UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rui Seara Júnior

IMPLEMENTAÇÃO DE UM CODIFICADOR DE VOZ PADRONIZADO EM DSP

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Jorge Muniz Barreto
Orientador

Florianópolis, outubro de 2000

IMPLEMENTAÇÃO DE UM CODIFICADOR DE VOZ PADRONIZADO EM DSP

Rui Seara Júnior

Esta Dissertação foi julgada adequada para a obtenção do título de **Mestre em Ciência da Computação** Área de Concentração **Sistemas de Conhecimento** e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Jorge Muniz Barreto, D.Sc.A.
Orientador

Prof. Fernando A. Ostuni Gauthier, Dr.
Coordenador do Curso de Pós-Graduação

Banca Examinadora:

Prof. Jorge Muniz Barreto, D.Sc.A.
Presidente

Prof. Sidnei Noceti Filho, D.Sc.

Lúcia H. Martins Pacheco

Profa. Lúcia Helena Martins Pacheco, M.Sc.

Prof. João Bosco da Mota Alves, Dr.

Agradecimentos

Ao Prof. Jorge Muniz Barreto, por aceitar orientar-me nesta dissertação.

Ao Prof. Rui Seara, pelos esclarecimentos e sugestões sobre as técnicas de processamento de sinais, mais especificamente em codificação de voz.

À Prof[a]. Izabel Christine Seara, pelas sugestões e correções gramaticais na redação desta dissertação.

Aos colegas de trabalho do LINSE, especialmente aos Engenheiros Walter Antônio Gontijo e Eider Lúcio de Oliveira, pelas diversas discussões e sugestões durante a elaboração da parte prática deste trabalho.

Aos membros da banca, pelas valiosas correções e sugestões.

À Maria Emília da Silva, pelo apoio e paciência durante a realização deste trabalho.

Enfim, a todos que, direta ou indiretamente, apoiaram-me na realização desta dissertação.

Sumário

Lista de Tabelas

Lista de Figuras

Resumo

Neste trabalho, é proposta uma implementação de um codificador de voz padronizado em DSP. Foi escolhido como padrão de codificação o da recomendação G.723.1 do ITU-T. Esse padrão opera com duas taxa de compressão (5.3 e 6.3 kbits/s) e foi desenvolvido para utilização principalmente em aplicações para Internet. A família de DSPs adotada foi a ADSP-21XX da Analog Devices, que tem como principal característica operação em ponto fixo com precisão de 16 bits e baixo custo. Dentre as principais características da implementação proposta, pode-se destacar a flexibilidade, robustez, eficiência, baixo custo e capacidade de instanciação. Esta solução possui um alto grau de competitividade com outras soluções semelhantes disponíveis no mercado.

Abstract

This work proposes a Digital Signal Processor (DSP) implementation of a standard speech coder. The used standard has been the recommendation G.723.1 of the ITU-T. That recommendation operates in two bit-rates (5.3 and 6.3 kbits/s), which has been developed for Internet applications. The DSP family used is the low cost ADSP-21XX of Analog Devices, which uses a fixed-point 16 bits architecture. Among the main characteristics of the proposed implementation, one can remark: the flexibility, robustness, efficiency, low-cost and instantiation capability. This solution has a high degree of competitiveness as compared with other ones available.

Capítulo 1

Introdução

1.1. Motivação

Desde nossos primórdios a voz tem sido a principal forma de comunicação entre seres humanos. Inicialmente esse tipo de comunicação era possível somente quando os interlocutores estavam juntos. Depois, com a invenção do telefone, tornou-se possível transmitir a voz por longas distâncias.

Com os avanços tecnológicos, a voz, um sinal originalmente na forma analógica, é convertida (digitalizada) para a forma digital. O sinal de voz digital, entre outras características, é mais robusto contra distorções e pode ser processado mais facilmente. Desta maneira, torna-se possível que computadores passem a ser utilizados para transmitir, armazenar, manipular, reconhecer e até sintetizar sinais de voz [1].

O atributo mais importante de uma representação digital de um sinal de voz é a sua taxa de bits, a qual especifica a quantidade de bits que é necessária para compor um segundo de voz. Quando a voz é armazenada ou transmitida, o espaço de armazenamento ou a largura de banda necessária para transmissão são funções diretas

da taxa de bits do formato digital do sinal de voz escolhido. Assim, pode-se assumir que o custo de um sistema de armazenamento ou transmissão de voz também são funções da taxa de bits do sinal digital de voz.

Desta forma, a necessidade de multiplicação de recursos, principalmente referente ao aumento de desempenho dos canais de comunicação, depende exclusivamente da taxa de bits utilizadas na representação do sinal digital de voz. Assim, surgiram os codificadores de voz (*Speech Coders*), desenvolvidos com o objetivo de diminuir a taxa de bits dos sinais digitais de voz. Um sistema de codificação de voz é sempre formado por um codificador e um decodificador. O codificador recebe o sinal digital original de voz e produz um *bitstream* (cadeia ou conjunto de bits que representam os parâmetros do sinal de voz codificado) com baixa taxa de bits. Esse *bitstream* é então utilizado como entrada do decodificador, que produz uma aproximação do sinal original de voz.

Durante alguns anos diversas pesquisas sobre codificação de voz foram realizadas, gerando inúmeros esquemas de codificadores com o objetivo de comprimir a voz. Esses esquemas visavam principalmente três principais atributos: taxa de bits, qualidade de síntese da voz e complexidade computacional. Um aspecto comum a quase todos os esquemas propostos era o compromisso entre estes três atributos. Um aumento de qualidade, mantendo-se a taxa bits, geralmente acarretava um aumento considerável na complexidade computacional, assim sendo, na maior parte dos esquemas, esses atributos relacionavam-se de forma concorrente. Por um longo tempo, a implementação de codificadores de alta qualidade com baixa taxa de bits dependia fundamentalmente de estruturas de *hardware* capazes de suportar suas operações em tempo real.

Por outro lado, com a necessidade dos fabricantes de sistemas de comunicação produzirem equipamentos que "conversem" entre si, padrões para a codificação de sinal voz foram estabelecidos por diversos órgãos de padronização mundiais. Um dos principais órgãos de padronização mundial é o *International Telecommunication Union – Telecommunication Standardization Sector* (ITU-T). Este órgão padronizou uma série de esquemas de codificação de voz para diferentes aplicações, de maneira a atender a necessidade dos fabricantes desses sistemas de telecomunicações.

Geralmente, os padrões que seguem uma recomendação fixam os atributos de qualidade e taxa de bits. O atributo complexidade computacional, por sua vez, depende em parte do projetista do sistema de comunicação e da plataforma de *hardware* disponível para sua implementação. Assim, por algum tempo, a difusão dos sistemas de codificação padronizados ficou atrelada ao desenvolvimento dessas plataformas e do conhecimento do projetista (tanto da plataforma de *hardware* quanto do esquema do codificador a ser implementado), não sendo garantida, para a maior parte desses esquemas, a possibilidade de implementá-lo para operação em tempo real com baixo custo.

Com o advento dos processadores de sinais digital (DSP – *Digital Signal Processor*) de alta performance e baixo custo, foi possível vislumbrar-se a possibilidade da realização de codificação de sinal de voz de alta qualidade e baixa taxa de bits para operação em tempo real.

Entretanto, o sucesso de tais implementações depende do conhecimento do projetista, o qual necessita de um profundo conhecimento tanto da arquitetura da plataforma de *hardware* alvo, quanto dos detalhes do esquema de codificação a ser desenvolvido.

Assim, para este trabalho, propõe-se o estudo e a implementação de um codificador de voz em DSP para operação em tempo real. Foi escolhido o esquema de codificação proposto pela recomendação G.723.1 [2,3] do ITU-T devido à atualidade (estrutura mais evoluída de codificação de voz recomendada) e ao seu grande potencial de utilização para as aplicações de telecomunicações disponíveis atualmente no mercado (VoIP – voz sobre IP, videofone, videoconferência, dentre outras aplicações).

1.2. Contribuições

Esta dissertação resultou em um número importante de contribuições. No caráter individual, este trabalho proporcionou um entendimento profundo da arquitetura e funcionamento dos DSPs da família ADSP-21XX [4], assim como a aprendizagem e compreensão das técnicas de processamentos de sinais envolvidos em um sistema de codificação de voz. Tal aprendizado tornou possível a implementação de outros padrões

de codificação bem como a implementação do padrão desenvolvido para outra família de DSPs.

No aspecto da técnica de implementação, podemos dividir as contribuições deste trabalho em dois grupos: organização do codificador e capacidade de instanciação[1]. Em relação à organização do codificador, foi proposta e utilizada uma técnica que divide o codificador em blocos, permitindo uma maior flexibilidade na utilização desta implementação. Este aspecto juntamente com o aspecto capacidade de instanciação dá a este trabalho a possibilidade de adaptação às próximas gerações de avanços na capacidade de processamento dos DSPs.

Assim, esta implementação do codificador proposto pela recomendação G.723.1 estará sempre "*up-to-date*" às inovações na capacidade de processamento dos DSPs, permitindo que, sem nenhuma alteração de implementação, ocorra um aproveitamento máximo destas plataformas, o que só ocorre devido à existência de um esquema eficiente de instanciação. Isto é, quando surgir um DSP com maior capacidade de processamento, será possível, através do recurso de instanciação, utilizar mais processos de codificação, aproveitando desta maneira todo este aumento de capacidade.

1.3. Estrutura da dissertação

O Capítulo 2 consiste em uma revisão geral dos principais aspectos e técnicas dos codificadores de voz. O Capítulo 3 apresenta uma descrição da atuação dos órgãos de padronização no que se refere aos codificadores de voz, assim como a descrição de alguns desses padrões. O Capítulo 4 descreve de forma sucinta o funcionamento do codificador de voz da recomendação G.723.1. O Capítulo 5 apresenta uma descrição da arquitetura da família de DSPs (ADSP-21xx) utilizado para a realização da implementação proposta neste trabalho. O Capítulo 6 apresenta uma abordagem de solução para o problema de implementação do codificador G.723.1 nos processadores da família ADSP-21XX. Detalhes da organização da implementação como também uma proposta para a definição da estrutura de instanciação são apresentadas e discutidas neste capítulo. No Capítulo 7, são apresentados os resultados obtidos na implementação

[1] Por capacidade de instanciação podemos entender a possibilidade de se utilizar mais de um processo de codificação dentro de um mesmo DSP.

proposta, como também uma comparação com outras implementações disponíveis no mercado. Finalmente, o Capítulo 8 apresenta as conclusões e observações finais desta dissertação.

Capítulo 2

Aspectos de Codificação de Voz

2.1. Introdução

Até 1980, devido ao alto custo de codificação e a baixa qualidade da voz codificada, os codificadores de voz foram muito pouco utilizados. Entretanto, com o grande aumento na eficiência dos *hardwares* de processamento digital de sinais e os recentes avanços das pesquisas em codificação de voz, esse panorama foi significativamente alterado, e atualmente a codificação de voz é utilizada em um grande número de aplicações.

O sinal digital é normalmente gerado por um conversor analógico/digital (A/D) e é representado no formato PCM linear (*Pulse-Code-Modulation*). Desta maneira, para garantir uma boa qualidade de voz para a utilização em telefonia (largura de banda limitada entre 300 e 3400 Hz), o sinal PCM linear deve ser obtido para uma taxa de amostragem de 8 kHz e uma resolução de 16 bits por amostra, o que resulta em uma taxa de bits de 128 kb/s. Essa taxa de bits pode ser considerada como a taxa de bits

padrão para sinais digitais de voz (PCM) não comprimidos para ser usada em sistemas telefônicos.

Na maioria dos codificadores de voz, o sinal de voz original (sem compressão) difere do sinal de voz reconstruído (após compressão). Isso ocorre porque, para reduzir a taxa de bits, é necessário extrair redundâncias do sinal de voz, assim como, reduzir a precisão da representação do sinal de voz ou dos parâmetros em um processo de codificação que geralmente utiliza um modelo analítico para o sinal de fala [1].

A representação de um valor ou de um vetor em uma pré-determinada precisão finita é chamado quantização. A distorção que surge no sinal de voz reconstruído, resultante de um processo de quantização, é chamada de ruído de quantização. Às vezes, o que ocorre é que alguns modelos matemáticos de produção natural do sinal de fala não conseguem reproduzir exatamente o sinal de voz correspondente (mesmo quando seus parâmetros não são quantizados). Tais codificadores sofrem de distorções induzidas pelo modelo. Na literatura de codificação de voz, a distorção induzida pelo modelo é normalmente também tratada como um ruído de quantização.

Na codificação de voz, a figura da distorção é muito importante. Em algumas aplicações, que utilizam codificação de voz, o principal objetivo é que a voz reconstruída pareça natural, enquanto, em outras aplicações, deseja-se a máxima similaridade entre o sinal de voz original e o sinal de voz reconstruído [1].

Um aspecto interessante na codificação de voz é a dificuldade de se definir formalmente uma maneira eficaz de avaliar a qualidade de um determinado codificador, pois este critério deveria ser baseado na percepção de cada indivíduo. A variável subjetiva atua como um parâmetro fundamental no mundo da codificação de voz. Isto pode ser facilmente exemplificado quando imaginamos um sinal PCM linear em que os erros de quantização não dependem da amplitude do sinal de voz. Entretanto, o ruído de quantização é melhor percebido em sinais com amplitude pequena do que em sinais com grandes amplitudes. Sinais com grandes amplitudes "mascaram" o ruído de quantização. Um método simples para aproveitar esse efeito de "mascaramento" é utilizar um processo de quantização em uma escala logarítmica, de forma que o tamanho do passo entre níveis de quantização tornem-se maiores à medida que a amplitude aumenta.

Quantizadores logarítmicos com 8 bits (correspondendo a uma taxa de 64 kb/s) foram utilizados na rede de telefones da Europa, América do Norte e Japão. Esses algoritmos de quantização faziam parte diretamente dos processos de conversão A/D e D/A. Essa tecnologia foi padronizada em 1972 como o padrão G.711 no CCITT (atual ITU-T) . Alguns anos mais tarde, o CVSD [5,6] (Continously-Variable-Slope Delta modulation) foi introduzido para utilização em comunicação militar. Da mesma forma que o G.711, o CVSD era também implementado em *hardware*, diretamente nos processos de conversão analógica para *bitstream* e vice-versa [7].

Em 1980, novas tecnologias de *hardware* levaram à separação dos conversores A/D e D/A dos processos de codificação. Assim, o sinal passou a ser inicialmente convertido em um sinal PCM linear com 128 kb/s e em seguida codificado para uma menor taxa de bits por um algoritmo de compressão de voz [7,8].

Uma das primeiras aplicações na qual se utiliza esta abordagem foi em comunicação segura através da rede de telefonia analógica. Para essa aplicação, o sinal de voz era inicialmente convertido em um *bitstream* digital, o qual era criptografado e então transmitido na linha telefônica através de um modem. Para permitir uma larga abrangência desse sistema com as tecnologias de modems existentes, foi introduzido um padrão de codificação com taxa de 2.4 kb/s (FS1015) [9]. Devido à grande importância desse sistema e dado o aspecto de comunicação segura, uma baixa qualidade de voz e um custo altíssimo foram aceitos na época, viabilizando assim a implantação do sistema. Codificação de voz com custos baixos só era possível com taxa de bits altas. Em 1983, o CCITT (ITU-T) definiu um padrão de codificação a 32 kb/s, que foi padronizado como G.721 [10]. Esse padrão permitiu uma alta qualidade de voz a um baixo custo.

Os padrões de codificação FS1015 e G.721 são exemplos de padrões definidos para serem largamente utilizados e permitirem a interoperabilidade de equipamento nos dois extremos dos sistemas de codificação. Se a codificação for realizada com o objetivo de armazenamento do sinal de voz em dispositivos locais, não é necessário a utilização de codificadores padrões. Como resultado, sistemas de armazenamento de

sinal de voz normalmente utilizam sistemas de codificação proprietários[1], que podem ser freqüentemente atualizados [7,8].

2.2. Atributos dos Codificadores de Voz

Cada codificador de voz é desenvolvido para uma aplicação em particular. Assim cada codificador possui características ou atributos que se enquadram melhor para uma determinada aplicação. Dentre os atributos de um codificador, podemos destacar:

- taxa de bits
- qualidade de voz (subjetiva)
- complexidade (processamento e memória)
- atraso
- sensibilidade a erros de canal
- largura de banda do sinal

A seguir será brevemente descrito cada um destes atributos. Existem outros atributos que não foram mencionados, que podem ser importantes para uma determinada classe de aplicações, dentre eles estão: capacidade do codificador em transmitir sinais que não sejam de voz (ex.: dados), capacidade de transmissão de sinalização de linha e tons de discagem e o suporte para reconhecimento de voz e identificação de locutor [7,11].

2.2.1. Taxa de Bits

A redução da taxa de bits do *bitstream* é a maior motivação para utilização de codificadores de voz. Dependendo do sistema e aspectos de projeto, são utilizados codificadores com taxas de bits fixas ou variáveis.

A maior parte dos padrões de codificação de voz descrevem codificadores com taxa de bits fixa. Codificadores com taxa de bits fixa são mais simples de projetar do que os com taxa variável, pois não necessitam de mecanismos nem critérios que determinem qual taxa eles devem utilizar em um dado intervalo de tempo. Para

[1] O termo codificação de voz proprietária se aplica a sistemas de codificação não padronizados, utilizados normalmente em aplicações locais (ex.: armazenamento).

aplicações em que esteja disponível um único canal de transmissão com banda constante, a taxa de bits não pode ultrapassar um determinado valor. Sendo assim, um codificador de voz em sua taxa máxima (desde que suportada pelo canal de comunicação) pode ser a melhor solução. Codificadores com taxa fixa em aplicações de comunicação segura geralmente possuem taxas de bits baixas: 0.8 a 4.8 kb/s. Codificadores usados para telefonia via satélite e celular possuem taxas que variam em torno de 3.3 a 13 kb/s. Codificadores que se propõem à utilização em redes telefônicas geralmente possuem taxas de 16 kb/s ou maiores.

Codificação de voz utilizando taxa variável de bits, por sua vez, é importante para diversas aplicações. Por exemplo, na telefonia móvel utilizando o esquema CDMA (*Code-Division-Multiple-Access*) [12], a taxa de bits de cada usuário pode variar independentemente do usuário. Aplicações que não necessitem de codificação em tempo real (armazenamento de sinal de voz), também são fortes candidatas a utilizarem codificadores com taxa variável de bits. O sistema mais simples que utiliza taxa de bits variável é o dos codificadores de dois estados (ou com detecção de atividade de voz). Esses codificadores possuem dois estados de atuação: um para silêncio e ruídos de fundo e outro para quando a voz está presente. Outros codificadores utilizam mais estados de atuação, que são alterados conforme fatores externos ao codificador (carga da rede, espaço de disco remanescente, etc.) [7].

A Tabela 2.1, apresentada a seguir, mostra a taxa de bits para alguns codificadores padronizados [7,11].

Tabela 2.1: Taxa de bits de codificadores de voz padronizados

Padrão	Taxa	Tipo de Taxa
Recomendações do ITU		
G.711 PCM	64 kb/s	Fixa
G.726, G727 ADPCM	16, 24, 32, 40 kb/s	Fixa
G.722 Wideband Coder	48, 56, 64 kb/s	Fixa
G.722.1	24, 32 kb/s	Fixa
G.728 LD-CELP	16 kb/s	Fixa
G.729 CS-ACELP	8 kb/s	Fixa
G.729 CS-ACELP Anexo A	8 kb/s	Fixa
G.729 CS-ACELP Anexo B	8 / 3.2 / 0 Kb/s	Variável
G.723.1 MPC-MLQ	5.3 e 6.4 kb/s	Fixa
G.723.1 Anexo A	5.3 e 6.4 / 1.1 / 0 Kb/s	Variável
Padrões de Celulares		
RPE-LTP (GSM)	13 kb/s	Fixa
IS-54 VSELP (TIA)	7.95 kb/s	Fixa
PDC VSELP (RCR Jap)	6.7 kb/s	Fixa
IS-96 QCELP (TIA)	8.5 / 4 / 2 / 0.8 kb/s	Variável
PDC PSI-CELP	3.45 kb/s	Fixa
U. S. DoD telefonia segura		
FS-1015 LPC-10E	2.4 kb/s	Fixa
FS-1016 CELP	4.8 kb/s	Fixa
MELP	2.4 kb/s	Fixa

2.2.2. Qualidade de Voz (Subjetiva)

A qualidade do sinal de voz reconstruído é também um atributo muito importante de um codificador de voz. Entretanto, esse atributo é problemático, pois a avaliação da qualidade de um sinal de voz é um problema extremamente complexo. Até o presente momento, não foi encontrado nenhum critério objetivo que possa ser

completamente equivalente a critérios perceptuais para avaliar a qualidade de diversos codificadores de voz diferentes [13].

Com a diminuição da taxa de bits, a qualidade do sinal de voz reconstruído torna-se cada vez mais dependente das características do sinal de entrada, tornando muito complicado definir o comportamento deste codificador quando utilizado em aplicações no "mundo real". Assim, muitos testes subjetivos realizados por humanos são necessários para determinar se um codificador é adequado ou não para ser utilizado em aplicações reais. Múltiplas codificações ("*tandeming*"), robustez a ruídos de fundo, suporte de sinais de áudio (ex.: música) são normalmente incluídos nesses testes [14].

Três medidas são normalmente usadas para determinar a qualidade subjetiva de um codificador de voz: o DRT (*Diagnosgtic Rhyme Test*), o DAM (*Diagnostic Acceptability Measure*) e o MOS (*Mean Opinion Score*) [7].

O DRT fornece medidas de inteligibilidade, já o DAM fornece uma caracterização do codificador de voz em relação à distorção. O MOS, por sua vez, é a reunião de todos os aspectos de qualidade em um único número e é, provavelmente, a medida subjetiva mais utilizada para determinar a qualidade de um codificador. O MOS é calculado a partir de resultados obtidos em testes de *category-rating*, realizados por 20 a 60 ouvintes destreinados. Os ouvintes classificam um conjunto de sinais com um valor em uma escala de 1 (qualidade não aceitável) a 5 (qualidade excelente). Um MOS de valor 4.0 ou mais define que o sinal de voz tem uma boa qualidade, o que ocorre normalmente quando o sinal reconstruído não consegue ser distinguido do sinal original. Valores MOS entre 3.5 e 4.0 definem que o sinal reconstruído tem qualidade suficiente para comunicação natural, sendo, dessa forma, adequado à telefonia. Um MOS com valor inferior a 3.0 indica que o sinal de voz reconstruído é inteligível, entretanto pode perder a naturalidade e pode-se tornar impossível o reconhecimento do falante. A Tabela 2.2 apresenta os valores MOS para alguns codificadores padronizados e a Figura 2.1 um gráfico comparativo da qualidade *versus* a taxa de bits [11].

Tabela 2.2: Avaliação de qualidade utilizando MOS para alguns codificadores padronizados

Padrão	MOS
Recomendações do ITU	
G.711 PCM	4.10
G.726	3.85
G.728 LD-CELP	3.61
G.729 CS-ACELP	3.92
G.729 CS-ACELP Anexo A	3.87
G.729 CS-ACELP Anexo B	3.88
G.723.1 MPC-MLQ	3.89
G.723.1 Anexo A	3.85
Padrões de Celulares	
RPE-LTP (GSM)	3.50
IS-54 VSELP (TIA)	3.54
PDC VSELP (RCR Jap)	-
IS-96 QCELP (TIA)	3.52
PDC PSI-CELP	3.49
U. S. DoD telefonia Segura	
FS-1015 LPC-10E	1.5
FS-1016 CELP	3.3
MELP	3.42

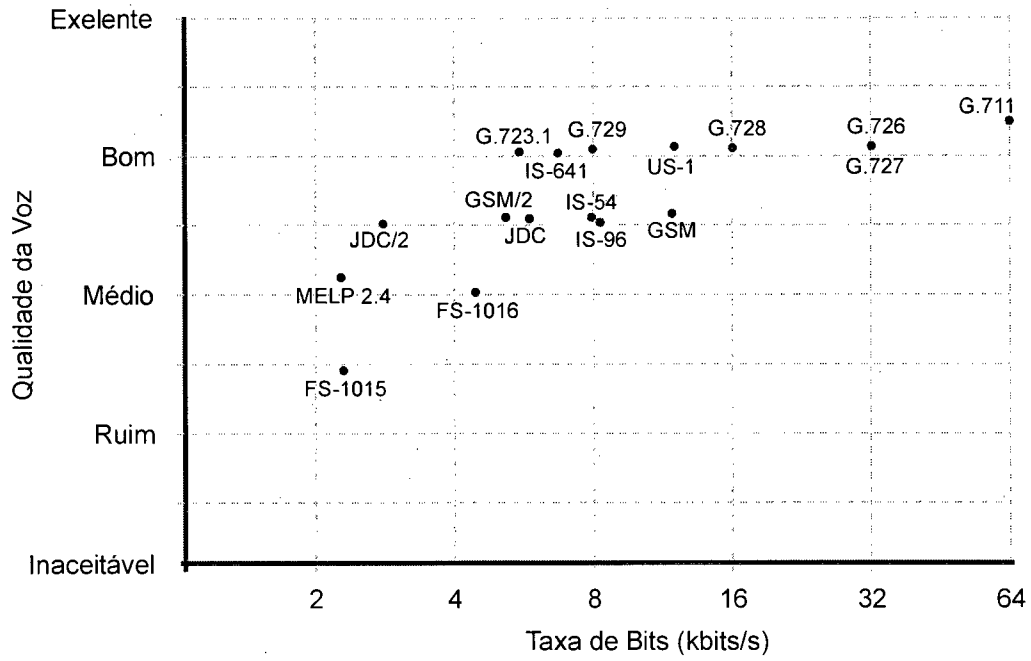

Figura 2.1: Comparação da Qualidade de Voz versus a Taxa de Bits (Adaptado de [11] p. 45)

2.2.3. Complexidade

A complexidade computacional e a necessidade de memória de um codificador de voz determinam o custo e o consumo de energia do *hardware* no qual este codificador será implementado. Exceto para algumas aplicações, o codificador de voz necessita ser operado em tempo real, o que significa que o processamento de um quadro de voz precisa ser realizado em um tempo igual ou inferior a sua duração. Paralelamente, o *bitstream,* gerado da codificação, precisa ser decodificado em tempo real para quase todas as aplicações comerciais [7,11].

Para manter o custo e o consumo de energia baixos, algoritmos de codificação de voz normalmente são implementados em processadores de sinal digital (DSP - *Digital Signal Processor*). É importante também que as exigências computacionais de um determinado algoritmo possam ser atendidas por um único DSP. Para aplicações comerciais de grande escala (telefones móveis, máquinas de respostas), nas quais cada usuário necessita de um dispositivo de codificação de voz separado são utilizados DSPs ponto-fixo com 16 bits (*chips* que possuem custo geralmente baixo) [7].

A utilização de DSPs de ponto fixo com 16 bits, entretanto, tem uma desvantagem, que é a sua dificuldade de programação. Por essa razão, DSPs com ponto

flutuante de 32 bits são geralmente utilizados para o desenvolvimento de sistemas que demandem operações em tempo real. DSPs de ponto flutuante são também algumas vezes utilizados em aplicações nas quais vários usuários precisam compartilhar a mesma plataforma de *hardware* (ex.: sistema de armazenamento central), e para o qual o custo da unidade é menos importante do que o custo de desenvolvimento. Aplicações de codificação de voz também são implementadas em plataformas de *hardware* de propósito geral (ex.: PCs), que possuem normalmente processamento em ponto flutuante.

A capacidade de processamento e de armazenamento dos DSPs tem crescido muito nos últimos anos. Isso tem ocorrido devido a avanços nas tecnologias de circuitos integrados [15]. Um codificador de voz que necessite hoje de um *hardware* com custo alto, pode, daqui a cinco anos, apresentar um *hardware* de custo bastante baixo. Em 1980, por exemplo, um DSP de ponto fixo podia realizar cerca de 1 milhão de operações por segundo (1 MIPS) e possuía cerca de 128 *words* (palavras de 16 bits) de memória RAM (*Random Access Memory*) e 1 *kword* de memória ROM (*Read Only Memory*). Atualmente temos DSPs com capacidade de processamento de dezenas de MIPS, centenas de *kwords* de memória, com custo relativamente muito baixo e consumindo muito menos energia.

Podemos observar, na Tabela 2.3, o avanço na capacidade de processamento das famílias de DSPs fabricados pela empresa Analog Devices.

Tabela 2.3: Avanço da capacidade de processamento dos DSPs

Família de DSP's	Período de Atuação	Capacidade de Processamento
ADSP2101 – ADSP217X	1986 – 1994	20 – 33 MIPS
ADSP218X	1994 – 1998	33 – 75 MIPS
ADSP219X	Lançada em 1999	Mais de 300 MIPS

Para implementações que compartilham o mesmo DSP para outras funções, a memória RAM pode ser classificada como estática e automática. Por exemplo, um codificador de voz e um modem podem compartilhar o mesmo DSP, e o DSP pode estar realizando, em um determinado momento, qualquer uma das duas funções. Nesse caso,

a memória estática guarda as variáveis dinâmicas do sistema (ex.: memória do filtro) as quais ficarão reservadas para o codificador de voz permanentemente. Já, a memória automática poderá ser sobre-escrita sempre que for necessário executar uma ou outra função no DSP, servindo assim como uma memória de rascunho.

Na Tabela 2.4, pode-se observar a capacidade de processamento e a quantidade de memória exigidas por alguns codificadores de voz. Para a elaboração dessa tabela, foram levados em conta informações de projetistas de *softwares* de codificação de voz [7,11].

Tabela 2.4: Capacidade de processamento e memória exigida por algoritmos de codificação de Voz

Padrão	Capacidade de Processamento	Carga de Memória (Programa/Código)
Recomendações do ITU		
G727 ADPCM	5,5 MIPS	300 / 250 words
G.728 LD-CELP	30,0 MIPS	
G.729 CS-ACELP	20 MIPS	10.8 / 5.4 kwords
G.729 CS-ACELP Anexo A	9,2 MIPS	7,6 / 4.8 kwords
G.729 CS-ACELP Anexo AB	9,7 MIPS	10,7 / 5.4 kwords
G.723.1 MPC-MLQ (5.3 / 6.3 kb/s)	21,6 / 23,3 MIPS	9,9 / 11,7 kwords
G.723.1 Anexo A (5.3 / 6.3 kb/s)	21,8 / 23,5 MIPS	10.1 / 12,7 kwords

2.2.4. Atraso

O atraso é um atributo que tem grande importância quando se utiliza o codificador de voz para comunicação bilateral. Existem dois limiares para atrasos de codificação. O limiar mais básico de um atraso é aquele que começa a afetar a dinâmica de uma conversação. Por exemplo, atrasos maiores do que 150 ms são percebidos em conversações altamente interativas. Entretanto, atrasos unilaterais na faixa de 500 ms são suportáveis em diversas situações sem prejuízos significativos na naturalidade da conversação [16]. O segundo limiar aplica-se somente quando a presença de um *cancelador* de *eco* não está garantida no sistema. Ecos de rede podem se tornar perceptíveis mesmo quando o atraso é inferior a 100 ms. Por essa razão, algumas

implementações de codificadores possuem estritas limitações quanto ao atraso, mesmo que sejam utilizadas em aplicações nas quais um *cancelador de eco* esteja presente [7].

O atraso de um codificador é normalmente dividido em quatro componentes: atraso do algoritmo, atraso de processamento, atraso de multiplexação e atraso de transmissão. O atraso do algoritmo ocorre porque a maioria dos codificadores opera codificando quadros de sinal de voz (*frames*). Assim, um quadro de voz precisa ser formado antes de se iniciar a codificação. Além disso, os algoritmos de codificação costumam ter *look-ahead*, ou seja, usam segmentos do sinal de voz posteriores ao quadro que está sendo codificado (Figura 2.2). Desta maneira, a soma do tamanho do quadro com o *look-ahead* formam o atraso referente ao "atraso do algoritmo" [7,11].

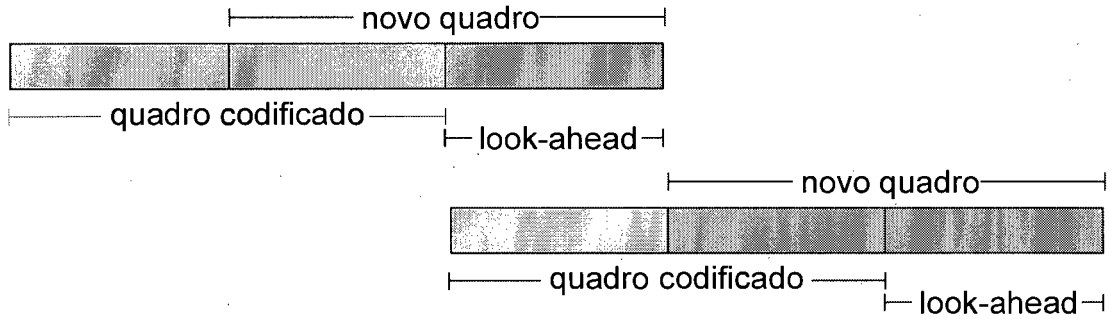

Figura 2.2: Componentes de atraso do algoritmo

A Tabela 2.5 ilustra os atrasos de algoritmos para alguns codificadores padronizados.

Tabela 2.5: Atrasos de algoritmo para alguns padrões de codificação

Padrão	Tamanho do Frame	Look-ahead	Atraso de Algoritmo
Recomendações do ITU			
G.711 PCM	0,125 ms	0 ms	0,125 ms
G.726, G727 ADPCM	0,125 ms	0 ms	0,125 ms
G.722 Wideband Coder	0,125 ms	1,5 ms	1,625 ms
G.728 LD-CELP	0,625 ms	0 ms	0,625 ms
G.729 CS-ACELP	10,0 ms	5,0 ms	15,0 ms
G.723.1 MPC-MLQ	30,0 ms	7,5 ms	37,5 ms
Padrões de Celulares			
RPE-LTP (GSM)	20 ms	0 ms	20 ms
IS-54 VSELP (TIA)	20 ms	5 ms	25 ms
PDC VSELP (RCR Jap)	20 ms	5 ms	25 ms
IS-96 QCELP (TIA)	20 ms	5 ms	25 ms
PDC PSI-CELP	40 ms	10 ms	50 ms
U. S. DoD telefonia segura			
FS-1015 LPC-10E	22.5 ms	90 ms	112,5 ms
FS-1016 CELP	30 ms	7.5 ms	37,5 ms
MELP	22.5 ms	23 ms	45,5 ms

O atraso computacional representa o atraso gerado pelo tempo necessário para codificar/decodificar um quadro de voz. Esse atraso é normalmente igual ou inferior ao tamanho do quadro.

O atraso de multiplexação ocorre em diversos sistemas de transmissão em que o *bitstream* gerado pelo codificador precisa ser empacotado para ser transmitido e, quando recebido, precisa ser desempacotado antes de ser repassado ao decodificador. Esse tempo de empacotamento e desempacotamento é que dá origem ao atraso de multiplexação.

Finalmente, o atraso de transmissão é gerado pelo tempo necessário para transmitir um pacote de um lado ao outro de uma comunicação. Em canais dedicados,

ou de alta velocidade, esse atraso somado ao de multiplexação é normalmente igual ao tamanho do quadro. Entretanto, em canais de comunicação compartilhados, esse atraso pode-se tornar muito grande.

Com base nestes componentes, pode-se estimar o atraso total de comunicação em um sentido, baseando-se somente no tamanho do quadro. O atraso do algoritmo é de, pelo menos, a duração de um quadro. O atraso computacional é normalmente também de um quadro e os atrasos de multiplexação e transmissão somam ainda mais um quadro. Sendo assim, pode-se dizer que o atraso total de um sistema de codificação é de, pelo menos, três quadros. Deve-se lembrar que em diversos sistemas de codificação, dependendo do algoritmo utilizado, ainda somam a este atraso o *look-ahead.*

A relação entre o tamanho do quadro e o atraso do sistema pode ser ilustrada comparando-se dois codificadores com taxas de bits semelhantes. O ITU-T G.728 (16 kb/s), projetado para situações em que o eco nem sempre é cancelado possui um quadro de 0.625 ms. Em contrapartida, o GSM 13 kb/s [17], projetado para situações em que o eco é normalmente cancelado, usa um quadro de 20 ms.

2.2.5. Sensibilidade a erros de canal

Em muitas aplicações, o *bitstream* recebido após ser transmitido chega com erros. Entretanto, é muito importante que a voz reconstruída não seja afetada por esses erros de canal. Os tipos de erros de canal os suportados pelos codificadores de voz suportem são divididos em duas classes: erros aleatórios e erros repentinos (*burst*). Essa última classe de erros é tipicamente encontrada nos ambientes de telefonia móvel. Estas duas classes de erros requerem diferentes estratégias para reduzir seus impactos sobre o sinal de voz reconstruído [7].

2.2.6. Largura de Banda do Sinal

Um sinal de voz pode ser limitado em cerca de 10 kHz sem afetar a sua percepção [18]. Entretanto, em telecomunicações, a largura de banda do sinal de voz é normalmente muito mais limitada. Sistemas de telefonia geralmente limitam a largura de banda do sinal de voz entre 300 e 3400 Hz. Essa limitação de banda resulta no som

característico dos telefones. Tanto o limite inferior de 300 Hz quanto o limite superior de 3400Hz afetam a qualidade da voz nesse sistema.

Na maioria dos codificadores de voz digitais, a voz é amostrada em 8 kHz, resultando em uma largura de banda máxima, baseada no teorema de amostragem, de 4 kHz. Na prática, o sinal tem usualmente sua banda limitada em 3600 kHz. Alternativamente, o ITU-T padronizou um codificador de voz de banda larga (G.722 com largura de banda de 7 kHz e 64, 56 e 48 kb/s). Codificadores de voz de banda larga ainda não são muito comuns, pois a maioria dos sistemas de telefonia trabalha em banda estreita.

Observa-se também que, quando utilizado codificadores de banda larga para codificar a voz que será em algum momento transmitida em banda estreita, a qualidade perceptual da voz nesses casos se torna inferior a dos codificadores de banda estreita para a mesma taxa de bits.

Atualmente, tem-se dado mais atenção aos codificadores de banda larga devido às recentes aplicações de videofone e videoconferência, para as quais os sinais de voz nunca precisarão trafegar sobre conexões de banda estreita [7].

2.3. Classificação dos procedimentos de codificação de Voz

Tradicionalmente, os codificadores de voz são separados em duas classes: codificadores por aproximação de forma de onda e codificadores paramétricos. Pode-se dizer, ainda, que existe uma terceira classe de codificadores chamados "híbridos". Nessa terceira classe, encontram-se os codificadores que possuem características de funcionamento mistas das duas classes originais.

O comportamento típico da qualidade *versus* taxa de bits dos codificadores por aproximação de forma de onda e paramétrico pode ser observado na Figura 2.3. Os codificadores por aproximação de forma de onda convergem para a qualidade do sinal de voz original (normalmente recebem notas de MOS entre bom e excelente), enquanto os codificadores paramétricos ficam limitados à qualidade do modelo de fala que utilizam (esta qualidade nos codificadores paramétricos atuais é muito boa) [7,8].

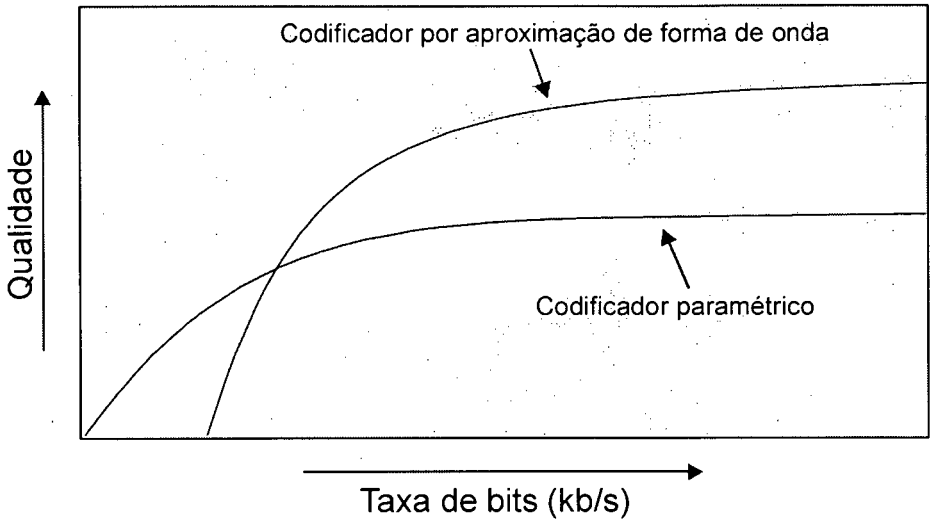

Figura 2.3: Comportamento qualidade *versus* taxa de bits dos codificadores por aproximação de forma de onda e paramétricos (Adaptado de [7] p. 26).

2.3.1. Codificadores por aproximação de forma de onda

Os codificadores por aproximação de forma de onda (codificadores de forma de onda) têm por objetivo minimizar algum critério de medida de similaridade entre o sinal de voz reconstruído e o sinal de voz original. Nos codificadores de forma de onda mais básicos, esse critério é normalmente a medida do erro médio quadrático entre o sinal de voz original e o sinal de voz reconstruído. O erro médio quadrático, nesse caso, é calculado para blocos (períodos) do sinal que, no limite, podem ser constituídos por apenas uma amostra.

Entretanto, critérios, como erro médio quadrático, que não consideram os fenômenos de mascaramento inerentes ao sistema auditivo humano, são pouco utilizados nos codificadores de forma de onda atuais. A introdução de uma filtragem que considere os efeitos de máscara, antes de realizar qualquer minimização por algum critério, pode aumentar significativamente o desempenho da codificação. Atualmente, nos codificadores de voz mais recentes, a propriedade do ouvido humano não perceber alguns tipos de alterações do sinal é utilizada [19].

Em todos os codificadores de forma de onda, a diferença entre os dois sinais são minimizadas bloco a bloco. Outra característica dessa classe de codificadores é que, à

medida que se aumenta a taxa de bits, o sinal reconstruído aproxima-se mais do sinal original. Por preservarem a forma de onda original, as medidas de SNR (Razão Sinal-Ruído) dessa classe de codificadores são geralmente positivas quando expressadas em dB [7].

2.3.2. Codificadores Paramétricos

Nos codificadores paramétricos, o sinal de voz é caracterizado através de um conjunto de parâmetros de um modelo. Esses parâmetros normalmente são quantizados sem levar em consideração o sinal de voz original. Nesses codificadores, a SNR é pequena, normalmente negativa (quando expressa em dB), e não tem correlação com a qualidade do sinal reconstruído.

É importante notar que nem todos os codificadores que utilizam modelos para gerar o sinal de voz reconstruído são necessariamente codificadores paramétricos. Os parâmetros do modelo para codificadores de forma de onda são quantizados de forma a maximizar a similaridade entre o sinal de voz original e o reconstruído, o que não é o caso dos codificadores paramétricos. Desta maneira, a qualidade de voz obtida em um codificador paramétrico está sempre limitada à capacidade do modelo [19].

Parece natural e muito comum basear os modelos dos codificadores paramétricos no sistema de produção de voz humana. Assim, são normalmente identificadas estruturas que emulem o trato vocal, assim como estruturas que modelem a onda de pressão gerada pelas nossas cordas vocais. Entretanto, na prática, a importância dada à exata modelagem do sistema de produção de voz humana é menor do que a importância dada a outros atributos dos codificadores de voz, como complexidade computacional e qualidade da voz.

Capítulo 3

Padrões de Codificação de Voz

3.1. Introdução

A década de 80 foi marcada por um grande aumento no aparecimento das tecnologias de codificação de voz. Dentre os fatores responsáveis por esse aumento, destacam-se o amadurecimento das tecnologias de codificação de voz e o aumento da demanda por novas tecnologias de comunicação. Dessa forma, é apresentado neste capítulo, alguns órgãos de padronização, o funcionamento dos processos de padronização e a forma de avaliação de desempenho dos codificadores.

A padronização é motivada principalmente pela necessidade de existência de formas comuns de comunicação, necessidade que tem chamado a atenção de muitos fabricantes, provedores e usuários de serviços de comunicação. Por hipótese, vamos imaginar uma nova aplicação na qual cada fabricante possua um padrão proprietário de comunicação. Isso provocaria uma incompatibilidade entre os produtos de todos os fabricantes. Por outro lado, vamos imaginar que uma determinada aplicação comece a

ter uma grande demanda no mercado, e que vários fabricantes estejam interessados em implementar tal aplicação. Isso, irá levar à necessidade de se estabelecer um padrão para comunicação entre os diversos produtos, de diferentes fabricantes. Essa necessidade pode então levar ao aparecimento de um novo processo de padronização, que será conduzido por um órgão de padronização (ISO, ITU-T, ...) [7].

Diversos órgãos de padronização são responsáveis pela definição dos novos padrões de codificação. O ITU (*International Telecommunications Union*) é parte da UNESCO (*United Nations Economic, Scientific and Cultural Organization*) e é responsável pelo conjunto de padrões utilizados em comunicação no mundo. Originalmente o ITU era formado pelo CCITT e o CCIR. O CCITT atuava na área de padrões para telecomunicações, incluindo padrões de codificação de voz, e o CCIR atuava na área de padrões para rádio. Em 1993, o ITU foi reorganizado e o CCITT formou o ITU-T (ITU *Telecommunication Standards Sector*). O ITU-T possui grupos de estudo (*Study Groups*), responsáveis por partes dos processos de padronizações. O grupo de estudo 15 (*Study Group* 15 – SG15), por exemplo, é responsável pela formulação dos padrões para codificação de voz, o SG12 é responsável pela avaliação e caracterização do desempenho de codificadores na rede e juntamente com o SG15 atua nos testes dos codificadores de voz. Outros grupos de estudo, tanto do ITU-T quanto do ITU-R (ITU Radio Standartization Sector – anteriormente CCIR), fazem solicitações de novos padrões de codificação que atendam às necessidades de suas novas aplicações [2,3,20-27].

Padrões de telefonia celular, por outro lado, são padronizados por órgãos de padronização regionais. Esses órgãos são responsáveis pela definição do sistema celular completo, no qual a codificação de voz é uma pequena parte, entretanto vital. Na Europa, o TCH-HS, que faz parte do ETSI (*European Telecommunications Standards Institute*), é atualmente responsável pela definição do padrão da telefonia digital celular na Europa. Na América do Norte, essa função é realizada pelo TIA (*Telecommunications Industry Association*). Assim como nestas regiões, diversas outras regiões possuem seu próprio órgão de padronização [7].

Existem outros órgãos que desenvolvem padrões para aplicações mais específicas. O Inmarsat (*International Maritime Satellie Corporation)*, por exemplo, é

responsável pela regulamentação de comunicação em satélites geoestacionários e possui diversos padrões para comunicação telefônica através de satélites. Padrões também podem ser criados por instituições, governamentais ou não, para qualquer tipo de regulamentação dentro de um país. Os Estados Unidos, assim como algumas outras nações ou conjunto delas, criaram o seu próprio padrão de codificação de voz para ser aplicado em telefonia segura.

Não se pode deixar de observar que apesar dos codificadores de voz serem padronizados, estes não estão disponíveis para utilização por qualquer pessoa gratuitamente. Codificadores de voz e padrões de codificação de voz são desenvolvidos através do trabalho de diversas pessoas, e, desta forma, a necessidade de licença ou do pagamento de *royalties* é a forma utilizada para recompensar o trabalho dos pesquisadores envolvidos nesse processo de criação [7].

3.2. Atributos dos codificadores de voz

No capítulo anterior, foram descritos os atributos dos codificadores de voz. Nesta seção, pretende-se complementar as informações apresentadas.

Codificadores de voz possuem atributos que podem ser divididos basicamente em quatro grupos: taxa de bits, qualidade, complexidade e atraso. Para uma determinada aplicação, alguns atributos são pré-determinados, enquanto relações de compromisso têm de se manter para os demais. Por exemplo, o canal de comunicação pode determinar um limite para a taxa de bits, ou o custo pode determinar um limiar de complexidade. A qualidade, por sua vez, está relacionada com a taxa de bits e a complexidade, e algumas vezes, com o atraso. Assim, antes de se estabelecer um padrão, é necessário se estabelecer metas e necessidades para cada um destes atributos. Em processos de padronização, pode-se ainda destacar mais dois atributos muito importantes: o método de especificação e validação de conformidade, e a agenda de trabalho. A seguir será apresentada a descrição desses atributos, de forma a complementar o que foi descrito no capítulo anterior [7,8].

3.2.1. Taxa de Bits

A largura de banda do sinal de voz em comunicações telefônicas está limitada entre aproximadamente 300 e 3400 Hz, e a voz é normalmente amostrada com uma

freqüência de 8kHz. Os codificadores de voz padronizados nos últimos anos, utilizam esse padrão de sinal operando com taxas de bits entre 800 bits/s e 16 kbits/s. Alguns desses codificadores, principalmente aqueles padronizados para telefonia celular, possuem também um codificador de canal associado, o que eleva suas taxas de bits a 22.8 kbits/s. As menores taxas, 800 bits/s a 4800 bits/s, estão associadas inicialmente a aplicações de telefonia segura e atualmente começam a ser utilizadas também em telefonia baseada em satélites. As taxas de bits nos codificadores utilizados em telefonia celular variam entre 3.3 kbits/s e 13 kbits/s.

Codificadores de voz de banda larga (50 a 7000 Hz – amostrados em 16 kHz), os que fornecem maior inteligibilidade e menor cansaço ao ouvinte durante longas conversações, operam em taxas a partir de 24 kbits/s [7,11].

3.2.2. Atraso

Codificadores com baixa taxa de bits podem ser considerados codificadores de blocos (também chamados de quadros – *frames*), pois codificam a voz quadro a quadro. Dessa forma, dependendo da aplicação, o atraso total do sistema de codificação é um múltiplo do tamanho do quadro. Pode-se considerar que neste tipo de sistema o atraso mínimo é de 3 a 4 vezes a duração do seu quadro. Por exemplo, diversos codificadores com taxas de bits baixa possuem quadros de 20 ms, resultando em um atraso, em um dos sentidos, de aproximadamente 60 a 80 ms. A origem desse atraso está descrita no Capítulo anterior.

Quando se deseja estabelecer um padrão de codificação, o atraso é um fator muito importante em sistemas em que se pretende usar o codificador para conversação em tempo real. Pesquisas mostraram que, se o atraso é muito grande, os usuários do sistema irão perceber, o que é verdade mesmo não havendo presença de eco [11].

3.2.3. Complexidade

Muitos codificadores de voz são implementados inicialmente em DSPs e posteriormente em dispositivos VLSI (*Very Large Scale Integration*) de propósito específico. Assim, velocidade e memória (RAM) são dois importantes componentes na avaliação da complexidade destes codificadores, pois quanto mais rápido e mais genérico for o *hardware*, maior será o custo de desenvolvimento. Estes componentes

ainda incidem sobre outro fator muito importante dentro do projeto de um sistema, o consumo de potência, que é crítico para diversas aplicações. Desta maneira, observa-se que a complexidade é um atributo que influencia no custo e no consumo de potência do *hardware* utilizado para sua implementação [7].

3.2.4. Qualidade

O primeiro padrão de codificação totalmente digital foi proposto pela recomendação CCITT G.711 com taxa de 64 kbits/s [20], esse padrão introduz uma distorção de 1 QDU (*Quantization Distortion Unit*).

O segundo padrão digital de codificação de voz foi o G.721 32 kb/s ADPCM [21]. Esse padrão foi desenvolvido para ser utilizado em conjunto com o G.711, de forma que sua entrada e saída estivessem e fossem, respectivamente, codificadas por ele. Foi observado que essa combinação introduz uma distorção de 3.5 QDU. O grupo de qualidade de voz do ITU (*ITU – SQEG - Speech Quality Experts Group*) determina que em comunicações internacionais ponta-a-ponta, deve haver no máximo 14 QDU e que, em comunicações domésticas, este valor não pode ultrapassar 4 QDU. O padrão de codificação G.728 16 kb/s LD-CELP introduz a mesma quantidade de distorção do G.721. Esses padrões de codificação são os únicos padrões do ITU considerados pelo SQEG "*toll quality*" [13,14].

Os padrões de telefonia celular digital foram criados por órgãos regionais. As primeiras gerações dos padrões de codificação para telefonia celular digital eram bastante boas. Os usuários não podiam dizer que a voz era tão clara e boa quanto a original de 64kbits/s, ou as codificadas pelos padrões das recomendações G.721 e G.728, entretanto, levando-se em consideração o tipo de serviço (telefonia celular), estes padrões foram considerados adequados, e os seus usuários podiam manter uma comunicação por um longo período de tempo [7].

Codificadores de voz com menores taxas de bits foram criados para outras aplicações. A voz nesses codificadores é inteligível, entretanto possui uma quantidade de distorção suficiente para que ocorra a fatiga do ouvinte em comunicações que durem muito tempo. A qualidade pobre desses codificadores foi constatada em testes subjetivos.

Codificadores com baixa taxa de bits têm seu desempenho degradado quando a sua entrada possui outros tipos de sinais (ruídos ou músicas de fundo), juntos com o de voz. Isso ocorre porque os codificadores de baixa taxa realizam sua compressão baseados em modelos de produção de voz, que, na maior parte das vezes, não servem para sinais de outra natureza. Como resultado, estes ruídos de fundo são transformados em sons que não soam naturais. Esses sons podem não influenciar muito, assim como podem tornar o codificador não utilizável.

Além da clareza, existem outros componentes que afetam a qualidade de um codificador, por exemplo a sensibilidade a erros de canal (apresentado como um atributo no capítulo anterior). Nos padrões para telefonia digital celular, são destinados alguns bits extras para serem usados como código de canal, a fim de manter a integridade das informações transmitidas. Além disso, nem todos os bits do *stream* de voz codificada são igualmente sensíveis a alterações. Desta maneira, é uma prática comum nos projetos de codificadores definir duas ou três classes de bits, as quais recebem proteção de acordo com sua importância. Outro componente de qualidade é a capacidade do codificador ignorar e regerar um quadro quando este estiver danificado.

Com o propósito de economizar banda, detetores de atividade de voz são também utilizados pelos codificadores. O funcionamento desses detetores está baseado na interrupção do *bitstream* de um dos interlocutores nos períodos em que o interlocutor está em silêncio. Neste momento, o "recebedor" passa a gerar ruído de conforto para o ouvinte. Este método é utilizado em sistemas de telefonia celular e outros sistemas digitais de comunicação a fim de se poder aumentar o número de canais ou circuitos de comunicação. Entretanto, estas técnicas de detecção de voz podem causar um grande impacto na qualidade dos codificadores que a utilizam. O nível de degradação da voz que utiliza tal técnica é medido através de testes subjetivos. Existem dois problemas principais na implementação desta técnica: a velocidade de re-detecção da voz (na saída do estado de silêncio) e a descontinuidade do ruído de fundo [28].

3.2.5. Especificação e Validação de conformidade

Padrões de codificação de voz podem ser especificados de diferentes maneiras. Entretanto, a exigência mínima de um padrão é a padronização do *bitstream*. O codificador de voz LPC-10 original tem aproximadamente 15 anos e, em sua

especificação, somente o *bitstream* foi especificado, ou seja, todos os parâmetros a serem quantizados, juntamente com suas tabelas de quantização e a ordem que esses parâmetros devem ser transmitidos. Com o passar dos anos, a fim de aumentar a qualidade da voz sintetizada diversas melhorias foram feitas tanto no codificador quanto no decodificador deste padrão.

Ao mesmo tempo que este tipo de especificação parece ser um ponto forte do padrão de codificação, ele também pode ser seu maior ponto fraco. Isto acontece, porque esse tipo de especificação não garante aos usuários que todas as suas implementações terão o mesmo nível de qualidade. Por vários anos, o governo americano comprou equipamentos que incorporavam esta tecnologia de codificação, entretanto teve que estabelecer sua própria metodologia de testes para determinar se os produtos fornecidos pelos diversos fabricantes estavam de acordo com sua necessidade. Além disso, também havia a necessidade de assegurar o desempenho do CODEC (codificador/decodificador) de fabricantes diferentes. Assim, foi necessário testar cada implementação de codificador com cada implementação de decodificador, restando poucos fabricantes que se credenciaram a esta solução e, além disso, havia também o custo da realização dos testes.

Na contra mão deste tipo de especificação, estão as especificações ditas "bit-exact", como os padrões recomentados pelas normas G.721, G.722, G.728 do ITU e o padrão Europeu de celular digital (RPE-LTP). Quando amostras do sinal de voz digital são inseridas em um codificador "bit-exact", este irá fornecer um bitstream também "bit-exact", independente do fabricante que implementou esse codificador. As recomendações de padrões "bit-exact" possuem vetores de testes, que devem ser usados para testar e assegurar que uma determinada implementação está "correta" [11,21,23,26].

A implementação de codificadores de voz com aritimética em ponto flutuante possui um grau de dificuldade maior, pois DSPs que operam em ponto flutuante, de fabricantes diferentes diversos pequenas diferenças nas implementações de suas operações. Essas diferenças provocam diferenças nos codificadores implementados nestes DSPs, fazendo com que um mesmo padrão de codificação possua variações. Isso torna a validação da implementação mais complexa do que nos casos em que são

utilizadas arquiteturas de ponto fixo, para a qual todos os fabricantes possuem um grupo comum de operações e no qual todos os algoritmos "bit-exact" podem ser implementados mais facilmente. Assim, surge um outro tipo de especificação, que se situa entre os dois tipos apresentados anteriormente. Nesse tipo de especificação, os vetores de testes fornecidos trazem um grau de variação aceitável, considerado não perceptível pelo ouvido humano.

É importante lembrar que o processo de validação da especificação precisa existir, pois sem ele não se poderia garantir a interoperabilidade das diversas implementações de um determinado codificador, assim como não seria possível assegurar a qualidade dos serviços que utilizassem estas tecnologias [7].

3.3. Atuação dos órgãos de padronização

É apresentado a seguir uma breve descrição de como os órgãos de padronização (ITU, T1A1,TIA, ETSI) atuam.

3.3.1. ITU – *International Telecommunication Union*

O ITU-T é responsável pela criação dos padrões (ou recomendações – como são normalmente chamados) de codificação de voz para redes de telefônia, que incluem tanto telefonia "através de fios" (*wired*) quanto a telefonia sem fio (*wireless*). O ITU-T é organizado em grupos de estudos (SG) que atuam em diferentes áreas de experimentação, o Grupo de Estudo 15 (SG15), por exemplo, tem a responsabilidade de criar recomendações para codificadores de voz. Entretanto, outros grupos de estudos ou órgãos regionais podem solicitar que seja desenvolvida uma determinada recomendação para uma aplicação específica. Por exemplo, recentemente o SG14, responsável pela padronização de modems, solicitou que fosse criada uma recomendação para um codificador de voz que viabilizasse a utilização de voz e dados simultaneamente. Exemplos dessa natureza mostram a iteração e a dinâmica dos grupos de estudo do ITU-T [7].

No processo de padronização do ITU-T, a primeira etapa é a elaboração de um documento formal, chamado Termo de Referência (*Terms of Reference*), no qual deverão constar as possíveis aplicações para a nova recomendação. Por sua vez, as aplicações determinam quais os aspectos que esse novo padrão deverá atender, no qual

podem ser feitos requerimentos para todos os atributos do codificador. Nesse documento também podem existir objetivos a serem cumpridos, os quais são desejáveis mas não imprescindíveis. Finalmente, um termo de referência deve incluir um plano de trabalho com um cronograma de atividades. Assim, baseando-se nos requisitos e objetivos, um plano de trabalho pode ser definido. Esse plano geralmente irá conter testes subjetivos para medir a qualidade do sinal de voz reconstruído, e testes objetivos para a medida de outros atributos [7].

O Grupo de Estudo 15 (SG15) tem toda a responsabilidade pela criação de termos de referência para codificação de sinais de voz. Enquanto isso, o SGEQ (Speech Quality Experts Group) do Grupo de Estudo 12 ajuda a determinar a qualidade alvo de voz, como também é responsável pela definição e condução dos programas de testes. Esses programas determinam quando um codificador candidato satisfaz os requisitos estabelecidos no termo de referência. Nos programas de testes, procura-se sempre incluir pelo menos três línguas diferentes, a fim de que o codificador seja submetido a maior variedade possível de situações.

3.3.2. Organizações de Padronização Norte Americanas

A ANSI (*American National Standrards Institute*) é o órgão de padronização responsável por todos os padrões dentro dos Estados Unidos. Incluídas na ANSI, existem duas outras organizações que definem padrões de codificação para aplicações específicas, são elas: a T1A1, que desenvolve padrões para a *North American Telecommunications* e o TIA (*Telecommunications Industry Association*), que desenvolve padrões para a telefonia celular e outras aplicações de telecomunicações [7].

Recentemente, subgrupos da T1A1 têm seguido o trabalho do ITU-T, que tem recebido adendos para serem usados nos Estados Unidos. Outra função importante da T1A1 é servir como intermediário para todas as contribuições americanas nos Grupos de Estudo do ITU-T. Essas contribuições são, antes de serem enviadas ao ITU-T, estudadas e revistas tanto pela T1A1 quanto pelo Grupo de Estudos C do Departamente de Estado Americano.

A TIA por sua vez está organizada em comitês e subcomitês divididos por aplicações. O Comitê TR-45, por exemplo, está encarregado dos padrões norte

americanos de telefonia digital celular. Este comitê por sua vez possui os subcomitês TR-45.3, encarregado do padrão TDMA, e TR-45.5, encarregado do CDMA. Outro comitê da TIA é o TR-30 responsável por modem e padrões de dados e voz simultâneos [7,8].

3.3.3. ETSI – *European Telecommunications Standards Institute*

O ETSI (*European Telecommunications Standards Institute*) é o equivalente Europeu ao ANSI Americano, com a diferença de que entre seus membros estão países e empresas. O ETSI é uma organização de fabricantes de equipamentos e surgiu a partir do CEPT (*Conference of European Post and Telephones*), que atualmente está restrito ao fornecimento de serviços. Qualquer fabricante de equipamentos adaptados à Europa pode fazer parte do ETSI, o que faz com que companhias originalmente americanas ou japonesas façam parte desta organização. Os participantes do ETSI são tanto organizações de padronização quanto organizações de pesquisas em novas tecnologias de telecomunicações. Por exemplo, a RACE (*Research on Advanced Communications in Europe*) é uma organização que realiza projetos de pesquisa, que patrocinam uma variedade de outros projetos de pesquisa em telecomunicações, alguns dos quais envolvendo codificação de voz. Entretanto, a RACE não cria novos padrões, seus resultados de pesquisa são utilizados como base para que outras organizações de padronização juntamente com a ETSI criem novos padrões.

O melhor exemplo de uma organização de padronização que funciona com a ETSI e teve um grande impacto sobre os padrões de codificação de voz é o TCH-HS. Esse órgão era originalmente conhecido como o GSM (*Groupe Special Mobile*) e desenvolveu o padrão TDMA em 1987 utilizado na telefonia celular digital Pan-Européia (O GSM foi fundado antes do ETSI ser separado do CEPT). Uma parte do padrão GSM era o codificador de voz 13 kb/s "*Regular Pulse Excitation Coder*". Atualmente o TCH-HS padronizou um novo codificador que opera com a metade da taxa do codificador original, duplicando a capacidade do sistema GSM [7].

3.3.4. RCR Japonesa

O padrão de telefonia celular Japonês foi criado pela RCR. Esse órgão é caracteristicamente fechado e no recente processo para padronização de um novo

padrão com metade da taxa, permitiu a participação somente das companhias que planejavam submeter um codificador candidato. As companhias que tivessem seus codificadores candidatos de alguma forma excluídos do processo, não eram mais bem vindas às reuniões. Não existe também nenhuma documentação disponível sobre o que é discutido nessas reuniões, nem mesmo para aqueles que participam [7].

3.4. Padrões atuais de codificação

Nesta seção serão apresentados e descritos os processos de padronização de algumas recomendações de codificadores de voz feitas pelo ITU-T [2,3,20-27].

3.4.1. G.711 64 kb/s PCM

O G.711, padronizado pelo CCITT (atual ITU-T), especifica na realidade dois padrões: um para a America do Norte e para o Japão (μ-law PCM); e outro para o resto do mundo (A-law PCM). Ambos os padrões utilizam 8 bits para representar o sinal de voz, e apresentam uma relação sinal-ruído de 35 dB. O padrão A-law PCM possui grande resolução para amostras com valores pequenos e uma faixa equivalente a 12 bits em PCM linear. Já o μ-law PCM possui uma faixa equivalente a 13 bits em PCM linear, apresentando, entretanto, uma granularidade maior para amostras de pequeno valor. Esses dois padrões possuem qualidade equivalente. Um detalhe interessante desses padrões é que se pode conseguir uma taxa de 56 kbits/s, colocando-se um em série com o outro e descartando-se o último bit (bit menos significativo) [7,20].

3.4.2. G.721, G.723, G726 e G.727 ADPCM

O primeiro codificador ADPCM (*Adaptative Differencial PCM*) a ser padronizado pelo CCITT foi o 32 kb/s (G.721) [21]. Esse codificador foi desenvolvido por duas razões: primeiramente se desejava utilizá-lo em sistemas de multiplicação de circuitos/canais digitais (DCME), pois a aplicação dessa codificação representava uma duplicação da capacidade destes sistemas. Além disso, este padrão de codificação quando combinado a técnicas de interpolação de sinal de voz, que podem fornecer uma compressão de 2.5:1, poderia ser usado para aumentar, em até 5 vezes, a capacidade de transmissão dos cabos submarinos e *links* de satélites. A segunda razão era que, em diversas situações, os *links* PCM operavam com padrões diferentes em suas

extremidades (μ-law em uma e A-law em outra). Assim, esse padrão foi criado de forma a trabalhar com entradas tanto em μ-law quanto em A-law, entretanto um fato interessante era que esse padrão não aceitava entradas no formato PCM linear. Outra característica importante desse padrão era a sua propriedade chamada "*synchromous tandeming*", na qual se um circuito de comunicação envolvesse duas codificações ADPCM com codificações μ-law ou A-law entre elas, a segunda codificação ADPCM não causava mais nenhuma degradação na voz. O G.721 foi padronizado em 1984 e teve que ser repadronizado em 1986 por causa de alguns pequenos problemas em sua primeira versão. Um de seus problemas era que as redes norte americanas não suportavam longas cadeias de zeros, fazendo com que a versão revisada deste padrão eliminasse o *codword* composto só de zeros. Essa modificação (repadronização) foi feita alterando-se o quantizador de 16 para 15 níveis.

O G.723 foi padronizado em 1988 para ser utilizado em aplicações de DCME [22]. Essa nova padronização do ADPCM inclui duas taxas de bits adicionais: 24 kb/s e 40 kb/s. A taxa de 24 kb/s era para ser usada quando o tráfego estivesse momentaneamente maior do que a capacidade de transmissão. Já a taxa de 40 kb/s foi adicionada para possibilitar ao codificador a capacidade de transmitir os sinais de modems a 9600 b/s, já que não eram satisfatoriamente codificados pela G.721.

O G.726 [24] por sua vez apresentava a unificação dos padrões de codificação G.721 e G.723. Em adição, esse novo padrão incorporou mais uma taxa de compressão, 16 kb/s, e mais uma vez sua aplicação alvo era os DCME. As taxas de 16 e 24 kb/s não podiam ser consideradas "*toll quality*"[1]. Entretanto, como essas taxas só eram utilizadas pelos DCME em períodos curtos de tempo, em que a carga do sistema ultrapassasse a sua capacidade, o sistema em sua média de operação era considerado "*toll quality*".

Finalmente, o padrão G.727 [25] incluía todas as taxas do G.726, com a diferença de que todos os seus quantizadores tinham um número ímpar de níveis. Assim, o quantizador de 2 bits era incluso no de 3 bits que era incluso no de 4 bits que por sua vez era incluso no de 5 bits. Isso se fazia necessário em equipamentos de

[1] Codificação de voz mantendo, subjetivamente, a mesma qualidade da voz original (transparência na qualidade de voz).

multiplexação de pacotes (PCME) para o qual o bit menos significativo poderia ser descartado, caso houvesse necessidade de reduzir a taxa de transmissão, em casos de sobrecarga [7].

3.4.3. G.728 16 kb/s LD-CELP

O programa de trabalho para a padronização do G.728 [26] foi iniciado pelo CCITT em 1988 e tinha a intenção de criar um padrão de codificação de voz universal de 16 kb/s. Esse padrão tinha por objetivo ser "*toll quality*", o que significava que sua qualidade deveria ser igual ou superior a qualidade de seu antecessor o de 32 kbits/s (G.721). Já, por universal, se entendia que este codificador deveria poder ser utilizado em qualquer lugar ou situação. Entretanto, acabou-se restringido o uso desse padrão à comunicação em redes. O G.728 não podia ser utilizado nos troncos principais de comunicação, no entanto servia bem para aplicações terminais ou aplicações ponto a ponto. Por ser um codificador com baixo atraso, seu uso era bastante interessante também para aplicações que necessitavam dessa característica. A primeira grande aplicação deste padrão foi na recomendação H.320 (videofones) [29]. Trabalhos continuam sendo realizados pelo Grupo de Estudos 15 (SG15) do ITU-T a fim de desenvolver extensões a este padrão que possibilitem sua utilização em equipamentos de multiplicação de circuitos/canais e para aplicações de rádio.

A especificação original do codificador CELP (*Code Excited Linear Prediction*) da recomendação G.728 foi feita em ponto flutuante, o que exigia que sua implementação seguisse exatamente o algoritmo especificado pela recomendação. Um conjunto de vetores de teste para verificação da implementação foi criado. Posteriormente, uma implementação em ponto fixo foi requisitada e completada em 1994.

O desempenho desta recomendação foi exaustivamente testado pelo SQEG e foi mostrado que este padrão oferecia grande robustez a ruídos de fundo e músicas. Foi também observado que o G.728 suportava melhor erros de bits aleatórios do que os padrões anteriores. Foi mostrado também que o G.728 suportava todos os sinais de sinalização de rede telefônica e era capaz de passar sinais de modems de 2400 bits/s [7].

3.4.4. G.729 8 kb/s CS-ACELP

O programa de trabalho para o codificador G.729 foi iniciado em 1990 pelo CCITT. Esse esforço foi iniciado em resposta a uma tarefa requisitada ao CCIR de desenvolvimento de um codificador de voz para ser usado no projeto de um futuro serviço público de telefone móvel. Havia um requerimento em relação ao tamanho do atraso neste padrão, entretanto este foi deixado de lado em 1991 passa reduzir a complexidade de implementação de tal padrão.

Em 1993, dois codificadores candidatos foram testados, um da NTT (*Nippon Telephone e Telegraph*) e outro desenvolvido através da parceria entre a France Telecon/CNET e a Universidade de Sherbrooke na Inglaterra. Ambos os candidatos alcançaram os requerimentos necessários para excitação de voz limpas, entretanto ambos falharam nos requerimentos quando a entrada apresentava diferentes níveis, voz com ruídos de fundo e músicas. Assim, em 1994, um acordo determinou que pesquisas fossem realizadas por estas duas organizações e também pela AT&T a fim de desenvolver um codificador capaz de alcançar todos os requisitos desejados. Em 1995, foi então testado um codificador com tais características, e teve sua aprovação no final desse mesmo ano [7,11,27].

3.4.5. G.723.1 6.3/5.3 kb/s

O programa de trabalho para o codificador G.723.1 foi iniciado em 1993 pelo ITU-T como parte de um grupo de padrões para especificar um sistema de videofone com baixa taxa de bits, e planejava-se utilizar este sistema em PSTN (*Public Switched Toll Network*) sobre modems de alta velocidade. Outros padrões, como codificadores de vídeo e modems, foram desenvolvidos para este sistema.

Havia a intenção de que este codificador de voz tivesse qualidade tão alta quanto fosse possível, com uma complexidade não muito grande e um tamanho de quadro com, no máximo, 30 ms. Em março de 1994, testes subjetivos foram realizados pela AT&T em 5 codificadores candidatos, dos quais dois foram escolhidos para serem melhor estudados de forma a juntar esforços dos projetistas para criar um codificador que satisfizesse a todos os critérios. Os grupos escolhidos foram o DSP Group AudioCodes, Ltd. e a France Telecom/CNET em parceria com a Universidade de Sherbrook [7,2,20].

3.5. Conclusão

O objetivo deste Capítulo foi o de situar este trabalho no contexto dos sistemas de compressão de sinais de voz, mostrando o que são e como surgiram os padrões de codificação. Desta maneira, foram apresentados os grupos que atuam nos processos de padronização como também o mecanismo que gera o surgimento dos padrões de codificação de voz.

No próximo capítulo será descrito com mais detalhes a recomendação G.723.1, utilizada como modelo para a implementação proposta neste trabalho.

Capítulo 4

Codificador de Voz G.723.1

4.1. Introdução

Neste capítulo, é apresentada de forma sucinta, uma descrição do funcionamento do sistema de codificação definido na recomendação G.723.1 [2,3]. Esse padrão de codificação, como mencionado no capítulo anterior, foi desenvolvido para ser utilizado no fornecimento de serviços de multimídia para operação em baixas taxas de bits [2]. A aplicação alvo considerada no projeto deste codificador foi o videofone com baixa taxa de bits, o qual está definido na recomendação H.324 [30]. Posteriormente este padrão passou a incorporar, juntamente com o G.711 [20], o grupo básico de codificadores de voz utilizados na definição de terminais multimídia, padronizados pela recomendação H.323 [31].

Este codificador opera em duas taxas de bits, 5.3 e 6.3 kbps, sendo que a taxa mais alta possui uma qualidade de voz reconstruída considerada "toll quality" (MOS 3.89). A taxa menor também possui uma boa qualidade de voz (MOS 3.82), entretanto a sua existência dá ao projetista de sistemas que incorporam este padrão de codificação a

flexibilidade de alterar a taxa de operação do codificador durante o seu uso. Isto é, torna-se possível alterar a taxa de operação deste codificador a cada 30 ms [2]. Essa característica é muito desejada quando, por exemplo, deseja-se projetar um sistema a ser utilizado em canais de comunicação que operem com taxas variáveis.

Este padrão foi desenvolvido para codificar voz, entretanto, com algumas restrições, não existem impedimentos à codificação de músicas ou de outros sinais. O atraso de codificação é de 37,5 ms, do qual 30 ms é o tamanho de quadro e 7,5 ms é um segmento de *look-ahead* incorporado ao processo de codificação [2].

A seguir é apresentada uma breve descrição do codificador e do decodificador.

4.2. Descrição do Sistema de Codificação

Este codificador foi concebido para operar com sinais digitais em PCM linear (com 16 bits por amostra) com uma taxa de amostragem de 8 kHz. Assim, sinais analógicos devem ser, antes de entregues ao codificador, filtrados na largura de banda de telefone (ITU-T G.712 [32]), amostrados em 8kHz e então convertidos em PCM linear. A saída do decodificador, também está no formato PCM linear com uma taxa de amostragem de 8 kHz. Se o sinal de saída desejado for um sinal analógico, a saída do decodificador deve então ser convertida adequadamente para este tipo de sinal. O *bitstream* utilizado na comunicação entre o codificador e o decodificador está definido na recomendação G.723.1 [2,3].

4.2.1. Codificador

O princípio de funcionamento deste codificador é o de análise por síntese, que busca minimizar um sinal de erro ponderado perceptualmente. O codificador trabalha com quadros do sinal de voz (*frames*) de 240 amostras, o que, para 8 kHz, representam um segmento de duração de 30 ms. No processo de codificação, cada quadro do sinal de voz é dividido em sub-quadros de 60 amostras. Para cada um desses sub-quadros, é obtido um conjunto de coeficientes de um filtro LPC (*Linear Prediction Coder*) de décima ordem, sendo que o último sub-quadro será quantizado por um quantizador vetorial preditivo. Os coeficientes do filtro LPC não-quantizados são então usados para construir o filtro preditor de curto atraso com ponderação perceptual, o qual será usado

para filtrar o bloco completo a fim de obter o sinal de voz ponderado perceptualmente [2,3].

Para cada dois sub-blocos (120 amostras), é estimado o período de *pitch* em laço aberto, utilizando-se para tal o sinal de voz ponderado. O período de *pitch* é procurado em uma faixa de 18 a 142 amostras. Usando a estimação de *pitch*, um filtro de conformação de ruído harmônico é construído. Desta maneira, o filtro de síntese LPC, o filtro de formante com ponderação perceptual e o filtro de forma de ruído harmônico são combinados para gerar uma determinada resposta ao impulso, que é utilizada nos procedimentos seguintes.

Assim, usando a estimação do *pitch* e a resposta ao impulso, é calculado um preditor de período de *pitch* de laço fechado. Em tal procedimento é utilizado um preditor de *pitch* de quinta ordem. Esse período de *pitch* é calculado com o menor valor de diferença em torno da estimação de *pitch* de laço aberto. A contribuição do preditor de *pitch* é então subtraída do vetor alvo original resultando em uma diferença que juntamente com o período de *pitch*, é enviado ao decodificador.

Finalmente o componente não periódico da excitação é aproximado. Para o codificador operando na taxa alta, é utilizado, para este cálculo, o algoritmo MP-MLQ (*Multi-pulse Maximum Likelihood Quantization*) e, para a taxa baixa, o algoritmo ACELP (*Algebraic-code-excitation*).

A seguir, na Figura 4.1 é mostrado o diagrama de blocos do codificador.

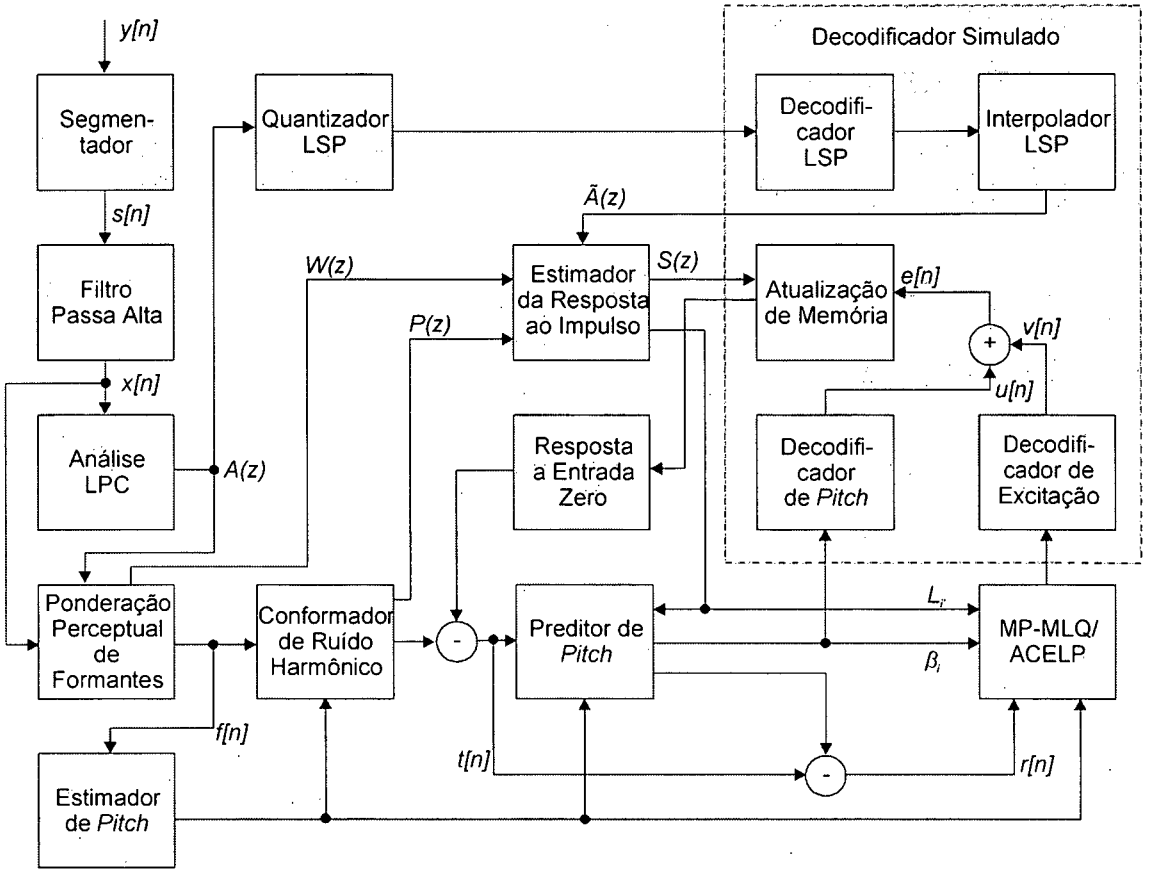

Figura 4.1: Diagrama em blocos do Codificador da Recomendação G.723.1 (Adaptado de [2] p. 3)

4.2.2. Decodificador

A operação de decodificação é também realizada quadro-a-quadro. Inicialmente, os índices dos coeficientes do filtro LPC quantizados são decodificados de maneira a gerar o filtro LPC de síntese. Para cada sub-quadro, tanto a excitação do *codebook* adaptativo quanto a excitação do *codebook* fixo são decodificadas e usadas como entrada para o filtro de síntese. Um pós-filtro adaptativo, considerando os parâmetros do interpolador LSP e do filtro de síntese, é utilizado para minimizar e suavizar as imperfeições do sinal de voz sintetizado. Finalmente é realizado um escalamento de ganho para manter o nível de energia compatível com o nível do sinal de voz original [2,3].

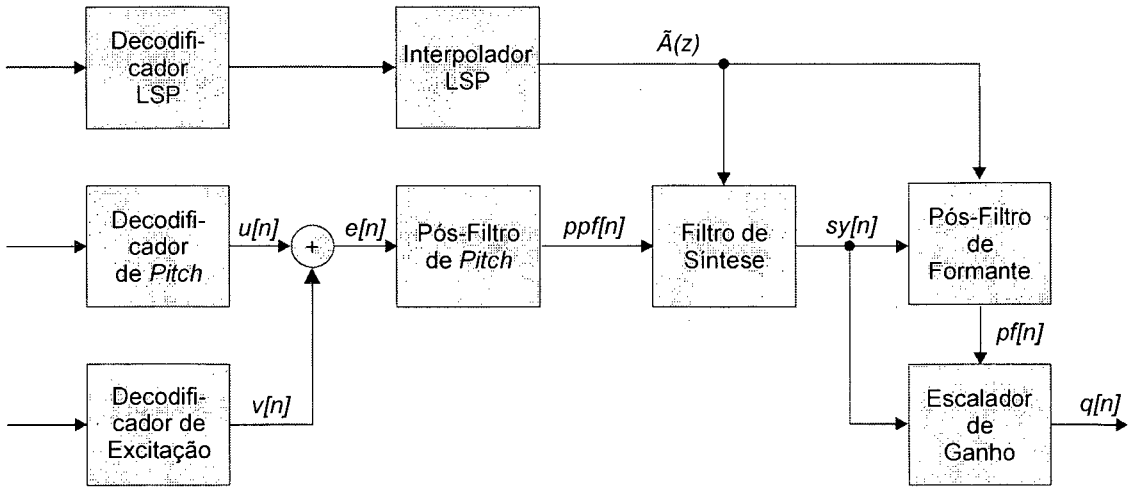

Figura 4.2: Diagrama em blocos do Decodificador da Recomendação G.723.1 (Adaptado de [2] p. 17)

Os detalhes de operação de cada bloco do codificador e decodificador podem ser obtidos nas referências [2] e [3]. A completa inclusão destes detalhes viola direitos autorais definidos pelo ITU-T, por essa razão os detalhes não estão mostrados neste trabalho.

Capítulo 5

Descrição da Arquitetura da Família de DSP ADSP-2100

5.1. Introdução

A família de DSPs ADSP-2100 da Analog Devices compreende um conjunto de microprocessadores programáveis encapsulados em um único *chip*. Esses microprocessadores compartilham uma arquitetura otimizada para o processamento digital de sinais e outras aplicações que exijam processamento numérico com alto desempenho. Os diversos processadores (DSPs) desta família se diferenciam, entre si, principalmente pelo tipo de periféricos que são acrescentados a sua arquitetura básica. Periféricos como memória interna, *timer*, portas seriais e portas paralelas estão disponíveis em diferentes membros desta família. O ADSP-21msp58/59, por exemplo, acrescenta a sua arquitetura básica uma interface analógica para conversão de sinais na banda de voz.

É importante salientar que apesar do DSP escolhido para esta implementação ter sido o ADSP-2181, a arquitetura básica dos DSPs desta família é idêntica [4,34-36].

5.2. Arquitetura

Todos os processadores da família ADSP-2100 contêm três unidades computacionais independentes e totalmente funcionais, que são: a unidade lógica e aritmética (ALU), a unidade de multiplicação/acumulação (MAC) e a unidade de deslocamento. Essas unidades processam diretamente dados com 16 bits, assim como possuem suporte para cálculos com outras precisões [4].

A arquitetura básica deste DSP também inclui dois geradores de endereços de dados e um sequenciador de programa. O sequenciador de programa é responsável pela execução de desvios condicionais e repetições em um único ciclo de máquina. Já os dois geradores de endereços de dados possibilitam que uma única instrução possa buscar dois operandos na memória. Juntos, o sequenciador de programas e os geradores de endereço mantêm as unidades computacionais sempre funcionando, possibilitando assim rendimento máximo do DSP.

Esta família de DSPs utiliza em sua memória uma arquitetura de tipo *Harward* modificada, na qual a memória de dados armazena dado e a memória de programa armazena código e dado. Todos os DSPs desta família possuem memória interna no *chip*, sempre divididas em memórias de dados e programa. A utilização da memória interna, graças a sua velocidade, possibilita que, em um único ciclo, seja possível buscar um operando da memória de dados, um operando da memória de programa e ainda uma instrução na memória de programa.

Os DSPs desta família podem possuir, periféricos como portas seriais, *timer*, portas HIP, portas DMA e interface analógicas [4].

Portas Seriais

As portas seriais (chamadadas SPORTs) fornecem uma interface de comunicação serial completa, incluindo lógicas de compressão e expansão (*A*-law, μ-law). As portas seriais fornecem uma interface simples entre uma grande variedade de dispositivos seriais populares. Cada porta serial pode gerar um

clock programável interno ou ainda aceitar um sinal de *clock* externo. A porta serial 0 possui a opção de funcionar com múltiplos canais.

Timer

Um timer/contador programável de 8-bits preescaldados fornece a geração de interrupções periódicas.

Portas HIP (Host Interface Port)

A porta HIP permite uma conecção direta (sem nenhuma lógica adicional) com um outro processador. O HIP é constituído por 16 pinos de dados e 11 pinos de controle. Esta porta é extremamente flexível e foi projetada para permitir uma interface simples com uma variedade de outros processadores. Por exemplo, o Motorola 68000, o Intel 8051 ou outro DSP desta família podem ser facilmente conectados à porta HIP.

Portas DMA

Tanto a porta DMA interna (IDMA) quanto a porta Byte DMA (BDMA) do ADSP-2181 fornecem transferência de dados de forma eficiente para e da memória interna deste DSP.

A Tabela 5.1 mostra algumas características de alguns membros da família de DSPs ADSP-2100. Nela se pode observar quais periféricos estão presentes em cada membro, assim como a sua quantidade de memória, a sua velocidade e a sua tensão de alimentação.

Tabela 5.1: Características dos DSPs da família ADSP-21XX (Adaptado de [4] p. 1-2)

Característica	2101	2103	2105	2115	2111	2171	2181	2183	2187
Unidade Lógica/Aritimética	✓	✓	✓	✓	✓	✓	✓	✓	✓
Multiplicador/Acumulador	✓	✓	✓	✓	✓	✓	✓	✓	✓
Deslocador	✓	✓	✓	✓	✓	✓	✓	✓	✓
Gerador de Endereços de Dados	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sequenciador de Programa	✓	✓	✓	✓	✓	✓	✓	✓	✓
Memória de Dados	1 K	1K	512	512	1K	2K	16 K	16 K	32 K
Memória de Programa	2 K	2 K	1 K	1 K	2K	2K	16 K	16 K	32 K
Timer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Porta Serial 0 (multicanal)	✓	✓		✓	✓	✓	✓	✓	✓
Porta Serial 1	✓	✓	✓	✓	✓	✓	✓	✓	✓
Porta HIP					✓	✓			
Porta DMA							✓	✓	✓
Tensão de Alimentação	5 V	3.3 V	5 V	5 V	5 V	5 V	5 V	3.3 V	3.3 V
Mipagem (MIPS)	20	10	13.8	20	20	33	33	33	52

Cada processador desta família possui internamente quatro barramentos que conectam a memória interna com suas outras unidades funcionais. São eles: barramento de Endereço da Memória de Dados (DMA BUS), barramento de Dados da Memória de Dados (DMD BUS), barramento de Endereços da Memória de Programa (PMA BUS) e barramento de Dados da Memória de Programa (PMD BUS) [4].

Os DSPs desta família possuem um alto nível de paralelismo, talhados para utilização em Processamento de Sinais Digitais. Em um único ciclo, os DSPs desta família são capazes de executar:

- Geração do próximo endereço de programa
- Busca da próxima instrução
- Execução de uma ou duas movimentações de dados

- Atualização de um ou dois apontadores de dados
- Realização de uma computação (ALU/MAC/Descolamento)

E podem ainda, no mesmo ciclo (se possuírem o periférico em questão):

- Receber ou transmitir dados através da porta serial
- Receber ou transmitir dados através da porta HIP
- Receber ou transmitir dados através da porta DMA
- Receber ou transmitir dados através da interface analógica

Aplicações de processamento de sinais criam necessidades de desempenho especiais, as quais distinguem a arquitetura de um DSP das arquiteturas de outros microprocessadores e microcontroladores. Assim, este DSP, além de executar uma instrução em alta velocidade, deve ter uma boa desenvoltura nas seguintes áreas: aritmética rápida e flexível, extensão de faixa dinâmica, busca de dois operandos em um único ciclo, suporte de *buffer* circular em *hardware* e repetições e desvios sem *overhead* [4].

A Figura 5.1 apresenta um diagrama da arquitetura básica dos DSPs desta família.

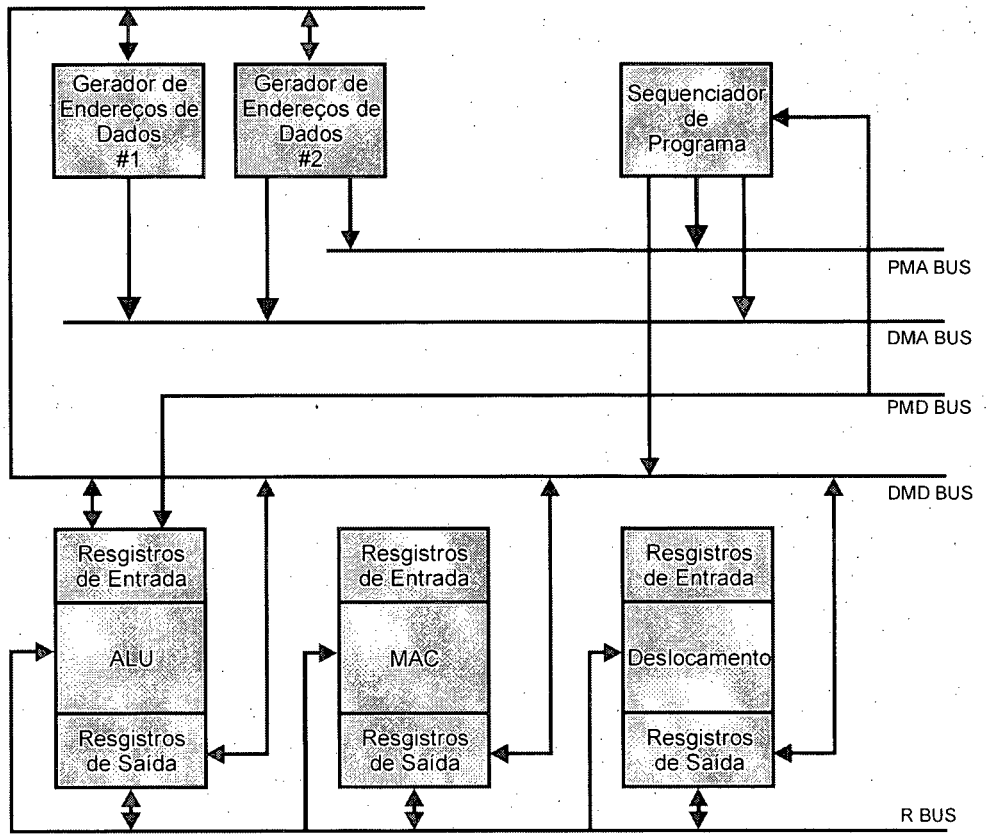

Figura 5.1: Diagrama da Arquitetura Básica dos DSPs da família ADSP-2100
(Adaptado de [4] p. 1-6)

5.3. Unidades Computacionais

As unidades computacionais presentes nesta família de DSPs são: unidade lógica e aritmética (ALU), multiplicador/Acumulador (MAC) e unidade de deslocamentos [4,35,36].

Todos os dispositivos na família ADSP-2100 trabalham com 16 bits em ponto fixo. A maior parte das operações nestes DSPs são realizadas com a representação numérica de complemento de dois. Existem operações (ou modos de operações) que podem trabalhar com representações numéricas não sinalizadas ou ainda trabalhar com cadeias de bits.

5.3.1. Unidade Lógica e Aritmetica (ALU)

A unidade lógica e aritmética fornece um conjunto padrão de operações lógicas e aritméticas. As funções aritméticas incluídas nesta unidade são: adição, subtração, negação, incremento, decremento e valor absoluto. Existem ainda, no bloco aritmético, duas operações primitivas de divisão, com as quais podem ser construídas operações de divisões. As funções lógicas, por sua vez, são: AND, OR, XOR e NOT. A Figura 5.2 apresenta um diagrama em blocos da Unidade Lógica e Aritmética.

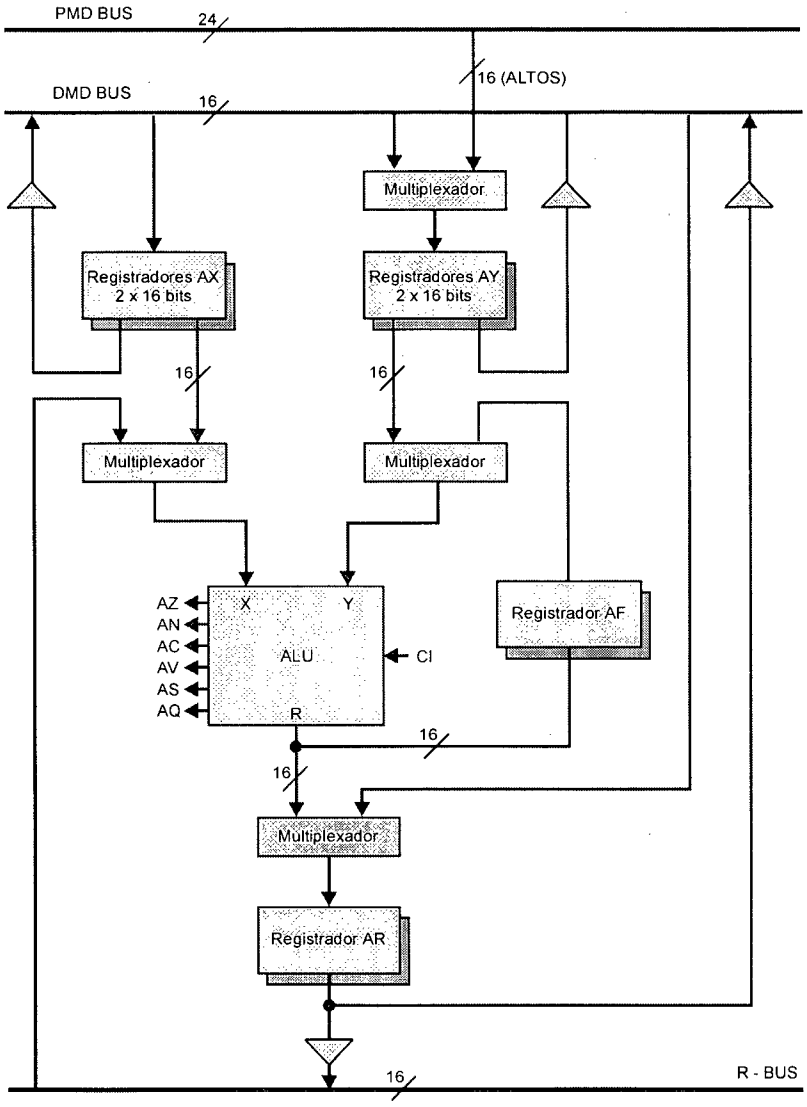

Figura 5.2: Diagrama em blocos da Unidade Lógica e Aritmética (Adaptado de [4] p. 2-6)

A Unidade Lógica e Aritmética (ALU) trabalha com precisão de 16 bits e possui duas portas de entrada (X e Y) e uma porta de saída (R). Essa unidade pode receber

também um sinal de *carry*, ou seja um bit de *carry* do registrador de estado aritmético do processador (ASTAT). Esta unidade lógica e aritmética gera seis sinais de estado: o estado zero (AZ), o estado negativo (AN), o estado de *carry* (AC), o estado de *overflow* (AV), o estado do sinal da entrada X (AS) e o estado de quociente (AQ). Todos os sinais de estado são escritos no registro de estado aritmético (ASTAT) no final do ciclo de execução de uma operação na ALU [4].

A porta de entrada X da ALU aceita dados tanto do arquivo de registro AX, quanto do barramento R (R BUS). O barramento R está conectado a todos os registradores de saída de todas as unidades computacionais, permitindo assim que qualquer resultado gerado por elas possa ser utilizado diretamente como um operando da ALU. O arquivo de registro AX é dedicado à porta de entrada da ALU e consiste de dois registradores, AX0 e AX1. Estes registradores podem ser lidos ou escritos do barramento de dados da memória de dados (DMD BUS). O conjunto de instruções deste DSP também permite que esses registradores sejam carregados a partir do barramento de dados da memória de programa (PMD BUS), entretanto esta carga não ocorre diretamente, tendo que ser utilizada a unidade de troca de barramento DMD-PMD. O arquivo de registro AX possui duas portas (*dual-ported*), podendo ser usado na ALU e simultaneamente carregado a partir do barramento de dados.

A porta de entrada Y da ALU, por sua vez, também pode receber dados de duas fontes: do arquivo de registro AY e do registro de realimentação da ALU (AF – *ALU Feedback*). O arquivo de registro AY é dedicado à entrada Y da ALU e, assim como o arquivo de registro AX, consiste de dois registradores: AY0 e AY1. Esses registradores podem ser lidos e escritos a partir do barramento de dados da memória de dados (DMD BUS) e escritos a partir do barramento de dados da memória de programa (PMD BUS) diretamente. No entanto, da mesma forma que os registradores AX0 e AX1, os registradores AY0 e AY1 não podem ser diretamente lidos a partir do barramento de dados da memória de programa, sendo necessário utilizar a unidade de troca de barramento (DMD-PMD). O arquivo de registros AY também possui duas portas, podendo ser usado na ALU e carregado a partir do barramento de dados simultaneamente.

A saída da ALU pode ser carregada tanto no registro de realimentação da ALU (AF) quanto no registro de resultado (AR). O registro AF é interno à ALU e permite que os resultados gerados por ela possam ser diretamente usados como entrada em sua porta Y. O registrador AR pode alimentar tanto o barramento R quanto o barramento de dados da memória de programa, e pode ser carregado diretamente a partir do barramento de dados da memória de programa. O conjunto de instruções permite que o registrador AR também possa ser carregado no barramento de dados da memória de programa, entretanto da mesma forma como ocorre com os arquivos de registros AX e AY, é necessário para executar tal operação, a passagem pela unidade de troca de barramento (DMD-PMD).

Quaisquer registros associados com a ALU podem ser lidos e escritos no mesmo ciclo. A leitura desses registros ocorre no início do ciclo, e a escrita, no final. Uma leitura de registro carrega o valor escrito no final do último ciclo, não sendo possível que um valor escrito no registro possa ser lido no mesmo ciclo, sendo necessário para tal esperar pelo próximo ciclo. Isso permite que um registro de entrada da ALU possa ser usado como operando no início do ciclo e atualizado no final desse mesmo ciclo, para ser utilizado em uma próxima instrução. É também permitido que o registrador de resultado possa ter seu valor armazenado em memória e atualizado com um novo resultado no mesmo ciclo.

A ALU desta família de processadores possui dois bancos de registros, ou seja, possui dois conjuntos de registradores AR e AF como também dois conjuntos de arquivos de registros AX e AY, entretanto somente um banco de registro está acessível por vez. O banco de registradores adicional permite um chaveamento de contexto extremamente rápido e eficiente, permitindo que novas tarefas, como serviços de tratamento de interrupções, possam ser executadas sem a necessidade de transferir o contexto do processador para a memória.

A seleção do banco de registros primário ou alternativo é feita através do bit 0 do registro de controle do modo de operação do processador (MSTAT) (0 = banco primário, 1 = banco alternativo) [4].

5.3.2. Multiplicador / Acumulador (MAC)

Esta unidade fornece operações em alta velocidade de multiplicação, multiplicação com adições cumulativas e multiplicação com subtrações cumulativas. Esta unidade também possui funções de realimentação, permitindo que parte do resultado de uma multiplicação possa ser diretamente reutilizado para uma outra multiplicação no ciclo seguinte. A Figura 5.3 mostra uma visão geral do diagrama de blocos da unidade de MAC.

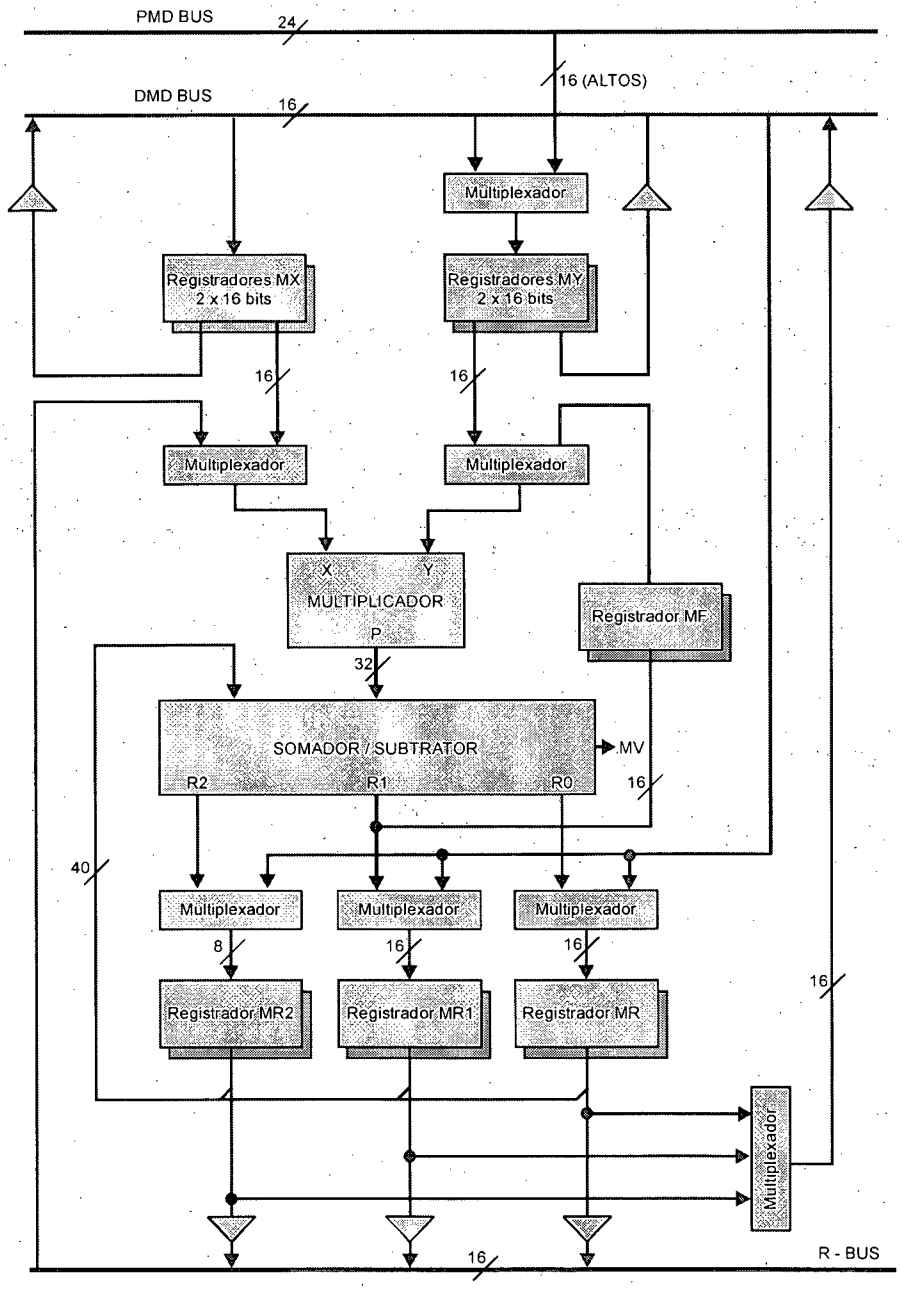

Figura 5.3: Diagrama de blocos da MAC (Adaptado de [4] p. 2-14)

O multiplicador desta unidade possui duas portas de entrada, X e Y, de 16 bits de largura e uma porta de saída de produto, P, com 32 bits de largura. Os 32 bits de produto são passados ao módulo somador/subtrator que adiciona ou subtrai o novo produto (P) do conteúdo do registrador de resultado da unidade de MAC (MR). O módulo somador/subtrator também pode passar diretamente o resultado da multiplicação para o registro MR. Esse registro possui 40 bits de largura e está organizado em três registros menores: MR2 (8bits), MR1 (16 bits) e MR0 (16 bits).

O módulo somador/subtrator por possuir mais de 32 bits é capaz de suportar alguns *overflows* intermediários durante uma série de multiplicações e acumulações. Assim, quando uma acumulação ultrapassa 32 bits, ou seja, quando os nove bits mais significativos do registrador MR (40 bits) são significantes (não indicam sinal), é ativado o *flag* MV, que indica um *overflow* na unidade de MAC.

Os registros de entrada e saída do MAC têm um funcionamento semelhante aos registros de entrada e saída da ALU. Os arquivos de registros de entrada MX e MY funcionam exatamente como os arquivos de registros de entrada AX e AY da ALU.

A saída do somador/subtrator é passada tanto para o registrador MF quanto para o registrador MR (MR2 MR1 MR0). O MF é um registro de realimentação e, assim como o registro AF da ALU, serve diretamente como um operando da MAC. Este registrador recebe os 16 bits mais significativos do resultado em 32 bits. O registrador de adição/subtração de 40 bits MR, como mencionado anteriormente, está organizado em três registradores (MR2 MR1 MR0), sendo que cada um deles pode ser carregado a partir do barramento de dados da memória de dados (DMD BUS) como também do barramento R (R BUS).

Todos os registradores associados ao MAC podem ser lidos e escritos em um mesmo ciclo. Seu funcionamento é idêntico ao funcionamento dos registradores associados à ALU. O MAC, assim como a ALU, também contém um banco alternativo de registradores, e sua seleção também é feita no bit 0 do registrador de modo de operação do processador [4].

5.3.3. Unidade de deslocamento (*BARREL SHIFT*)

A unidade de deslocamento fornece um conjunto completo de funções de deslocamento para entradas de 16 bits, gerando saídas de 32 bits. Essas funções incluem deslocamentos aritméticos, deslocamentos lógicos e normalização. Esta unidade é capaz ainda de realizar operações de extração de expoente, assim como extração de expoente comum para um conjunto de valores. Essas funções básicas podem ser combinadas a fim de fornecer uma implementação eficiente para qualquer tipo de manipulação e controle de formatos numéricos, incluindo a representação completa de ponto flutuante. A Figura 5.4 apresenta o diagrama em blocos desta unidade.

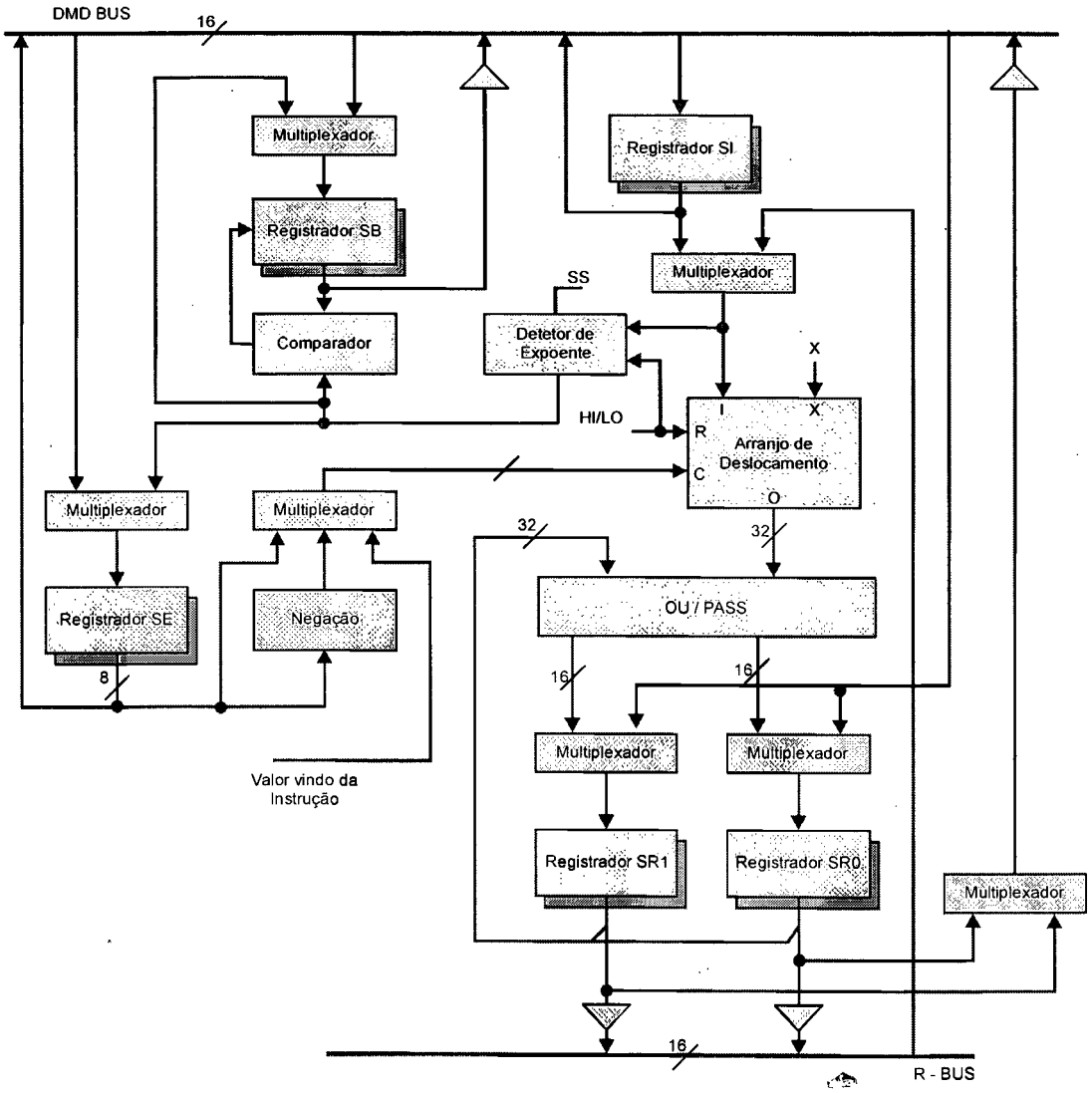

Figura 5.4: Diagrama em blocos da unidade de deslocamento (*BARREL SHIFTER*)

(Adaptado de [4] p. 2-23)

Pode-se observar na Figura 5.4 que a unidade de deslocamento pode ser dividida em quatro partes: o arranjo de deslocamento, a lógica de OU / PASS, o detetor de expoente e a lógica de comparação de expoente.

O arranjo de deslocamento toma como entrada valores de 16 bits e pode deslocá-lo para qualquer posição dentro dos 32 bits do campo de saída em um único ciclo. O deslocamento é determinado por um código de controle (C) e um sinal de referência HI/LO.

O arranjo de deslocamento e sua lógica de funcionamento estão cercados por um conjunto de registradores. O registrador de entrada de deslocamento (SI) pode ser usado como entrada do arranjo de deslocamento como também do detetor de expoente. O registrador SI possui 16 bits e pode ser lido e escrito a partir do barramento de dados da memória de dados (DMD Bus). O arranjo de deslocamento e o detetor de expoente também podem utilizar como entrada os registradores AR, SR e MR através do barramento R (R BUS). O registrador de resultado do deslocamento (SR) possui 32 bits e está dividido em dois registradores: SR1 e SR0. Esses registradores (SR1 e SR0) podem ser lidos ou escritos a partir do barramento de dados da memória de dados e podem ser escritos no barramento R (R BUS). O registrador SR também serve de realimentação para o bloco de lógica de OU / PASS, de forma a permitir operações de deslocamento com dupla precisão.

O registrador de expoente de deslocamento, SE (*shifter expoent*), possui 8 bits e guarda o valor do expoente durante operações de normalização e desnormalização. Este registrador pode ser lido e escrito através dos 8 bits menos significativos do barramento de dados da memória de dados. Este registrador utiliza a representação numérica de complemento de dois.

O registrador SB (*shifter block*), por sua vez, é extremamente importante em blocos de operações em ponto flutuante, pois ele guarda o valor do expoente do bloco, ou seja, o valor do expoente pelo qual todos os elementos do bloco precisam ser deslocados a fim de normalizar o maior valor. Este registro possui 5 bits e serve para guardar o valor do expoente do bloco mais recente. O registrador SB pode ser lido e escrito dos 5 bits menos significativos do barramento de dados da memória de dados.

Esse registrador também opera usando a representação numérica de complemento de dois.

Quando os registradores SB (5 bits) e SE (8 bits) são escritos no barramento de dados da memória de dados, eles têm seus sinais estendidos para formar um número de 16 bits.

Qualquer um dos registros SR, SE e SI desta unidade computacional pode ser lido e escrito em um mesmo ciclo. Assim como nas unidades funcionais anteriores, a leitura desses registros ocorre no início do ciclo, e a escrita, no final.

A unidade de deslocamento, assim como as demais, também possui um banco de registradores alternativos, que inclui os registradores (SE, SB, SI e SR1 e SR0). A seleção do banco é também feita através do bit 0 do registrador de modo de operação do processador.

O deslocamento dos dados de entrada da unidade de deslocamento é feita, como dito anteriormente, com base no código de controle C e pelos sinais de referência HI e LO. O código de controle é um valor de 8 bits sinalizado que indica a direção e o número de deslocamentos que o valor de entrada deve receber. Códigos com valores positivos indicam deslocamento para a esquerda (*upshift*) e códigos com valores negativos indicam deslocamento para a direita (*downshift*). O código de controle pode vir de três fontes: o conteúdo do registrador SE, o valor negativo do registrador SE ou ainda um valor imediato passado diretamente através da instrução.

Os sinais HI/LO determinam os pontos de referência para o deslocamento. No estado HI, todos os deslocamentos têm como referência o SR1 (os 16 bits mais significativos do registrador SR), e, no estado LO, a referência passa a ser o registrador SR0. Estes sinais são especialmente interessantes quando se deseja realizar deslocamentos de palavras de 32 bits, pois pode-se indicar qual das palavra de 16 bits (alta/baixa) se está operando. Este sinal pode ser alterado toda a vez que uma operação de deslocamento for executada.

O deslocador preenche qualquer bit a direita do valor de entrada com zeros e os bits a esquerda com um bit de extensão de sinal (X). Esse bit de extensão pode possuir três possíveis fontes, dependendo do modo com que a instrução é executada. As três

fontes são: o bit mais significativo do valor de entrada, o bit AC do registrador de estado aritmético (ASTAT) ou zero.

A lógica OU / PASS fornece o suporte fundamental para as operações de deslocamento com valores de entrada de dupla precisão (32 bits). Esta unidade permite que se possa combinar, através do operador lógico OR, o resultado da nova operação de deslocamento com o antigo conteúdo do registrador SR. Quando não se deseja realizar este tipo de combinação, está lógica permite que o novo valor do deslocamento seja simplesmente passado, sem modificações, para o registrador SR.

O detetor de expoente extrai o expoente do valor de entrada. Esse detetor de expoente pode operar de três formas diferentes, que determinam como o valor de entrada é interpretado. No estado HI, o valor de entrada é interpretado como um número de precisão simples ou a palavra mais significativa de um valor com dupla precisão. Nesse caso, o detetor de expoente determina o número de bits e gera um código que indica quantos bits o valor de entrada deve ser deslocado à direita, a fim de eliminar todos menos um bit de sinal. Este código é negativo, servindo para indicar quantos bits formam a mantissa do número efetivamente, retirando-se a parte redundante.

A outra forma de operação do detetor de expoente é no estado HIX (HI-*extend*). Neste modo, a entrada é interpretada como o resultado de uma adição ou subtração realizada pela ALU, a qual pode ter gerado um estado de *overflow*. Desta maneira, o detetor de expoente toma o estado de overflow (AV) em consideração. Se o bit AV está setado, será somado à unidade 1 a saída deste detetor, indicando que um bit extra é necessário na mantissa normalizada. Caso o bit AV não esteja ativado, este modo de operação funciona exatamente como no modo HI. Quando é executada uma operação de deteção de expoente, é gerado um valor de sinal de deslocamento, escrito no registrador ASTAT. Esse bit será igual ao bit mais significativo do valor de entrada deste módulo, exceto quando o bit AV estiver ativado.

Na forma de operação do detetor de expoente no estado LO, a entrada é interpretada como a palavra menos significativa de um número de dupla precisão. Neste estado, o detetor de expoente interpreta o bit SS do registrador ASTAT como sendo o sinal do número que se está operando. Neste modo, o registrador de SE é alterado somente se ele já contiver o valor –15, o que iria ocorrer somente quando a parte alta

deste número – que deve ser processada antes da parte baixa – só tiver bits de sinal. A saída do detetor de expoente, neste estado, é também deslocada em –16 para representar o fato de que a entrada é a palavra menos significativa.

A unidade de comparação de expoente é usada para achar o maior valor de expoente em um vetor de valores de entrada. Essa lógica juntamente com o detetor de expoente encontra o expoente de um conjunto de valores. O comparador compara o valor encontrado pelo detetor de expoente com o valor armazenado no registrador SB, e atualiza este registrador somente quando o novo valor de expoente for maior do que o valor já contido neste registrador [4].

5.4. Gerador de Endereços e Sequenciador de Programa

Esta família de DSPs utiliza dois geradores de endereços dedicados e um poderoso sequenciador de programa para garantir um eficiente uso das unidades computacionais. Os geradores de endereços de dados (DAGs) fornecem endereços de memória quando dados são transmitidos para ou dos registradores de entrada ou saída das unidade computacionais. Cada DAG controla quatro ponteiros de endereços (*ix*). Quando um ponteiro é utilizado na forma de endereçamento indireto, ele é pós-modificado pelo valor do registro de modificação (*mx*). A presença de dois geradores de endereços permite que sejam gerados dois endereços simultaneamente, para busca de dois operandos em um mesmo ciclo.

Um registrador de valor de comprimento (*lx*) pode ser associado a cada um dos ponteiros para implementar modos de endereçamento modular, ou seja, *buffers* circulares. O primeiro DAG (DAG1) pode fornecer endereços somente para o barramento de dados da memória de dados. Já o segundo DAG (DAG2) fornece endereços para ambos os barramentos.

O sequenciador de programa fornece endereços de instruções à memória de programa. Esse sequenciador é dirigido pelo registrador de instruções, o qual guarda a instrução que está sendo executada. Esse registrador de instrução introduz um *pipeline* de apenas um nível no fluxo do programa. As instruções são carregadas em um ciclo do processador e no ciclo seguinte, são executadas. Para diminuir o *overhead* de ciclos, este sequenciador suporta desvios condicionais, chamadas a subrotinas e retornos de

subrotinas em um único ciclo. Com um contador de repetições interno e uma pilha de repetições, o processador executa códigos com repetições sem nenhum *overhead.* As operações de repetições não necessitam de instruções de desvios explícitas [4].

5.5. Conclusão

Com esta noção da arquitetura da família de DSPs utilizada [4] e o conhecimento da estrutura do codificador G.723.1 [2,3], podemos passar a descrição do processo de implementação deste codificador para a família ADSP-21XX. Essa descrição enfatizará as técnicas utilizadas, assim como as decisões de projetos adotadas para possibilitar esta implementação de forma flexível, otimizada e robusta.

Capítulo 6

Implementação do Codificador de Voz G.723.1 usando a família ADSP-21XX

6.1. Introdução

Neste capítulo é apresentado o processo de implementação da recomendação G.723.1 no DSP ADSP-2181. Como em qualquer processo de desenvolvimento de software, existem algumas etapas a serem realizadas antes de se iniciar a implementação propriamente dita. Assim, as fases de coleta de requisitos e de definição de estratégia foram realizadas antes do inicio da implementação. Como o objetivo deste trabalho foi o de implementar o codificador da recomendação G.723.1, algumas etapas iniciais já estavam previamente definidas pela referida recomendação.

Desta forma, neste capítulo serão discutidos os processos de coleta de requisitos, definição de estratégias de projeto e implementação do codificador G.723.1. No final deste capítulo será apresentada a metodologia adotada para a realização dos testes de conformidade do algoritmo implementado.

6.2. Coleta de requisitos

Nesta fase procura-se identificar os requisitos do processo de desenvolvimento. Como tratamos de um algoritmo de processamento de sinais, mais exatamente, de um algoritmo de codificação ("compressão") de voz seguindo uma recomendação pré-definida, estes requisitos já estão de certa forma pré-estabelecidos.

Assim, antes de começar a pensar na recomendação G.723.1 e na sua aplicação real, é importante estabelecer uma visão de como um algoritmo de codificação de voz opera. Um codificador de voz é composto basicamente por dois módulos: codificador e decodificador. O módulo codificador é responsável por receber amostras do sinal de voz, analisar estas amostras através de diversas técnicas de processamento de sinais, extraindo seus parâmetros, e organizar esses parâmetros em uma cadeia de bits que será transmitida ao decodificador.

O módulo decodificador deve receber uma cadeia de bits do módulo codificador e, através de técnicas de reconstrução, inversas às técnicas de análise usadas pelo módulo codificador, gerar o sinal de voz reconstruído. É importante lembrar que, em qualquer sistema de compressão de voz, o módulo codificador deve operar independente do módulo decodificador, pois durante o seu funcionamento eles não irão compartilhar informações, sendo que seu único contato ocorrerá através da cadeia de bits.

Considerando que o codificador de voz recomendado pela G.723.1 possui duas taxas de compressão, é importante que o módulo codificador possua um mecanismo capaz de selecionar a taxa de operação desejada do sistema. O módulo decodificador, por outro lado, não precisa deste mecanismo, pois a informação da taxa de operação já se encontra na cadeia de bits.

Desta maneira, uma visão global sobre os requisitos mínimos de operação de um sistema de codificação pode ser estabelecida. Pode-se abstrair que o módulo codificador deverá ter uma porta de entrada, por onde irá receber as amostras de voz; uma chave seletora, que indicará sua taxa de compressão; e uma porta de saída, por onde sairão as cadeias de bits (sinal de voz codificado), como representado na Figura 6.1.

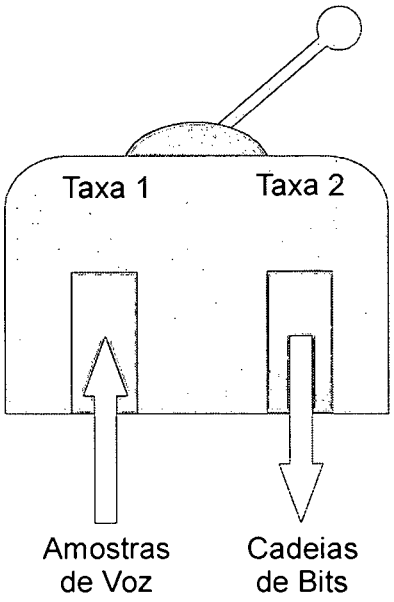

Figura 6.1: Abstração do módulo codificador

Abstraindo o módulo decodificador, pode-se observar que este terá uma porta de entrada, por onde receberá as cadeias de bits e uma porta de saída por onde sairão as amostras de voz reconstruídas. Uma abstração deste módulo pode ser vista na Figura 6.2.

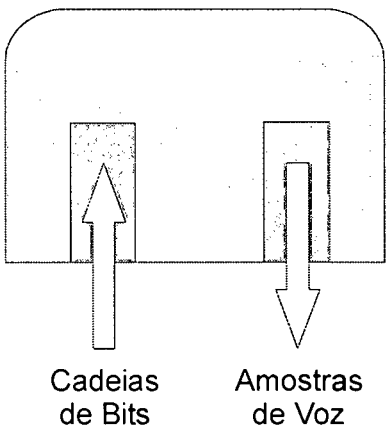

Figura 6.2: Abstração do módulo decodificador

Ainda dentro da fase de coleta de requisitos, deve-se observar que o processo de codificação, agora recomendado pela G.723.1, trabalha com blocos de amostras de voz. Sendo assim, quando as amostras forem entregues ao codificador, deve-se respeitar o tamanho do bloco de voz desta recomendação (30 ms – 240 amostras em 8000 Hz). Sabe-se também que as cadeias de bits geradas pelo codificador deverão conter um

número de bits referentes àquele bloco. Esse número de bits depende da taxa de operação que estiver sendo utilizada (5.3 kpbs = 159 bits/bloco e 6.3 kbps = 189 bits/bloco).

6.3. Projeto de Implementação

O projeto do codificador propriamente dito já está definido na recomendação G.723.1. Assim, a divisão do codificador em blocos, a função de cada bloco, a comunicação dos blocos não podem ser alterada. Entretanto, é possível realizar alterações na forma de funcionamento interno dos blocos que formam o codificador. Dessa forma, durante o processo de implementação, diversos blocos ("funções") do codificador foram modificados, respeitando-se sempre aspectos de entrada e saída, a fim de se conseguir otimizar sua velocidade de execução [36].

Nesta etapa, foi identificado que este codificador (código+dados) está estruturado em quatro blocos básicos: o bloco de instruções (código de execução do codificador), o bloco de memória de variáveis (informações do passado da codificação), o bloco de tabelas (tabelas de quantização, *codebooks*) e o bloco de memória de rascunho (variáveis temporárias) (Figura 6.3).

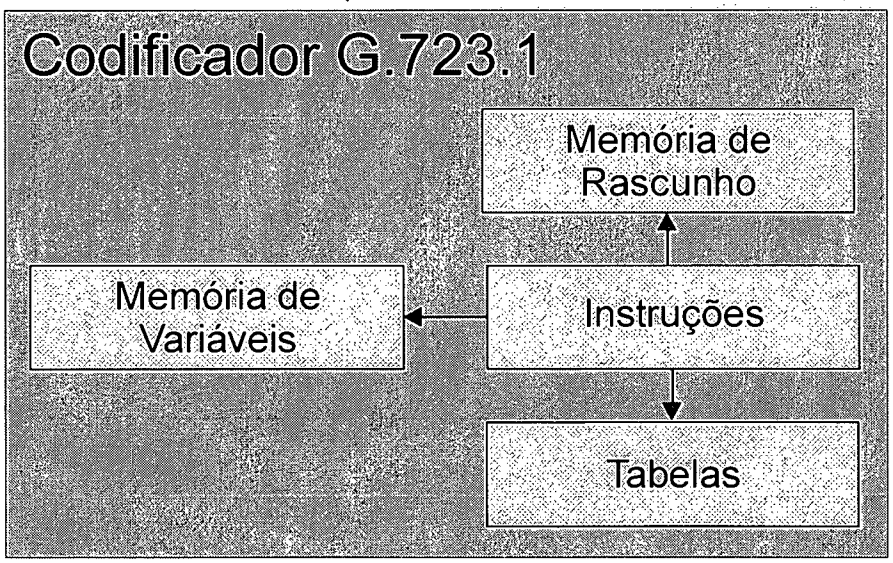

Figura 6.3: Diagrama da estrutura de implementação do codificador G.723.1

Levando em consideração que a plataforma alvo deste projeto é o DSP, foi realizada a divisão do codificador em blocos com base na sua forma de utilização (ex.: o

bloco tabela será armazenado em memória de dados e formado apenas por tabelas, as quais não são alteradas durante o processo de codificação).

Esta forma de divisão possui uma grande vantagem que é a flexibilidade. Esse aspecto aparece quando lembramos que alguns DSPs da família de ADSP-21XX possuem além de blocos de memória RAM (Memória de Acesso Aleatório), blocos de memória ROM (Memória somente para leitura). Desta maneira, quando se desejar utilizar o resultado desta implementação em DSPs com esta característica, é possível alocar os blocos de tabelas e de instruções em memória ROM, e os demais em memória RAM, de forma que sejam aproveitados da melhor forma possível os recursos de memória disponíveis.

6.4. Implementação

Como citado anteriormente, a plataforma alvo desta implementação é a família de DSPs ADSP-21XX. Assim, neste item serão descritos os aspectos de implementação considerados para execução de tal tarefa. Inicialmente, são mostrados quais os tipos de informação, e de que maneira estão organizadas em cada um dos blocos identificados na etapa de projeto. Em seguida, será apresentado o processo de desenvolvimento da técnica de instanciação proposta e utilizada nesta implementação.

6.4.1. Instruções

O bloco de instruções é formado pelo código de execução do processo de codificação. Este bloco foi implementado respeitando-se a organização descrita na recomendação G.723.1, a qual define a divisão em blocos ("funções") e a comunicação entre estes blocos. Entretanto, como a principal meta desta implementação é a construção deste codificador de maneira otimizada, cada um dos blocos foi implementado de forma otimizada, o que implicou em algumas modificações em suas formas internas originais. É importante observar que esta implementação também se aproveitou de alguns detalhes (buffers circulares, multi-instruções) da arquitetura do DSP utilizado.

6.4.2. Memória de Variáveis

Este codificador necessita de uma região de memória na qual estará armazenado o passado do processo de codificação. À região de memória que armazena estes vetores foi dado o nome de "Memória de Variáveis". Esta memória, nesta implementação, utiliza menos de 1 *kwords*.

A seguir é mostrada a organização da memória de variáveis em linguagem C para esta implementação. Esta memória é composta basicamente de dois módulos, um referente ao codificador e outro ao decodificador.

```
typedef  struct   {

   Word16   HpfZdl    ;                        /*   1 word    */
   Word32   HpfPdl    ;                        /*   2 words   */
   Word16   SinDet    ;

   Word16   PrevLsp[LpcOrder] ;                /*  10 words   */

   Word16   PrevWgt[PitchMax] ;                /* 145 words   */
   Word16   PrevErr[PitchMax] ;                /* 145 words   */
   Word16   PrevExc[PitchMax] ;                /* 145 words   */

   Word16   PrevDat[LpcFrame-SubFrLen] ;       /* 120 words   */

   Word16   WghtFirDl[LpcOrder] ;              /*  10 words   */
   Word16   WghtIirDl[LpcOrder] ;              /*  10 words   */
   Word16   RingFirDl[LpcOrder] ;              /*  10 words   */
   Word16   RingIirDl[LpcOrder] ;              /*  10 words   */

   Word32 Err[SizErr];                         /*  10 words   */

   } CODSTATDEF  ;                             /* Total = 628 words */


typedef  struct   {

   Word16   Ecount;                            /*   1 word    */
   Word16   InterGain;                         /*   1 word    */
   Word16   InterIndx;                         /*   1 word    */
   Word16   Rseed;                             /*   1 word    */
   Word16   Park;                              /*   1 word    */
   Word16   Gain;                              /*   1 word    */

   Word16   PrevLsp[LpcOrder];                 /*  10 words   */

   Word16   PrevExc[PitchMax];                 /* 145 words   */

   Word16   SyntIirDl[LpcOrder];               /*  10 words   */
   Word16   PostFirDl[LpcOrder];               /*  10 words   */
   Word16   PostIirDl[LpcOrder];               /*  10 words   */

   } DECSTATDEF;                               /* Total = 191 words */
```

É importante lembrar que, como a implementação foi realizada completamente em linguagem *assembler*, estas definições de estruturas em C servem apenas como modelo de referência, do qual aproveita-se somente a organização (posição e tamanho das variáveis).

6.4.3. Tabelas

As tabelas de dados fixas, ou simplesmente tabelas, são estruturas de dados, carregadas na memória, que contêm os dados que alimentam todas as rotinas do codificador. Entre esses dados, pode-se identificar vetores de quantização, *codebooks* e tabelas para interpolação em cálculos de senos e cosenos.

Nesta implementação da recomendação G.723.1, as tabelas ocupam 11 *kwords* de memória.

6.4.4. Memória de Rascunho

Na implementação de um codificador de voz em DSP, não existe a figura do sistema operacional, o qual, quando presente, realiza as tarefas de gerenciamento de memória. Assim, neste tipo de implementação, diferentemente da implementação em sistemas de propósito geral em que o sistema operacional está presente, o gerenciamento de memória tem de ser realizado pelo próprio algoritmo de codificação.

Nesta implementação, optou-se então pela definição de uma área de memória de rascunho, que tem a função de armazenar todas as variáveis temporárias (variáveis das rotinas e áreas de memória alocadas dinamicamente) necessárias durante o processo de codificação.

Esta forma de utilização da memória temporária (memória de rascunho), entretanto, exige, entretanto que sejam observados alguns cuidados fundamentais, pois, como não existe um esquema de gerenciamento de memória, é necessário, antes de usar a memória, garantir que ela esteja livre. Para se ter esta garantia, uma solução encontrada foi realizar de forma estática a alocação das variáveis das rotinas e de toda a memória dinâmica. Dessa forma, todas as variáveis das rotinas e os blocos de memória alocados dinamicamente durante o processo de codificação passaram a possuir uma região exclusiva de memória com um endereço fixo (Figura 6.4).

Esta forma de utilização da memória é bastante simples, porém, tem o inconveniente de estar sempre ocupando uma grande quantidade de memória, independente de estar ou não utilizando. Desta maneira, este esquema tem um alto custo computacional, quando comparado (no critério de ocupação de memória) com esquemas que utilizam gerenciamento de memória.

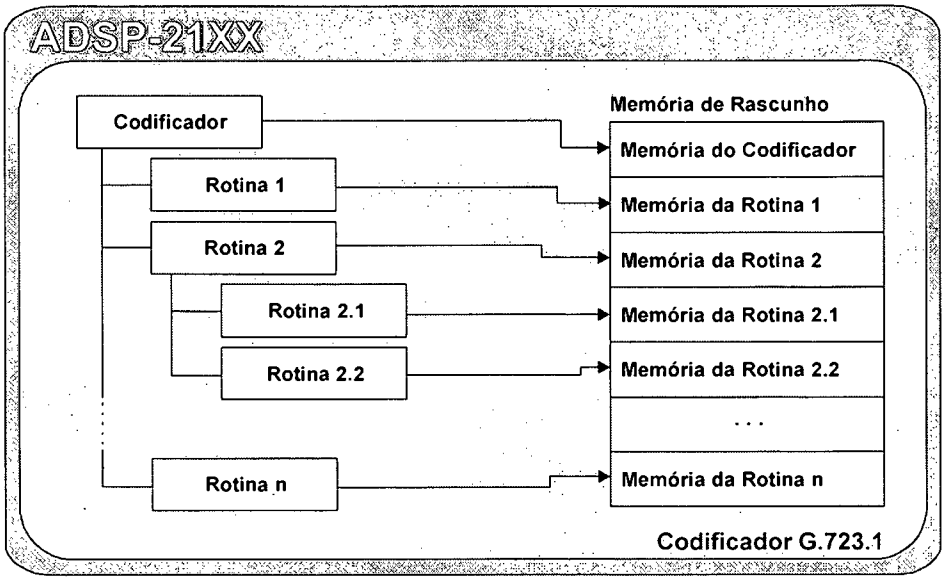

Figura 6.4: Utilização da memória de rascunho estática

Para buscar minimizar o tamanho da memória de rascunho, foi utilizado um esquema no qual esta área de memória é dividida em blocos. Neste esquema, cada um dos níveis de rotinas utiliza um bloco de memória (Figura 6.5).

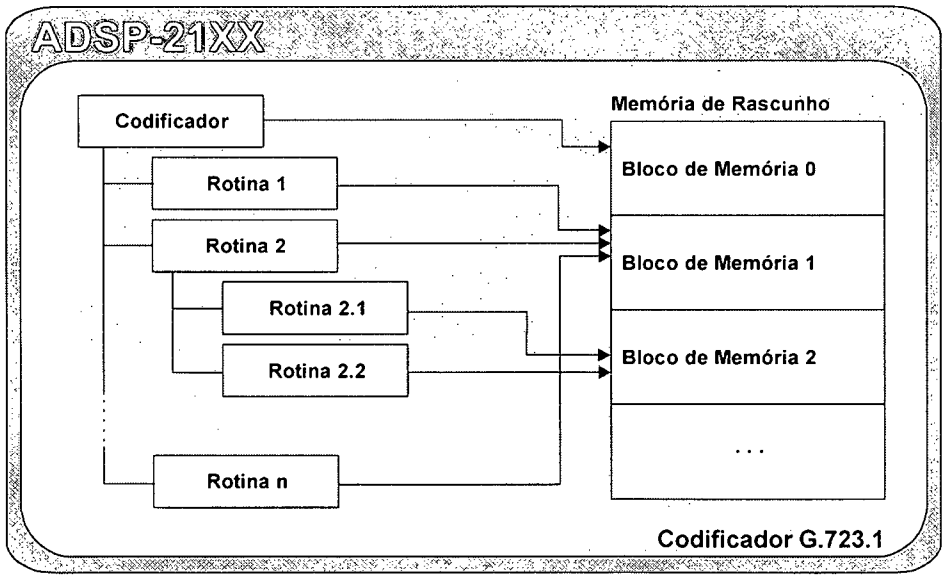

Figura 6.5: Memória de rascunho com divisão em blocos

Este esquema de divisão/utilização de memória de rascunho não proporciona a ocupação ótima da memória, no entanto não impõe o custo de um sistema de gerenciamento de memória. Este esquema apresentou uma relação de custo x benefício

boa, pois o custo referente ao "desperdício" de memória foi menor do que o custo de complexidade da utilização de um algoritmo de gerenciamento.

6.5. Técnica de Instanciação

Até agora, foram tratadas somente as técnicas e artifícios utilizados na implementação do codificador. Contudo, um outro aspecto muito importante neste tipo de implementação é o suporte à instanciação que depende fundamentalmente dos recursos computacionais disponíveis no DSP escolhido. A instanciação de algoritmos em DSP pode ser definida como a capacidade de criação de diversos processos, dentro de um mesmo DSP, que executem uma mesma tarefa.

A técnica mais simples de instanciação consiste na utilização de uma cópia do processo para cada instância (Figura 6.6), em que as instruções, as tabelas, a memória de rascunho e a memória de variáveis serão repetidas para cada uma das cópias. Um aspecto importante desta técnica é que o código de execução do codificador utiliza um esquema de endereçamento estático, não sendo necessária, a implementação de nenhum mecanismo de endereçamento de memória dinâmico, para acessar os dados de cada uma das instâncias.

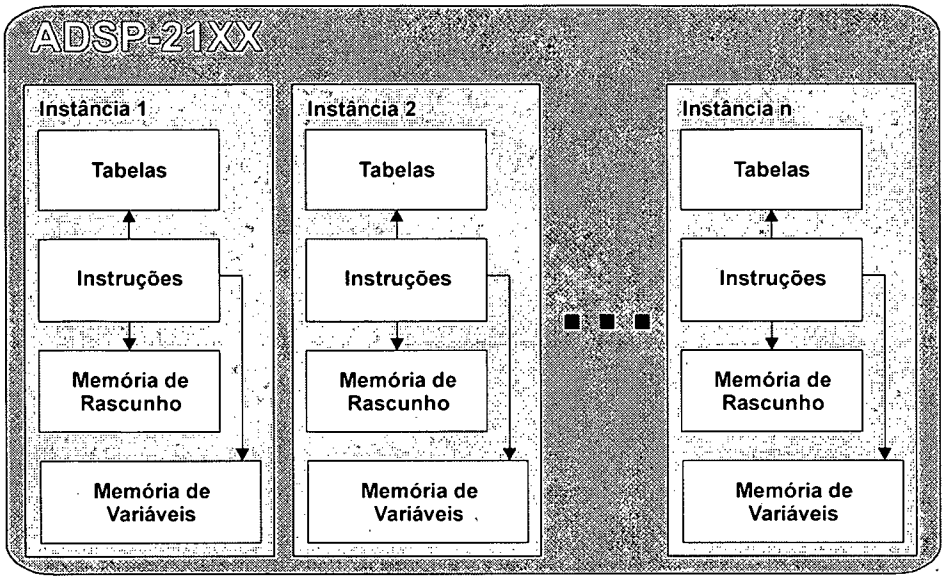

Figura 6.6: Instanciação através de cópia

Esta técnica de instanciação associa um custo computacional muito alto. Isto ocorre porque cada nova instância criada necessita que seja colocada uma nova cópia completa do codificador. Tomando como exemplo o ADSP-2181 que possui 32 K Words de memória, e esta implementação da recomendação G.723.1 que necessita 19 K Words, pode-se observar que não é possível a utilização de mais de uma instância neste modelo de DSP.

Uma variação da primeira técnica consiste na retirada das tabelas e da memória de rascunho do escopo da instância. Nesta técnica, cada instância será composta somente por instruções e pela memória de variáveis. Esta técnica continua sendo de aplicação bastante simples, pois, assim como a primeira, mantém um esquema de endereçamento estático (Figura 6.7).

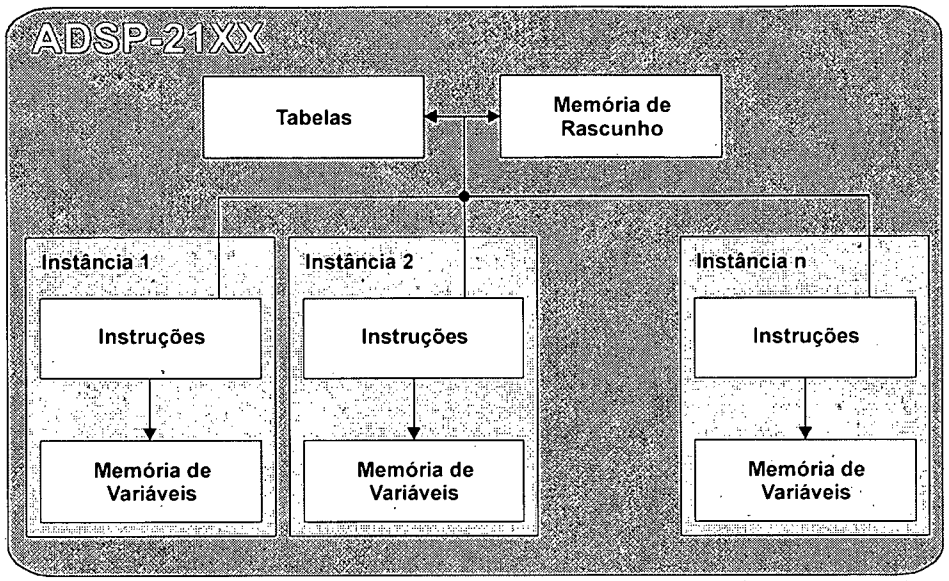

Figura 6.7: Técnica de instanciação por cópia com utilização de memória de rascunho e tabelas globais

A abordagem ilustrada pela Figura 6.7 já apresenta uma melhora considerável no custo de instanciamento, uma vez que as tabelas e a memória de rascunho representam mais de 50 % (11 k words) da utilização de memória neste DSP. Na tabela 6.1, é possível observar o número de instâncias suportado por um alguns DSPs da família ADSP-218X, utilizando a técnica de instanciação por cópia com utilização de memória de rascunho e tabelas globais.

Tabela 6.1: Capacidade de instanciação da memória de alguns DSPs, utilizando-se a técnica de instanciação por cópia com utilização da memória de rascunho e tabelas globais

	ADSP-2181	ADSP-2187	ADSP-2189
Quantidade de memória do DSP (*kwords*)	32	64	80
Necessidade de memória fixa do codificador G.723.1 (*kwords*)	11	11	11
Necessidade de memória por instância do codificador G.723.1 (*kwords*)	8	8	8
Número máximo de instâncias suportada na memória.	**2**	**6**	**8**
Ocupação total da memória utilizando-se o número máximo de instâncias (kwords)	27	59	75

Esta técnica de instanciação, apesar de ser adequada ainda não proporciona uma utilização ótima da memória, pois as instruções ainda precisam ser copiados para cada instância criada. Desta maneira, surge uma nova técnica de instanciação, a qual retira do escopo das instâncias também as instruções. No entanto nesta técnica, é necessária a utilização de mecanismos de endereçamento dinâmico, para que seja possível acessar, através de um conjunto comum de instruções, a memória de variáveis de cada uma das instância criadas (Figura 6.8).

Uma solução trivial, para a implementação de um esquema de endereçamento dinâmico, é a criação de uma variável de *offset,* a qual irá indicar a localização do início de cada bloco de memória de variáveis, de cada uma das instâncias. A modificação da implementação exigida para utilização desta técnica não é complexa, sendo necessário somente somar seu conteúdo a um endereço, agora relativo, das variáveis localizadas dentro da memória de variáveis. Um outro cuidado necessário é o de carregar a variável de *offset* com um valor apropriado, antes de executar rotinas de codificação.

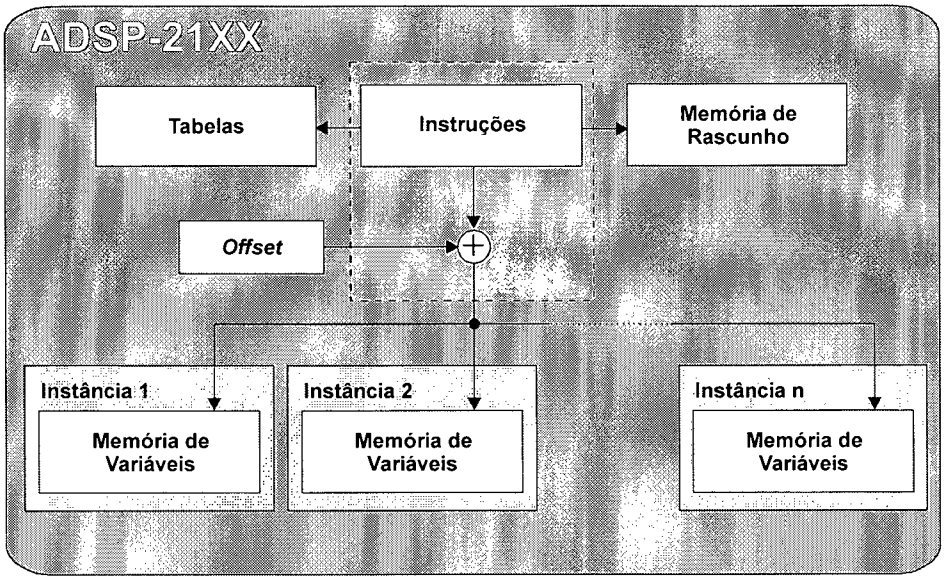

Figura 6.8: Instanciação da memória de variáveis utilizando esquema de endereçamento com base em uma variável de *offset*

Desta forma, esta técnica consegue proporcionar uma ocupação ótima da memória, sem a necessidade de criação, em esquemas de instanciação, de nenhuma ocupação de memória redundante. Considerando a implementação da recomendação em questão, teremos, com esta técnica, uma ocupação de menos de 1 *kword* por instância criada, referente ao bloco de memória de variáveis. Assim, pode-se observar na Tabela 6.2 o número de instâncias possíveis de serem alocadas na memória de alguns DSPs da família ADSP-218X.

Tabela 6.2: Capacidade de instanciação da memória de alguns DSPs, utilizando-se a técnica de instanciação da memória de variáveis

	ADSP-2181	ADSP-2187	ADSP-2189
Quantidade de memória do DSP (*kwords*)	32	64	80
Necessidade de memória fixa do codificador G.723.1 (*kwords*)	17	17	17
Necessidade de memória por instância do codificador G.723.1 (*kwords*)	> 1	> 1	> 1
Número máximo de instâncias suportada na memória.	**15**	**47**	**63**
Ocupação total da memória utilizando-se o número máximo de instâncias (kwords)	32	64	80

Esta técnica, entretanto, necessita calcular um endereço todas as vezes que precisar realizar um acesso a uma variável dentro da memória de variáveis. Considerando que estas variáveis são acessadas mais de uma vez durante a execução do codificador, torna-se necessário recalcular diversas vezes um mesmo endereço. Assim, utiliza-se nesta implementação um algoritmo de gerenciamento de instâncias, o qual é responsável pelo fornecimento dos endereços das variáveis da memória de variáveis ao codificador (Figura 6.9).

Baseado nas discussões apresentadas neste capítulo, o algoritmo de gerenciamento de instâncias utilizado na implementação funciona proposta da seguinte maneira:

- É criado um vetor de ponteiros que contém os endereços de todas as variáveis existentes dentro da memória de variáveis.
- Este vetor é inicializado com os endereços da primeira instância.
- Toda vez que o codificador necessita acessar uma variável dentro desta memória de variáveis, ele busca o endereço neste vetor.
- Quando se troca a instância em operação, os endereços são todos recalculados e passam a apontar a memória de variáveis da nova instância. Essa troca é feita chamando-se uma função do bloco de gerenciamento de instâncias.

Entre as vantagens deste esquema de endereçamento, pode-se destacar: os endereços das variáveis da memória de variáveis precisam ser calculados apenas uma vez, o programador da aplicação que utiliza este codificador não precisa conhecer nenhum detalhe da organização do codificador (nome da variável de *offset*, localização dos blocos de memória de variáveis) e este esquema de instanciação tem custo zero quando se utiliza apenas uma instância.

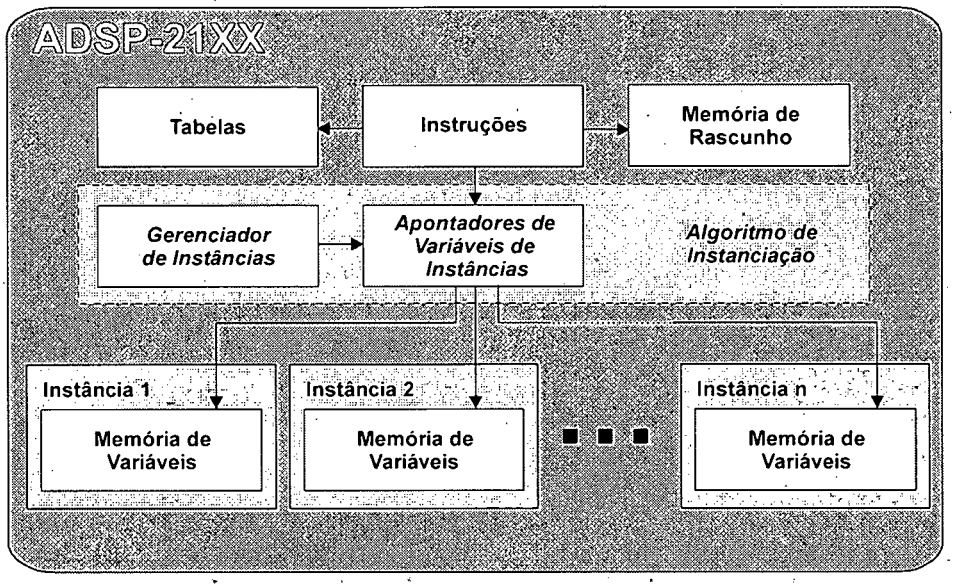

Figura 6.9: Esquema de instanciação utilizando o algoritmo de gerenciamento de instâncias

Esta técnica possui a mesma ocupação de memória da técnica mostrada anteriormente, sendo desta forma, adequada para utilização nesta implementação.

Capítulo 7

Resultados Obtidos

7.1. Introdução

Neste capítulo, serão apresentados os resultados alcançados na implementação em DSPs (mais especificamente na família ADSP-21XX) do codificador de voz padronizado pela recomendação G.723.1. Serão tratados inicialmente os aspectos operacionais desta implementação, passando-se em seguida a discutir aspecto de custo de implementação. São salientados também os aspectos de conformidade desta implementação com o que foi definido pela recomendação utilizado, sendo também mostrado uma comparação desta implementação com as demais disponíveis no mercado.

7.2. Aspectos Operacionais

Como apresentado no Capítulo 6, a implementação proposta visa obter uma implementação flexível, robusta e eficiente da recomendação G.723.1 para a família de DSPs ADSP-21XX.

O aspecto **flexibilidade** foi alcançado através das técnicas de organização utilizadas na montagem desta implementação. Essas técnicas definem que o codificador de voz é dividido em blocos, de acordo com a natureza das informações que o compõem. Por exemplo, todas as tabelas de dados, que alimentam o processo de codificação, podem ser colocadas em um bloco, pois uma de suas principais características é o fato de elas serem utilizadas somente para leitura e nunca serem modificadas. Assim, esta divisão permite que este codificador possa ser facilmente adaptado a qualquer configuração de memória disponível em DSPs da família ADSP-21XX.

No aspecto de **eficiência** da implementação, podemos destacar a economia de memória e a velocidade de operação. A economia de memória foi alcançada através do emprego de técnicas de organização aplicadas diretamente sobre o bloco de memória de rascunho, que, representava o maior problema de ocupação desnecessária de memória. Tais técnicas, conforme explicado no capítulo anterior, conseguiram minimizar a quantidade de memória utilizada nesse bloco. A eficiência na velocidade de operação foi conseguida através da utilização, nas computações exigidas pelas rotinas que compõem este padrão de codificação, de todos os recursos disponíveis da arquitetura do DSP. A obtenção desse aproveitamento (eficiência) só foi possível devido ao conhecimento, por parte do projetista, dos detalhes de funcionamento da arquitetura do DSP e do padrão de codificação.

A **robustez** desta implementação foi alcançada através da conformidade de tal implementação com o padrão definido pela recomendação. Este aspecto será detalhado mais adiante.

Finalmente, ainda sobre aspectos operacionais de tal implementação pode-se destacar outros dois resultados desta implementação, isto é, a facilidade de uso e a capacidade de instanciação.

7.3. Custo de implantação

Como reflexo dos resultados obtidos com esta implementação, podemos destacar o baixo custo de implantação deste algoritmo. Este fato pode ser confirmado quando consideramos os seguintes aspectos: o DSP de ponto fixo em 16 bits é atualmente a

plataforma de *hardware* de menor custo para implementação de codificadores de voz (seu custo é 5 ou mais vezes menor do que um DSP em ponto flutuante, ou alguma outra plataforma genérica - PC); e a capacidade de instanciação desta implementação permite que mais de um canal de codificação seja utilizado em um mesmo DSP, dividindo, desta forma, o custo do *hardware*, pelo número de instâncias suportadas por um determinado DSP.

7.4. Conformidade da implementação

Para garantir a conformidade desta implementação, foram realizadas diversas baterias de testes, conforme especificado pela recomendação G.723.1 [2,3]. Tais testes foram realizados através da excitação do módulo codificador e decodificador com vetores de testes padrões fornecidos pelo ITU-T. Nesta fase de testes, sempre que erros eram identificados, voltava-se à fase de implementação para correção do problema, e a fase de testes era reiniciada.

Desta maneira, pôde-se garantir a conformidade desta implementação com o padrão definido pela recomendação G.723.1.

7.5. Soluções Semelhantes Disponíveis no Mercado

Devido ao apelo deste padrão de codificação, diversos laboratórios no mundo (empresas especializadas em implementação de codificadores de voz em DSP) se especializaram neste tipo de implementação. Pode-se encontrar no mercado diversas soluções, de vários laboratórios em diferentes plataformas. Desta maneira, como uma forma de comparação, foram pesquisados alguns laboratórios que possuíam uma solução com as mesmas características propostas por este trabalho (implementação da recomendação G.723.1 em DSP da família ADSP-21XX). Assim, a Tabela 7.1 a seguir apresenta uma comparação do resultado deste trabalho com o resultado de alguns outros laboratórios.

Tabela 7.1: Comparação entre os parâmetros da implementação proposta com outras implementações disponíveis no mercado com características semelhantes

Nome do Laboratório	MIPS[1] (5.3/6.3)	Memória utilizada	Memória por canal adicional
Analogical Systems	19.44[2]	21.98 *kwords*	0.98 *kwords*
Adaptative Digital Technologies	16.4/16.7	21.4 *kwords*	1.4 *kwords*
VoCAL Technologies Ltd.	20.0/28.0	NI[3]	NI[3]
SPA: Signal Processing Associates	22.89/24.69	21.36 *kwords*	NA[4]
Implementação Proposta	**16.5/22.6**	**19 *kwords***	**0.95 *kwords***

Pode-se observar na Tabela 7.1, a complexidade computacional (expressadas em MIPS), a ocupação total de memória, e a necessidade de memória adicional requerida por cada canal de codificação instanciado de cada uma das diferentes implementações disponíveis no mercado. Podemos notar que a complexidade computacional da implementação proposta encontra-se entre as menores listadas na tabela. Com relação a ocupação de memória constata-se que o resultado obtido pela abordagem proposta deu origem à implementação de menor custo de ocupação de memória (tanto a memória geral quanto a memória adicional por canal).

[1] Este campo mostra o número de instruções por segundo (MIPS) necessário para executar o processo de codificação e decodificação do padrão proposto pela recomendação G.723.1 para as taxas de 5.3 e 6.3 kbits/s.

[2] Este laboratório não informou se este valor de MIPS é referente ao codificador operando em taxa alta ou baixa.

[3] NI indica que este fabricante não informou o parâmetro deste campo.

[4] NA indica que o suporte à instanciação não está disponível na implementação deste laboratório.

Capítulo 8

Conclusões

Esta dissertação teve como objetivo estudar e apresentar uma proposta de implementação de um codificador de voz padronizado em DSP. O codificador de voz escolhido foi o da recomendação G.723.1, devido às características desse padrão serem muito atraentes para as atuais aplicações que necessitam codificação de voz, principalmente em Internet. Como plataforma de *hardware*, foi escolhido a família de DSPs ADSP-21XX. Essa escolha considerou principalmente os aspectos de custo e capacidade computacional.

A implementação proposta organiza o codificador em quatro blocos: instruções, tabelas, memória de variáveis e memória de rascunho. Essa organização apresenta como principal vantagem a flexibilidade de utilização. Essa característica torna-se muito vantajosa em uma implementação que visa atender uma família de DSPs.

Uma outra característica da solução proposta é o aspecto da eficiência de implementação, na qual se teve como objetivo a redução da utilização de memória e um

aumento na velocidade de processamento. Tais fatores resultam em uma redução de carga computacional que se reflete diretamente no aspecto de custo do sistema.

Nesta implementação, ainda se pode destacar que a robustez e/ou conformidade com recomendação G.723.1 foi plenamente alcançada. Isso foi possível graças à realização de um procedimento exaustivo de testes recomendados pelo ITU-T.

Podemos afirmar assim que a implementação proposta neste trabalho apresenta um alto grau de competitividade com outras soluções disponíveis no mercado, o que só foi possível devido ao rigoroso procedimento de desenvolvimento e testes mencionados anteriormente.

Um outro aspecto incorporado a esta solução foi a capacidade de instanciação. Essa característica reforça ainda mais os aspectos de flexibilidade, eficiência, robustez e baixo custo da implementação proposta. Isto ocorre porque a instanciação permite que esta implementação tenha a possibilidade de executar as tarefas de codificação e decodificação de diversos canais de comunicação, simultaneamente, dentro de um mesmo DSP. Um outro fator muito favorecido pela capacidade de instanciação é o poder de expansão que é dado aos sistemas que utilizam esta implementação.

Para a elaboração de trabalhos futuros, podemos sugerir a implementação deste codificador em outras plataformas de hardware (famílias de DSPs da Texas Instruments), assim como uma implementação em um processador de propósito geral (*Intel Pentium*). Um outro trabalho muito interessante que pode ser desenvolvido é a avaliação da uma implementação deste mesmo codificador em um DSP de ponto flutuante comparando-a com implementações em uma plataforma de propósito geral. Essa avaliação pode levar em consideração os aspectos de número máximo de canais de codificação, custo computacional de cada uma das implementações, vantagens e desvantagens de cada uma das plataformas como também o custo final do sistema.

Referências Bibliográficas

[1] A. Gersho e R. M. Gray, *Vector Quantization and Signal Compression.* Dordrecht, Holland: Kluwers Academic Publishers, 1991.

[2] ITU-T Rec. G.723.1, *Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s*. Março 1996.

[3] ITU-T Rec. G.723.1, *Annex A: Silence compression scheme,* Novembro 1996.

[4] *ADSP-2100 Family – User's Manual.* 3ª Edição, Analog Devices, Setembro 1995.

[5] N. S. Jayant e P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs NJ: Prentice-Hall, 1984.

[6] Military agency for standardization (MAS), North Atlantic Treaty Organization (NATO), "Technical standards for analogue-digital conversion of voice signals using continously variable slope and delta modulation (csvd)", 1984. STANAG 4380 2nd darft, edition 1.

[7] W. B. Kleijn e K.K. Paliwal, *Speech Coding and Synthesis*. Elsevier Science B. V., 1998.

[8] A. M. Kondoz, *Digital Speech: Coding for Low Bit Rate Communications Systems*. John Wiley, Março 1998.

[9] T. E. Tremain, *The government standrard linear predictive coding algorithm*, Speech Technology, pp. 40-49, Abril 1982.

[10] CCITT Study Group XVIII, *Temporary document 18, draft recommendation, g.78zz*. Novembro 1983.

[11] R. V. Cox, *Three New Speech Coders from the ITU Cover a Range of Application.* IEEE Communication Magazine, September 1997.

[12] R. Pickholtz, L. Milstein e D. Schilling, *Spread spectrum for mobile communications*. IEEE Trans. Vehic. Techn., vol. 40, n°. 2, pp.313-322, 1991.

[13] S. Furui e M. M. Sondhi (Editores), *Advances in Speech Signal Processing.* Marcel Dekker, Inc, 1991.

[14] M. E. Perkins, K. Evans, D. Pascal e L. A. Thorpe, *Characterizing the Subjective Performance of the ITU-T 8 kb/s Speech Coding Algorithm – ITU-T G.729.* IEEE Communication Magazine, September 1997.

[15] J. Bodie, *Digital Signal processor.* Bell System Technical Journal, vol. 60, n° 7, pp. 1431-1439, 1981.

[16] S. Dimolitsas e J. Phipps, *Experimental quantification of voice transmission quality of mobile-satellite personal communications systems.* IEEE Journal on Selected Areas in Comm., vol. 13, n° 2, 1995.

[17] P. Vary, R. Hoffman, K. Hellwig e R. Sluyter. *A regular-pulse excited linear predictive code.* Speech Comm., vol. 7, n° 2, pp.209-215, 1988.

[18] P. Denes w E, Pinson, *The Speech chain.* Garden City, New York: Anchor Books, 1963.

[19] W. B. Kleinj, R. P. Ramachandran e P. Kroon, *Interpolation of the pitch-predictor parameters in analysis-by-synthesis speech coders.* IEEE Transaction Speech and Audio Process., vol 2, no. 1, pp. 42-54, 1994.

[20] ITU-T Rec. G.711, *Pulse Code Modulation (PCM) of Voice Frequencies.* 1988.

[21] ITU-T Rec. G.721, *Adaptive Pulse Code Modulation (ADPCM).* 1988.

[22] ITU-T Rec. G.723. *Adaptive Pulse Code Modulation (ADPCM).* 1988.

[23] ITU-T Rec. G.722, *7 kHz Audio - Coding Within 64 kbit/s.* 1988.

[24] ITU-T Rec. G.726, *40, 32, 24, 16 kbit/s Adaptive Differencial Pulse Code Modulation (ADPCM).* 1990.

[25] ITU-T Rec. G.727, *5-, 4-, 3- and 2-bits Sample Embedded Adaptive Pulse Code Modulation (ADPCM).* 1990.

[26] ITU-T Rec. G.728, *Coding of Speech at 16 kbit/s Using Low-Delay Code Excited Linear Prediction.* Setembro 1992.

[27] ITU-T Rec. G.729, *Coding of Speech at 8kbit/s using Conjugate-Structure Algebraic-Code-Excited Linear Predictive (CS-ACELP) Coding.*

[28] A. Benyassine, E. Shlomot e Huan-Yu Su, *ITU-T Recommendation G.729 Annex B: A Silence Compression Scheme for Use with G.729 Optimized for V.70 Digital Simultaneous Voice and Data Application.* IEEE Communication Magazine, Setembro 1997.

[29] ITU-T Rec. H.320, *Narrow-band visual telephone systems and terminal equipment.* Março 1996.

[30] ITU-T Rec. H.324, *Terminal for low bit rate Multimedia Communication.* Março 1996.

[31] ITU-T Rec. H.323, *Packet-based multimedia communications systems.* Setembro 1999.

[32] ITU-T Rec. G.712, *Transmission Performance Characteristics of Pulse Code Modulation.* Setembro 1992.

[33] ADSP-2100 Family: Assembler Tools & Simulator Manual, Novembro 1994.

[34] Digital Signal Processing Appliacation using the ADSP-2100 Family, vol. 1, Prentice Hall, 1990.

[35] Digital Signal Processing Appliacation using the ADSP-2100 Family, vol. 2, Prentice Hall, 1992.

[36] M. Taka, R. Maruta e S. Unagami, *DSP Implementations of Sophisticated Speech Codecs.* IEEE Journal on Selected Areas in Communications. Vol. 6, n° 2, pp. 274-282, Fevereiro 1988.