UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA DE CAMPINAS

Este exemplar corresponde a redação final da dissertação defendida por Luciano Baracho Rocha e aprovada pela comissão julgadora em 19/12/1984

# FILTRAGEM DIGITAL COM MICROPROCESSADOR UTILIZANDO ARITMÉTICA DISTRIBUÍDA

Autor: Luciano Baracho Rocha

Orientador: Dr. Normonds Alens

106/84

Dissertação apresentada à Faculdade de Engenharia da Campinas para a obtenção do grau de MESTRE EM ENGENHARIA ELÉTRICA.

DEZEMBRO - 1984

À minha esposa

Cristina

Aos meus pais

Roberto e Diocelle

Aos meus sogros

Hélio e Maria

## AGRADECIMENTOS

Inicialmente agradeço ao Professor Dr. Normonds Alens pela paciência e segurança com que orientou este trabalho, fazendo com que o mesmo atingisse os resultados almejados.

Agradeço a todos aqueles que tornaram possível o êxito deste trabalho, em especial:

- Aos Profs. Ivo Mezzadri e Ataíde Moacyr Ferrazza, ex-Diretor e atual Diretor Geral do CEFET-PR, respectivamente;

- Ao Prof. Aramis Demeterco, Diretor de Relações Empresariais, pela coordenação do PICD durante o meu afastamento para realizar o Curso de Mestrado e pelo apoio e incentivo para a conclusão da tese;

- Aos Profs. Engº Joarez Vrubel ex-Coordenador do Curso Técnico de Eletrônica, Bel. Wanderley Veiga, ex-Chefe do Departamento de Eletrônica, Engº Álvaro Luiz Stelle, atual Chefe do Departamento de Eletrônica, Engº João Cruz Erbano Filho, Coordenador do Curso de Engenharia Industrial Elétrica, modalidade Eletrônica, por garantirem as condições e carga horária necessárias;

- Ao Prof. Dr. Wang Binseng, por permitir a utilização dos recursos do laboratório de Setor de Engenharia Médica do CEB/UNICAMP;

- Ao Prof. Engº Eduardo Tavares Costa pelas valiosas discussões e pelo apoio na realização da parte prática da tese junto ao Setor de Engenharia Médica do CEB/UNICAMP;

- Aos colegas e companheiros do Grupo de Estudos de Microprocessadores do Departamento de Eletrônica pelas críticas e sugestões;

- À Maria de Lourdes Malta Serra, pela datilografia e ao Sr. Carlos Roberto dos Santos pelos desenhos;

- Aos parentes e amigos, cujo estímulo e dedicação jamais serão esquecidos.

## RESUMO

Neste trabalho apresentamos uma síntese das características e das técnicas de projeto de filtros digitais de resposta ao impulso finita com fase linear. Em seguida, considera-se o problema da filtragem digital de sinais em tempo real. Para isso descreve-se um algoritmo de cálculo conhecido como aritmética distribuída. Sugere-se uma estrutura de sistema para implementar este algoritmo com microprocessador. A estrutura de programação do sistema é discutida. Coeficientes de filtros de resposta ao impulso finita e fase linear são calculados e suas respostas em freqüência simuladas. Um circuito não recursivo é implementado, atingindo uma taxa de amostragem da ordem de 14,7 kHz. A estrutura projetada é flexível, permitindo ensaiar diversos filtros apenas com a mudança da tabela na memória. Vários filtros são ensaiados e é realizada a comparação entre as respostas em freqüência simulada e medidas. Discute-se a possibilidade do aumento da taxa de amostragem e do número de coeficientes do filtro. Finalmente, são alinhadas sugestões para futuros progressos.

## ABSTRACT

This work describes a synthesis of the characteristics and design of finite impulse response digital filters with linear phase. Forward we consider the digital filtering problem of real time signals. For this purpose we describe an algorithm of calculus known as distributed arithmetic. We suggest a system structure to implement this algorithm with a microprocessor. The system programation structure is discussed. Coefficients of finite impulse response filters with linear phase are calculated and its frequency responses are simulated. A not recursive circuit is implemented, getting a sampling rate in the order of 14,7 kHz. The projected structure is flexible allowing to test several filters with a change of a table in its memory. Some filters are tested and is accomplished the comparation between the frequency responses simulated and measured. We discuss the possibility of the increasing of the sampling rate and the filter coefficients. Finally we demonstrated suggestions for future progress.

ÍNDICE

# INTRODUÇÃO

## 1. IDÉIA GERAL DA TESE

A aplicação de técnicas digitais ao processamento de sinais elétricos faz parte de uma área da Engenharia Elétrica conhecida como processamento digital de sinais. Esta área vem apresentando um interesse crescente no Brasil, sendo cada vez maior o número de textos, artigos e cursos oferecidos, tanto no meio universitário como empresarial. Este interesse se justifica pelo fato de haverem atualmente inúmeras aplicações do processamento digital de sinais na solução de problemas distintos. Nosso primeiro contato com a área foi através de duas disciplinas do curso de Mestrado em Engenharia Elétrica da Faculdade de Engenharia de Campinas, ministradas pelo Dr. Normonds Alens[1]. Surgiu então a idéia de desenvolver a dissertação de Mestrado sobre o processamento digital de sinais, para contribuir com a atualização e desenvolvimento destes conteúdos no Brasil.

Um dos problemas do processamento digital é a filtragem de sinais utilizando os circuitos digitais ("hardware") disponíveis. Quando se pretende realizar a filtragem em "tempo real", as formas básicas de implementação de filtros mostram-se ineficazes pois o tempo necessário para calcular uma saída torna-se muito elevado devido ao grande número de operações envolvidas. Para contornar este problema Little [8] sugere um algo-

---

(1) Orientador da tese.

ritmo conhecido como aritmética distribuída. Por sua vez Farhang-Boroujeny e Hawkins [6] aplicam este algoritmo conjugado ao uso de microprocessadores para controlar o processo de filtragem. Enfoque teórico semelhante ao de Little é o "vetor multiplicador" de Peled e Liu [12], e uma estrutura de circuito usando esta teoria com microprocessadores é sugerido por Allen e Holt [1].

Decidimos enfocar especificamente neste trabalho o problema da implementação de filtros digitais em tempo real, utilizando os algoritmos citados e fazendo uso também das facilidades de um microprocessador. Com este objetivo apresentamos no capítulo I uma rápida revisão da teoria de filtros digitais de resposta ao impulso finita com fase linear e das suas principais técnicas de projeto. No capítulo II seguimos com uma descrição da teoria da aritmética distribuída e da estrutura de sistema com microprocessador para realizar a filtragem de acordo com esta teoria. A seguir, no capítulo III considerou-se o problema do cálculo de coeficientes de filtros e da simulação de sua resposta em freqüência. Neste capítulo incluimos algum auxílio computacional para o projeto de filtros. No capítulo IV apresentamos o circuito do filtro digital projetado, incluindo medidas para alguns filtros cujos coeficientes foram calculados e simulados no capítulo III. No capítulo V concluímos com uma análise do projeto e medidas, indicando as modificações possíveis e as sugestões para futuros progressos.

## 2. BREVE VISÃO HISTÓRICA SOBRE O PROCESSAMENTO DIGITAL DE SINAIS[1]

A aspiração ao emprego de técnicas digitais no processamento de sinais vem desde depois da II Guerra Mundial, ou até mais cedo. Entretanto, as limitações da tecnologia existente impediram seu rápido desenvolvimento, favorecendo o uso de técnicas analógicas. Entretanto, as ferramentas matemáticas usadas no processamento digital de sinais são muito antigas, com raízes nos séculos 17 e 18. As teorias de controle, teoria da amostragem e transformada "z" já eram bastante conhecidas e estudadas como disciplinas no período 1945-1950. A partir de 1960 ocorre o aparecimento de uma teoria formal do processamento digital de sinais. Paralelamente o potencial de tecnologia de circuitos integrados torna-se apreciável, reduzindo o custo, velocidade e tamanho dos componentes. Destaca-se como contribuição notável para esta área os trabalhos de Kaiser (nos laboratórios da Bell) na área do projeto de filtros digitais e síntese.

Apesar do progresso realizado, as técnicas digitais ainda possuíam duas limitações: uma, era o fato de não permitirem o processamento dos sinais em tempo real, e o processamento digital era estudado como simples extensão do processamento analógico.

A partir de 1965, com o artigo de Cooley-Tukey sobre um método rápido de computar a transformada de Fourier dis-

---

(1) Nesta seção nos baseamos principalmente em Oppenheim e Schafer [11], Rabiner e Gold [15].

creta (conhecido por FFT), evidenciou-se a maior economia do emprego do método digital na análise de espectro. Isto acelerou o seu uso na área de espectro de sismologia, espectro de sonar e voz, espectro de vídeo de sistemas de radar e mostrou não ser o processamento digital uma simples extensão do processamento analógico. O advento da integração em larga escala, a redução do custo, tamanho e o aumento da velocidade dos componentes digitais, permitiu a implementação em "hardware" comum de filtros digitais, e também processadores especiais para FFT, e evidenciou uma independência das técnicas digitais em relação às analógicas. Em 1979 aparecem artigos sugerindo a aplicação de microprocessadores [1-6] favorecidos com o emprego de algoritmos matemáticos [8-12], permitindo a construção de filtros digitais para processamento em tempo real.

Atualmente a importância do processamento digital de sinais estende-se pelas áreas da engenharia biomédica, acústica, radar, sonar, sismologia, comunicação de voz, comunicação de dados, ciência nuclear, etc. Também as técnicas de processamento bidimensional de sinais têm seu emprego nos raios-X, fotografias aéreas, fotografias por satélites, etc.

Em nosso país esta área vem despertando a atenção paulatinamente, sendo de grande importância para o seu desenvolvimento o trabalho pedagógico realizado, com a inclusão destes conteúdos em cursos de graduação ou pós-graduação nas escolas.

## 3. QUADRO GERAL DA ÁREA DE PROCESSAMENTO DIGITAL DE SINAIS

O processamento digital, de acordo com Oppenheim-Schafer [11], está relacionado com a representação de seqüências de números ou símbolos e o processamento destas seqüências. Rabiner e Gold [15] assim observam: "sob um ponto de vista, o processamento digital de sinais é uma coleção de algoritmos de computador e, assim podem ser ensinados como simplesmente outro ramo da matemática computacional. Sentimos, entretanto, que o conteúdo da teoria do processamento digital de sinais tem muito em comum com a teoria clássica de filtros e circuitos e a teoria de transformadas da forma em que é presentemente ensinada em currículos padrões de engenharia e que é de grande importância a manutenção desta estrutura formal".

Tendo apresentado breves referenciais para o estudo do processamento digital de sinais, passamos agora a apresentar um quadro geral desta área.(1)

Os fundamentos matemáticos desta área residem,principalmente, na teoria dos sistemas lineares discretos, invariantes no tempo e na teoria da transformada de Fourier discreta. A maior subdivisão da área é a filtragem digital e a análise de espectro. O campo da filtragem digital compreende os filtros de resposta ao impulso infinita (RII) e resposta ao impulso finita (RIF). O campo de análise de espectro é subdividido em cálculo do espectro via transformada discreta de Fourier (TDF) e via técnicas estatísticas, para o caso de sinais aleatórios. Os as-

---

(1) Segundo Rabiner e Gold [15].

pectos restantes são relacionados com a implementação de filtros digitais e analisadores de espectro. Para isso se faz necessário o estudo da teoria da quantização, avaliando os problemas da precisão finita do "hardware" e/ou "software" empregado. Quanto ao problema da implementação também é necessário considerar as limitações e desvantagens do uso de "hardware" e "software".

As aplicações que surgem deste quadro são das mais variadas.(1) Na área específica da telefonia, segundo Oppenheim [11],"as técnicas digitais prometem economia e flexibilidade incrivelmente aumentadas na implementação de comutação e sistemas de transmissão". Assim, as técnicas de processamento digital tem seu emprego em sistemas de comutação digital, transmissão digital, terminais com sistemas de modulação codificada de pulso (PCM) e multiplex por divisão de frequências (FDM), sistemas de supervisão e outros. A reprodutibilidade dos circuitos envolvidos é uma grande vantagem, uma vez que não depende primordialmente da tolerância dos circuitos envolvidos,como é o caso dos circuitos analógicos.

Além da telefonia podemos encontrar aplicações no processamento digital de sinais de audio, voz e imagem, e aplicações que incluem sonar e radar.

---

(1) Ver Oppenheim [10] para maiores detalhes sobre aplicações.

# CAPÍTULO I: PROJETO DE FILTROS DE RESPOSTA AO IMPULSO FINITA E FASE LINEAR

## 1.1. CONSIDERAÇÕES GERAIS

Segundo Rabiner e Gold [15], podemos dividir o processo geral de projeto de filtro digital (para realização tanto em "software" como em "hardware") em quatro passos básicos:

1. Resolver o problema de aproximação para determinar os coeficientes do filtro que satisfaça determinadas especificações;
2. Escolher uma estrutura específica na qual o filtro será realizado e quantizar os seus coeficientes para um comprimento de palavra finito;
3. Quantizar as variáveis do filtro digital, i.e., a entrada e a saída e comprimentos de palavras de variáveis intermediárias;
4. Verificar, por simulação que o projeto resultante vai de encontro às especificações estabelecidas.

Neste capítulo iremos nos dedicar ao primeiro ítem dos quatro passos mencionados, apresentando um resumo das características dos filtros de resposta ao impulso finita (RIF) com fase linear, e das suas principais técnicas de projeto.(1)

---

(1) Maiores detalhes podem ser encontrados em [2-11-15].

## 1.2. RESUMO DAS CARACTERÍSTICAS DOS FILTROS RIF COM FASE LINEAR

### 1.2.1. Condições para Linearidade de Fase.

Se {h(n)} representa uma seqüência causal de duração finita, definida sobre o intervalo $0 \leq n \leq N-1$, a transformada z de {h(n)} será:

$$H(z) = \sum_{n=0}^{N-1} h(n).z^{-n} \qquad (1.1)$$

A transformada de Fourier de {h(n)}, então é

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h(n).e^{-j\omega n} \qquad (1.2)$$

Para o caso em que h(n) é real, limitações adicionais sobre $H(e^{j\omega})$ são obtidas expressando-a em termos de módulo e fase, i.e.,

$$H(e^{j\omega}) = H^*(e^{j\omega}).\ e^{j\theta(\omega)} = \pm\ |H(e^{j\omega})|\ e^{j\theta(\omega)} \qquad (1.3)$$

onde $H^*(e^{j\omega})$ é uma função real, assumindo valores negativos e positivos. Da equação (1.2) e restrição de h(n) real concluímos que a magnitude (módulo) da Transformada de Fourier é uma função simétrica e a fase é uma função anti-simétrica.

O "atraso de fase" e o "atraso de grupo" são definidos respectivamente por:

$$\alpha = -\frac{\theta(\omega)}{\omega} \quad \text{e} \quad \beta = -\frac{d\theta(\omega)}{d\omega} \qquad (1.4)$$

Para a maioria dos filtros RIF práticos deseja-se obter $\alpha$ e $\beta$ constantes. Isto implica que a resposta de fase deve ser linear, i.e.,

$$\theta(\omega) = -\alpha\omega \quad -\pi < \omega < \pi \tag{1.5}$$

Partindo das equações (1.2) e (1.3) e impondo a condição (1.5) pode-se chegar a:

$$\alpha = \frac{N-1}{2} \tag{1.6}$$
$$h(n) = h(N-1-n)$$

As relações em (1.6) indicam a necessidade da resposta ao impulso ser simétrica em torno de um ponto para obter a resposta de fase linear.

Para N par, devemos ter o tipo de resposta indicada na figura 1.1.a. com um centro de simetria entre duas amostras centrais. Para N ímpar, o centro de simetria coincide com a amostra central, conforme a figura 1.1.b.

Se somente o atraso de grupo deve ser constante um segundo tipo de filtros RIF com fase linear resulta; neste caso temos

$$\theta(\omega) = \beta - \alpha\omega \tag{1.7}$$

Impondo estas restrições à resposta ao impulso da mesma forma que na situação anterior, após algum trabalho matemático chega-se às seguintes condições:

$$\alpha = \frac{N-1}{2}$$
$$\beta = \pm \frac{\pi}{2} \tag{1.8}$$
$$h(n) = -h(N-1-n) \qquad 0 \leq n < N-1$$

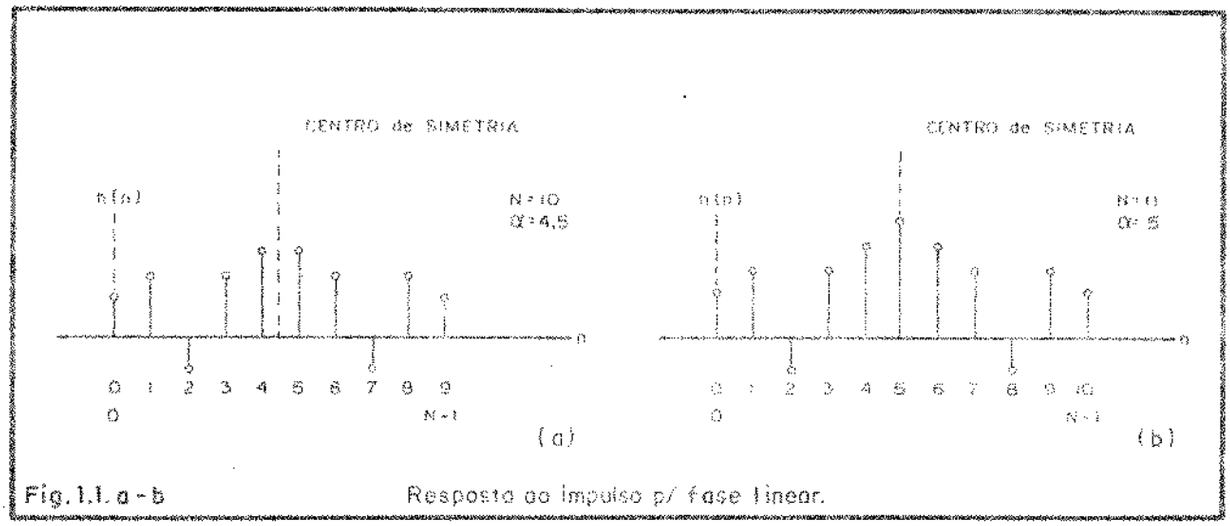



Fig. 1.1. a - b — Resposta ao impulso p/ fase linear.

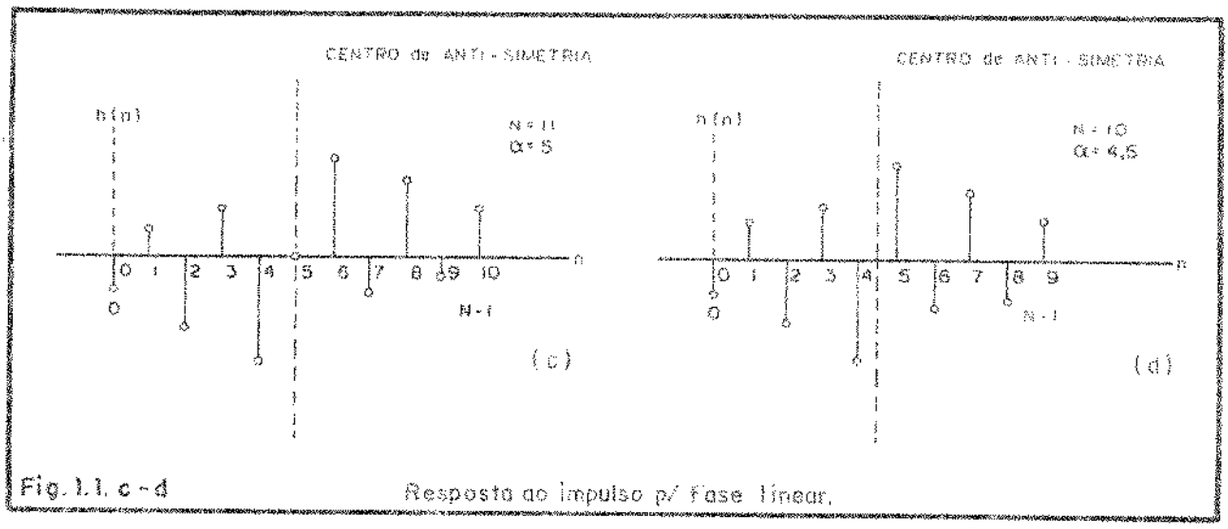



Fig. 1.1. c - d — Resposta ao impulso p/ fase linear.

Portanto, os filtros que satisfazem as condições (1.8.) possuem o mesmo atraso de (N-1)/2 amostras, como os anteriores, entretanto sua resposta ao impulso será anti-simétrica. As figuras 1.1.c. e 1.1.d. ilustram esta situação.

### 1.2.2. Resposta em Freqüência de Filtros RIF com Fase Linear.

Aplicando-se as condições (1.6) e (1.8) na equação (1.3) chega-se, após algum trabalho, a expressões para a resposta em freqüência para cada caso de resposta ao impulso da seção anterior. Para o caso da resposta ao impulso simétrica, N ímpar, temos:

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \left[ \sum_{n=0}^{(N-1)/2} a(n) \cos(\omega n) \right] \qquad (1.9)$$

$$a(n) = 2h\left[\frac{(N-1)}{2} - n\right], \quad n = 1,2,\ldots,(N-1)/2$$

$$a(0) = h[(N-1)/2]$$

Para N par, resposta ao impulso simétrica,

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \left\{ \sum_{n=0}^{N/2-1} b(n)\cos\left[\omega\left(\frac{N}{2} - n - \frac{1}{2}\right)\right] \right\} \qquad (1.10)$$

com

$$b(n) = 2h\left(\frac{N}{2} - n\right), \quad n = 1, 2, \ldots, \frac{N}{2}$$

Estas respostas em freqüência, juntamente com os casos das respostas anti-simétricas são representadas na figura 1.2. Os dois últimos casos cujas equações não foram apresentadas (respostas anti-simétricas) são mais adequados para aproximação de transformadores de Hilbert e diferenciadores.

### 1.2.3. Posição dos Zeros

A posição dos zeros no plano z dos filtros RIF com fase linear é bem definida devido às condições de simetria da resposta ao impulso. Examinando a transformada z de tais filtros, pode-se concluir que:

a) se existir um zero $z_i = r_i e^{j\theta_i}$, com $r_i \neq 1$, $\theta_i \neq 0,\pi$, então haverá outro zero em $z^{-1} = (1/r_i)e^{-j\theta_i}$ (recíproco).

b) para cada zero complexo de H(z) haverá um outro que será o conjugado do anterior.

c) zeros colocados no círculo unitário apresentam seus conjugados, que são também os seus recíprocos.

d) zeros reais fora do círculo unitário ocorrem em pares recíprocos.

e) zeros reais no círculo unitário estão ou em z=+1 ou z=-1. Neste caso ele é seu próprio recíproco e complexo conjugado.

A figura 1.3. ilustra a posição típica dos zeros para filtros RIF com atraso constante.

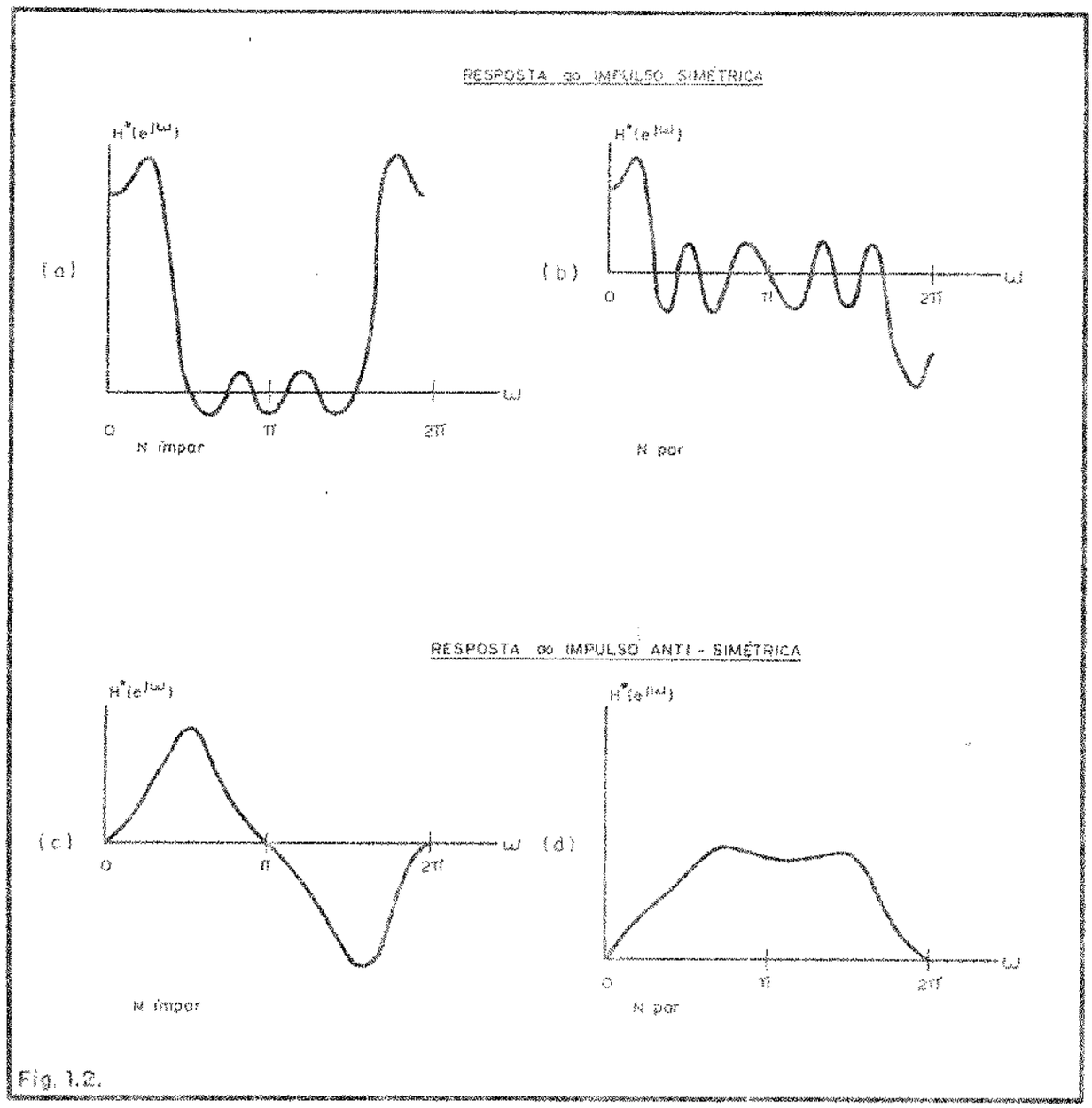

RESPOSTA ao IMPULSO SIMÉTRICA
H*(e^jω)
(a)
ω
0
π
2π
N ímpar
H*(e^jω)
(b)
ω
0
π
2π
N par
RESPOSTA ao IMPULSO ANTI - SIMÉTRICA
H*(e^jω)
(c)
ω
0
π
2π
N ímpar
H*(e^jω)
(d)
ω
0
π
2π
N par

Fig. 1.2.

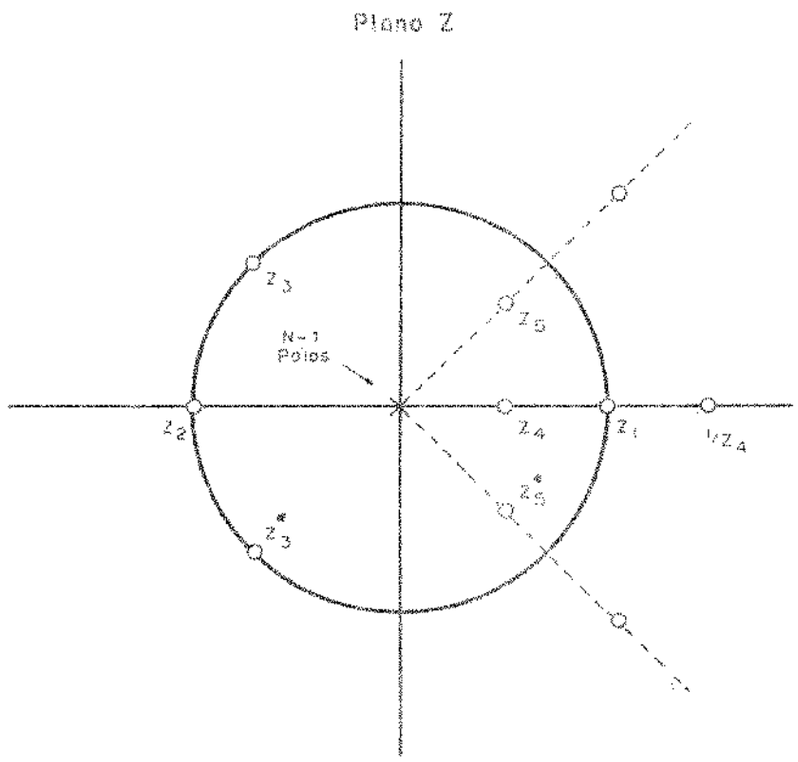



Fig. 1.3. Posição típica dos zeros de filtros de fase linear.

## 1.3. TÉCNICAS DE PROJETO DE FILTROS DIGITAIS

As principais técnicas de projeto de filtros digitais de resposta ao impulso finita (RIF) são: a) técnica das janelas ("windowing"); b) técnica de amostragem em freqüência ; c) técnica de projeto de filtros ótimos. Nesta seção daremos uma idéia resumida de cada uma delas. Maiores detalhes podem ser encontrados em [2-11-15]. Os filtros implementados na parte prática (Capítulo IV) foram calculados usando a técnica das janelas e do projeto ótimo. No caso do projeto ótimo usamos o programa Fortran desenvolvido por McClellan, que faz uso do algoritmo de Remez. Este programa pode ser encontrado em Rabiner e Gold [15].

### 1.3.1. Técnica das Janelas

Consideremos $H(e^{j\omega})$, a resposta em freqüência de um filtro digital qualquer. Uma das características desta resposta é a sua periodicidade. Isto permite expandir $H(e^{j\omega})$ através de uma série de Fourier. Assim, $H(e^{j\omega})$ pode ser escrita como:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)\, e^{-j\omega n} \qquad (1.11.a)$$

$$h(n) = \frac{1}{2\pi} \int_{0}^{2\pi} H(e^{j\omega})\, e^{j\omega n}\, d\omega \qquad (1.11.b)$$

Os coeficientes h(n) da série são identificados como os coeficientes da resposta impulsiva do filtro digital. Temos que a representação através de série de Fourier acima cor

responde a um filtro de resposta ao impulso infinita e, também, não realizável, pois começa em - $\infty$. Pensando,agora, em obter filtros de resposta finita, uma forma imediata seria "truncar" a série de Fourier infinita dentro de um intervalo finito,por exemplo, $n = \pm M$. Este procedimento, entretanto, leva ao conhecido "fenômeno de Gibbs", cujo resultado é a existência de uma quantidade fixa de "overshoot" e "ripple" antes e depois de uma descontinuidade na resposta em freqüência. Este "ripple" não diminui sua amplitude com o aumento da duração da resposta impulsiva. Apenas o "overshoot" é confinado num espaço de freqüência cada vez menor com o aumento de N. Concluímos, daí, não ser um método razoável o truncamento direto da série de Fourier para obter filtros RIF.

Uma forma encontrada para atenuar este efeito é usar uma seqüência finita w(n), chamada janela, para controlar a convergência da série de Fourier, resultando numa resposta mais suave. A técnica das janelas é ilustrada na figura 1.4. Na parte superior da figura é mostrada a resposta em freqüência periódica desejada $H(e^{j\omega})$ e os coeficientes da série de Fourier $\{h(n)\}$ . A seguir, aparece a seqüência de duração finita w(n) com sua transformada de Fourier $W(e^{j\omega})$. $W(e^{j\omega})$ deve possuir uma região central que contenha a maior parte da energia das janelas e lobos laterais com decaimento rápido. Para produzir filtros RIF a seqüência $\tilde{h}(n) = w(n).h(n)$ é formada. A terceira linha mostra $\tilde{h}(n)$ e sua transformada $\tilde{H}(e^{j\omega})$, a qual representa a convolução circular de $H(e^{j\omega})$ e $W(e^{j\omega})$. Na última linha, finalmente, aparece a seqüência realizável g(n), que é a versão deslocada de $\tilde{h}(n)$. A escolha da função w(n) envolve consideração dos seguintes fatores:

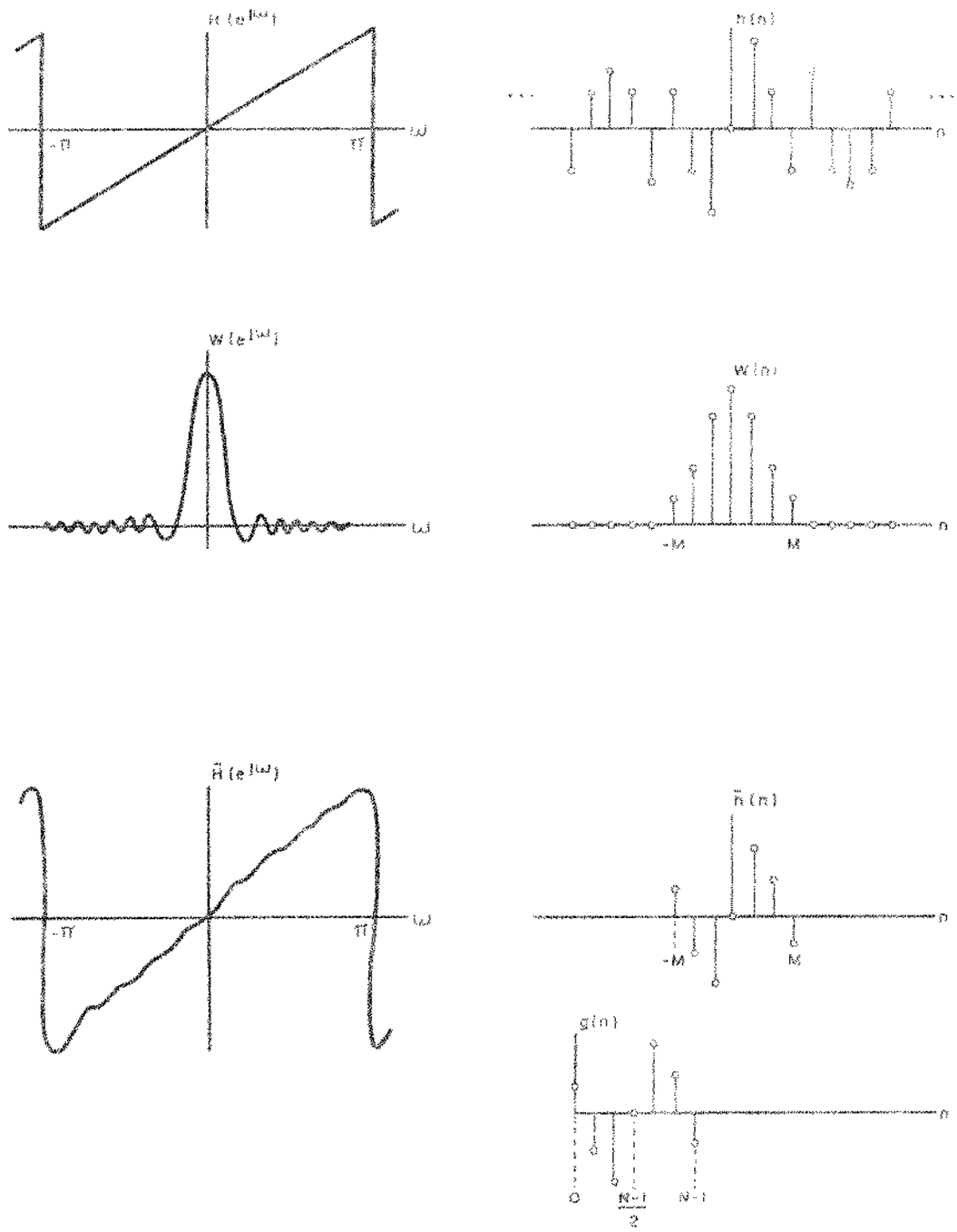



Fig. 1.4. Técnica das janelas.

1) a largura da região central da janela, que deve ser a menor possível;
2) as oscilações nas regiões laterais da janela, que devem decair o mais rapidamente possível.

Estes fatores implicam que os filtros apresentarão faixas de transição em cada lado das descontinuidades e oscilações ("ripple") na resposta em freqüência.

Várias propostas de janelas existem. Vamos apresentar as mais freqüentes na literatura.

Janela Retangular:

$$w_R(n) = \begin{cases} 1 & |n| \leq \frac{N-1}{2} \\ 0 & \text{qualquer outro } n \end{cases} \qquad (1.12)$$

Comentário: corresponde a "truncar" diretamente a série de Fourier, e seu efeito em $H(e^{j\omega})$ já foi comentado anteriormente. A relação entre a máxima amplitude dos lobos laterais e a amplitude do lobo central é 22,3% para N = 11 e decresce lentamente quando N é aumentado. A largura do lobo central é $2.\omega_s/N$, sendo $\omega_s$ a freqüência de amostragem em radianos por segundo. Seu espectro aparece na figura 1.5.

Janela Generalizada de Hamming:

$$w_H(n) = \begin{cases} \alpha + (1-\alpha)\cos\left(\frac{2\pi n}{N-1}\right) & |n| \leq \frac{N-1}{2} \\ 0 & \text{outros valores de } n \end{cases} \qquad (1.13)$$

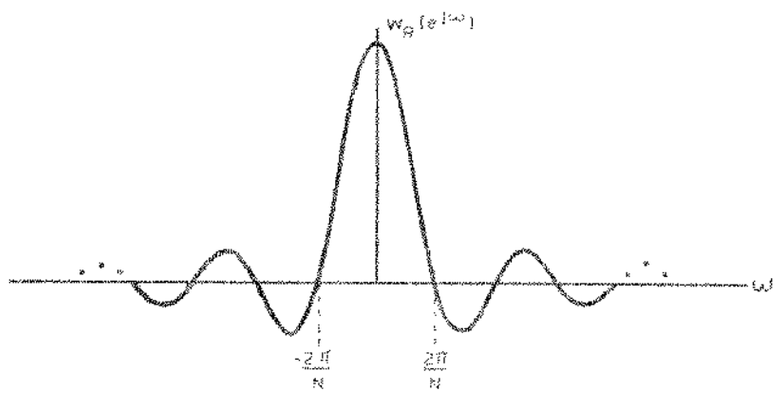



Fig. 1.5. Espectro da janela retangular.

sendo $0 \leq \alpha \leq 1$ . Se $\alpha = 0,54$ temos a chamada janela de "Hamming"; se $\alpha = 0,5$, temos uma janela "hanning".

Comentário: a largura do lobo principal é aproximadamente $4\omega_s/N$.

Janela de Blackman:

$$w_B(n) = \begin{cases} 0,42 + 0,5 \cos\left(\frac{2\pi n}{N-1}\right) + 0,08 \cos\left(\frac{4\pi n}{N-1}\right) & |n| \leq \frac{N-1}{2} \\ 0 & \text{outros valores de } n \end{cases} \qquad (1.14)$$

Comentário: apresenta uma redução das oscilações devido ao fenômeno de Gibbs em troca de uma maior largura da região central , que é aumentada para aproximadamente 6 $\omega_s/N$.

Janela de Kaiser:

$$w_K(n) = \begin{cases} \frac{Io(\beta)}{Io(\alpha)} & |n| \leq \frac{N-1}{2} \\ 0 & \text{outros valores de } n \end{cases} \qquad (1.15)$$

onde $\alpha$ é um parâmetro independente que indica o compromisso existente entre a amplitude dos lobos laterais e largura do lobo principal e

$$\beta = \alpha\sqrt{1-\left(\frac{2n}{N-1}\right)^2} \qquad (1.16)$$

sendo $I_0(x)$ a função de Bessel de ordem zero de primeira espécie (ou modificada).

Esta função pode ser avaliada com qualquer grau de precisão usando a série

$$I_0(x) = 1 + \sum_{k=1}^{\infty}\left[\frac{1}{k!}\left(\frac{x}{2}\right)^k\right]^2 \qquad (1.17)$$

Comentário: a vantagem desta janela consiste em que podemos variar o "ripple" continuamente, desde valores da janela de Blackman, até o alto valor da janela retangular através da variação de $\alpha$. A largura da região central, como nas outras janelas, pode ser ajustada pela variação de N.

A figura 1.6. mostra uma comparação destas janelas.

## 1.3.2. Técnica de Amostragem em Freqüência

Uma seqüência de duração finita pode ser representada por sua transformada de Fourier discreta. Então, um filtro RIF tem sua representação em termos de "amostras em freqüência" como

$$H(k) = \sum_{n=0}^{N-1} h(n)\, e^{-j(2\pi/N)nk} \qquad (1.18)$$

$$= H(z)\big|_{z = e^{j(2\pi/N)k}} \quad , k = 0,1,\ldots,N-1$$

Pode-se calcular a transformada z de h(n), H(z), e, portanto, $H(e^{j\omega})$ através das amostras em freqüência de acordo com

$$H(e^{j\omega}) = \frac{e^{-j\omega(N-1)/2}}{N} \sum_{k=0}^{N-1} \frac{H(k)\, e^{-j(\pi k/N)}\ \sin(\omega N/2)}{\sin(\omega/2 - \pi k/N)} \qquad (1.19)$$

Esta expressão mostra que a resposta em freqüência do filtro é uma combinação linear das amostras H(k) com fun

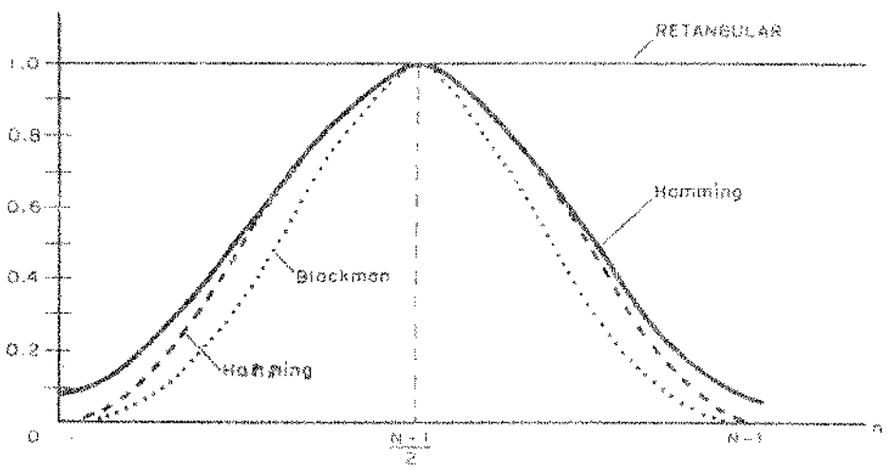



Fig. 1.6. Algumas janelas típicas p/ projeto de filtros RIF (1).

ções de interpolação em freqüência.Cada amostra em freqüência produz uma resposta em freqüência igual a uma constante multiplicada por funções de interpolação, funções estas que devem produzir um bom cancelamento do "ripple" nas faixas de frequências adjacentes. Concluímos que, para aproximarmos qualquer resposta em freqüência contínua, devemos obter "amostras em freqüência" de N pontos igualmente espaçados e avaliar a resposta freqüência contínua como uma interpolação da resposta em freqüência amostrada.

A figura 1.7 dá uma idéia do método. A curva sólida representa a resposta em freqüência desejada. Os pontos representam o conjunto de amostras em freqüência escolhido. A curva (b) mostra uma curva contínua, resultante da interpolação das amostras em freqüência.

Se deixarmos as amostras em freqüência que estão na faixa de transição sem especificação, isto é, como uma variável de projeto, podemos usar este recurso para conseguir filtros muito bons, pela otimização destas amostras em freqüência. Esta idéia está representada na figura 1.8. O processo de cálculo consiste em escrever um conjunto de inequações lineares para as amostras fixas. Este conjunto de inequações lineares é resolvido usando técnicas de programação linear, fornecendo valores para as amostras em freqüência não sujeitas a restrições.

### 1.3.3. Técnica de Projeto de Filtros Ótimos

A técnica de amostragem em freqüência resumida na seção anterior faz uso de um procedimento iterativo para chegar a um filtro RIF que possua a maior atenuação mínima para um dado N. No caso de filtros projetados por esta técnica existe uma

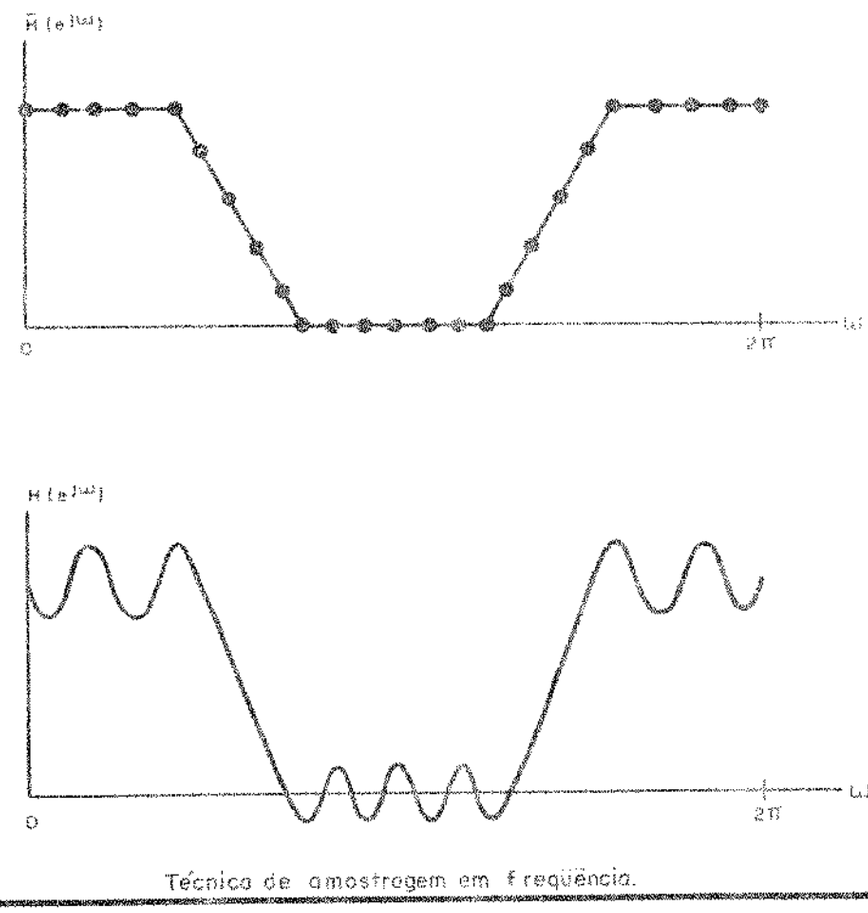



Fig. 1.7. Técnica de amostragem em frequência.

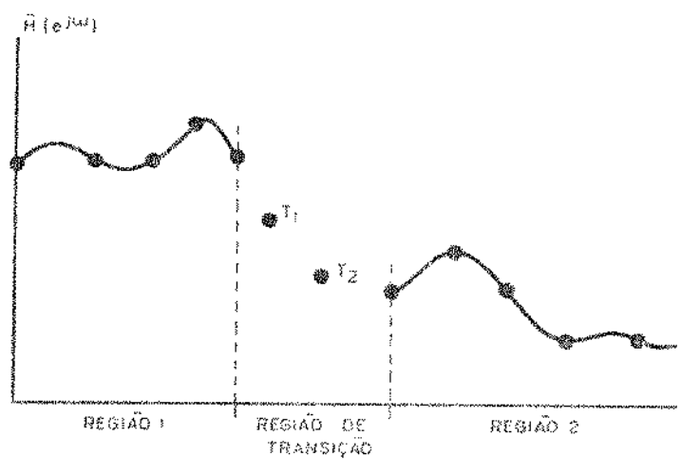



Fig. I.8. Especificação p/ um filtro com amostragem em freqüência.

limitação indesejável na escolha das freqüências da corte (não é possível prever seu valor). Além disso, o erro de aproximação (diferença entre curva desejada e obtida), tende a ser maior em torno da região de transição e menor nas regiões mais distantes dela. Pode-se pensar que se o erro de aproximação for distribuído uniformemente em freqüência, uma dada especificação deve ser alcançada com um filtro de menor ordem. Esta forma de resolver o problema conduz à chamada aproximação "equiripple", representada na fig. 1.9.

Suponha que desejamos projetar um filtro passa baixa levando em conta os parâmetros indicados na figura 1.9, i.e., M, $\delta_1$, $\delta_2$, $\omega_p$ e $\omega_s$. Não é possível especificar independentemente todos estes parâmetros. Entretanto existem algoritmos de projeto nos quais alguns parâmetros são fixados e um procedimento iterativo é usado para obter ótimos ajustes dos parâmetros restantes. Dois trabalhos foram desenvolvidos para resolver este problema. Herrmann e Schuessler, e, mais tarde Hofstetter desenvolveram estudos em que se fixavam M, $\delta_1$ e $\delta_2$, sendo $\omega_p$ e $\omega_s$ variáveis. Parks e McClellan, e Rabiner, desenvolveram processos em que M, $\omega_p$ e $\omega_s$ são fixados e $\delta_1$ e $\delta_2$ são variáveis. O procedimento utilizado por Herrmann e Schuessler consiste em determinar o número de extremos (máximos e mínimos) que teria $H(e^{j\omega})$, escrito como um polinômio trigonométrico, para em seguida escrever um conjunto de equações lineares, cuja solução irá fornecer um conjunto de coeficientes e as freqüências onde os extremos ocorrem. Este método envolve a solução de equações não lineares e deve ser resolvido por procedimento iterativo. Hoffstetter, Oppenheim e Siegel, procurando resolver estas limitações por métodos computacionais, ao invés de escreverem

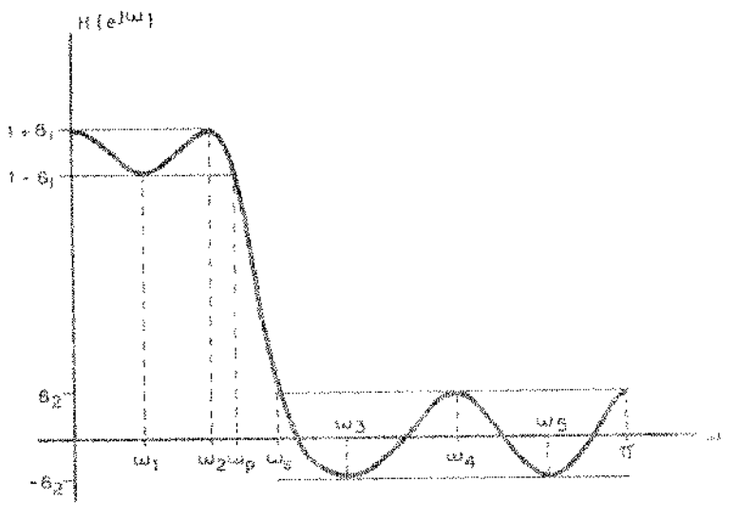



Fig. 1.9. Aproximação "equiripple" para um filtro passa baixa.

equações não lineares, usaram técnicas iterativas para produzir um polinômio trigonométrico que tivesse os extremos desejados. Fica, ainda, o problema do controle sobre $\omega_p$ e $\omega_s$. Parks e McClellan mostraram que com M, $\omega_p$ e $\omega_s$ fixos, o problema do projeto de filtros torna-se um problema de aproximação de Chebychev. Para dar uma idéia da formulação deste problema, vamos definir uma função de erro de aproximação, $E(\omega)$, tal que,

(1.20)

onde $W(\omega)$ é uma função de ponderação. O problema do projeto consiste em obter um algoritmo que minimize o

$$\text{máximo } |E(\omega)|, \tag{1.21}$$

Parks e McClellan reformulam a idéia do número de extremos, através do chamado "teorema das alternâncias". Este teorema dá como condição necessária e suficiente para a obtenção de filtros ótimos a existência de M+2 extremos na função $E(\omega)$. Daí, apresentam um procedimento iterativo para obtenção de filtros ótimos. Neste procedimento utilizam um algoritmo de cálculo conhecido como "algoritmo de Remez". Este procedimento, para cálculo de filtros ótimos, condensado num programa em linguagem Fortran foi por nós utilizada no capítulo III.

## 1.4. FILTROS RIF E RII

Reservamos esta seção para estabelecer alguns aspectos de comparação entre filtros RIF e RII de certa forma procurando explicar a nossa preferência neste trabalho pelos fil-

tros RIF. Esta comparação não é fácil, uma vez que existem diferentes técnicas de projeto para ambos. Um critério de comparação mais objetivo compara os filtros passa baixa RIF ótimos com os filtros RII elípticos que possuem especificações em freqüência equivalentes. A principal base de comparação é o número de multiplicações por amostra de entrada exigida na maioria das realizações padrão de cada filtro. Este estudo leva à conclusão de que os filtros elípticos são em geral mais eficientes em alcançar uma determinada especificação do módulo da resposta em freqüência do que os filtros ótimos. Entretanto, os filtros RIF possuem a útil propriedade de apresentarem fase linear, i.e.,não há distorção por atraso de grupo. Para os filtros elípticos geralmente há uma grande quantidade de distorção por atraso de grupo.

Um outro tipo de argumento consiste em comparar os filtros quanto às facilidades de cálculo de cada tipo. Assim, os filtros RII, por exemplo, podem ser dimensionados usando calculadoras simples, pois empregam fórmulas relativamente compactas. Já os filtros RIF, mesmo aqueles calculados pelo método das janelas, exigem um procedimento iterativo, mais adequado para o emprego das facilidades computacionais. Os filtros RII, entretanto, estão limitados em termos de flexibilidade, a tipos passa-baixa, passa alta, passa faixa e rejeita faixa; enquanto isso, os filtros RIF possuem maior facilidade para alcançar várias especificações arbitrárias da resposta em freqüência desejada (filtros de múltiplas bandas p.ex.).

# CAPÍTULO II: FILTRAGEM DIGITAL COM USO DE ARITMÉTICA DISTRIBUÍDA E MICROPROCESSADOR

## 2.1. INTRODUÇÃO

Várias formas de implementação de filtros digitais foram sugeridas. Jackson, Kaiser e McDonald[7] realizam uma implementação de filtros digitais através de conexão paralela ou em cascata de seções de segunda ordem. Para implementar cada seção faz-se uso de somadores, multiplicadores e registradores de deslocamento. Entretanto, devido ao elevado tempo para calcular uma amostra de saída apresentam severas restrições quanto à sua largura de faixa. Peled e Liu [12] apresentam uma técnica que dispensa o uso de multiplicadores, reduzindo o tempo de processamento. Isto é conseguido usando uma tabela com constantes pré-calculadas e armazenadas numa RAM. Estas constantes são endereçadas através de um "vetor" que é função dos dados das entradas e saídas. Allen e Holt [1], aproveitando a idéia do vetor multiplicador de Peled e Liu sugerem o emprego de um microprocessador, aumentando a flexibilidade do sistema e a velocidade das operações. Um método semelhante é desenvolvido por Farhang-Boroujeny e Hawkins [6], baseado no algoritmo de Little [8], também fazendo uso de microprocessador associado a algum "hardware". Assim, a associação de algoritmos matemáticos com microprocessadores constitui-se num método eficiente para resolver o problema da implementação de filtros digitais para processamento do sinal em tempo real. Passaremos a descrever este método nas seções seguintes.

## 2.2. EQUAÇÃO DE DIFERENÇAS FINITAS E COEFICIENTES DO FILTRO

Em muitas aplicações uma classe de sistema lineares invariantes no tempo tem um papel importante. Esta classe constitui-se daqueles sistemas em que, a entrada x(n) e a saída y(n) satisfazem a equação de diferenças finitas, linear, com coeficientes constantes, de ordem N, na forma

$$y(n) = \sum_{k=0}^{M} a_k . x(n-k) - \sum_{k=1}^{N} b_k . y(n-k) \qquad (2.1)$$

Se a resposta ao impulso unitário do sistema é de duração finita, então iremos chamá-lo de "sistema de resposta ao impulso finita" (RIF). Se N = 0 na equação (2.1), de forma que

$$y(n) = \sum_{k=0}^{M} a_k . x(n-k) \qquad (2.2)$$

então a equação corresponde a um sistema RIF. Sabemos que um sistema linear invariante no tempo é completamente caracterizado pela sua resposta ao impulso h(n), na forma

$$y(n) = \sum_{k=-\infty}^{\infty} h(k) . x(n-k) = h(n) * x(n) \qquad (2.3)$$

Comparando as equações (2.2) e (2.3) concluímos imediatamente que

$$h(n) = \begin{cases} a(n), & n = 0,1,\ldots,M \\ 0 & n<0 \end{cases} \qquad (2.4)$$

Isto é, os coeficientes da equação de diferenças finitas são iguais aos coeficientes da resposta ao impulso do filtro. Assim, um sistema RIF pode sempre ser caracterizado por uma equação de diferenças na forma da equação (2.2) com N = 0. Em contraste, para um sistema de resposta ao impulso infinita(RII), N deve ser maior do que zero. Neste trabalho trataremos exclusivamente de sistemas do tipo RIF.

## 2.3. ALGORITMO DE CÁLCULO

Passaremos a descrever o algoritmo de cálculo que servirá de base para este trabalho. Consideremos a equação (2.2) para um sistema RIF, repetida a seguir:

$$y(n) = \sum_{k=0}^{M} a_k . x(n-k) \qquad (2.5)$$

Nesta equação os x(n-k), representam amostras de entrada, y(n), amostra de saída, e os $a_k$, são coeficientes. Se a representação aritmética em complemento de dois é usada para representar os valores envolvidos, com a quantização do sinal em B bits , então, podemos escrever para x(n)

$$x(r) = -x^0(r) + \sum_{i=1}^{B-1} x^i(r).2^{-i} \qquad (2.6)$$

Substituindo (2.6) em (2.5) temos que

$$y(n) = \sum_{k=0}^{M} a_k \left[ -x^0(n-k) + \sum_{i=1}^{B-1} x^i(n-k).2^{-i} \right] \qquad (2.7)$$

$$y(n) = \sum_{k=0}^{M} \sum_{i=1}^{B-1} a_k . x^i(n-k) . 2^{-i} - \sum_{k=0}^{M} a_k . x^0(n-k) \qquad (2.8)$$

Invertendo-se a ordem das somatórias obtemos a equação básica do algoritmo de Little,

$$y(n) = \sum_{i=1}^{B-1} \sum_{k=0}^{M} a_k . x^i(n-k) . 2^{-i} - \sum_{k=0}^{M} a_k . x^0(n-k) \qquad (2.9)$$

Definindo

$$C_n^i = \sum_{k=0}^{M} a_k . x^i/n-k) \qquad (2.10)$$

então a equação 2.9 pode ser reescrita como

$$y(n) = \sum_{i=1}^{B-1} C_n^i . 2^{-i} - C_n^0 \qquad (2.11)$$

Como $x^i(n-k) = 0$ ou 1, então um vetor $v [ x^i(n), x^i(n-1), \ldots, x^i(n-M) ]$ pode ser usado para calcular cada parcela $C_n^i$ da soma indicada pela equação (2.11). Desta forma os valores de $C_n^i$ necessários para o cálculo de $y(n)$ são selecionados ao invés de calculados, de acordo com

$$C_n^i = v [ x_n^i , x_{n-1}^i , \ldots, x_{n-M}^i ] \qquad (2.12)$$

A função "v" tem M+1 argumentos binários, resultando em $2^{M+1}$ combinações possíveis entre os coeficientes. Estas combinações deverão estar prê-calculadas e armazenadas numa tabela. Se, por exemplo, M=4, então teremos $2^5 = 32$ combinações distintas. Assim,por exemplo $v(9) = v(1001) = a_0 + a_3$.

Tendo em vista a implementação do sistema a equação (2.11) pode ser reescrita como

$$y(n) = \{[C_n^{B-1} \cdot 2^{-1} + C_n^{B-2}].2^{-1} + \ldots + C_n^1\}.2^{-1} - C_n^0 \qquad (2.13)$$

## 2.4. ALTERAÇÃO NO ALGORITMO PARA PERMITIR A DIVISÃO DOS "BUFFERS"

Uma forma de considerar o problema da filtragem digital também usando a aritmética distribuída foi apresentada por Farhang-Boroujeny e Hawkins [6]. Este algoritmo parte da equação de diferenças finitas, válida para uma estrutura recursiva, de acordo com a equação 2.1 e, no final,chega a uma equação básica que permite a divisão dos vetores de endereços em sub- conjuntos.

As vantagens desta divisão serão comentadas na parte de implementação, na seção seguinte.Para descrever o algoritmo, façamos $\{A_n\}$ ser definido como o conjunto dos elementos $\{a_k.x(n-k)\}_{k=0}^{M}$ e $\{b_k.y(n-k)\}_{K=1}^{N}$ e $\{A_{n,r}\}_{r=1}^{p}$ uma série de sub-conjuntos de $A_n$, sendo p o número de sub-conjuntos ou divisões. Definindo os elementos de $A_{n,r}$ como $\{d_{r,j}\ w_{n,r,j}\}_{j=1}^{m_i}$, sendo $d_{r,j}$ um dos coeficientes do filtro,$a_k$ ou $b_k$, e $w_{n,r,j}$ a correspondente variável x(n-k) ou y(n-k), e $m_i$ o número de elementos do sub-conjunto $A_{n,r}$. Usando as definições acima, pode-

mos reescrever a equação 2.1 como

$$y(n) = \sum_{r=1}^{p} \sum_{j=1}^{m_i} d_{r,j} \; w_{n,r,j} \qquad (2.14)$$

Utilizando a representação de complemento de dois para $w_{n,r,j}$, sendo B o número de dígitos binários para caracterização de $w_{n,r,j}$, temos:

$$w_{n,r,j} = -w^{0}_{n,r,j} + \sum_{i=1}^{B-1} w^{i}_{n,r,j} \; 2^{-i} \qquad (2.15)$$

sendo $w^{i}_{n,r,j} = 0$ ou 1.

Substituindo a equação (2.15) em (2.14) obtemos:

$$y(n) = \sum_{r=1}^{p} \sum_{j=1}^{m_i} d_{r,j} \left[ -w^{0}_{n,r,j} + \sum_{i=1}^{B-1} w^{i}_{n,r,j} \; 2^{-i} \right] \qquad (2.16)$$

$$= \sum_{r=1}^{p} \left[ \sum_{j=1}^{m_i} d_{r,j} \; (-w^{0}_{n,r,j}) + \sum_{j=1}^{m_i} d_{r,j} \; w^{i}_{n,r,j} \; .2^{-i} \right]$$

Definindo

$$C^{i}_{n,r} = \sum_{j=1}^{m_i} d_{r,j} \; w^{i}_{n,r,j} \quad ; \quad 0<i \leq B-1 \qquad (2.17)$$

temos:

$$y(n) = \sum_{r=1}^{p} \sum_{i=1}^{B-1} C^{i}_{n,r} \; . \; 2^{-i} - \sum_{r=1}^{p} C^{0}_{n,r} \qquad (2.18)$$

Invertendo a ordem das somatórias de acordo com a idéia básica do algoritmo de Little, temos:

$$y(n) = \sum_{i=1}^{B-1} \left( \sum_{r=1}^{p} C^{i}_{n,r} \right) . \; 2^{-i} - \sum_{r=1}^{p} C^{0}_{n,r} \qquad (2.19)$$

Pela definição de $C^{i}_{n,r}$, vemos que ele pode assu-

mir $2^{m_i}$ valores distintos para um conjunto $\{d_{r,j}\}_{j=1}^{m_i}$ dado. Se estes $2^{m_i}$ valores são calculados e armazenados em posições de memória adequadas, então um endereço na forma $(w^i_{n,r,1}, w^i_{n,r,2}, \ldots, w^i_{n,r,m_i})$ pode ser usado para chamar $C^i_{n,r}$ quando for necessário. A equação (2.19) pode ser reescrita na forma:

$$y(n) = \left\{ \ldots \left[ \left( \sum_{r=1}^{p} C^{B-1}_{n,r} \right) \cdot 2^{-1} + \sum_{r=1}^{p} C^{B-2}_{n,r} \right] \cdot 2^{-1} + \ldots + \sum_{r=1}^{p} C^{1}_{n,r} \right\} 2^{-1} - \sum_{r=1}^{p} C^{0}_{n,r} \qquad (2.20)$$

## 2.5. ESTRUTURA DO SISTEMA

Uma estrutura de implementação possível para um filtro digital baseada no algoritmo de cálculo descrito anteriormente é mostrada na figura 2.1.

Neste caso escolhemos um filtro não recursivo, com 6 coeficientes, com quantização do sinal de entrada de 4 bits. O cálculo de y(n) segue a idéia apresentada pela equação (2.13). Os números no registro X indicam o peso de cada bit em termos de notação posicional. A letra "S" indica o bit de sinal. Assim, -3 corresponde ao bit menos significativo, etc. No início do processo todos os bits menos significativos estão nas linhas de entrada do "buffer" e o microprocessador irá lê-los e usá-los como endereço para ter acesso a $C^3_n$. O resultado é dividido por dois por meio de uma instrução de deslocamento para a direita, de acordo com a equação (2.13). Quando o conteúdo do "buffer" foi lido, o conteúdo do registro X tem que ser deslocado de 1 bit, tornando os próximos bits menos significativos disponíveis para o microprocessador. O microprocessador novamente emprega estes

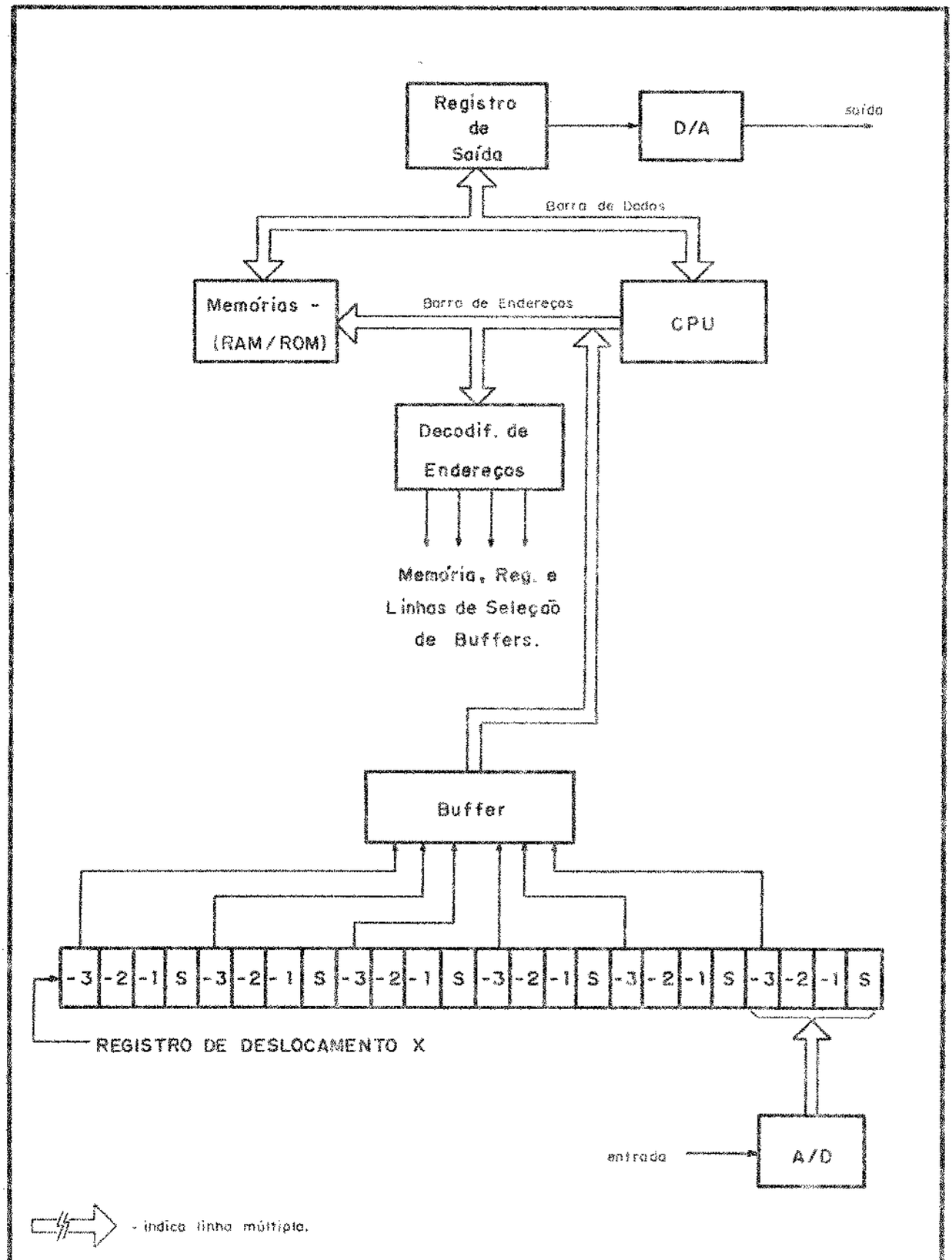



Fig. 2.1. Estrutura do sistema para implementar aritmética distribuída com microprocessador.

bits para ter acesso a $C_n^2$. Este valor é somado com o conteúdo do acumulador, e o novo conteúdo é dividido por dois. O processo continua até o momento em que os bits de sinal aparecem nas linhas de entrada do "buffer". Neste instante $C_n^0$ é obtido à partir da tabela e subtraído do conteúdo atual do acumulador. O último valor computado no acumulador é y(n), que é, em seguida, transferido para o registro de saída.

Supondo que a equação (2.20) seja usada, então, a estrutura da figura 2.1 se modificaria para incluir mais de um "buffer", ou p "buffers". Se p=2, com quantização de B=4 bits, teremos dois"buffers" a serem selecionados. Assim, para cálculo de y(n), seleciona-se o primeiro "buffer" para obter $C_{n,1}^3$, e na seqüência, o segundo "buffer" para $C_{n,2}^3$. Em seguida estes valores são somados, multiplicados por $2^{-1}$ e assim por diante, de acordo com a equação (2.20).

Uma estrutura recursiva é indicada no diagrama de blocos da figura 2.2.

Supondo que esta estrutura corresponda a um filtro recursivo de segunda ordem, então a funcão "v" para este circuito terá cinco argumentos binários, resultando em $2^5 = 32$ valores distintos. Ela pode ser realizada armazenando estes 32 valores num dispositivo de memória e usando os argumentos binários para endereçá-los. Um método conveniente para obter estes argumentos usando cinco registros de deslocamento é mostrado na figura 2.3. Os dois registros externos são carregados com x(n) e y(n-1) com um sinal de controle (c). Após cada pulso de relógio um novo vetor $v(x_n^j, x_{n-1}^j, x_{n-2}^j, y_{n-1}^j, y_{n-2}^j)$ está presente nas linhas de endereço $(a_0-a_4)$, as quais são usadas para endereçar o dispositivo de memória. A disposição dos registros de

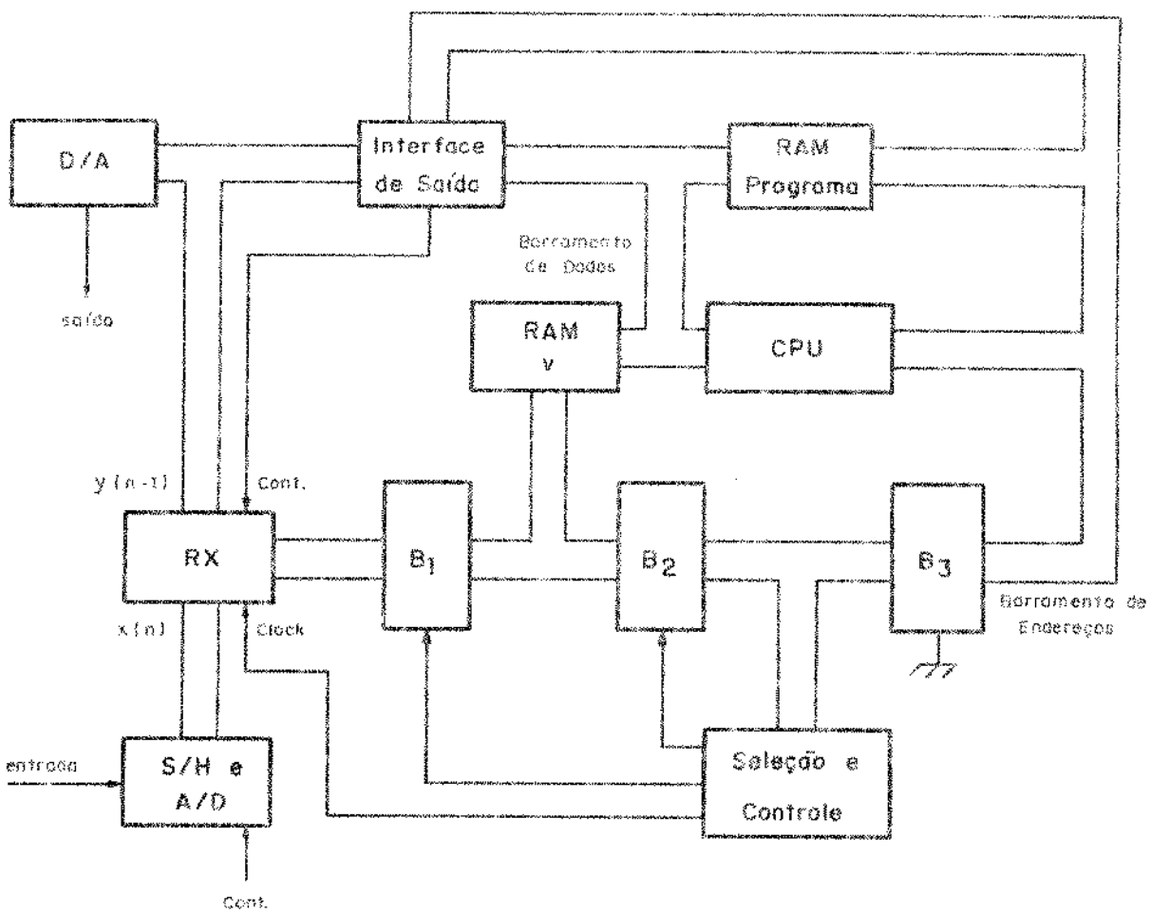



Fig. 2.2. Estrutura recursiva para implementar aritmética distribuída com microprocessador.

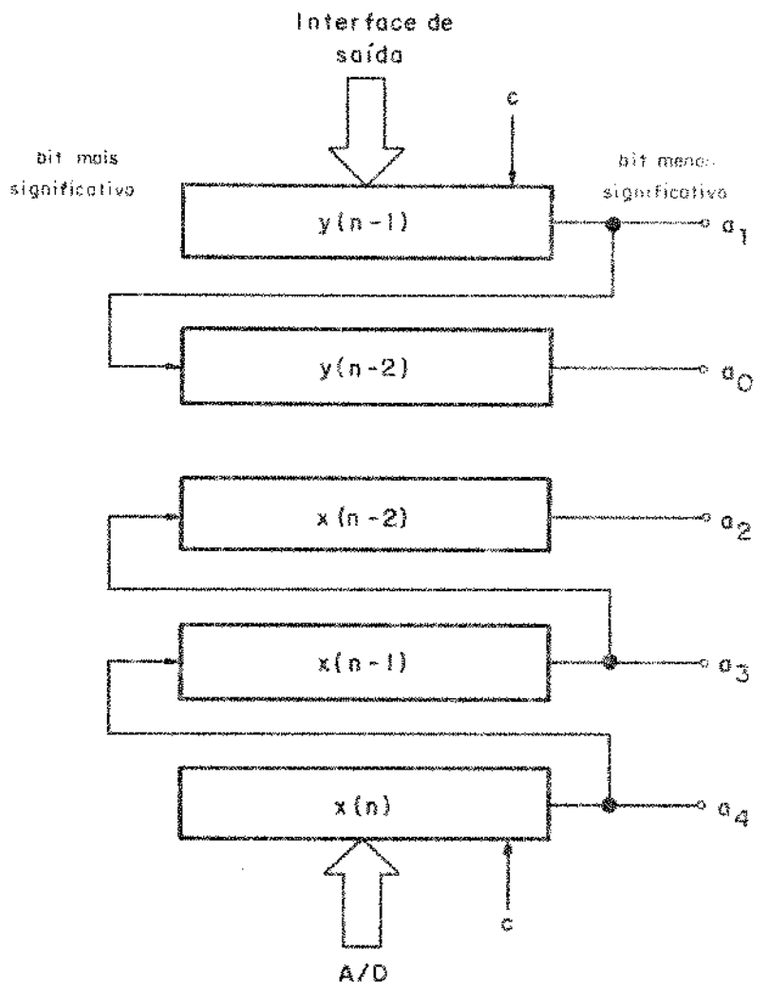



Fig. 2.3. Estrutura para o registro X.

deslocamento também faz com que x(n) seja deslocado para o registro x(n-1), realizando o atraso unitário.

Neste tipo de sistema o microprocessador pode ser usado para: a) calcular os valores de "v"; b) habilitar o acesso dos registros que geram o argumento de "v" até a RAM; c) calcular os valores de y(n)a partir dos valores de "v" selecionados na RAM; d) processar a saída do valor y(n).

Devemos notar que tanto o microprocessador quanto o circuito adicional, indicado como RX na fig. 2.2, devem endereçar a RAM, o primeiro para armazenar os valores de "v" e o segundo para produzir uma saída de um valor específico de v. Daí a necessidade de que o barramento de endereços do microprocessador e do circuito adicional tenham possibilidade de entrar no 3º estado (alta impedância ) para evitar problemas de operação. Além disso o endereçamento do circuito adicional deve estar sincronizado com o endereçamento do microprocessador, de modo que o dado fique válido quando o mesmo aparece no barramento de dados da RAM. Na figura 2.2., dá-se uma idéia de como isto pode ser conseguido. Os "buffers" B1 e B2, unidirecionais e com "terceiro estado" são selecionados pelo endereço EEH e seu complemento, respectivamente. Se o endereço EEH aparece no barramento de endereços do microprocessador o "buffer" B1 é habilitado ( saída igual à entrada) durante o tempo em que EEH permanece no barramento. Ao mesmo tempo B2 é inibido (alta impedância).Isto é conseguido no bloco de seleção e controle através de um circuito combinacional. Para qualquer endereço diferente de EEH, B2 é habilitado, permitindo o acesso do microprocessador à RAM. O mesmo sinal que habilita B1 é invertido, atrasado e usado para produzir um deslocamento unitário no registro RX.

Para as estruturas de implementação sugeridas nas figuras 2.1 e 2.2, antes que a filtragem tenha início, os valores de "v" devem estar calculados e armazenados em uma tabela na memória. O fluxograma da figura 2.4 indica como os valores de "v" podem ser calculados.

## 2.6. ESTRUTURA DA PROGRAMAÇÃO DO SISTEMA

Existem várias maneiras de estruturar a programação do sistema para a implementação do algoritmo de cálculo com aritmética distribuida. No caso de usarmos um microprocessador de 8 bits,como o Z-80 da Zilog, ou 8085 da Intel, estas formas irão depender do modo de endereçamento utilizado. Um dos modos de endereçamento possível é o modo de endereçamento indireto por registro, cujo formato está indicado na figura 2.5.a. Nesta forma de endereçamento,um dos pares de registros do microprocessador (HL, BC ou DE) é carregado com o endereço da memória que contém o dado (em nosso caso, o endereço de um "buffer"). Um exemplo de utilização é mostrado na figura 2.5.b para a instrução LD A, (HL).

Sempre que o endereço do par HL aparecer no barramento de endereços, podemos usá-lo para selecionar o "buffer" relacionado com a tabela. Este modo de endereçamento tem a desvantagem de permitir a seleção de apenas um "buffer". Caso tivéssemos vários "buffers" a serem selecionados em seqüência por endereços distintos, então,podemos optar por uma facilidade conheci-

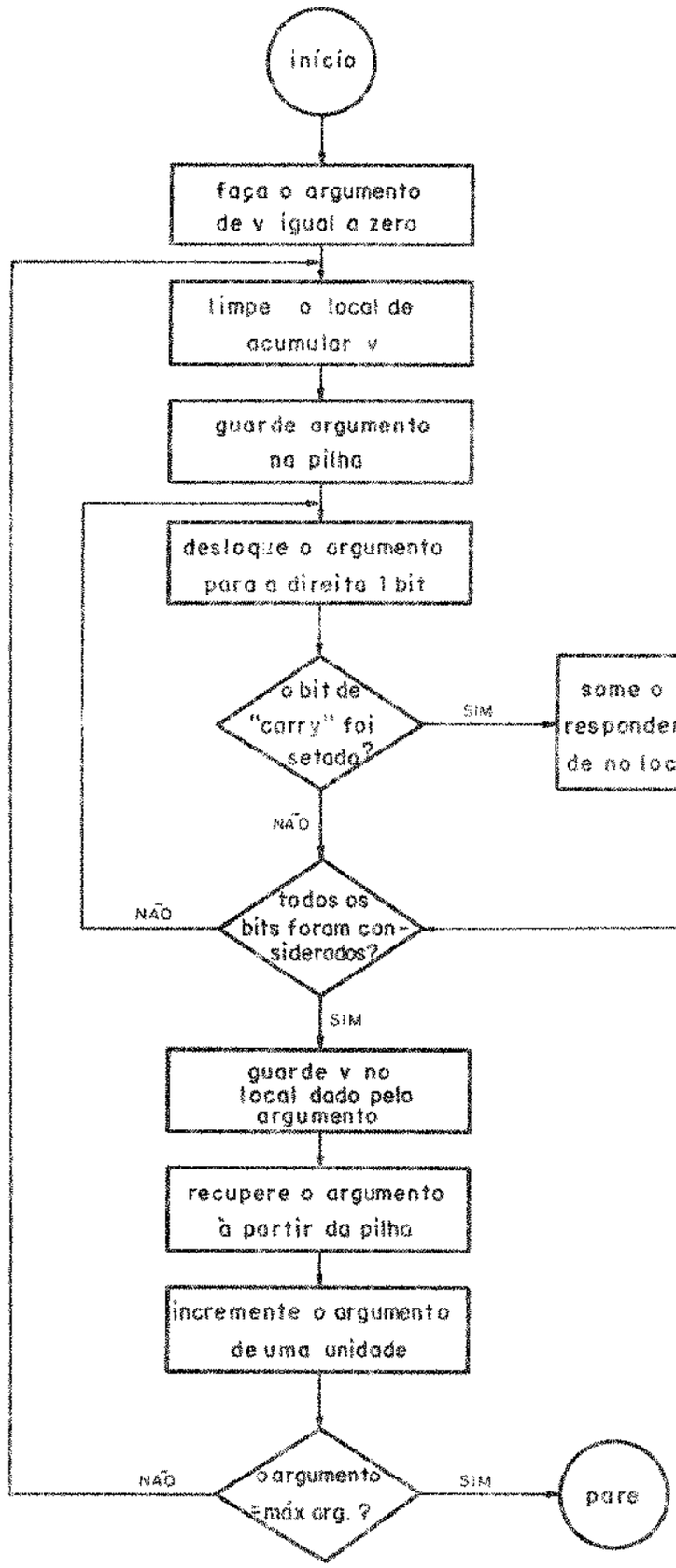



Fig. 2.4. Fluxograma para cálculo da tabela com os valores de "v".

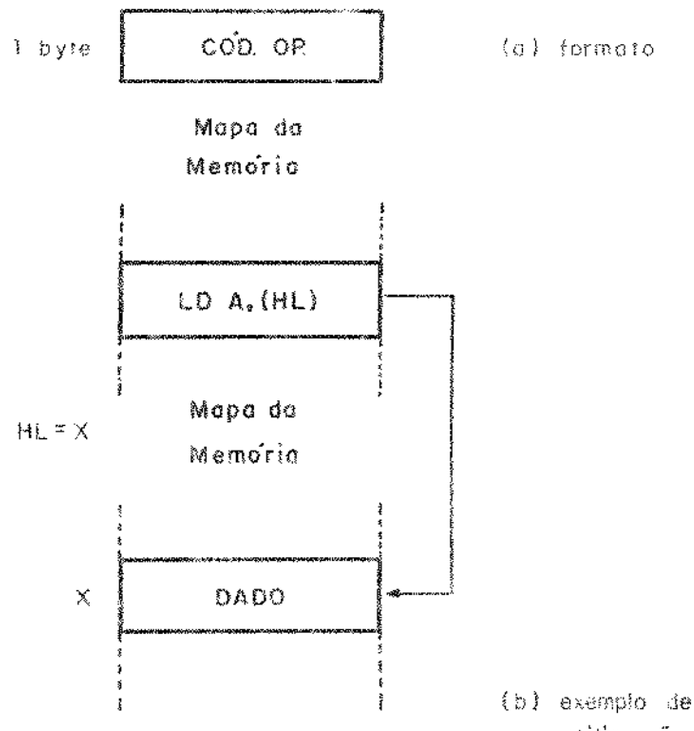



Fig. 2.5. Modo de endereçamento indireto por registro.

da como "modo de endereçamento direto"(d.a.m.).[1] Neste caso temos uma instrução cujo formato é indicado na figura 2.6.a.Nesta instrução, os bytes 2 e 3 especificam uma posição na memória. O conteúdo de 8 bits desta posição é guardado no acumulador. Um exemplo de uso é indicado na figura 2.6.b.

Se usarmos o modo de endereçamento indireto, com o par HL, então a estrutura do programa fica como indicado na figura 2.7. Neste caso supomos que a estrutura de implementação segue a forma indicada na figura 2.2 sem divisão dos "buffers". Assim, uma instrução inicial carrega o par HL com o endereço EEH. Quando a instrução LD A, (HL) aparecer, então o endereço EEH irá aparecer no barramento selecionando o "buffer" B1, e o vetor correspondendo aos bits menos significativos dos dados armazenados no registro X servirão como endereço para a RAM que contém a tabela. O dado na saída de dados da RAM cujo endereço é apontado pelo vetor em RX, é carregado no acumulador, sendo, em seguida, multiplicado por $2^{-1}$ através de uma instrução de deslocamento aritmético para a direita do acumulador (SRA).

Durante a execução desta instrução o registro de deslocamento RX recebe um pulso de relógio através da ação re-

---

(1) Este tipo de recurso é encontrado mais facilmente nos microprocessadores de 16 bits,como o F100-L da Ferranti, p. ex.; os microprocessadores Z-80 ou 8080 possuem este modo somente para as instruções de carregamento,mas não para operações aritméticas de soma e subtração.
Numa estrutura com vários "buffers", com o microprocessador Z-80, a seleção de um deles pode ser feita usando o "modo de endereçamento indexado" através dos registradores IX e IY (ver Barden Jr. [3]).

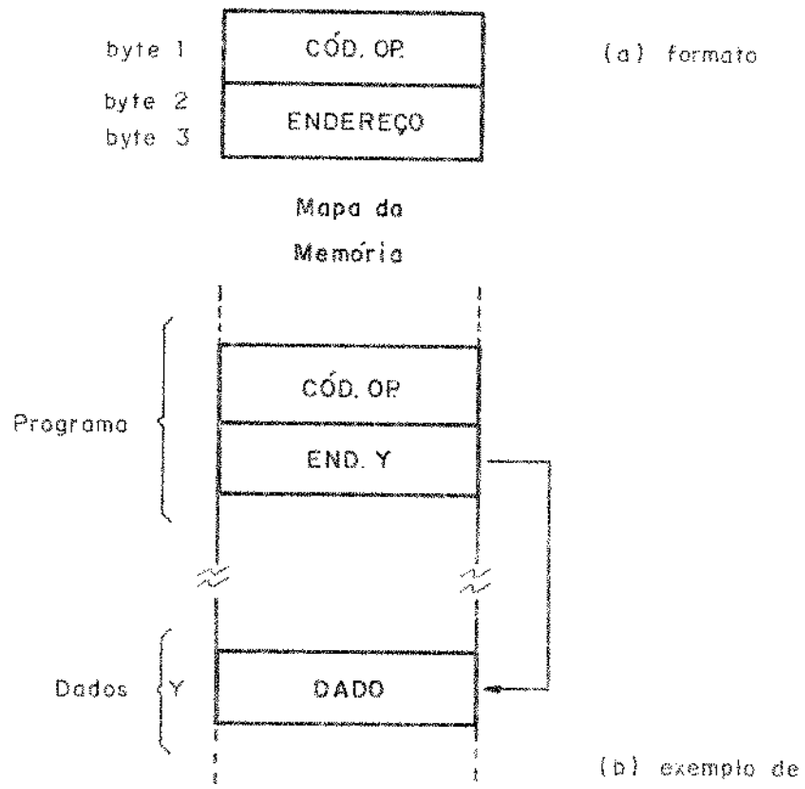



Fig. 2.6. Modo de endereçamento direto.

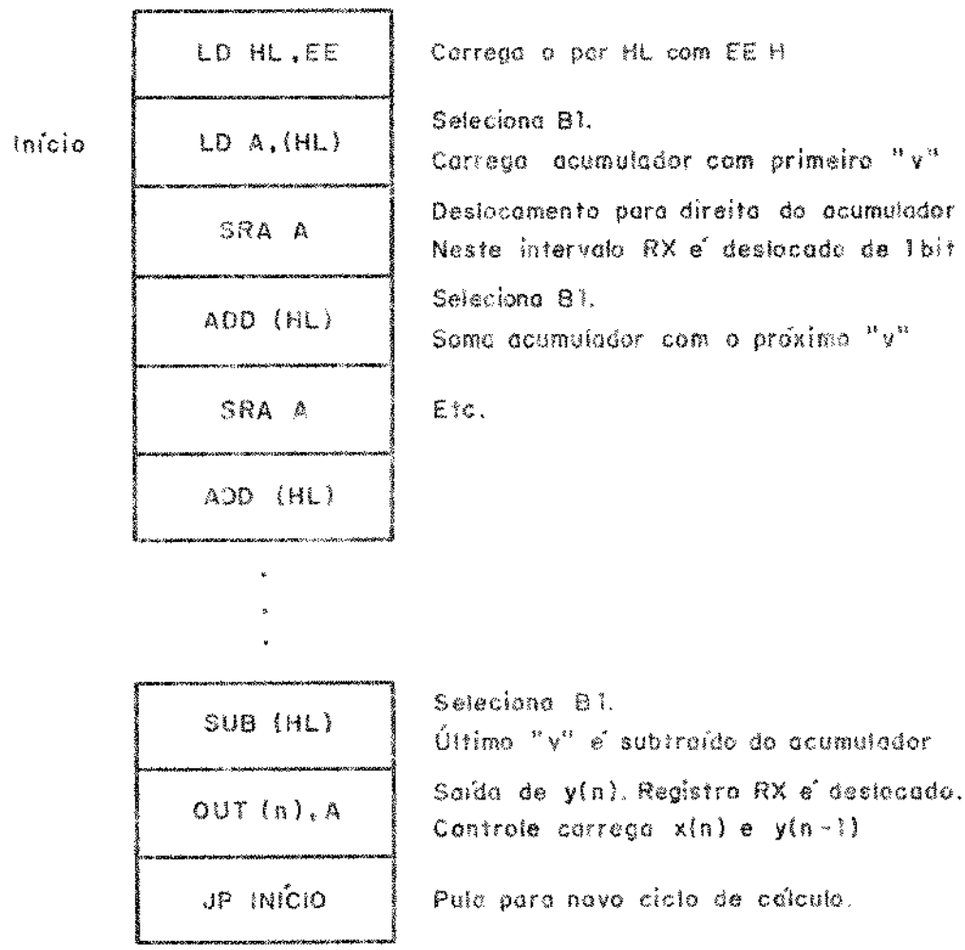


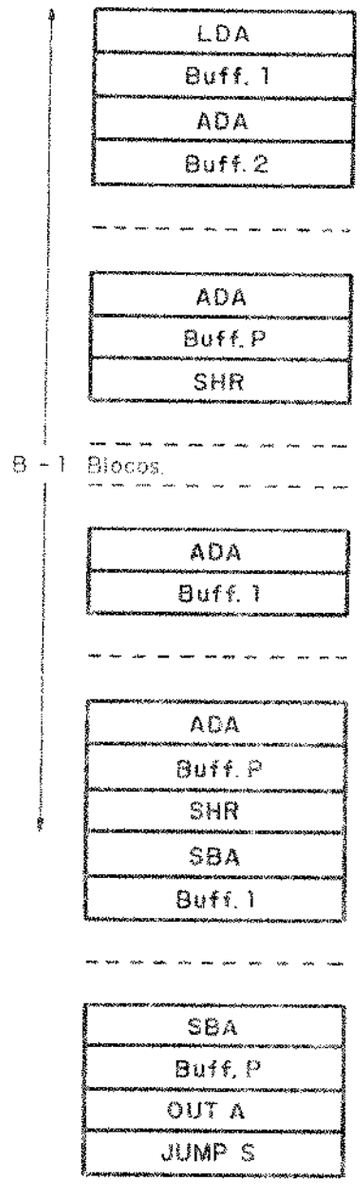

Fig. 2.7. Mapa da memória de programa para o modo de endereçamento indireto.

tardada do sinal de selecionamento de B1. Isto causa um deslocamento de 1 bit em todos os registros de RX, fazendo com que um novo argumento apareça na saída de RX. Como sabemos, a saída de RX é o argumento de um vetor de endereços "v", mencionado na equação (2.12). Quando a instrução ADD (HL) é executada o "buffer" B1 é selecionado novamente,agora com outro argumento. O dado correspondendo a este argumento é somado ao conteúdo prévio do acumulador. Novamente os conteúdos de RX recebem um pulso de relógio durante a execução de uma instrução SRA A, e assim por diante. Na oitava vez que o "buffer" B1 é selecionado, os bits de sinal estão endereçando a RAM. O dado na saída da RAM é, então, subtraído do conteúdo do acumulador, através da instrução SUB A.

Após obter y(n), seu valor é jogado para a saída através de uma instrução OUT(n),A. Nesta operação um pulso de controle deve ser usado para provocar uma nova amostragem no circuito de entrada e para realizar o deslocamento unitário em RX.

Se optarmos pelo modo de endereçamento direto,com divisão em vários grupos de "buffers", então o mapa do programa na memória ficará como indicado na figura 2.8, sendo que a seqüência de cálculo segue a equação (2.20). Os "buffers" de 1 a p aparecem seqüencialmente no mapa de memória, o que significa que cada "buffer" terá um número diferente de endereços para seu acesso. Este múltiplo endereçamento pode ser conseguido através de circuitos decodificadores de endereço. Neste método os "buffers" são endereçados seqüencialmente enquanto o contador de programa avança; nestas ocasiões o acesso do microprocessador à RAM deve ser inibido.

Da descrição é possível verificar que a velocidade do sistema é dependente do número de divisões dos "buffers"




LDA = carrega acumulador.
ADA = Acc ← Acc + operando.
SBA = Acc ← Acc - operando.
SHR = deslocamento aritmético do acumulador p/ a direita.

Fig. 2.8. Mapa do programa na memória.

(p), que, por sua vez, está relacionada com o número de coeficientes do filtro e da capacidade de memória disponível no sistema. O tempo de processamento para cada amostra pode ser calculado aproximadamente usando a seguinte expressão:

$$T_{proc.} = T_{load} + \{p(B-1)-1\}T_{add} + p.T_{sub} + (B-1)T_{shift} + T_{out} + T_{jump} \quad (2.21)$$

Supondo que os tempos para cada instrução são aproximadamente iguais então a equação (2.21) pode ser aproximada para:

$$T_{proc.} \cong [B(p+1)+1].T_{inst.} \quad (2.22)$$

## 2.7. CONCLUSÃO

Neste capítulo apresentamos um método para realizar filtragem digital em tempo real. Neste método, para uma determinada estrutura, a freqüência de amostragem não depende do número de coeficientes do filtro, sendo que o aumento destes apenas implica numa maior capacidade de memória do sistema. Uma vez tendo atingido o limite de memória, é possível realizar a divisão dos "buffers" que fornecem os endereços da tabela na memória, permitindo, em troca de uma redução da freqüência de amostragem, implementar filtros com número maior de coeficientes. A estrutura de programação do sistema dependerá das facilidades de programação encontradas nos microprocessadores existentes.

É interessante, ainda, comparar o sistema adotado na figura 2.1, no qual usamos o microprocessador junto com algum circuito adicional, com outro, no qual se realizasse o cál-

...lo de y(n) de modo diverso. Para isso, Farhang-Boroujeny e Hawkins [6] sugerem uma comparação baseada no número de ciclos de máquina necessários para calcular uma saída para filtro passa-baixa com 32 coeficientes usando o microprocessador M6800 da Motorola. Assim, se usássemos a implementação direta, por "software", com sub-rotinas para multiplicação, seriam necessários 6272 ciclos de máquina. Ainda usando a implementação por "software", mas com aritmética distribuída, o número de ciclos de máquina seria 360 (p=4). Na implementação com auxílio de algum "hardware" adicional o número de ciclos se reduziria a 148 . Isto torna evidente a vantagem da implementação da figura 2.1 sobre as outras.

# CAPÍTULO III: PROJETO DOS COEFICIENTES DO FILTRO DIGITAL, SIMULAÇÃO DA RESPOSTA EM FREQÜÊNCIA E PROGRAMAS

## 3.1. COMENTÁRIO INICIAL

O projeto de um filtro digital envolve uma série de fatores diversos como a escolha da estrutura para a realização do filtro, quantização das variáveis de entrada e saída, o cálculo de coeficientes, etc. Uma vez escolhida uma estrutura para implementar o filtro, conforme o esquema apresentado no capítulo II, podemos sugerir uma divisão do projeto em duas partes : a primeira envolvendo a solução de um problema de aproximação de um determinado tipo de filtro, a segunda envolvendo o projeto do filtro quanto ao seu "hardware". As duas partes, são, às vezes, interdependentes, justificando-se a divisão mais para efeitos de clareza de compreensão. Neste capítulo vamos nos dedicar aos detalhes que se relacionam à experiência por nós adquirida na primeira parte, deixando para o capítulo IV a descrição do projeto do "hardware".

Procuramos englobar nesta primeira parte do projeto todos os recursos que desenvolvemos até o ponto de conseguir a memória do sistema gravada com as características de um determinado filtro. Neste sentido podemos sugerir, didaticamente, as seguintes operações:

1) cálculo dos coeficientes da resposta ao impulso do filtro para atender determinadas especificações;
2) quantização dos coeficientes em função da precisão da estrutura a ser adotada;
3) simulação da resposta em freqüência;

4) cálculo da tabela de coeficientes a ser gravada na memória;
5) desenvolvimento do programa de operação do filtro;
6) gravação do programa de operação e da tabela de coeficientes na memória do sistema.

É nosso objetivo neste capítulo, apresentar a quem se proponha a projetar filtros digitais com a estrutura apresentada no capítulo II, na parte de "software", uma seqüência de operações, incluindo a descrição dos recursos utilizados.

## 3.2. DETERMINAÇÃO DA RESPOSTA AO IMPULSO

O problema de calcular coeficientes para atender a determinadas especificações envolve o conhecimento das técnicas de projeto de filtros digitais resumidas no primeiro capítulo para os filtros RIF. Por uma questão de simplicidade iniciamos pelo projeto de um filtro passa-baixa utilizando a técnica das janelas. Entre as janelas sugeridas, escolhemos a de Hamming, cuja equação (1.13), com $\alpha = 0{,}54$, é repetida aqui por comodidade:

$$w_H(n) = \begin{cases} \alpha + (1-\alpha)\cos\left(\frac{2\pi n}{N-1}\right), & |n| \leq \frac{N-1}{2} \\ 0, & \text{outros valores de } n \end{cases} \qquad (3.1)$$

Para o caso de um filtro passa-baixa a resposta em freqüência desejada, já considerando o atraso necessário para formar um filtro causal de fase linear, é definida por

$$H_d(e^{j\omega}) = \begin{cases} e^{-j\omega\alpha}, & |\omega| \leq \omega_c \\ 0, & \text{outro } \omega \text{ qualquer} \end{cases} \qquad (3.2)$$

A correspondente resposta ao impulso é:

$$h_d(n) = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega(n-\alpha)} \, d\omega =$$

$$= \frac{\text{sen } [\omega_c(n-\alpha)]}{\pi(n-\alpha)} \quad , \quad n \neq \alpha \qquad (3.3)$$

Para obter um filtro de fase linear, causal, com duração finita, de comprimento N, devemos fazer:

$$h(n) = h_d(n) \cdot w(n) \qquad (3.4)$$

sendo

$$\alpha = \frac{N-1}{2} \qquad (3.5)$$

Na escolha de um valor para N, evitamos um valor muito alto, uma vez que seria necessário aumentar demasiadamente a memória do sistema, ou efetuar modificações no circuito. A escolha de maiores valores de N é discutida no capítulo V. Assim o valor de N = 7 foi inicialmente escolhido. Dispúnhamos da freqüência de amostragem, $f_a$ =9026,5Hz(1), pois, conforme já comentamos na seção anterior, há uma interdependência entre o projeto de "hardware" e de "software" (um valor de $f_a$ foi inicialmente estimado em função do tempo das instruções, e, posteriormente, com o funcionamento do circuito, medido). Escolhida a freqüência de corte analógica $f_c$ = 500 Hz, podemos proceder ao cálculo do

(1) O projeto poderia iniciar apenas com as especificações da freqüência de corte digital, sendo que a freqüência analógica seria obtida como uma função da freqüência de amostragem usada no sistema.

filtro. Para tornar mais simples esta tarefa, apresentamos no apêndice 3A no final deste capítulo um programa para calcular os valores da resposta ao impulso do filtro; com esse programa chegamos aos resultados apresentados na tabela 3.1.

Outra opção para projeto de filtros é usarmos o programa Fortran desenvolvido por McClellan, apresentado em Rabiner e Gold [15] que, conforme comentamos no primeiro capítulo, utiliza a técnica de projeto de filtros ótimos. Este método permite um melhor controle dos parâmetros envolvidos. O programa tem como entrada: a) o valor de N; b) o tipo de filtro (passa-faixa, diferenciador ou transformador de Hilbert); c) limite das faixas de freqüência; d) valor desejado (ideal) para as faixas (0 ou 1); e) peso desejado para o "ripple" em cada faixa. Portanto, a ondulação("ripple")da resposta em freqüência não é especificada, sendo função destes parâmetros. Na saída do programa temos os valores da resposta ao impulso e os valores das freqüências extremas da resposta em freqüência. A seguir apresentamos alguns exemplos de filtros calculados por este método.

EXEMPLO 3.1: Filtro passa-baixa, N = 7.

Dados de entrada:

7, 1, 2, 16

0., 0.0554, 0.0997, 0.5

1., 0.

1., 2.

A primeira linha corresponde aos valores de N, tipo do filtro (1 = passa-faixa múltiplo; 2 = diferenciador; 3 = transformador de Hilbert), número de faixas desejadas e a densidade da grade de freqüências, respectivamente. Na segunda, temos as freqüên-

| n | $\omega(n)$ | $h_d(n)$ | $h(n) = \omega(n).h_d(n)$ |
|---|---|---|---|
| 0 | 0,0800 | 0,0917 | 0,0073 |
| 1 | 0,3100 | 0,1020 | 0,0316 |
| 2 | 0,7700 | 0,1086 | 0,0836 |
| 3 | 1,0000 | 0,1108 | 0,1108 |
| 4 | 0,7700 | 0,1086 | 0,0836 |
| 5 | 0,3100 | 0,1020 | 0,0316 |
| 6 | 0,0800 | 0,0917 | 0,0073 |

Resposta ao impulso de um filtro Passa Baixa,
$n = 7$, $fa = 9026{,}5$ Hz, $fc = 500$ Hz,
calculado pelo programa do apêndice 3A

**Tabela 3.1.**

cias digitais (normalizadas) limites das faixas. Na terceira linha estão os valores da resposta em freqüência desejada (portanto é um filtro passa-baixa). Na última linha temos o peso que desejamos para a ondulação nas faixas. Os coeficientes da resposta deste filtro (incluindo os valores das freqüências extremas) são apresentados na tabela 3.2. Nos exemplos seguintes, os dados seguem a mesma estrutura descrita.

EXEMPLO 3.2: Filtro passa alta, N = 7

Dados de entrada:

7, 1, 2, 16

0., 0.2216, 0.2770, 0.5

0., 1.

2., 1.

Os coeficientes para este filtro aparecem na tabela 3.3.

EXEMPLO 3.3: Filtro passa-faixa, N = 7

Dados de entrada:

7, 1, 3, 16

0., 0.1108, 0.1662, 0.2770, 0.3324, 0.5

0., 1., 0.

1., 1., 1.

Os coeficientes para este filtro aparecem na tabela 3.4.

EXEMPLO 3.4: Filtro passa-baixa, N = 24

Dados de entrada:

24, 1, 2, 16

0., 0.2130, 0.2879, 0.5

1., 0.

1., 1.

Os coeficientes para este filtro aparecem na tabela 3.5.

```
                        FINITE IMPULSE RESPONSE (FIR)
                        LINEAR PHASE DIGITAL FILTER DESIGN
                        REMEZ EXCHANGE ALGORITHM                    58

                        BANDPASS FILTER

                FILTER LENTH=  7

                ***** IMPULSE RESPONSE *****
                     H(  1) = 0.1495717*E+00 = H(  7)
                     H(  2) = 0.8445927*3E-01 = H(  6)
                     H(  3) = -3.2543623*7E-01 = H(  5)
                     H(  4) = 0.1666730*1E+00 = H(  4)

                          BAND  1          BAND  2          BAND
 LOWER BAND EDGE       0.000000000      0.079200000
 UPPER BAND EDGE       0.055400000      0.500000000
 DESIRED VALUE         1.000000000      0.000000000
 WEIGHTING             1.000000000      2.000000000
 DEVIATION             0.439438142      0.219719071
 DEVIATION IN DB      -7.142945091     -13.16264493

 EXTREMAL FREQUENCIES
    0.0554000    0.0997000    0.1934500    0.3418875    0.5000000
```

Tabela 3.2. Filtro passa - baixa, N = 7

```
                        FINITE IMPULSE RESPONSE (FIR)
                        LINEAR PHASE DIGITAL FILTER DESIGN
                        REMEZ EXCHANGE ALGORITHM

                        BANDPASS FILTER

                FILTER LENTH=  7

                ***** IMPULSE RESPONSE *****
                     H(  1) = 0.1154214*9E+00 = H(  7)
                     H(  2) = 0.7773040*4E-01 = H(  6)
                     H(  3) = -0.3949627*8E+00 = H(  5)
                     H(  4) = 0.5693406*9E+00 = H(  4)

                          BAND  1          BAND  2          BAND
 LOWER BAND EDGE       0.000000000      0.276999999
 UPPER BAND EDGE       0.221600000      0.500000000
 DESIRED VALUE         0.000000000      1.000000000
 WEIGHTING             2.000000000      1.000000000
 DEVIATION             0.165705508      0.331411015
 DEVIATION IN DB      -15.61326122     -9.592661262

 EXTREMAL FREQUENCIES
    0.0000000    0.1406250    0.2214000    0.2770000    0.5000000
```

Tabela 3.3. Filtro passa - alta, N = 7

```
                    FINITE IMPULSE RESPONSE (FIR)
                    LINEAR PHASE DIGITAL FILTER DESIGN        59
                    REMEZ EXCHANGE ALGORITHM

                    BANDPASS FILTER

          FILTER LENTH=  7

          ***** IMPULSE RESPONSE *****
               H(  1) =-0.70084723E-01 = H(  7)
               H(  2) =-0.38300056E+00 = H(  6)
               H(  3) = 0.70084723E-01 = H(  5)
               H(  4) = 0.24261551E+00 = H(  4)

                          BAND  1          BAND  2          BAND  3
LOWER BAND EDGE        0.000000000      0.166200001      0.332400002
UPPER BAND EDGE        0.110800000      0.276999999      0.500000000
DESIRED VALUE          0.000000000      1.000000000      0.000000000
WEIGHTING              1.000000000      1.000000000      1.000000000
DEVIATION              0.319585603      0.319585603      0.319585603
DEVIATION IN DB       -9.908255935     -9.908255935     -9.908255935

EXTREMAL FREQUENCIES
   0.0000000   0.1108000   0.2770000   0.3324000   0.5000000
```

**Tabela 3.4.** Filtro passa - faixa, N = 7

```
                    FINITE IMPULSE RESPONSE (FIR)
                    LINEAR PHASE DIGITAL FILTER DESIGN
                    REMEZ EXCHANGE ALGORITHM

                    BANDPASS FILTER

          FILTER LENTH= 24

          ***** IMPULSE RESPONSE *****
               H(  1) =-0.61479290E-02 = H( 24)
               H(  2) =-0.13164053E-01 = H( 23)
               H(  3) = 0.12965059E-01 = H( 22)
               H(  4) = 0.14519336E-01 = H( 21)
               H(  5) =-0.18845799E-01 = H( 20)
               H(  6) =-0.25556880E-01 = H( 19)
               H(  7) = 0.32278748E-01 = H( 18)
               H(  8) = 0.43442139E-01 = H( 17)
               H(  9) =-0.58314562E-01 = H( 16)
               H( 10) =-0.86462992E-01 = H( 15)
               H( 11) = 0.14713271E+00 = H( 14)
               H( 12) = 0.44984133E+00 = H( 13)

                          BAND  1          BAND  2          BAND
LOWER BAND EDGE        0.000000000      0.237999999
UPPER BAND EDGE        0.213000000      0.500000000
DESIRED VALUE          1.000000000      0.000000000
WEIGHTING              1.000000000      1.000000000
DEVIATION              0.016665615      0.016665615
DEVIATION IN DB      -35.563573360    -35.563573360

EXTREMAL FREQUENCIES
   0.0000000   0.0416667   0.0859375   0.1276042   0.1666667
   0.1979167   0.2130000   0.2880000   0.3036250   0.3374792
   0.3791459   0.4260209   0.4755001
```

**Tabela 3.5.** Filtro passa - baixa, N = 24

## 3.3. QUANTIZAÇÃO DOS COEFICIENTES

O problema da quantização dos coeficientes deve ser analisado em função do comprimento finito da palavra do sistema. Em nosso caso, temos um comprimento igual a 8 bits. Assim, dado um determinado coeficiente decimal, com várias casas após a vírgula, ele será convertido num número binário, na representação complemento de dois, com 8 bits; podemos optar por arredondar ou por truncar o número binário simplesmente. Maiores detalhes sobre truncamento ou arredondamento podem ser vistos em Oppenheim [11]. Considerando os coeficientes calculados na tabela 3.1, por exemplo, vemos na tabela 3.6. como ficam representados os coeficientes após o processo de arredondamento.

## 3.4. SIMULAÇÃO DA RESPOSTA EM FREQÜÊNCIA

A simulação da resposta em freqüência é uma parte muito importante no projeto por fornecer uma previsão inicial clara dos resultados a serem obtidos. Esta fase deve preceder a parte de medidas no circuito, pois sem ela não teríamos referencial para avaliar os resultados das medidas. A obtenção da resposta em freqüência é direta, através do uso das equações (1.9) e (1.10) cuja magnitude reproduzimos a seguir. Assim, para filtros RIF, simétricos, N ímpar, tem-se:

$$H^*(e^{j\omega}) = \sum_{n=0}^{(N-1)/2} a(n) \cos(\omega n) \qquad (3.6)$$

onde

$$a(n) = 2h\left[\left(\frac{N-1}{2}\right) - n\right], \quad n = 1, 2, \ldots, (N-1)/2$$

| n | $h(n)_{10}$ | $h(n)_2$ | $h(n)_{16}$ |
|---|---|---|---|
| 0 | 0,0073 | 0,0000001 | 01 |
| 1 | 0,0316 | 0,0000100 | 04 |
| 2 | 0,0836 | 0,0001010 | 0B |
| 3 | 0,1108 | 0,0001110 | 0E |
| 4 | 0,0836 | 0,0001010 | 0B |
| 5 | 0,0316 | 0,0000100 | 04 |
| 6 | 0,0073 | 0,0000001 | 01 |

Tabela 3.6. Quantização dos coeficientes do filtro da tabela 3.1 .

e

$$a(0) = h\,[(N-1)/2\,]$$

Para N par, temos:

$$H^{*}(e^{j\omega}) = \sum_{n=0}^{N/2-1} b(n)\cos\left[\omega\left(\frac{N}{2} - n - \frac{1}{2}\right)\right]$$

sendo

$$b(n) = 2h\left(\frac{N}{2} - n\right), \qquad n = 1, 2, \ldots, \frac{N}{2}$$

Estas equações podem ser facilmente usadas para o traçado da resposta em freqüência com auxílio de uma calculadora com traçador gráfico, como por exemplo o modelo 9820A da HP. Desta maneira simulamos as respostas em freqüência para os filtros das tabelas 3.1 a 3.5, que aparecem nas figuras 3.1 a 3.5, respectivamente. Os filtros das tabelas 3.2 a 3.4 tiveram seus coeficientes divididos por dois para evitar problemas de "overflow". Esta divisão implica apenas no deslocamento da resposta em freqüência, de modo uniforme, na vertical.

## 3.5. CÁLCULO DA TABELA DE COEFICIENTES

Uma vez calculados os coeficientes para um determinado tipo de filtro, devemos calcular a tabela que contém as combinações possíveis entre os seus valores, conforme o algoritmo de cálculo descrito na seção 2.3. Esta tabela deve ser gravada na memória do sistema. Para formar a tabela seguindo o fluxograma da figura 2.4 desenvolvemos um programa em linguagem de máquina, utilizando o Sistema de Desenvolvimento de Programas (SDP-85) do Centro de Engenharia Biomédica da Unicamp. Este pro-

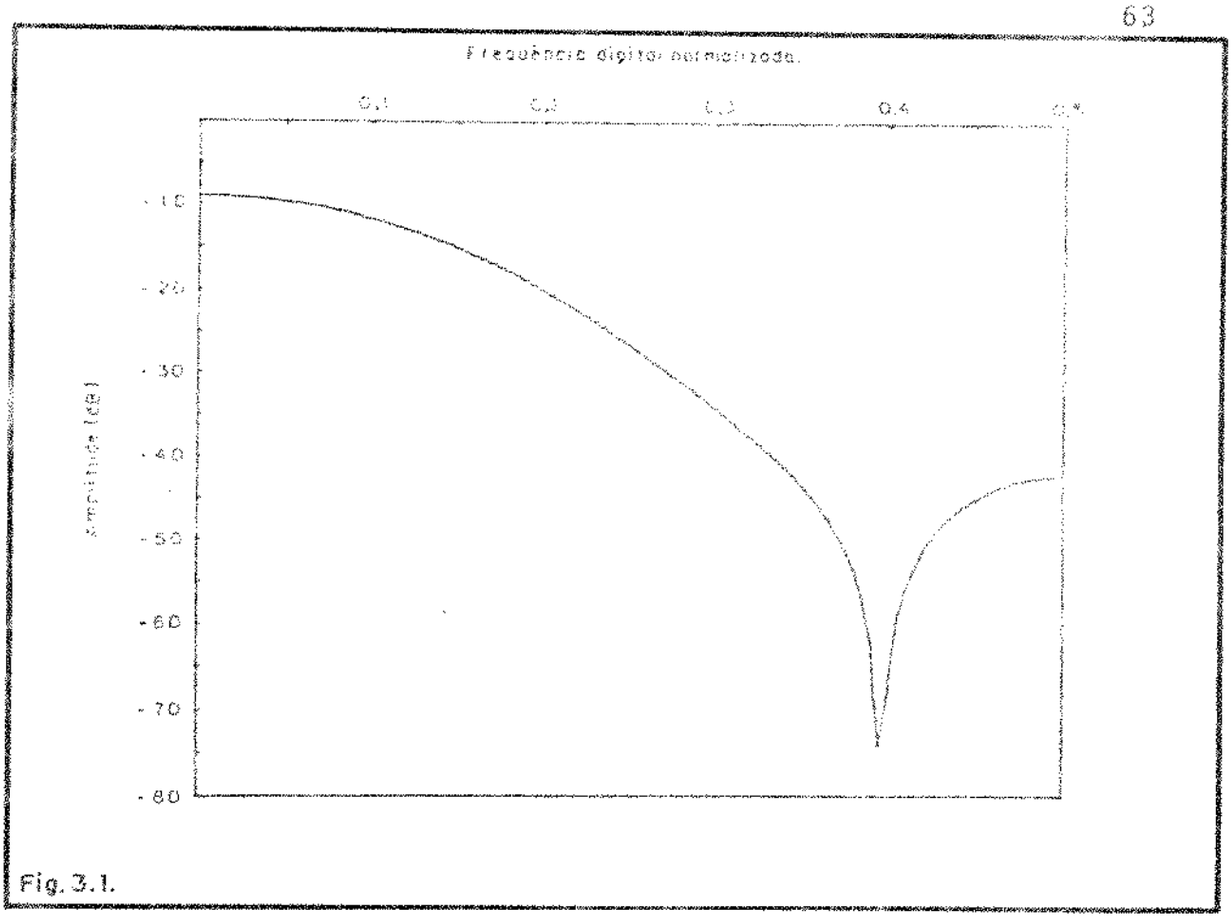

Frequência digital normalizada.
0,1
0,2
0,3
0,4
0,5
-10
-20
-30
-40
-50
-60
-70
-80
Amplitude (dB)

Fig. 3.1.

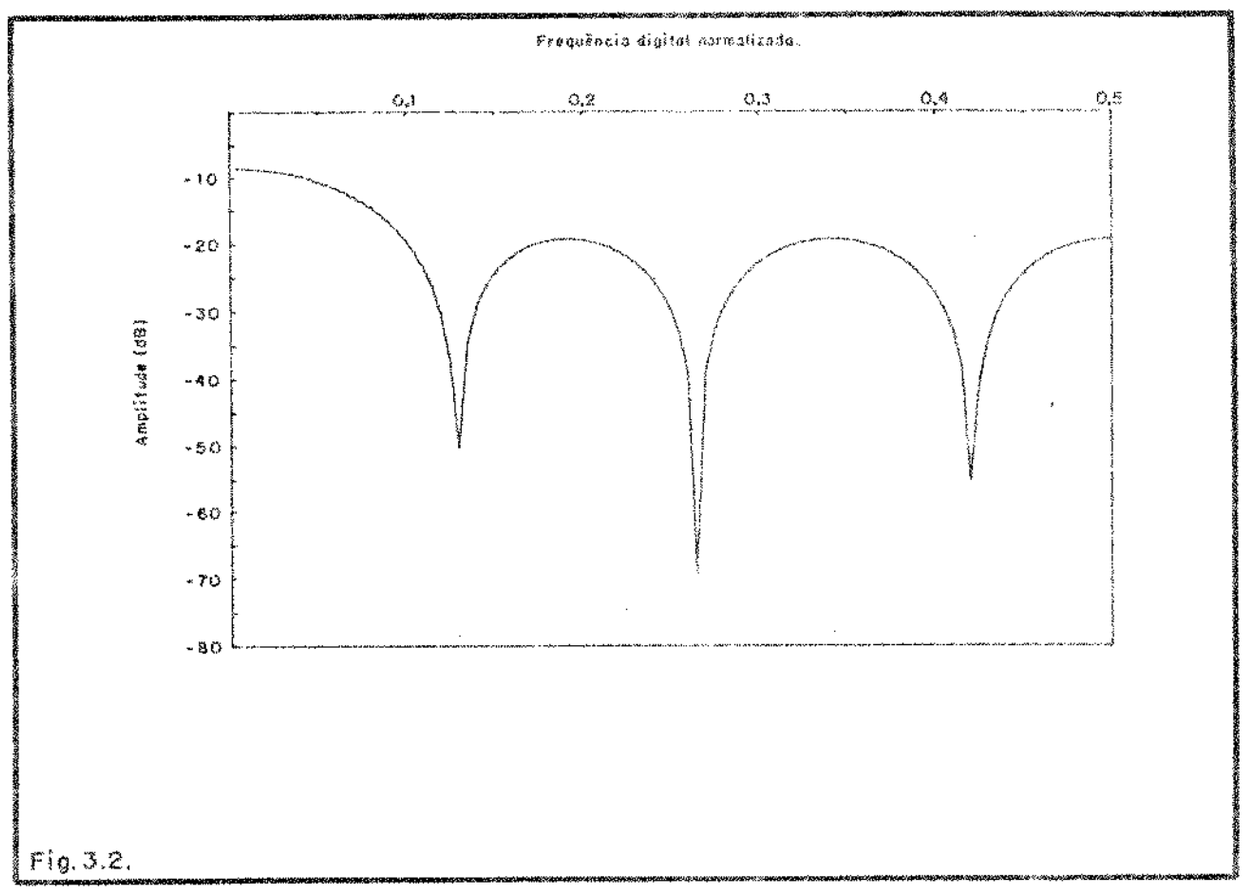

Frequência digital normalizada.
0,1
0,2
0,3
0,4
0,5
-10
-20
-30
-40
-50
-60
-70
-80
Amplitude (dB)

Fig. 3.2.

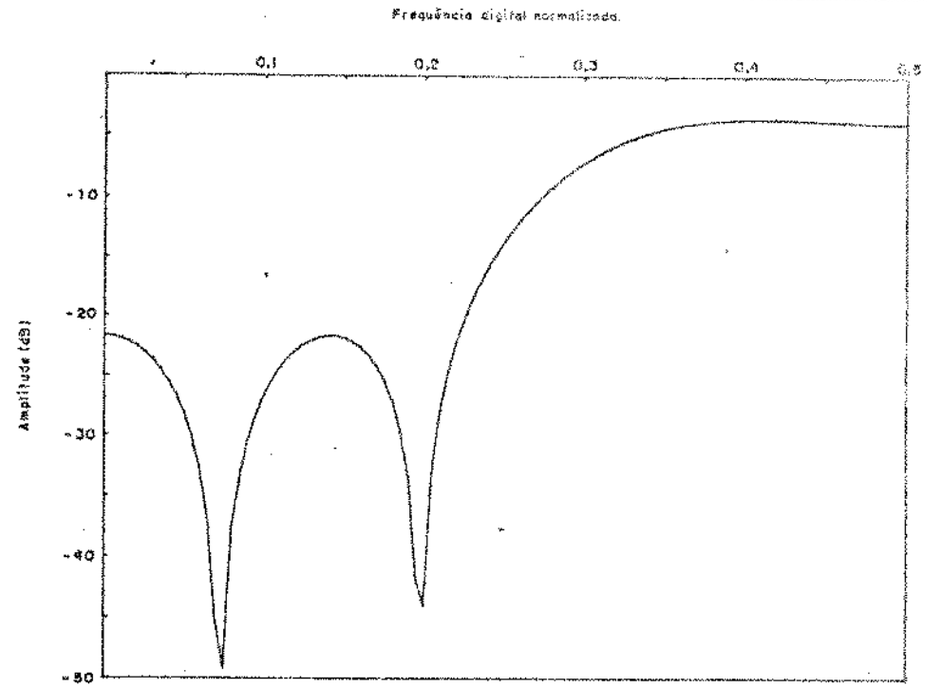

Frequência digital normalizada.
0,1
0,2
0,3
0,4
0,5
-10
-20
-30
-40
-50
Amplitude (dB)

Fig. 3.3.

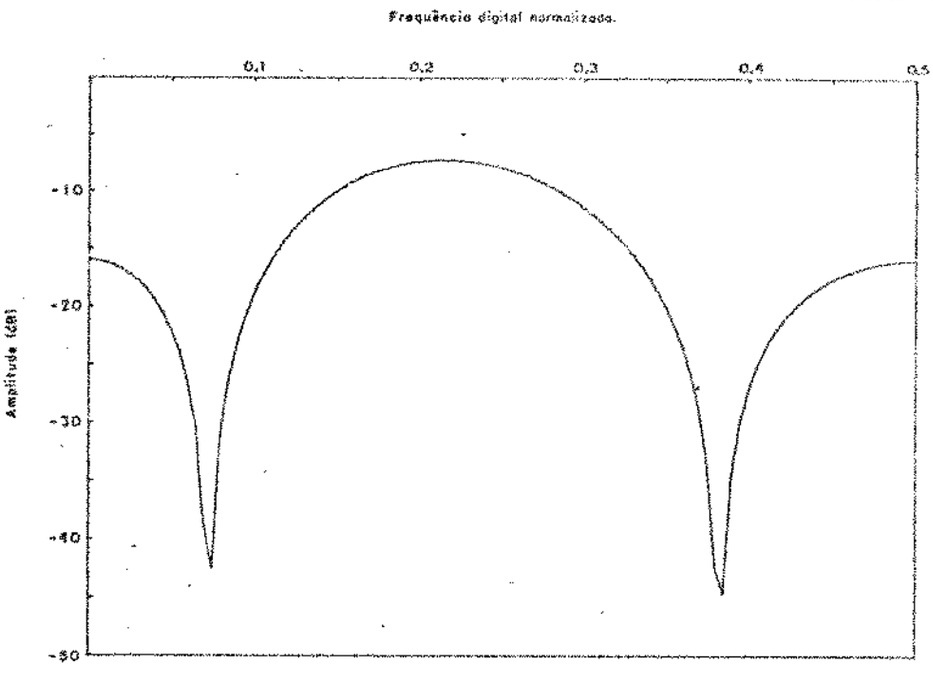

Frequência digital normalizada.
0,1
0,2
0,3
0,4
0,5
-10
-20
-30
-40
-50
Amplitude (dB)

Fig. 3.4.

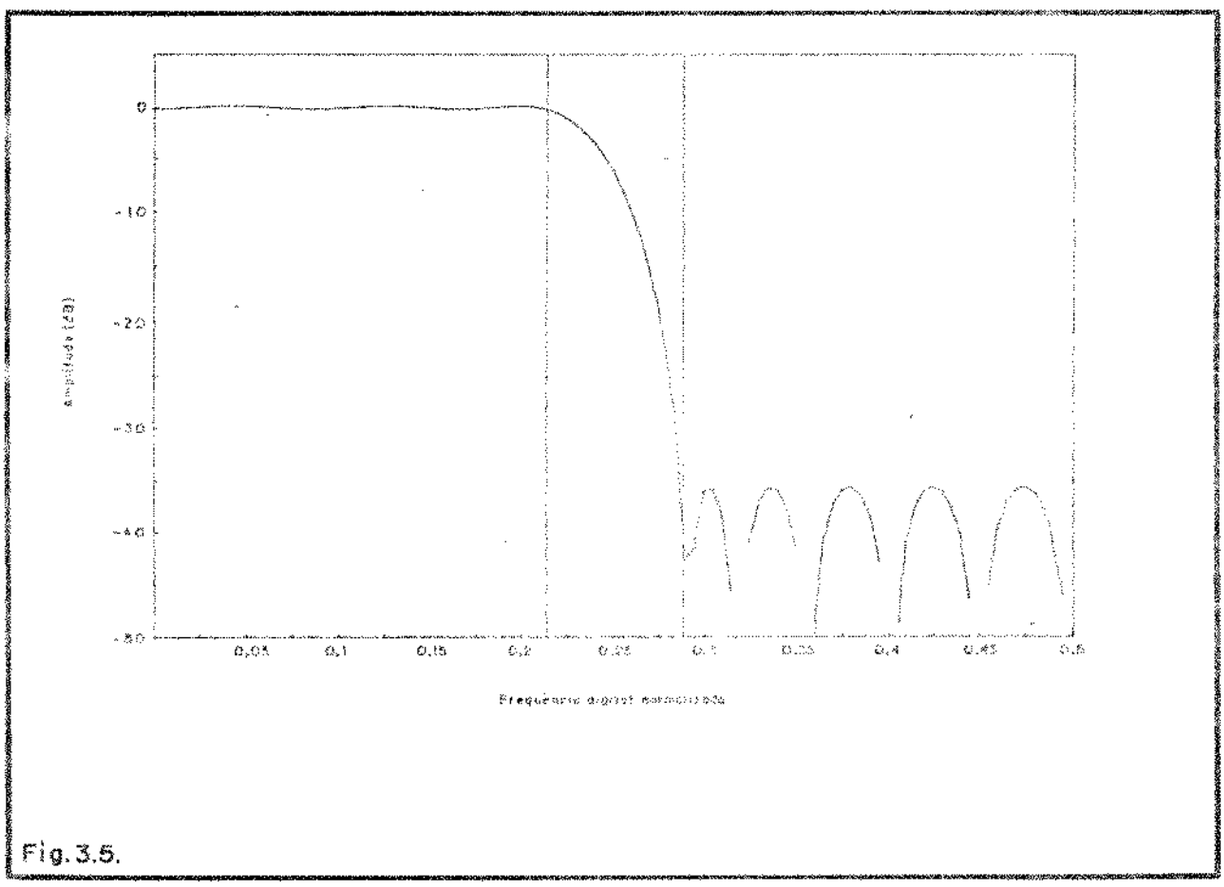



Fig. 3.5.

grama se encontra no apêndice 3B deste capítulo e está adaptado para os endereços do sistema SDP-85. Com ele, pudemos calcular as combinações de coeficientes para os filtros das tabelas 3.1 a 3.4 que aparecem na tabela 3.7 (a até d), respectivamente, e são armazenadas na memória do filtro, através do uso de um gravador de EPROM integrado ao sistema.

## 3.6. PROGRAMA DE OPERAÇÃO DO FILTRO

O programa de operação do filtro foi formulado considerando o número de repartições p = 1. Para desenvolvê-lo e gravá-lo em EPROM usamos o mesmo recurso da seção anterior, isto é, o Sistema de Desenvolvimento de Programas do CEB/ UNICAMP (SDP-85). O programa é mostrado na tabela 3.8 a seguir.

As duas primeiras instruções servem para a definição do endereço da pilha operacional e do endereço fictício a ser carregado no par HL, respectivamente. O endereço fictício escolhido, 0A00H, inibe o acesso do microprocessador à memória,e permite o acesso à tabela, através de um endereçamento por RX. Este endereço irá aparecer sempre que um endereçamento através do par HL for executado. É o caso das operações de carregamento, soma e subtração com o acumulador. A instrução SRA A com código CB2FH produz um deslocamento aritmético para a direita no acumulador, conforme a fig. 3.6. Esta instrução mantém o sinal do número no acumulador.

A instrução OUT terá duas finalidades no circuito. A primeira é produzir a saída de um dado, a segunda é, através da ativação do sinal $\overline{IORQ}$, comandar o início de uma nova conversão no A/D. Para que o circuito tenha tempo suficiente para con-

(a)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 04 | 05 | 08 | 0C | 0F | 10 | 0E | 0F | 12 | 13 | 19 | 1A | 1D | 1E | 08 | 0C | 0F | 10 | 16 | 17 | 1A | 18 | 19 | 1A | 1D | 1E | 24 | 25 | 28 | 29 | 04 |
| 03 | 08 | 09 | 0F | 10 | 13 | 14 | 12 | 13 | 16 | 17 | 10 | 1E | 21 | 22 | 0F | 10 | 13 | 14 | 1A | 18 | 1E | 1F | 1D | 1E | 21 | 22 | 28 | 29 | 2C | 20 | 01 | 02 |
| 05 | 06 | 0C | 00 | 10 | 11 | 0F | 10 | 13 | 14 | 1A | 18 | 1E | 1F | 0C | 00 | 10 | 11 | 17 | 18 | 18 | 1C | 14 | 18 | 1E | 1F | 25 | 26 | 29 | 2A | 05 | 06 | 09 |
| 0A | 10 | 11 | 14 | 13 | 13 | 14 | 17 | 18 | 1E | 1F | 22 | 23 | 10 | 11 | 14 | 15 | 18 | 1C | 1F | 20 | 1E | 1F | 22 | 23 | 29 | 2A | 20 | 2E | | | | |

(b)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 09 | 05 | 0E | 06 | 0F | 0B | 14 | 06 | 0F | 08 | 14 | 0C | 15 | 11 | 1A | 06 | 0F | 08 | 14 | 0C | 15 | 11 | 1A | 0C | 15 | 11 | 1A | 12 | 18 | 17 | 20 | 05 |
| 0E | 04 | 13 | 09 | 14 | 10 | 19 | 08 | 14 | 10 | 19 | 11 | 1A | 16 | 1F | 08 | 14 | 10 | 19 | 11 | 1A | 16 | 1F | 11 | 1A | 16 | 1F | 17 | 20 | 1C | 23 | 09 | 12 |
| 0E | 17 | 0F | 18 | 14 | 10 | 0F | 18 | 14 | 10 | 15 | 1E | 1A | 23 | 0F | 18 | 14 | 10 | 15 | 1E | 1A | 23 | 15 | 1E | 1A | 23 | 18 | 24 | 20 | 29 | 28 | 17 | 13 |
| 1C | 14 | 10 | 19 | 22 | 14 | 10 | 19 | 22 | 1A | 23 | 1F | 28 | 14 | 10 | 19 | 22 | 1A | 23 | 1F | 28 | 1A | 23 | 1F | 26 | 20 | 29 | 25 | 2C | | | | |

(c)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 03 | 05 | 08 | E7 | E4 | EC | EF | 24 | 27 | 29 | 2C | 08 | 0E | 10 | 13 | E7 | E4 | EC | EF | 0E | 01 | 03 | 06 | 08 | 0E | 10 | 13 | F2 | F5 | F7 | FA | 05 |
| 08 | 0A | 0D | EC | EF | F1 | F4 | 29 | 2C | 2E | 31 | 10 | 13 | 15 | 18 | EC | EF | F1 | F4 | 03 | 06 | 09 | 0B | 10 | 13 | 15 | 18 | F7 | FA | FC | FF | 01 | 06 |
| 08 | 0B | EA | ED | EF | F2 | 27 | 2A | 2C | 2F | 0E | 11 | 13 | 16 | EA | ED | EF | F2 | 01 | 04 | 06 | 09 | 0E | 11 | 13 | 16 | F5 | F8 | FA | FD | 08 | 08 | 00 |
| 10 | EF | F2 | F4 | F7 | 2C | 2F | 31 | 34 | 13 | 16 | 18 | 18 | EF | F2 | F4 | F7 | 06 | 09 | 0B | 0E | 13 | 16 | 18 | 18 | FA | FD | FF | 02 | | | | |

(d)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | FC | FE | FA | 04 | 00 | 02 | FE | 10 | 0C | 0E | 0A | 14 | 10 | 12 | 0E | 04 | 00 | 02 | FE | 08 | 04 | 06 | 02 | 14 | 10 | 12 | 0E | 18 | 14 | 16 | 12 | FE |
| FA | FC | F8 | 02 | FE | 00 | FC | 0E | 0A | 0C | 08 | 12 | 0E | 10 | 0C | 02 | FE | 00 | FC | 06 | 02 | 04 | 00 | 12 | 0E | 10 | 0C | 16 | 12 | 14 | 10 | FC | F8 |
| FA | F6 | 00 | FC | FE | FA | 0C | 08 | 0A | 06 | 10 | 0C | 0E | 0A | 00 | FC | FE | FA | 04 | 00 | 02 | FE | 10 | 0C | 0E | 0A | 14 | 10 | 12 | 0E | F4 | F6 | F8 |
| F4 | FE | FA | FC | F8 | 0A | 06 | 08 | 04 | 0E | 0A | 0C | 08 | FE | FA | FC | F8 | 0E | 16 | 00 | FC | 0E | 0A | 0C | 08 | 12 | 0E | 10 | 0C | | | | |

Tabela 3.7. Combinação dos coeficientes para os filtros das tabelas 3.1 a 3.4 (a serem gravadas na memória do sistema).

```
                        0000            ORG     0000H
                        0001 ;
                        0002 ;
                        0003 ;          FILTRO DIGITAL
                        0004 ;
                        0005 ;  DESENVOLVIDO POR LUCIANO BARACHO ROCHA
                        0006 ;
                        0007 ;  PROGRAMA DE OPERACAO DO FILTRO
                        0008 ;
                        0009 ;  SEM-CEB/UNICAMP  -  CEFET-PR
                        0010 ;
0000 31 F0 0F           0011            LD      SP,STACK        ;PILHA
0003 21 00 0A           0012            LD      HL,FICT         ;ENDERECO FALSO
0006 7E                 0013 INICIO     LD      A,(HL)          ;ACESSO AO DADO DA
                                                  TABELA
0007 CB 2F              0014            SRA     A
0009 86                 0015            ADD     A,(HL)
000A CB 2F              0016            SRA     A
000C 86                 0017            ADD     A,(HL)
000D CB 2F              0018            SRA     A
000F 86                 0019            ADD     A,(HL)
0010 CB 2F              0020            SRA     A
0012 86                 0021            ADD     A,(HL)
0013 CB 2F              0022            SRA     A
0015 86                 0023            ADD     A,(HL)
0016 CB 2F              0024            SRA     A
0018 86                 0025            ADD     A,(HL)
0019 CB 2F              0026            SRA     A
001B 96                 0027            SUB     (HL)
001C D3 00              0028            OUT     0,A             ;SAIDA DE Y(N)
001E 76                 0029            HALT                    ;NOVA AMOSTRAGEM
001F C3 06 00           0030            JP      INICIO
                        0031 ;
                        0032 ;
                        0033 ;DEFINICAO DE ENDERECOS
                        0034 ;
                        0035 ;
                        0036 STACK      EQU     0FF0H           ;ENDERECO DA PILHA
                        0037 FICT       EQU     0A00H           ;ENDERECO FALSO
                        0038 ;
                        0039 ;
                        0040            END
ERR COUNT: 000

FICT   0A00       INICIO 0006       STACK  0FF0

                        0000            ORG     0066H
                        0001 ;
                        0002 ;
                        0003 ; ENDERECO DA INTERRUPCAO
                        0004 ;
                        0005 ;
                        0006 ; DADO JA' CONVERTIDO PELO CONVERSOR A/D
                        0007 ;
                        0008 ;
0066 C9                 0009            RET
                        0010 ;
                        0011 ;
                        0012            END
ERR COUNT: 000
```

**Tabela 3.8.** Programa de operação do filtro.

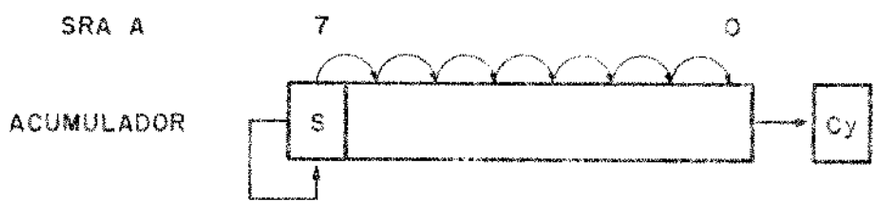



Fig. 3.6. Deslocamento aritmético do acumulador.

verter um novo dado, antes de iniciar novo ciclo de operação, a instrução seguinte é um HALT. Nesta situação o microprocessador, gera continuamente ciclos de busca de códigos de operação (ou ciclos M1), e o contador de programa (PC) não avança. A saída do estado de HALT é feita por uma interrupção não mascarada, através do sinal $\overline{NMI}$. Um degrau descendente na entrada NMI do Z-80 faz com que o conteúdo do PC seja guardado na pilha operacional, e o indicador da pilha (SP) decrementado. Então a instrução na posição 0066H é executada. Esta instrução é simplesmente uma instrução RET, que restaura o conteúdo do PC armazenado na pilha e faz com que a CPU inicie a execução da próxima instrução. Esta é um salto incondicional para a posição INÍCIO; a partir daí, o processo se repete, para o cálculo de outro y(n).

# APÊNDICE 3A

Programa em linguagem "BASIC" para cálculo de coeficientes de filtro digital de resposta ao impulso finita (RIF) utilizando a janela generalizada de Hamming. Filtro passa baixa , freq.de corte e freq.de amostragem em Hertz.

```
1 REM: CÁLCULO DE FILTRO DIGITAL RIF - JANELA GENERALIZADA    DE
       HAMMING.
5 INPUT "N, F. DE AMOST. (FA), F. DE CORTE (FC), ALFA"; N, FA,FC,
  ALFA
10 PI = 3.141592654
15 DIM W(N), HD(N), H(N)
20 K=(N-1)/2
25 FOR I= 0 TO K-1
30 W(I) = ALFA+((1-ALFA)*(COS((2*PI*I)/(N-1)))
35 WC = 2*PI*FC/FA
40 HD(I)=SIN(WC*(I-K))/(PI (I-ALFA))
45 H(I)=W(I)*HD(I)
50 NEXT I
55 HD(K)= 2*FC/FA
60 W(K)= 1
65 FOR I= 1 TO K
70 H(K+I) = H(K-I)
75 NEXT I
80 H(K)= HD(K)*W(K)
85 FOR I = 0 TO N-1
90 PRINT"W(";I;")=";W(I); TAB(15)"HD(";I;")=";HD(I);TAB(30)"
   H(";I;")=H(I)
95 NEXT I
100 END
```

```
                        0000            ORG     0000H
                        0001 ;
                        0002 ;
                        0003 ; PROGRAMA DE CALCULO DA TABELA DO FILTRO DIGITAL
                        0004 ;
                        0005 ;                      7 COEFICIENTES
                        0006 ;
                        0007 ;
                        0008 CONT       EQU     1000H           ; ENDERECO DO
                                                                  CONTADOR
                        0009 ;
                        0010 GUARDA     EQU     1001H           ; ENDERECO DO ACC.
                                                                  TEMPORARIO
                        0011 ;
                        0012 TBL        EQU     1400H           ; ENDERECO INICIAL
                                                                  DA TABELA
                        0013 ;
0000 21 00 10           0014 PROG       LD      HL,CONT
0003 36 00              0015            LD      (HL),0          ; (CONT)=00
0005 11 FF 13           0016            LD      DE,TBL-1
0008 06 00              0017 END3       LD      B,0             ; SOMA=00
000A 0E 07              0018            LD      C,7             ; C=NUMERO DE BITS
000C 13                 0019            INC     DE              ; DE=TBL
000D 21 94 00           0020            LD      HL,COEFIC       ; HL=END. DOS
                                                                  COEFICIENTES
                        0021 ;
0010 E5                 0022            PUSH    HL
0011 21 00 10           0023            LD      HL,CONT
0014 7E                 0024            LD      A,(HL)
0015 1F                 0025 DIVD       RRA                     ; (CONT)/2
0016 D2 23 00           0026            JP      NC,END1
0019 32 01 10           0027            LD      (GUARDA),A      ; CY=1,(GUARDA)=ACC
                                                                  TEMPORARIO
                        0028 ;
001C E1                 0029            POP     HL
001D 7E                 0030            LD      A,(HL)          ; ACC=(COEFIC)
001E 80                 0031            ADD     A,B             ; SOMA COM B
001F 47                 0032            LD      B,A             ; RESULTADO EM B
0020 C3 27 00           0033            JP      PULA
0023 E1                 0034 END1       POP     HL              ; CONSERTA A PILHA
0024 32 01 10           0035            LD      (GUARDA),A      ; ATUALIZA ACC.
                                                                  TEMPORARIO
                        0036 ;
0027 23                 0037 PULA       INC     HL              ; COEFIC.=COEFIC.+1
0028 0D                 0038            DEC     C
0029 CA 33 00           0039            JP      Z,END2
002C 3A 01 10           0040            LD      A,(GUARDA)      ; ACC=ACC/2
002F E5                 0041            PUSH    HL              ; GUARDE END.
                                                                  COEFIC. NA PILHA
                        0042 ;
0030 C3 15 00           0043            JP      DIVD            ; DIVIDE NOVAMENTE
0033 78                 0044 END2       LD      A,B
0034 12                 0045            LD      (DE),A          ; GUARDA NA TABELA
0035 CD 57 00           0046            CALL    NMOUT
0038 CD 46 00           0047            CALL    ESPACO
003B 21 00 10           0048            LD      HL,CONT
003E 34                 0049            INC     (HL)
003F 3E 80              0050            LD      A,80H           ; SOMOU TODOS
                                                                  OS COEFICIENTES?
                        0051 ;
0041 BE                 0052            CP      (HL)
0042 C2 08 00           0053            JP      NZ,END3
0045 FF                 0054            RST     38H
                        0055 ;
                        0056 ;
                        0057 ;
                        0058 ;
```

```
                        0059 ;
                        0060 ;    SUBROTINAS UTILIZADAS
                        0061 ;
                        0062 ;
0046 3E 20              0063 ESPACO  LD     A,20H
0048 CD 7C 00           0064         CALL   C0
004B CD 88 00           0065         CALL   C02
004E 3E 20              0066         LD     A,20H
0050 CD 7C 00           0067         CALL   C0
0053 CD 88 00           0068         CALL   C02
0056 C9                 0069         RET
                        0070 ;
                        0071 ;
0057 F5                 0072 NMBOUT  PUSH   AF
0058 0F                 0073         RRCA
0059 0F                 0074         RRCA
005A 0F                 0075         RRCA
005B 0F                 0076         RRCA
005C E6 0F              0077         AND    0FH
005E CD 74 00           0078         CALL   PRVAL
0061 CD 7C 00           0079         CALL   C0
0064 CD 88 00           0080         CALL   C02
0067 F1                 0081         POP    AF
0068 E6 0F              0082         AND    0FH
006A CD 74 00           0083         CALL   PRVAL
006D CD 7C 00           0084         CALL   C0
0070 CD 88 00           0085         CALL   C02
0073 C9                 0086         RET
                        0087 ;
                        0088 ;
0074 C6 30              0089 PRVAL   ADD    A,30H
0076 FE 3A              0090         CP     3AH
0078 D8                 0091         RET    C
0079 C6 07              0092         ADD    A,7
007B C9                 0093         RET
                        0094 ;
                        0095 ;
007C F5                 0096 C0      PUSH   AF
007D DB FB              0097 C01     IN     A,0FBH
007F E6 01              0098         AND    1
0081 CA 7D 00           0099         JP     Z,C01
0084 F1                 0100         POP    AF
0085 D3 FA              0101         OUT    0FAH,A
0087 C9                 0102         RET
                        0103 ;
                        0104 ;
0088 F5                 0105 C02     PUSH   AF
0089 DB FD              0106 C021    IN     A,0FDH
008B E6 01              0107         AND    1
008D CA 89 00           0108         JP     Z,C021
0090 F1                 0109         POP    AF
0091 D3 FC              0110         OUT    0FCH,A
0093 C9                 0111         RET
                        0112 ;
                        0113 ;
                        0114 ;
                        0115 ;
                        0116 ;
                        0117 ;
```

```
                    0118 ; VALOR DOS COEFICIENTES
                    0119 ;
                    0120 ;
0094 00             0121 COEFIC  DEFB  0                 ; VALOR DE A0
0095 04             0122         DEFB  4                 ; VALOR DE A1
0096 0A             0123         DEFB  0AH               ; VALOR DE A2
0097 0E             0124         DEFB  0EH               ; VALOR DE A3
0098 0A             0125         DEFB  0AH               ; VALOR DE A4
0099 04             0126         DEFB  4                 ; VALOR DE A5
009A 00             0127         DEFB  0                 ; VALOR DE A6
                    0128 ;
                    0129 ;
                    0130         END
ERR COUNT: 000

CO      007C    CO1     007D    CO2     0080    CO21    0099
COEFIC  0094    CONT    1000    DIVD    001A    END1    0023
END2    0033    END3    0008    ESPACO  0096    GUARDA  1001
NMOUT   0057    PROG    0000    PRVAL   007A    PULA    0027
TBL     1400
```

Obs:

Coeficientes para os filtros das tabelas 3.1. a 3.4. usados neste programa resultando na tabela 3.7.

| Coef. | Tab.3.1 | Tab.3.2 | Tab.3.3 | Tab.3.4 |
|---|---|---|---|---|
| $a_0 = a_6$ | 00 | 09 | 03 | FC |
| $a_1 = a_5$ | 04 | 05 | 05 | EE |
| $a_2 = a_4$ | 0A | 06 | E7 | 04 |
| $a_3$ | 0E | 06 | 24 | 10 |

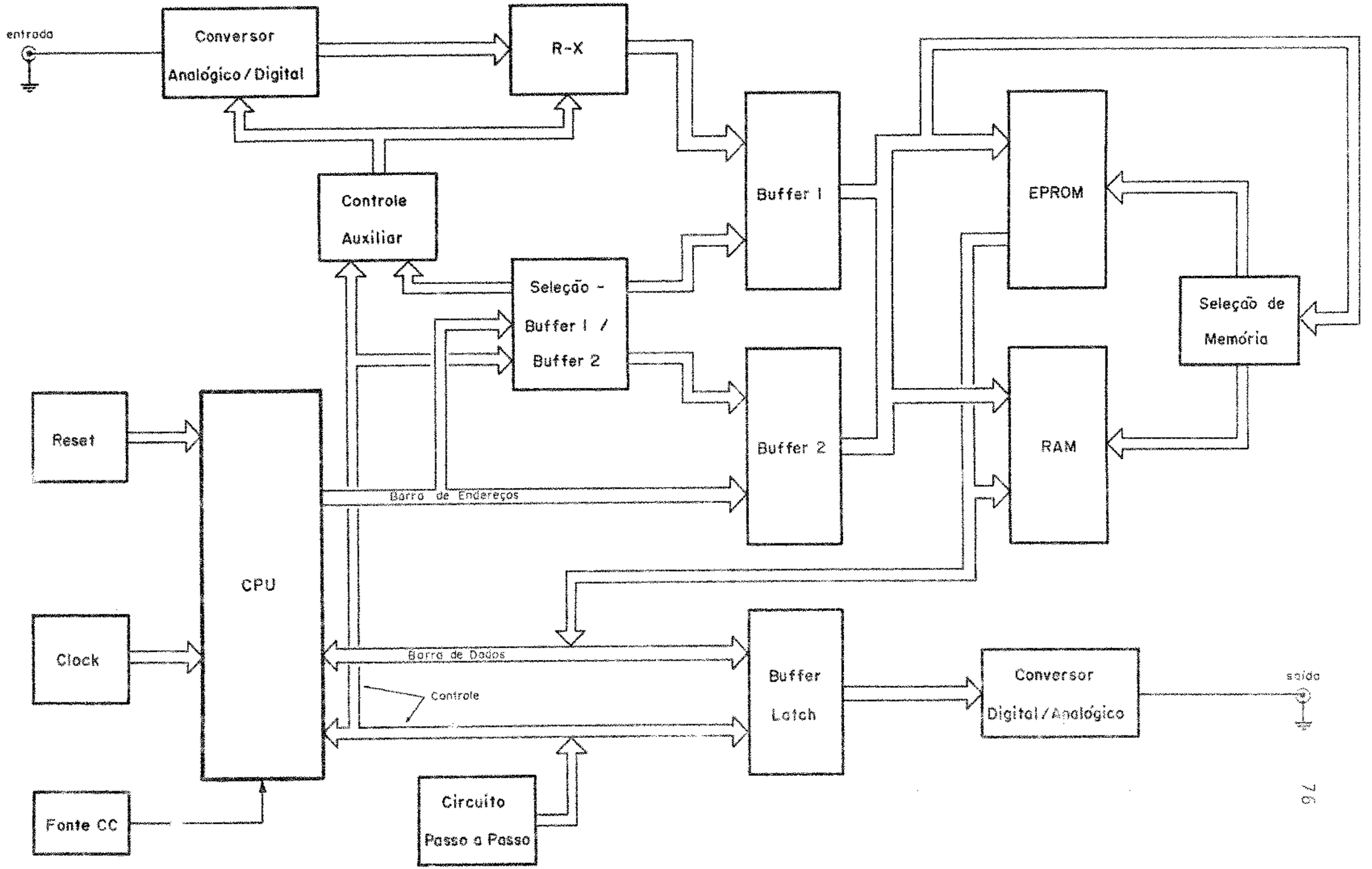
# CAPITULO IV: DESCRIÇÃO DO CIRCUITO E MEDIDAS REALIZADAS COM O FILTRO DIGITAL

## 4.1. INTRODUÇÃO

No capítulo anterior nos dedicamos à descrição dos processos de cálculo de coeficientes e da parte dos programas ("software") do filtro. Falta-nos uma descrição detalhada do projeto do circuito utilizado ("hardware"). Neste capítulo pretendemos nos dedicar a isto e também, apresentaremos algumas medidas realizadas. O diagrama de blocos do circuito está na página 76 . No final do capítulo, página 110 está o diagrama esquemático e na página 111 aparece a lista dos componentes utilizados. Na descrição seguinte revisaremos a função de cada parte do circuito dando ênfase ao aspecto de como esta função é conseguida pelos seus elementos.

## 4.2. CIRCUITO DE ENTRADA

Este bloco é constituído pelo amostrador - segurador ("sample-and-hold") e conversor A/D (CIs 1 e 2, fig.4.4.a). Sua função é converter uma amostra do sinal analógico de entrada do filtro em um sinal digital quantizado em 8 bits, após receber uma ordem de controle. Para realizar a conversão analógica - digital usamos o conversor AD570KD da firma Analog Devices[20]. Este circuito integrado de 18 pinos tem em seu interior um conversor A/D completo, incluindo circuito de referência e"clock", e realiza a conversão utilizando o método de aproximação sucessiva, num tempo típico de 25 µs. Sua precisão à temperatura ambien


entrada
Conversor
Analógico / Digital
R-X
Controle
Auxiliar
Seleção -
Buffer 1 /
Buffer 2
Buffer 1
Buffer 2
EPROM
Seleção de
Memória
RAM
Reset
CPU
Clock
Fonte CC
Barra de Endereços
Barra de Dados
Controle
Circuito
Passo a Passo
Buffer
Latch
Conversor
Digital / Analógico
saída

te (25ºC) é de ± 1/2 LSB (bit menos significativo) no máximo; permite entrada unipolar (0 a + 10V) ou bipolar (-5 a + 5V). Na figura 4.1 indicamos através de diagramas a forma de operação deste conversor. Assim, quando um pulso de 2 µs (mínimo) é aplicado na entrada $\overline{BC}$ ("blank and convert"), a sua borda de descida limpa a saída do conversor e dá início a uma nova conversão. Durante a conversão o sinal $\overline{DR}$ ("data ready") permanece em nível alto. Após 25 µs o sinal $\overline{DR}$ vai para baixo, indicando que o dado está presente na saída do conversor. Durante o intervalo de conversão, a saída do conversor permanece no estado de alta impedância, ou terceiro estado.

Escolhemos a operação bipolar do conversor A/D para possibilitar trabalharmos diretamente com sinais negativos na entrada. Assim, para uma tensão de entrada de -5V temos o código 00000000 na saída, e 11111111 correspondendo à entrada de +4.96V (5V-1LSB).

O conversor A/D deve operar em sincronismo com um circuito "sample and hold" (amostrador - segurador) para evitar que o sinal na sua entrada varie durante a realização de uma conversão.

Em nosso circuito utilizamos o integrado LF 398A, da National Semiconductor [23] que realiza a função de um circuito S/H controlado pelo A/D. Um diagrama interno deste circuito é mostrado na figura 4.2, incluindo alguns dados típicos de operação.

Os dois circuitos devem ser corretamente conectados para operação. O próprio conversor A/D, através do sinal $\overline{DR}$ indica ao S/H quando manter o sinal constante na entrada durante o intervalo de uma conversão. Na figura 4.3 indicamos o mo

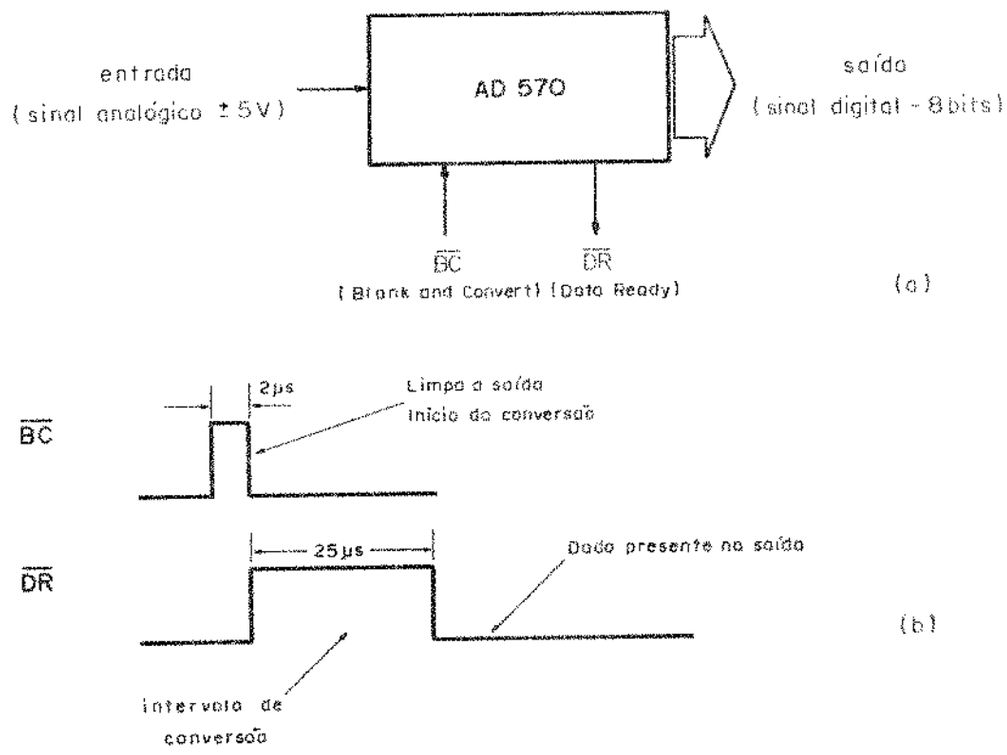
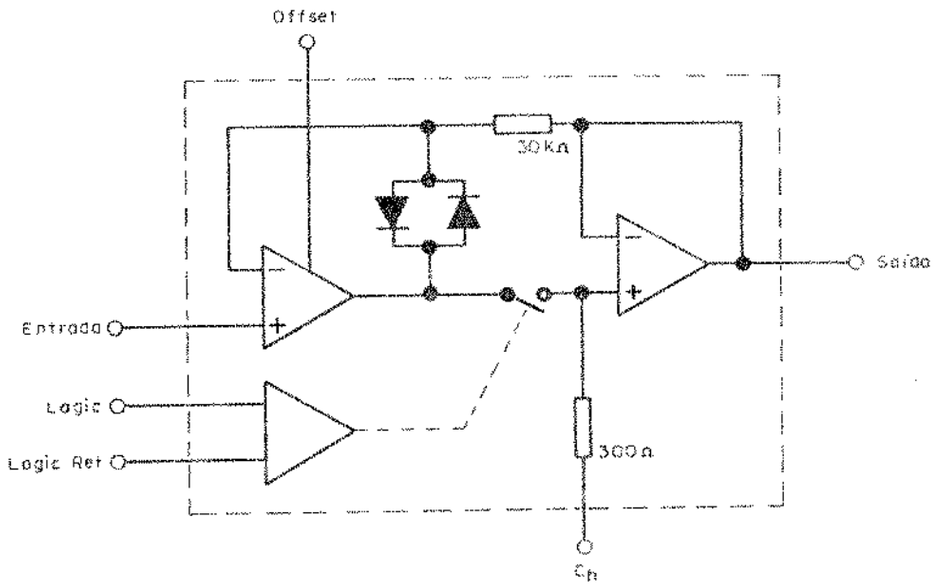



Fig. 4.1. Conversor A/D ; (a) diagrama de blocos ; (b) diagrama de tempo.




**Tempo de aquisição : 4 a 20μs (4μs típico)**

**Impedância de entrada : $10^{10}$ Ω**

**Impedância de saída : 0,5 Ω**

**Threshold diferencial lógico : 1,4V** (b)

Características do LF 398A -
(a) diagrama ; (b) valores típicos.

Fig. 4.2.

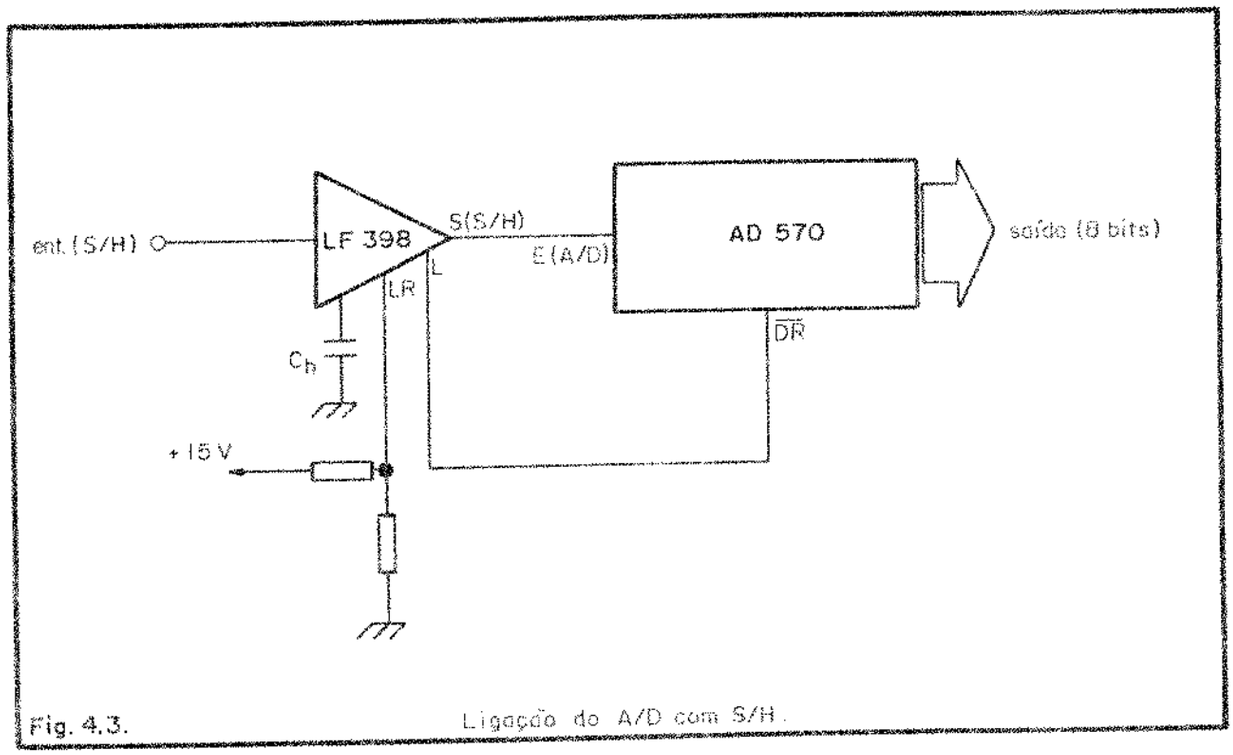



Fig. 4.3. Ligação do A/D com S/H.

do de conectar o A/D com o S/H. Quando $\overline{DR}$ está em nível alto (portanto, o A/D está convertendo um dado), o S/H mantém na sua saída a amostra do sinal constante. Esta amostra é a tensão no capacitor $C_h$ (h = hold).

No circuito de entrada devemos considerar o problema da forma de representação dos números no filtro. Como trabalharemos com números com sinal, na representação em "complemento de dois", há necessidade de adequar a forma do número binário na saída do A/D, tornando-a compatível com a representação utilizada. Note que isto pode ser facilmente conseguido utilizando um inversor lógico logo após a saída correspondente ao bit mais significativo do A/D. Assim, p. ex., a um sinal na entrada do A/D de -5V, corresponderá à $10000000_2$ na saída, 0V a $00000000_2$ e +4,96V a $01111111_2$.

Completando a descrição do circuito de entrada resta comentar a razão da escolha dos componentes; o AD570 foi escolhido principalmente pela sua funcionalidade e desempenho, apesar de seu custo ser bastante elevado. É um conversor rápido para a finalidade e possui boa precisão ($\pm$1/2 LSB) e estabilidade com a temperatura ($\pm$1LSB-22 ppm/ºC). O seu tempo de conversão (25µs) é bom para a finalidade desejada, pois, neste protótipo não tivemos intenção de usar frequências de amostragem superiores a 20kHz. Também há a vantagem de dispensar uma entrada de relógio externa, pois possui circuito interno de geração de relógio.

Quanto ao S/H, pesou principalmente o aspecto de custo e facilidade de encontrar o componente no comércio, uma vez que o circuito é de relativa simplicidade, podendo inclusive ser projetado com componentes discretos.

## 4.3. REGISTROS DE DESLOCAMENTO (RX)

Este circuito, ou bloco, que doravante designaremos abreviadamente por RX compõe-se de registros de deslocamento e "buffers". Na figura 4.4.b este bloco é constituído por CI-4, CI-5 a CI-16, CI-21 e CI-22.

O CI-4 é um registro de deslocamento de 8 bits,com entrada paralela e saída em série. É controlado por uma entrada indicada por S/L ("Shift-load"). Quando esta entrada está em nível 1 ocorre um deslocamento série na palavra armazenada no registro após aplicação de um pulso na entrada CK ("clock"). Quando S/L está em 0, um pulso em CK carrega uma nova palavra neste registro. Os CIs 5 a 16 são registros de deslocamento de 8 bits, com entrada em série e saída em série. Suas entradas CK("clock") são interligadas e produzem um deslocamento unitário ao recebe - rem um pulso de relógio. OS CIs 21 e 22 formam um "buffer" com terceiro estado, que serve para isolar este bloco da outra seção do circuito. O funcionamento é o seguinte: suponhamos que o registro de entrada, CI-4, está carregado com uma palavra de 8 bits. Idêntica situação ocorre com os CIs 5 a 16. No primeiro registro está a palavra correspondente à enésima amostra do sinal de entrada; no segundo, à enésima menos um, e assim por diante. As saídas deste conjunto de registros, estão posicionados os bits menos significativos de cada uma destas palavras.

Note que as saídas de RX estão conectadas através dos "buffers" com terceiro estado (B1) ao barramento de endereços da memória. Elas irão servir, no momento apropriado, como um vetor de endereços apontando para posições de uma tabela previamente gravada na memória do sistema. O passo seguinte será deslo

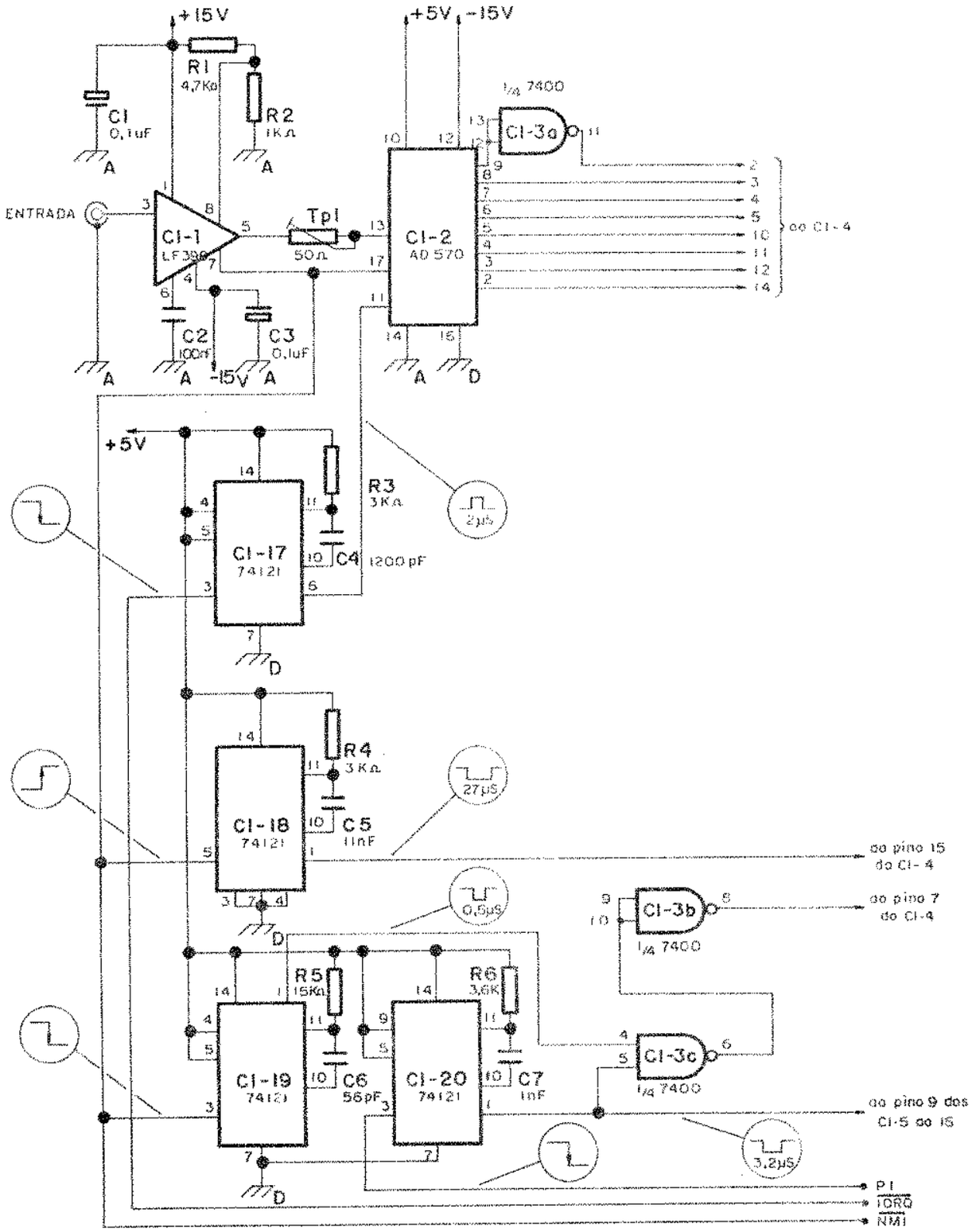



Fig. 4.4. a — Circuito de entrada e controle auxiliar.

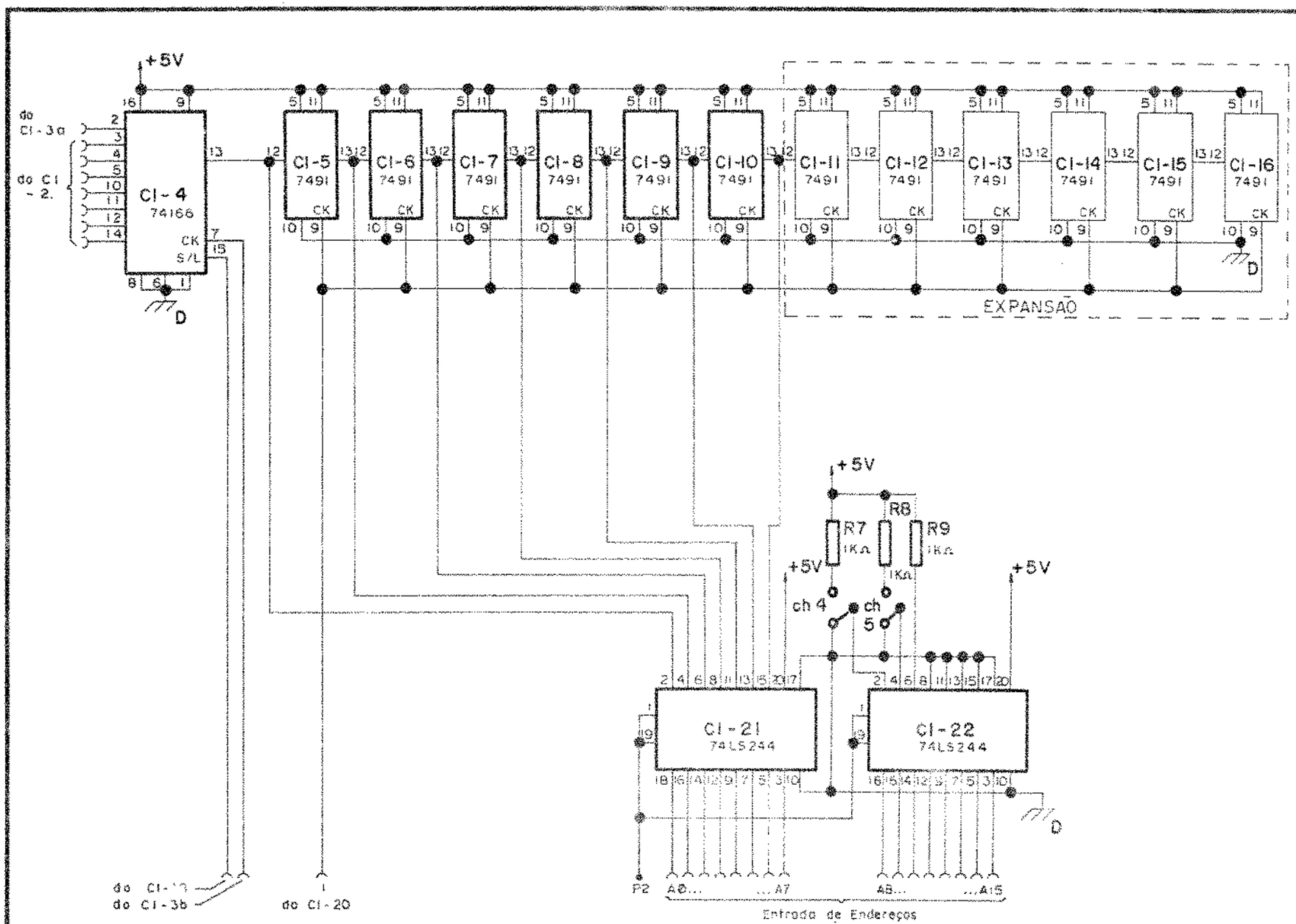



Fig. 4.4. b

Circuito "RX" e "buffer".

car de 1 bit todos os registradores. Assim, novo "vetor" de endereçamento estará nas saídas de RX e assim sucessivamente. Após 7 pulsos de deslocamento, aplicados a todos os registros simultaneamente, o bit mais significativo de cada palavra (correspondente ao sinal) estará nas saídas de RX. Após o oitavo pulso a enésima palavra passou para o CI-5, a enésima menos um para o CI-6, etc. Então, nova palavra será armazenada no CI-4 utilizando o comando de carregamento (S/L=0). Todas estas operações de deslocamento devem ser sincronizadas corretamente. Isto irá ficando mais claro à medida que formos descrevendo as outras partes do circuito. Os CIs 21 e 22 devem ter o estado de alta impedância para isolar o acesso ao registro RX quando o endereçamento normal, através da CPU, estiver sendo utilizado (pinos 1,19 são entradas de controle do 3º estado para CIs 21 e 22).

A finalidade deste bloco é produzir um endereçamento da memória em função dos bits das palavras de entrada. O número de registradores, e, portanto, de saídas de RX é uma função do número de coeficientes do filtro, e é limitado pela capacidade de memória do sistema. O número de coeficientes usado nas medidas foi 7, sendo para isto, necessários no circuito os CIs 5 a 10. Os restantes ficam reservados para uma provável expansão com um número maior de coeficientes.

As entradas dos CIs 22 e 23, através das ligações indicadas, e das chaves ch-4 e ch-5, permitem o endereçamento de 4 tabelas distintas na EPROM, a partir dos endereços 400H, 500H, 600H e 700H. Nesta situação podemos escolher 4 filtros (de 7 coeficientes) e selecioná-los pela simples mudança na posição das chaves.

## 4.4. CPU

A CPU escolhida para este filtro é o microprocesador Z-80A, da Zilog [3], representada pelo CI-25 da figura 4.5. Esta escolha foi baseada em algumas vantagens sobre outros similares, como por exemplo a existência de instruções como SRA (que não existe no 8080) e sua simplicidade de ligações(dispensa "interfaces", "clock" único,etc.),mas o circuito pode ser adaptado facilmente a outros microprocessadores como o 8085 da Intel. A função da CPU é o controle das operações para cálculo das amostras de saída do filtro.

## 4.5. MEMÓRIA

A memória do sistema está na figura 4.6. O CI-33 é uma EPROM 2716[22], de 2 k bytes (0000H a 03FFH). Os CIs 34 e 35 constituem a memória RAM do sistema com 1k byte (400H a 7FFH). A seleção da EPROM ou RAM é feita através do CI-36, um decodificador de endereços ligado ao barramento de endereços em A11 e A12. A memória RAM 2114 [24] é usada apenas como pilha operacional para operações com o "stack-pointer" (SP). A memória EPROM contém a partir de 0000H até 0022H o programa de operação do filtro. O espaço que sobra (mais de 1 k byte) é usado para guardar os números da tabela formada a partir dos coeficientes.A tabela da verdade para o CI-36 é apresentada na figura 4.7.

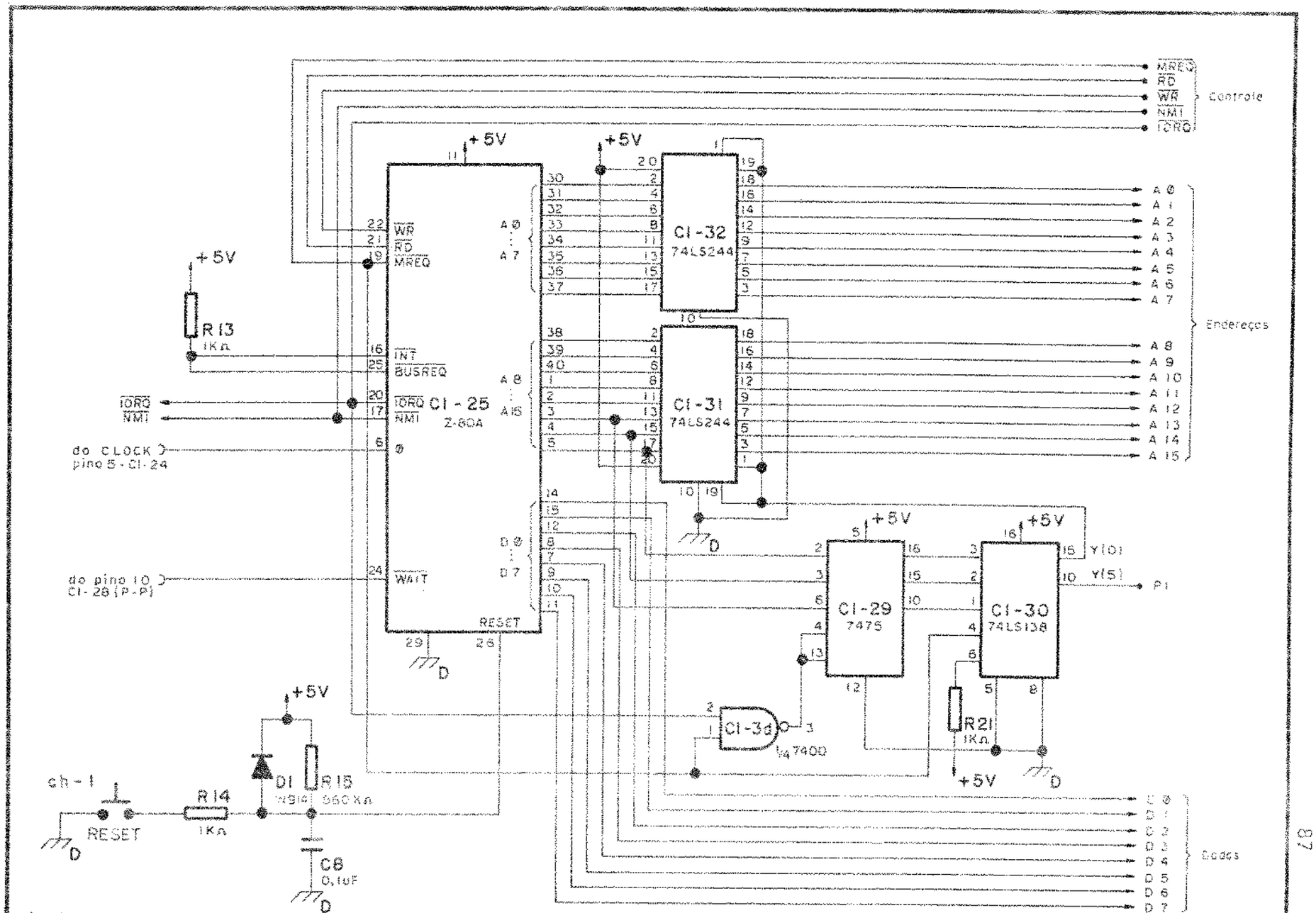



Fig. 4.5. C.P.U. - Circuitos de seleção (B 1/B 2),"buffer"2 e"reset"

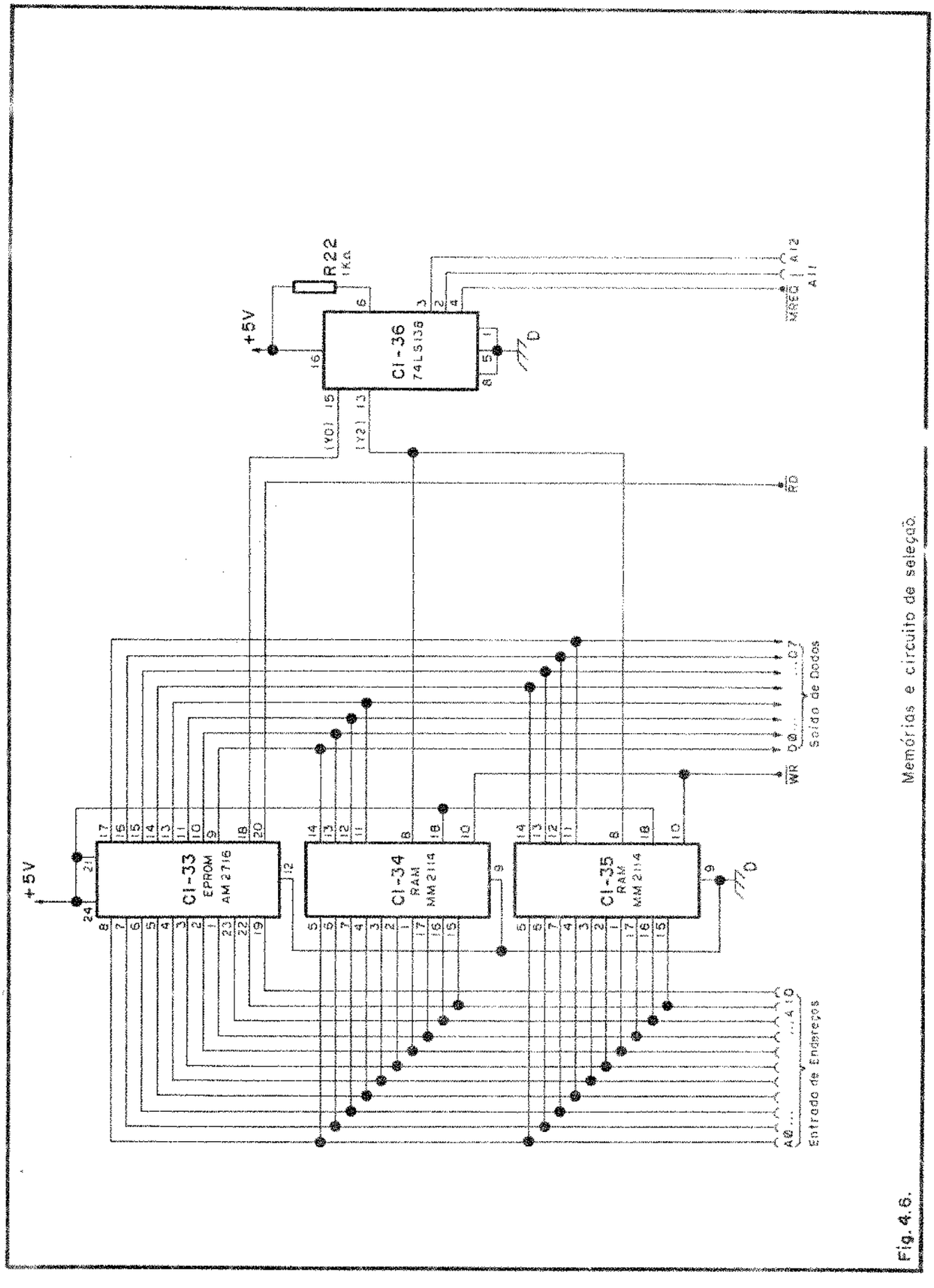



Fig. 4.6. Memórias e circuito de seleção.

| Entradas | | | | | Saídas | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Habilitação | | Seleção | | | | | | | | | | |
| G1 | *G2 | C | B | A | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
| X | H | X | X | X | H | H | H | H | H | H | H | H |
| L | X | X | X | X | H | H | H | H | H | H | H | H |
| H | L | L | L | L | L | H | H | H | H | H | H | H |
| H | L | L | L | H | H | L | H | H | H | H | H | H |
| H | L | L | H | L | H | H | L | H | H | H | H | H |
| H | L | L | H | H | H | H | H | L | H | H | H | H |
| H | L | H | L | L | H | H | H | H | L | H | H | H |
| H | L | H | L | H | H | H | H | H | H | L | H | H |
| H | L | H | H | L | H | H | H | H | H | H | L | H |
| H | L | H | H | H | H | H | H | H | H | H | H | L |

*G2 = G2A + G2B
H = Nível alto
L = Nível baixo
X = Não importa

Fig. 4.7. Tabela da verdade para o 74LS138.

## 4.6. CIRCUITO DE SAÍDA

O circuito de saída, figura 4.8 é composto por um "buffer-latch" (CI-37), um inversor lógico (CI-28d), pelo conversor D/A (CI-38). Uma vez calculada uma saída y(n), ela irá para o bloco de saída onde será convertida em um nível analógico e mandada para a saída do filtro. O conversor utilizado é o AD 559 da Analog Devices [20], cujas características principais são:

1) Conversor D/A de 8 bits, num único integrado monolítico de 16 pinos.
2) Precisão (erro relativo a plena escola): $\pm$ 1/2 LSB (0,19% da FS).
3) Alimentação: $V_{CC}$= +4,5V dc a + 5,5V dc - 8mA máx.
   $V_{EE}$= -11,5 Vdc a -16,5Vdc - 16 mA máx.
4) Faixa de Temperatura: 0 a 75ºC
5) Corrente de saída (todos os bits ON, $I_{REF}$= 2mA): 1,99mA$\pm$2%

O CI-39 (741), D2 (diodo zener, $V_Z$=4V) e T1 (Fet MPF 102) tem a função de gerar uma tensão de referência aplicada ao pino 14 do conversor D/A. O circuito de referência sugerido pelo fabricante usava o integrado AD580, uma fonte de referência de 2,5V. Entretanto julgamos a solução adotada mais econômica. Apenas a tensão de referência ($V_{REF}$) ficou maior, fato que nos levou a alterar um pouco os valores dos resistores do circuito em relação à sugestão do fabricante.

Na saída temos o CI-40, o amplificador operacional CA 3140, cuja função é a de um conversor corrente - tensão, uma vez que a saída do conversor D/A é em corrente.

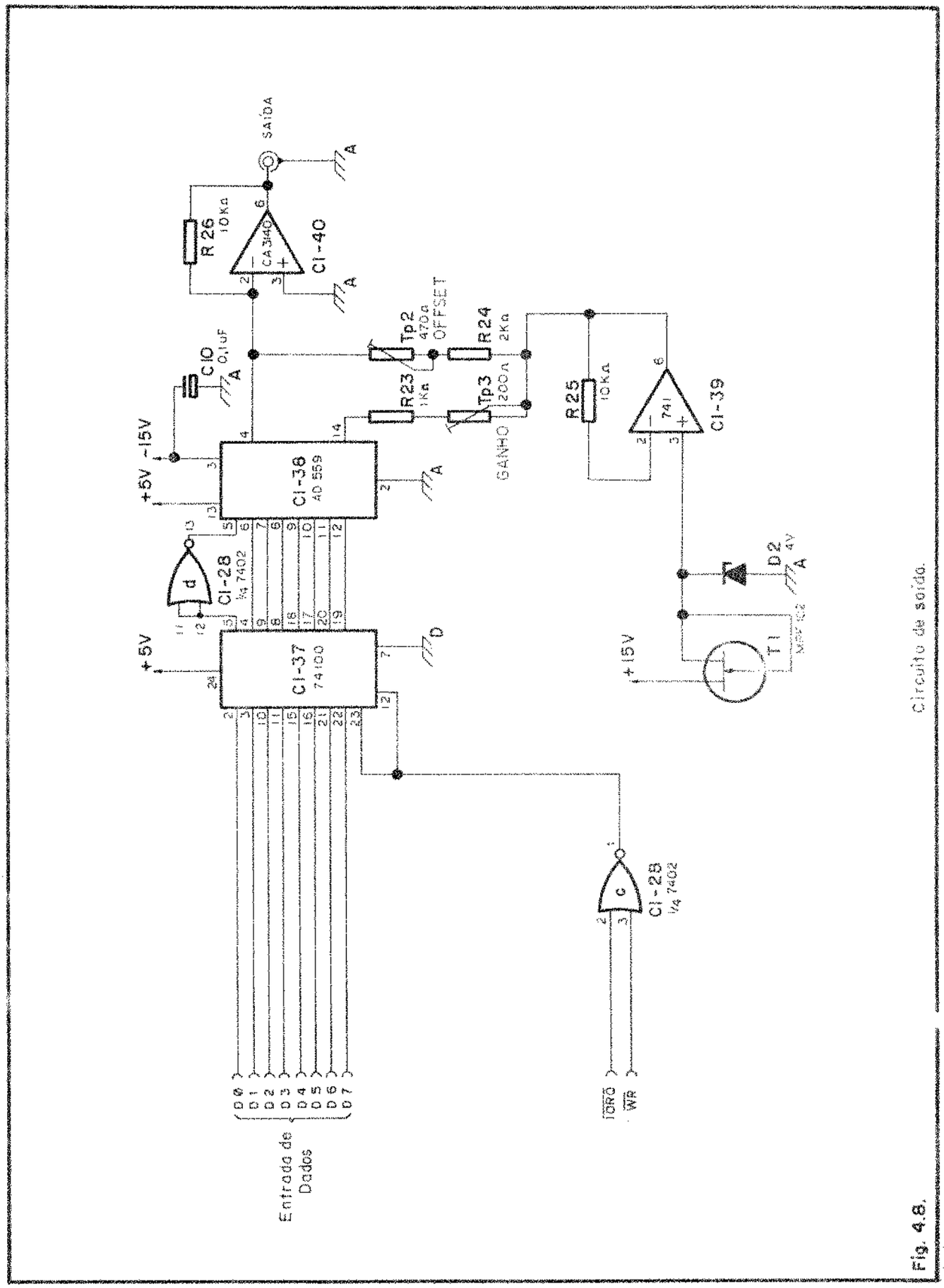



Fig. 4.8. Circuito de saída.

Um problema de ajuste prático aparece nesta parte do circuito: a calibração do D/A. Com a operação bipolar do dispositivo escolhida para a saída, o procedimento para calibração é o seguinte:

1) Com todos os bits OFF (nível baixo) ajuste o potenciômetro de "offsett" bipolar para $E_{saída}$ = 10,000 V
2) Com o bit 1 (MSB) ON (nível alto) e todos os outros bits OFF, ajuste o potenciômetro de ganho para $E_{saída}$ = 0,000V.
3) Com todos os bits ON, verifique que $E_{saída}$ = +10V-1LSB = +9,912V (1LSB = 20V/256=78,2mV).

Este conversor também pode ser substituído pelo conversor D/A 1408 ou 1508 da Motorola, pois é totalmente compatível, inclusive na pinagem.

## 4.7. CIRCUITO PARA GERAÇÃO DO RELÓGIO

O circuito para geração de relógio é necessário para o funcionamento da CPU do filtro. Em nosso caso foi usado um circuito convencional, que utiliza um cristal com freqüência de 3,574 MHz, associado a inversores lógicos, como mostra a figura 4.9. O sinal assim gerado passa por um inversor que serve de isolamento entre estágios. Em seguida, sofre uma divisão por 2(1),

---

(1) Esta divisão pode ser eliminada, retirando o CI-24, e ligando diretamente a saída do inversor à entrada de "clock" da CPU.

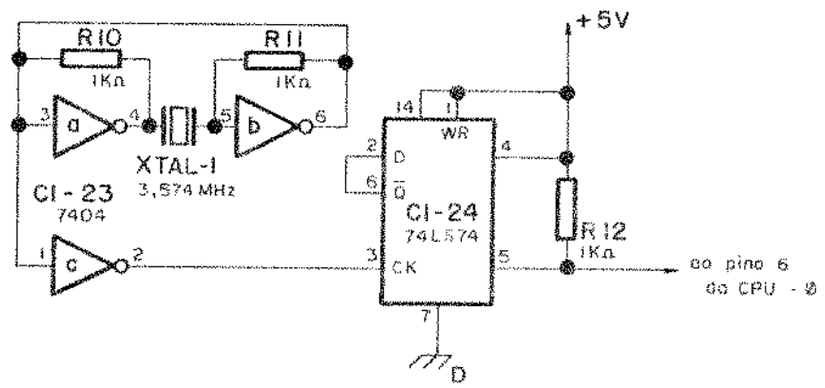



Fig. 4.9. Geração de relógio (clock)

através de um "flip-flop" do tipo D (CI-24) que resulta num sinal com freqüência de 1,787 MHz, dentro dos limites do Z-80A (máximo f clock= 4MHz). Em nossos ensaios fizemos uso tanto das freqüências de 3,574 MHz e 1,787MHz, o que resulta nos períodos T = 279,8 ns e T = 559,6 ns, respectivamente.

## 4.8. CIRCUITOS DE SELEÇÃO E CONTROLE

A operação normal do circuito exige uma série de sinais de seleção e controle dos diversos blocos, operando na seqüência exata. Na descrição destes circuitos de seleção e de controle optamos por dividí-los de acordo com a sua função específica. O correto dimensionamento destes circuitos é um dos pontos fundamentais do projeto.

### 4.8.1. Seleção e Controle de B1 e B2

Esta operação ocorre durante todo o programa de operação de filtro. B1 deve ser selecionado quando se deseja apontar para um endereço da tabela na EPROM. B2 neste caso é inibido. B2 é selecionado quando a CPU está em busca de uma instrução, ou para a saída de dados. B1 é realizado na figura 4.4.b por CI-21 e CI-22, e B2 na figura 4.5 por CI-31 e CI-32. Todos estes "buffers" possuem um estado de alta impedância, controlados pelas entradas 1,19 (ativo nível baixo).

Também na figura 4.5, o CI-29 é um "buffer latch" (tabela da verdade da figura 4.7) que mantém a saída estável para o CI-30, um decodificador de endereços. O endereço A000H presente no barramento de endereços da CPU, através das linhas de ende

reço A13, A14 e A15 da CPU seleciona o B1(Y5=0) e coloca B2 em alta impedância. Qualquer outro endereço que aparece no barramento mantém B1 em alta impedância e B2 ativo (Y0=0).

### 4.8.2. Deslocamento Unitário em RX

A saída Y5(figura 4.5) sofre uma queda toda vez que o endereço A000H aparece no barramento da CPU. Este sinal é usado para provocar um deslocamento unitário em RX. Na figura 4.4.a este sinal é aplicado no CI-20, um circuito monoestável (ponto P1). À saída do monoestável é gerado um pulso que irá provocar um deslocamento unitário em RX (CIs 5 a 16, figura 4.4.b). A duração deste pulso deve ser corretamente calculada, de acordo com a seqüência das operações necessárias. O deslocamento em RX não deve ocorrer durante a leitura de um dado na tabela, isto é,quando estão sendo executadas instruções LD A,(HL), ADD (HL) ou SUB (HL). Para dar uma idéia deste cálculo, mostramos na figura 4.10 um diagrama onde está indicada a seqüência das instruções que o microprocessador executa, bem como o número de ciclos T necessários para cada instrução.

Observando o diagrama, na situação mais crítica , podemos calcular o tempo de atraso para o deslocamento unitário em RX (em relação ao acionamento do monoestável CI-19 por Y5),como

$$7T < T_A < 11T \qquad (4.1)$$

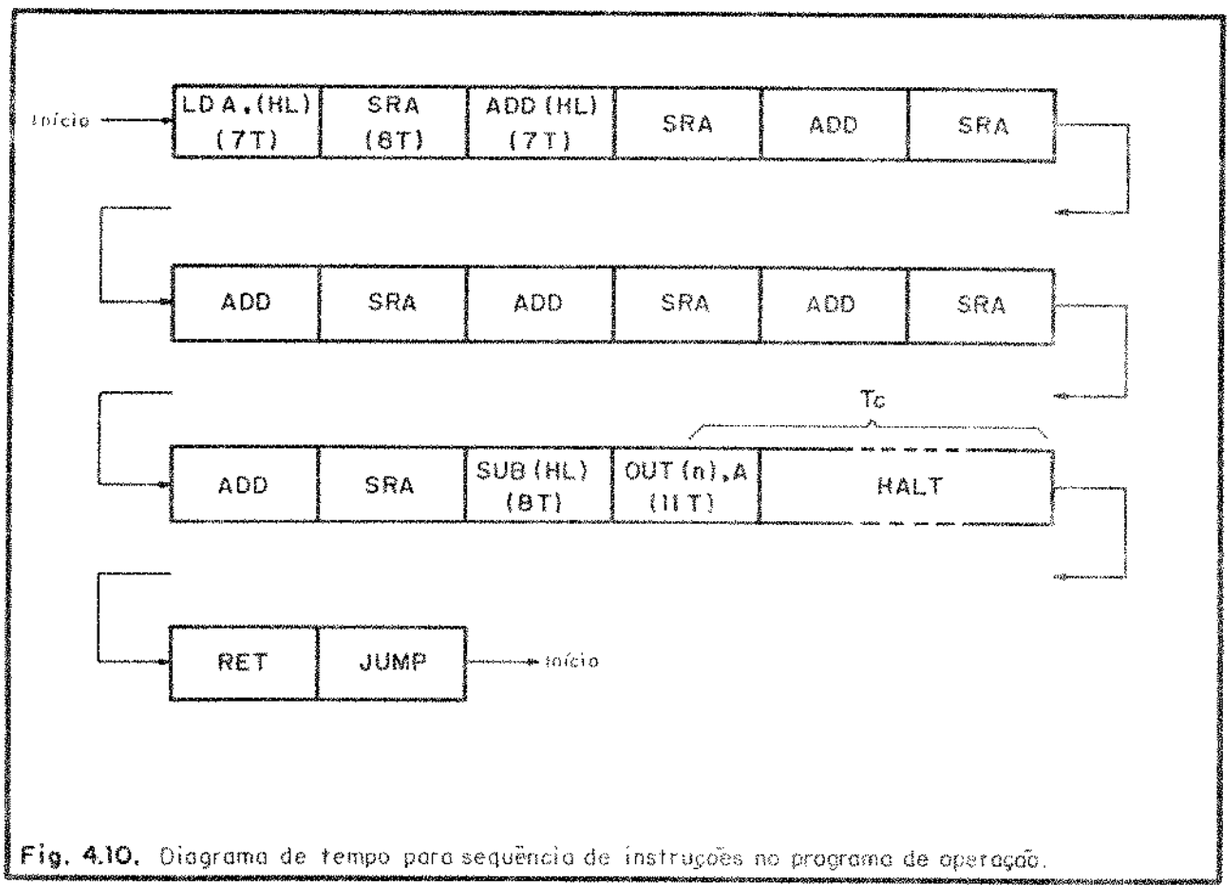



**Fig. 4.10.** Diagrama de tempo para sequência de instruções no programa de operação.

### 4.8.3. Sinal para Saída de um Dado

Durante a execução da instrução OUT, os sinais $\overline{IORQ}$ e $\overline{WR}$ vão para o nível baixo, indicando que um dado está presente na linha de dados do microprocessador. Estes sinais são usados, conforme indica a figura 4.8 para controlar o CI-28, um "buffer-latch". Os dados são transferidos para as saídas deste circuito e ficam disponíveis para o conversor D/A. Durante a execução de outras instruções, o CI-28 isola o circuito de saída do microprocessador.

### 4.8.4. Pulso para Provocar uma Nova Amostragem no A/D

Este pulso irá aparecer logo após a execução de uma instrução de saída de dado (OUT). Durante a execução desta instrução o sinal $\overline{IORQ}$ do Z-80 vai para o nível baixo. Esta descida é aplicada no monoestável CI-17, figura 4.4.a que fornece na saída um pulso de 2 μs. Este pulso é aplicado na entrada $\overline{BC}$ do A/D, iniciando uma nova conversão de dado.

### 4.8.5. Pulso para Retirar o Microprocessador do Estado de Halt

Para esta operação, lembramos que o microprocessador deve sair do estado de HALT quando a conversão de um novo dado pelo A/D foi completada. Usamos então, o sinal $\overline{DR}$ do A/D, ligando-o diretamente à entrada $\overline{NMI}$ da CPU, conforme indicado na figura 4.4.a.

### 4.8.6. Pulso para Carregar Novo Valor na Entrada de RX

Após o A/D apresentar em sua saída um novo dado , este novo valor deve ser transferido para RX. A execução desta operação exige duas condições: a entrada S/L do CI-4,figura 4.4.b., deve permanecer em nível lógico 0; um pulso de relógio deve ser aplicado na entrada CK. Assim, o monoestável CI-18, ligado ao controle $\overline{DR}$, após o início da conversão ($\overline{DR}$ vai para 1), mantém sua saída no nível baixo durante 27 µs. Após a conversão ($\overline{DR}$ vai para 0), outro monoestável, o CI-19, gera um pulso de 0,5 µs na entrada CK do CI-4.

## 4.9. CIRCUITO PARA INICIALIZAÇÃO DAS OPERAÇÕES

O objetivo deste circuito é inicializar o sistema. Na figura 4.5 a chave ch-1 e os componentes associados atuam diretamente sobre a entrada $\overline{RESET}$ da CPU (ativo nível baixo), fazendo com que o microprocessador inicie a operação na posição 0000H. A função do diodo e capacitores é evitar problemas de ruído gerados pela chave.

## 4.10. OPERAÇÃO PASSO A PASSO

O circuito do filtro está projetado para realizar as operações passo a passo, isto é, a cada ciclo de máquina do microprocessador. Este tipo de recurso é muito útil nos primeiros testes de funcionamento do circuito, ou, posteriormente,quando se deseja verificar o funcionamento de algum bloco isoladamente. A idéia básica deste controle consiste na atuação sobre o

sinal $\overline{\text{WAIT}}$ do microprocessador. A figura 4.11 mostra como o ciclo de busca de uma instrução ("fetch cycle") se atrasa quando a linha de "WAIT" estiver ativa. Durante $T_2$ e cada subseqüente $T_W$, a CPU pesquisa a linha de "WAIT" na queda do sinal de "clock" e se ela estiver ativa um estado de "WAIT" é preparado para o próximo ciclo. Desta forma nenhuma instrução é executada.

O circuito para operação passo a passo é apresentado na figura 4.12. Nesta figura o CI-26b (um"flip-flop" D) faz a seleção entre a operação normal e a passo a passo do circuito. A saída do CI-26b mantém a linha de $\overline{\text{WAIT}}$ da CPU em nível alto garantindo a operação normal. Caso contrário, a linha de $\overline{\text{WAIT}}$ estará sempre ativa, sendo modificada quando acionamos a chave ch-2 (chave de contato momentâneo). Este acionamento provoca um pulso descendente na saída do CI-26a (um "flip-flop" D),que, por sua vez, gera um pulso na saída do CI-27 (um monoestável). Este pulso aplicado na linha de $\overline{\text{WAIT}}$ terá duração suficiente para permitir que a CPU saia do estado de WAIT e execute um ciclo de máquina, mas retorna a zero antes da execução do próximo ciclo. Assim, para execução de cada ciclo de máquina o microprocessador terá de aguardar que a chave ch-2 seja acionada. Como cada instrução do programa possui vários ciclos de máquina, podemos verificar passo a passo as ocorrências no circuito.

## 4.11. SEQUÊNCIA DAS OPERAÇÕES

Para completar a descrição do circuito apresentamos na figura 4.13 alguns sinais de seleção e controle fotografados durante a operação do filtro. Na figura 4.13.a, de cima para baixo, aparecem os seguintes sinais:

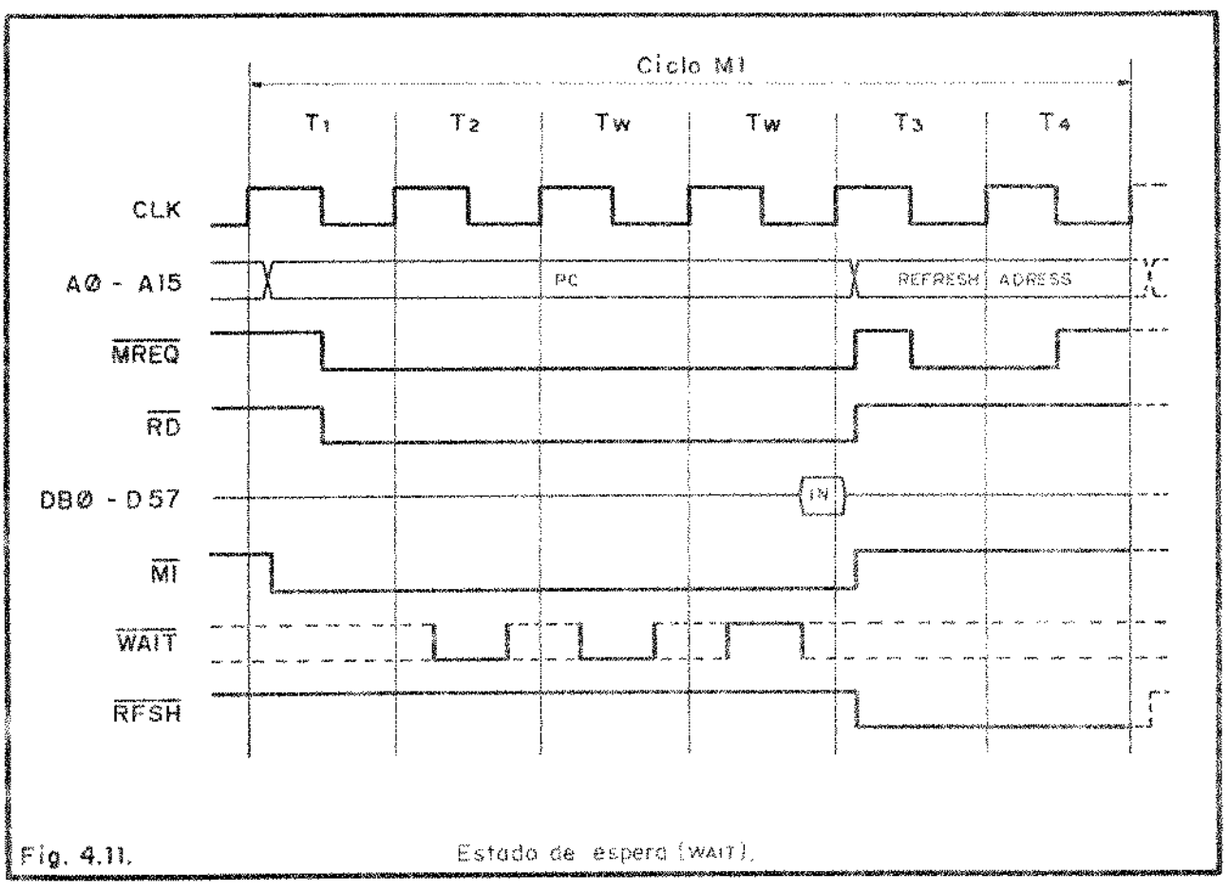



Fig. 4.11. Estado de espera (WAIT).

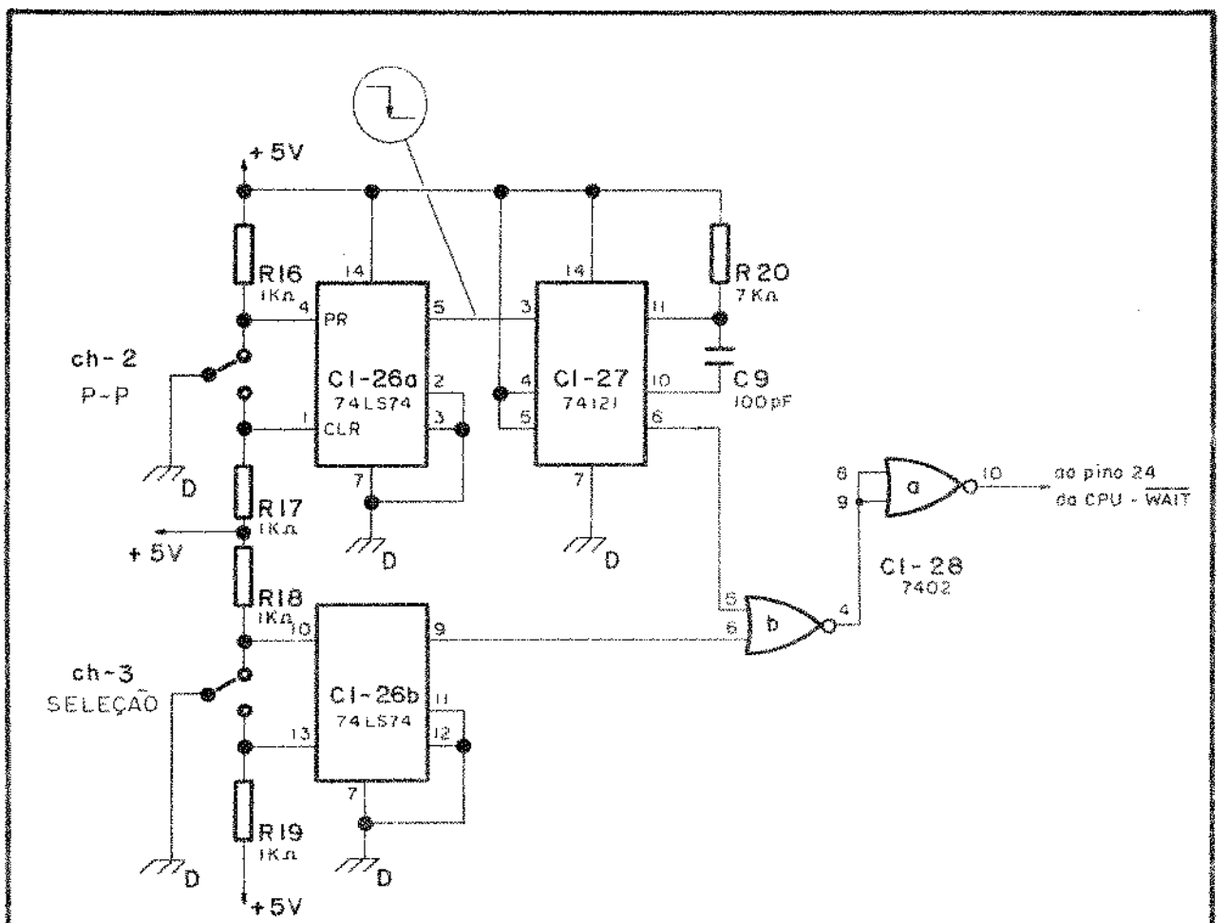
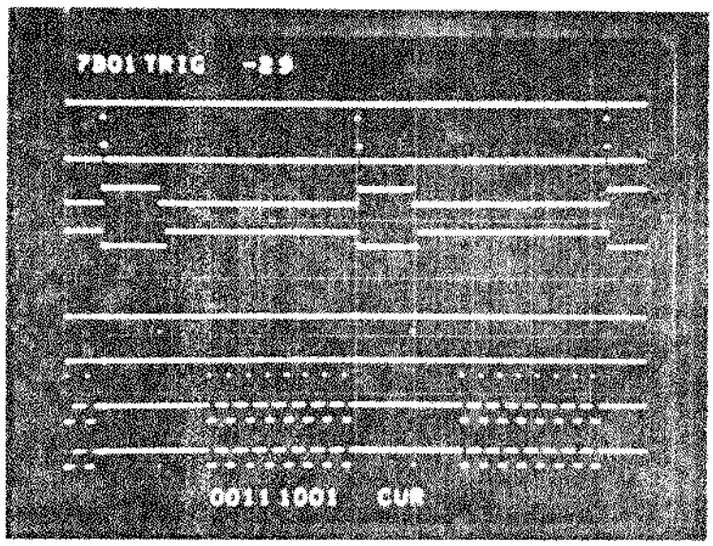
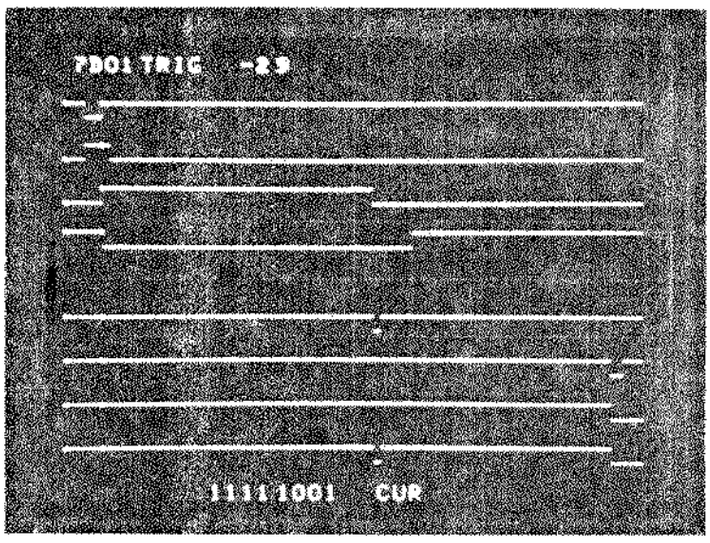



Fig. 4.12. Circuito para operação passo a passo.




(a) Sinais




(b) Ampliação

Fig. 4.13. Sinais de controle do filtro.

1º) $\overline{IORQ}$, da CPU (CI-25, fig. 4.5);

2º) $\overline{BC}$, do A/D (CI-2, pino 11, fig. 4.4.a);

3º) $\overline{DR}$ , do A/D (CI-2, pino 17, fig. 4.4.a);

4º) S/L, do RX (CI-4, pino 15, fig. 4.4.b);

5º) carregamento de RX (pino 1, CI-19, fig. 4.4.a);

6º) seleção de B1 (ponto P2, fig. 4.4.b);

7º) deslocamento em RX (pino 1, CI-20, fig. 4.4.a);

8º) CK do CI-4 (pino 7, CI-4, fig. 4.4.b).

A figura 4.13.b mostra os mesmos sinais da figura 4.13.a, mas ampliando os detalhes desta última.

Na figura 4.13 podemos observar que, após a descida do sinal $\overline{IORQ}$ (primeiro sinal), um pulso é gerado na entrada $\overline{BC}$ de A/D (segundo sinal). Na parte final deste pulso o A/D inicia a conversão, o que é indicado pela subida do sinal $\overline{DR}$ (terceiro sinal). Esta subida é usada para provocar um nível baixo na saída do CI-18, figura 4.4.a (quarto sinal).Este sinal, aplicado à entrada S/L (pino 15, CI-4), permanece em zero até que um pulso para carregar o novo valor no A/D seja gerado. Este pulso (quinto sinal) é comandado pela descida do sinal $\overline{DR}$, através do CI-19,figura 4.4.a. O "buffer" B1 é selecionado 8 vezes seguidas, em cada ciclo de cálculo (sexto sinal). O sinal que seleciona B1 é atrasado e usado para provocar um deslocamento unitário em RX. Isto é conseguido com o CI-20, figura 4.4.a, um monoestável, cuja saída (sétimo sinal) gera os pulsos para deslocamento de RX. Na entrada CK do CI-4 temos os pulsos para deslocamento e carregamento deste registro (oitavo sinal).

## 4.12. TEMPO DE PROCESSAMENTO

O tempo necessário para o cálculo de uma saída y(n) depende do tempo de execução das instruções e da conversão de dado na entrada. A tabela da figura 4.14 nos indica o tempo das instruções.

Com os valores da tabela e com p = 1, B = 8, na equação (2.10), podemos estimar o tempo de processamento de uma saída y(n):

$$T_{proc} = 7T + 6x7T + 7T + 7x8T + 11T + 10T = 133T \qquad (4.2)$$

Devemos acrescentar a este resultado o valor de 10T correspondente à instrução RET e o tempo em que o microprocessador fica no estado de "HALT" que depende do tempo de conversão e carregamento de um novo valor em RX, que estimaremos em 27 μs aproximadamente. Assim, temos:

$$T_{proc} \cong 143T + 27\ \mu s \qquad (4.3)$$

Usando a equação (4.3) e os valores de T usados no circuito de acordo com a seção 4.7, podemos calcular o tempo de processamento, a freqüência de amostragem, e comparar com os valores medidos no circuito. Esta comparação aparece na tabela da figura 4.15.

## 4.13. MEDIDAS EFETUADAS

Os filtros cujos coeficientes foram calculados nas tabelas 3.1, 3.2, 3.3 e 3.4 foram ensaiados e a comparação entre os valores teóricos (resultado da simulação) e práticos (medidas realizadas) aparecem nas figuras 4.16, 4.17,4.18 e 4.19,res

| | Mnemônico Z-80 | Nº ciclos T | Nº de Bytes |
|---|---|---|---|
| 1 | LD A,(HL) | 7 | 1 |
| 2 | ADD A,(HL) | 7 | 1 |
| 3 | SUB (HL) | 7 | 1 |
| 4 | SRA A | 8 | 2 |
| 5 | OUT (n),A | 11 | 2 |
| 6 | HALT | * | 1 |
| 7 | RET | 10 | 1 |
| 8 | JP | 10 | 3 |

* depende da conversão na entrada.

Fig. 4.14. Tabela para indicação do tempo - e número de bytes para as instruções do programa de operação.

| f (MHz) | T (ns) | T proc.(μs) | $f_o$ calculada (KHz) | $f_o$ medida (KHz) |
|---|---|---|---|---|
| 3,575 | 279,7 | 67,0 | 14,9 | 14,7 |
| 1,788 | 559,4 | 107,0 | 9,4 | 9,0 |

Fig. 4.15. Tempo de processamento e freqüência de amostragem do circuito.

pectivamente. Nas abcissas desses gráficos temos os valores das freqüências digitais normalizadas.

## 4.14. COMENTÁRIOS SOBRE AS MEDIDAS

Conforme se pode observar nas figuras 4.16 a 4.19, há uma maior aproximação entre os valores teóricos e práticos na região de freqüências mais baixas, e um maior afastamento na região de altas freqüências. Isto é esperado, principalmente no caso do filtro passa-alta. Ocorre que, para as freqüências mais altas, é menor o número de amostras do sinal em cada ciclo, produzindo uma forma de onda na saída mais distorcida; pode-se dizer que a relação entre o sinal e o ruído de quantização diminui consideravelmente para freqüências próximas a $f_a/2$. Também ocorre uma maior diferença para valores de baixa amplitude, onde também é baixa a relação sinal-ruído.

Outro problema verificado é o "overflow". Quando os coeficientes do filtro são muito altos, os valores da tabela na EPROM também o são. Estes valores aparecem somados no processo de cálculo de y(n), podendo ultrapassar os limites máximos permissíveis da representação numérica no acumulador. Isto produzirá uma forma de onda na saída distorcida. Para evitar este problema, pode-se dividir todos os coeficientes do filtro por uma constante, conforme foi feito para os filtros das figuras 4.17, 4.18 e 4.19. Nestes casos, todos os coeficientes foram divididos por 2, resultando num deslocamento para baixo uniforme da resposta em freqüência.

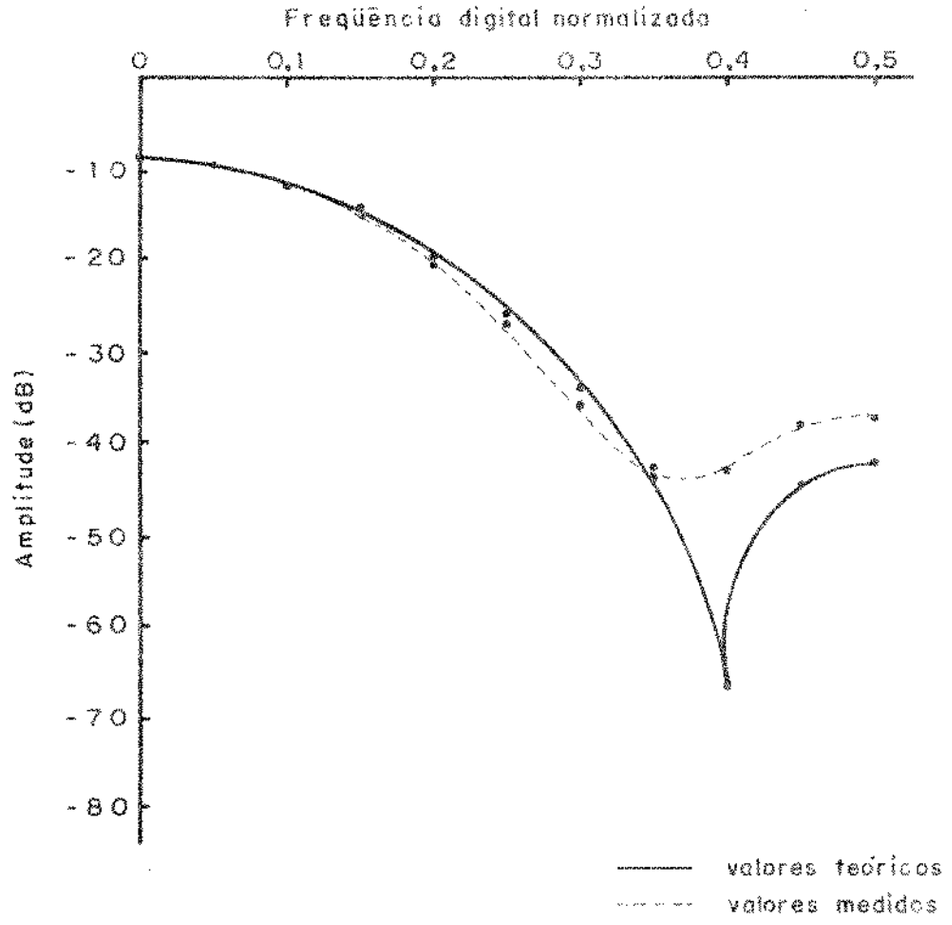



Fig. 4.16. Filtro passa baixa da tabela 3.1

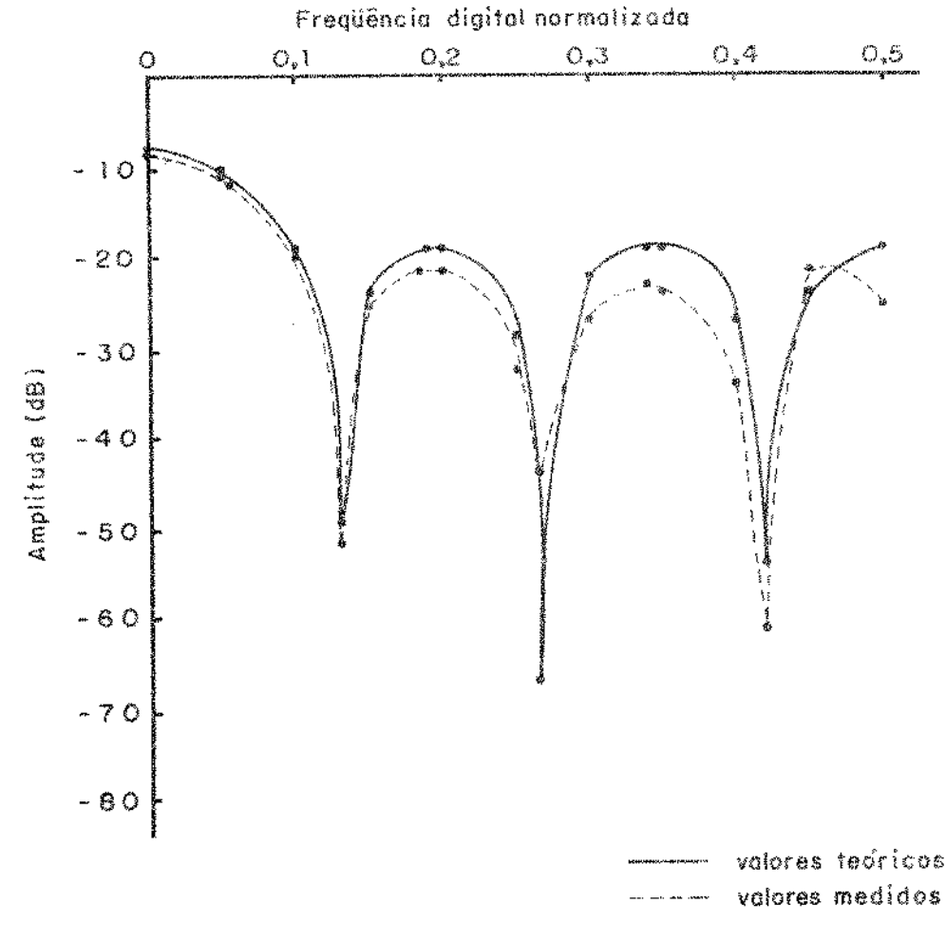



Fig. 4.17. Filtro passa baixa da tabela 3.2

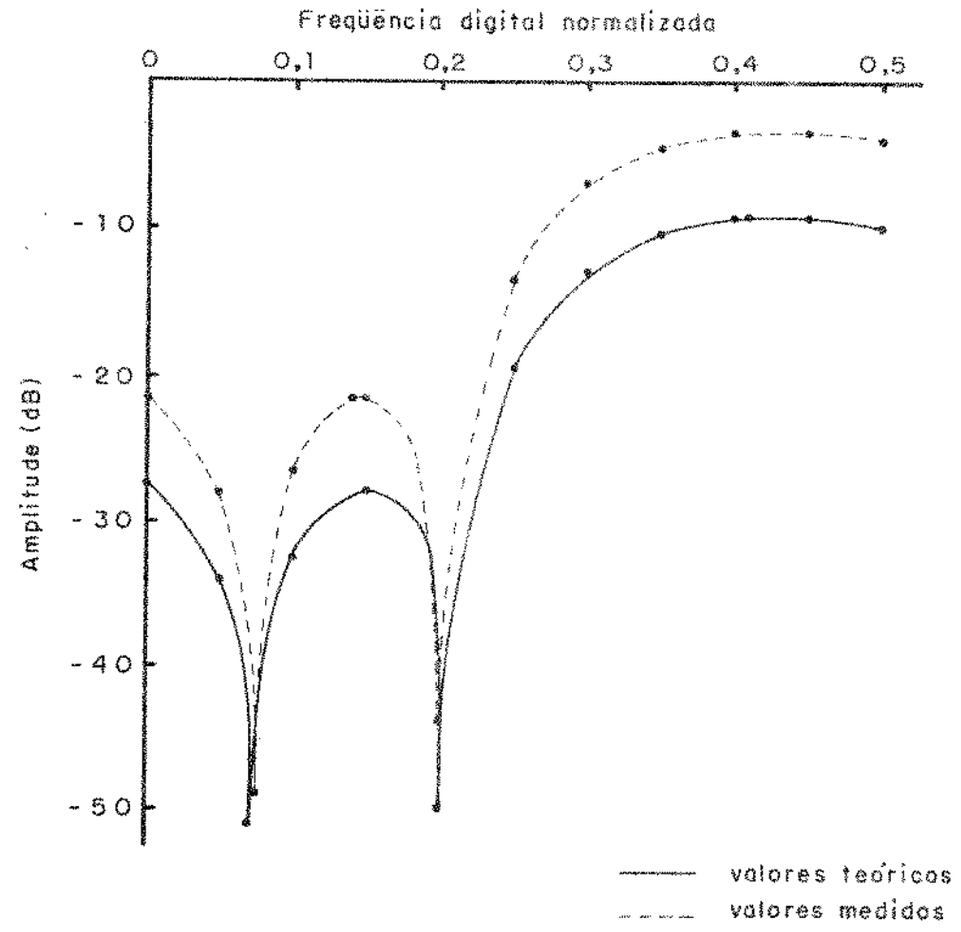



Fig. 4.18. Filtro passa alta da tabela 3.3

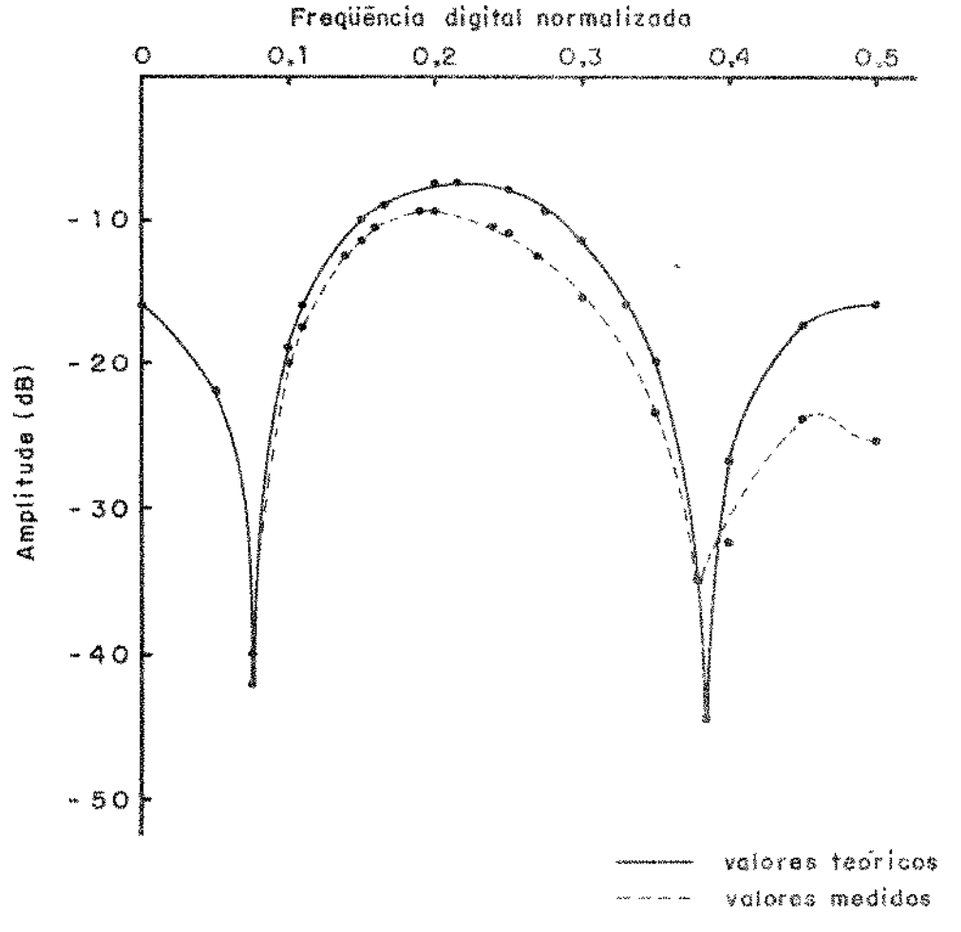
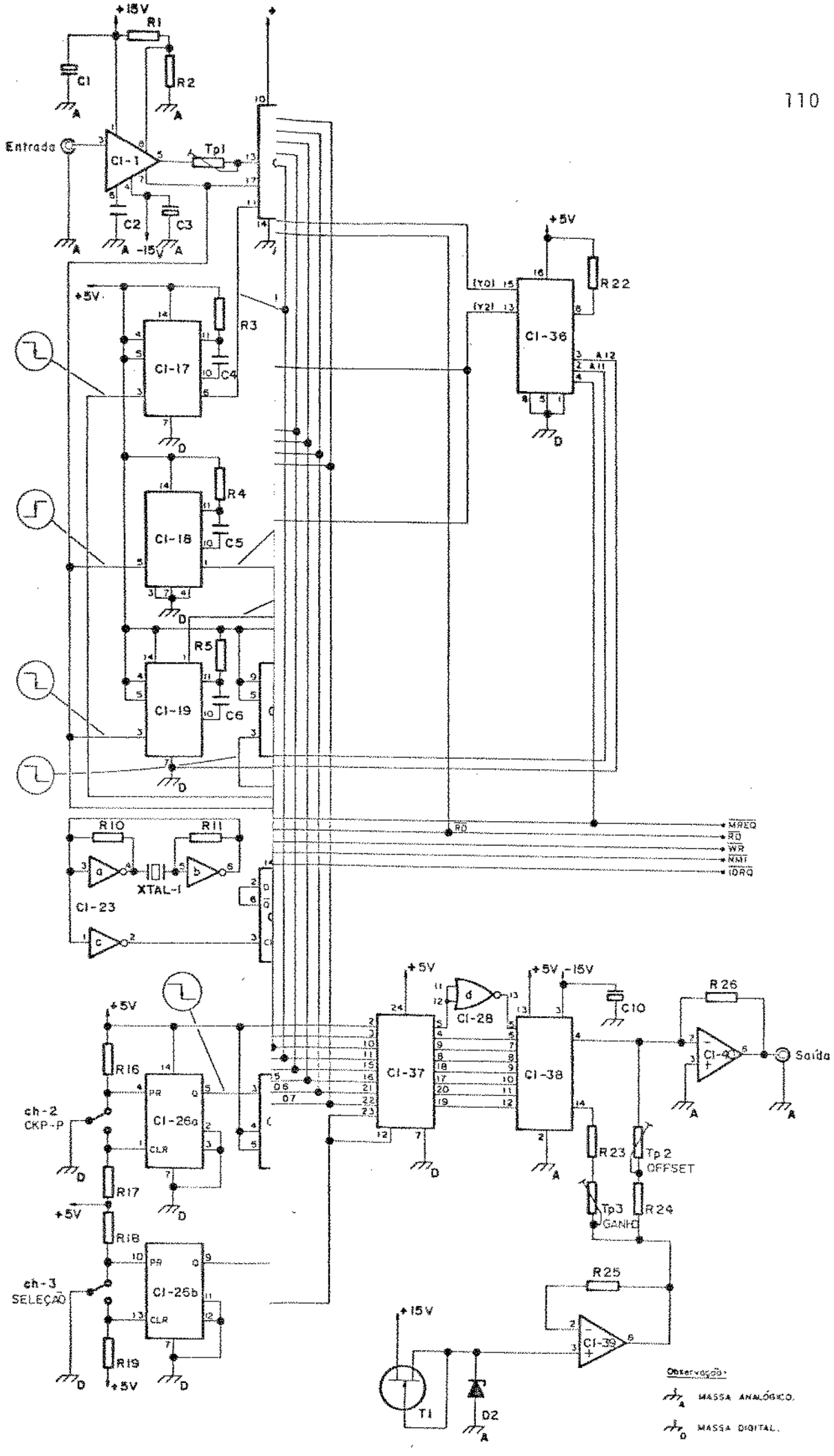



Fig. 4.19. Filtro passa faixa da tabela 3.4


+15V
R1
C1
R2
Entrada
CI-1
Tp1
C2
C3
-15V
+5V
R3
CI-17
C4
R4
CI-18
C5
R5
CI-19
C6
R22
CI-36
R10
R11
XTAL-1
CI-23
MREQ
RD
WR
NMI
IORQ
CI-37
CI-28
CI-38
C10
R26
CI-4a
Saída
ch-2
CKP-P
CI-26a
R16
R17
R18
ch-3
SELEÇÃO
CI-26b
R19
R23
Tp 2
OFFSET
Tp3
GANHO
R24
R25
CI-39
T1
D2
Observação:
MASSA ANALÓGICO.
MASSA DIGITAL.

LISTA DE COMPONENTES

RESISTORES

$R_1$ - 4,7 kΩ
$R_2$ - 1 kΩ
$R_3$ - 3 kΩ
$R_4$ - 3 kΩ
$R_5$ - 15 kΩ
$R_6$ - 3,6 kΩ
$R_7$ - 1 kΩ
$R_8$ - 1 kΩ
$R_9$ - 1 kΩ
$R_{10}$ - 1 kΩ
$R_{11}$ - 1 kΩ
$R_{12}$ - 1 kΩ
$R_{13}$ - 1 kΩ
$R_{14}$ - 1 kΩ
$R_{15}$ - 560kΩ
$R_{16}$ - 1 kΩ
$R_{17}$ - 1 kΩ
$R_{18}$ - 1 kΩ
$R_{19}$ - 1 kΩ
$R_{20}$ - 7 kΩ
$R_{21}$ - 1 kΩ
$R_{22}$ - 1 kΩ
$R_{23}$ - 1 kΩ
$R_{24}$ - 2 kΩ
$R_{25}$ - 10 kΩ
$R_{26}$ - 10 kΩ

TRIMPOTS:

$T_{P1}$ - 50Ω
$T_{P2}$ - 200Ω
$T_{P3}$ - 470Ω

CAPACITORES:

$C_1$ - 0,1 µF
$C_2$ - 100 nF
$C_3$ - 0,1 µF
$C_4$ - 1200 pF
$C_5$ - 11 nF
$C_6$ - 56 pF
$C_7$ - 1 nF
$C_8$ - 0,1 µF
$C_9$ - 100 pF
$C_{10}$ - 0,1 µF

DIODOS:

$D_1$ - 1N 914
$D_2$ - 1N 750A ou 1N4731 (Zener de 4 volts)

TRANSISTORES

$T_1$ - MPF 102 (JFET)

CRISTAL

XTAL- 1 : 3,574 MHz

CIRCUITOS INTEGRADOS:

CI-1 - LF 398
CI-2 - AD 570 JD
CI-3 - SN 7400
CI-4 - SN 74166
CI-5 - SN 7491
CI-6 - SN 7491
CI-7 - SN 7491
CI-8 - SN 7491
CI-9 - SN 7491
CI-10 - SN 7491
CI-11 - SN 7491
CI-12 - SN 7491
CI-13 - SN 7491
CI-14 - SN 7491
CI-15 - SN 7491
CI-16 - SN 7491
CI-17 - SN 74121
CI-18 - SN 74121
CI-19 - SN 74121
CI-20 - SN 74121
CI-21 - SN74LS244
CI-22 - SN74LS244
CI-23 - SN7404
CI-24 - SN74LS74
CI-25 - Z-80A
CI-26 - SN74LS74
CI-27 - SN74121
CI-28 - SN7402
CI-29 - SN7475
CI-30 - SN74LS138
CI-31 - SN74LS244
CI-32 - SN74LS244
CI-33 - AM2716
CI-34 - MM2114
CI-35 - MM2114
CI-36 - SN74LS138
CI-37 - SN74100
CI-38 - AD559KD
CI-39 - LM741
CI-40 - CA3140

# CAPITULO V: CONCLUSÃO

## 5.1. SÍNTESE DO PROJETO REALIZADO

Neste trabalho apresentamos um projeto de um filtro digital de resposta ao impulso finita, implementado em uma estrutura que faz uso da aritmética distribuída. Esta forma de implementação objetiva a execução da filtragem em tempo real. A estrutura projetada é não recursiva, faz uso do microprocessador Z-80, possui 2k bytes de EPROM e 1k byte de RAM e permite a implementação de filtros com um máximo de 10 coeficientes. As freqüências de amostragem obtidas foram 9,0 kHz e 14,7 kHz.

O projeto de coeficientes da resposta ao impulso do filtro utilizou a técnica das janelas e o projeto ótimo. Foram usados programas em BASIC e FORTRAN, respectivamente, para estes cálculos. As respostas em freqüência foram simuladas. Um programa em linguagem de máquina foi desenvolvido para cálculo da tabela de coeficientes. Esta tabela e o programa de operação foram gravados na memória EPROM do filtro digital. "O hardware" digital projetado faz uso de circuitos integrados TTL, e é alimentado por três tensões: +5V, +15V e -15V.

As medidas realizadas para filtros com 7 coeficientes se aproximaram bastante da simulação, exceto nas freqüências mais próximas da freqüência de amostragem.

## 5.2. DISCUSSÃO DO PROJETO

### 5.2.1. Tempo de Processamento

O tempo de processamento poderia ser diminuído

neste circuito, para obtenção de maiores freqüências de amostragem. As maneiras de se conseguir isto são:

a) aumento da freqüência do relógio: substituindo o cristal utilizado, poderíamos aumentar a freqüência do circuito de relógio. O valor máximo para o Z-80 é 2,5 MHz, e para o Z-80A , 4MHz. Isto implica em alcançar freqüências de amostragens da ordem de 11,9kHz e 15,9kHz para o Z80 e Z80A respectivamente.

b) redução do tempo de conversão:

O tempo de conversão pode ser diminuido pela escolha de um conversor A/D mais rápido. Também existe a possibilidade de modificar o circuito para que a conversão se processe logo após o registro RX ser carregado com um dado, quase que paralelamente ao desenvolvimento das operações. Quando novo dado fosse necessário em RX, bastaria um sinal de controle para transferí-lo do A/D. O tempo de processamento, considerando a freqüência de relógio de 1,787 MHz seria de 133T= 74,43 μs; isto corresponde ao tempo das instruções sem considerar RET, HALT e o tempo de conversão, pois esses não mais seriam necessários. Isto elevaria a freqüência de amostragem para 13,4kHz. No caso do relógio de 2,5MHz e 4MHz, a freqüencia de amostragem seria de 18,8 kHz e 33,3kHz respectivamente.

### 5.2.2. Projeto de Filtros com N Elevado

O circuito apresentado permite a realização de filtros com N menor ou igual a 10. Esta limitação decorre da capacidade de memória disponível para guardar a tabela e o programa de operação (2k bytes = 2048 posições de endereço). Então, usando mais memória, poderíamos implementar filtros com 15 coe-

ficientes, no máximo. Este é o limite de endereçamento da estrutura sem repartir os"buffers"(isto é $p = 1$, conforme capítulo II). Repartindo os"buffers"em grupos[1], é possível aumentar N. Isto resulta num tempo de processamento maior, logo,numa freqüência de amostragem mais baixa. A tabela 5.1 indica alguns valores que seriam encontrados caso fosse feita uma divisão nos "buffers".Para a montagem desta tabela foi considerada a equação (2.21).

A tabela foi montada considerando que cada grupo de"buffers"que é lido contém o número máximo de saídas, isto é,15. Entretanto outras combinações são possíveis. Por exemplo, pode-se considerar p = 4, com 8 linhas de endereço em cada conjunto. Assim, a cada"buffer"corresponde uma tabela com $2^8$=256 elementos, e no total a memória ocupada pelas 4 tabelas será de 1k byte. Neste caso o valor de N = 32.

Esta modificação não irá alterar o valor de $f_a$ e $T_{proc.}$ da tabela 5.1, mas tem a vantagem de ocupar menos memória.

Concluímos que há um compromisso entre $T_{proc}$; $f_a$, P e a capacidade de memória do sistema.

### 5.2.3. Projeto de Filtro Recursivo

O esquema do filtro pode ser facilmente transformado para uma estrutura recursiva. Para isso basta acrescentar à saída um conjunto de registros, semelhantes a RX,e ligar as suas saídas nas linhas de endereçamento da memória. A figura 2.2 in-

(1) Ver seção 2.4.

| p | f. relógio (MHz) | t. proc (μs) | $f_a$ (KHz) | N máx |
|---|---|---|---|---|
| 1 | 1,787 | 54,8 | 18,2 | |
| 1 | 2,5 | 39,2 | 25,5 | 15 |
| 1 | 4 | 24,5 | 40,8 | |
| 2 | 1,787 | 86,2 | 11,6 | |
| 2 | 2,5 | 61,6 | 16,2 | 30 |
| 2 | 4 | 38,5 | 26,0 | |
| 3 | 1,787 | 117,5 | 8,5 | |
| 3 | 2,5 | 84,0 | 11,9 | 45 |
| 3 | 4 | 52,5 | 19,1 | |
| 4 | 1,787 | 148,9 | 6,7 | |
| 4 | 2,5 | 106,4 | 9,4 | 60 |
| 4 | 4 | 66,5 | 15,0 | |

Tabela 5.1. Estudo da divisão dos "buffers".

dica a forma de implementar a estrutura recursiva.

## 5.3. SUGESTÃO PARA FUTUROS PROGRESSOS

As modificações que sugerimos no projeto seriam de duas ordens basicamente:

a) extensão da capacidade do filtro já montado: a freqüência de amostragem poderia ser aumentada com o uso de um conversor A/D mais rápido e/ou aumento da freqüência de relógio; o valor de N pode ser aumentado com uso de mais memória e/ou divisão dos "buffers".

b) utilizar um microcomputador associado a um "hardware":

Como vemos o circuito projetado é praticamente um microcomputador, pois possui CPU, memória, interfaces. Assim, sugerimos um estudo para o uso de um microcomputador comercial associando-o a um "hardware" especial, que permita o endereçamento da tabela vetorada pelo sinal de entrada. O sistema assim constituído teria todas as facilidades de um microcomputador e funcionaria como um filtro.

Assim, o cálculo da tabela, dos coeficientes da resposta ao impulso, da resposta em freqüência, poderiam ser realizados no mesmo sistema.

Esta estrutura utilizando um microcomputador comercial e uma interface é bastante versátil e poderia ser usada em laboratórios. Sugerimos que o conjunto seja chamado de "filtro digital programável".

c) utilizar um microprocessador de maior comprimento de palavra.

Se o projeto fosse adaptado para um microprocessador de 16 ou 32 bits várias vantagens seriam obtidos; por exemplo,

poderia implementar filtros com maior número de coeficientes, com maiores taxas de amostragem e fazer uso de facilidades peculiares de endereçamento desse tipo de microprocessadores.

## 5.4. CONCLUSÃO FINAL

O filtro projetado atendeu às especificações de projeto quanto à velocidade de operação (processamento em tempo real) e precisão dos resultados (comparação entre valores teóricos e práticos da resposta em freqüência). O protótipo desenvolvido é adequado para aplicações didáticas em cursos sobre processamento digital de sinais, por exemplo. O número de coeficientes pode ser ampliado. A freqüência de amostragem da ordem de 14,3kHz foi considerada satisfatória, mas pode ser aumentada. A implementação pode evoluir constituindo um "filtro digital programável" e para o uso de microprocessadores de maior comprimento de palavra. A estrutura de programasapresentada pode ser seguida com facilidade para o cálculo de filtros.

A estrutura projetada é bastante versátil permitindo implementar vários tipos de filtros com uma simples mudança da tabela de coeficientes na memória.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] ALLEN, P. J. & HOLT, G. J. Controlling a vector-multiplier based digital filter by using a microprocessor. Computer and digital techniques, London, 2(4):177-80, Aug. 1979.

[2] ANTONIOU, Andreas. Digital filters: analysis and design. New Delhi, Tata McGraw-Hill, 1980. 524 p.

[3] BARDEN JUNIOR, William. The Z-80 microcomputer handbook. Indianapolis, Howard W. Sams, 1978. 304 p.

[4] BATISTA, Léo & KATAKURA, Gérson M. Elementos de programação em BASIC. São Paulo, E. Blücher, 1983. 131 p.

[5] BERLIN, Howard M. Projetos com amplificadores operacionais com experiências. São Paulo, EDITELE, 1977. 231 p.

[6] FARHANG-BOROUJENY, B. & HAWKINS, G. J. Study of the use of microprocessors in digital filtering. Computer and digital techniques, London 2(4):169-76, Aug. 1979.

[7] JACKSON, Leland B.; KAISER, James F.; McDONALD, Henry S. An approach to the implementation of digital filters. IEEE-transactions on audio and electroacoustic, New York, 16: 413-21, Sept. 1968.

[8] LITTLE, Warren D. An algorithm for high-speed digital filters. IEEE-transaction on computers, New York, 23 (5): 466-9, May, 1974.

[9] McGLYNN, Daniel R. Microprocessors: technology, architecture and applications. New York, John Wiley, 1976.207 p.

[10] OPPENHEIM, Alan V. et alii. Applications of digital signal processing. Englewood-Cliffs, Prentice-Hall, 1975. 499p.

[11] OPPENHEIM, Alan V. & SCHAFER, Ronald W. Digital signal processing. Englewood-Cliffs, Prentice-Hall, 1975.585 p.

[12] PELED, A. & LIU, B. A new hardware realization of digital

Filters. IEEE-transaction on acoustic speed and signal processing, New York, 22:456-62, Dec. 1974.

[13] POOLE, Low; MCNIFF, Martin; COOK, Steven. Apple II: guia do usuário. São Paulo, McGraw-Hill do Brasil, 1984. 375p.

[14] PROJETO com circuitos integrados TTL. Rio de Janeiro, Guanabara Dois, 1978. 325 p.

[15] RABINER, Laurence R. & GOLD, Bernard. Theory and application of digital signal processing. Englewood Cliffs, Prentice-Hall, 1978. 762 p.

[16] SERRA, Celso Penteado. Prática de programação do 8080A. São Paulo, Monitor, 1981. 256 p.

[17] TAUB, Herbert & SCHILLING, Donald. Digital integrated electronics. Tokyo, McGraw-Hill Kogakusha, 1977. 650 p.

[18] TITUS, Christofer A.; LARSEN, David G.; TITUS, Jonathan A. 8080/8085 software design. Indianapolis, Howard W. Sams, 1980, 2v.

[19] ZIEMER, Rodger E.; TRANTER, William H.S.; FANNIN, Ronald D. Signal and systems: continuous and discret. New York, Mcmillan Publishing, 1983. 487 p.

Manuais Técnicos

[20] ANALOG devices: data acquisition products catalog. Massachusetts, Analog Devices, 1978. 600 p.

[21] LINEAR databook. Santa Clara, National Semiconductor Corporation, 1976.

[22] MANUAL DE hardware DGT-100, s.l., DIGITUS, s.d.

[23] MEMORY data book. Santa Clara, National Semiconductor Corporation, 1976.

[24] MOTOROLA memory data manual. Austin, Motorola, 1982.

[25] THE TTL data book for design engineers. 2ª ed. Dallas, Texas Instruments Incorporated, 1981.

[26] VEIGA, Wanderley. Pico computador: manual de operação. Curitiba, Departamento de Eletrônica do CEFET-PR, s.d.