqrt{\frac{E\{SNR_prio_{\Delta}(w)\}}{E\{SNR_prio_{\Delta}(w)\} + 1}} \quad (II.15)$$

Pode-se também alterar a equação II.15 com a introdução de fatores de controle de redução de ruído, como a seguir:

$$|H_{\Delta}(w)|_{\rho,\lambda} = \left(\frac{E\{SNR_prio_{\Delta}(w)\}}{E\{SNR_prio_{\Delta}(w)\} + \rho} \right)^{\lambda} \quad (II.16)$$

Na equação II.16, ρ e λ são fatores que podem representar uma subestimação ou uma superestimação do ruído, dependendo dos valores a eles atribuídos. A figura II.3 ilustra curvas de $|H_{\Delta}(w)|_{\rho,\lambda}$ para alguns valores de ρ e λ , com variações da SNR_prio de 0,5 a 50.

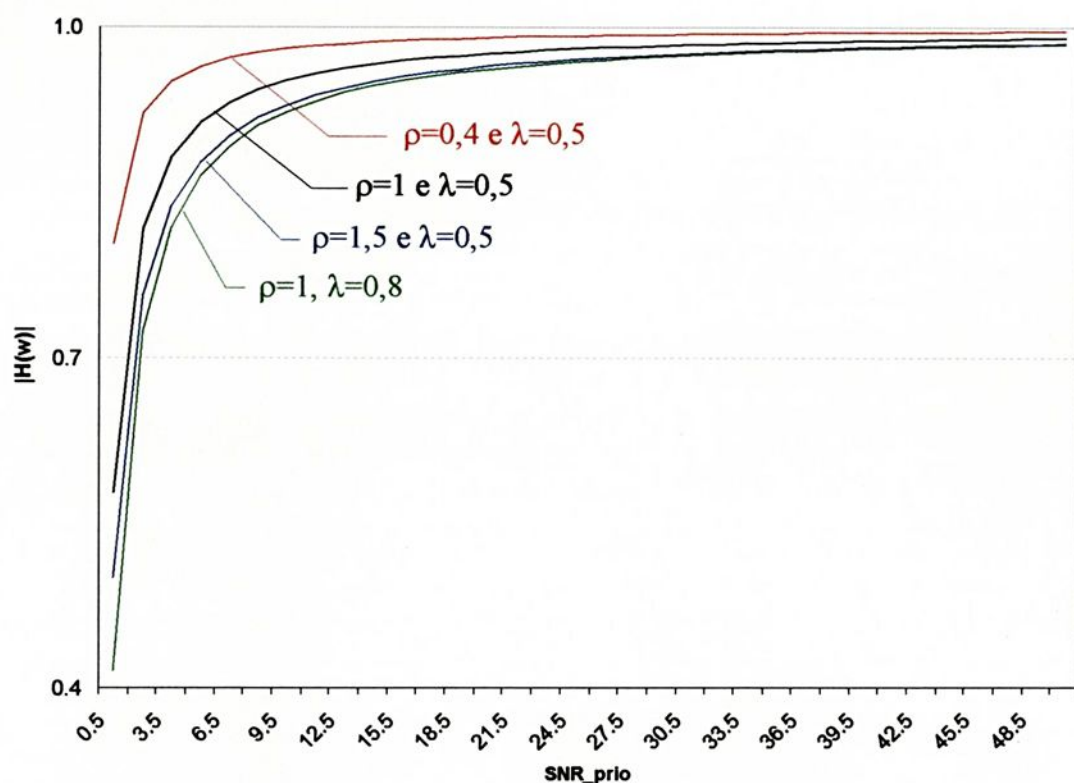


Figura II.3 – Funções de Transferência da SEM usando fatores de controle de redução de ruído

A equação II.16 apresenta uma nova técnica para o projeto de um filtro redutor de ruído que, deste modo, prevê uma estimação da Relação Sinal/Ruído a priori (SNR_prio). Observa-se que não há mais a retificação de meia-onda como mostrada na equação II.10, nem a alteração proposta na equação II.9.

Este novo modelo de filtro possibilita a minimização do ruído musical, garantindo ainda uma redução do ruído aditivo significativa. No entanto, para obter-se o resultado desejado, é preciso estimar corretamente a Relação Sinal/Ruído a priori (SNR_prio). Deve-se destacar que uma superestimação poderá causar distorções perceptíveis no sinal de voz processado, ao passo que uma subestimação produzirá um ruído musical forte. Portanto, é preciso encontrar

um meio-termo, que minimize estes dois fatores, objetivando a supressão do ruído musical com redução significativa do ruído aditivo. A estimação da SNR_{prio} será discutida no item II.4.5.

A figura II.4 ilustra o diagrama em blocos do algoritmo de redução de ruído em sinais de voz baseado na SEM, destacando-se a estimação da SNR_{prio} , fator essencial na obtenção da resposta desejada.



Figura II.4 – Redução de ruído usando a SEM

II.4 – Metodologia de Implementação da SEM

A implementação do filtro redutor de ruído utilizando a SEM envolve técnicas de processamento de sinais, como janelamento temporal do sinal amostrado, aplicação da Transformada de Fourier de Curto Prazo (STFT), etc. Isto é necessário para garantir a estacionariedade do sinal de voz ruidoso e para a estimação da relação sinal/ruído a priori (SNR_{prio}), que se baseia nos espectros de amplitude das janelas processadas pela STFT e nas estimações da potência do ruído aditivo e da relação sinal/ruído a posteriori (SNR_{post}).

Baseado nestes requisitos, tem-se, na figura II.5, a metodologia de implementação da SEM. Inicialmente o sinal de voz ruidoso amostrado passa por uma janela temporal de tamanho limitado, assegurando, assim, trechos do sinal que sejam praticamente estacionários e permitindo o uso da STFT. Após o cálculo da STFT tem-se, na frequência, um sinal complexo que é separado em fase e amplitude. A fase do sinal ruidoso é mantida e a filtragem é aplicada somente ao espectro de magnitude do sinal, como indicado no diagrama.

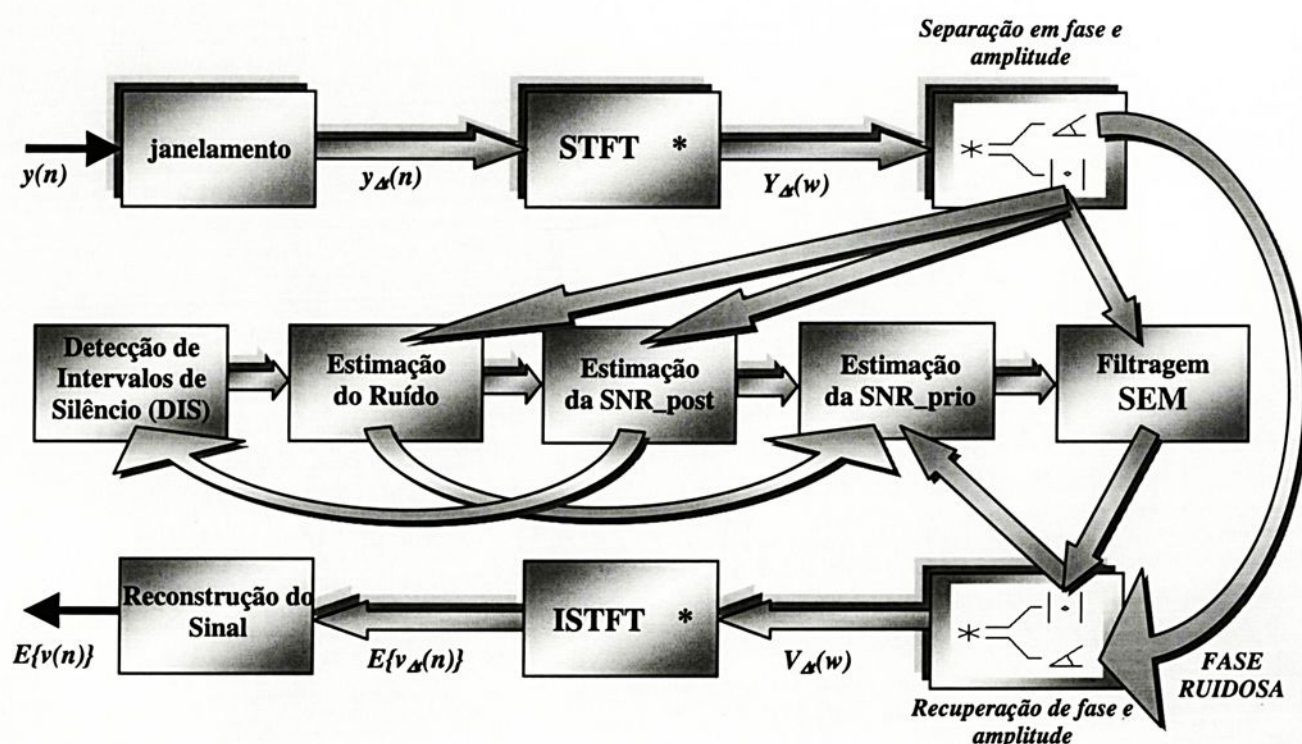


Figura II.5 – Metodologia de implementação do filtro redutor de ruído usando a SEM

O objetivo da Detecção de Intervalos de Silêncio (DIS)² é viabilizar a estimação da potência do ruído. A DIS implementada é baseada na variância da relação sinal/ruído a posteriori e será assunto do Capítulo III.

Na recuperação de fase e amplitude, usa-se a fase do sinal ruidoso e a magnitude do sinal de voz estimada pelo filtro para reconstruir o sinal complexo usado no cálculo da ISTFT. Finalmente, para reconstruir o sinal filtrado usa-se um método de síntese.

As seções seguintes descrevem, separadamente, a função de cada bloco do diagrama ilustrado na figura II.5.

² São considerados intervalos de silêncio os períodos entre duas falas consecutivas no sinal de voz ruidoso. Nestes intervalos, como não há voz, somente o ruído aditivo estará presente.

II.4.1 – Janelamento

O janelamento é aplicado ao sinal amostrado $y(n)$ e tem a função de tomar trechos de voz praticamente estacionários e de permitir o uso da STFT. Um diferente tipo de janela pode ser usado, como por exemplo a janela de Hamming ou a de Hanning. Associado à janela, deve-se definir também o intervalo de sobreposição das janelas consecutivas, que depende também do tipo de janela adotada. Um valor típico é 50% para uso com a janela de Hanning. O tamanho da janela deve ser o suficiente para garantir a estacionariedade do sinal de voz, não devendo exceder a 40 ms [7]. Por exemplo, para um sinal amostrado a 8 kHz, podem ser usadas janelas com 256 amostras, correspondentes a 32 ms.

II.4.2 – Transformada Rápida de Fourier de Curto Prazo (STFT)

A STFT calcula a Transformada de Fourier Discreta (DFT) para pequenos trechos do sinal de voz, considerados praticamente estacionários. A STFT calculada é um sinal discreto de amostras complexas. A partir desse sinal, pode-se obter os espectros de amplitude e fase da janela processada. A escolha do número de pontos para os espectros da STFT é um fator relevante, pois existe uma relação entre qualidade dos espectros e tempo de cálculo da STFT, isto é, com mais pontos têm-se espectros mais definidos, porém o processamento será mais lento e vice-versa. Um valor típico para o projeto proposto seria uma STFT de 256 pontos.

II.4.3 – Estimação do Ruído

A filtragem proposta é função da Relação Sinal/Ruído a priori (SNR_prio) conforme mostrado na equação II.16. Para a estimação da SNR_prio é necessário primeiramente estimar a potência do ruído aditivo, como apresentado no diagrama da figura II.5.

A potência do ruído será medida nos intervalos de silêncio definidos pela DIS, a partir das janelas do sinal ruidoso desses intervalos. Para evitar que, no cálculo do ruído estimado, transições abruptas ocorram entre janelas adjacentes, suaviza-se o resultado por meio de uma filtragem passa-baixas de primeira ordem [3]. Assim, obtém-se a potência estimada do ruído, como segue:



$$E\{|R_{\Delta}(w)|^2\}_k = \alpha.E\{|R_{\Delta}(w)|^2\}_{k-1} + (1-\alpha).\{|Y_{\Delta}(w)|^2\}_{k-m} \quad (\text{II.17})$$

onde k é o índice de interação, incrementado a cada janela analisada. A equação II.17 introduz uma dependência entre a potência do ruído atual em relação à potência do ruído das janelas anteriores. A constante α , que pode ser interpretada como um fator de esquecimento, controla o grau de dependência e varia normalmente entre 0 e 1. Esta estimação pode evitar que ocorra uma variação brusca de potência de uma janela para a outra, o que contribuiria fortemente para o surgimento do ruído musical. Portanto, deve-se calcular adequadamente um valor para α . O cálculo de α é baseado no tamanho da janela de análise, da frequência de amostragem do sinal e da constante de tempo que garanta o esquecimento das informações. Para o filtro a ser implementado, trabalhando-se com quadros de sinal de 128 amostras³, frequência de amostragem de 8 kHz e considerando um tempo de esquecimento padrão de 140 ms⁴, um valor típico para α é de aproximadamente 0,86.

É necessário estimar um valor inicial para a potência média do ruído, antes que a Detecção de Intervalos de Silêncio (DIS) comece a atuar. Destaca-se que a DIS é baseada na relação sinal/ruído a posteriori (SNR_post), como será visto no Capítulo III. Como a SNR_post depende da própria potência estimada do ruído (eq. II.11), esta inicialização deve ser garantida para que a DIS comece a atuar. A estimação inicial da potência média do ruído pode ser obtida a partir de algumas janelas anteriores da potência do sinal ruidoso $|Y(w)|$, considerando que estas janelas contenham somente trechos de ruído.

O fator m presente na equação II.17, representa um atraso aplicado ao sinal ruidoso usado na atualização da potência média estimada do ruído. Este atraso, que será discutido no Capítulo III, faz-se necessário devido ao método de detecção de intervalos de silêncio empregado. Destaca-se que o valor atribuído a m deverá ser um número inteiro. Este valor determina o número de janelas atrasadas.

³ Um quadro de 128 amostras do sinal ruidoso amostrado corresponde a janelas de análise de 256 amostras, considerando um "overlap" de 50%, necessário para o janelamento do sinal usando, por exemplo, janela de Hanning.

⁴ Após 140 ms, deseja-se que a potência média obtida não considere mais do que 25% dos valores passados.



II.4.4 – Estimação da Relação Sinal/Ruído a posteriori (SNR_post)

A estimação da relação sinal/ruído a posteriori (SNR_post) pode ser obtida diretamente do espectro de amplitude do sinal ruidoso captado e da potência do ruído estimada. Assim, considerando-se que $|Y_{\Delta}(w)|^2$ representa o espectro de potência do sinal ruidoso para a janela analisada, tem-se:

$$E\{SNR_post_{\Delta}(w)\} = \frac{|Y_{\Delta}(w)|^2}{E\{|R_{\Delta}(w)|^2\}} \quad (II.18)$$

Verifica-se que a SNR_post é diretamente proporcional a variações do sinal ruidoso, o que confirma sua sensibilidade em relação às mudanças bruscas de fase do ruído.

II.4.5 – Estimação da Relação Sinal/Ruído a priori (SNR_prio)

A estimação da Relação Sinal/Ruído a priori (SNR_prio) é fundamental para a realização da filtragem usando a SEM, visto que a função de transferência para este filtro é função direta do valor estimado da SNR_prio, como mostrado na equação II.16. Esta estimação depende do próprio sinal de voz estimado, o que caracteriza um sistema não causal. Para resolver este problema, considera-se que a variação de potência de uma janela para outra não é significativa. Assim, utiliza-se um estimador recursivo baseado no valor estimado da SNR_prio da janela anterior e no valor da SNR_post da janela analisada [3]. Este estimador é definido como a seguir:

$$E\{SNR_prio_{\Delta}(w)\}_k = \beta \cdot \frac{E\{|V_{\Delta}(w)|^2\}_{k-1}}{E\{|R_{\Delta}(w)|^2\}_k} + (1 - \beta) \cdot T[E\{SNR_post_{\Delta}(w)\}_k - 1] \quad (II.19)$$

onde k é o índice de interação, incrementado a cada janela analisada, $T[\bullet]$ indica uma transformação sobre $E\{SNR_post_{\Delta}(w)\} - 1$ e β representa o grau de dependência da SNR_prio calculada com base no valor estimado da voz para a janela anterior, em relação a SNR_prio calculada a partir da SNR_post da janela analisada, lembrando-se que $SNR_post_{\Delta}(w) = SNR_prio_{\Delta}(w) + 1$, conforme definido na equação II.14. Como o valor estimado da SNR_post pode ser menor que 1, deve-se utilizar a seguinte transformação $T[\bullet]$:

$$\text{Se } (E\{SNR_post_{\Delta}(w)\} - 1) \leq \varepsilon \quad \text{então } (E\{SNR_post_{\Delta}(w)\} - 1) = \varepsilon \quad (\text{II.20})$$

Esta limitação de $(E\{SNR_post_{\Delta}(w)\} - 1)$ tem a função de minimizar o ruído musical. Adotando-se $\varepsilon=0$ estar-se-ia realizando a “retificação de meia-onda” proposta anteriormente na equação II.10, o que significa uma potência nula para a voz, na frequência onde isto ocorre. Assim, propõe-se um valor mínimo positivo para ε .

Voltando-se à equação II.19, tem-se que a contribuição da SNR_post na SNR_prio depende do valor de β . Verificou-se que fortes variações na SNR_post podem afetar o desempenho da SNR_prio . Assim, o valor típico adotado para β varia entre 0,9 e 1.

II.4.6 – Filtragem usando a SEM

A cada janela processada, a filtragem é realizada aplicando-se a função de transferência da SEM ao espectro de amplitude do sinal ruidoso da janela em análise. Uma implementação prática eficiente para processamento em tempo-real é fazer $\lambda=1/2$ na equação II.16. Com esta atribuição, a função de transferência implementada é a seguinte:

$$|H_{\Delta}(w)| = \sqrt{\frac{E\{SNR_prio_{\Delta}(w)\}}{E\{SNR_prio_{\Delta}(w)\} + \rho}} \quad (\text{II.21})$$

O parâmetro ρ deverá funcionar como um ajuste fino para o nível de redução de ruído desejado, destacando-se que valores inadequados de ρ poderão provocar maiores distorções no sinal processado. Assim, o valor típico adotado para ρ varia entre 0,5 e 1,5.

Usando-se o valor estimado da SNR_prio para a janela analisada, a filtragem usando a SEM envolve poucas operações matemáticas, como mostrado na equação II.21.

No diagrama da figura II.5 pode-se observar que somente o espectro de amplitude do sinal é filtrado, mantendo-se a fase do sinal ruidoso.

II.4.7 – Transformada Inversa de Fourier de Curto Prazo (ISTFT)

A ISTFT calcula a Transformada Inversa de Fourier Discreta (IDFT) para as janelas do sinal, consideradas estacionárias. Este procedimento é necessário para a reconstrução do sinal no tempo.

O bloco correspondente a ISTFT recebe sinais complexos proveniente do bloco de recuperação de fase e amplitude, ilustrado no diagrama da figura II.5, onde a fase ruidosa e o espectro estimado de amplitude do sinal de voz filtrado são convertidos nos valores complexos usados como entrada para a ISTFT.

II.4.8 – Reconstrução do sinal no tempo

A reconstrução do sinal no tempo é feita por um método de síntese. O método mais adotado baseado na STFT é o “Overlap Addition” (OLA), podendo também ser utilizado um banco de filtros [8].

II.5 – Conclusões

Neste capítulo foram apresentadas as técnicas de redução de ruído baseadas na SE Clássica e na SEM, detalhando-se o algoritmo implementado.

Com relação à implementação, destacou-se a importância de uma correta estimação do ruído aditivo e da SNR_prio, objetivando a eliminação do ruído musical com significativa redução de ruído aditivo. Também foi verificado que estas estimações são baseadas nas janelas anteriores processadas, incorporando-se no entanto uma parcela de informações da janela atual de análise. Este tipo de estimação é denominada de “decisão dirigida” e não é linear, o que dificulta uma análise teórica do seu comportamento.



Com o objetivo de controlar o nível de redução de ruído utilizou-se o parâmetro ρ na função de transferência do filtro. A inclusão deste parâmetro é de fácil implementação e funciona como um fator de superestimação ou subestimação do ruído, dependendo do valor a ele atribuído.

Mostrou-se que um ponto importante para a implementação do filtro redutor de ruído é a Detecção de Intervalos de Silêncio (DIS), a partir da qual estima-se a potência do ruído. Este fato é fundamental, pois os parâmetros básicos do filtro, isto é, as relações sinal/ruído a posteriori e a priori, são baseados na potência média estimada do ruído.



CAPÍTULO III

DETECÇÃO DE INTERVALOS DE SILÊNCIO EM SINAIS DE VOZ USANDO A VARIÂNCIA DA RELAÇÃO SINAL/RUÍDO (DISS)

III.1 – Introdução

A detecção de intervalos de silêncio (DIS) no sinal de voz ruidoso é necessária para a estimação do ruído aditivo, fator essencial no cálculo da função de transferência do filtro. Nos intervalos de silêncio, como não há voz, somente o ruído aditivo estará presente. Desta maneira, pode-se medir o nível de ruído nestes intervalos e conseqüentemente estimar o comportamento do ruído aditivo presente no sinal de voz, considerando que o mesmo é estacionário.

A detecção de intervalos de silêncio em sinais de voz ruidosos não é tarefa fácil. Para um sinal de voz com uma baixa relação sinal/ruído (SNR), torna-se difícil identificar estes intervalos. Um bom detector deve ser capaz de identificar trechos de ruído em sinais com SNRs relativamente baixas e também nos casos em que o nível médio de ruído varia com o tempo.

Neste trabalho, visando a obtenção de um detector eficiente, *uma nova técnica de detecção de intervalos de silêncio foi desenvolvida*. Nesta nova técnica, a detecção é baseada na variância da relação sinal/ruído a posteriori (SNR_{post}) e foi denominada *Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS)*

III.2 – Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS)

A DIS implementada no filtro é baseada na variância da relação sinal/ruído a posteriori estimada ($E\{SNR_{post}\}$). O algoritmo da figura III.1 ilustra este método.



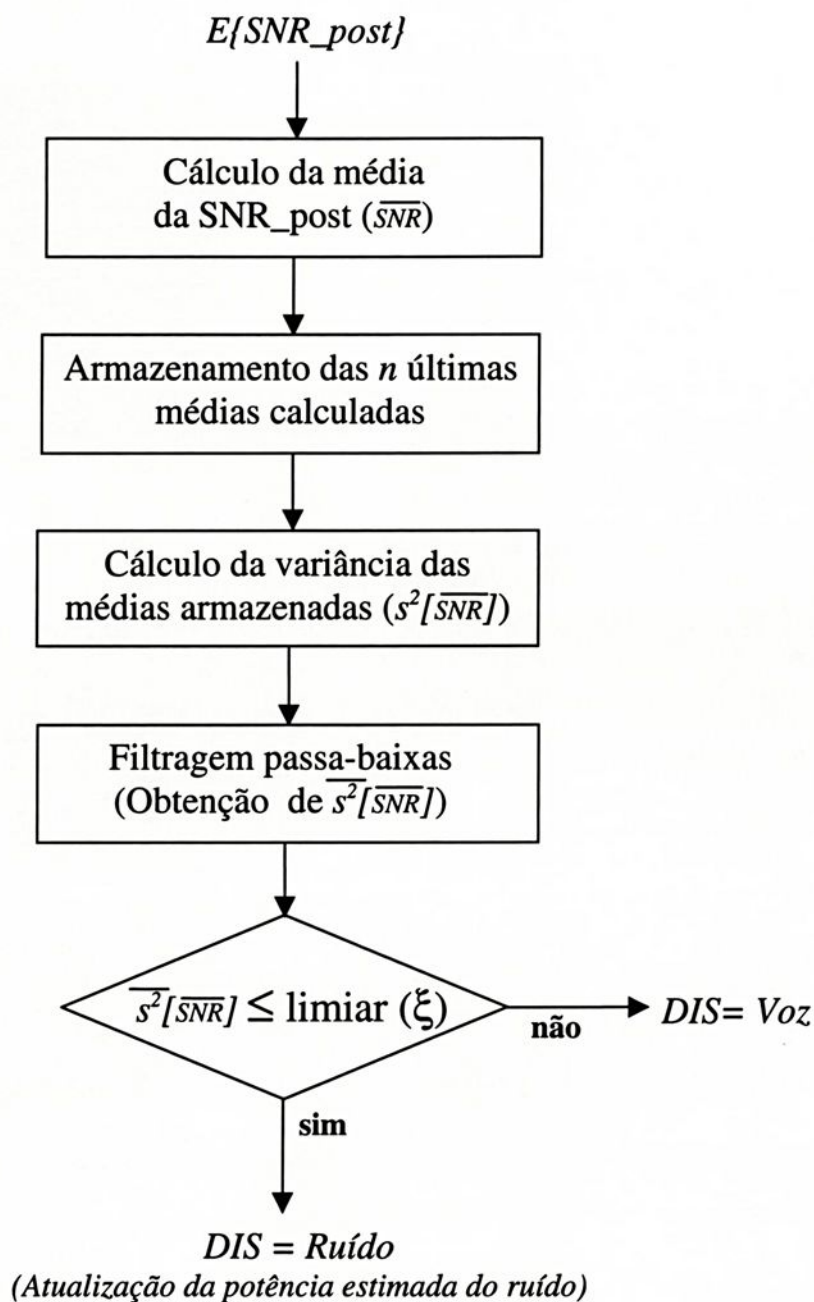


Figura III.1 – Algoritmo da DIS baseada na variância da SNR (DISS)

Após uma estimativa inicial da potência média do ruído usando algumas janelas anteriores do sinal ruidoso, a SNR_post é calculada e a DISS estará apta a atuar automaticamente. Para simplificação, neste capítulo, a SNR_post será denominada apenas de SNR.

A DISS considera as variações do valor médio da SNR estimada. Inicialmente, calcula-se a média aritmética das amostras do espectro estimado da SNR, como mostra o diagrama da figura III.1. Posteriormente, são armazenadas as n últimas médias calculadas, correspondente a um intervalo de n janelas de sinal. O armazenamento é necessário para o cálculo da variância destas n médias, que é realizado a cada janela analisada. Os valores da variância nos intervalos de silêncio e voz apresentam diferenças em valores absolutos bem superiores as obtidas com a SNR. Portanto, a partir deste sinal de variância, torna-se mais fácil estabelecer um limiar de decisão para a DIS, que no diagrama da figura III.1 é identificado como ξ . Usar como base um sinal que permite a definição de um limiar de decisão menos crítico torna o sistema de detecção mais estável, garantindo uma melhor detecção, o que justifica o uso da variância da SNR para a DIS. Além disso, como as variâncias calculadas nos trechos de voz, bem como nos intervalos de silêncio (ruído), apresentam variações em torno de níveis médios, utiliza-se uma filtragem passa-baixas de primeira ordem para suavizar a curva das variâncias. O sinal resultante pode ser visto como uma “variância média” ($\overline{s^2[SNR]}$). Este valor é comparado a um limiar de decisão pré-fixado (ξ) para a identificação dos intervalos de silêncio. A filtragem passa-baixas corresponde a um sistema recursivo, que atualiza a “variância média”, como segue:

$$\overline{s^2[SNR]}_k = \gamma \cdot \overline{s^2[SNR]}_{k-1} + (1-\gamma) \cdot s^2[SNR]_k \quad (\text{III.1})$$

onde k é o índice de interação incrementado a cada janela analisada, $s^2[SNR]$ é a variância calculada e $\overline{s^2[SNR]}$ representa a “variância média” calculada pela filtragem passa-baixas. A atualização desta média é feita a cada janela analisada.

A equação III.1 introduz uma dependência entre a “variância média” atual e a “variância média” calculada na interação anterior. A constante γ , que pode ser interpretada como um fator de esquecimento, controla o grau de dependência e varia normalmente entre 0 e 1.

Finalmente, a “variância média” calculada na filtragem passa-baixas de primeira ordem é comparada a um limiar de decisão (ξ). Neste projeto, considerou-se ξ constante, pois a atualização de ξ aumentaria a complexidade do sistema e não garantiria uma melhor detecção.

O limiar de decisão (ξ) adotado, juntamente com os valores atribuídos a γ e n , definem a sensibilidade do detector. Normalmente, para uma detecção capaz de operar com sinais ruidosos de baixa SNR, o limiar ξ deve ser um valor pequeno, menor do que 1. Destaca-se

também que, para valores fixos de γ e n , quanto maior for o valor de ξ , menor será a sensibilidade e um sistema de detecção menos sensível deve ser aplicado nos casos em que uma SNR relativamente alta seja garantida. Caso contrário, deve-se atribuir a γ , n e ξ , valores que assegurem a DIS para baixas SNRs.

Com relação aos parâmetros n e γ , estes podem causar um atraso na detecção, suficiente para comprometer a DISS. O problema ocorre quando trechos de voz são identificados como intervalos de silêncio, levando a uma atualização da potência do ruído com trechos de voz. Para resolver este problema, propõe-se o uso do atrasador definido na estimação da potência média do ruído (equação II.17 do capítulo anterior), onde o valor de m determinou o número de janelas atrasadas, devendo corresponder, teoricamente, ao atraso máximo provocado pelos fatores n e γ . Assim, para garantir a estabilidade do sistema, a atualização do ruído deverá ocorrer após, no mínimo, $m=n+1$ janelas. Este fato poderá ser comprovado na análise dos resultados obtidos, assunto da próxima seção.

A DIS usando a variância da SNR é robusta. Este método de detecção é sempre estável e um eventual erro se propaga somente até o próximo intervalo de silêncio, quando a “variância média” e a potência média estimada do ruído são atualizadas. A duração do intervalo de silêncio, necessário para que a potência da voz seja eliminada da potência média do ruído, depende principalmente dos valores definidos para os parâmetros n e γ . O desenvolvimento seguinte mostra o que ocorre se um trecho de voz provocar uma superestimação da potência do ruído, e a correção deste erro, considerando-se para este caso, um nível de ruído constante. Assim, considera-se que, na ocorrência de um trecho de voz, um erro na DIS atualizou a potência estimada do ruído com janelas deste intervalo, fazendo com que:

$$E\{|R_{\Delta r}(w)|^2\} > |R_{\Delta r}(w)|^2 \quad (\text{III.2})$$

Supondo que o ruído mantém-se constante até que o próximo intervalo de silêncio ocorra, tem-se:

$$E\{SNR_{post_{\Delta r}}(w)\} = \frac{|Y_{\Delta r}(w)|^2}{E\{|R_{\Delta r}(w)|^2\}} = \frac{|R_{\Delta r}(w)|^2}{E\{|R_{\Delta r}(w)|^2\}} \quad (\text{III.3})$$



Analisando as equações III.2 e III.3, conclui-se que $E\{SNR_{post}\} < 1$, indicando uma diminuição no valor estimado da SNR_{post} , que conseqüentemente provocará também uma diminuição na variância da SNR . Considerando que este último intervalo de silêncio é relativamente grande, o suficiente para que a “variância média”, obtida com o sistema recursivo da figura III.1, seja menor do que o limiar de decisão, a potência estimada do ruído voltará a ser estimada de forma correta. A partir desse ponto, um período máximo, que corresponde ao atraso provocado pelo fator α usado na estimação recursiva da potência do ruído (eq. II.17), garante a correção do erro e a estabilidade do sistema.

Generalizando, usar a variância da SNR como base para a detecção garante duas vantagens diretas. A primeira é que a DIS sempre atuará na ocorrência de intervalos de silêncio relativamente grandes, em todos os casos em que o nível de ruído diminuir durante o processamento do sinal ruidoso. Outra vantagem importante é que não é necessário atualizar o limiar de decisão (ξ), já que esta atualização seria apenas um ajuste fino que consumiria tempo de processamento e memória. Utilizar um limiar constante simplifica os cálculos, fato que deve ser considerado em uma implementação em tempo real.

Analisando-se somente trechos de ruído, conclui-se do desenvolvimento anterior que, no caso em que o nível de ruído diminui, a atualização da potência do ruído será sempre assegurada.

Por outro lado, a DIS baseada na variância da SNR também detecta variações no nível de ruído, quando este aumenta. Uma variação abrupta no nível do ruído é considerada como voz pelo detector, de maneira que a média estimada do ruído não seja atualizada neste caso. Em se tratando de uma variação abrupta transitória, o detector funciona corretamente, pois este efeito transitório não deve ser considerado na estimação do ruído. Porém, no caso em que ocorrer um aumento brusco no nível do ruído e esta alteração se mantiver por um tempo relativamente grande, a não atualização da média do ruído causará um erro na estimação. Este erro significa que, no processo de filtragem, a redução de ruído no sinal de voz processado será menor. Elevações lentas no nível do ruído são bem identificadas pelo detector, o que garante uma atualização da potência média estimada do ruído nestes casos. Existe um compromisso entre a máxima variação que o detector consegue identificar e a sua sensibilidade, definida pelos parâmetros ξ , γ e n . Quanto menos sensível for o detector, menor será a capacidade de detectar um aumento no nível médio de ruído.

Todas as características da DIS podem ser comprovadas pela análise dos resultados obtidos, discutidos na próxima seção.



III.3 – Resultados Obtidos

A DISS foi testada usando a mesma plataforma de desenvolvimento e execução do filtro redutor de ruído baseado na SEM. O detector implementado foi testado, operando em tempo real. O ambiente de desenvolvimento, bem como as etapas de elaboração deste detector e a obtenção e análise de diversos tipos de resultados obtidos no processamento, serão assuntos dos capítulos IV e V. Adianta-se que os gráficos aqui apresentados e analisados foram obtidos no ambiente de desenvolvimento Hypersignal RIDE e a execução do detector foi realizada numa placa TIGER 31/PC com 128 Kbytes de memória SRAM, que incorpora um processador de sinais digitais TMS320C31 operando a 60 MHz. Além disso, o sinal ruidoso processado foi amostrado a 8 kHz, as janelas de análise eram de 16 ms (256 amostras) com overlap de 50% e FFT de 256 pontos. O detector foi configurado com as seguintes atribuições: $n=4$ janelas (64 ms), $\gamma=0,8$ e um limiar de decisão $\xi=0,1$. Também, a estimação recursiva da potência do ruído foi configurada com $m=5$ janelas, correspondente a um atraso de 80 ms.

Os gráficos obtidos mostram a $\overline{\text{SNR}}$ (em azul), a “variância média” da $\overline{\text{SNR}}$ (em vermelho), áreas sombreadas em amarelo correspondentes a trechos de fala (se houver) e áreas sombreadas em cinza relativas a DISS. Estes gráficos foram obtidos e analisados para várias situações envolvendo trechos de voz ruidosa e trechos onde somente o ruído aditivo está presente. Estas situações são descritas a seguir.

Trecho de voz com baixa SNR

O sinal de voz ruidoso com baixa SNR, utilizado para os testes da DISS é apresentado na figura III.2. O ruído aditivo presente no sinal ruidoso corresponde a um ruído gerado por um carro em movimento (com o motor ligado). As figuras III.3 e III.4 mostram, respectivamente, os gráficos da energia e do número de cruzamentos de zero deste sinal⁵. Nota-se, pela análise destes gráficos, que é praticamente impossível estabelecer um limiar de decisão que identifique corretamente os intervalos de silêncio no sinal de voz analisado e, neste caso, um método convencional baseado nestes sinais ocasionaria erros na detecção.

⁵ Energia e Taxa de cruzamentos de zero, avaliados a cada janela de sinal, são parâmetros usados em muitos sistemas de detecção de intervalos de silêncio. A apresentação dos mesmos neste trabalho, permite uma comparação, mostrando a dificuldade em se estabelecer um limiar de decisão.



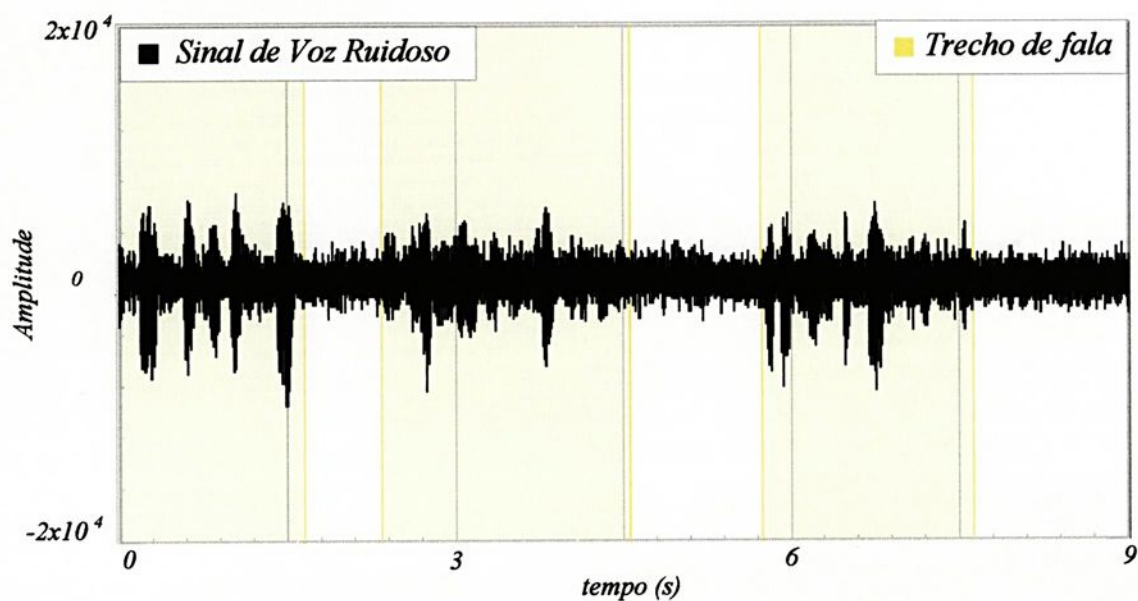


Figura III.2 – Sinal ruidoso com baixa SNR, usado para avaliação da DISS

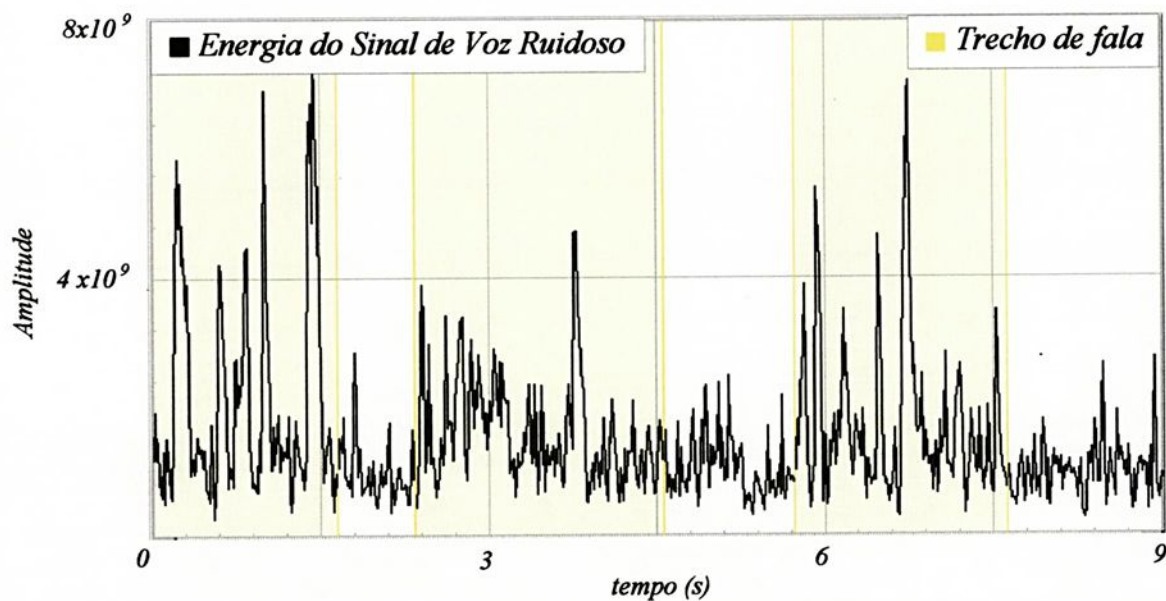


Figura III.3 – Energia do Sinal ruidoso da figura III.2

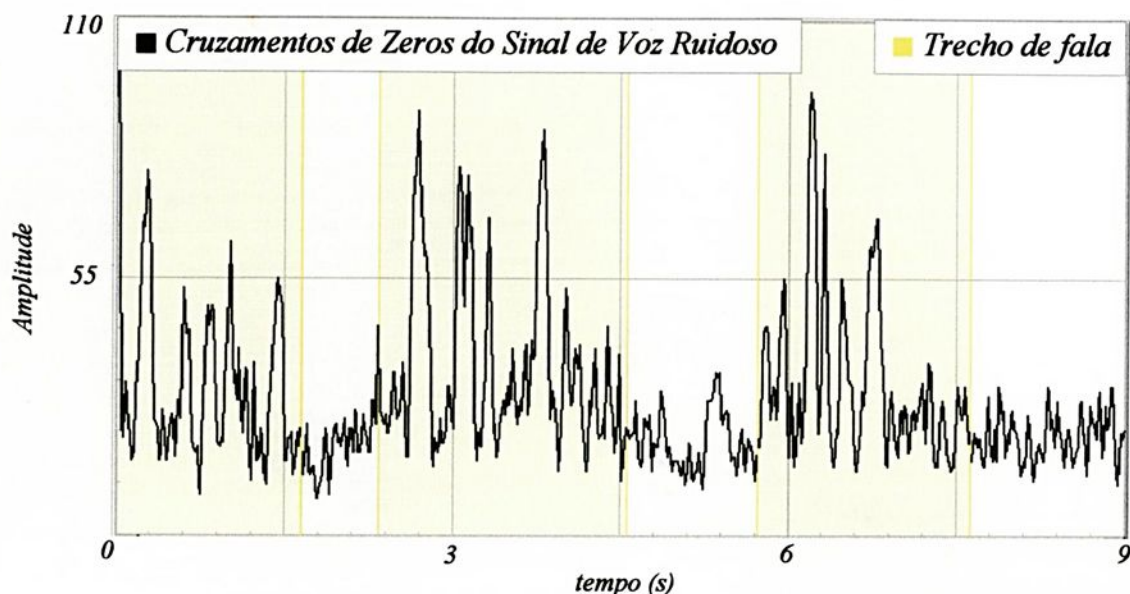


Figura III.4 – Cruzamentos de Zero do Sinal ruidoso da figura III.2

O gráfico da figura III.5 apresenta a “variância média” da $\overline{SNR}$ para o sinal ruidoso mostrado na figura III.2. Neste gráfico, também é mostrado a $\overline{SNR}$ do sinal ruidoso. Nota-se que a “variância média” tem grande diferença em relação a $\overline{SNR}$, que ocorre porque, nos trechos de silêncio, a $\overline{SNR}$ tem pequenas variações em torno de 1 (um), e em trechos de voz, a $\overline{SNR}$ atinge valores relativamente altos e tem variações bem maiores. Esta diferença tornou menos crítica a definição de um limiar de decisão (ξ) para a $DISS$, diferente também do que ocorre nos sinais de energia e cruzamentos de zero, mostrados, respectivamente, nas figuras III.3 e III.4.

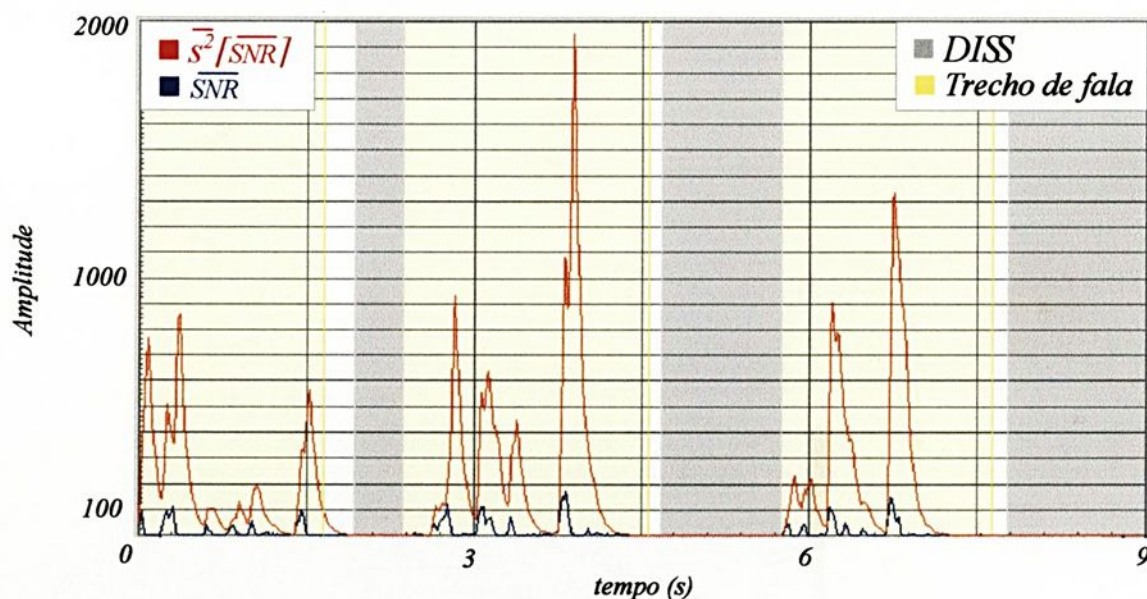


Figura III.5 – Análise da DISS para o sinal ruidoso da figura III.2

Trecho de voz com baixa SNR (análise do limiar de decisão)

Para tornar evidente a definição de limiares de decisão, tanto para a “variância média” da $\overline{SNR}$ (ξ), como para a própria $\overline{SNR}$, apresenta-se, na figura III.6, o mesmo trecho de voz analisado anteriormente, destacando-se os valores menores de amplitude (“zoom”). Observa-se que, para o limiar de decisão adotado ($\xi = 0,1$), a detecção atua corretamente, não atingindo o trecho de voz indicado no gráfico. Analisando a $\overline{SNR}$, conclui-se que é praticamente impossível encontrar um limiar mínimo que atualize o ruído somente nos intervalos de silêncio, e nunca o atualize em trechos de voz.

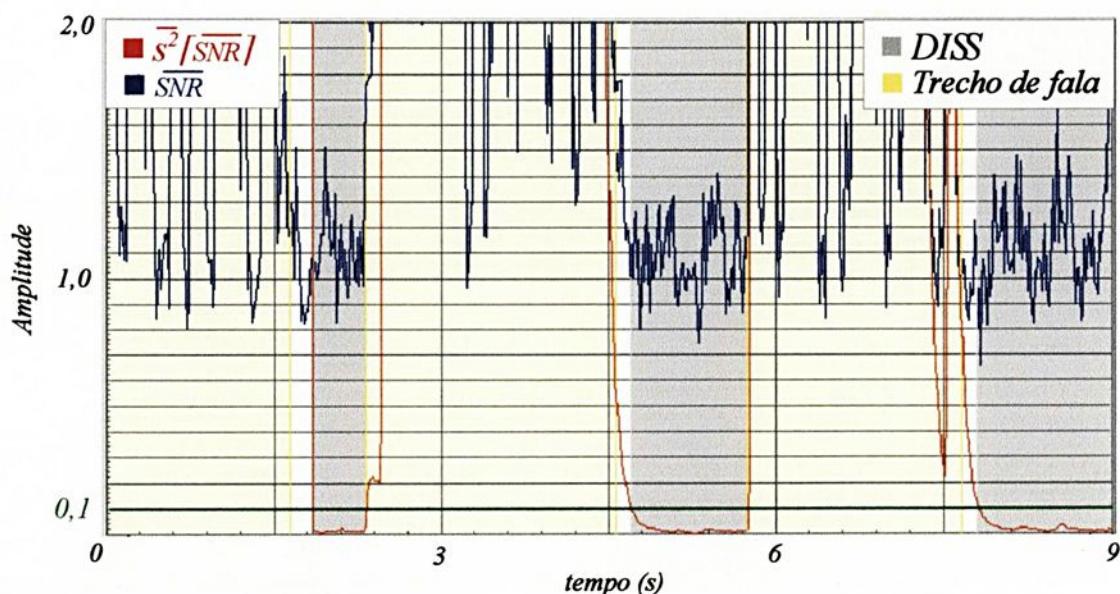


Figura III.6 – Análise do limiar de decisão (ξ) para um sinal ruidoso da figura III.2

Trecho de voz com SNR relativamente alta (análise do limiar de decisão)

A figura III.7 considera um sinal de voz ruidoso com SNR relativamente alta, onde dois trechos curtos de fala contendo as expressões “alô” e “teste” foram processados. Como a variância tem uma mudança de nível abrupta e muito forte na transição silêncio/voz, o atraso provocado pela filtragem passa-baixas de primeira ordem utilizada para o cálculo da “variância média” é muito pequeno na transição silêncio/voz e relativamente grande na

transição voz/silêncio. Para sinais com baixas SNRs, o efeito tenderia a ser menor. Este atraso deve ser considerado na definição do fator m definido na equação de atualização da potência média estimada do ruído (eq. II.17). Destaca-se que este sistema recursivo considera pouco a variância calculada para as janelas atuais e aplica um peso maior às variâncias das SNRs processadas anteriormente ($\gamma=0,8$). Porém, nos casos de transição silêncio/voz, como a variância das janelas do trecho inicial da voz é muito superior à última “variância média” calculada no trecho de silêncio, isto provocará um aumento significativo da média e quando comparada a um limiar de decisão ξ pequeno, fará com que a DISS atue rapidamente e pare de atualizar a potência estimada do ruído.

Por outro lado, nas transições voz/silêncio, o atraso provocado pelo sistema recursivo tem um efeito oposto, isto é, quanto maior a SNR, maior será o tempo necessário para a DISS identificar o intervalo de silêncio e começar a atualizar a potência estimada do ruído. Isto ocorre porque a “variância média” da $\overline{SNR}$ no trecho de voz é muito alta se comparada ao valor da variância no trecho de silêncio. Portanto, a atualização da “variância média” com a ocorrência de um intervalo de silêncio provocará um retardamento na atuação da DISS. No caso de uma conversação telefônica, por exemplo, desde que este tempo seja o suficiente para que o ruído seja atualizado corretamente, não há inconvenientes. Vale lembrar que este atraso pode ser controlado pela definição do fator de esquecimento γ usado pelo sistema recursivo da equação III.1

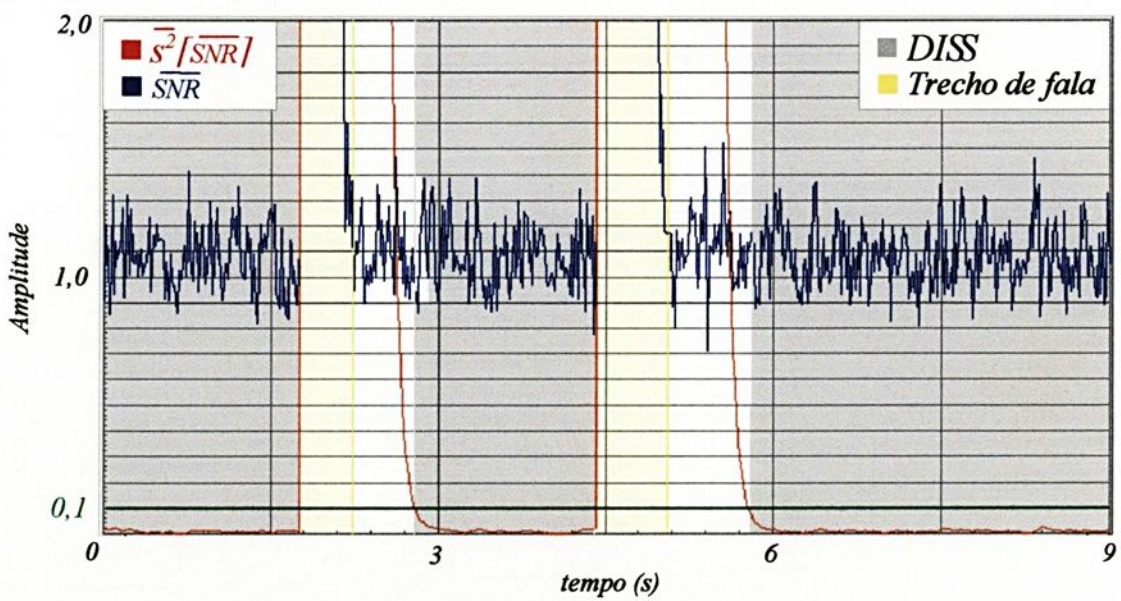


Figura III.7 – Análise do limiar de decisão (ξ) para um sinal ruidoso com alta SNR

Trecho de ruído (análise do limiar de decisão)

A figura III.8 foi obtida analisando-se somente um trecho de ruído, onde é possível visualizar o que ocorre em torno do limiar de decisão (ξ) quando o nível de ruído varia. No caso em que o nível de ruído diminui, o sistema tende a estabilizar-se e a atualização da potência estimada do ruído é realizada. No trecho de aumento do nível de ruído, o sistema também se estabilizou e atualizou a potência estimada do ruído.

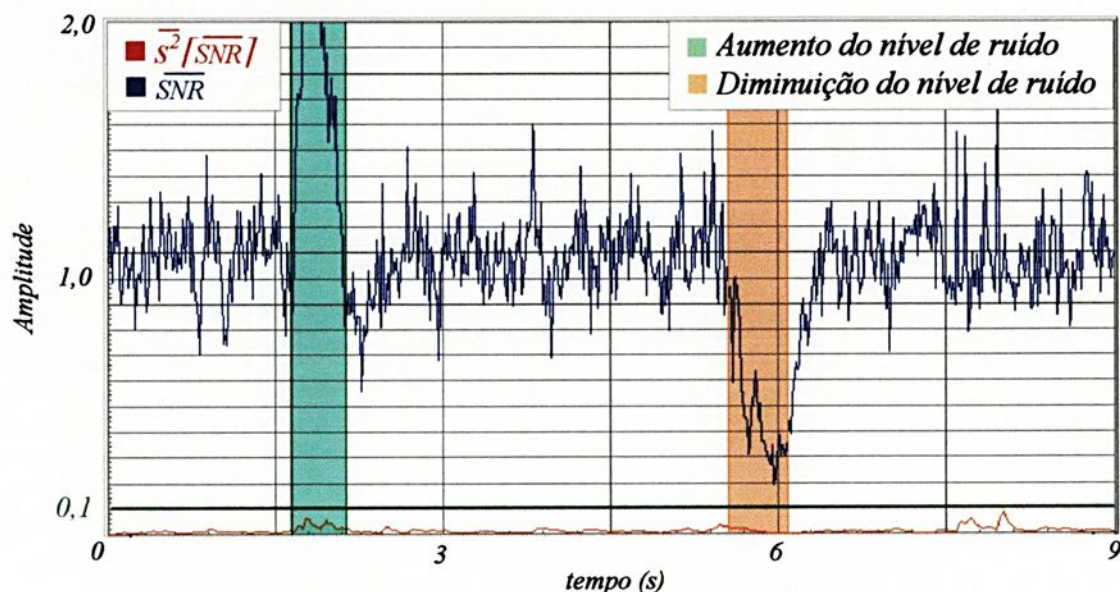


Figura III.8 – Variação do nível de ruído em intervalo de silêncio

III.4 – Conclusões

O método de detecção proposto neste capítulo, apresenta excelente desempenho, usando como parâmetro básico a variância da SNR. Inicialmente, a idéia era adotar uma detecção baseada somente na SNR. Porém, estudos e experimentos levaram ao desenvolvimento desta nova metodologia. A análise feita neste capítulo não foi das mais profundas, mas permite propor um novo método de detecção de intervalos de silêncio, com as seguintes características:

- uso de um limiar de decisão (ξ) fixo;
- decisões tomadas a partir da “variância média” da $\overline{SNR}$, onde é possível distinguir melhor, trechos de voz e silêncio, se comparado com outros métodos;
- atualização do nível de ruído quando este varia.

A sensibilidade do detector pode ser controlada pelo ajuste dos parâmetros ξ , γ e n . Melhores resultados podem ser obtidos se, para o ajuste destes parâmetros, for considerada também a aplicação (telefonia fixa, telefonia móvel, reconhecimento de fala, etc).



CAPÍTULO IV

IMPLEMENTAÇÃO DO FILTRO REDUTOR DE RUÍDO EM TEMPO REAL USANDO O TMS320C31

IV.1 – Introdução

O estudo apresentado no capítulo II mostrou uma nova técnica de redução de ruído em sinais de voz baseada na Subtração Espectral Clássica. Esta técnica, desenvolvida em [9], foi denominada *Subtração Espectral Modificada (SEM)*. Além disso, um novo método de Detecção de Intervalos de Silêncio (DIS) utilizando a variância da SNR foi proposto. Este método, denominado *DISS*, está descrito no capítulo III. Neste capítulo, apresenta-se o software desenvolvido para a implementação desse algoritmo em tempo real, assim como a plataforma de desenvolvimento e operação deste software. Esta plataforma é composta de uma placa TIGER 31/PC acoplada a um *slot* ISA de um microcomputador (PC), a partir do qual foi desenvolvido o projeto do filtro redutor de ruído usando um software de desenvolvimento denominado Hypersignal RIDE. Um aparelho de som (*microsystem*) também foi utilizado para reprodução e gravação dos sinais, facilitando uma avaliação subjetiva do filtro.

Assim, neste capítulo, será descrita inicialmente a plataforma de desenvolvimento e operação do filtro, composta basicamente da placa TIGER 31/PC e do software Hypersignal RIDE. Destaca-se que a placa TIGER 31/PC possui um processador de sinais digitais TMS320C31 e que o software RIDE permitiu o desenvolvimento do filtro em ambiente gráfico, com a implementação e interligação de blocos funcionais.

Posteriormente, o diagrama de blocos do filtro desenvolvido no RIDE é apresentado, seus módulos e blocos funcionais são descritos e possíveis alterações nos parâmetros desses blocos são discutidas. As adaptações e otimizações realizadas no filtro durante o processo de desenvolvimento também são apresentadas. Estas alterações visaram, principalmente, diminuir o tempo de processamento no TMS320C31 e o uso da memória da TIGER 31/PC. A avaliação dos resultados de testes realizados é assunto do capítulo V.



IV.2 – Plataforma de Desenvolvimento e Operação

A escolha de uma plataforma de hardware e software para o desenvolvimento e execução de uma aplicação de processamento de sinais digitais em tempo-real é uma tarefa que requer uma avaliação do tipo de aplicação, das características do processador de sinais (DSP) que deverá executar a aplicação e de uma estimativa da quantidade de memória requerida pela aplicação. Por outro lado, obter tempo-real, nas condições desejadas, não é tarefa das mais fáceis, pois requer otimização de funções de processamento de sinais e constantes avaliações do processamento no DSP. É claro que o objetivo é sempre utilizar um DSP que seja mais barato e que consiga executar a aplicação desejada. Visando a implementação de um algoritmo de redução de ruído em sinais de voz para operação em tempo real, uma plataforma de hardware e software foi escolhida. O hardware é composto basicamente de uma placa de processamento de sinais TIGER 31/PC [12] acoplada a um microcomputador (PC), como mostra a ilustração da figura IV.1. A placa TIGER 31/PC escolhida possui 128 KB de SRAM e incorpora um processador de sinais digitais TMS320C31 [17] operando a 60 MHz.



Figura IV.1 – Hardware usado no projeto do filtro redutor de ruído

O software escolhido para o desenvolvimento e execução do algoritmo é o Hypersignal RIDE [11], que trabalha sobre o Windows, e é utilizado a partir dos consoles do PC. O RIDE possui controladores que realizam a interface entre o PC e o DSP.

Destaca-se que o algoritmo de redução de ruído em sinais de voz baseado na Subtração Espectral Modificada (SEM) já havia sido implementado anteriormente, durante o desenvolvimento deste trabalho, no software Matlab/Simulink, para operações em tempo não-real. Esta implementação possibilitou a definição de um diagrama de blocos funcionais para a SEM e, acima de tudo, uma familiarização com interfaces gráficas (Graphical User Interface - GUI), o que contribuiu muito para o desenvolvimento do filtro no RIDE, para operação em tempo real.

IV.2.1 – A placa TIGER 31/PC

A TIGER 31/PC é uma placa de processamento de sinais para sistemas PC ISA ou para operação autônoma. Baseada no processador TMS320C31, a TIGER 31/PC opera em até 60 MHz, para obter uma taxa de até 60 MFLOPS. Ela pode ser usada em aplicações multimídia, tais como modem, fax, telefonia, processamento de voz e aplicações de áudio com qualidade de CD. A placa TIGER 31/PC é compatível com sistemas 386, 486 e Pentium, com conexão direta em slot ISA de 16 bits.

IV.2.1.1 – Características do hardware

A TIGER 31/PC possui as seguintes características básicas de hardware:

- Possui um processador TMS320C31 operando a 40, 50 ou 60 MHz;
- Aceita, na própria placa, de 128 KB a 1 MB de SRAM (RAM estática);
- Interface para linha telefônica analógica, entrada analógica estéreo/microfone e saída analógica estéreo, com qualidade de CD (16 bits), operando a até 50 kHz;
- Conversor A/D-D/A estéreo, operando a até 50 kHz com resolução de 16 bits;
- Uma porta serial de alta velocidade, operando a até 15 Mbits/s;
- Interface de 32 bits para o barramento ISA, suporta DMA e interrupção bidirecional;
- EPROM (até 128 KB) e conector para fonte de alimentação, permitindo uma operação autônoma.



IV.2.1.2 – Arquitetura

A arquitetura da TIGER 31/PC é baseada nas capacidades do processador TMS320C31. Além disso, esta placa possui uma arquitetura de barramento, que permite interfaceamento com um PC e com outras unidades de E/S. O diagrama em blocos da TIGER 31/PC é mostrado na figura IV.2, onde se destacam o DSP, as unidades de memória SRAM, uma unidade de memória EPROM, o conversor A/D-D/A, um módulo de interfaceamento para linha telefônica externa, e uma porta serial RS-232. O DSP ilustrado no diagrama da figura IV.2 é o TMS320C31-60, que opera a 60 MHz. Uma foto da placa TIGER 31/PC pode ser visualizada na figura IV.3, indicando um processador TMS320C31 incorporado. Os principais componentes da TIGER 31/PC são descritos a seguir.

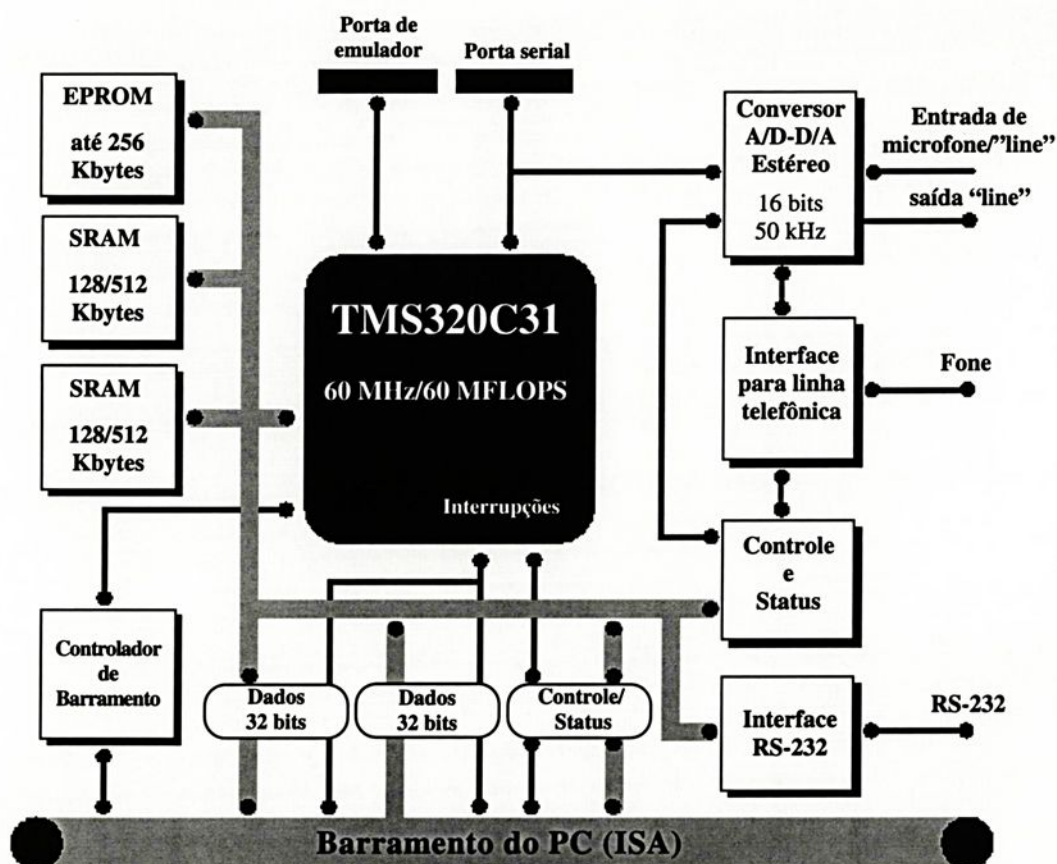


Figura IV.2 – Diagrama em blocos da TIGER 31/PC com o processador TMS320C31-60

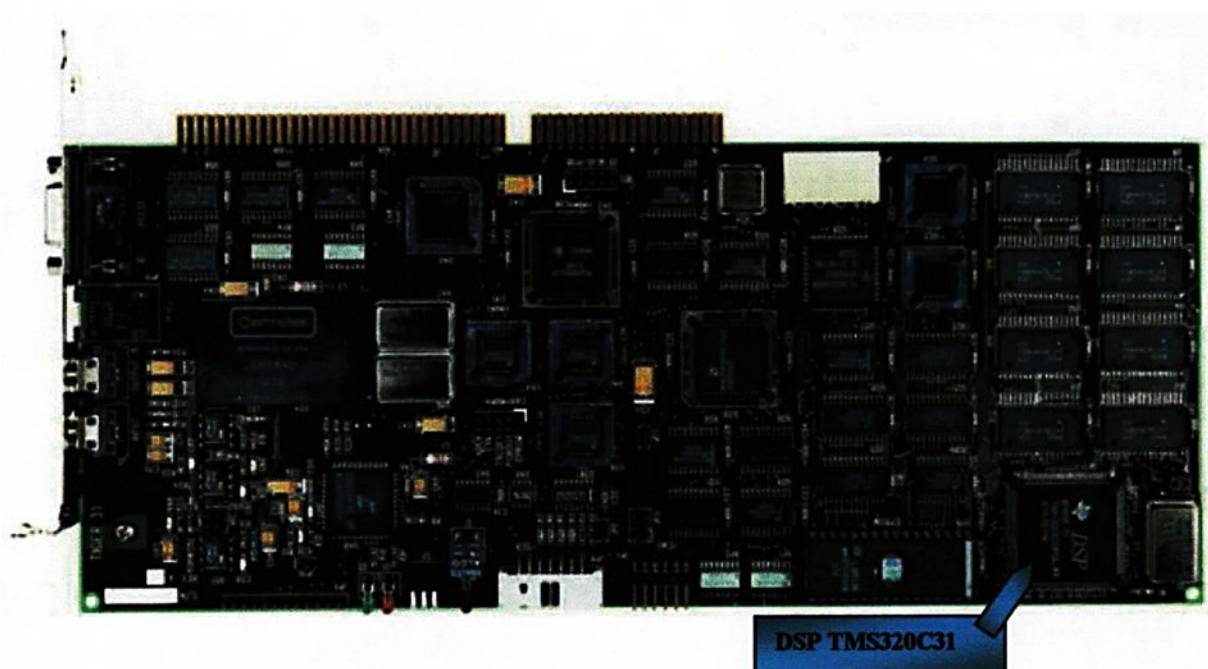


Figura IV.3 – Foto da placa TIGER 31/PC

Processador

TMS320C31 operando a 60 MHz (60 MFLOPS).

Memória

O TMS320C31 tem um barramento para acessar uma ou duas unidades de memória externa. Na placa TIGER 31/PC, cada uma destas unidades pode ter 128 Kbytes ou 512 Kbytes de RAM estática (SRAM). Na placa utilizada, tem-se 128 Kbytes.

Interface com o PC

A TIGER 31/PC possui um conector para o barramento PC/AT (ISA), com uma interface específica para a transferência de dados com o PC. A taxa de transferência máxima é de aproximadamente 2 Mbytes por segundo, podendo apresentar pequenas variações de PC para PC, dependendo da velocidade no barramento ISA.

As transferências podem ser realizadas sob o controle da CPU, por uma interrupção ou por DMA. Independentemente do modo de transferência de dados entre o PC e o DSP, o PC pode interromper a TIGER 31/PC em qualquer instante. Isto possibilita flexibilidade no projeto de software. Por exemplo, pode-se alterar o parâmetro de uma função do DSP, em pleno processamento no DSP.

A interface para o PC é projetada de tal forma que múltiplas placas TIGER 31/PC podem residir no barramento ISA para aplicações de multiprocessamento.

Entrada/Saída analógicas

A placa TIGER 31/PC tem um conversor analógico p/ digital e vice-versa (A/D-D/A) CS4216-KL da Cristal Semiconductor [12], que tem resolução de 16 bits (qualidade CD) e taxa de amostragem variável até 50 kHz. O sinal de entrada pode ter nível tipo microfone ou "line". A saída tem nível "line". Aplicações típicas incluem multimídia, reconhecimento e síntese de voz, análise espectral e processamento de sinal acústico. A placa também tem uma porta de entrada/saída serial (RS-232) que, quando usada, desabilita o conversor A/D.

Entrada/Saída serial

O TMS320C31 tem um canal serial de E/S para transferência de 8/16/32 bits em até 15 Mbps. O canal está disponível via um conector de 14 pinos para cabo tipo "fita". Quando este canal é utilizado, o conversor A/D-D/A é automaticamente desabilitado.

Conexão para linha telefônica

A TIGER 31/PC tem uma interface para linha telefônica analógica padrão, compatível com a rede telefônica brasileira. Esta interface usa um módulo CH1840 da Cermetek [12].

Suporte a emulador

A TIGER 31/PC tem um conector para suportar os emuladores externos *XDS-500/510* da Texas Instruments e o *TEM30* da DSP Research [12]. Um emulador é uma ferramenta de depuração externa que permite o controle direto sobre o processador TMS320C31 da TIGER



31/PC [16]. Contudo, como a TIGER 31/PC permite emulação própria, utilizar um outro emulador normalmente não é necessário. Adicionando-se o *TIGER Smart POD* e o seu software associado, a TIGER 31/PC pode ser atualizada como um emulador compatível com o *XDS-510*. Quando usada como um emulador, a TIGER 31/PC suporta o depurador *DB30* da Texas Instruments e o ambiente de desenvolvimento *Code Composer* da Go DSP.

IV.2.1.3 – Software

O suporte ao desenvolvimento de aplicativos para a TIGER 31/PC conta com as seguintes opções:

- Sistemas operacionais *QuickSTART* da TIGER e o *SPOX 2.x* da Spectron Microsystems;
- Compilador C, Assembler, Linker e Loader da Texas Instruments;
- Bibliotecas completas de apoio ao software para dispositivos na placa e para comunicações com um PC, disponíveis nas versões DOS e Windows, além de bibliotecas convencionais em C e das bibliotecas de funções da TIGER, denominada *PowerPack TIGER* [13].
- Depuração em nível de código C através de dois ambientes de depuração disponíveis: o *DB30* da Texas Instruments, em versão para DOS ou Windows, e o *GO DSP! Code Composer*, um ambiente integrado para desenvolvimento de código do DSP e depuração.
- Outros softwares de desenvolvimento disponíveis, incluindo o *Hypersignal RIDE* e o *DSP works*;
- Uma seleção de algoritmos de fax, modem e codificadores de voz são oferecidos, incluindo *G.728*, *CELP*, *ADPCM* e *VSELP*.

IV.2.2 – O software *Hypersignal RIDE*

O *Hypersignal RIDE* é um software para o desenvolvimento de aplicações para DSPs baseado em um ambiente gráfico para o sistema operacional Windows/NT. No desenvolvimento deste trabalho usou-se o RIDE (“Real-Time Integrated Development Environment”) em conjunto com a placa de processamento de sinais TIGER 31/PC.

O RIDE possui um conjunto de bibliotecas de funções usadas em processamento de sinais digitais e permite ao usuário criar suas próprias funções usando código fonte em C ANSI.

A principal motivação para o uso do *Hypersignal RIDE* é que este software permite o desenvolvimento de aplicações em ambiente gráfico, através da elaboração de diagrama de blocos, e permite operações com diferentes placas de processamento de sinais. Um projeto desenvolvido sobre a plataforma RIDE/PC-DSP pode ser convertido para uso na placa TIGER 31/PC operando sem o PC. Além do *Hypersignal RIDE* gerar diretamente um arquivo COFF para uso em outra plataforma usando o TMS320C31, há também uma opção onde o código fonte da aplicação em C ANSI pode ser obtido. Essa opção não existe no ambiente adquirido devido ao seu alto custo.

IV.2.2.1 – *Características básicas*

O *Hypersignal RIDE* é um programa capaz de simular algoritmos em um ambiente gráfico usando uma biblioteca de blocos funcionais (DLLs). Sua arquitetura permite a implementação rápida de algoritmos com base em estruturas conceituais e é composta de dois níveis de atuação: um é a interface do usuário (composta da área de trabalho e das barras de menus e ferramentas disponíveis ao usuário); o outro, é a biblioteca de blocos funcionais [10]. O software é baseado em *Dynamic Link Libraries* (DLLs), que podem ser consideradas como sub-rotinas ou blocos individuais. Esses blocos podem ser conectados entre si para formar a aplicação. Uma biblioteca padrão de blocos de simulação no PC e uma biblioteca padrão de blocos para execução no DSP estão disponíveis. Adicionalmente, novas funções podem ser desenvolvidas e incorporadas usando C ANSI e o aplicativo *Block Wizard* (parte integrante do RIDE).

Os principais pontos de destaque do RIDE são:

- Opera com 32 bits, incluindo todos os blocos de função (DLLs);



- O processamento em tempo real pode ser obtido através do uso de placa(s) com DSP(s) em conexão com o PC, com geração de um arquivo tipo COFF para aplicações autônomas;
- Possui o utilitário *Block Wizard*, que permite ao usuário criar novos blocos e DLLs (“Dynamic Link Library”). Se o usuário já tem o código fonte em C, ele pode ser adicionado a um bloco;
- Permite ao usuário elaborar blocos hierárquicos. Um bloco hierárquico contém dois ou mais blocos convencionais interligados, formando uma estrutura, com uma ou mais entradas/saídas;
- Suporte ao uso de caminhos de realimentação na estrutura de blocos;
- Compilação rápida do projeto (estrutura de blocos) para a simulação, permitindo ao usuário definir a ordem de compilação dos blocos;
- Uso de mais de um projeto na área de trabalho, visualizando-os lado a lado ou em cascata na tela;
- A seleção de blocos é feita através de uma janela de seleção de função (Seletor de Blocos), sendo possível acessar a área de trabalho enquanto o *Seletor de Blocos* pode ser acessado na tela, simultaneamente. Pode-se também apagar ou adicionar blocos definidos pelo usuário e listados no *Seletor de Blocos*;
- Possibilita a alteração de parâmetros dinâmicos de um bloco (a frequência de uma senóide de um bloco “Gerador de Onda”, por exemplo) durante a simulação do projeto, bem como observar o comportamento do sistema;
- Permite visualizar resultados graficamente na tela do PC, mesmo durante o processamento no DSP, através de blocos mostradores e de blocos que realizam a transferência de dados entre o DSP e o PC;
- Possui controladores de dispositivos para várias placas DSP.

Para dar uma idéia do ambiente de desenvolvimento Hypersignal RIDE, na figura IV.4 apresenta-se um exemplo com um projeto de anulação de eco (arquivo Echo.lst) na área de trabalho.



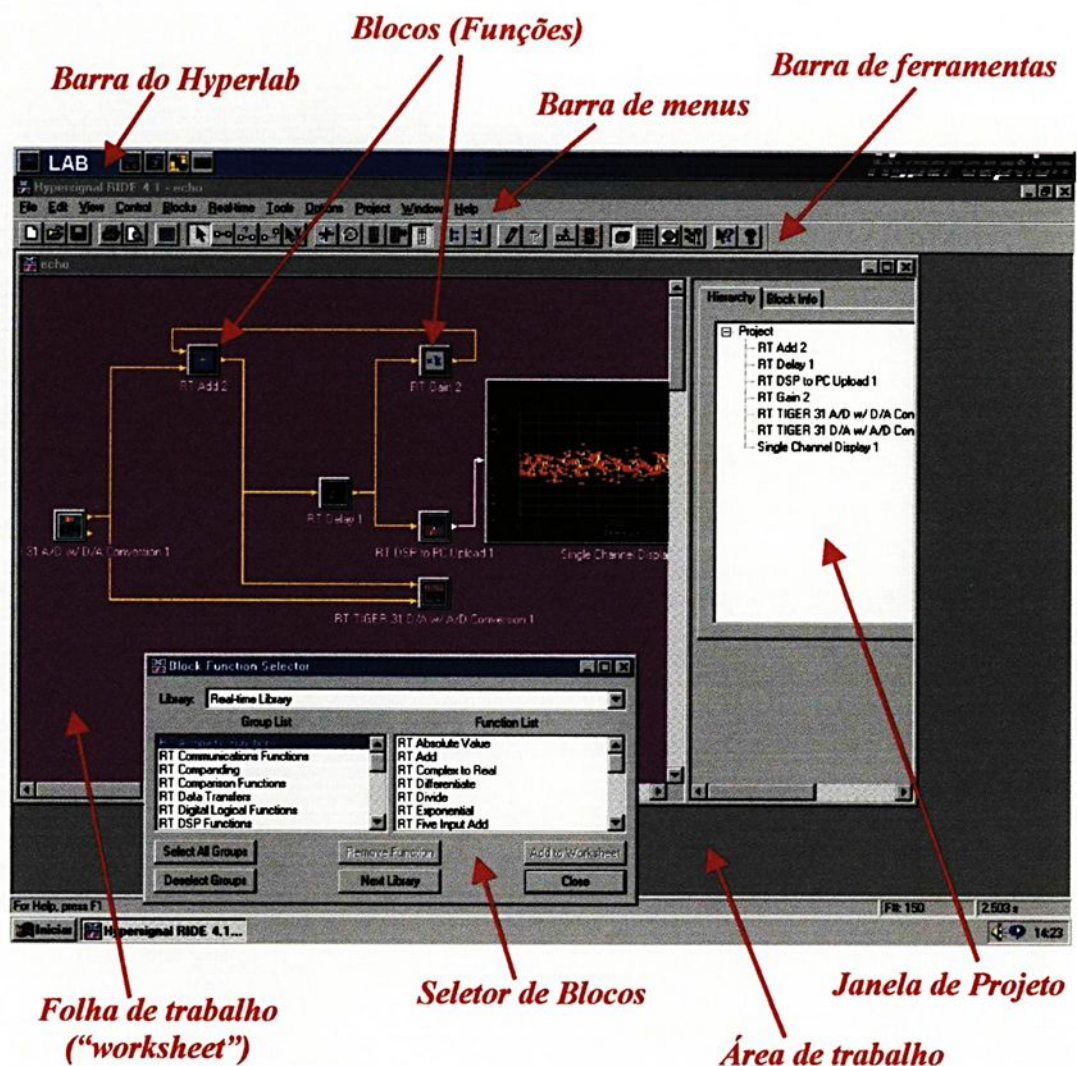


Figura IV.4 - Interface de usuário do RIDE com o projeto echo.lst aberto

IV.2.2.2 – Suporte às aplicações em tempo real

O RIDE permite o desenvolvimento completo de projetos para aplicações em tempo-real, com definições de mapa de memória, interrupções, entrada/saída, etc. A interação da plataforma RIDE com uma placa de processamento de sinais é feita por um controlador específico, denominado de “controlador de tempo-real”. Esse controlador realiza toda a interface entre o PC e a placa com DSP, fornecendo informações sobre fluxo de dados, viabilizando a conexão dos arquivos objetos, carregando código, dados e parâmetros para a memória do DSP, e monitorando todas as operações no DSP.

Algumas das características básicas e capacidades do RIDE com relação ao processamento de sinais usando DSPs, incluem:

- Suporte a multiprocessamento com o uso de vários DSPs;
- Suporte à criação de aplicações para DSPs virtuais;
- “Linker” de objeto integrado;
- Suporte a outras bibliotecas;
- Controle do mapa de memória;
- Suporte a interrupção;
- Suporte à sincronização;
- Suporte à “depuração”;
- Acesso a informações de alocação de memória;
- Análise relativa ao tempo de processamento de blocos individuais e/ou do algoritmo completo no DSP;
- Exportação de uma aplicação para arquivo COFF ou para a memória *flash*;
- Operações de memória;
- Importação de arquivos externos para uso com o DSP;
- Obtenção de estatísticas do sistema;
- Controle direto do DSP;
- Acesso a arquivos texto com informações sobre a aplicação e o processamento no DSP.



IV.3 – Desenvolvimento do Filtro no RIDE

O sistema de redução de ruído em sinais de voz implementado no RIDE é baseado na metodologia de implementação da *Subtração Espectral Modificada (SEM)*, ilustrada na figura II.5, e no algoritmo de *Deteção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS)*, mostrado na figura III.1. Este sistema dinâmico foi projetado para operação em tempo-real. Assim, todos os blocos funcionais usados na sua elaboração pertencem à biblioteca de tempo-real (RT) do RIDE que, no tempo de execução, serão carregados na placa TIGER 31/PC e executados pelo TMS320C31. O modelo implementado possui um diagrama principal com vários subsistemas, relativos às principais etapas do processamento. Cada subsistema representa um bloco hierárquico do RIDE, com entrada(s) e saída(s), contendo internamente blocos funcionais interligados para executar a tarefa para o qual foi programado. Os subsistemas desenvolvidos para uso no filtro redutor de ruído (incluindo os tamanhos de quadros de sinal utilizados) são:

- Transformação Tempo-Frequência (128/129/256/512 amostras)
- DIS e Estimação do Ruído (129 amostras)
- Cálculo da Função de Transferência do Filtro ($|H(w)|$) (129 amostras)
- Filtragem (129/256/512 amostras)
- Transformação Frequência-Tempo (128/256/512 amostras)

A modelagem do filtro redutor de ruído em sinais de voz realizada no RIDE é mostrada na figura IV.5.

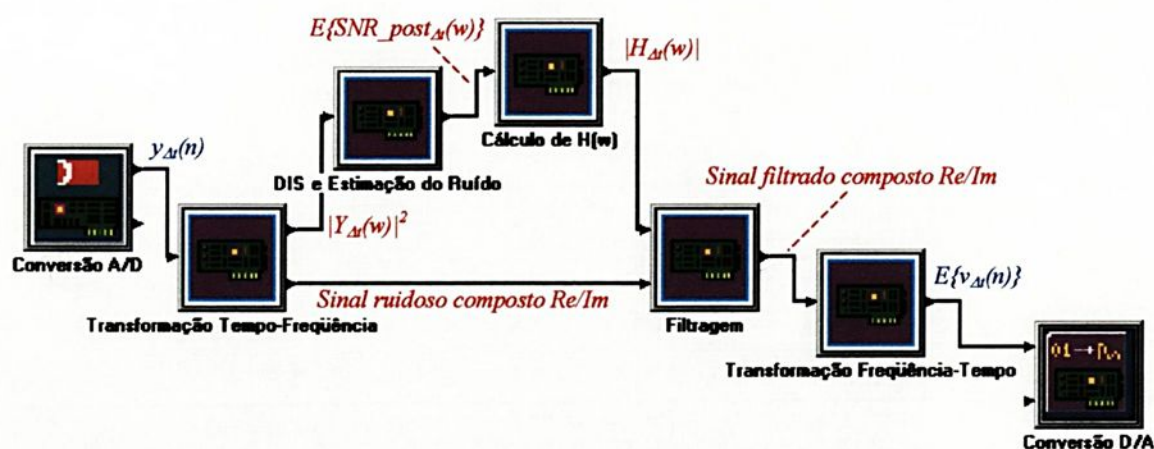


Figura IV.5 – Diagrama Principal do Filtro elaborado no RIDE

Os blocos conversores A/D e D/A mostrados na figura IV.5 indicam que a entrada do sinal ruidoso deve ser um sinal analógico fornecido na entrada “line” da placa TIGER 31/PC e que a saída do sinal filtrado é um sinal também analógico disponível na saída “line” da TIGER 31/PC.

No modelo da figura IV.5, inicialmente o sinal analógico de entrada (“line in”) da placa TIGER 31/PC é amostrado e digitalizado pelo bloco de conversão analógico para digital denominado “Conversão A/D”, com uma taxa de amostragem de 8 kHz. Definiu-se também um tamanho de quadro de saída do conversor A/D de 128 amostras. Porém, como foi aplicado um efeito de “overlap” de 50%, o quadro de análise é de 256 amostras (32 ms), suficiente para garantir a estacionariedade do sinal. O “overlap” é necessário para que o sinal possa ser janelado. O janelamento evita o surgimento de descontinuidades no sinal processado. Destaca-se que os blocos funcionais do algoritmo são executados no TMS320C31, onde o processamento é realizado, quadro a quadro, num “loop” infinito, a menos que ocorra uma interrupção a partir do PC. Os conversores A/D e D/A operam de modo síncrono e a taxa de amostragem pode ser alterada pelo usuário no bloco de conversão D/A, podendo variar de 7350 Hz a 44 kHz, com valores intermediários específicos. Porém, a quantidade e complexidade dos blocos envolvidos na aplicação determinaram uma frequência de amostragem máxima permitida para o processamento do filtro em tempo real. Analogamente, observa-se que a quantidade de memória de dados requerida na placa TIGER 31/PC é diretamente proporcional ao tamanho de quadro escolhido. Estes assuntos são discutidos no Capítulo V.

Após a realização da conversão A/D, o quadro de amostras digitais do sinal ruidoso ($y_{dt}(n)$) é transferido para o subsistema “Transformação Tempo-Frequência”. Este subsistema fornece, na saída, o espectro de potência janelado $|Y_{dt}(w)|^2$ e um sinal também janelado, composto de parte real (Re) e imaginária (Im) referentes à FFT do sinal ruidoso. A partir do espectro de potência deste sinal, realiza-se a DIS, estima-se o Ruído e calcula-se a Função de transferência do filtro.

No subsistema de filtragem, a função de transferência obtida é aplicada às janelas do sinal complexo da FFT. Este procedimento equivale a filtrar a magnitude do sinal ruidoso e manter a fase ruidosa.



Após a filtragem, o subsistema “Transformação Freqüência-Tempo” realiza a conversão inversa, do domínio da freqüência para o do tempo, utilizando-se um processo de síntese. A saída deste subsistema é um quadro do sinal de voz estimado $E\{v_{\Delta t}(n)\}$ contendo 128 amostras.

Finalmente, o conversor D/A converte os quadros processados fornecendo em sua saída o sinal de voz temporal filtrado.

Os subsistemas que compõem o diagrama da figura IV.5 são descritos nas próximas seções. Destaca-se que cada subsistema é composto de blocos interligados além das suas respectivas entrada(s) e saída(s).

IV.3.1 – Transformação Tempo-Freqüência

O subsistema de transformação do sinal ruidoso do domínio do tempo para o da freqüência, desenvolvido no RIDE, é mostrado na figura IV.6.

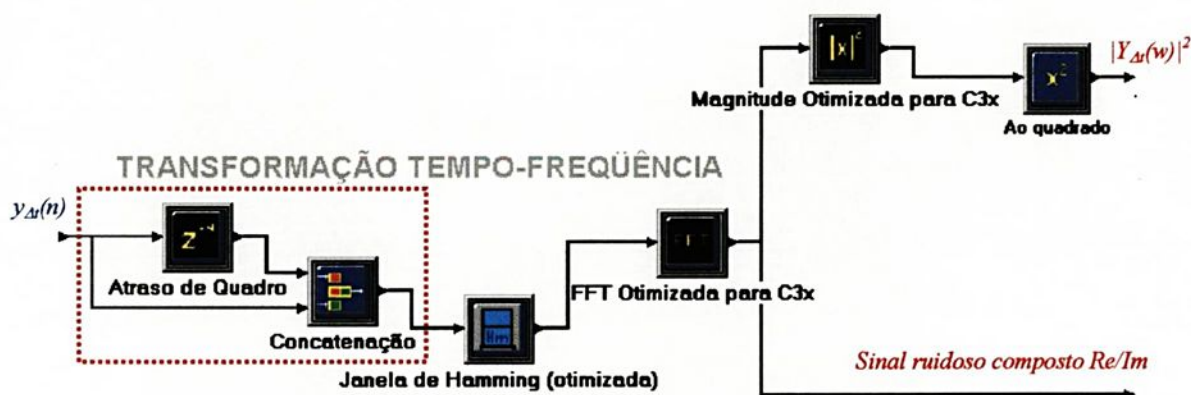


Figura IV.6 – Subsistema Transformação Tempo-Freqüência

Inicialmente aplica-se aos quadros do sinal temporal original de entrada $y_{\Delta t}(n)$ um efeito de “overlap” de 50%, e um janelamento, utilizando-se a janela de Hamming. O “overlap de 50%” é realizado pelos blocos contidos no retângulo tracejado (em vermelho) ilustrado na figura IV.6. As Figuras IV.7 e IV.8 ilustram, respectivamente, os efeitos “overlap” e

janelamento para um trecho do sinal. Como o objetivo foi mostrar o efeito do “overlap” e do janelamento, não foram consideradas as amplitudes das amostras.

Analisando a figura IV.7, a cada interação, tem-se um vetor de 256 amostras do sinal ruidoso na saída do bloco “Concatenação”, enquanto que o quadro de entrada do subsistema é de apenas 128 amostras. Neste exemplo, foram consideradas as primeiras 384 amostras do sinal na entrada do subsistema e as respectivas saídas do bloco “Concatenação” para o período analisado. Nota-se que, na saída deste bloco, as 128 últimas amostras do quadro anterior são as 128 primeiras amostras do quadro em análise. O porquê da utilização do efeito de “overlap” será entendido após definir-se o janelamento e a reconstrução do sinal empregando um processo inverso, denominado “overlap addition” e descrito no item IV.3.5.

O janelamento é realizado pelo bloco “janela de Hamming (Otimizada)” ilustrado no diagrama da figura IV.6. Para o sistema implementado escolheu-se a janela de Hamming, que evitará uma mudança brusca nos valores das amostras, na transição entre janelas, o que ocasionaria descontinuidades no sinal processado. O efeito do janelamento aplicado a um trecho do sinal pode ser observado no sinal janelado mostrado na figura IV.8.

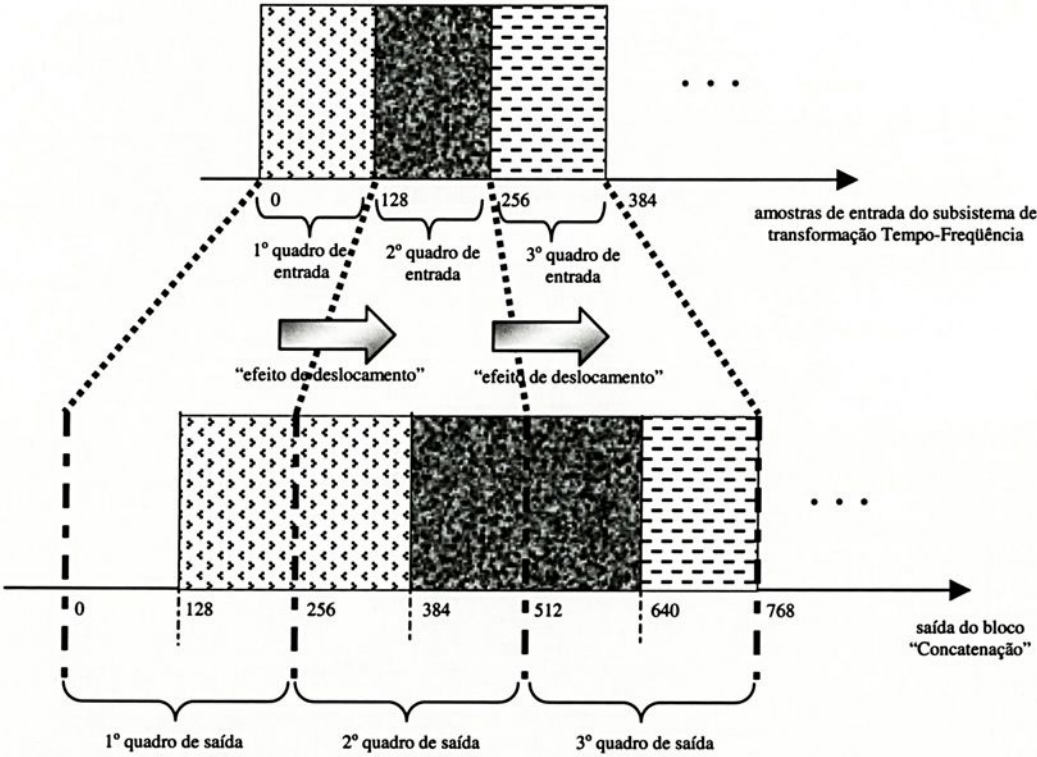


Figura IV.7 – Efeito “Overlap de 50%” para um trecho do sinal



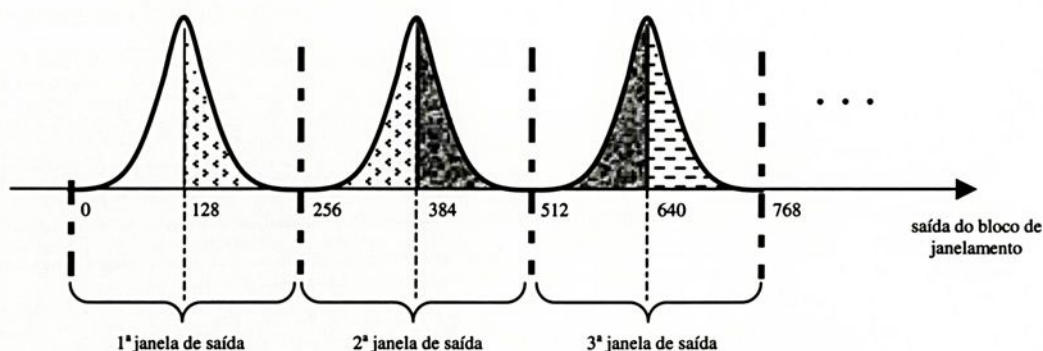


Figura IV.8 – Janelamento realizado para os quadros de saída do bloco “Concatenação” do exemplo da figura IV.7

Após o janelamento do sinal, aplica-se a FFT (otimizada para o TMS320C31) ao sinal janelado, usando 256 pontos. A saída da FFT é um sinal, em frequência, composto de parte Real (Re) e Imaginária (Im). Este sinal será utilizado posteriormente pelo subsistema de filtragem. A partir da FFT, calcula-se também o espectro de magnitude (otimizado para o TMS320C31) e o espectro de potência para o quadro analisado ($|Y_{dt}(w)|^2$). A potência do sinal ruidoso será usada como entrada no subsistema “DIS e Estimação do Ruído”. Destaca-se que, para os cálculos da FFT e do espectro de magnitude do sinal ruidoso, foram utilizados blocos otimizados para o TMS320C31 ao invés dos blocos convencionais do RIDE, o que significou uma redução significativa no tempo de processamento. Este ponto é importante porque estas funções consomem um tempo de processamento relativamente grande durante a realização da filtragem na TIGER 31/PC.

IV.3.2 – DIS e Estimação do Ruído

A Detecção de Intervalos de Silêncio (DIS) tem a função de identificar os trechos do sinal ruidoso onde somente o ruído está presente, ou seja, trechos onde não há voz. A estimação da potência do ruído é realizada a partir dos espectros de potência do sinal ruidoso obtidos nos intervalos de silêncio.

Neste trabalho, um novo método de DIS foi proposto e implementado, baseado na variância da relação sinal/ruído a posteriori estimada ($E\{SNR_{post_dt}(w)}\}$), e denominado *Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS)* (ver capítulo III).

Como a SNR_{post} é função da própria potência do ruído, a $DISS$ poderá atuar somente após uma estimação anterior da potência do ruído. O subsistema de DIS e Estimação do Ruído é mostrado na figura IV.9, onde os blocos contidos no retângulo tracejado (em vermelho) realizam a **DIS baseada na variância da SNR (DISS)**. A saída do bloco denominado “Multiplex 1” corresponde ao sinal que controla a atualização do ruído. Este sinal é identificado como a saída da $DISS$, e será igual a 1 (um) nos intervalos de silêncio detectados pela $DISS$ ou 0 (zero) caso contrário.

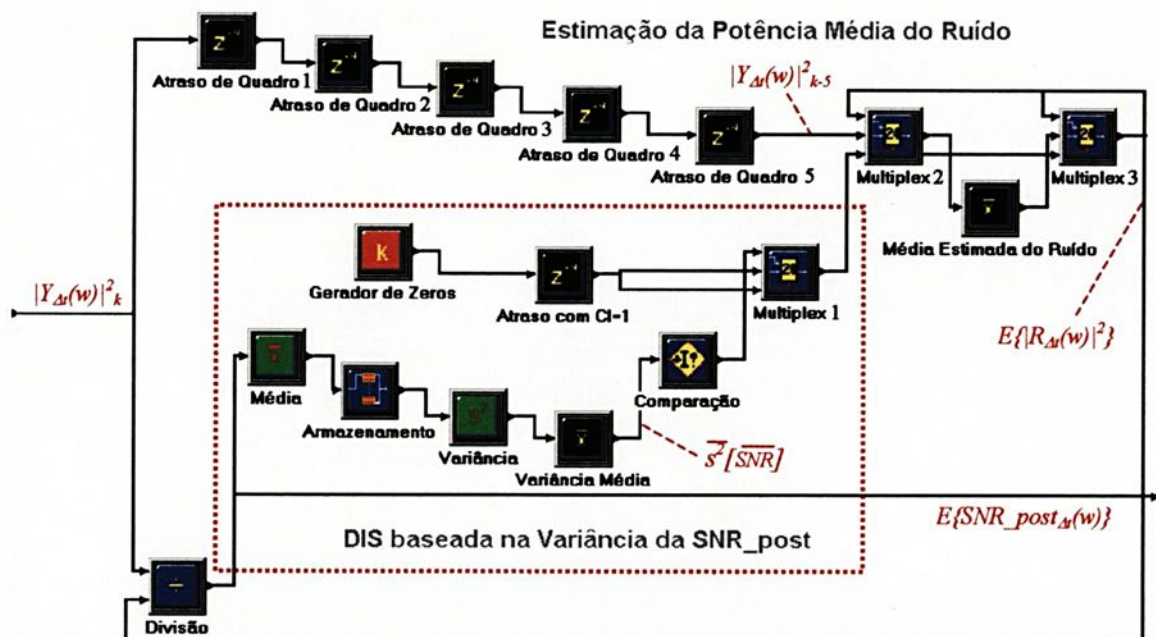


Figura IV.9 – Subsistema de DIS e Estimação da Potência do Ruído

A estimação da potência do ruído é realizada a partir dos espectros de potência do sinal ruidoso obtidos nos intervalos de silêncio, identificados a partir da $DISS$. Para realizar esta estimação, o subsistema da figura IV.9 foi esquematizado baseado nas seguintes informações:

- se o valor fornecido pela DIS for igual a 1 (um), isto significa a detecção de um intervalo de silêncio no trecho do sinal ruidoso em processamento;
- se o valor fornecido pela DIS for igual a 0 (zero), isto significa presença de voz no trecho em processamento.

Os blocos de estimação da potência do ruído calculam uma média recursiva da potência do ruído do sinal de entrada atrasado, de acordo com a equação II.17, onde o valor de α foi escolhido de maneira a dar um peso maior às janelas anteriores em relação à janela atual, evitando assim variações bruscas de potência de uma janela para outra. Para este projeto atribuiu-se a α o valor de 0,86. O valor de m foi escolhido de maneira a compensar o atraso provocado na detecção de intervalos de silêncio (DIS), devido ao método empregado, em que calcula-se uma “variância média” da SNR, a partir da variância das últimas janelas processadas e das variâncias calculadas anteriormente. De acordo com os valores dos parâmetros definidos para a DIS, atribuiu-se a m o valor de 5 janelas (80 ms). Este atraso é realizado pelos cinco blocos denominados “Atraso de Quadro”, ilustrados na figura IV.9, e corresponde ao atraso aplicado a potência do sinal ruidoso usada para a atualização da média do ruído. A atualização do ruído é realizada pelo bloco “Média Estimada do Ruído” que, juntamente com os blocos “Multiplex 2” e “Multiplex 3”, também mantém, durante um trecho de voz, a média calculada no último intervalo de silêncio identificado pela *DISS*.

Os blocos denominados “Multiplex 2” e “Multiplex 3” funcionam como chaves seletoras, fazendo com que o ruído seja atualizado somente se a saída da DIS for igual a 1 (um), ou seja, eles fornecem o espectro de potência do sinal ruidoso ao bloco “Média Estimada do Ruído” somente nos intervalos de silêncio, onde apenas o ruído aditivo estará presente. Caso contrário, eles manterão na saída a última média calculada. Desta maneira, no processamento de trechos de voz, o filtro manterá na sua saída a atualização feita no processamento da última janela de intervalo de silêncio detectada pela DIS. Uma nova atualização do ruído só será realizada quando o próximo intervalo de silêncio for detectado e a DIS informar aos blocos “Multiplex 2” e “Multiplex 3” esta mudança.

Analisando os blocos que compõem a *DISS*, a estimação inicial da potência do ruído é garantida pelos blocos “Gerador de zeros”, “Atraso com CI=1” e “Multiplex 1”. Estes blocos têm a função de fornecer, na saída do “Multiplex 1”, valores iguais a 1 durante as primeiras janelas processadas. A partir daí, a SNR_{post} servirá de base para uma DIS automática. Para esta atualização inicial da potência estimada do ruído, considera-se que as dez primeiras janelas processadas são ruído (160 ms). A definição deste valor é feita no bloco “Atraso com CI=1”. No caso em que este trecho inicial seja voz, o erro na estimação do ruído será corrigido no próximo intervalo de silêncio, tornando o sistema estável, como mostrou o desenvolvimento realizado anteriormente nas páginas 36 e 37 (seção III.2).



A *DISS* calcula uma variância média a partir do valor estimado da *SNR*_{post}, usando a metodologia ilustrada na figura III.1 do capítulo anterior. Inicialmente o bloco “Média” calcula a média dos espectros da *SNR*_{post} da janela em análise. O bloco “Armazenamento” foi configurado para armazenar as quatro últimas médias calculadas para a *SNR*_{post}. Este valor corresponde ao valor definido para o parâmetro *n* da *DISS*, descrito no Capítulo III. As médias armazenadas são utilizadas para o cálculo da variância da *SNR*_{post}. Portanto, a variância é calculada a cada janela processada, considerando as quatro últimas médias da *SNR*_{post} armazenadas. Posteriormente, um sistema recursivo representado pelo bloco “Variância Média” calcula $\overline{s^2[SNR]}$, considerando as variâncias calculadas anteriormente e a variância atual. O bloco “Variância Média” realiza a equação III.1 do capítulo anterior onde, para este projeto, atribuiu-se a γ o valor de 0,8, que foi escolhido de maneira a dar um peso maior à “variância média” calculada anteriormente em relação à “variância média” atual. Isto evita que uma eventual variação brusca na variância calculada de um trecho para outro subsequente, num período de fala, possa resultar num erro na DIS. Finalmente, a “variância média” calculada ($\overline{s^2[SNR]}$) é comparada a um limiar de decisão fixo ξ , que foi definido no bloco “Comparação” como $\xi=0,1$. O resultado da comparação define a DIS, como segue:

- Se a “variância média” for menor do que 0,1 então a saída do bloco “Comparação” será igual a 1, indicando que o trecho em análise é ruído;
- Caso contrário, a saída do bloco “Comparação” será igual a 0, indicando presença de voz.

O atraso de cinco janelas ($m=5$), aplicado às janelas da potência do sinal ruidoso utilizadas na atualização do ruído garante que, na transição silêncio/voz, um atraso na *DISS* não provocará uma estimação do ruído com trechos de voz, o que comprometeria a estabilidade do sistema, pois a *DISS* depende do valor estimado da *SNR* do sinal ruidoso, que é função do próprio ruído estimado.

Na execução do programa na TIGER 31/PC, as dez primeiras janelas processadas são consideradas como ruído, e a partir daí, a saída do bloco “Comparação” é transferida para a saída do bloco “Multiplex 1”, isto é, a DIS passa a ser automática, dependendo somente dos valores da “variância média” da $\overline{SNR}$.



IV.3.3 – Cálculo da Função de Transferência do Filtro ($|H(w)|$)

A função de transferência do filtro usando a SEM depende somente do valor estimado da relação sinal/ruído a priori (SNR_prio). O subsistema de cálculo da função de transferência deste filtro é mostrado na figura IV.10, onde os blocos contidos no retângulo tracejado (em vermelho) realizam a *Estimação da SNR_prio* , com base na equação II.19.

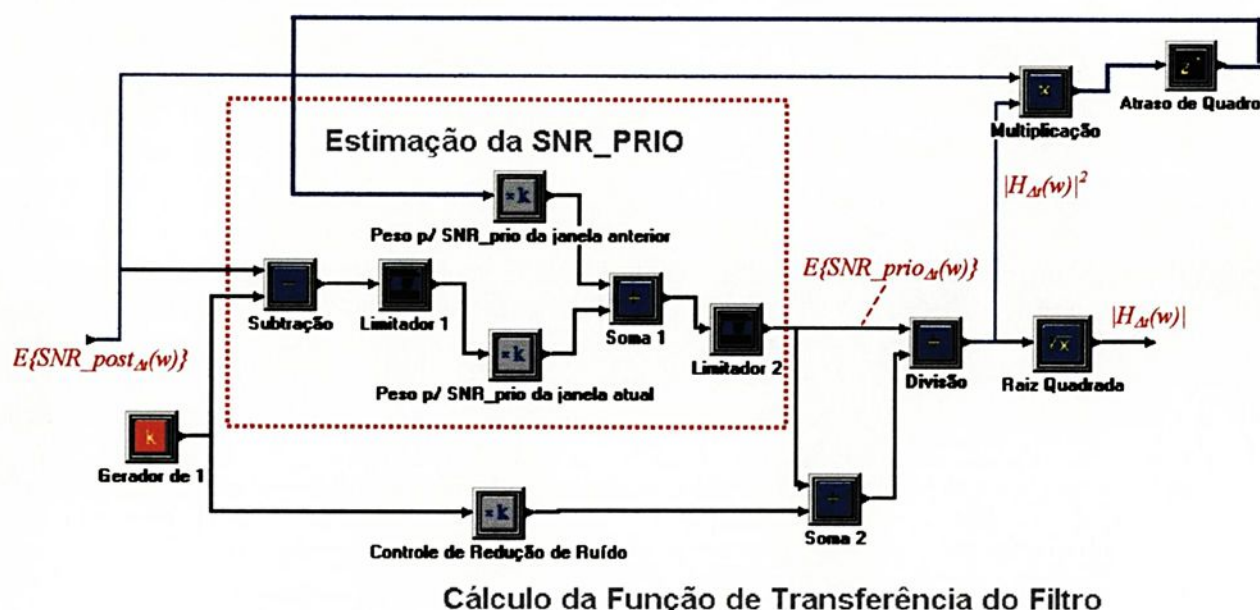


Figura IV.10 – Subsistema de Cálculo da Função de Transferência do Filtro ($|H(w)|$)

A estimação da relação sinal/ruído a priori (SNR_prio) é o que diferencia este filtro redutor de ruído de outro baseado somente na SE Clássica. A essência da Subtração Espectral Modificada (SEM) está na correta estimação deste parâmetro.

A estimação da SNR_prio utiliza, de acordo com a equação II.19, o valor estimado da potência da voz filtrada da janela anterior, a potência estimada do ruído e a relação sinal ruído a posteriori estimada ($E\{SNR_post\}$) da janela atual como parâmetros de entrada. A estimação da SNR_prio é realizada a cada janela processada. Desta maneira, a estimação é obtida calculando-se uma média ponderada entre a SNR_prio da janela anterior (obtida a partir do valor estimado da potência do sinal de voz filtrado da janela anterior) e a SNR_prio atual (obtida a partir do valor estimado da SNR_post da janela atual).

Visando otimizar a implementação em tempo real, verifica-se que é possível modificar a equação II.19 reduzindo significativamente o tempo de processamento. A alteração proposta consiste em escrever a equação II.19 somente em função da SNR_post, que é o parâmetro de entrada deste subsistema. Assim, tem-se:

$$\frac{E\{|V_{\Delta}(w)|^2\}_{k-1}}{E\{|R_{\Delta}(w)|^2\}_k} = \frac{\{|H_{\Delta}(w)|^2 \times |Y_{\Delta}(w)|^2\}_{k-1}}{E\{|R_{\Delta}(w)|^2\}_k} = \{ |H_{\Delta}(w)|^2 \times E\{SNR_post_{\Delta}(w)\} \}_{k-1} \quad (IV.1)$$

Substituindo a equação IV.1 em II.19, obtém-se:

$$E\{SNR_prio_{\Delta}(w)\}_k = \beta \cdot \{ |H_{\Delta}(w)|^2 \cdot SNR_post_{\Delta}(w) \}_{k-1} + (1-\beta) \cdot [E\{SNR_post_{\Delta}(w)\}_k - 1] \quad (IV.2)$$

onde k representa o índice de interação, incrementado a cada janela analisada.

No subsistema de cálculo da função de transferência do filtro usando a SEM, ilustrado no diagrama da figura IV.10, foi implementada a equação IV.2, para a estimação da SNR_prio. A mudança proposta na equação IV.1 está indicada com setas representando o fluxo de dados entre blocos, destacadas em cor azul, e é realizada pelos blocos “Multiplicação” e “Atraso de Quadro”. Essa alteração, além de reduzir o tempo de processamento, minimiza os erros de arredondamento, eliminando o cálculo de uma raiz quadrada, uma potenciação e uma divisão.

A variável β , indicada na equação IV.2, define os pesos aplicados aos valores calculados da SNR_prio da janela anterior e da SNR_prio da janela atual. Considerando que uma maior contribuição da SNR_post poderia afetar o desempenho na estimação da SNR_prio, gerando o ruído musical, atribui-se um peso bem menor para a parcela da SNR_prio da janela atual, adotando $\beta=0,99$. Esta atribuição significa configurar o ganho do bloco “Peso para SNR_prio da janela anterior” para 0,99 e o ganho do bloco “Peso para SNR_prio da janela atual” para 0,01. Estes blocos estão ilustrados no diagrama da figura IV.10.

Nos blocos limitadores do diagrama da figura IV.10, pode-se definir tanto um limite inferior quanto um limite superior para o sinal de saída. No subsistema de cálculo da função de transferência do filtro, a parcela da SNR_prio, calculada a partir da SNR_post, foi limitada a um valor mínimo de 0,1, representando o valor de ε definido na equação II.19. Esta limitação tem como objetivo minimizar o ruído musical, como foi descrito no item II.4.5. Além disso, o

valor estimado da SNR_{prio} foi limitado a valores entre 0,05 e 100, evitando desta maneira uma forte atenuação do sinal, minimizando distorções na voz processada.

O cálculo da função de transferência do filtro usando a *SEM* realiza a equação II.21, onde o parâmetro ρ é definido no bloco “Controle de Redução de Ruído” do subsistema da figura IV.10. Inicialmente, o fator $SNR_{prio} + \rho$ (presente na equação II.21) é calculado pelos blocos “Gerador de 1”, “Controle de Redução de Ruído” e “Soma 2”. Para as condições normais de operação do filtro atribui-se a ρ o valor de 1, o que significa o uso da equação característica da *SEM*, apresentada na equação II.15. No entanto, este parâmetro, que funciona como um fator de superestimação/subestimação do ruído, pode ser alterado para um valor que melhor se adapte a aplicação para a qual o filtro será destinado. Posteriormente, o processamento efetuado pelos blocos “Divisão 2” e “Raiz Quadrada” definem a função de transferência do filtro ($|H(w)|$) que é utilizada pelo subsistema de filtragem.

IV.3.4 – Filtragem

A filtragem consiste em aplicar a função de transferência do filtro ao espectro de magnitude do sinal, mantendo a fase ruidosa, como foi discutido na seção II.4.6. Porém, uma adaptação foi realizada, visando utilizar os blocos otimizados para o TMS320C31 que calculam a FFT, a magnitude do sinal e a IFFT. Destaca-se que foi necessária a utilização destes blocos, pois o tempo de processamento destas funções é relativamente grande se comparado a outras funções do algoritmo implementado. Porém, estes blocos otimizados operam sinais compostos, constituídos de parte real (Re) e imaginária (Im). A partir deste conceito, a separação dos espectros de magnitude e fase para aplicação da função de transferência somente ao espectro de magnitude, consumiria um tempo muito grande, pois, posteriormente, para o cálculo da IFFT otimizada, seria necessário reconstruir um sinal filtrado composto, que serviria de entrada para esta função. No entanto, prova-se matematicamente que estes cálculos são equivalentes a aplicar a função de transferência às partes real e imaginária do sinal composto proveniente da FFT e, posteriormente, realizar a transformação do domínio do tempo para o da frequência, para a recuperação do sinal temporal filtrado. O seguinte desenvolvimento ilustra este fato.

A magnitude e a fase de um sinal ruidoso complexo Y qualquer podem ser equacionadas, respectivamente, como segue:



$$Magnitude_{ruidosa} = \sqrt{[\text{Re}(Y)]^2 + [\text{Im}(Y)]^2} \quad e \quad fase_{ruidosa} = \arctg \frac{\text{Im}(Y)}{\text{Re}(Y)} \quad (\text{IV.3})$$

Aplicando-se uma função de transferência H às partes real e imaginária do sinal complexo Y , com o objetivo de realizar uma filtragem, têm-se:

$$Magnitude_{filtrada} = H \times \sqrt{[\text{Re}(Y)]^2 + [\text{Im}(Y)]^2} = H \times Magnitude_{ruidosa} \quad (\text{IV.4})$$

$$fase_{filtrada} = \arctg \frac{H \times \text{Im}(Y)}{H \times \text{Re}(Y)} = \arctg \frac{\text{Im}(Y)}{\text{Re}(Y)} = fase_{ruidosa}$$

O desenvolvimento mostrado nas equações IV.3 e IV.4 comprova que a aplicação da função de transferência do filtro às partes real e imaginária do sinal composto proveniente da FFT realizam a filtragem proposta pela *Subtração Espectral Modificada (SEM)*. Aplicando-se este conceito, o subsistema que realiza a filtragem foi elaborado como mostra a figura IV.11.

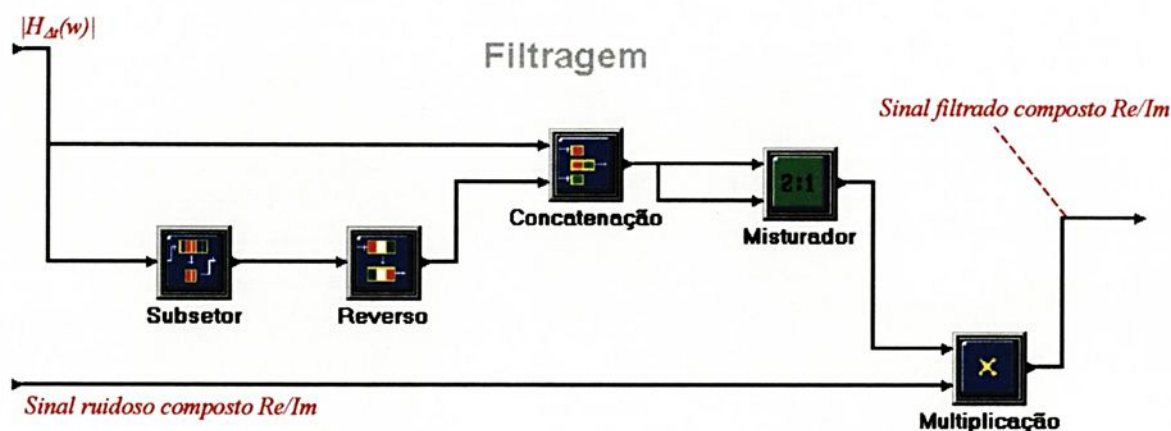


Figura IV.11 – Subsistema de Filtragem

No subsistema ilustrado na figura IV.11, a função de transferência de entrada, contendo 129 amostras, é preparada convenientemente para que possa ser multiplicada pelo sinal composto de parte real e imaginária proveniente da FFT. Este sinal tem 512 amostras. A preparação da função de transferência, realizada pelos blocos “Subsetor”, “Reverso”, “Concatenação” e “Misturador”, é baseada na maneira como o sinal proveniente do bloco de FFT é composto

(intercalando as partes real e imaginária, amostra a amostra) e na simetria das partes real e imaginária de um espectro complexo da FFT. Após o ajuste da função de transferência para um vetor com 512 amostras, este é multiplicado pelo sinal ruidoso composto, proveniente da FFT otimizada (subsistema Transformação Tempo-Frequência). O resultado da multiplicação é um sinal filtrado composto que será usado pelo bloco de IFFT no subsistema “Transformação Frequência-Tempo”.

IV.3.5 – Transformação Frequência-Tempo

O subsistema “Transformação Frequência-Tempo” tem a função de recuperar no tempo o sinal filtrado e para isso, após o cálculo da IFFT, usa-se um processo de síntese, denominado “overlap addition”. Este processo corresponde ao inverso do que foi realizado pelo subsistema Transformação Tempo-Frequência, descrito anteriormente no item IV.3.1. O subsistema “Transformação Frequência-Tempo” elaborado no RIDE é mostrado na figura IV.12.

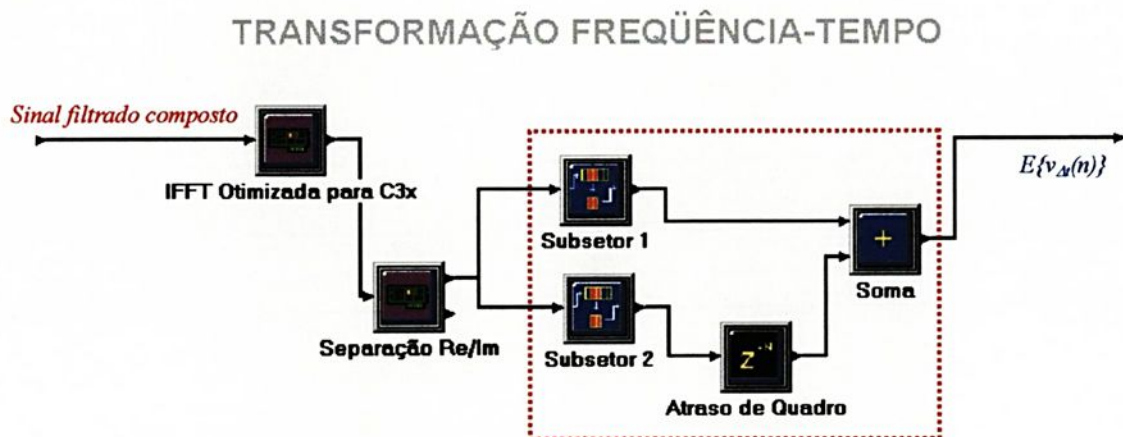


Figura IV.12 – Subsistema Transformação Frequência-Tempo

Na transformação do sinal filtrado do domínio da frequência para o do tempo, ilustrada na figura IV.12, o sinal filtrado composto proveniente do subsistema de filtragem serve de entrada para o cálculo da Transformada Rápida de Fourier Inversa (bloco “IFFT Otimizada para C3x”), que recupera as janelas do sinal temporal filtrado. Para a reconstrução dos quadros do sinal filtrado é necessário efetuar um processo inverso ao “overlap” realizado

anteriormente no subsistema “Transformação Tempo-Freqüência”. Este processo inverso, denominado “Overlap Addition”, realizado pelos blocos contidos no retângulo tracejado (em vermelho) ilustrado na figura IV.12, é aplicado à parte real do sinal temporal complexo de saída da IFFT otimizada. O “Overlap Addition” recupera os quadros do sinal de voz filtrado e corresponde a sobreposição e soma de amostras, como ilustra o exemplo da figura IV.13.

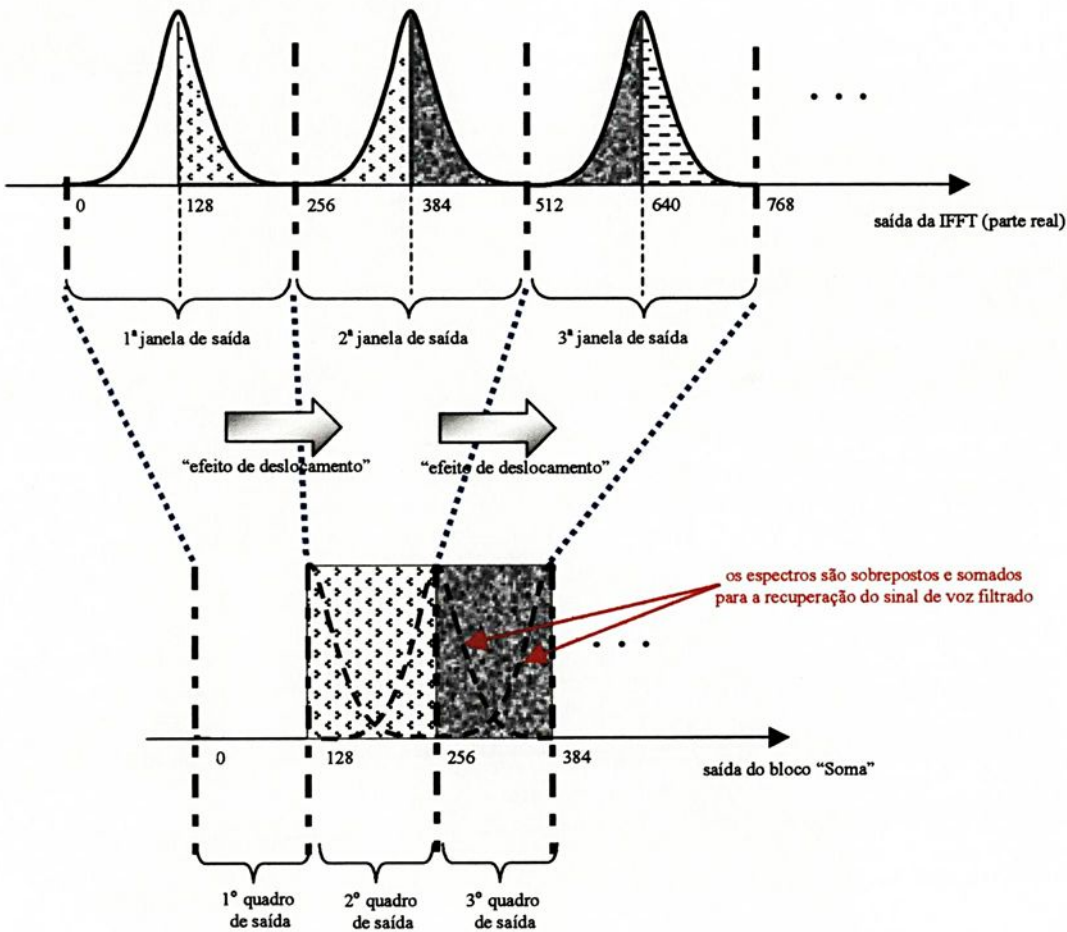


Figura IV.13 – Efeito do “overlap addition” aplicado às três primeiras janelas de saída do bloco “Separação Re/Im”

Destaca-se que, na transformação do sinal, do domínio do tempo para a freqüência, foram tomados quadros do sinal de entrada contendo 128 amostras, aplicou-se um efeito de “overlap” de 50% a cada quadro de entrada, resultando em quadros de 256 amostras e, posteriormente, aplicou-se o janelamento a estes quadros, usando-se a janela de Hamming.

A figura IV.13 ilustrou o efeito de “overlap addition” aplicado as 768 amostras do sinal janelado, correspondente a 384 amostras do sinal temporal. Também, como nos exemplos das figuras IV.7 e IV.8, não foram considerados os valores das amostras, pois o objetivo foi exemplificar somente o efeito do “Overlap Addition” na recuperação do sinal temporal filtrado.

IV.4 – Conclusões

Neste capítulo descreveu-se a implementação, para operação em tempo real, do algoritmo de redução de ruído em sinais de voz usando a *Subtração Espectral Modificada (SEM)* e um novo método de detecção de ruído em sinais de voz, baseado na variância da relação sinal/ruído e denominado *Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS)*. A plataforma usada no desenvolvimento e execução deste filtro redutor de ruído, composta basicamente de uma placa TIGER 31/PC (acoplada a um PC) e do software de desenvolvimento Hypersignal RIDE, foi apropriada para este tipo de aplicação. Destaca-se que o filtro implementado foi processado com sucesso pelo TMS320C31, operando em tempo-real.

A TIGER 31/PC é um hardware apropriado para o desenvolvimento de uma aplicação em tempo-real envolvendo sinais de voz, pois ela possui um processador ideal para este tipo de aplicação, além das suas portas de E/S analógicas e da interface para linha telefônica externa. A TIGER 31/PC utilizada contém 128 Kbytes de SRAM e um processador de sinais digitais TMS320C31 que opera a 60 MHz. O filtro foi configurado para utilizar a entrada e a saída analógica da TIGER 31/PC.

O Hypersignal RIDE foi o software de desenvolvimento adotado para o projeto do filtro proposto. O RIDE foi configurado, neste projeto específico, para ser utilizado com a placa TIGER 31/PC. O RIDE é executado a partir do PC, em ambiente Windows, e possui uma interface de usuário que permitiu o desenvolvimento do filtro em ambiente gráfico, através da elaboração de um diagrama de blocos funcionais. O RIDE possui várias funções de processamento de sinais específicas para o TMS320C31. Muitas destas funções foram utilizadas no filtro projetado.

O algoritmo de redução de ruído implementado no RIDE pode ter a maioria de seus parâmetros alterados pelo usuário, inclusive durante a execução no TMS320C31.



O RIDE é um programa que gera uma quantidade de código de programa e dados minimizada. Isto é importante, pois corresponde a memória requerida na placa TIGER 31/PC utilizada. Além disso, o filtro implementado também pode ser transportado para outras plataformas, em formato COFF, gerado pelo próprio RIDE.

As várias ferramentas disponíveis no RIDE, as suas capacidades com relação a operações de processamento de sinais em tempo real e a praticidade de uso, principalmente no desenvolvimento de aplicações, justificam o seu uso na implementação do filtro proposto.

Na elaboração do modelo do filtro redutor de ruído no RIDE, procurou-se utilizar um número mínimo de blocos de processamento e, quando possível, blocos otimizados, com o objetivo de minimizar o tempo de processamento e o uso da memória.

Para facilitar a visualização, análise e alteração de parâmetros, criaram-se subsistemas, usando o próprio RIDE, com as principais funções do filtro de redução de ruído proposto. Estes subsistemas, incluindo os blocos de conversão A/D e D/A, foram interligados, formando o modelo de redução de ruído em sinais de voz desenvolvido. Vale observar que, na execução do filtro, como se trabalha com o sinal janelado, o processamento é feito janela a janela, ou seja, cada ciclo de processamento corresponde a uma janela de sinal processada.

Os dados de operação do filtro implementado incluem: taxa de amostragem de 8 kHz, tamanho de quadro de entrada e saída de 128 amostras, quadro de análise de 256 amostras (após "overlap" de 50%) e FFT/IFFT de 256 pontos. Os subsistemas "*DIS e Estimação do Ruído*" e "*Cálculo de $|H(w)|$* " operam com 129 amostras. Estes dados, além do número e complexidade das funções (blocos) usadas no projeto do filtro, influenciam diretamente na memória requerida e na capacidade de processamento no TMS320C31.



CAPÍTULO V

RESULTADOS OBTIDOS E AVALIAÇÕES

V.1 – Introdução

O filtro redutor de ruído baseado na *SEM* e na *DISS* implementado no RIDE para operação em tempo-real, foi avaliado através de resultados obtidos no próprio RIDE, durante a execução do programa e após o processamento. A avaliação final foi feita em dois pontos distintos: 1°. desempenho do sistema em tempo real e 2°. qualidade do processamento. No primeiro caso, verificou-se principalmente a memória requerida para o processamento do filtro na TIGER 31/PC e o tempo de processamento dos blocos e subsistemas no TMS320C31. As medidas realizadas para a obtenção destes resultados de análise foram utilizadas para avaliar a capacidade do sistema, como por exemplo, a frequência de amostragem máxima com que o sistema pode operar em tempo real, sem distorcer o sinal de voz original. No segundo caso, foram analisados o nível de redução de ruído atingido na filtragem e as distorções resultantes do processamento. Foram analisados também a inteligibilidade e o conforto de audição do sinal filtrado, quando comparado ao sinal ruidoso original, associando-se a inteligibilidade à distorção e conforto auditivo à redução de ruído [9].

Neste capítulo, inicialmente apresentam-se as estatísticas do DSP para o filtro implementado no RIDE, que inclui memória utilizada e tempo de processamento.

Para uma análise geral do resultado da filtragem são apresentados e comparados gráficos contendo trechos de sinais de voz, incluindo voz original (sem ruído), voz ruidosa e o sinal de voz filtrado obtido após o processamento. Posteriormente, para avaliação do nível de redução de ruído atingido na filtragem, plotam-se os gráficos da relação sinal/ruído para o sinal ruidoso e para o sinal filtrado, e as reduções de ruído obtidas após a filtragem. Para avaliação da distorção provocada no sinal após o processamento, foram traçados e analisados os espectrogramas para os sinais de voz original, ruidoso e filtrado. Uma análise informal também foi realizada através de testes subjetivos.



V.2 – Estatísticas do filtro implementado no RIDE

O filtro redutor de ruído implementado no RIDE é composto de 49 blocos funcionais específicos para execução em tempo-real no TMS320C31, sendo que 28 destes são blocos exclusivos. O número de blocos exclusivos é um fator importante, pois está diretamente relacionado com a área de memória de programa ocupada na placa TIGER 31/PC. No RIDE, o código de um bloco e seus parâmetros são carregados na memória do DSP, em áreas distintas. Como um bloco do RIDE é similar a uma sub-rotina, somente o código dos blocos exclusivos é carregado, o que economiza memória. Em outras palavras, quando um tipo de bloco aparece mais do que uma vez no algoritmo, o código deste bloco é armazenado somente uma vez na memória. Com relação aos parâmetros definidos para as várias cópias deste bloco, estes sim, são independentes e armazenados em outras áreas de memória do TMS320C31.

As seções seguintes mostram a memória disponível e utilizada pelo filtro na placa TIGER 31/PC, bem como o tempo de processamento no TMS320C31. Todos os resultados obtidos são baseados no processamento em tempo-real na plataforma utilizada, considerando as características normais de operação do filtro, que são: quadro de entrada e saída de 128 amostras, quadro de análise de 256 amostras (overlap de 50%), FFT/IFFT de 256 pontos e frequência de amostragem de 8 kHz.

V.2.1 – Memória

O sistema de processamento de sinais utilizado para o armazenamento e execução do filtro é composto de uma placa TIGER 31/PC contendo 128 Kbytes de memória RAM, que incorpora um processador TMS320C31. Este DSP possui, internamente, 8 Kbytes de memória RAM. Portanto, o sistema disponibiliza 136 Kbytes para serem usados para programa e dados. Desta memória disponível, são reservados 32 Kbytes de memória para programa e 104 Kbytes para dados. O filtro desenvolvido utiliza 28 Kbytes para programa, correspondente a 87,7% da área total disponível para programa, e 42 Kbytes para dados, correspondente a 39,9% da área total disponível para dados. A tabela V.1 mostra a memória disponível e a memória utilizada, considerando vários segmentos contínuos de memória, denominados seções.



ÁREA ACESSADA	ENDEREÇO (hex)	CONTEÚDO	SEÇÕES	TOTAL DISPONÍVEL (em bytes)	TOTAL USADO (em bytes)
	0 0FFF	ÁREA NÃO USADA			
MEMÓRIA EXTERNA AO TMS320C31, LOCALIZADA NA PLACA (128 KB)	1000 2FBF	CODE	.text	32512	28516 (87,71%)
	2FC0 31BF	DATA	.bss	2048	524 (25,59%)
	31C0 33BF		.cinit	2048	388 (18,95%)
	33C0 34BF		.const	1024	0
	34C0 5DBF	HEAP	.system	41984	4096 (9,76%)
	5DC0 5FBF	DATA	.data	2048	0
	5FC0 8FFF		.hyper	49408	37064 (75,02%)
	9000 809907	ÁREA NÃO USADA			
MEMÓRIA RAM INTERNA DO TMS320C31	809908 809FBF	STACK	.stack	6880	0
	809FC0 809FDF	CODE	vectors	128	128 (100%)
	809FE0 809FFF		trap	128	128 (100%)
	80A000 FFFFFF	ÁREA NÃO USADA			

Tabela V.1 – Mapa de memória do TMS320C31 para o sistema implementado

A área cinza destacada na tabela V.1 corresponde à memória de programa (“code”). Os outros segmentos constituem a memória de dados. Destaca-se que há dois tipos básicos de seções: as que são inicializadas no carregamento do programa (.text, .cinit, .const e .data) e as que reservam espaço na RAM, para que o programa possa utilizá-lo durante a execução para criar e armazenar variáveis (.bss, .stack e .system). O conteúdo destas seções é descrito como segue [14 e 16]:

- .text Código executável e constantes ponto-flutuante;
- .cinit Tabelas com valores das variáveis e constantes de inicialização;

- *.const* Constantes ponto-flutuante e “switch tables”;
- *.data* Dados de inicialização;
- *.bss* Reserva espaço para variáveis globais e estáticas;
- *.stack* Usada para passar argumentos para funções e para alocar variáveis locais;
- *.sysmem* Grande área que pode ser usada durante a execução do programa, pelas funções de memória dinâmica, “malloc”, “calloc” e “realloc”

Os resultados obtidos em termos de memória utilizada são satisfatórios. Por exemplo, no trabalho desenvolvido em [18], onde foi implementada a *Subtração Espectral (SE) Clássica*, a memória total utilizada para programa foi de 18 kbytes. Nesse mesmo trabalho não foi fornecida a quantidade de memória de dados usada. Considerando-se o desempenho do algoritmo implementado, os 10 kbytes a mais são razoáveis.

V.2.2 – Tempo de Processamento

As informações disponíveis no RIDE, com relação ao tempo de processamento, são baseadas no número de ciclos de instrução que o TMS320C31 leva para executar um bloco, ou até mesmo a aplicação completa desenvolvida no RIDE. Considerando que o TMS320C31 usado no projeto possui um ciclo de instrução de 33 ns, foi montada a tabela V.2, contendo valores aproximados do tempo de processamento de cada bloco dos subsistemas do filtro. Os tempos apresentados correspondem ao processamento de um quadro de sinal (16 ms). Os blocos destacados (em cor cinza) na tabela V.2 representam funções que consomem um tempo de processamento relativamente grande, se comparado ao tempo de processamento dos outros blocos que compõem o filtro. Destaca-se que estes blocos, que representam os cálculos da FFT, da Magnitude e da IFFT, possuem código otimizado para o TMS320C31 e são executados num tempo bem menor, se comparado a outros blocos convencionais do RIDE com funções similares. O algoritmo implementado foi adaptado para utilizar estes blocos, o que melhorou muito a performance do sistema, reduziu o tempo de processamento e aumentou significativamente a capacidade de operação do filtro.



<i>Bloco</i>	<i>Tempo de processamento aprox. (em ms)</i>	<i>Percentual do tempo total disponível (16 ms)</i>	<i>Subsistema</i>
<i>Atraso de Quadro</i>	0,067	0,42%	Transformação Tempo- Frequência
<i>Concatenação</i>	0,061	0,38%	
<i>Janela de Hamming (Otimizada)</i>	0,073	0,46%	
<i>FFT Otimizada para C3x</i>	1,120	7,00%	
<i>Magnitude Otimizada para o C3x</i>	0,362	2,26%	
<i>Ao Quadrado</i>	0,035	0,21%	
<i>Atraso de Quadro 1</i>	0,067	0,42%	DIS e Estimação do Ruído
<i>Atraso de Quadro 2</i>	0,067	0,42%	
<i>Atraso de Quadro 3</i>	0,067	0,42%	
<i>Atraso de Quadro 4</i>	0,067	0,42%	
<i>Atraso de Quadro 5</i>	0,067	0,42%	
<i>Multiplex 1</i>	0,002	0,01%	
<i>Multiplex 2</i>	0,031	0,19%	
<i>Multiplex 3</i>	0,031	0,19%	
<i>Média Estimada do Ruído</i>	0,045	0,28%	
<i>Gerador de Zeros</i>	0,001	0,01%	
<i>Atraso com CI=1</i>	0,007	0,04%	
<i>Divisão</i>	0,284	1,78%	
<i>Média</i>	0,092	0,58%	
<i>Armazenamento</i>	0,003	0,02%	
<i>Variância</i>	0,004	0,03%	
<i>Variância Média</i>	0,002	0,01%	
<i>Comparação</i>	0,002	0,01%	
<i>Peso p/ SNR_prio da janela anterior</i>	0,038	0,24%	Cálculo de H(w)
<i>Subtração</i>	0,039	0,24%	
<i>Limitador 1</i>	0,097	0,61%	
<i>Peso p/ SNR_prio da janela atual</i>	0,038	0,24%	
<i>Soma 1</i>	0,036	0,22%	
<i>Limitador 2</i>	0,097	0,61%	
<i>Gerador de 1</i>	0,031	0,19%	
<i>Controle de Redução de Ruído</i>	0,038	0,24%	
<i>Soma 2</i>	0,036	0,22%	
<i>Divisão</i>	0,287	1,79%	
<i>Raiz Quadrada</i>	0,354	2,21%	
<i>Multiplicação</i>	0,039	0,24%	
<i>Atraso de Quadro</i>	0,067	0,42%	
<i>Subsetor</i>	0,030	0,19%	Filtragem
<i>Reverso</i>	0,110	0,69%	
<i>Concatenação</i>	0,065	0,41%	
<i>Misturador</i>	0,115	0,72%	
<i>Multiplicação</i>	0,140	0,88%	
<i>IFFT Otimizada para C3x</i>	1,251	7,82%	Transformação Frequência- Tempo
<i>Separador Re/Im</i>	0,121	0,76%	
<i>Subsetor 1</i>	0,035	0,21%	
<i>Subsetor 2</i>	0,031	0,19%	
<i>Atraso de Quadro</i>	0,072	0,45%	
<i>Soma</i>	0,038	0,24%	
TOTAL:	5,762	36,01%	

Tabela V.2 – Tempo de processamento dos blocos do filtro no TMS320C31



A tabela V.3 mostra o tempo de processamento que os subsistemas ou funções principais do filtro levam para processar um quadro de sinal (16 ms).

<i>Subsistema</i>	<i>Tempo de Processamento (em ms)</i>	<i>Percentual do tempo total (16 ms)</i>
<i>Transformação Tempo-Freqüência</i>	<i>1,72</i>	<i>10,7 %</i>
<i>DIS e Estimação do Ruído</i>	<i>0,84</i>	<i>5,3%</i>
<i>Cálculo de $H(w)$</i>	<i>1,20</i>	<i>7,5%</i>
<i>Filtragem</i>	<i>0,45</i>	<i>2,8%</i>
<i>Transformação Freqüência-Tempo</i>	<i>1,55</i>	<i>9,7%</i>
<i>TOTAL:</i>	<i>5,76</i>	<i>36,0%</i>

Tabela V.3 – Tempo de processamento dos subsistemas/funções do filtro no TMS320C31

Observa-se nas Tabelas V.2 e V.3 que não há indicação do tempo gasto pelos blocos “Conversão A/D” e “Conversão D/A” usados para disparar o conversor A/D-D/A da placa TIGER, que é de aproximadamente 0,14 ms. A justificativa é que estes blocos não realizam processamento, apesar da necessidade de controle no processo de conversão A/D e D/A. Ainda nas Tabelas V.2 e V.3 pode-se verificar que o TMS320C31 usa apenas 36% do tempo total disponível (tempo de duração de um quadro). Considerando estes fatores, foi elaborada a figura V.1 que ilustra graficamente a distribuição do tempo total no TMS320C31.

Para que haja sincronismo entre as conversões A/D e D/A, o bloco D/A absorve o tempo ocioso antes da aquisição de um novo quadro no bloco A/D. A partir deste conceito e considerando o tempo gasto pelos blocos que ativam a conversão A/D e D/A, pode-se estimar a freqüência máxima de operação em tempo real do filtro. O tempo total gasto pela filtragem é de aproximadamente 5,9 ms, que corresponde a uma freqüência de amostragem máxima de aproximadamente 21,7 kHz. Na prática, considerando as freqüências de amostragem



disponíveis para o Conversor A/D-D/A da placa TIGER 31/PC, o algoritmo operou em tempo-real a 19,2 kHz.

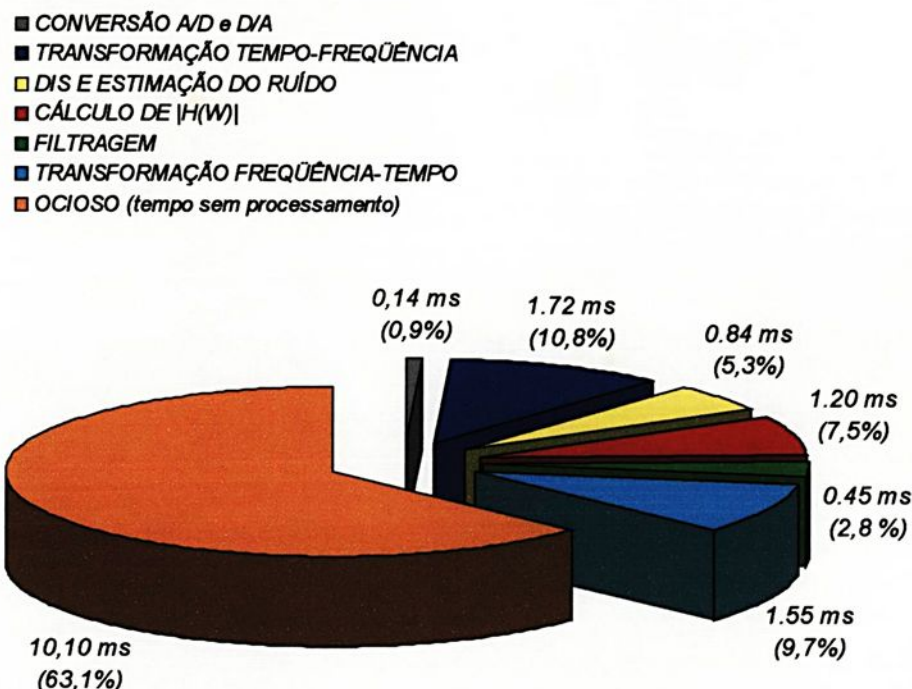


Figura V.1 – Distribuição do tempo de processamento no TMS320C31

A análise do tempo de processamento consumido pelo filtro durante o processamento no TMS320C31 foi usada para avaliar a capacidade do sistema implementado, e pode ser interpretada de duas maneiras:

- a primeira, como já foi colocado, é que o filtro tem capacidade para operar a até 19,2 kHz, considerando a plataforma de hardware utilizada. Uma sugestão é trabalhar com quadros de análise e FFT de 512 amostras e aumentar a frequência de operação do filtro para 16 kHz. Para esta realização pode-se utilizar a mesma plataforma de hardware, com

exceção de que é necessário acrescentar memória RAM à placa TIGER 31/PC, o que é permitido.

- a segunda é que este filtro, operando a 8 kHz, pode ser aperfeiçoado ou configurado, através do acréscimo de novas funções. Por exemplo, um outro método de Detecção de Intervalos de Silêncio (DIS) poderia ser implementado, ou mesmo, novos parâmetros de controle de redução de ruído ou de distorções no processamento poderiam ser incorporados.

O filtro desenvolvido, operando a 8 kHz, está consumindo aproximadamente 36,9% da capacidade do TMS320C31, com relação ao tempo de processamento para operação em tempo-real.

Mais uma vez, comparando os resultados obtidos com os dados disponíveis no trabalho [18], tem-se um desempenho satisfatório. O tempo obtido em [18] para a *Subtração Espectral (SE) Clássica* foi de aproximadamente 4 ms, considerando um *clock* de 60 MHz. A diferença de aproximadamente 2 ms é mais que razoável, visto que o desempenho do filtro aqui implementado é bem superior ao da *SE Clássica*.

V.3 – Avaliação da filtragem realizada pelo filtro implementado

Para uma avaliação da filtragem em tempo real, foram processados três sinais ruidosos: um com alta relação sinal/ruído (SNR), outro com uma SNR média e um último com baixa SNR. Os sinais ruidosos foram gerados no software *Matlab*, em formato .WAV, a partir de um ruído de carro acrescido a um sinal de voz puro de aproximadamente 7,5 segundos de fala, contendo a frase: “*O convênio permite o intercâmbio porque visa a integração entre alunos de culturas diferentes*” de um locutor masculino. Para a geração de sinais com diferentes SNRs, foram aplicados ganhos ao ruído antes que este fosse somado ao sinal de voz puro original. A avaliação da filtragem foi realizada a partir das formas de onda, dos gráficos de níveis de redução de ruído e dos espectrogramas. Os parâmetros definidos para o filtro



correspondem aos valores apresentados no capítulo IV, definidos em várias seções que descreveram os subsistemas do filtro, com exceção do parâmetro de controle de redução de ruído (ρ) (descrito na seção IV.3.3 e definido na equação II.16). Alterando este parâmetro, foram realizadas duas filtrações para cada sinal ruidoso: a primeira fazendo $\rho=1$ e uma segunda com $\rho=1,5$. Destaca-se que quanto maior o valor de ρ maior será a redução de ruído atingida após a filtração, porém, maiores serão as distorções provocadas no sinal processado, em relação ao sinal de voz puro (original). Apesar de se conhecer os efeitos físicos provocados pela variação do parâmetro ρ no sinal processado, esta alteração serviu para uma avaliação subjetiva, realizada com várias pessoas, que analisaram auditivamente a filtração obtida com $\rho=1$ e $\rho=1,5$. O resultado desta avaliação subjetiva é apresentado na seção V.3.5.

V.3.1 – Formas de Onda dos Sinais temporais

Os sinais de voz ruidosos usados como entrada para o filtro redutor de ruído, correspondem a um mesmo trecho de voz somado a um mesmo trecho de ruído, mas com ganhos diferentes, gerando assim, sinais ruidosos com diferentes relações sinal/ruído.

A forma de onda do sinal de voz puro (original) utilizado na formação dos sinais ruidosos é mostrada na figura V.2. As formas de onda dos sinais ruidosos com SNRs alta, média e baixa, são mostrados, respectivamente, nas figuras V.3, V.4 e V.5. As formas de onda dos correspondentes sinais filtrados, processados pelo filtro configurado com $\rho=1$, são apresentados nas figuras V.6, V.7 e V.8. Para o filtro configurado com $\rho=1,5$, os sinais filtrados são mostrados nas figuras V.9, V.10 e V.11.



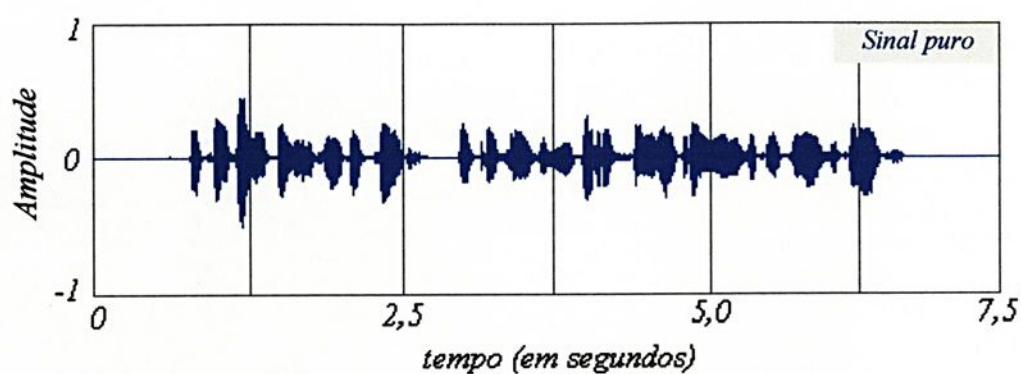


Figura V.2 – Sinal puro original



Figura V.3 – Sinal ruidoso com alta SNR



Figura V.4 – Sinal ruidoso com média SNR

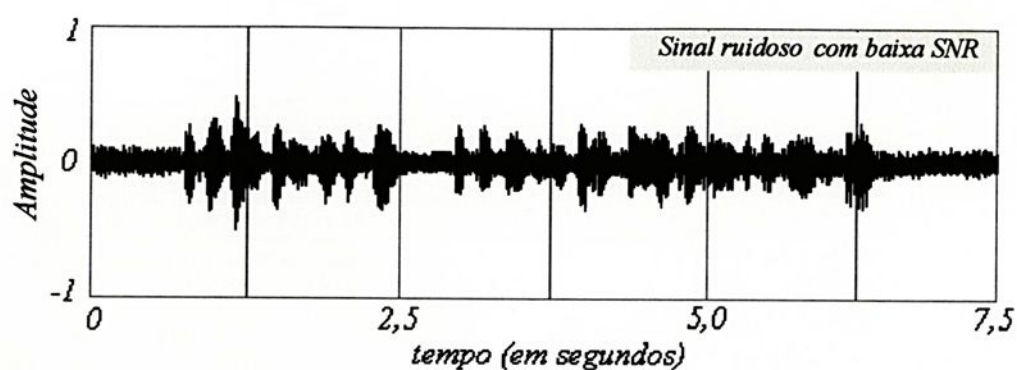


Figura V.5 – Sinal ruidoso com baixa SNR

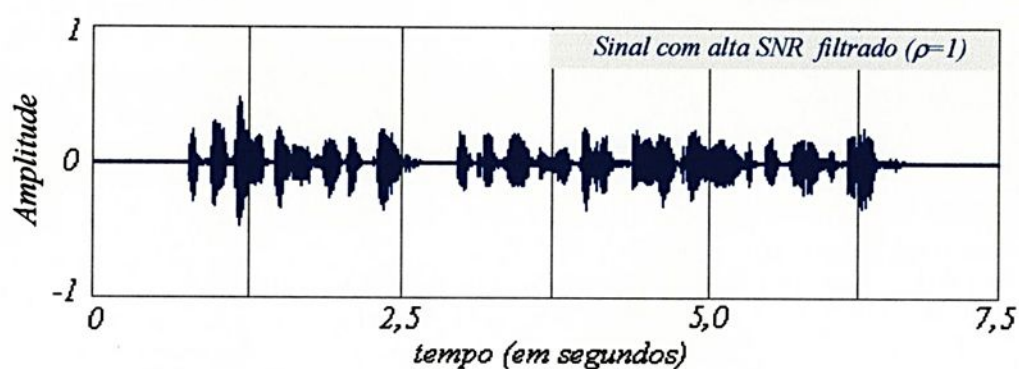


Figura V.6 – Sinal com alta SNR filtrado ($\rho=1$)

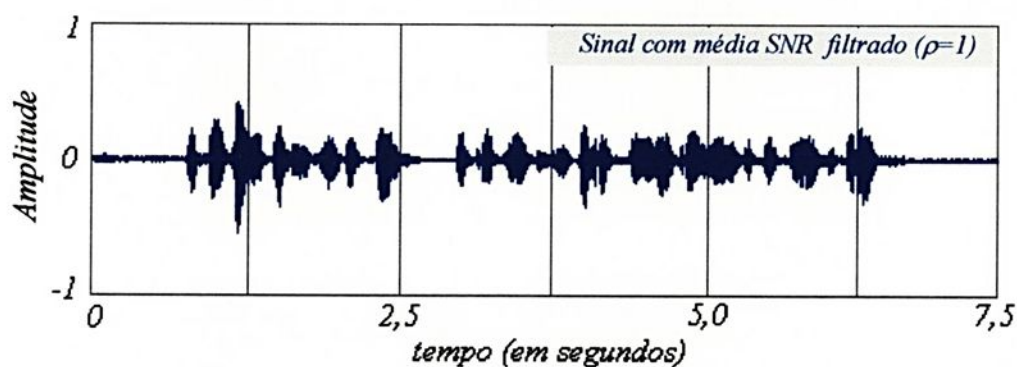


Figura V.7 – Sinal com média SNR filtrado ($\rho=1$)



Figura V.8 – Sinal com baixa SNR filtrado ($\rho=1$)



Figura V.9 – Sinal com alta SNR filtrado ($\rho=1,5$)

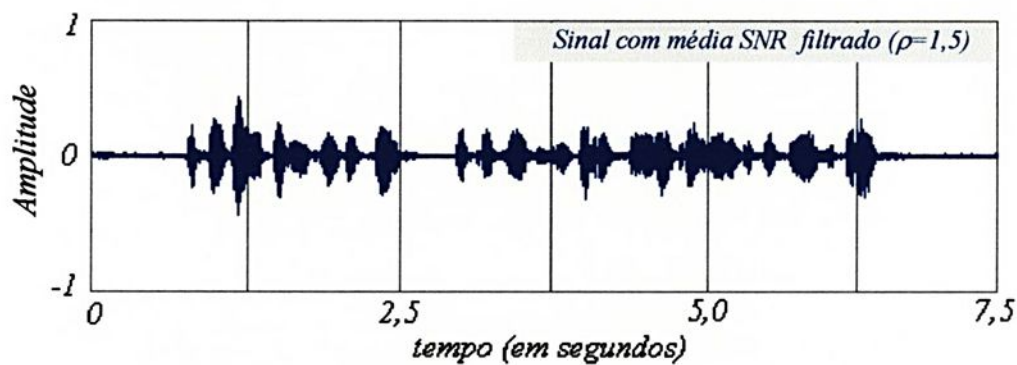


Figura V.10 – Sinal com média SNR filtrado ($\rho=1,5$)

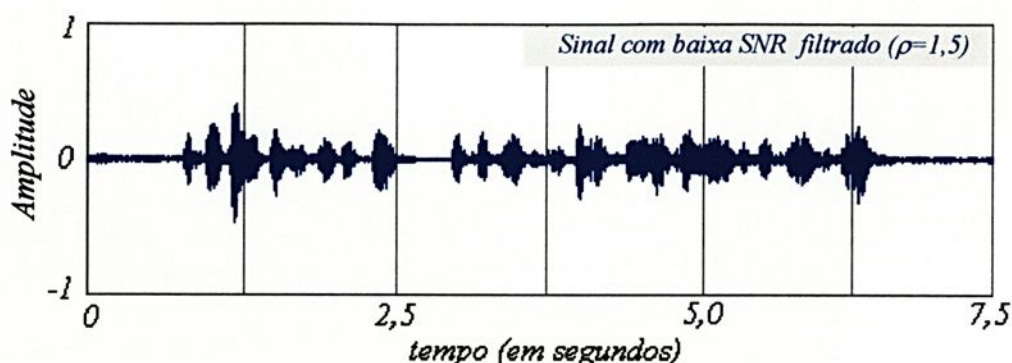


Figura V.11 – Sinal com baixa SNR filtrado ($\rho=1,5$)

Observa-se, analisando os gráficos das figuras anteriores, que nos intervalos de silêncio⁶ o ruído foi bastante atenuado. Nos trechos de voz, também se reduziu o ruído. Para os sinais filtrados com $\rho=1,5$ a redução de ruído é maior, se comparado aos resultados obtidos com $\rho=1$. No entanto, em trechos de voz onde a SNR é relativamente baixa, as distorções são maiores quando se atribui a ρ o valor de 1,5.

V.3.2 – Detecção de Intervalos de Silêncio (DIS)

A Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISV) implementada no filtro, fornece um sinal de controle da atualização da potência do ruído estimada, como foi descrito na seção IV.3.2. Este sinal, que tem amplitude zero (0) nos trechos de fala do sinal ruidoso e um (1) nos intervalos de silêncio, é mostrado nas Figuras V.12, V.13 e V.14, correspondendo a DIS realizada, respectivamente, no processamento dos sinais ruidosos com alta, média e baixa SNR.

⁶ Os intervalos de silêncio compreendem aproximadamente os períodos de 0 a 0,8 segundo, 2,6 a 2,8 segundos e 6,7 a 7,5 segundos.

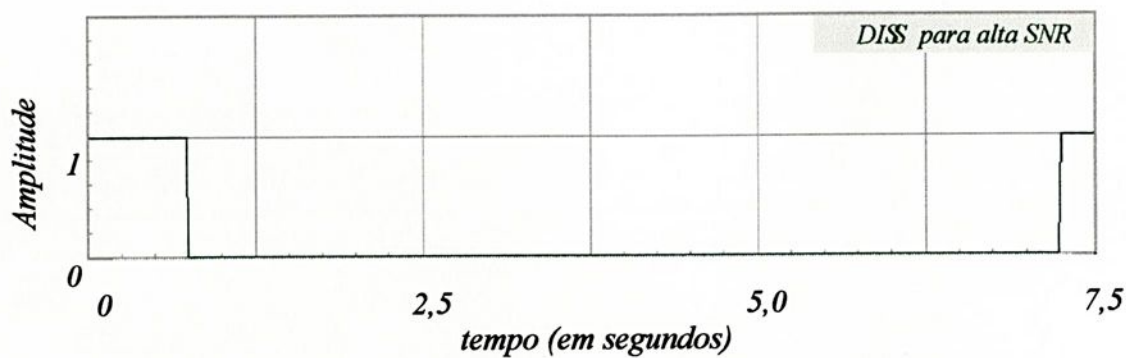


Figura V.12 – DIS do sinal ruidoso com alta SNR

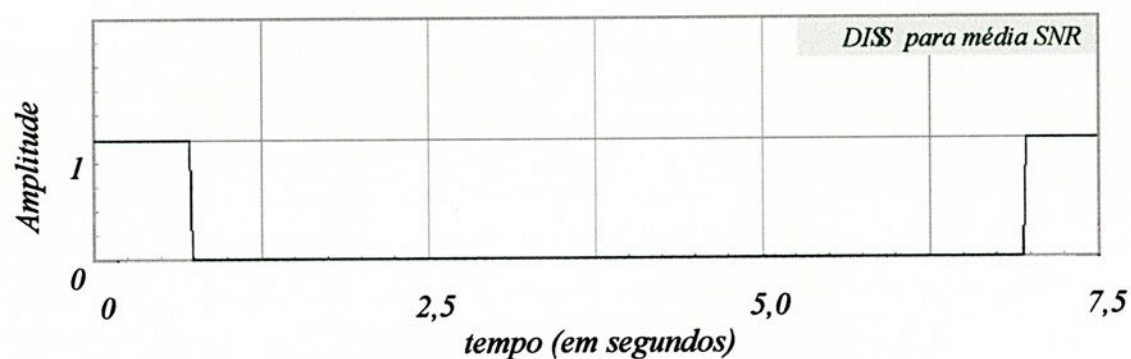


Figura V.13 – DIS do sinal ruidoso com SNR média

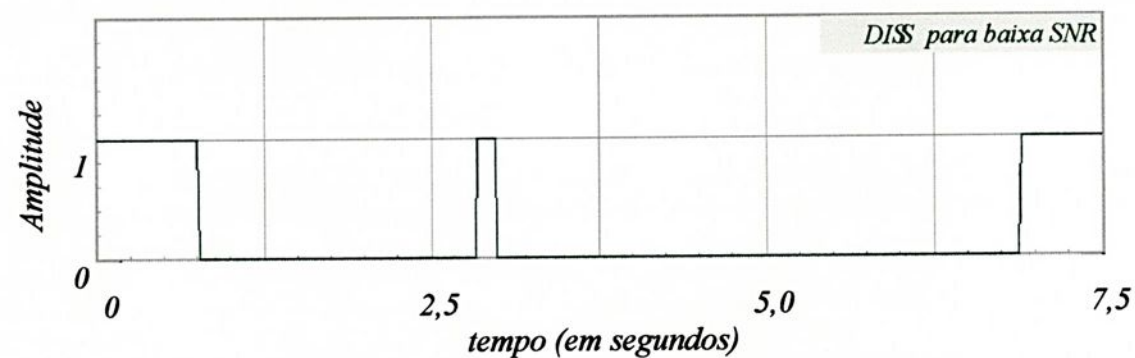


Figura V.14 – DIS do sinal ruidoso com baixa SNR

Nota-se, analisando os sinais fornecidos pela *DISS*, mostrados nas Figuras V.12, V.13 e V.14, que a detecção de intervalos de silêncio foi realizada corretamente, mesmo considerando o sinal com baixa SNR. Esta análise pode ser feita comparando-se os sinais da *DISS* com a forma de onda do sinal puro original, mostrado no gráfico da figura V.2. Uma correta *DIS* é a garantia de que a filtragem realizada corresponde à técnica de redução de ruído implementada, no caso, a *Subtração Espectral Modificada (SEM)*.

V.3.3 – Relações Sinal/Ruído e Redução de Ruído obtida na filtragem

Seja $y[n]$ o sinal ruidoso, $E\{v[n]\}$ o sinal de voz estimado, obtido após a filtragem, e $E\{r[n]\}$ o ruído estimado. Considerando que os sinais já estão discretizados, uma análise temporal das relações sinal/ruído pode ser obtida como a seguir:

$$SNR_post(k) = \frac{\sum_n (y[n])^2}{\sum_n (E\{r[n]\})^2} \quad e \quad SNR_prio(k) = \frac{\sum_n (E\{v[n]\})^2}{\sum_n (E\{r[n]\})^2} \quad (V.1)$$

onde:

$SNR_post(k)$ = relação sinal / ruído a posteriori temporal;

$SNR_prio(k)$ = relação sinal / ruído a priori temporal;

n = corresponde ao tamanho da janela em análise, no caso, $n = 128$;

k = corresponde ao tempo, incrementado a cada janela analisada.

A potência estimada do ruído é obtida durante os intervalos de silêncio, calculando-se um valor médio do ruído, como foi descrito na seção II.4.3.

O nível de redução de ruído obtido na filtragem pode ser definido como a seguir:

$$R_{dB}(k) = 10 \log \frac{\sum_n (y[n])^2}{\sum_n (E\{v[n]\})^2} \quad (V.2)$$



O valor de R na equação V.2 fornece um indicativo do nível de redução de ruído obtido na filtragem, sem mostrar detalhes de variações frequenciais do sinal filtrado com relação ao sinal de voz puro original. Destaca-se que o denominador da equação V.2 jamais será nulo, porque nos intervalos de silêncio do sinal de voz estimado a partir do sinal ruidoso, haverá sempre um ruído residual.

Após o processamento dos sinais ruidosos, foram obtidos gráficos no próprio RIDE, que mostram o comportamento das relações sinal/ruído a posteriori (SNR_{post}) e a priori (SNR_{prio}), bem como o nível de redução de ruído obtido na filtragem. Todos estes gráficos estão em decibéis (dB).

Os gráficos das figuras V.15, V.16 e V.17 mostram, respectivamente, as relações sinal/ruído a posteriori ($SNR_{post}(k)$) para os sinais ruidosos com alta SNR (fig. V.3), média SNR (fig. V.4) e baixa SNR (fig. V.5). As figuras V.18, V.19 e V.20 mostram as correspondentes relações sinal/ruído a priori ($SNR_{prio}(k)$) para o filtro configurado com $\rho=1$ e as figuras V.21, V.22 e V.23 valem para $\rho=1,5$. Os gráficos que mostram os níveis de redução de ruído obtidos após as filtrações dos sinais ruidosos com alta, média e baixa SNR, considerando $\rho=1$, são mostrados, respectivamente, nas figuras V.24, V.25 e V.26. Os gráficos de redução de ruído para $\rho=1,5$ são ilustrados nas figuras V.27, V.28 e V.29.

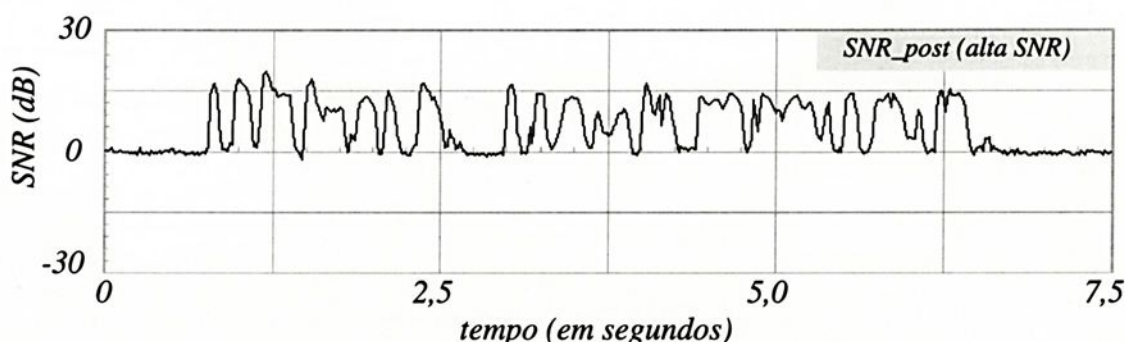


Figura V.15 – SNR_{post} para o sinal ruidoso com alta SNR



Figura V.16 – SNR_post para o sinal ruidoso com média SNR

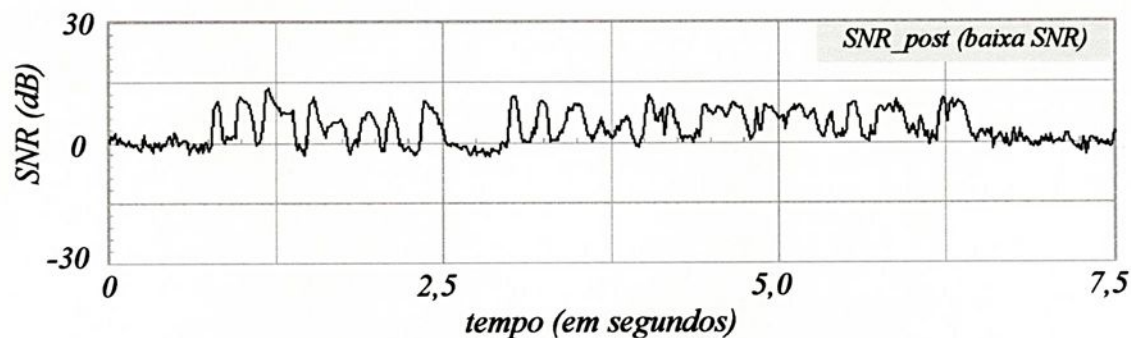


Figura V.17 – SNR_post para o sinal ruidoso com baixa SNR

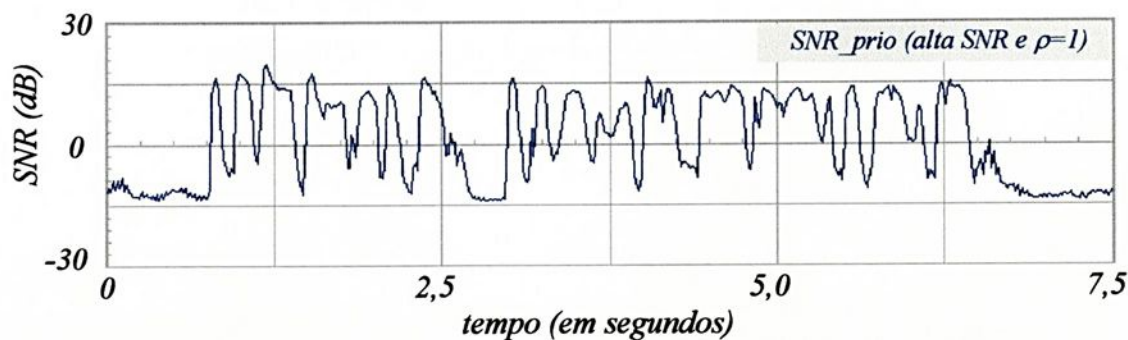


Figura V.18 – SNR_prio para o sinal ruidoso com alta SNR e $\rho=1$

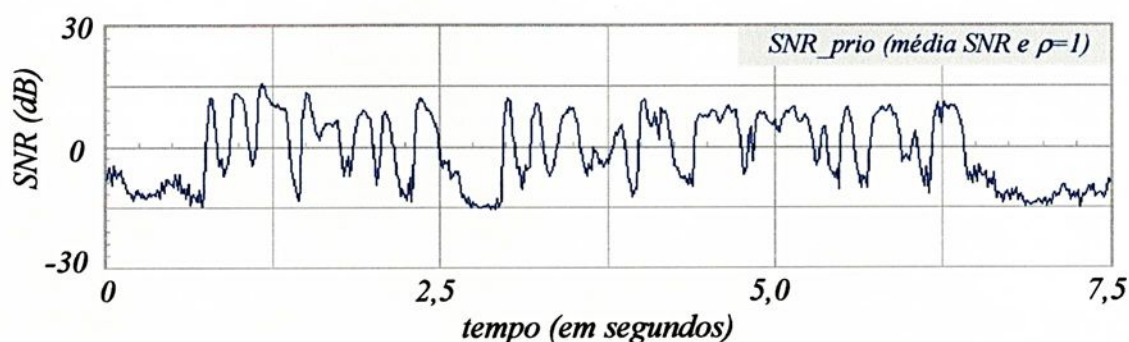


Figura V.19 – SNR_prio para o sinal ruidoso com média SNR e $\rho=1$

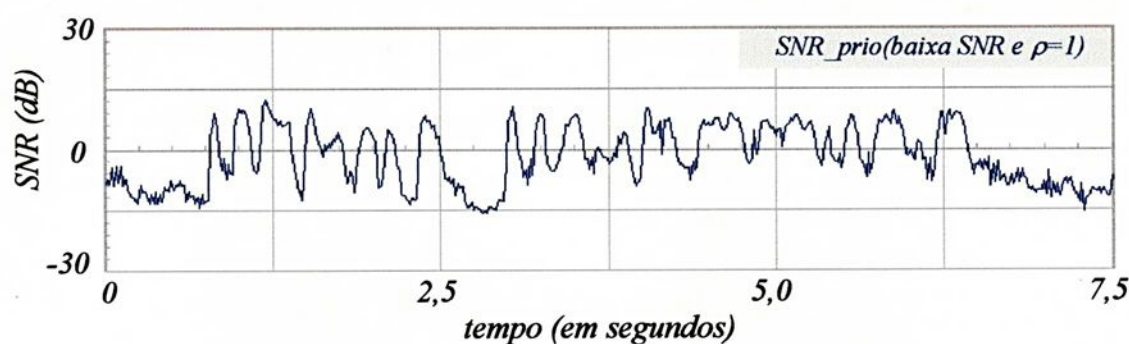


Figura V.20 – SNR_prio para o sinal ruidoso com baixa SNR e $\rho=1$

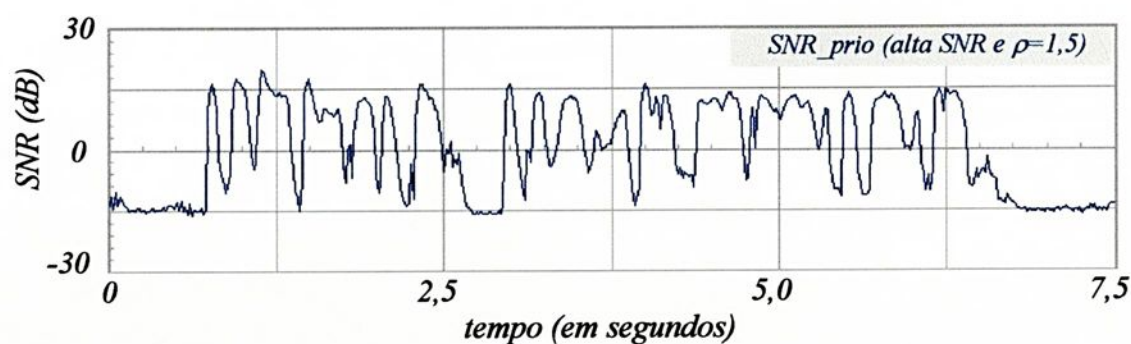


Figura V.21 – SNR_prio para o sinal ruidoso com alta SNR e $\rho=1,5$

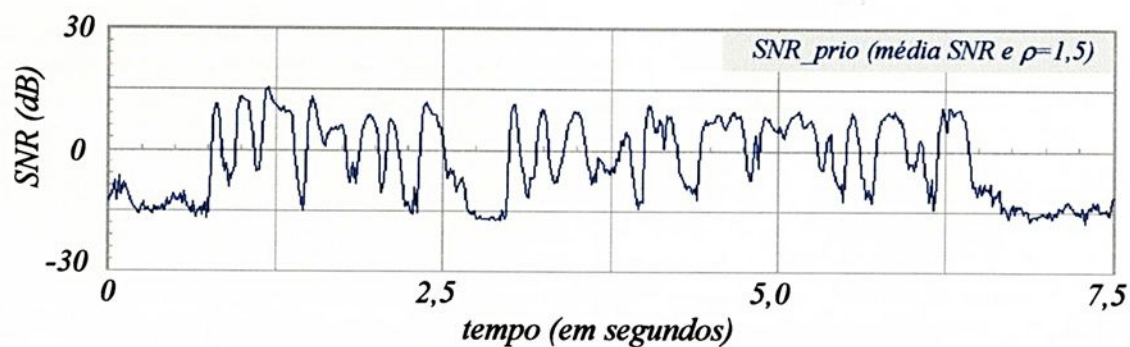


Figura V.22 – SNR_{prio} para o sinal ruidoso com média SNR e $\rho=1,5$

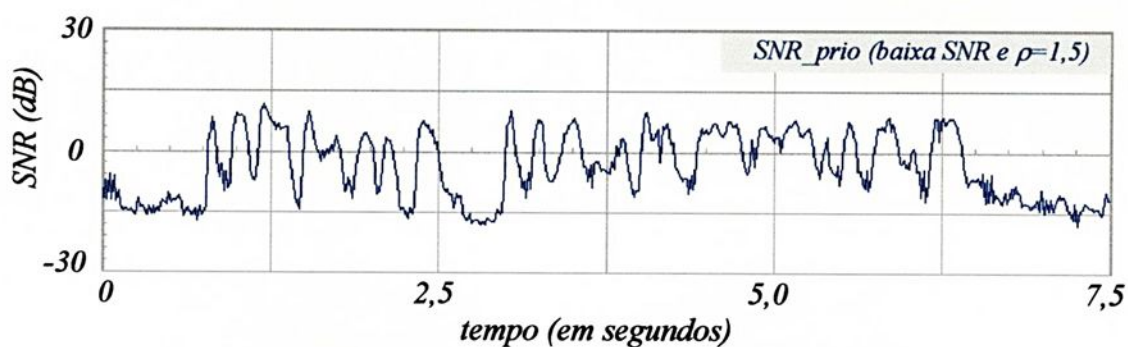


Figura V.23 – SNR_{prio} para o sinal ruidoso com baixa SNR e $\rho=1,5$



Figura V.24 – Redução de Ruído obtida após a filtragem do sinal de alta SNR ($\rho=1$)

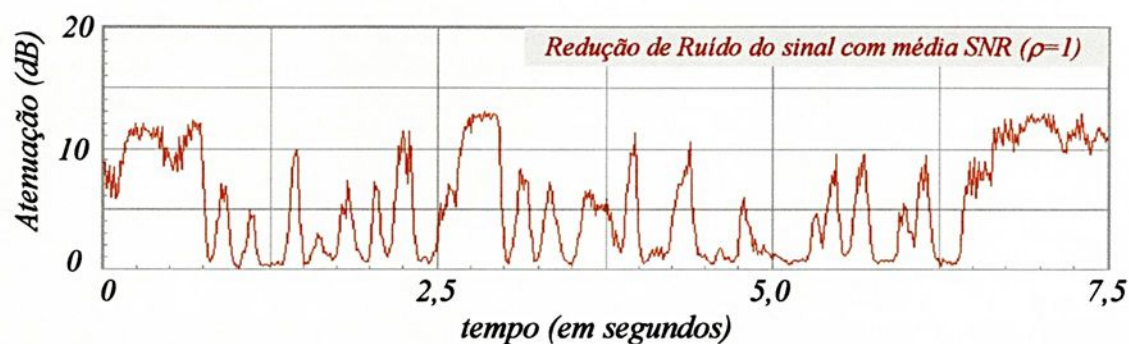


Figura V.25 – Redução de Ruído obtida após a filtragem do sinal de média SNR ($\rho=1$)

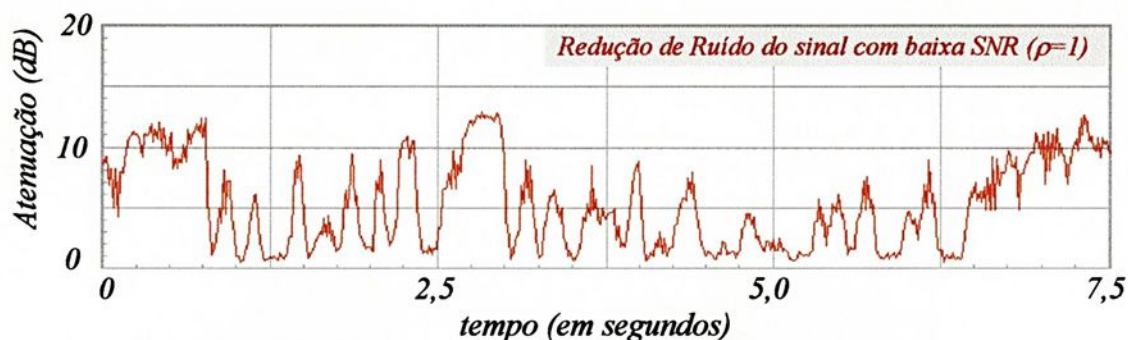


Figura V.26 – Redução de Ruído obtida após a filtragem do sinal de baixa SNR ($\rho=1$)

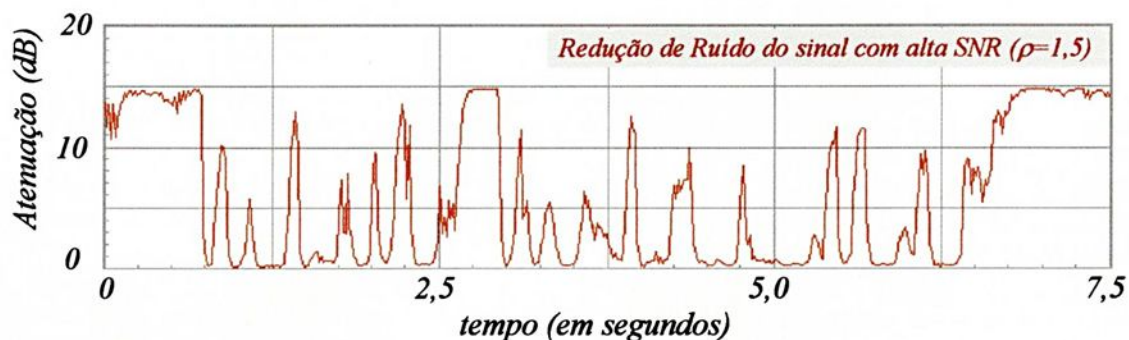


Figura V.27 – Redução de Ruído obtida após a filtragem do sinal de alta SNR ($\rho=1,5$)



Figura V.28 – Redução de Ruído obtida após a filtragem do sinal de média SNR ($\rho=1,5$)

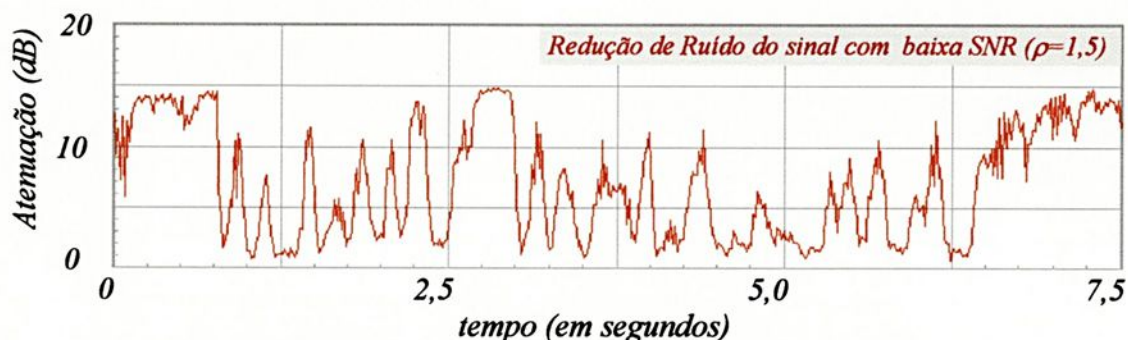


Figura V.29 – Redução de Ruído obtida após a filtragem do sinal de baixa SNR ($\rho=1,5$)

Nota-se, analisando os gráficos de redução de ruído, que, nos intervalos de silêncio, na média, para $\rho=1$, a redução de ruído está em torno de 10 dB e para $\rho=1,5$, próxima de 15 dB.

V.3.4 – Espectrogramas

O espectrograma do sinal de voz puro é mostrado na figura V.30. Os espectrogramas dos sinais ruidosos com alta, média e baixa SNRs, são apresentados, respectivamente, nas figuras V.31, V.32 e V.33. Para os correspondentes sinais filtrados, foram obtidos espectrogramas considerando $\rho=1$, ilustrados nas figuras V.34, V.35 e V.36, e $\rho=1,5$, ilustrados nas figuras V.37, V.38 e V.39. Analisando estas figuras, observa-se uma melhoria no espectro após a

filtragem. O “ruído de carro” adicionado ao sinal de voz puro para a formação do sinal ruidoso, degradou o espectro de voz, como pode ser observado nos gráficos das figuras V.31, V.32 e V.33. Comparando-se o espectrograma do sinal ruidoso e filtrado ao espectrograma do sinal puro, para cada caso de relação sinal/ruído (SNR) verifica-se que as componentes espectrais da voz filtrada identificam-se melhor com as componentes espectrais da voz pura, em relação ao sinal ruidoso original. Isto pode ser analisado, comparando-se os espectrogramas do sinal ruidoso e do sinal filtrado ao espectrograma do sinal de voz puro, mostrado na figura V.30. Entretanto, observa-se também, principalmente na faixa de 800 Hz a 1.800 Hz que os sinais processados apresentam distorções (fraca energia nos intervalos de voz).

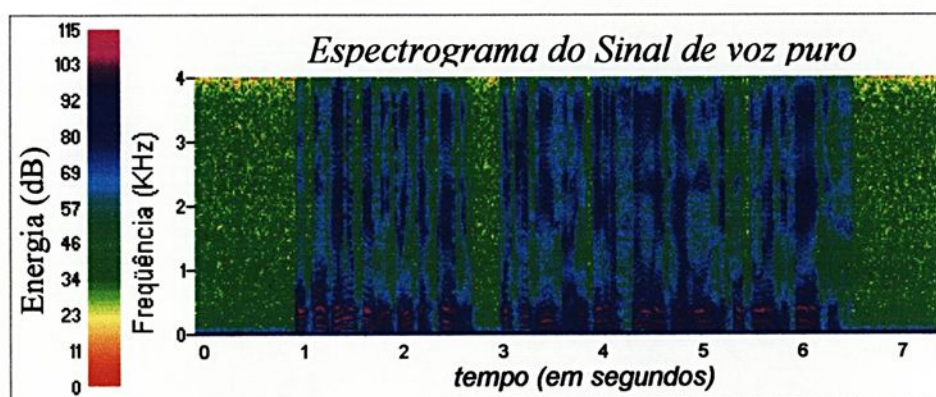


Figura V.30 – Espectrograma do sinal de voz puro original

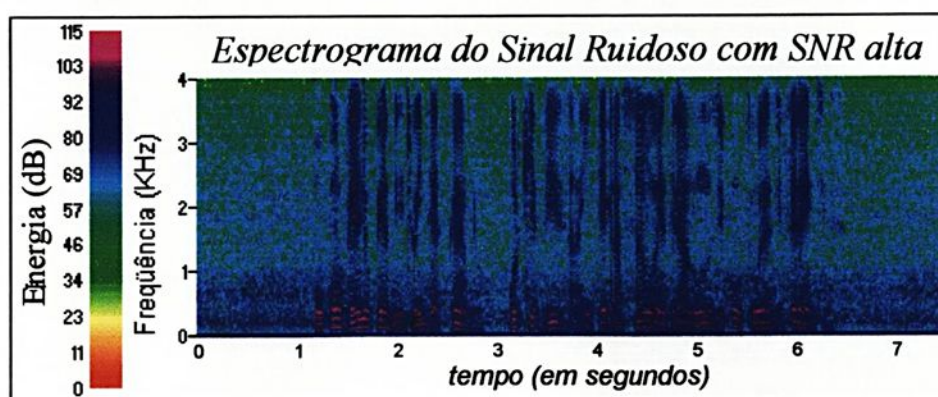


Figura V.31 – Espectrograma do sinal de voz ruidoso com alta SNR



Figura V.32 – Espectrograma do sinal de voz ruidoso com média SNR



Figura V.33 – Espectrograma do sinal de voz ruidoso com baixa SNR

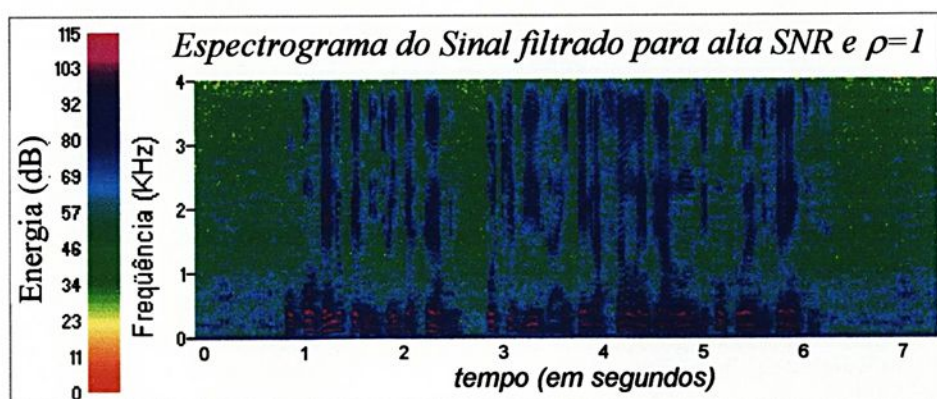


Figura V.34 – Espectrograma do sinal de voz filtrado para alta SNR e $\rho=1$

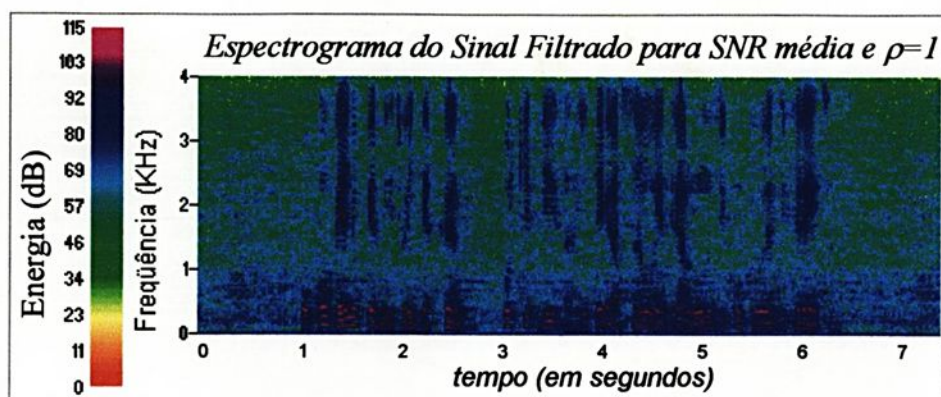


Figura V.35 – Espectrograma do sinal de voz filtrado com média SNR e $\rho=1$

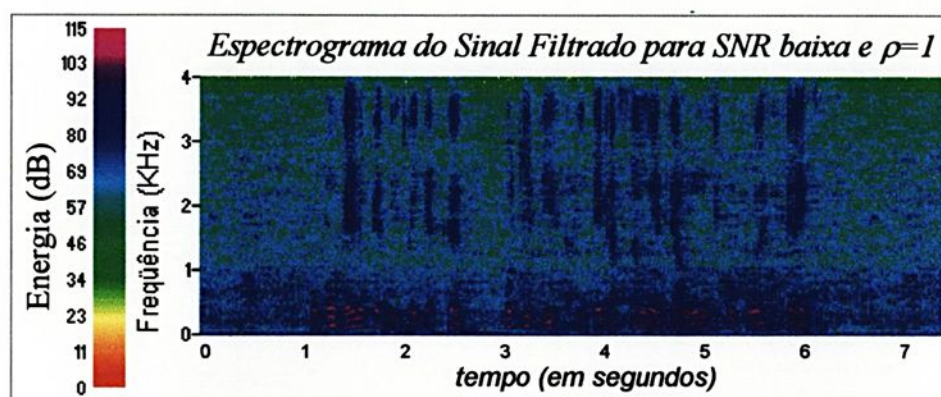


Figura V.36 – Espectrograma do sinal de voz filtrado com baixa SNR e $\rho=1$

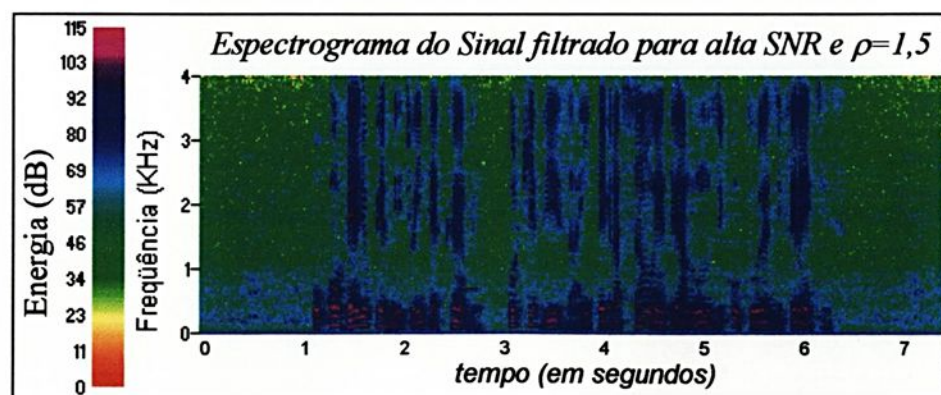


Figura V.37 – Espectrograma do sinal de voz filtrado com alta SNR e $\rho=1,5$

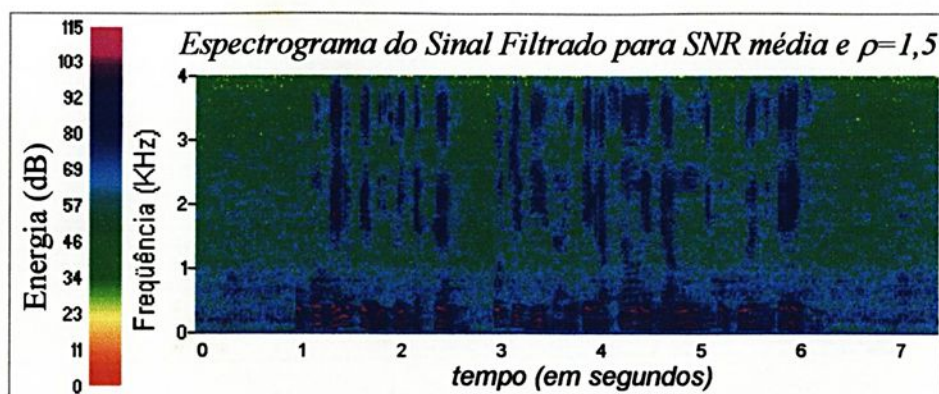


Figura V.38 – Espectrograma do sinal de voz filtrado com média SNR e $\rho=1,5$

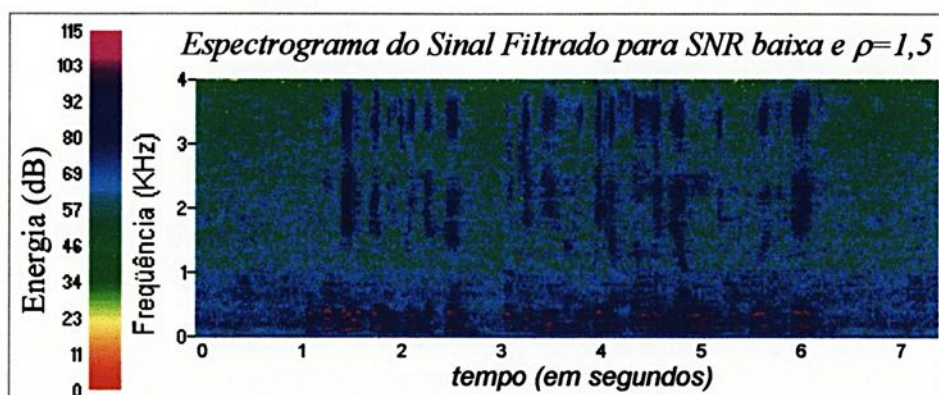


Figura V.39 – Espectrograma do sinal de voz filtrado com baixa SNR e $\rho=1,5$

V.3.5 – Testes subjetivos

Os testes subjetivos consistiram na reprodução dos sinais puro, ruidoso e filtrado para dez pessoas (ouvintes), sendo oito do sexo masculino e duas do feminino. As avaliações foram realizadas por cada uma destas pessoas, considerando os seguintes casos:

1. Comparação entre sinal puro, sinal ruidoso com alta SNR e sinal com alta SNR filtrado, com $\rho=1$
2. Comparação entre sinal puro, sinal ruidoso com alta SNR e sinal com alta SNR filtrado, com $\rho=1,5$

3. Comparação entre sinal puro, sinal ruidoso com média SNR e sinal com média SNR filtrado, com $\rho=1$
4. Comparação entre sinal puro, sinal ruidoso com média SNR e sinal com média SNR filtrado, com $\rho=1,5$
5. Comparação entre sinal puro, sinal ruidoso com baixa SNR e sinal com baixa SNR filtrado, com $\rho=1$
6. Comparação entre sinal puro, sinal ruidoso com baixa SNR e sinal com baixa SNR filtrado, com $\rho=1,5$

O resultado desta comparação é um conceito de 0 a 5. O objetivo é que, na comparação do sinal filtrado com o sinal ruidoso, sejam avaliados subjetivamente o nível de redução e o conforto auditivo obtidos após a filtragem. Quando o sinal filtrado é comparado ao sinal puro original, a avaliação corresponde a uma análise subjetiva das distorções provocadas no processamento. O gráfico da figura V.40 ilustra o resultado desta avaliação para os casos anteriores, considerando uma média dos conceitos obtidos.

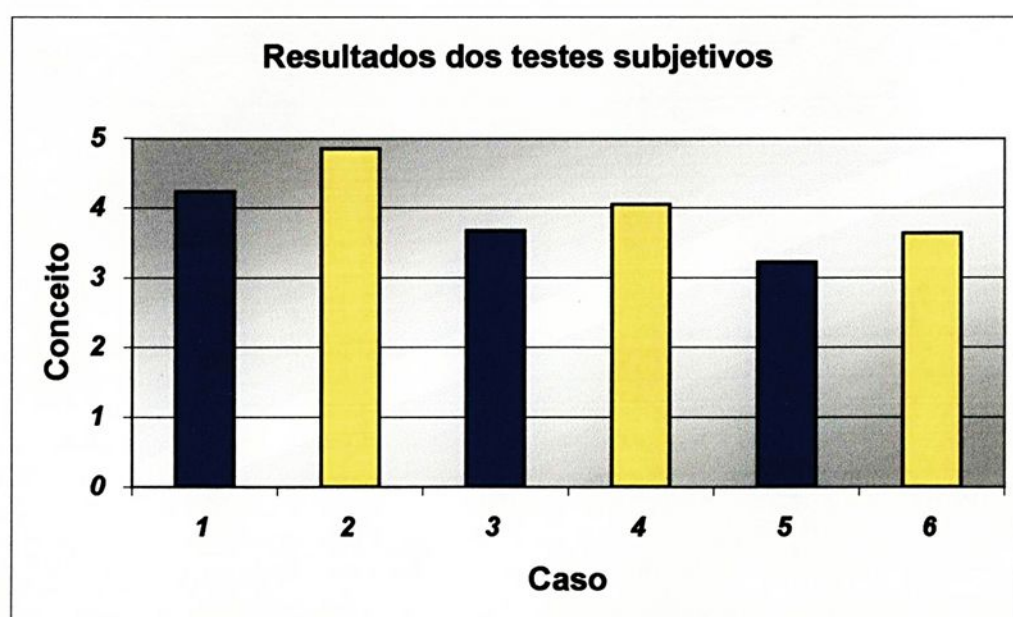


Figura V.40 – Conceitos médios obtidos nos testes subjetivos para os seis casos analisados

Os resultados subjetivos ilustrados na figura V.40 mostram que, o sinal processado com $\rho=1,5$, apesar de ter maiores distorções, principalmente nos casos de baixa SNR, provoca um conforto auditivo maior. Isto se deve ao fato da redução de ruído obtida na filtragem ser maior, aproximadamente 15 dB (medida nos intervalos de silêncio). Os resultados dos testes subjetivos mostraram, além de avaliar a filtragem, a influência do parâmetro ρ no desempenho do filtro. No entanto, o ajuste deste parâmetro, assim como os outros parâmetros do filtro, devem ser feitos considerando a aplicação para a qual este será destinado.

V.4 – Conclusões

O objetivo deste capítulo foi mostrar a eficiência do sistema redutor de ruído para sinais de voz implementado em tempo real.

O filtro implementado consumiu um total de 87,7% (28 Kbytes) da memória de programa e 39,9% (42 Kbytes) da memória de dados disponíveis na placa TIGER 31/PC. O tempo de processamento também foi observado obtendo-se bons resultados. Este corresponde a 36,9% da capacidade de processamento total do TMS320C31 operando a 60 MHz. Destaca-se que a taxa de amostragem é de 8 kHz e que o filtro pode operar em tempo real, na plataforma utilizada, em até 19,2 kHz.

O resultado dos testes mostrou que o filtro implementado efetua uma significativa redução de ruído. Considerando a aplicação em sistemas telefônicos convencionais, as distorções do sinal filtrado em relação ao sinal puro são poucas, desde que a relação sinal/ruído do sinal ruidoso não seja muito pequena. Outro fator fundamental para o bom desempenho deste filtro é a correta estimação do ruído, viabilizada por um método de detecção de intervalos de silêncio eficiente. O detector usando a variância da relação sinal/ruído (*DISS*) mostrou-se eficiente para todos os sinais ruidosos processados, o que garantiu a aplicação da técnica de *Subtração Subtração Espectral (SEM)* corretamente.

O resultado dos testes subjetivos mostrou que o sinal processado é mais confortável que o sinal ruidoso.



CAPÍTULO VI

COMENTÁRIOS FINAIS

Neste trabalho implementou-se um algoritmo de redução de ruído em sinais de voz em tempo real usando o processador de sinais digitais TMS320C31. O algoritmo implementado possui baixa carga computacional, pois é baseado em técnicas simples, porém eficientes: a *Subtração Espectral Modificada (SEM)* e a *Detecção de Intervalos de Silêncio usando a Variância da Relação Sinal/Ruído (DISS)*.

Os estudos para atingir o objetivo final deste trabalho iniciaram-se com a elaboração do filtro redutor de ruído no Simulink, uma interface gráfica do software Matlab. Este desenvolvimento, embora não descrito neste trabalho, foi uma importante etapa, pois permitiu a definição de um diagrama de blocos para o filtro, a realização de testes, bem como uma familiarização com o desenvolvimento de aplicações em ambiente gráfico, utilizando funções de processamento de sinais digitais.

A plataforma utilizada no projeto do filtro para operação em tempo real, composta basicamente da placa TIGER 31/PC com um processador de sinais digitais TMS320C31 e do software de desenvolvimento Hypersignal RIDE, é ideal para aplicação em áudio. O desenvolvimento deste trabalho mostrou que esta plataforma, incluindo a quantidade de memória RAM disponibilizada na TIGER 31/PC, foi suficiente para o processamento do filtro em tempo real. Os resultados de testes mostraram que os recursos do TMS320C31 e da TIGER 31/PC foram bem aproveitados.

Uma implementação do mesmo algoritmo em linguagem C poderia gerar um código mais otimizado. No entanto, o desenvolvimento descrito neste trabalho mostrou que o filtro está otimizado para o ambiente gráfico para o qual foi desenvolvido. O fato de trabalhar-se com diagrama de blocos permitiu flexibilidade na implementação do algoritmo e facilitou o desenvolvimento de um novo método de detecção de intervalos de silêncio. Trabalhar com interfaces gráficas parece ser uma tarefa fácil, mas não é. Para exemplificar este fato, basta dizer que, inicialmente, o primeiro arquivo referente ao filtro, gerado no RIDE, operava em tempo real a apenas 7350 Hz. A última versão, apresentada neste trabalho, opera a até 19200 Hz. Para se chegar a esta versão, mais de 300 (trezentos) arquivos (de filtro) foram criados, buscando sempre reduzir o número de funções utilizadas e/ou utilizar funções otimizadas.



A avaliação dos resultados de testes, realizados durante a execução do filtro, mostrou que o filtro implementado apresentou um ganho significativo na SNR após a filtragem, que pode ser controlado pelo valor de ρ (um fator de subestimação/superestimação do ruído). Os espectrogramas mostraram que, para uma SNR não muita baixa, as distorções são pequenas. Com relação ao tempo de processamento, obtiveram-se ótimos resultados. A execução de uma janela de sinal consome aproximadamente 5,9 ms, incluindo o tempo necessário para a conversão A/D e D/A. Além disso, o filtro requer uma área de memória de 28 Kbytes para programa e 42 Kbytes para dados.

Os testes subjetivos realizados mostraram que a filtragem melhora o sinal ruidoso, tornando-o mais inteligível e confortável do ponto de vista auditivo.

O trabalho desenvolvido pode ser melhorado em dois aspectos:

1. Alterar o algoritmo de redução de ruído visando minimizar distorções no sinal processado;
2. Implementar o filtro diretamente em linguagem C, buscando minimizar tempo de processamento e quantidade de memória.

Por fim, deve-se destacar que esse sistema será testado o mais breve possível em uma linha telefônica analógica.



REFERÊNCIAS BIBLIOGRÁFICAS

- [1] BOLL, S. F. "Suppression of Acoustic Noise in Speech using Spectral Subtraction", *IEEE Trans. Acoust., Speech, and Signal Processing (ASSP)*, vol. 29, p113-120, April-1979.
- [2] CAPPÉ, O. "Elimination of the Musical Noise Phenomenon with the Ephraim and Malah Noise Suppressor", *IEEE Trans. on Acoust., Speech and Signal Processing (TASSP)*, April 1994.
- [3] EPHRAIM, Y & Malah, D., "Speech Enhancement Using Minimum Mean Square Error Short-Time Spectral Amplitude Estimator", *IEEE Trans. on Acoust., Speech and Signal Processing*, vol. 32, N° 6, December-1984.
- [4] EPHRAIM, Y, "Statistical-Model-Based Speech Enhancement Systems", in *Proc. of IEEE*, Vol. 80, N° 10, p. 1526-1555, october, 1992.
- [5] FLANAGAN, J. L. "Speech Analysis Synthesis and Perception", Ed. Springer-Verlag, 1972.
- [6] McAULAY, J. R. & MALPASS, M, "Speech Enhancement Using a Soft-Decision Noise Suppression Filter", *IEEE Trans. on Acoust., Speech and Signal Processing*, vol. 28, N° 2, April-1980.
- [7] OPPENHEIM & SCHAFER, 1989, "Discrete-Time Signal Processing", Ed. Prentice-Hall, 1989
- [8] RABINER, L. R. & SHAFER, R. W, "Digital Processing of Speech Signals", Ed. Prentice-Hall, 1988.
- [9] VIEIRA FILHO, J. "Redução de Ruído em Sinais de Voz nos Sistemas Rádio Móveis Veiculares", *Tese de Doutorado (Unicamp)*, 1996.
- [10] RIDE Function Reference, Hyperception Inc., 1998
- [11] RIDE User's Manual, Hyperception Inc., 1998
- [12] TIGER 31/PC Reference Manual, DSP Research Inc., 1995
- [13] TIGER PowerPack Reference Manual, DSP Research Inc., 1995
- [14] TMS320 C3x/C4x Optimizing C Compiler User's Guide, Texas Instruments Inc, 1998
- [15] TMS320 DSP Development Support Reference Guide, Texas Instruments Inc., 1998
- [16] TMS320 DSP Product Family Glossary, Texas Instruments, 1998
- [17] TMS320C31, TMS320LC31 Digital Signal Processors, Texas Instruments Inc, 1999
- [18] DAVIDEK, V. et alli, " Implementing a Noise Cancellation System with the TMS320C31", in: First European DSP Education and Research, Conference, Paris, sept. 1996.



**UNIVERSIDADE ESTADUAL PAULISTA - UNESP
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
AV. BRASIL, 56 - CAIXA POSTAL 31
15385-000 - ILHA SOLTEIRA - SP**

