Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Semicondutores, Instrumentos e Fotônica

"Implementação Monolítica de uma Rede Neural para Assistência em Equalização Adaptativa de Sinais"

Candidato : M.Sc. Wilfredo Machaca Luque
Orientador : Prof. Dr. Furio Damiani

Tese apresentada à FEEC/UNICAMP, como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica, Área de Eletrônica e Comunicações.

Este exemplar corresponde a redação final da tese defendida por .....
..... e aprovada pela Comissão
Julgada em ...../...../.....

.....
Orientador

Campinas, Abril de 1999.

9915928

UNIDADE BC
N.º CHAMADA :
V. Ex.
TOMBO BC/ 38408
PROC 229/99
C D X
PREÇO R$ 11,00
DATA 21/08/99
N.º CPD

CM-00125598-1

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

L974i

Luque, Wilfredo Machaca
Implementação monolítica de uma rede neural para assistência em equalização adaptativa de sinais. / Wilfredo Machaca Luque.--Campinas, SP: [s.n.], 1999.

Orientador: Furio Daminani.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Circuitos integrados - Projetos. 2. Equalizadores (Eletrônica). 3. Filtros adaptativos. 4. Processamento de sinais – Técnicas digitais. 5. Redes neurais. I. Damiani, Furio. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

A mi esposa, Lucy Doris
y mi hijo, Sérgio André

Agradecimentos

Agradeço ao Prof. Furio Damiani pela orientação, confiança e apoio no desenvolvimento do presente trabalho.

Ao CNpQ pelo apoio financeiro.

Ao Marcelo Jara pelas frutíferas discussões sobre o modelo de Kohonen e pela amizade brindada.

Ao Prof. Carlos Reis pelas fotografias da placa de teste, pela oportunidade de participar nos projetos do Laboratório de Pesquisa Magneti Marelli.

Ao José Torrico pela amizade e também pela ajuda no entendimento do módulo de desenvolvimento da Motorola, DSP96002ADM.

À minha companheira de sempre, Lucy Doris, pela confiança, compreensão e apoio em todas as etapas deste trabalho.

Aos colegas e amigos do DSIF : Marcelo Tosin, Dahge, Juan Carlos, Walter, Everson, Fábio, Eliane, Li Bin Bin, Marco, Marcos, Raquel.

Aos colegas do Lab. de Pesquisa Magneti Marelli : Jorge, Beto, Edevaldo, Edison, Danielle, Marcos, Isaias,

Resumo

Neste trabalho apresentam-se o projeto e as medidas de caracterização de uma rede neural para uso na equalização adaptativa de sinais, implementada em um ASIC *(Aplication Specific Integrated Circuit)* VLSI.

O equalizador estudado é um DFE *(Decision FeedBack Equalizer),* para comunicação digital usando modulação QAM *(Quadrature Amplitude Modulation)* com uma constelação de 16 símbolos. O equalizador adaptativo que utiliza o ASIC projetado no estágio de decisão foi comparado com equalizadores convencionais.

A rede neural utilizada segue o modelo conhecido como Mapas Auto-Organizados de Kohonen *(SOM - Self-Organizing Maps).*

O ASIC, realizado em tecnologia CMOS de 0.8μm, implementa uma rede neural de 4 células (2x2) e é modular, permitindo seu uso para constelações de 16 símbolos.

Finalmente, são apresentados os resultados dos testes, realizados com a ajuda do módulo de desenvolvimento DSP96002ADM da Motorola.

Summary

The design and test of a Kohonen neural network VLSI ASIC for adaptive signal equalization purposes is presented.

The ASIC is used in the decision stage of a DFE (Decision Feedback Equalizer) and QAM (Quadrature Amplitude Modulation) with 16 symbols was considered.

A comparative study of this adaptive equalizer and conventional types was done.

The ASIC was fabricated in a CMOS 0.8μm technology and contains 2x2 Kohonen cells, fashioned in a modular circuit.

The test results, done with the help of a Motorola-DSP96002ADM development module are shown.

Conteúdo

Capítulo 1: Introdução 5

Capítulo 2: O Sistema de Comunicação Digital 8
2.1 O Esquema de Modulação 9
2.2 O Canal de Transmissão - Modelo 11
2.3 As Distorções no Sistema de Transmissão 13
2.4 Filtros Adaptativos 15
2.5 Equalização de Sinais 17

Capítulo 3: Os Mapas Auto-Organizados de Kohonen (SOM). A Rede Neural. 20
3.1 Alguns Conceitos Preliminares 20
3.1.1 Reconhecimento de Padrões 20
3.1.2 Vetores de *Características* e Espaço de *Características* 21
3.1.3 Funções de Discriminação 21
3.1.4 Técnicas de Classificação 22
3.1.5 Classificação Vizinho Mais Próximo 22
3.1.6 Distâncias 23
3.2 O Conceito de Auto-Organização 23
3.3 O Algoritmo de Kohonen 25
3.3.1 Justificativa Biológica 27
3.4 Atualização dos Pesos Internos 28
3.4.1 Valores Iniciais dos Pesos Internos 28
3.5 Vizinhança Topológica 29
3.5.1 Redução da Vizinhança 29
3.5.2 Funções de Densidade de Probabilidade 29

Capítulo 4: Equalização Adaptativa de Sinais Baseada em Redes Neurais 32
4.1 O Equalizador Linear ou Equalizador Transversal Linear (LTE) 33
4.2 O Algoritmo de Adaptação dos Coeficientes - LMS 35
4.3 Equalizadores para Sistemas com Modulação QAM 38
4.4 O Equalizador com Decisão Realimentada (DFE) 40
4.5 Equalização Adaptativa Baseada em Redes Neurais. Neuro-Equalização 41
4.5.1 Equalização baseada no *Multilayer Perceptron* 42
4.5.2 Equalização baseada em Redes Neurais Recorrentes 44
4.6 Equalização Adaptativa Baseada no Mapa Auto-Organizado (SOM) 46
4.6.1 O Algoritmo de Adaptação 48
4.6.2 A Vizinhança Topologica 49

Capítulo 5: Implementação em *Software* - Simulações e Resultados 51
5.1 O Gerador de Sinais 52
5.1.1 As Distorções Não-Lineares 54
5.2 O Canal de Transmissão 55

5.2.1 A Geração do Sinal de Ruído 57
5.3 O Equalizador Convencional 58
5.4 O Neuro-Equalizador 60
5.5 Simulações e Resultados 61

Capítulo 6: Implementação em *Hardware* - Projeto do ASIC 68
6.1 Representação e Metodologia de Projeto 68
6.2 Arquitetura da Rede Neural 70
6.3 Aritmética e Representação Numérica 71
6.4 Arquitetura da Célula-Neurônio 75
6.4.1 Primeira Proposta 76
6.4.2 Segunda Proposta 78
6.4.3 Terceira Proposta 80
6.5 Descrição do Circuito 83
6.5.1 A Rede Neural 83
6.5.2 A Célula Neurônio 87
6.5.2.1 A Unidade Soma de Produtos *(SOP - Sum Of Products)* 88
6.5.2.2 Registradores Internos 90
6.5.2.3 O Bloco Interface de Vizinhança 91
6.5.2.4 Os Blocos Multiplexadores 91
6.5.2.5 O Bloco Decodificador de Endereços 92
6.5.2.6 A Máquina de Estados 93
6.5.3 A Unidade WTA *(Winner-Takes-All)* 99
6.6 Simulações Pré-*Layout* 100
6.7 Síntese do *Layout* 109
6.8 Verificação do *Layout* - Extração de Parâmetros Parasitas 110
6.9 Simulação Pós-*Layout* 112
6.10 Características do ASIC 114

Capítulo 7: Testes 116
7.1 Esquema de Teste 116
7.2 O Módulo DSP96002ADM da Motorola 119
7.3 A Placa Interface 119
7.4 O Programa de Interface e Teste 120
7.5 Resultados e Discussão 122

Conclusão 129

Referências Bibliográficas 131

Anexos 137

Capítulo 1:

Introdução

A rápida integração das comunicações, computação e tecnologia multimídia tem se tornado a força chave do impulso para a rede global de informação tanto para aplicações científicas como de engenharia. As pessoas e suas máquinas deveriam ser capazes de acessar a informação e se comunicarem uns com os outros de forma fácil, confiável e segura utilizando um meio ou uma combinação de meios: voz, texto e vídeo. Além disso, a demanda crescente de mobilidade física e portabilidade dos serviços motiva fortemente a evolução das comunicações pessoais.

A indústria crescente e de constantes mudanças das telecomunicações tem se mostrado aberta ao uso dos filtros adaptativos, que têm diversas aplicações nos campos de utilização do processamento de sinais[Cow85]. Atualmente, a importância da filtragem adaptativa cresce com o desenvolvimento da tecnologia de microcomponentes, tornando possível a implementação dos, cada vez mais complexos, algoritmos de correção. Os campos de aplicação mais importantes para a filtragem adaptativa são: o cancelamento de eco, a codificação de voz e os sistemas de comunicação[Jon95].

A transmissão da informação em condições adversas tem gerado uma grande necessidade de filtros adaptativos. Tradicionalmente, o filtro adaptativo é cascateado com um canal de transmissão linear ou não linear. O papel desse filtro, chamado de equalizador de sinais, é tal que a sua função de transferência seja o inverso da do canal, de forma a compensar a resposta indesejável do canal que pode ser variante no tempo, nos casos de comunicações móveis. O equalizador tem como um dos seus propósitos a diminuição da interferência intersimbolica *(ISI - Inter Symbol Interference)* de forma a maximizar a probabilidade de decisões corretas[Qur85]. Outro propósito é a compensação de não linearidades.

A estrutura mais comum de filtro adaptativo é a do filtro transversal ou FIR *(Finite Impulse Response)* com coeficientes ajustáveis. Os filtros adaptativos podem ser dedicados à compensação de distorções não lineares como o Equalizador com Decisão Realimentada *(DFE - Decision Feedback Equalizer)*, apesar de não ser um equalizador não linear completo, pois compensa um conjunto restrito de não linearidades. Uma vantagem importante do DFE é a sua capacidade de compensar a ISI de forma melhor do que o filtro transversal convencional. Existem vários algoritmos para a adaptação dos coeficientes do equalizador, sendo o mais comuns os algoritmos LMS *(Least Mean Square)*[Cow85, Bel87] e RLS *(Recursive Least Square)*.

As redes neurais artificiais têm sido utilizadas em diferentes áreas de aplicação, como no reconhecimento de padrões, sistemas complexos de filtragem adaptativa, robótica, etc. As redes neurais podem ser classificadas em supervisionadas e não supervisionadas segundo o tipo de aprendizagem. Nas primeiras, a aprendizagem é controlada por um supervisor, que pode ser uma função de minimização de erro ou de custo e que permite condicionar a modificação dos pesos em função da resposta da rede e da comparação com a saída desejada. Exemplos deste tipo de aprendizagem são os modelos de Widrow e Hoff

(ADALINE, Adaptive Linear Element) e o de Rosenblatt *(Perceptron)*[Wid90,Ros60,Jar97]. Um modelo importante é o estabelecido por Rumelhart, McClelland e Williams *(MLP - Multilayer Perceptron)*[Rum86] cuja regra de aprendizagem é conhecida como *backpropagation*.

Nas redes neurais com aprendizagem não supervisionada as próprias redes formam suas próprias classificações de dados de treinamento; para tanto, é necessário fazer duas suposições básicas sobre as redes:

- Uma classe é definida pelos padrões de entrada que compartilham *características* em comum;
- A rede deve ser capaz de identificar *características* comuns nos padrões que são proporcionados à entrada da rede. Os mapas auto-organizados de Kohonen[Koh89,Koh90c] *(SOM* ou *SOFM - Self-Organizing Feature Maps)* são redes deste tipo e utilizam um aprendizado não-supervisionado para modificar os pesos internos da rede e modelar as *características* encontradas nos dados de treinamento.

Talvez uma das razões para a utilização de redes neurais artificiais seja simplesmente a curiosidade que sempre impulsionou a ciência. No entanto, trabalhos pioneiros como o de Siu et al [Siu90], têm mostrado que equalizadores baseados na rede *MLP* foram superiores aos equalizadores convencionais em termos da medida de desempenho da taxa de erro de bits *(BER - Bit Error Rate)*. Uma segunda razão para se utilizar redes neurais nos equalizadores adaptativos é a de engenharia, dado que uma determinada solução implica em complexidade computacional e medidas de desempenho e é nesse sentido que a solução proposta por Kohonen et al [Koh90a, Koh90b], em 1990, é interessante pois tendo-se uma estrutura de equalizador mais simplificada consegue-se manter o mesmo desempenho e complexidade computacional.

As comunicações contemporâneas sem fio, móveis ou fixas, são baseadas em tecnologias de comunicação digital e o seu sucesso comercial tem sido possibilitado em parte pelo projeto de Processadores Digitais de Sinais (DSP) e circuitos integrados de aplicação específica *(ASIC - Application Specific Integrated Circuit)*.

Dentre as linhas de pesquisa do Departamento de Semicondutores, Instrumentos e Fotônica (DSIF) da FEEC/UNICAMP está a implementação de estruturas de Redes Neurais Artificiais tanto em *software* quanto em *hardware* e sua aplicação em diversas áreas do conhecimento, tais como processamento digital de sinais para reconhecimento de sinais acústicos[Jar95a] e em telecomunicações para equalização adaptativa de sinais, sendo este último assunto do presente trabalho.

Nesse sentido um primeiro trabalho de implementação em circuito VLSI da Rede Neural ou Mapa Auto-Organizado de Kohonen foi realizado[Jar97]. Essa primeira realização visou uma implementação genérica do modelo da rede neural, de forma que fosse possível a configuração de todos os parâmetros do algoritmo. Foram estudadas várias métricas de distância para a determinação da célula vencedora do processo competitivo que se dá a cada novo vetor apresentado à rede, analisando a sua influência na convergência do algoritmo e na formação dos mapas finais.

As implementações em *hardware* das redes neurais podem ser classificadas em arquiteturas de propósito geral e de aplicação específica. Na primeira classe, as arquiteturas implementam um conjunto de modelos de redes neurais e geralmente isto é feito através da programação das conexões topológicas e algoritmos de aprendizagem, tais como os aceleradores para computadores seqüênciais. Na segunda classe o *hardware* é projetado de forma a obter o melhor casamento para o modelo específico e seus circuitos são geralmente otimizados para se obter tamanhos compactos ou alto desempenho e serem utilizados em

aplicações de tempo real, tal como equalizadores baseados em redes neurais para sistemas de comunicações digitais.

No presente trabalho apresenta-se o projeto do ASIC de uma rede neural, que segue o modelo de Kohonen, com 4 células-neurônio, para ser aplicado na equalização adaptativa de sinais modulados segundo o esquema QAM-16. Foi possível integrar 4 células-neurônio (nós processadores) em um único *chip*, e sendo o seu projeto modular, é possível formar estruturas de maior nível, como QAM-16. O protótipo do ASIC foi elaborado dentro do Programa de Fabricação de Circuitos Integrados da FAPESP. A tecnologia de implementação foi CMOS 0.8μm da *foundry* AMS *(Austria Mikro Systeme).*

Como ferramentas de projeto foram utilizadas a linguagem C na especificação de alto nível do sistema equalizador e a linguagem VHDL para o projeto do ASIC, contando para tanto com o conjunto de ferramentas da empresa *Mentor Graphics Corp.*

Os testes foram realizados com ajuda do módulo de desenvolvimento *DSP96002ADM (Application Development Module)* da *Motorola*[Mot_a]. Este módulo proporciona versatilidade, e numa aplicação em tempo real a conjunção de um DSP *(Digital Signal Processor)* comercial e do ASIC projetado para implementar o SOM demonstrou ser uma boa solução. Para a realização do teste foi necessário o projeto de uma placa de interface para alojar o *chip* projetado.

A organização da dissertação é a seguinte:

No capítulo 2, faz-se uma breve revisão de conceitos básicos sobre um sistema de comunicação digital, abordando assuntos de modulação, canal de transmissão, distorções no sistema de transmissão e filtros adaptativos.

No capítulo 3, realiza-se uma introdução ao modelo dos mapas auto-organizados ou rede neural de Kohonen.

No capítulo 4, trata-se da equalização adaptativa de sinais baseada em redes neurais. Estudam-se brevemente as estruturas convencionais de equalização e o algoritmo de adaptação dos coeficientes. Finalmente aborda-se o tema da equalização assistida pela rede neural de Kohonen.

No capítulo 5, apresenta-se a implementação na linguagem C do sistema de equalização convencional e do assistido pela rede neural de Kohonen, mostrando as simulações e os resultados obtidos.

No capítulo 6, mostra-se o projeto do ASIC que implementa a rede neural de Kohonen. Apresenta-se a arquitetura da célula-neurônio e as suas diversas propostas, simulações pré e pós *layout* e características finais do ASIC.

No capítulo 7, apresenta-se o projeto da placa de interface com o módulo de desenvolvimento utilizado e o processo de teste do ASIC. Mostram-se também os resultados das medições.

Capítulo 2:

O Sistema de Comunicação Digital

O propósito de um sistema de comunicação é a transmissão de mensagens de uma fonte para um destino através de um meio, o canal de transmissão. Na figura 2.1 mostra-se um sistema típico de comunicação digital (modelo em banda básica). As características das mensagens dependem do tipo de informação gerada pela fonte, como, por exemplo, um sinal de aúdio, de vídeo ou de dados.

As partes essenciais de um sistema de comunicação são o transmissor, o canal de transmissão e o receptor, cada um cumprindo uma função específica. O transmissor processa o sinal de entrada para produzir outro sinal que se adeque às características do canal de transmissão; tal processo geralmente envolve modulação e também codificação. Para comunicações de longa distância geralmente requer-se uma onda portadora alternada de alta freqüência, a qual é modulada pelas mensagens codificadas, que no caso de modulação AM *(Amplitude Modulation)* simplesmente consiste na multiplicação da onda portadora pela mensagem.

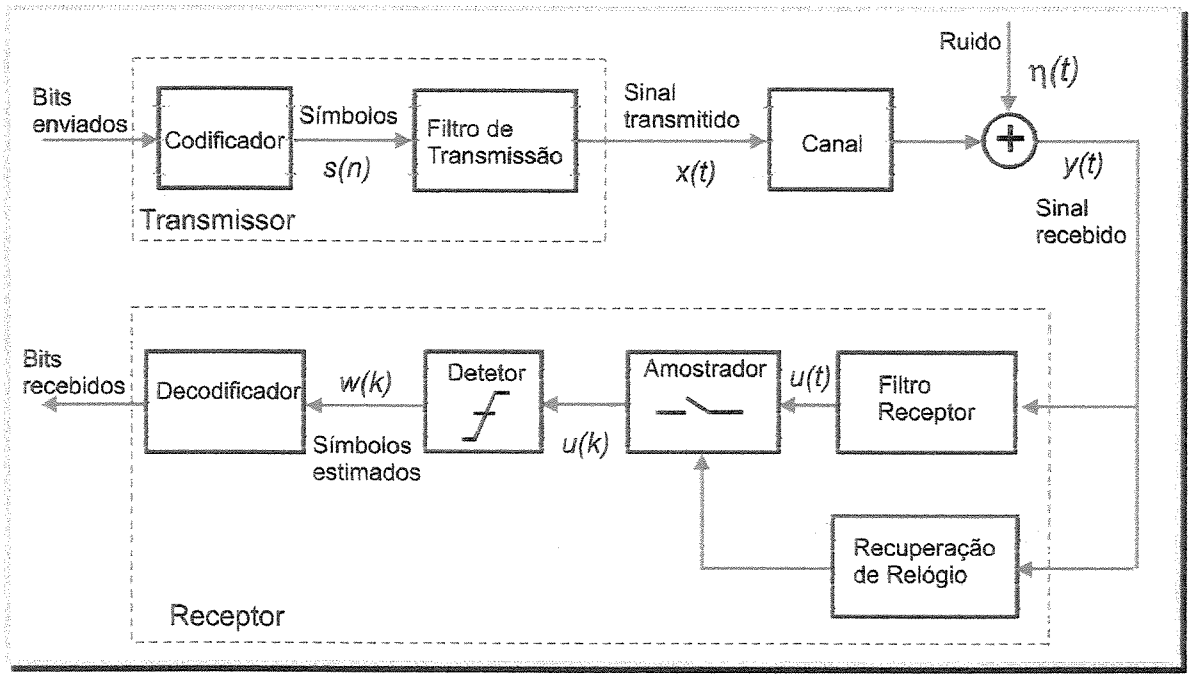


Figura 2.1. Diagrama dos Elementos de um Sistema de Comunicação Digital (modelo em banda básica).

A seqüência de bits recebidos pelo transmissor são mapeados pelo codificador (vide figura 2.1), agrupando-se em grupos de p bits, numa seqüência de símbolos que contém 2^p valores discretos. Esse sinal, que é uma seqüência discreta de símbolos, é transformado num sinal contínuo no tempo para poder ser transmitido num canal contínuo no tempo. Essa transformação é realizada pelo de filtro de transmissão.

O canal de transmissão, normalmente a parte que não é possível mudar, une os dois extremos do sistema de comunicação, podendo ser um par de fios, um cabo coaxial, um enlace de rádio ou uma fibra óptica. Cada tipo de canal introduz um grau de atenuação na transmissão, a potência do sinal decrescendo em função da distância da fonte.

O receptor opera sobre o sinal que procede do canal de transmissão e o prepara para uma forma em que possa ser reutilizado, para isto o sinal é amplificado de forma a compensar as perdas da transmissão, demodulado e decodificado para reverter o processamento que lhe foi dado no transmissor. Além disso, a filtragem e a equalização são importantes na recepção do sinal. Como estágio final, antes da conversão à forma desejada, tem-se a detecção do sinal.

Para tornar possível a estimativa da seqüência de símbolos enviada pelo transmissor, é necessário conhecer com precisão a sua informação de temporização *(timing)* que é compreendida da sua freqüência e sua fase. Esta tarefa é realizada pelo bloco de recuperação de relógio e de portadora (vide figura 2.1). Idealmente, o sinal contínuo no tempo deveria ser amostrado no instante quando é igual ao símbolo transmitido de forma a evitar a interferência com os símbolos vizinhos. O bloco detetor proporciona uma estimativa do símbolo transmitido a partir da seqüência de símbolos amostrados, aplicando-lhe uma série de limiares de decisão.

2.1 O Esquema de Modulação

O projeto dos filtros e equalizadores adaptativos, utilizados no bloco receptor de um sistema de comunicação, depende do tipo de modulação e de codificação do sinal transmitido.

A modulação é o processo pelo qual certas características da onda portadora são mudadas de acordo com a onda modulante. Em comunicações digitais a onda modulante é composta por dados binários ou por uma codificação *M-ária* destes. No entanto a portadora normalmente é uma onda senoidal que pode ser modulada em amplitude (ASK - *Amplitude-Shift Keying*), em freqüência (FSK - *Frequency-Shift Keying*) ou em fase (PSK - *Phase-Shift Keying*).

Em um esquema *M-ário* de modulação podem ser enviados M sinais, $s_1(t)$, $s_2(t)$, ... ,$s_M(t)$, em intervalos de duração T. M é o número de sinais possíveis e é igual a 2^p, sendo p inteiro. Cada um dos M sinais é chamado de símbolo e a velocidade com que estes símbolos são transmitidos mede-se em *bauds* (símbolos por segundo). A duração de um símbolo, T, é igual a pT_b, onde T_b é a duração de um bit. Esses sinais são gerados mudando seja a amplitude, a freqüência ou a fase, assim temos os conhecidos esquemas de modulação digitais ASK *M-ário*, PSK *M-ário* e FSK *M-ário*.

Como caso particular de modulação QAM *(Quadrature Amplitude Modulation) M-ário* é empregado combinando dois métodos diferentes de modulação numa forma híbrida que consiste na variação de amplitude e de fase simultaneamente [Hay88]. Esse conjunto de sinais conformam a chamada constelação e consiste de uma estrutura reticular de pontos-mensagem, os símbolos, como pode-se observar na figura 2.2 para o caso de M=16.

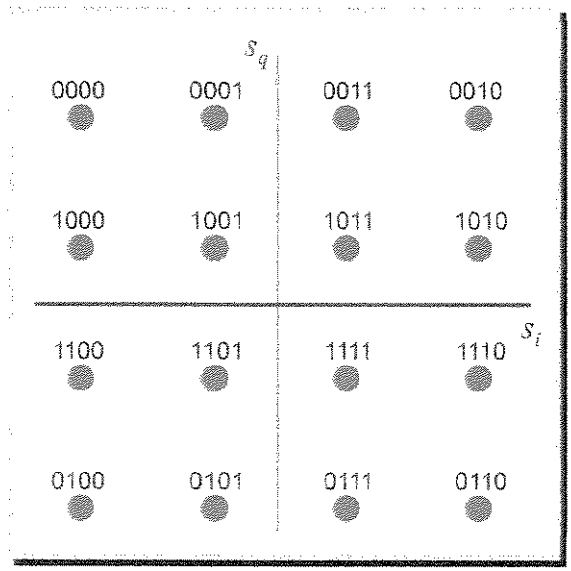


Figura 2.2. Constelação de sinais QAM M-ário com M=16. s_i, componente em fase; s_q, componente em quadratura

Qualquer sinal passa-faixa é possível descrevê-lo na forma de quadratura [Car86].

$$s_c(t) = A_c\left[s_i(t)\cos(w_c t + \theta) - s_q(t)\mathrm{sen}(w_c t + \theta)\right] \tag{2.1}$$

onde: $f_c = \omega_c / 2\pi$ é a freqüência da portadora, A_c a amplitude e θ a fase, todas constantes. Os sinais s_i e s_q que contêm o dado a ser transmitido são as componentes em fase (i) e em quadratura (q). Estes sinais são modulados por um conjunto discreto de amplitudes a_i e b_i, elementos de uma matriz *LxL*, onde $L=(M)^{1/2}$ que para o caso de QAM-16 *(L=4)*, correspondente à figura 2.2, tem-se a seguinte matriz:

$$(a_i, b_i) = \begin{bmatrix} (-3,3) & (-1,3) & (1,3) & (3,3) \\ (-3,1) & (-1,1) & (1,1) & (3,1) \\ (-3,-1) & (-1,-1) & (1,-1) & (3,-1) \\ (-3,-3) & (-1,-3) & (1,-3) & (3,-3) \end{bmatrix} \tag{2.2}$$

Os sistemas de rádio da primeira geração usavam modulações como a de PSK-2 e 4, pouco depois o PSK-8 foi também introduzido. Seguiram os esquemas de modulação QAM tornando-se bastante comuns os de QAM-16 e QAM-64. Atualmente são utilizados os de QAM-256 e QAM-1024.

A modulação linear é a mais utilizada nos sistemas de rádio digital e baseia-se no fato de que no mesmo canal têm-se duas ondas portadoras idênticas defasadas 90^0 uma da outra. Os sinais digitais que são modulados sofrem antes uma conversão D/A e uma filtragem passa-baixa, para depois passarem por um filtro passa-faixa antes de serem somados. No diagrama da figura 2.3 mostra-se um transmissor QAM *M-ário* ao qual dá-se como entrada uma

seqüência binária a uma razão de $R_b=1/T_b$, o conversor série/paralelo gera então duas seqüências binárias cujas razões são $R_b/2$. Cada derivação sofre uma conversão 2-L e a partir destas seqüências geram-se as componentes em quadratura e em fase e finalmente são multiplexadas em quadratura para gerar o sinal QAM *M-ário*.

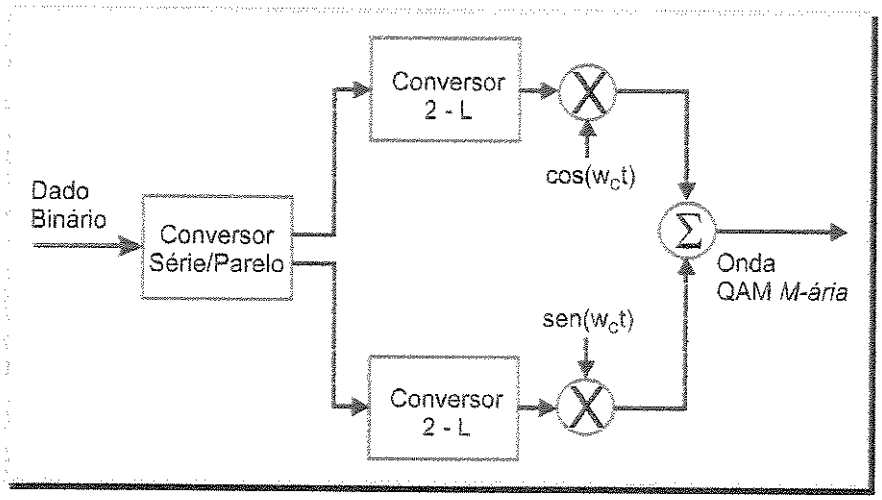


Figura 2.3. Diagrama de blocos de um transmissor QAM *M-ário*

Os métodos de modulação e de codificação utilizados no sistema de comunicação afetam a velocidade de transmissão dos símbolos. A modulação pode ser vista como uma operação de processamento de sinais e a codificação como uma operação de processamento do símbolo utilizada para o melhoramento do desempenho do sistema de comunicação.

2.2. O Canal de Transmissão - Modelo

O uso de um determinado tipo de modulação e codificação depende da aplicação e das características do canal, tais como a largura de faixa e a potência que são os recursos mais importantes dos quais o projetista dispõe. Outras características a serem consideradas são: o grau de determinação das respostas em amplitude e em fase do canal, se o canal é linear ou não-linear; e o quanto o canal é imune à interferência externa.

O canal telefônico é uma excelente opção para comunicações a longa distância devido a seu desenvolvimento (os últimos 100 anos) ter aglomerado no mundo uma grande variedade de meios de transmissão (cabos, fibras ópticas, rádio e satélites) e sistemas complexos de comutação. Este canal usa uma faixa de 300 Hz a 3400 Hz, uma relação sinal-ruído em torno de 30 dB e uma resposta quase linear. Devido às transmissões de dados e imagens serem sensíveis a atrasos de fase, uma transmissão eficiente de tais sinais requer o uso de equalizadores adaptativos e esquemas de modulação complexos para manter a resposta larga da amplitude e a resposta linear da fase dentro da faixa de freqüência utilizada.

Um sistema de rádio de microondas, que opera geralmente na faixa de 1 a 3 GHz, consiste basicamente de um transmissor e de um receptor equipados com antenas, instaladas em torres altas com o objetivo de transmissor e receptor se "enxergarem". As condições de propagação não mudam significativamente desde que haja entre transmissor e receptor um caminho de linha reta; nestas condições o canal de rádio opera como um meio de transmissão

não-dispersivo capaz de proporcionar uma comunicação altamente confiável e de alta velocidade. Porém devido a condições meteorológicas o desempenho do canal pode cair, gerando-se o fenômeno conhecido como desvanecimento de multicaminho, que se refere ao fato da propagação entre transmissor e receptor acontecer por vários caminhos e de comprimentos diferentes. Assim, o receptor recebe uma soma de réplicas do sinal com diferentes atrasos interferindo entre si; como conseqüência o sinal recebido tem um desvanecimento: a sua amplitude muda com o tempo. Quando essas réplicas chegam em fase, elas se somam e reforçam, porém quando chegam com fase contrária, elas se cancelam.

Em geral, somado ao símbolo transmitido existem versões com atraso e defasadas: devido ao fato de estar lidando com sinais QAM é conveniente utilizar uma representação complexa para os sinais e para o canal. Neste trabalho utilizaremos uma versão simplificada do modelo de um canal de transmissão multicaminho[Koh90b].

A relação entre a entrada e saída do canal é:

$$y(t) = \int_{-\infty}^{+\infty} h_t(\tau, t).x(\tau).d\tau \tag{2.3}$$

onde: $y(t)$ é o sinal de saída do canal, $x(t)$ é sinal de entrada e $h(\tau,t)$ é a resposta ao impulso, dependente do tempo, do canal.

A expressão (2.3) pode ser simplificada tendo-se em conta a característica de variação lenta do canal multicaminho e considerando dois elementos do sinal adjacentes, a dependência com o tempo pode ser desprezada e portanto a relação anterior pode ser escrita assim:

$$y(t) = \int_{-\infty}^{+\infty} h(t-\tau).x(\tau).d\tau \tag{2.4}$$

onde: $h(\tau)$ é a resposta ao impulso, não dependente do tempo, do canal.

Quando a entrada é limitada em banda e o canal tem uma característica não dependente do tempo, a sua resposta ao impulso pode ser substituída por uma série, sem perda de informação, do ponto de vista da análise de sinais. Portanto, a resposta ao impulso pode ser escrita assim [Car86, Koh90b]:

$$h(\tau) = \sum_k a_k.\delta\left(\tau - \frac{k}{W}\right) \tag{2.5}$$

onde: δ é a função delta de Dirac, W é a largura de banda do sinal de entrada e a_k tem a expressão:

$$a_k = \frac{1}{W}.h_b\left(\frac{k}{W}\right) \tag{2.6}$$

onde: $h_b(k/W)$ é a resposta ao impulso do canal limitado à banda $(-W/2, W/2)$.

Em geral pode ser suficiente levar em conta somente dois dos termos da série da expressão 2.5 para a resposta ao impulso do canal de transmissão, para tanto deve ser fixado em um certo valor o espaçamento entre os *taps* (cada uma das derivações de um filtro digital) e o modelo é ajustado escolhendo valores apropriados para os coeficientes. Na figura 2.4 mostra-se uma representação simples do modelo dinâmico para um canal de transmissão

multicaminho variante no tempo.

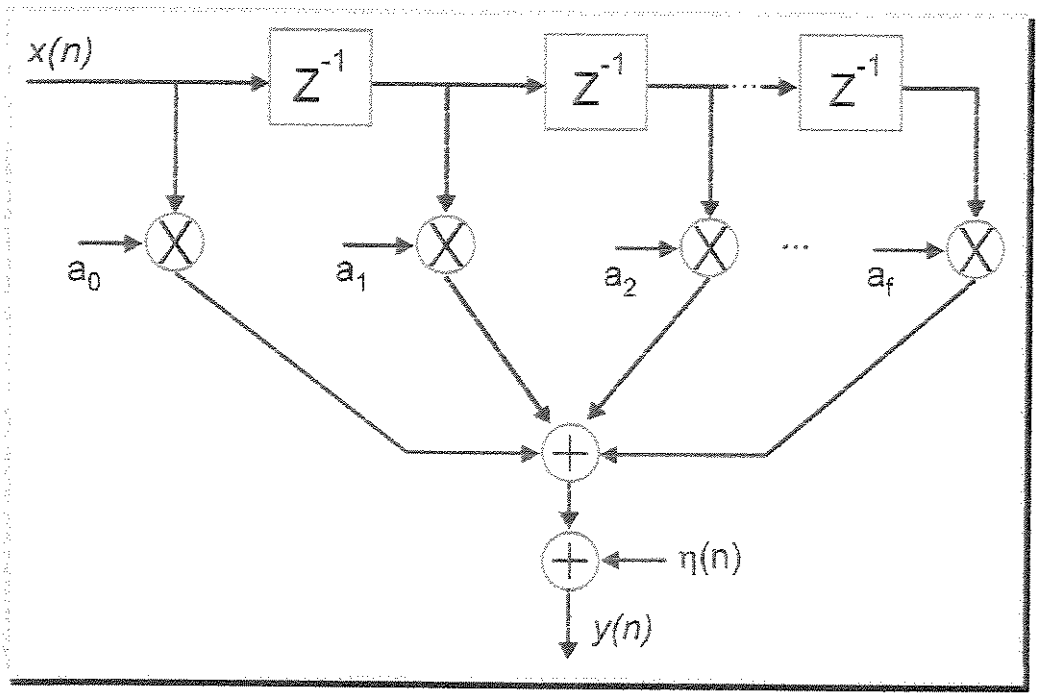


Figura 2.4. Diagrama do modelo multicaminho do canal de transmissão. *x(k)* é o sinal de entrada, *y(k)* é o sinal de saída, *η(k)* é o sinal de ruído e *f* o número de caminhos. Z é a variável da transformada Z.

2.3 As Distorções no Sistema de Transmissão

No percurso do sinal dentro do sistema de transmissão surgem vários fenômenos não desejados, por exemplo a atenuação que é indesejável devido ao fato de diminuir a potência do sinal no receptor. No entanto, existem fenômenos indesejáveis mais sérios, tais como o ruído, a interferência e a distorção, esta última afetando os níveis do sinal e portanto mudando seu formato. Embora essas perturbações aconteçam em algum ponto do sistema de transmissão, pelo método padrão espera-se que ocorram no canal de transmissão e tanto o transmissor quanto o receptor são considerados ideais.

Na figura 2.5a mostra-se o diagrama da constelação dos sinais recebidos em condições ideais representando a função de densidade de pontos-mensagem e na figura 2.5b pode-se observar a mesma constelação de sinais, porém com um certo espalhamento ao redor dos ponto-mensagem, o qual é provocado pela introdução do ruído.

O ruído é um sinal aleatório causado por processos naturais podendo ser internos ou externos ao sistema de transmissão. Quando acrescentado ao sinal mensagem seu conteúdo de informação pode ser alterado parcialmente ou ficar impossível de recuperar. O nível do ruído pode ser diminuído utilizando filtros, porém sempre permanecerá alguma parcela de ruído que não pode ser eliminada e esta parcela é uma das principais limitantes de um canal de comunicação.

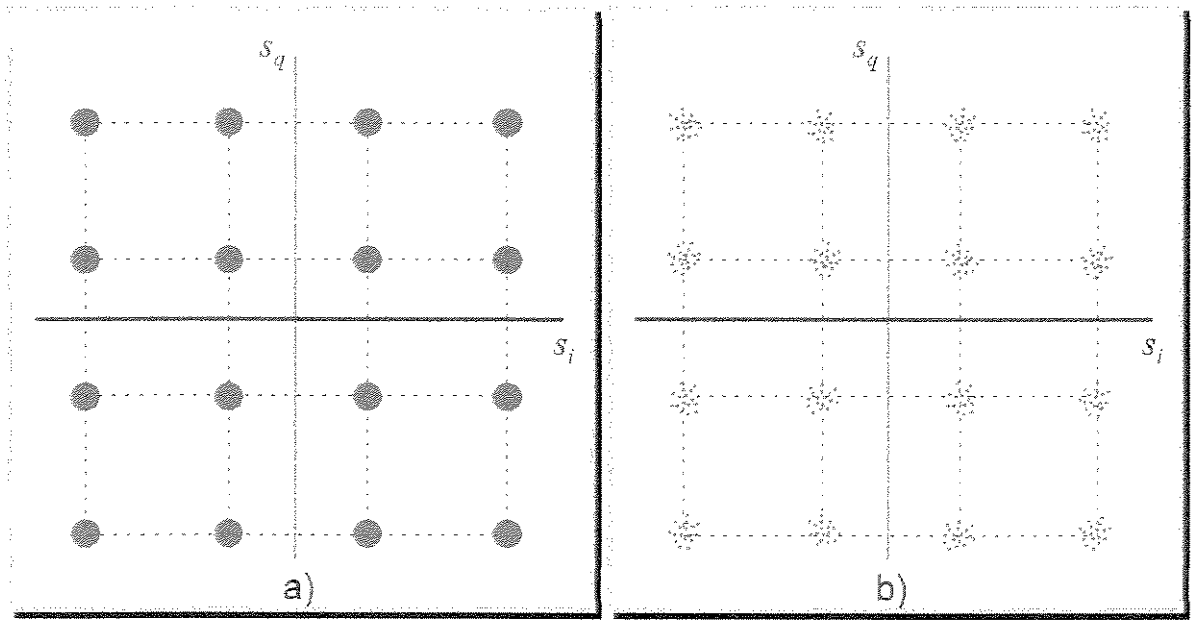


Figura 2.5. Constelação de sinais QAM. **a)** ideal e **b)** com ruído.

A relação *SNR (Signal to Noise Ratio)* caracteriza a ordem de grandeza do ruído em relação ao sinal transmitido e determina-se pela seguinte equação:

$$(SNR)_{dB} = 10.\log_{10}\frac{\sigma_x^2}{\sigma_\eta^2} \tag{2.7}$$

onde: σ_x^2 é a variância do sinal de entrada e σ_η^2 a variância do sinal de ruído.

A interferência é um fenômeno provocado por sinais originados em outros transmissores, linhas de transmissão elétrica, circuitos de comutação e outros. Comumente a interferência dá-se em sistemas de rádio onde os receptores captam o sinal de interesse interceptando vários sinais ao mesmo tempo; este fenômeno pode ser reduzido utilizando uma filtragem apropriada, contanto que o sinal interferente seja de uma banda de freqüência diferente da do sinal desejado.

Outro tipo de interferência produz-se quando transmitem-se símbolos, utilizando um determinado canal de transmissão onde a abrangência de um símbolo toma parte por um tempo maior do que o tempo normal utilizado para transmitir um símbolo. Isto provoca um sobreposição dos símbolos transmitidos em diferentes tempos, chamando-se este fenômeno de Interferência Intersímbolica *(ISI - InterSymbol Interference)* e que pode ser combatido com a utilização de equalizadores adaptativos de sinais.

Finalmente, as distorções que são perturbações das ondas de sinal, são provocadas pela resposta imperfeita do sistema para com o sinal que ao contrario do ruído e da interferência, desaparecem quando o sinal é retirado do sistema de transmissão. Normalmente as distorções são geradas por um fenômeno de mudança lenta tal como a variação de temperatura em algum circuito do transmissor ou do meio de transmissão, de forma que os

níveis do sinal transmitidos de dois símbolos consecutivos já não são bem distanciados. Em aplicações típicas, como um enlace fixo ou móvel de rádio, o canal de transmissão, além da dispersão não linear, provoca também efeitos não desejados como distorções não lineares e deslocamentos de fase. O nível destes efeitos pode variar também pela fabricação e envelhecimento dos equipamentos. Se as distorções forem lineares, podem ser corrigidas pela utilização de um equalizador linear de sinais, porém se as distorções forem não lineares a correção implica em um processo mais complexo, sendo uma opção a utilização de algoritmos de equalização não lineares[Koh90a].

Nos diagramas da figura 2.6 mostram-se alguns tipos de distorções não lineares das constelações dos sinais QAM.

2.4 Filtros Adaptativos

Wiener e Kolmogorov, na década dos 40, foram os pioneiros em dar uma solução ao problema de otimização de um filtro utilizado para a extração do ruído de um sinal com característica estatística conhecida e que não muda com o tempo (processo estacionário). O projeto de um filtro de tal natureza pressupõe uma informação *a priori* sobre a estatística do sinal e é ótimo somente quando as características estatísticas do sinal de entrada coincidem com as do filtro. Quando tais características não são conhecidas inteiramente é possível o uso de filtros adaptativos que têm tido bastante importância nos últimos anos nas áreas de telecomunicações, controle, sismologia, etc[Hay84].

Um filtro adaptativo é um dispositivo no qual os parâmetros são automaticamente ajustados com base nas estimativas das características estatísticas do sinal de entrada. A teoria de filtros adaptativos, similar à teoria de filtros ótimos, trata do problema da busca de um conjunto de parâmetros do filtro a partir do conhecimento do sinal de entrada e sob algum critério de otimização. Também o projeto destes filtros implica em achar um algoritmo de ajuste dos parâmetros do filtro em situações nas quais as características do sinal de entrada não estão ao todo disponíveis, portanto o desempenho do filtro adaptativo converge ao filtro ótimo depois de um certo número de iterações do algoritmo.

Os filtros adaptativos podem ser de configuração de laço aberto ou de laço fechado. Os primeiros consistem de dois processos, captura e aprendizado sobre a estatística do sinal e passagem dos resultados a um algoritmo não recursivo para o cálculo dos parâmetros do filtro. Este tipo de filtros têm a desvantagem de requerer de uma circuitaria muito complexa e cara. No segundo tipo de filtros a estatística do sinal de entrada não é estimada explicitamente, porém o cálculo dos parâmetros do filtro é realizado através de um algoritmo recursivo que os atualiza automaticamente com a chegada de novas amostras do sinal de entrada. Em cada iteração do algoritmo o filtro adaptativo "aprende" um pouco sobre a estatística do sinal de entrada e "melhora" os parâmetros do filtro de acordo com a nova informação captada, assim misturam-se os processos de aprendizado e cálculo num único processo e portanto a realização em *hardware* torna-se mais simples do que os filtros de laço aberto. Neste trabalho utiliza-se um filtro de laço fechado.

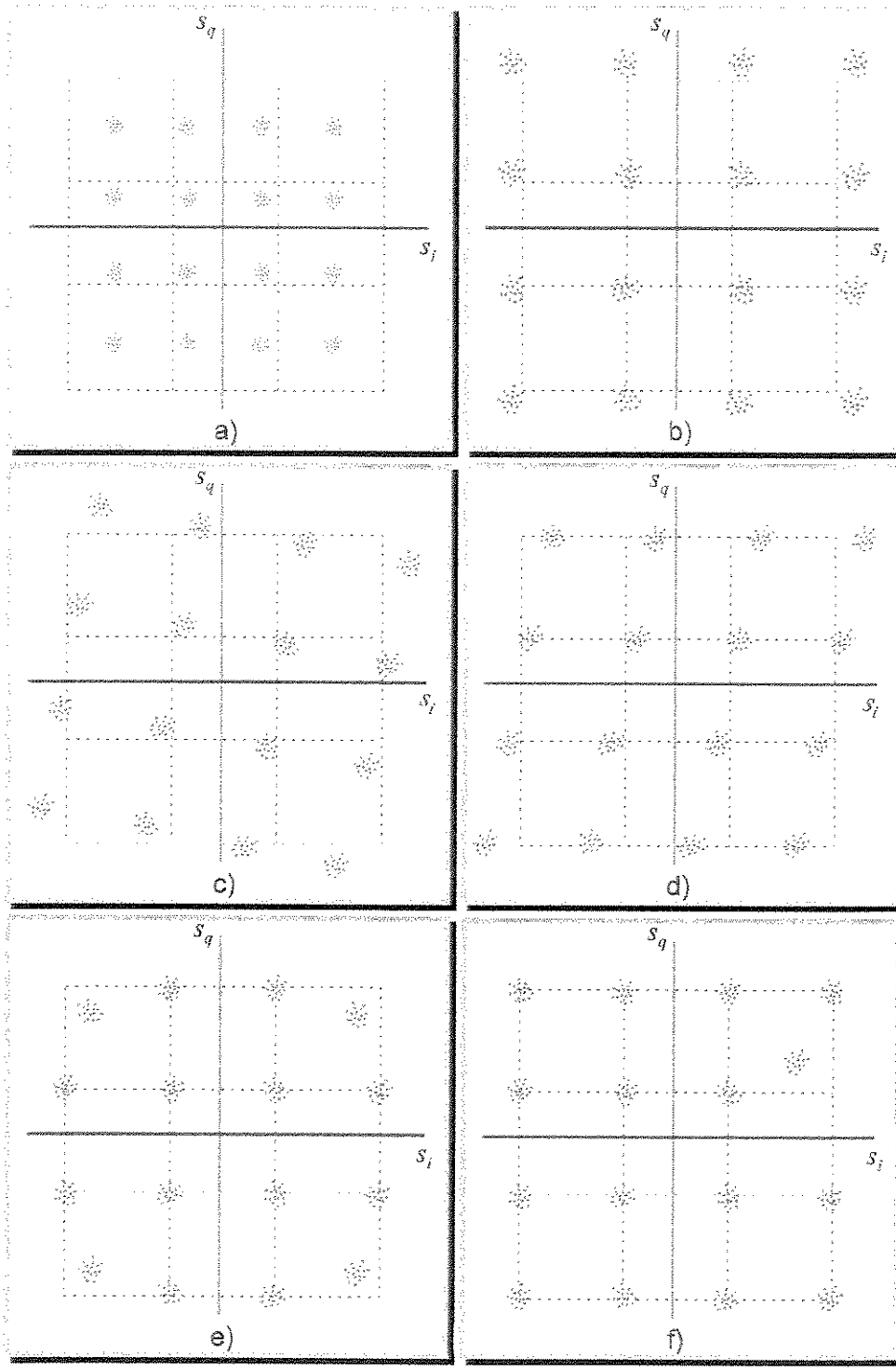


Figura 2.6 Alguns tipos de distorções não lineares da constelação QAM. **a)** *zoom-in*, **b)** *zoom-out*, **c)** rotação da grade, **d)** deslizamento da grade, **e)** colapso das esquinas e **f)** escape de um nível de quantização.

2.5 Equalização de Sinais

O objetivo da aplicação dos tipos de canal disponíveis para sistemas de transmissão de sinais digitais é a utilização eficiente destes, de forma que o projeto dos sistemas seja baseado na utilização das mais altas taxas de transmissão possíveis, porém sujeito a uma certa confiabilidade que é normalmente medida em termos de taxa de erro ou probabilidade média de erro de símbolos.

A transmissão de sinais digitais utilizando um canal de comunicação linear é limitada por dois fatores: a Interferência Intersímbolica *(ISI - InterSymbol Interferece)* e o ruído térmico aditivo. A ISI é causada pela dispersão do formato de pulso transmitido que por sua vez resulta dos desvios da característica ideal com amplitude constante e fase linear e da resposta em freqüência do canal. A ISI é também o fator mais importante no projeto de sistemas de transmissão de sinais digitais para canais limitados em banda, por exemplo o canal de telefone.

Na figura 2.1 mostra-se a seqüência *s(n)* que depende da informação binária na entrada e do tipo de modulação utilizado (QAM por exemplo). A expressão para o sinal na saída do filtro receptor em ausência de ruído pode ser escrita assim:

$$u(t) = \sum_n s(n).h_{tcr}(t - nT) \tag{2.8}$$

onde: T é o espaçamento entre símbolos e $h_{tcr}(t)$ é a resposta ao impulso equivalente ao conjunto em cascata dos blocos de filtro de transmissão, canal e filtro receptor.

Fazendo uma amostragem de $u(t)$ obtém-se:

$$u(k) = \sum_n s_n.h_{tcr}(k - n) = s_k.h_{tcr}(0) + \sum_{\substack{n \\ n \neq k}} s(n)h_{tcr}(k - n) \tag{2.9}$$

O primeiro termo da direita da equação 2.9 define o símbolo desejado no receptor e a série do segundo termo representa a Interferência Intersímbolica causada pela ação conjunta do filtro de transmissão, do canal e do filtro receptor. Cada termo desta interferência é proporcional à amostra da resposta ao impulso do canal, $h_c(t_0+iT)$, espaçado em múltiplos iT do intervalo de símbolos, T, longe do instante t_0, como se ilustra na figura 2.7. A ISI é zero unicamente se $h_c(t_0+iT)=0$, para $i \neq 0$, ou seja se a resposta ao impulso do canal tem cruzamentos no zero em intervalos espaçados em T. A ISI, se não corrigida, pode induzir uma reconstrução errônea quando da comparação dos níveis de limiar no estágio de decisão do receptor.

Para solucionar o problema da ISI requer-se um controle da resposta ao impulso h_{tcr}. Em principio para o caso ideal em que as características do canal são conhecidas inteiramente, sempre é possível projetar o par de filtros, o de transmissão e o de recepção, que faça com que a interferência intersímbolica seja pequena e ao mesmo tempo limitar o efeito do ruído aditivo do receptor minimizando a probabilidade média de erro de símbolos. Na realidade, as características dos canais não são totalmente conhecidas e o uso do par de filtros (transmissor e receptor) projetados na base de um canal com características médias pode não reduzir adequadamente a ISI. Então faz-se necessário o uso de equalizadores adaptativos os quais provêm um controle preciso da resposta temporal do canal e portanto utilizando a capacidade total do canal de transmissão.

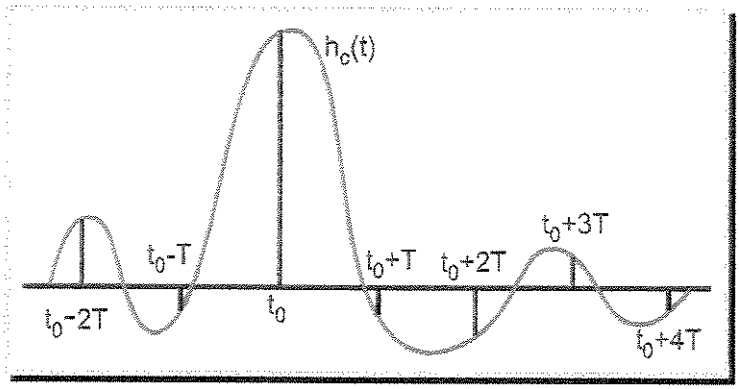


Figura 2.7 Resposta ao impulso do canal, $h_c(t)$, na presença de Interferência Intersimbolica. t_0 é o instante de amostragem

Existem duas filosofias básicas de equalização de sinais para sistemas de transmissão de sinais digitais: a pré-equalização que é realizada no transmissor e a pós-equalização realizada no receptor. No presente trabalho é utilizado um pós-equalizador, cujo diagrama pode ser observado na figura 2.8.

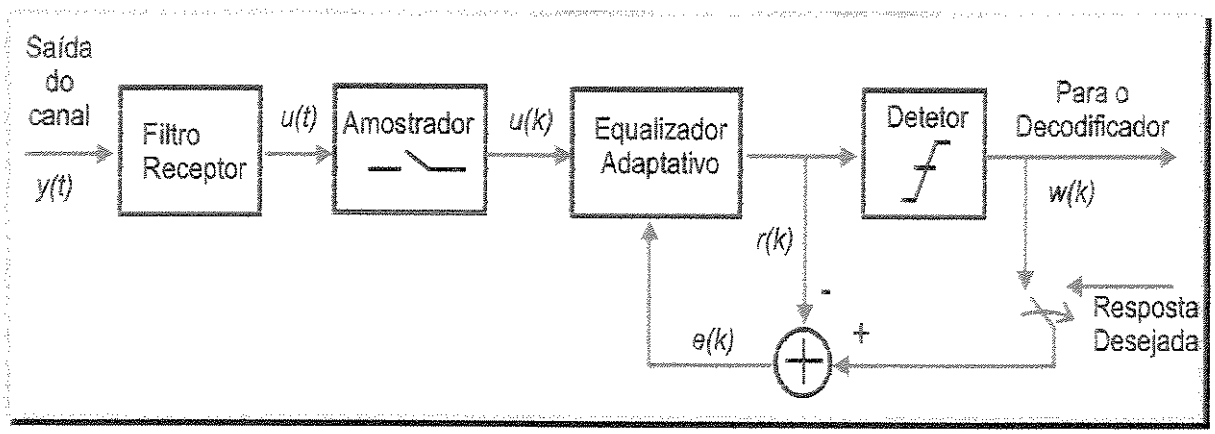


Figura 2.8. Equalizador Adaptativo (pós-equalizador) operando no modo dirigido por decisão

A operação do equalizador da figura 2.8 envolve duas fases: a de treinamento e a de ajuste. Durante a fase de treinamento transmite-se, para provar o canal, um sinal de teste conhecido, que consiste de uma seqüência de pseudo-ruído de amplo espectro e de acordo com a largura de banda do canal a ser equalizado. Sincronamente o equalizador recebe um sinal que corresponde à resposta desejada, da qual é subtraída a saída do equalizador para obter o sinal de erro, que por sua vez, é utilizado para o ajuste adaptativo dos coeficientes do equalizador. O algoritmo que é usado para esse ajuste, realiza a atualização dos coeficientes em cada período dos símbolos, partindo dos valores iniciais.

Depois da fase inicial de treinamento os coeficientes do equalizador continuam a ser atualizados porém desta vez no modo dirigido por decisão que consiste em que o sinal de erro recebe uma estimativa do dispositivo de decisão contido no módulo detetor. Em condições

normais de operação, as decisões do receptor são corretas com uma alta probabilidade, portanto a estimativa do sinal de erro é correta o suficiente de forma que permite ao equalizador adaptativo manter um ajuste apropriado dos seus coeficientes. Outro aspecto importante desse tipo de equalizador é que pode acompanhar variações lentas nas características do canal ou perturbações na entrada do receptor.

A tarefa principal de um equalizador tem sido o cancelamento de interferência intersímbolica e filtragem de ruído como visto anteriormente e nestas tarefas não interessa se o equalizador é capaz de compensar distorções não lineares ou não. Porém, a tendência é a busca de equalizadores completos que possam também compensar as distorções não lineares e os métodos adotados consistem da extensão das versões convencionais de equalizadores adaptativos ou a utilização de redes neurais artificiais.

Capítulo 3:

Os Mapas Auto-Organizados de Kohonen (SOM). A Rede Neural

As redes neurais artificiais segundo o tipo de aprendizagem podem ser classificadas em supervisionadas e não supervisionadas. Nas primeiras, a aprendizagem é controlada, em algum grau, por um supervisor que pode ser uma função de minimização de erro ou de custo e que permite condicionar a modificação dos pesos em função da resposta da rede e da comparação com a saída desejada. Exemplos deste tipo de aprendizagem são os modelos de Widrow e Hoff *(ADALINE, Adaptive Linear Element)* e o de Rosenblatt *(Perceptron)*[Wid90,Ros60,Jar97]. Um modelo importante é o estabelecido por Rumelhart, McClelland e Williams *(Multilayer Perceptron)*[Rum86] cuja regra de aprendizagem é conhecida como *backpropagation*.

Nas redes neurais com aprendizagem não supervisionada as redes formam suas próprias classificações dos dados de treinamento. Para tanto é necessário fazer duas suposições básicas sobre as redes: a primeira, uma classe é definida pelos padrões de entrada que compartilham *características* em comum; a segunda é que a rede deve ser capaz de identificar *características* comuns nos padrões que são proporcionados à entrada da rede. Os mapas auto-organizados de Kohonen[Koh89,Koh90c] *(SOM* ou *SOFM - Self-Organizing Feature Maps)* são redes deste tipo que trabalham sob estas suposições e utilizam um aprendizado não-supervisionado para modificar os pesos internos da rede e modelar as *características* achadas nos dados de treinamento.

3.1 Alguns Conceitos Preliminares

Apresentam-se a seguir alguns conceitos básicos para estabelecer uma base teórica sobre métodos determinísticos para classificação de padrões e que são úteis nas técnicas de neuro-computação.

3.1.1 Reconhecimento de Padrões

O objetivo principal do reconhecimento de padrões é a classificação. Um sistema de reconhecimento de padrões pode ser considerado como de dois estágios: um de extração de *características* e outro de classificação[Sch92]. Uma característica é a medida tomada do padrão de entrada para ser classificada, por exemplo, se se quer distinguir a letra "F" da letra "E" precisa-se comparar o número de traços verticais e horizontais. O classificador recebe uma lista das *características* medidas e sua tarefa é o mapeamento destas num estado de classificação, isto quer dizer, que dadas as *características* de entrada o classificador deve decidir qual a classe com a qual estas *características* se casam melhor. Para tanto, é necessário utilizar métricas e teoria de probabilidades.

3.1.2 Vetores de *Características* e Espaço de *Características*

Em geral é necessário ter um conjunto de padrões de entrada para poder realizar a tarefa de classificação e distinguir as diferentes categorias ou classes a que pertencem estes padrões. Chama-se de *vetor de características* [Sch92] o conjunto de *n* medidas do padrão de entrada, sendo cada uma delas uma *característica* única e a dimensão do vetor implica em um *espaço de características n*-dimensional. A figura 3.1a serve de ajuda para mostrar este conceito através de um exemplo simplista, neste gráfico observa-se a distinção entre dois grupos, os quais se diferenciam principalmente em peso e altura (classe1 e classe2). Para a construção do gráfico pode-se fazer uma série de medidas destas características que permitam classificar cada tipo de medida numa das duas classes e representa-los num plano Euclidiano bi-dimensional que definirá o *espaço de características*.

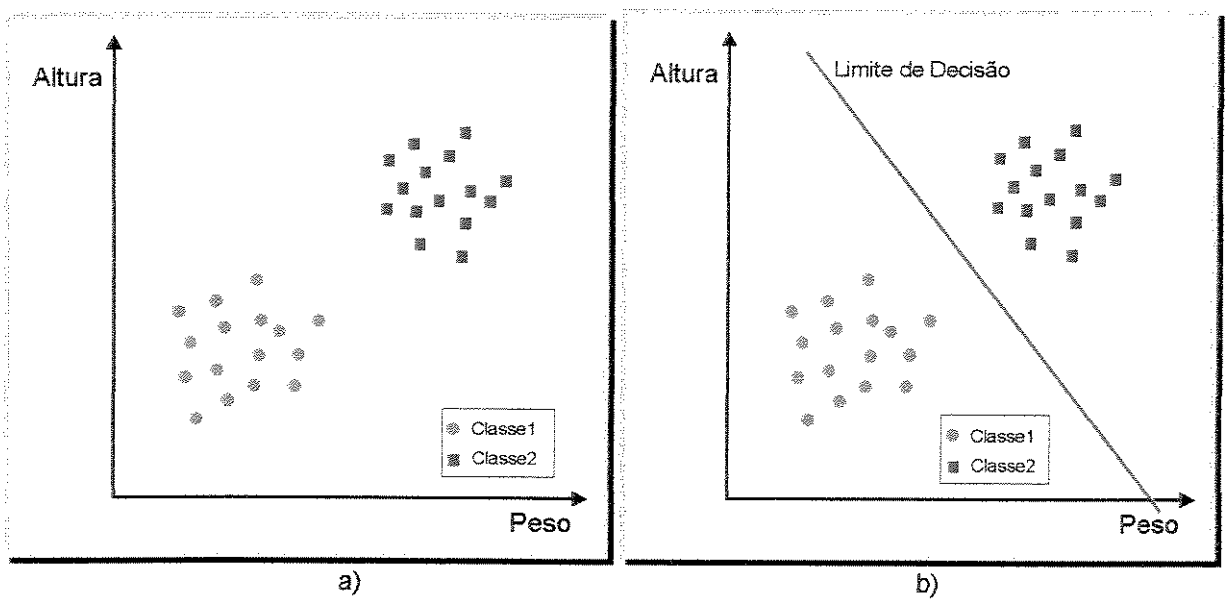


Figura 3.1. **a)** Representação de um *Espaço de Características* Euclidiano Bi-Dimensional. **b)** Uma reta mostrando o limite de decisão entre duas classes.

3.1.3 Funções de Discriminação

Observando o espalhamento das amostras na figura 3.1a podem-se distinguir duas classes *(clusters)*. A tarefa do classificador como já mencionado é a de atribuir a uma destas duas classes, os padrões novos de entrada. Intuitivamente pode-se imaginar uma linha separando arbitrariamente ambas as classes, fazendo isto a tarefa de classificação se limitaria à decisão do lado ao qual o padrão de entrada pertenceria (vide figura 3.1b).

Este limite de decisão, cuja definição matemática e a de *função de discriminação*[Sch92, Bez92, Koh89], é básica para as técnicas de reconhecimento de padrões e servem para mapear as características de entrada no espaço de classificação. No exemplo da figura 3.1b, pode-se observar que há uma infinidade de limites de decisão que poderiam se estabelecer, porém quanto mais simples esta função melhor para as implementações sejam em *hardware* ou *software*.

3.1.4 Técnicas de Classificação

Dentre as duas amplas categorias de técnicas de classificação têm-se as numéricas e as não numéricas. As primeiras (classificação linear, *k* vizinhos mais próximos) incluem medidas determinísticas e estatísticas que podem ser consideradas como medidas num espaço de padrões geométricos. As técnicas não numéricas (classificação Bayesiana, a qual usa estimativas probabilisticas) recaem dentro do domínio do processamento simbólico e podem implicar em técnicas como a de *conjuntos nebulosos (fuzzy)*[Bez92].

3.1.5 Classificação Vizinho Mais Próximo

Na figura 3.2a estão representadas duas classes no espaço de padrões, dado um padrão novo de entrada não classificado, *x*, a tarefa é decidir a qual das duas classes este pertence. As técnicas de vizinho mais próximo realizam esta decisão baseados na distância mais curta do padrão não classificado às amostras da periferia das classes, que no caso do exemplo a função de discriminação é a seguinte:

$$f(x) = mais_próximo(x, classe1) - mais_próximo(x, classe2) \qquad (3.1)$$

Para um espaço de padrões com classes bem diferenciadas como no exemplo da figura 3.2a a função *f(x)* poderá assumir um valor negativo para membros da classe 1 e positivo para membros da classe 2 ou viceversa, porém devido a um desempenho pobre o uso deste tipo de funções pode ser bastante restrito

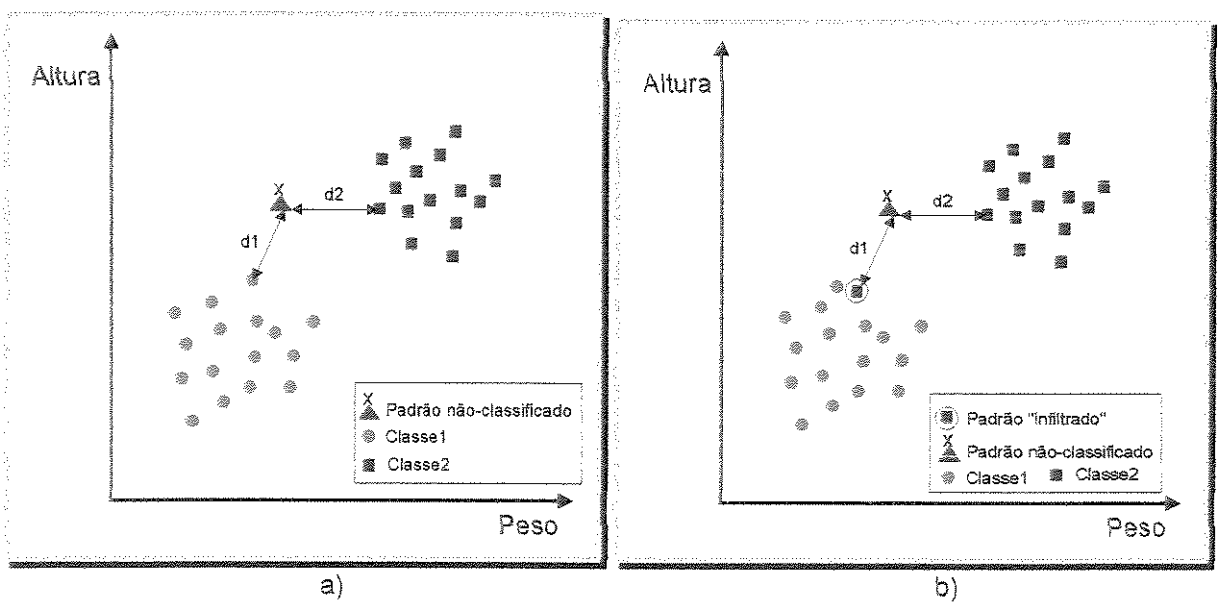


Figura 3.2 a) Classificação pelo método de Vizinho Mais Próximo. **b)** Erro produzido no método de Vizinho Mais Próximo se numa das classes existe um padrão "infiltrado". *d1* e *d2* são as distâncias às classes 1 e 2 respectivamente.

A técnica de classificação *k* vizinhos mais próximos[Sch92] consiste na obtenção da

distância média do padrão não classificado a um número k de padrões mais próximos de uma determinada classe. Esta técnica resolve o problema de no caso de um padrão "infiltrado" estar na periferia de uma classe vizinha à qual não corresponde, de forma que a função de discriminação quando avaliada para um padrão não classificado (que se encontre mais perto dessa classe) guie para uma falsa classificação, isto é representado graficamente na figura 3.2b.

3.1.6 Distâncias

As técnicas de classificação vizinho mais próximo precisam de uma forma confiável de medida da distância do padrão de entrada para as classes. A seguir mostram-se alguns dos métodos mais utilizados[Koh89].

A Distância Euclidiana, é a métrica mais comumente utilizada em algoritmos de aprendizado para redes neurais. Dados dois vetores, a Distância Euclidiana consiste no cálculo da raiz quadrada da somatória das diferenças entre as componentes correspondentes dos vetores ao quadrado, cuja equação é:

$$D_{Euclideana}(\mathbf{X},\mathbf{Y}) = \sqrt{\left(\sum_{n}(x_i - y_i)^2\right)} \tag{3.2}$$

onde x_i e y_i são os i-ésimos elementos do vetores n-dimensionais ***X*** e ***Y***, respectivamente.

A Distância *Manhattan* (Rio Claro), é uma versão simples da Distância Euclidiana que evita o cálculo da raiz quadrada e da potenciação, trazendo o benefício de maior rapidez no cômputo e implementação mais simples. A equação a seguir corresponde ao cálculo da somatória dos valores absolutos das diferenças entre as componentes respectivas de dois vetores:

$$D_{Manhattan}(\mathbf{X},\mathbf{Y}) = \sum_{n}|x_i - y_i| \tag{3.3}$$

onde x_i e y_i são os i-ésimos elementos dos vetores n-dimensionais ***X*** e ***Y***, respectivamente.

Para a Distância *Manhattan* os vetores que estejam no limite quadrado que é definido pela equação (3.3) são equidistantes ao vetor localizado no centro deste quadrado, diferente da Distância Euclidiana no qual esse limite é uma circunferência. Este fato traz como conseqüência a introdução de erros no cálculo da distância, no entanto é necessário se avaliar o compromisso entre precisão e rapidez de cálculo.

A Distância *Hamming*, é um caso particular da Distância Manhattan. Para o caso de vetores binários, quando é necessário saber o número de bits diferentes entre dois vetores, a sua realização é simples e pode ser implementada pela função OR-exclusivo devido a que:

$|x_i - y_i|$ é equivalente a x_i *XOR* y_i

3.2 O Conceito de Auto-Organização

O Conceito de Auto-Organização foi proposto pela primeira vez, sob o ponto de vista biológico, por von der Malsburg no ano de 1973 e cuja modelagem computacional foi

Sciences da *University of Helsinki*, inspirado nas idéias de van der Malsburg e Willshaw apresentou o modelo conhecido como *Self-Organizing Feature Maps* (SOFM), o qual tem se mostrado adequado para a solução de problemas de reconhecimento de padrões e de classificação[Koh90c].

Os organismos vivos constituem a forma mais excepcional de auto-organização e todos eles devem ter a capacidade de classificar os estímulos apresentados pelo meio ambiente, a fim de permitir-lhes a sua própria sobrevivência. Os organismos mais simples realizam procedimentos de classificação predeterminados, enquanto outros mais sofisticados organizam e classificam os estímulos de entrada, se adaptando notavelmente às mudanças e ao aumento de complexidade das características do seu ambiente. Esta capacidade foi definida como Percepção Aprendida por B. Farley[Far60], quem concebeu um modelo para o processo de classificação como um sistema de processamento de dados com a capacidade de computar medidas a partir das propriedades de algumas entradas apresentadas durante um certo período de tempo. Estas propriedades formam características físicas como cor, temperatura, peso, etc. ou relações entre as mesmas (exemplo, na percepção auditiva são consideradas como propriedades mensuráveis a freqüência e a intensidade dos sinais acústicos).

Dado que cada objeto pode ser definido em função de uma relação das propriedades associadas, se lhe atribui também o número de vezes que a propriedade é observada, i.e., a sua distribuição de probabilidade no espaço observado. Esta característica permite a formação de classes para cada grupo de objetos com o conjunto de propriedades similares, com a vantagem que pode ser distinguido um vasto número de classes, através do uso de poucas propriedades.

Um procedimento que permita organizar estas classes, essência do processo auto-organizável, seria o de criar conjuntos de regras de agrupamento baseadas na freqüência de ocorrência da propriedade e da sua continuidade no tempo, entre outros fatores. Desta forma um sistema auto-organizável, quando exposto a um número razoável de entradas vindas do seu ambiente, seja capaz de gradual e automaticamente classificar adequadamente os padrões de entrada. Além do mais, o sistema deve poder eficientemente reconhecer um padrão desconhecido que mostre suficiente correlação com algum dos já armazenados e classificados, quando submetido ao sistema.

A natureza exata das regras de agrupamento e classificação são desconhecidas e somente são evidentes a partir da observação das características biológicas e o comportamento de sistemas auto-organizáveis já existentes na natureza.

O cérebro usa um mapeamento espacial para modelar internamente estruturas complexas de dados e a proposta de T. Kohonen[Koh89,Koh90c] aproveita esta idéia de forma que a sua rede realiza compressão de dados nos vetores de entrada usando uma técnica chamada de *quantização vetorial* e a rede armazena os dados de forma tal que as relações espacial e topológica dos dados de treinamento são mantidas e representadas numa forma que conserva o significado completo.

Uma compressão de dados significa que dados multi-dimensionais podem ser representados num espaço de dimensão menor. Grande parte do córtex cerebral é ordenado como um plano bi-dimensional de neurônios interconectados, no entanto são capazes de lidar com conceitos de uma maior dimensão. A implementação do algoritmo de Kohonen é predominantemente bi-dimensional e um exemplo típico tem-se na figura 3.3 onde a rede mostrada é uma rede bi-dimensional de uma camada. A diferença mais notória e óbvia que se percebe é que os neurônios não estão dispostos em camadas como na rede de Perceptron

multi-camada[Rum86] (entrada, oculta e saída) e sim numa grade ampla. Todas as entradas estão ligadas a todos as células neurônio (nós) na rede; a realimentação está restrita às interconexões laterais das células vizinhas imediatas; e, não há uma camada separada para a saída, cada célula neurônio na grade é por si mesma um nó de saída[Lip87].

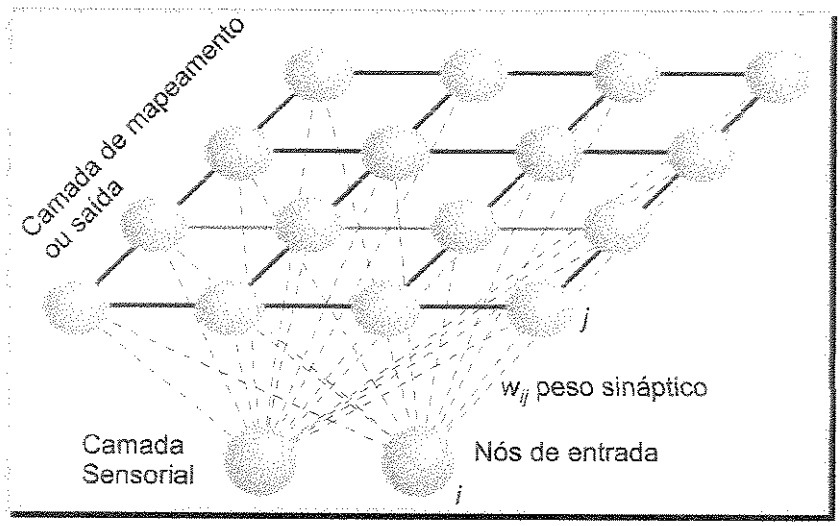


Figura 3.3 Mapa de Características de Kohonen (SOFM). Os nós de entradas estão conectados a todos os nós da única camada de neurônios.

3.3 O Algoritmo de Kohonen

O algoritmo de aprendizagem organiza os nós que estão na grade em vizinhanças locais que agem como classificadores de características sobre os dados de entrada. O mapa topográfico se organiza autonomamente por meio de um processo cíclico de comparações entre os padrões de entrada e os vetores de peso armazenados em cada nó. A resposta de treinamento da rede não é especificada para a entrada de treinamento e quando uma entrada tem *casamento* com o vetor de algum nó, a área em torno do nó é otimizada seletivamente para representar, na classe à qual o nó pertence, uma média dos dados de treinamento. Partindo-se de um ordenamento arbitrário dos nós da rede, esta conflui num mapa de características que tem uma representação local e é auto-organizado.

A seguir apresenta-se o algoritmo, cuja noção é simples[Lip87].

Passo 1 Iniciar a Rede.

Definem-se os

$$w_{ij}(t)$$

para serem os pesos sinápticos da entrada *i* ao nó *j* no instante de tempo *t* (vide figura 3.3).

onde: $0 \le i \le n-1$,

Iniciar os pesos, das *n* entradas aos nós, com valores aleatórios preferencialmente pequenos.

Assumir um raio de vizinhança grande ao redor do nó *j*, $N_j(0)$.

Passo 2 Apresentar a Entrada
Apresentar a entrada $u_0(t)$, $u_1(t)$, $u_2(t)$,..., $u_{n-1}(t)$, onde $u_i(t)$ é a entrada ao nó i no instante de tempo t.

Passo 3 Cálculo da Distância
Calcular a distância d_j entre a entrada e cada um dos nós de saída j, utilizando para tanto uma métrica, por exemplo calcular a distância euclidiana:

$$d_j = \sum_{i=0}^{n-1} \left(u_i(t) - w_{ij}(t)\right)^2$$

Passo 4 Selecionar a Distância Mínima
Designar o nó de saída com a menor distancia d_j como o nó vencedor j^*.

Passo 5 Atualizar os Pesos
Atualizar os pesos do nó vencedor, j^*, e seus vizinhos topológicos definidos por $N_{j^*}(t)$. Os novos pesos se calculam assim:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)\left(u_i(t) - w_{ij}(t)\right) \qquad (3.4)$$

Para j em $N_{j^*}(t)$, onde $0 \le i \le n-1$
O termo $\alpha(t)$ é o ganho, com variação entre 0 e 1, que decresce no tempo fazendo com que a adaptação dos pesos seja lenta. Também o a vizinhança se reduz em tamanho conforme a variável tempo se incrementa, gerando uma localização de máxima atividade.

Passo 6 Voltar ao passo 2.

Em resumo pode-se dizer do algoritmo de aprendizagem de Kohonen que primeiro é necessário achar a célula neurônio que melhor se casa com o padrão de entrada e segundo que dito nó e seus vizinhos topológicos devem incrementar a sua similaridade com a entrada.

Inicialmente todos os pesos internos (os pesos dos nós de entrada aos nós da rede) recebem valores aleatórios pequenos e cada nó portanto têm um único vetor de pesos cuja dimensionalidade depende do número de elementos dos vetores de entrada. Durante a fase de aprendizado, são apresentados à rede um conjunto representativo do conjunto completo de dados. Então realiza-se uma comparação entre os vetores de peso com cada vetor ou padrão de entrada, determinando-se a célula com vetor de pesos mais próximo ao padrão de entrada e denominando-o como célula vencedora. Este processo competitivo chama-se de *Winner-Takes-All (WTA)*. A célula vencedora toma o vetor de entrada e modifica seu vetor de pesos interno para se adequar à entrada tornando-se sensível ao padrão de entrada particular que proporcionará uma resposta máxima se o padrão for apresentado novamente.

Do algoritmo pode-se também observar que os pesos dos nós na vizinhança N_c do nó vencedor são também modificados, a razão para tal mudança é a tentativa da rede gerar regiões que respondam a valores espalhados em torno do padrão de entrada. Os nós em torno do nó vencedor têm portanto um *alinhamento* similar e ao final da fase de treinamento convergem para uma representação média de uma classe-padrão. Como conseqüência os

vetores que estejam espacialmente próximos aos vetores de treinamento serão classificados corretamente ainda que a rede não tenha um conhecimento prévio deles, demonstrando assim a generalização de propriedades que a rede proporciona.

Os dois principais aportes da rede neural de Kohonen para o aprendizado adaptativo auto-organizado são o processo de adaptação de pesos e o conceito de vizinhança topológica dos nós, idéias que são diferentes de outros tipos de redes neurais.

3.3.1 Justificativa Biológica

T. Kohonen[Koh89, Koh90c] baseou seu trabalho nos estudos da topologia da região do córtex cerebral e há uma boa evidência que suporta a sua teoria. Considerando que a ativação numa célula nervosa é propagada a outras células por meio de enlaces chamados de axônios, que têm um efeito de inibição ou de excitação à entrada de outra célula, no entanto é necessário considerar a questão de como esses enlaces são influenciados pela distância lateral ao neurônio transmissor. Na figura 3.4 mostra-se uma ilustração simplificada do modelo deste efeito através da função Chapéu Mexicano *(Mexican Hat)*.

Na figura 3.4 pode-se observar que as células fisicamente próximas à célula ativa têm o enlace mais forte no entanto que outras recaem numa região de inibição sendo este o fenômeno ao qual Kohonen atribui o desenvolvimento de mapas no cérebro localizados topologicamente. O modelo por ele utilizado considerou este efeito usando somente redes interconectadas localmente e restringindo a adaptação dos pesos sinápticos aos vizinhos locais.

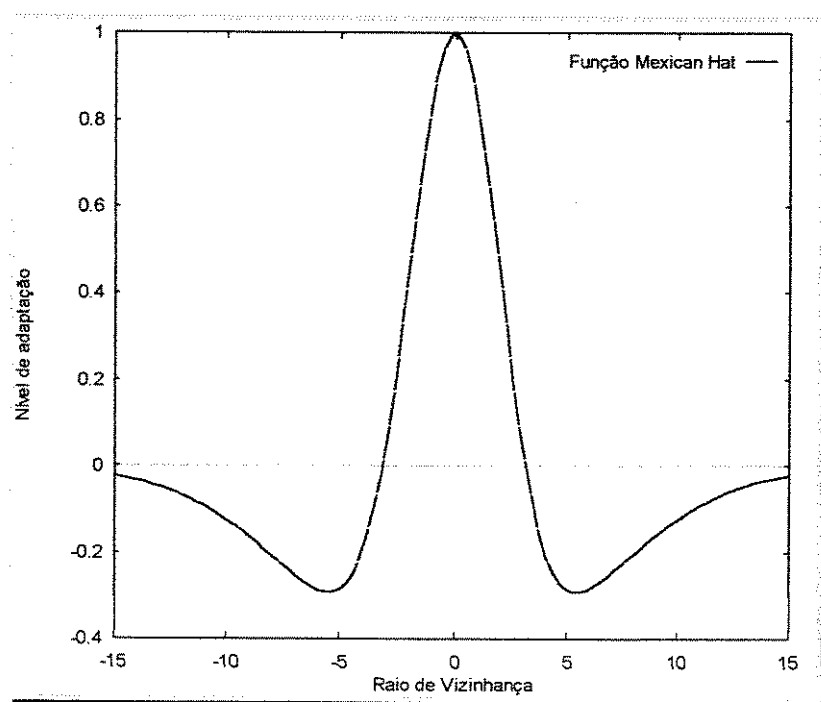


Figura 3.4. Efeito da interconexão lateral mostrado através da funcão Chapéu Mexicano.

3.4 Atualização dos Pesos Internos

Referindo-se à equação (3.4) observa-se que os novos valores dos pesos internos são proporcionais à diferença entre o vetor de entrada e o vetor de pesos. A unidade de proporcionalidade é $\alpha(t)$, conhecida como coeficiente de adaptação ou de aprendizado ou simplesmente ganho[Koh90c], faz com que a velocidade de adaptação diminua com o tempo (os passos da seqüência de treinamento). No processo de treinamento da rede podem ser visualizadas duas etapas. Na primeira, os nós que inicialmente foram ordenados aleatoriamente formam uma espécie de ordem topológica (mapas), tentando aglomerar os nós em classes *(clusters)* e refletir a variedade de classes contidas nos padrões de treinamento. Tem-se desta maneira um mapeamento grosseiro onde a rede encontra-se numa fase de descobrimento das classes em número e tipo, sendo as mudanças de orientação dos nós nos mapas, em grande escala, com valores de adaptação elevados (α>0.5) e grandes variações nos valores dos pesos internos em rápida elaboração do mapeamento. Na segunda etapa, uma vez estabelecido o mapeamento grosso, os nós, já localizados em mapas, sofrem um ajuste fino de adaptação aos padrões de entrada por meio de pequenas mudanças nos vetores de peso em cada nó e o fator de adaptação é reduzido conforme o treinamento progride. Tipicamente a segunda etapa tem lugar na faixa de passos entre 100 e 1000.

O recomendável é que a diminuição de $\alpha(t)$ seja gradual conforme a formação dos mapas, garantindo desta maneira uma representação mais precisa dos padrões de entrada, assim como a convergência da rede a uma solução dentro de um limite de tempo pré-definido. Em aplicações típicas, Kohonen recomenda que a velocidade de diminuição seja uma função linear do número de passos do conjunto de padrões de treinamento.

3.4.1 Valores Iniciais dos Pesos Internos

Ainda que sugerido que os valores dos pesos iniciais deveriam ser aleatórios e pequenos, se estes valores forem real e aleatoriamente distribuídos, a rede pode não convergir e precisar de ciclos de treinamento muito longos. A razão é devido aos vetores de treinamento recairem em classes sobre uma região limitada do espaço de padrões e se os vetores de pesos, armazenados nos nós da rede, estiverem aleatóriamente distribuídos então muitos dos vetores estariam numa orientação diferente da maioria dos padrões de entrada, portanto esses nós não ganhariam nenhum dos processos de comparação de melhor casamento e permaneceriam sem uso para formação de mapas topológicos. Como conseqüência as vizinhanças no mapa de características seriam pobremente povoadas de nós, resultando num desempenho de classificação muito pobre devido a incapacidade dos mapas distinguirem as entradas umas das outras.

A distribuição ótima de valores iniciais para os vetores de pesos é aquela que dá à rede aproximações iniciais, para as classes dos padrões de treinamento, com uma orientação pre-estabelecida[Koh96]. Isto somente é possível para algumas aplicações, pois o objetivo é que a rede encontre por si só essas agrupações, portanto essa proposta é pouco prática.

Um dos métodos de aproximação mais prático consiste em iniciar todos os pesos internos de forma que eles sejam normais e coincidentes, i.e., com o mesmo valor. O ordenamento dos vetores de treinamento é modificado de forma que nas etapas iniciais todos os vetores com orientação similar estejam agrupados, assim todos os nós no mapa de características terão uma orientação aproximada à dos vetores de entrada. Conforme o

treinamento progride as entradas voltam a ter lentamente a sua orientação original e como já tem-se definido um mapeamento aproximado, os nós simplesmente seguirão as mudanças dos padrões de entrada[Koh89, Koh90c].

3.5 Vizinhança Topológica

A técnica mais utilizada de aproximação dos valores iniciais dos pesos internos e indicada pelo próprio Kohonen é a baseada nas vizinhanças locais em torno de cada nó (vizinhanças topológicas[Koh89]) modeladas pela função Chapéu Mexicano, que define uma fronteira dinâmica que indica quantos dos nós que circundam o nó ou célula vencedora terão seus pesos internos modificados durante o processo de treinamento. Inicialmente cada nó da rede poderia ter uma vizinhança grande, tanto quanto o tamanho da rede, e quando selecionado como o vetor mais próximo do padrão de entrada este terá seus pesos atualizados, assim como todos os nós na sua vizinhança em forma similar. O tamanho da vizinhança diminui lentamente conforme os vetores de entrada são apresentados à rede até chegar a um limite pré-determinado. Para facilitar a compreensão desta idéia nas ilustrações da figura 3.5 observa-se como os nós de uma determinada classe são ajustados. Na figura 3.5a a rede mostra os vetores de pesos com valores iniciais aleatórios e uma vizinhança grande entorno de cada nó. As setas desenhadas dentro de cada círculo representando um nó, indicam uma representação espacial da orientação do vetor de pesos de cada nó. Na figura 3.5b pode-se ver a rede após muitos vetores de entrada terem sido apresentados e a região delimitada pelo quadrado mostra o inicio da formação de uma classe específica baseada na célula em destaque. O tamanho da vizinhança foi reduzido de forma que o número de nós que sofreram mudança nos seus pesos internos foi menor. Na figura 3.5c tem-se a rede no final do treinamento, a vizinhança ficou delimitada por uma célula com melhor resposta e 8 vizinhas topológicas representando uma média de valores de pesos distribuídos na classe.

É necessário notar que a rede forma as classes sem supervisão, porém a nomeação das mesmas deve ser feita separadamente, ao final do treinamento completo da rede.

3.5.1 Redução da Vizinhança

Foi mencionado que a vizinhança em torno da célula vencedora deve diminuir conforme os padrões de entrada forem apresentados à rede. No entanto, não existem regras estabelecidas que determinem com que velocidade esta deve ser reduzida, sendo necessário realizar experiências para um dado caso particular[Koh89, Koh90c].

O treinamento além de ser afetado pelas velocidades de adaptação e de redução da vizinhança, também é afetado pelo formato dos limites da vizinhança, podendo ser este quadrado, hexagonal ou mesmo circular, sendo sempre preferível começar com uma vizinhança grande e decrescer lentamente conforme os vetores de entrada ingressam à rede. Na figura 3.6 observa-se que a vizinhança $N_{j^*}(t)$ em torno da célula vencedora j^* é quadrada e diminui monotonicamente em função do tempo (passos de treinamento).

3.5.2 Funções de Densidade de Probabilidade

O fenômeno de formação de classes *(clusters)* durante a etapa de treinamento pode-se explicar também pelo uso da função de densidade de probabilidade *(fdp)*, a qual

descreve a distribuição dos padrões de entrada. Para um dado ponto no espaço de padrões a *fdp* define um valor para a semelhança do vetor de entrada com o ponto em questão. Dado um espaço de padrões com uma *fdp* conhecida (quer dizer, é sabido como os padrões estão distribuidos no espaço de padrões), é possivel mostrar que o mapa assumirá uma ordem tal que a densidade pontos dos nós tenderá a se aproximar da *fdp* do espaço de padrões. Isto é esquematizado na figura 3.7, onde é mostrada uma seqüência de diagramas representando os estados dos pesos internos para vários estágios do ciclo de treinamento [Jar97].

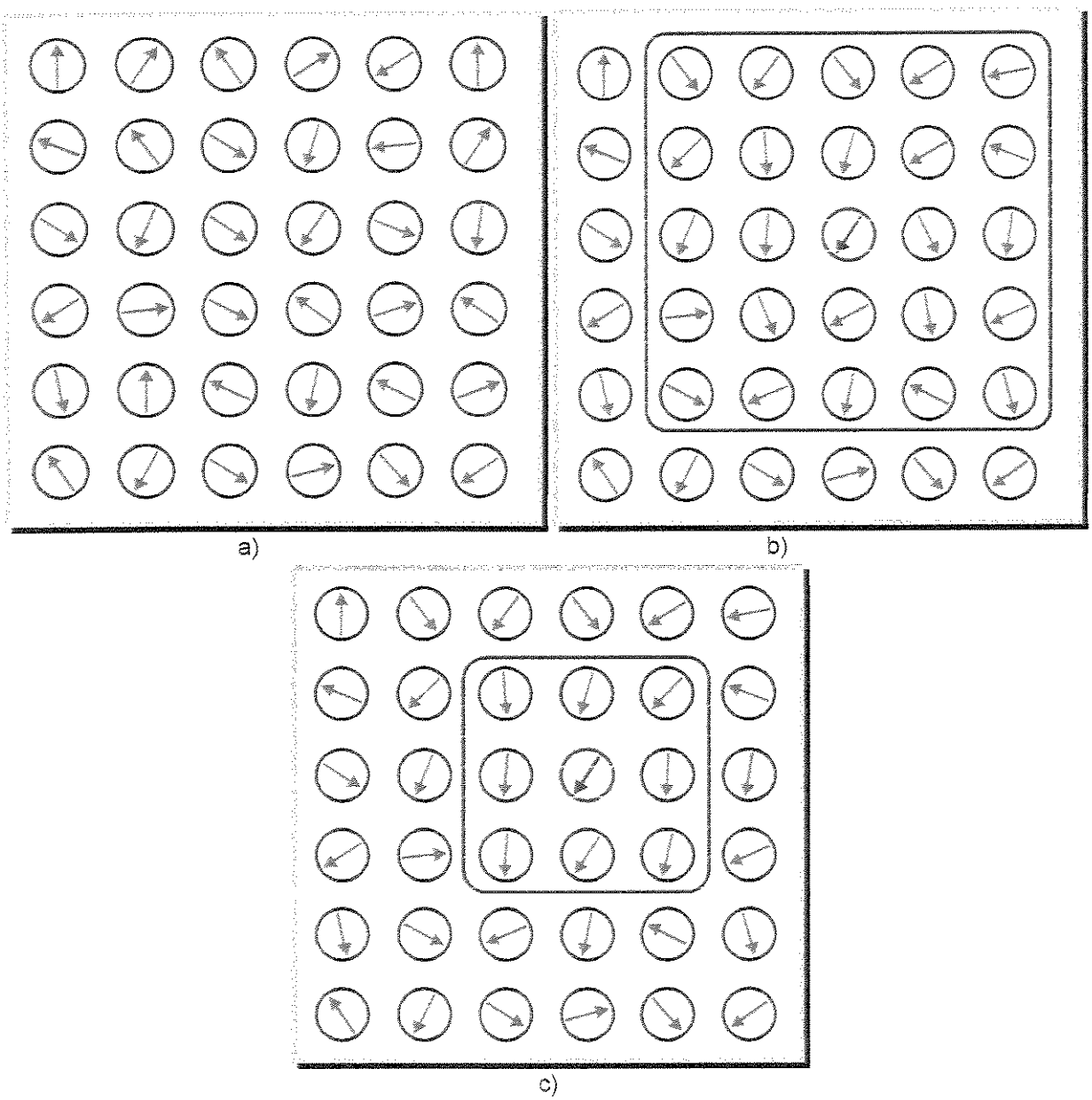


Figura 3.5 Treinamento numa vizinhança topologica. **a)** Valores iniciais aleatórios, **b)** formação de uma classe, **c)** mapa final para uma determinada classe.

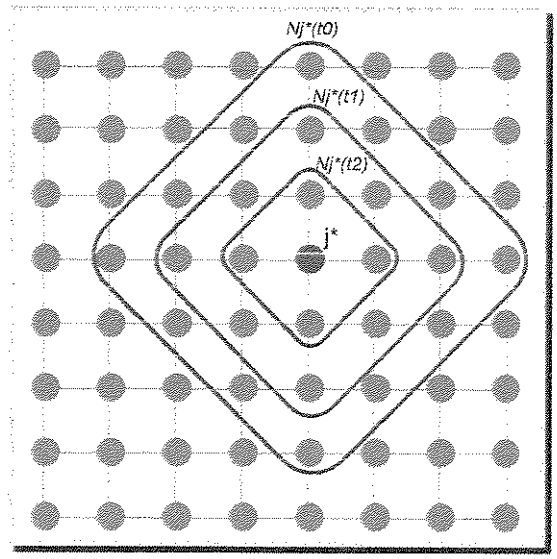


Figura 3.6 Vizinhança topológica $N_{j^*}(t)$ quadrada entorno da célula vencedora j^*. $t_0<t_1<t_2$.

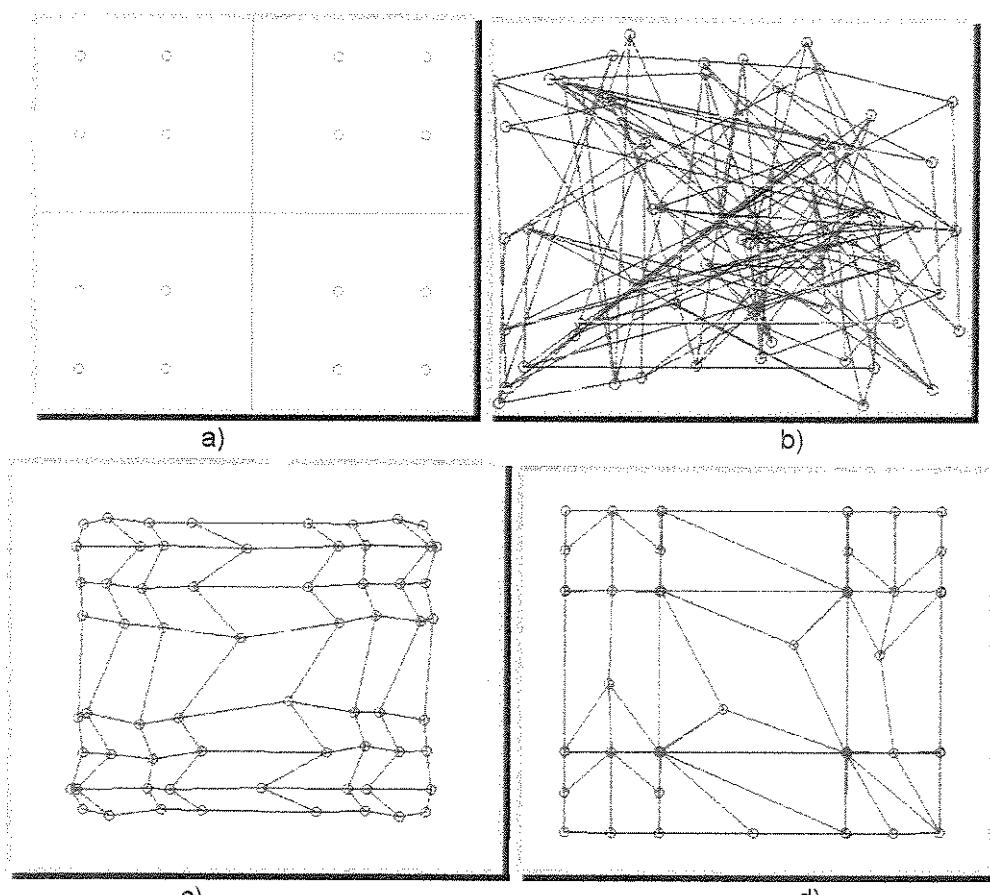


Figura 3.7 Pesos internos da rede. **a)** *fdp* dos padrões de entrada, **b)** *fdp* inicial dos pesos internos, **c)** *fdp* após várias centenas dos passos de treinamento, **d)** *fdp* ao final do treinamento

Capítulo 4:

Equalização Adaptativa de Sinais Baseada em Redes Neurais

A indústria crescente e de constantes mudanças das telecomunicações tem se mostrado aberta à aplicação dos filtros adaptativos, os quais têm diversas aplicações práticas nos vários campos de utilização das técnicas de processamento de sinais[Cow85]. Atualmente, a importância da filtragem adaptativa cresce de acordo com o desenvolvimento da tecnologia de microcomponentes tornando possível a implementação dos algoritmos de correção cada vez mais complexos. O campo de aplicação mais importante para a filtragem adaptativa junto com as técnicas de cancelamento de eco e de codificação de voz, é nos sistemas de comunicação[Jon95].

Um filtro adaptativo que é utilizado num sistema de comunicação para o tratamento de sinais é também conhecido como equalizador de sinais. Um equalizador se colocado no caminho do sinal tem como um dos seus propósitos a diminuição da interferência intersimbólica *(ISI - Inter Symbol Interference)*, de forma a maximizar a probabilidade de decisões corretas[Qur85]. Um segundo propósito é a compensação de não-linearidades.

A estrutura mais comum de filtro adaptativo é a do filtro transversal ou FIR *(Finite Impulse Response)* com coeficientes ajustáveis. Os filtros adaptativos podem ser dedicados à compensação de distorções não-lineares como o Equalizador com Decisão Realimentada *(DFE - Decision Feedback Equalizer)*, apesar de não ser um equalizador não linear completo, pois compensam um conjunto restrito de não linearidades. Um equalizador baseado em redes neurais é capaz de corrigir um variedade mais ampla de não linearidades. Uma vantagem importante do DFE é a sua capacidade de compensar a ISI de forma melhor do que o filtro transversal convencional.

Existem vários algoritmos para a adaptação dos coeficientes do equalizador, sendo o mais comum o algoritmo de gradiente conhecido como *LMS (Least Mean Square)*[Cow85, Bel87]. Este algoritmo mostrou ser também robusto.

Talvez uma das razões para a utilização de redes neurais artificiais seja simplesmente a curiosidade que sempre impulsionou a ciência. No entanto, trabalhos pioneiros como o de Siu et al, 1990, mostram que equalizadores baseados na rede *Multilayer Perceptron*[Lip87, Wid88] *(MLP)* foram superiores aos equalizadores convencionais em termos da medida de desempenho da taxa de erro de bits *(BER - Bit Error Rate)*, e também que há algumas dificuldades na aplicação dos MLP, como tempos muito longos de treinamento e falta de uma metodologia de seleção da arquitetura[Siu90]. Uma segunda razão para se utilizar redes neurais nos equalizadores adaptativos é a de engenharia, dado que uma determinada solução implica em complexidade computacional e medidas de desempenho. É nesse sentido que a solução proposta por Kohonen et al, 1990 resulta ser interessante pois tendo-se uma estrutura de equalizador mais simplificada consegue-se manter o mesmo desempenho e complexidade computacional[Koh90a, Koh90b].

4.1 O Equalizador Linear ou Equalizador Transversal Linear (LTE)

Até inícios da década de 60, para a transmissão de dados via canais telefônicos, usava-se a equalização por meio de um filtro receptor fixo ou ajustável para a compensação das características não-ideais deste tipo de canal. Com o aumento das taxas de transmissão parecia que essa estrutura de equalização não mais serviria para tal propósito, não tendo-se sucesso na retirada das distorções introduzidas pelas variações no sistema de comutação já que o canal para uma determinada conexão diferia bastante do comportamento médio para o qual o filtro tinha sido projetado. A compensação para o canal de características médio podia ser adequada para determinadas taxas de transmissão (ex. 2400 bits/s), no entanto para altas taxas essa solução não servia mais, surgindo a necessidade de se projetar equalizadores que fossem capazes de se ajustar por si mesmos, de forma a atender uma variedade maior de canais[Ben87].

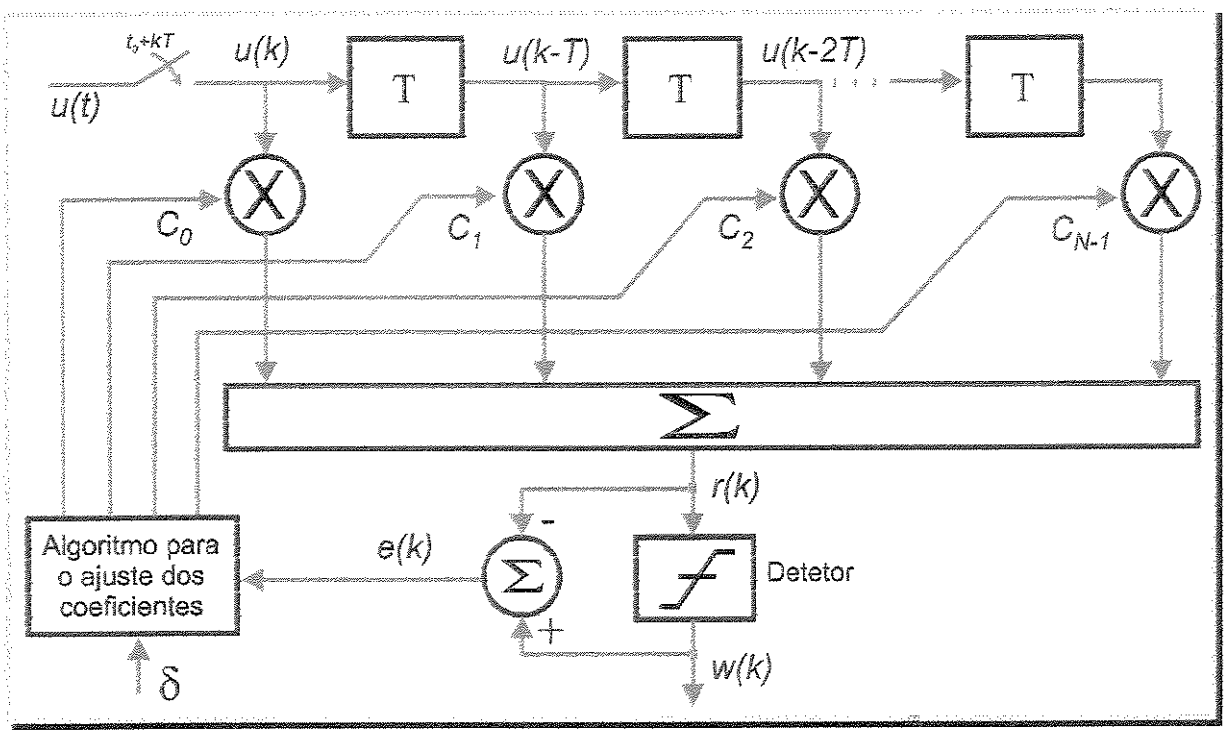


Figura 4.1 Diagrama de um Equalizador Linear Transversal

A primeira solução ao problema descrito foi uma proposta de Lucky (1965), cujo estudo baseou-se num algoritmo para o ajuste automático de um equalizador chamado de *Taped Delay Line (TDL)* ou Não Recursivo ou Transversal Linear[Qur85, Ben87] *(LTE - Linear Transversal Equalizer)* (vide figura 4.1). Neste equalizador o valor corrente e os valores passados, *u(t-fT)*, do sinal recebido são linearmente ponderados pelos coeficientes (ou *tap gains*) de cada derivação *(tap)*, C_f, do equalizador e somados para produzir o sinal de saída que vai para o dispositivo de decisão. Nas implementações digitais deste tipo de equalizador, amostras tomadas à mesma taxa de símbolos do sinal recebido são estocadas num registrador de deslocamento ou memória e as amostras na saída do equalizador (a soma de produtos)

$r(t_0+kT)$ ou $r(k)$ são computadas digitalmente um símbolo por vez de acordo com a seguinte equação:

$$r(k) = \sum_{f=0}^{N-1} C_f u(t_0 + kT - f.T) \qquad (4.1)$$

onde N é o número de coeficientes do equalizador e t_0 denota o instante de amostragem.

Os coeficientes do equalizador, C_f (onde $f=0,1,...,N-1$), podem ser escolhidos de forma a forçar as amostras do conjunto de respostas ao impulso do canal e do equalizador a serem zero em todos os instantes, espaçados em T, exceto em um, na abrangência do equalizador. Este tipo de equalizador chama-se de *Zero-Forcing (ZF)*[Pro95, Lee94] baseado no critério do mesmo nome, cuja idéia é ilustrada na figura 4.2, vide a comparação com a resposta ao impulso do canal (figura 2.7) na presença de ISI.

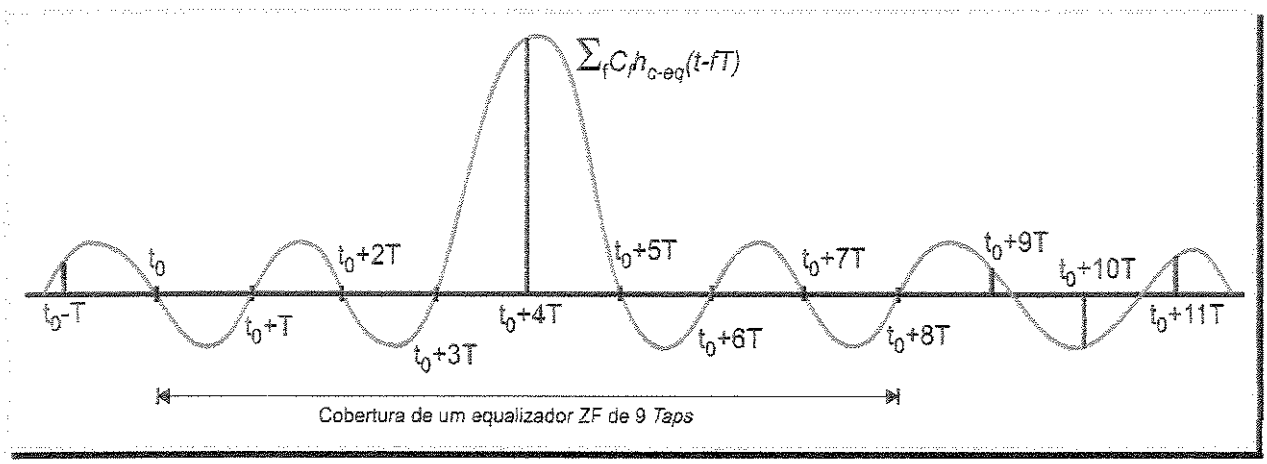


Figura 4.2 Resposta ao impulso da combinação canal e equalizador ZF, $h_{c\text{-}eq}$.

O critério ZF não leva em conta o efeito do ruído, porém garante a minimização do pior caso de ISI (distorção de pico, vide figura 4.3) somente quando o diagrama de olho se mostra inicialmente aberto, podendo esta condição não ser conseguida se as taxas de transmissão forem elevadas ou as características do canal forem mas. Já no critério de LMS, que é mais robusto, os coeficientes são escolhidos de forma a minimizar o erro quadrático médio *(MSE - Mean Square Error)*, portanto o equalizador baseado em LMS maximiza a relação sinal/distorção na saída do equalizador.

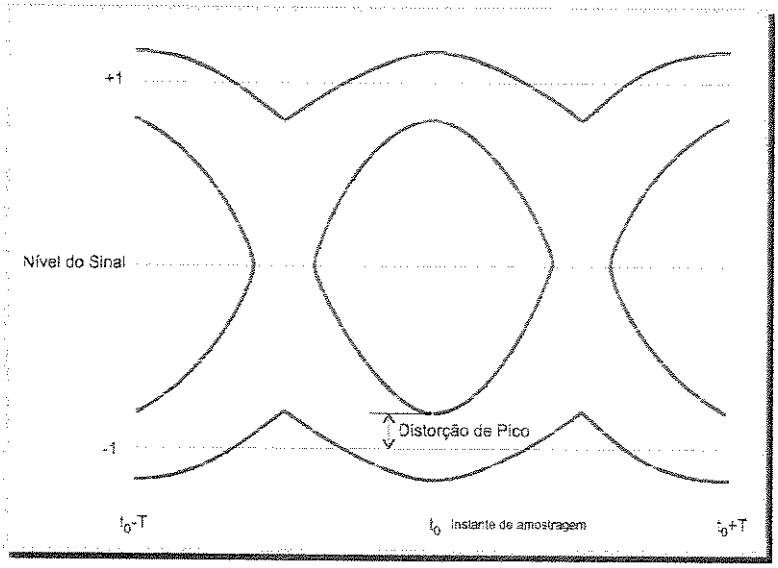


Figura 4.3 Diagrama de olho mostrando a distorção de pico.

O atraso introduzido por um equalizador depende da posição do *tap* principal ou de referência e tipicamente este corresponde ao *tap* de maior magnitude.

É freqüente o uso dos equalizadores LMS em *modems* de linha telefônica de altas velocidades devido a eles serem robustos na presença de ruído e ISI de grande magnitude e por serem melhores no aspecto da convergência do que os equalizadores ZF[Qur85]. O mesmo acontece para *modems* de canais de rádio exceto em casos particulares onde o equalizador ZF é escolhido por sua simplicidade de implementação.

4.2 O Algoritmo de Adaptação dos Coeficientes - LMS

Os algoritmos de adaptação dos coeficientes de um filtro adaptativo podem dividir-se em duas classes: a dos algoritmos de gradiente e a dos de não-gradiente. Neste trabalho será utilizado um algoritmo de gradiente.

Os algoritmos de gradiente baseiam-se no método de passos descendentes comumente utilizado para resolução numérica de sistemas de equações lineares e não-lineares. O algoritmo LMS[Cow85, Pro95, Lee94, Bel87] é talvez o algoritmo de gradiente mais amplamente utilizado; ele ajusta os coeficientes do filtro minimizando o erro quadrático médio entre a saída do filtro *u(k)* e a resposta desejada *d(k)*.

O sinal *u(k)* é filtrado para proporcionar uma estimativa do sinal de entrada *d(k)* de acordo com a seguinte equação:

$$r(k) = \sum_{f=0}^{N-1} C_f(k)\,u(k-f) = \mathbb{C}^T(k)\,\mathbb{U}(k) \tag{4.2}$$

onde :

$$\begin{aligned} \mathbb{C}(k) &= \left[C_0(k) \quad C_1(k) \quad \ldots \quad C_{N-1}(k)\right]^T \\ \mathbb{U}(k) &= \left[u(k) \quad u(k-1) \quad \ldots \quad u(k-N+1)\right]^T \end{aligned} \tag{4.3}$$

O erro da estimativa pode ser calculado assim:

$$e(k) = d(k) - r(k) = d(k) - \mathbf{C}^T(k)\mathbf{U}(k) \tag{4.4}$$

O quadrado deste erro é :

$$e^2(k) = d^2(k) - 2d(k)\mathbf{U}^T\mathbf{C}(k) + \mathbf{C}^T(k)\mathbf{U}(k)\mathbf{U}^T(k)\mathbf{C}(k) \tag{4.5}$$

O erro quadrático médio é o valor esperado de $e^2(k)$, que pode ser escrito assim:

$$E\left[e^2(k)\right] = E\left[d^2(k)\right] - 2\mathbf{R}(u,d)\mathbf{C}(k) + \mathbf{C}^T(k)\mathbf{R}(u,u)\mathbf{C}(k) \tag{4.6}$$

onde o vetor de correlação entre ***U(k)*** e *d(k)* é definido assim:

$$R(u,d) = E\left[d^2(k)\mathbf{U}^T(k)\right] \tag{4.7}$$

e onde a matriz de correlação do sinal de entrada ***U(k)*** define-se assim:

$$R(u,u) = E\left[\mathbf{U}(k)\mathbf{U}^T(k)\right] \tag{4.8}$$

O método de passos descendentes procura o mínimo erro quadrático médio (MSE) mudando o vetor de coeficientes proporcionalmente ao gradiente do erro quadrático médio com respeito ao vetor de coeficientes. O método de passos descendentes pode ser escrito assim:

$$\mathbf{C}(k+1) = \mathbf{C}(k) - 0.5\delta\,\nabla\left[E\left(e^2(k)\right)\right] \tag{4.9}$$

onde:

C(k+1) = vetor de parâmetros atualizado para ser usado no período da amostra seguinte, ***C(k)*** = vetor de parâmetros durante o período de amostra corrente

δ = ganho positivo do laço que controla a velocidade de convergência e a estabilidade do filtro

$\nabla[E(e^2(k))]$ = gradiente do erro quadrático médio com respeito ao vetor de coeficientes ***C(k)***.

Usando a equação (4.6), o gradiente do erro quadrático médio pode ser obtido assim:

$$\nabla\left[E\left(e^2(k)\right)\right] = -2\mathbf{R}(u,d) + 2\mathbf{R}(u,u)\mathbf{C}(k) \tag{4.10}$$

Fazendo o gradiente igual a zero para obter o mínimo erro quadrático médio, pode-se obter a seguinte solução:

$$\mathbf{C}_{LMS} = \mathbf{R}^{-1}(u,u)\mathbf{R}(u,d) \tag{4.11}$$

Esta última equação é conhecida como a equação de Wiener-Hopf[Cow85] que dá a solução ótima para ***C***.

O algoritmo LMS é o método mais popular utilizado para achar uma solução aproximada da equação (4.11), devido a que neste método não são necessárias medidas explícitas das funções de correlação e também não requer inversão de matriz. O algoritmo LMS pode ser escrito assim:

$$\mathbf{C}(k+1) = \mathbf{C}(k) - 0.5\hat{\nabla}\left[E\left(e^2(k)\right)\right] \tag{4.12}$$

onde: $\hat{\nabla}\left[E\left(e^2(k)\right)\right]$ é uma estimativa do gradiente do erro quadrático médio com respeito ao vetor coeficiente ***C(k)***. Esta estimativa pode-se obter tomando o gradiente de apenas uma amostra do erro ao quadrado, portanto:

$$\hat{\nabla}\left[E\left(e^2(k)\right)\right] = \nabla\left[e^2(k)\right] = 2e(k)\nabla e(k) \tag{4.13}$$

e da equação (4.4):

$$\nabla e(k) = \nabla\left[d(k) - \mathbf{C}^T(k)\mathbf{U}(k)\right] = -\mathbf{U}(k) \tag{4.14}$$

desta forma:

$$\hat{\nabla}\left[E\left(e^2(k)\right)\right] = -2e(k)\mathbf{U}(k) \tag{4.15}$$

Utilizando as equações (4.12) e (4.15) o algoritmo LMS pode ser rescrito assim:

$$\mathbf{C}(k+1) = \mathbf{C}(k) + \delta.e(k)\mathbf{U}(k) \tag{4.16}$$

a qual pode ser rescrita assim:

$$C_f(k+1) = C_f(k) + \delta.e(k)u(k-f) \tag{4.17}$$

para *f=0,1,...,N-1*, onde:

$C_f(k+1)$ = valor atualizado do *f*-ésimo coeficiente e que será utilizado durante o seguinte período de amostragem.

$C_f(k)$ = valor do *f*-ésimo coeficiente durante o período de amostragem corrente

A cada novo período de amostragem uma nova amostra é inserida no filtro e o processo é repetitivo. Após algumas iterações os coeficientes do filtro convergem a certos valores de forma que o erro quadrático médio é minimizado.

O ganho de laço, δ, controla a convergência e a estabilidade do filtro, tendo um

valor ótimo que depende do número de derivações *(taps)* utilizadas. É necessária alguma experimentação para a determinação do valor específico de δ.

4.3 Equalizadores para sistemas com modulação QAM

Os *modems* modernos para canal telefônico freqüentemente utilizam a modulação PSK *(Phase-Shift Keying)* para taxas baixas de transmissão e uma combinação de modulação em amplitude e fase ou modulação da amplitude em quadratura (QAM) para altas velocidades de transmissão, onde o ruído e as distorções do canal se tornam bastante influentes. Neste caso, os modems utilizam formas codificadas de QAM, como a modulação de código *trellis*[Pro95, Lee94], para aprimorar seu desempenho.

A figura 4.4 mostra um diagrama de um esquema de modulação QAM que pode ser utilizado tanto para implementar PSK quanto para QAM. Dois sinais modulados em amplitude são superpostos no lado do transmissor e separados no lado do receptor utilizando portadoras ortogonais ou em quadratura para a modulação e demodulação, respectivamente. Os sinais na saída do filtro passa-baixa no receptor são u_i, a componente em fase ou a parte real, e u_q a componente em quadratura ou a parte imaginária, a notação complexa se usa somente por conveniência sendo que ambas as componentes na realidade são sinais reais.

Um equalizador de banda base com coeficientes complexos $\mathbf{C}_f$ opera nas amostras do sinal $U(t)$ e produz na sua saída amostras complexas equalizadas:

$$\mathbf{r}(k) = r_i(k) + jr_q(k) \tag{4.18}$$

Na figura 4.5, ilustra-se o conceito de um equalizador complexo como um conjunto de 4 filtros transversais, com acoplamento cruzado, de duas entradas para duas saídas. Por um lado os coeficientes C_{if}, da parte real, (onde $f=0,1,...,N-1$) ajudam a combater a interferência intersimbólica nos canais em quadratura e em fase, por outro os coeficientes C_{qf}, da parte imaginária, (para $f=0,1,...,N-1$) combatem a interferência cruzada entre os dois canais, esta última podendo ser causada pela assimetria nas características do canal em torno da freqüência da onda portadora.

Os coeficientes do equalizador, como descrito em 4.2, são ajustados de forma a minimizar a média da magnitude ao quadrado do sinal de erro complexo:

$$\mathbf{e}(k) = e_i(k) + je_q(k) \tag{4.19}$$

onde e_i e e_q são as diferenças entre r_i e r_q e os valores desejados d_i e d_q (vide seção 4.2), portanto o método de atualização de coeficientes será:

$$\mathbf{C}_f(k+1) = \mathbf{C}_f(k) + \delta.\mathbf{e}(k).\mathbf{U}^*(t_0 + kT - fT) \tag{4.20}$$

para $f=0,1,...,N-1$, onde $\mathbf{U}^*$ é o complexo conjugado de $\mathbf{U}$. Note-se novamente a utilização da representação no domínio complexo para simplificar a escrita da equação, ao invés de escrever em separado as equações para as 4 multiplicações que na realidade têm de ser implementadas.

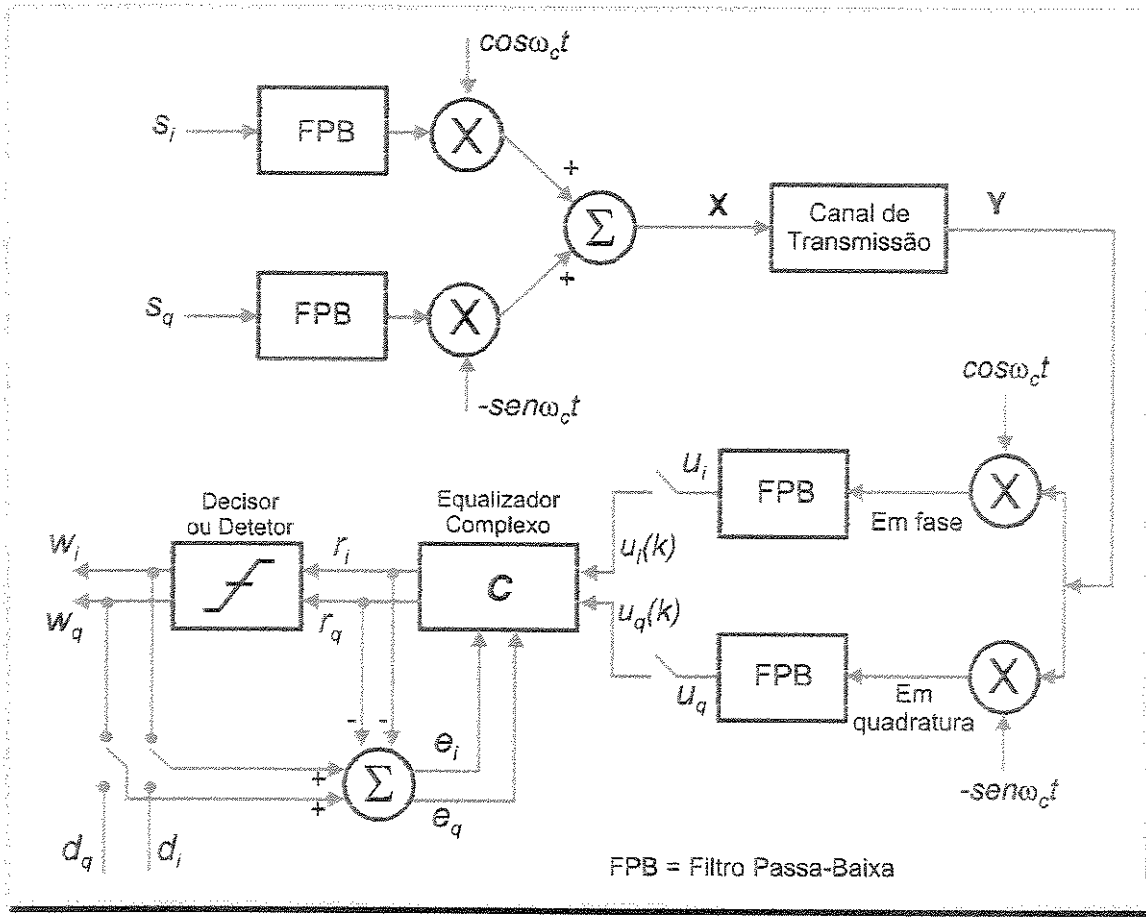


Figura 4.4 Diagrama de um equalizador complexo para um sistema de modulação QAM.

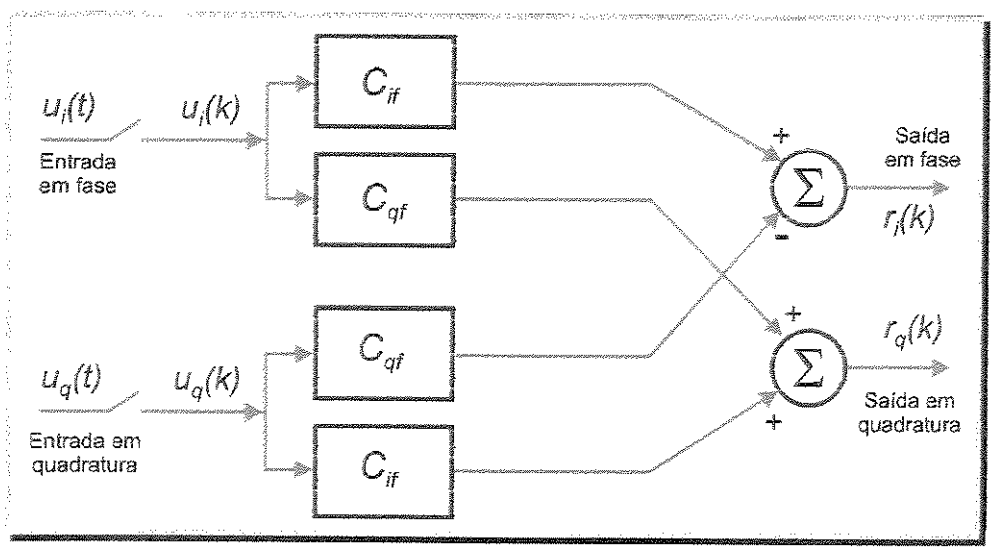


Figura 4.5 Equalizador Transversal Linear Complexo para um sistema de modulação QAM.

4.4 O Equalizador com Decisão Realimentada (DFE)

Este tipo de equalizador não linear é utilizado em canais com distorções de amplitude significativas e explora o princípio de Decisão com Realimentação para o cancelamento da interferência dos símbolos que já foram detectados. Na figura 4.6 mostra-se o diagrama da estrutura deste tipo de equalizador chamado de Equalizador com Decisão Realimentada (DFE) [Lee94, Pro95, Qur85]. O sinal equalizado é a soma das saídas das partes direta e realimentada do equalizador, sendo a parte direta similar ao equalizador transversal linear, apresentado anteriormente. As decisões realizadas sobre o sinal equalizado são realimentadas por meio de um segundo equalizador transversal linear, de forma que se os valores dos símbolos já detectados são conhecidos (pressupondo que as decisões passadas sejam corretas) então a ISI contribuída por esses símbolos pode ser cancelada de forma exata, subtraindo da saída do equalizador os valores dos símbolos passados, pré-ponderados apropriadamente mediante pesos.

Os coeficientes da parte direta e da realimentada podem ser ajustados simultaneamente de forma a minimizar o erro quadrático médio (MSE) e a equação para a atualização dos coeficientes da parte direta é a mesma que para o equalizador linear, vide a equação (4.17) para sinais unidimensionais e a equação (4.20) para sinais bidimensionais. Já os coeficientes da parte realimentada são ajustados de acordo com as seguintes equações:

$$\mathbf{C}_{bck,j}(k+1) = \mathbf{C}_{bck,j}(k) + \delta.\mathbf{e}(k).\mathbf{W}^{*}(t_0 + kT - jT) \tag{4.21}$$

para sinais bidimensionais, onde $\boldsymbol{W}^*$ é o complexo conjugado de $\boldsymbol{W}$ (estimativa do equalizador) e $j=1,\ldots,M$.
E os coeficientes:

$$C_{bck,j}(k+1) = C_{bck,j}(k) + \delta.e(k).w(k-j) \tag{4.22}$$

para sinais unidimensionais, onde: $w(k)$ é o k-ésima decisão de símbolo, $C_{bck,j}(k)$ é o j-ésimo coeficiente da parte realimentada no instante k e M é número total de coeficientes. Os valores ótimos, segundo o LMS, para os coeficientes $C_{bck,j}$ são aqueles tais que façam o ISI nulo, dentro da abrangência, em *taps*, da parte realimentada do equalizador, de forma similar ao equalizador ZF.

Não existe uma certeza se o equalizador DFE com um mesmo número total de coeficientes do que o equalizador linear, pode atingir um MSE menor. O desempenho de cada tipo de equalizador depende das características particulares de cada canal e a fase do amostrador. Porém o equalizador DFE pode compensar as distorções de amplitude sem incrementar em muito o nível de ruído, mais do que o equalizador linear, além de que o desempenho do equalizador DFE é menos sensível à fase do amostrador.

Os coeficientes de um equalizador linear são selecionados de forma que a resposta ao impulso combinada do canal e do equalizador se aproxime a um pulso unitário. No equalizador DFE a capacidade da parte realimentada em cancelar a ISI, permite maior liberdade na escolha dos coeficientes da parte direta do equalizador. A resposta ao impulso da combinação do canal e da parte direta do equalizador pode ter amostras não nulas que seguem ao pulso principal, ou seja a parte direta do equalizador DFE não precisa se aproximar

ao inverso da característica do canal, evitando portanto um crescimento do ruído e *sensibilidade à fase do amostrador*.

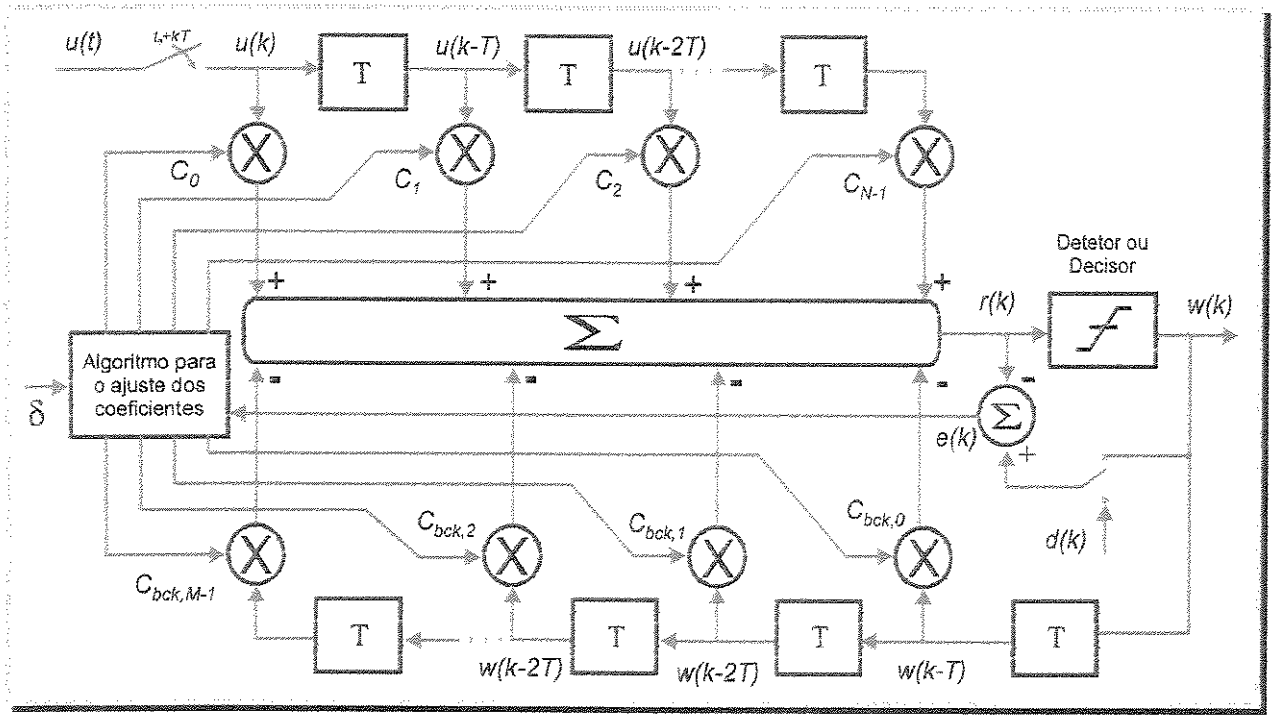


Figura 4.6 Diagrama de um Equalizador com Decisão Realimentada (DFE).

Para o equalizador DFE, se for realizada uma decisão incorreta, esta será realimentada e o equalizador refletirá este erro na sua saída após alguns símbolos, assim que essa decisão incorreta atravesse a parte realimentada do equalizador. Portanto, haverá uma tendência grande do erro se propagar e de se produzirem mais decisões incorretas; porém esta propagação não é crítica, por exemplo, em canais típicos ela produzirá pequenos distúrbios que degradarão levemente o desempenho do equalizador.

4.5 Equalização Adaptativa Baseada em Redes Neurais. Neuro-Equalização

As redes neurais artificiais como sistemas adaptativos podem ser utilizados para autoajustar-se aos estímulos de entrada. Nesse sentido, na área de detecção de sinais discretos, as redes neurais têm sido utilizadas para acompanhar as variações dos limites de decisão[Koh90b, Ber98, Gib90]. A figura 4.7 mostra os limites de decisão ótimos, para o caso de uma constelação QAM-4, em condições ideais de transmissão dos sinais, tanto para uma constelação sem distorção quanto para uma com distorção não-linear (vide figura 2.6e).

Muitos trabalhos de pesquisa têm se dedicado ao estudo das redes neurais em aplicações de telecomunicações utilizando o *MultiLayer Perceptron*, o qual é comumente treinado com o algoritmo de retro-propagação do erro conhecido como *backpropagation*[Rum95, Ben92, Siu93]. O algoritmo de Filtragem *Kalman* [Goo84] oferece uma velocidade maior de convergência do que o de retro-propagação. Outras propostas de algoritmos para o *MultiLayer Perceptron* são o *feedforward*[Pul93] e o *polinomial*[Xia94].

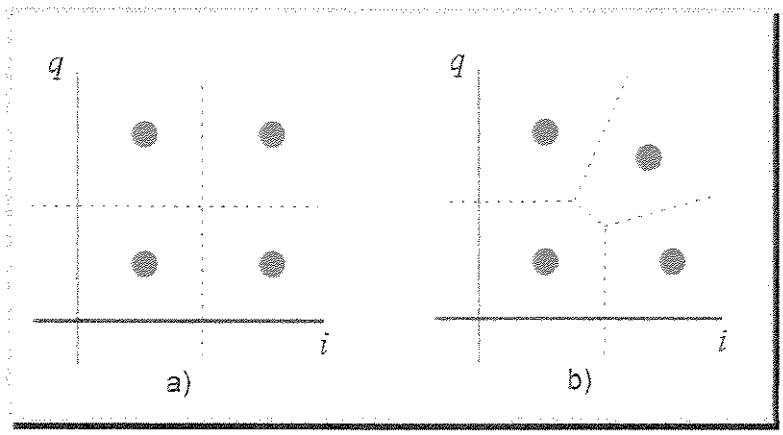


Figura 4.7 Limites de Decisão Ótimos para condições ideais numa constelação QAM-4: a) sem distorção, b) com distorção não-linear.

Além do *MultiLayer Perceptron* outras estruturas de redes neurais e algoritmos também têm sido utilizadas para aplicações em telecomunicações, como, por exemplo, as Redes Neurais Recorrentes[Kec94] *(RNN - Recurrent Neural Networks)* na equalização adaptativa de canais lineares e não lineares na presença de distorções severas. A RNN é vantajosa em relação ao equalizador linear convencional nos casos em que o canal têm nulos espectrais na sua largura de banda.

O Equalizador Bayesiano Adaptativo[Che93b] apresenta uma solução ótima para o processo de equalização de sinais e é derivado a partir da teoria de decisão de Bayes. Este equalizador transversal pode ser implementado utilizando estruturas adaptativas não lineares como a rede neural de Funções Base Radiais[Che93a, Mul96].

Por outro lado, uma mistura baseada na capacidade de aprendizado das redes neurais e nas estruturas *if-then* da lógica nebulosa (ou difusa) resulta nos chamados Filtros Adaptativos Neuro-Difusos[Lin97] para aplicações em equalização de canais com resposta não linear e em cancelamento adaptativo de ruído. Este tipo de filtros recai na classificação de MLP *feedforward*.

Finalmente, a união dos Mapas Auto-organizados de Kohonen[Koh90a, Koh90b, Koh96] (SOM) junto com o algoritmo chamado de contra-propagação do erro, resultou numa nova estrutura de equalização adaptativa, que é descrita em maior detalhe na seção 4.6. A aplicação dessa estrutura no presente trabalho está concentrada no assunto da detecção de sinais propagados por múltiplos percursos (caminhos) e que dá ao processo de equalização características particulares de compensação de distorções não-lineares. Outras aplicações interessantes para esta estrutura na área de telecomunicações são a decodificação e problemas de roteamento[Koh96].

4.5.1 Equalização Baseada no *MultiLayer Perceptron*

O *MultiLayer Perceptron*[Wid88, Lip87, Gib90] (MLP) é uma rede de conexões ponderadas, por pesos sinápticos, entre os nós de camadas consecutivas, onde os pesos são ajustáveis.

No caso da detecção de sinais, que foram propagados por múltiplos caminhos, o sinal do canal de transmissão é submetido a uma Linha de Retardo. Esta linha de atraso serve

de nível de entrada ao MLP (vide ilustração da figura 4.8).

Uma forma comum de atualização dos pesos sinápticos de um MLP é a utilização do algoritmo de retro-propagação do erro, onde os pesos sinápticos correspondem aos *coeficientes (taps) do equalizador transversal ou DFE convencional.*

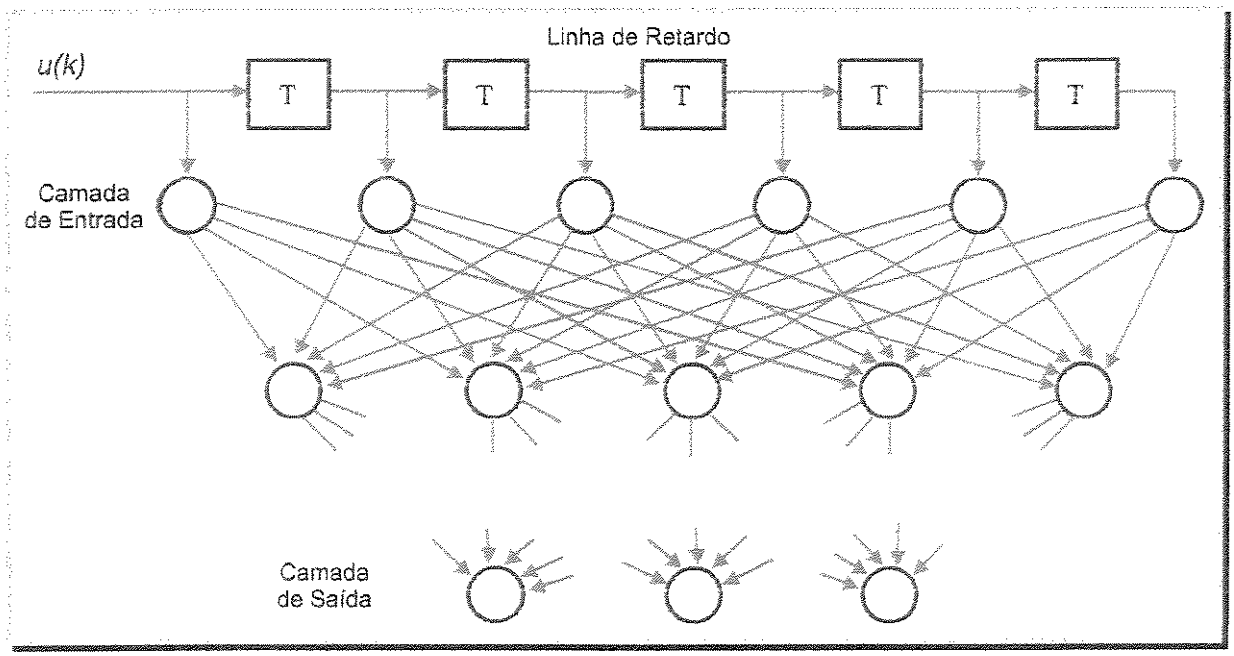


Figura 4.8 Estrutura de um *MultiLayer Perceptron* com uma Linha de Retardo.

A utilização do algoritmo de retro-propagação em estruturas de neuro-equalização tem sido reportada em várias publicações[Siu90, Gib91, Mey93, Cha95] e em particular Siu et al.[Siu90, Gib90], que fazem a comparação de um equalizador transversal com outro similar baseado em MLP e de um DFE com um DFE-MLP. Nestas estruturas o sinal é alimentado a partir da linha de retardo, que no caso do DFE inclui os sinais de realimentação próprios desta estrutura. Os testes dessas estruturas de neuro-equalização foram realizadas considerando modelos de canal de duplo e de triplo-caminho. Siu et al., reportaram que o MLP, treinado por retro-propagação, é superior à estrutura convencional de equalização treinada pelo algoritmo LMS. A estrutura MLP-DFE mostrou ter um desempenho superior à estrutura LMS-DFE quando os níveis de ruído são elevados, porém o desempenho se deteriora quando a relação sinal/ruído (SNR) melhora.

A estrutura original de equalização proposta por Gibson et al.[Gib91] foi projetada para sinais unidimensionais, portanto foi necessário modificar o algoritmo de forma a ser capaz de lidar com sinais bidimensionais[Ben92, Cha94], como os utilizados em modulação QAM. Chang et al.[Cha94] apresentaram no seu trabalho uma extensão da estrutura para o domínio de representação complexo de sinais. A estrutura de equalização utilizada foi o DFE, o esquema de modulação o QAM e o modelo de canal o de multicaminho com desvanecimento para enlaces de rádio. O algoritmo por eles proposto, chamado de *packet-wise backpropagation*, mostrou boas propriedades de convergência. Mostraram também que a estrutura proposta proporciona um desempenho melhor, na taxa de erro de bits (BER), do que o DFE

convencional.

A utilização do MLP utilizando outros algoritmos também foi reportada[Choi93]. Em particular o algoritmo de filtragem *Kalman*[Goo84], amplamente utilizado na identificação de sistemas dinâmicos não-lineares, que oferece velocidades de convergência mais rápidas e respostas de BER superiores ao algoritmo de retro-propagação. Porém este algoritmo exige custos computacionais mais altos requerendo uma implementação em processamento paralelo.

Duas modificações no algoritmo e estrutura originais do equalizador MLP foram apresentadas. A primeira chamada de *Feedforward Perceptron*[Pui93] apresenta simplicidade comparada com o MLP convencional, cuja análise em geral é complexa. A segunda *polinomial*[Che90, Xia94] mostrou ter uma estrutura mais simples e maior rapidez na convergência do que o MLP convencional.

4.5.2 Equalização Baseada em Redes Neurais Recorrentes

As Redes Neurais Recorrentes *(RNN - Recurrent Neural Networks)*[Gil94], nas quais cada célula é conectada a todas às outras células, são uma generalização das redes neurais e mostram ser sistemas dinâmicos não lineares. A sua utilização foi proposta para a equalização adaptativa de canais de transmissão com características lineares e não lineares[Kec94, Hac97] e sua estrutura de realimentação a faz atraente para a equalização de canais não lineares com nulos espectrais. Uma RNN de tamanho médio pode modelar com precisão a característica inversa de um canal de comunicação com um desempenho melhor do que os algoritmos tradicionais de equalização. Kechriotis et al.[Kec94] apresentaram na sua proposta uma estrutura para ser utilizada em *equalização cega*[Pro95, Ama98] de canais não lineares utilizado para um subconjunto das estatísticas do sinal transmitido, cuja estrutura pode-se ver no diagrama da figura 4.9. A equalização cega é particularmente útil quando as seqüências de treinamento não são possíveis de aplicar, como no caso de transmissão de sinais em redes de comunicações multiponto. Este esquema de equalização aproveita a propriedade ciclo-estacionária do sinal transmitido. As RNN por sua capacidade de aprendizado de mapeamentos não lineares de complexidade arbitrária, são uma solução ao problema da equalização cega de canais com características não lineares.

Finalmente, fazer uma comparação dos resultados obtidos nas diferentes publicações desta área de aplicação das redes neurais e das telecomunicações é uma tarefa muito complexa, devido à heterogeneidade de arquiteturas, algoritmos de treinamento, esquemas de modulação e modelos para os canais de transmissão. É difícil determinar qual algoritmo é melhor, uma vez que as publicações são baseadas em algoritmos diferentes e sistemas de comunicação simulados, sendo muito provável achar uma situação em que um determinado tipo de equalizador adaptativo proporcione um melhor desempenho do que os equalizadores convencionais (DFE, LTE).

Normalmente o sistema de comunicação simulado é relativamente simples e poucas publicações mencionam o uso de esquemas de modulação mais complexos como um PAM binário ou um QAM. Outro problema é em relação à escolha do modelo de canal a ser utilizado de forma a se ter resultados mais perto da realidade e os resultados serão mais confiáveis se for utilizado um modelo de canal real tal como um enlace de rádio ou um canal telefônico.

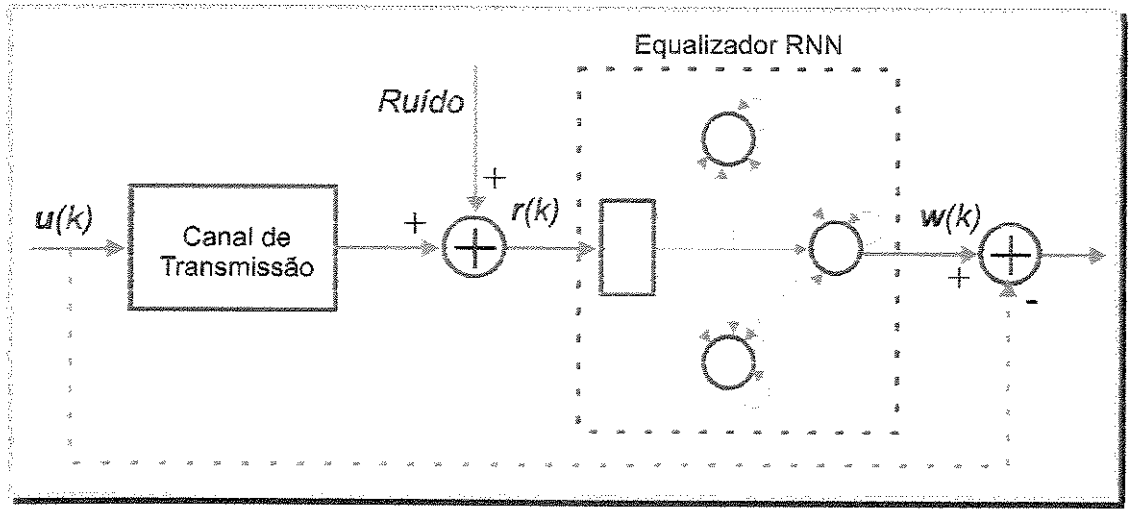


Figura 4.9 Diagrama de um equalizador de sinais baseado em Redes Neurais Recorrentes. A linha pontilhada, de *u(k)* para a saída, é usada quando se requer treinamento da rede e não é usada quando o esquema de equalização é cego.

4.6 Equalização Adaptativa Baseada no Mapa Auto-Organizado (SOM)

Os equalizadores adaptativos baseados no algoritmo de Mapas Auto-Organizados de Kohonen (vide capítulo 3) foram inicialmente desenvolvidos no Laboratório de Computação e Ciências da Informação da Universidade de Helsinque[Koh96, Koh90a, Koh90b, Koh91, Rai91, Koh92]. A idéia principal desta proposta é a da utilização do SOM para o seguimento das variações dos sinais num sistema de comunicação digital de forma que a rede neural auto-organizada compense adaptativamente as distorções não lineares e o equalizador convencional compense as distorções lineares. Cada nó ou célula da rede neural tem o seu estado (símbolo) correspondente na constelação de símbolos o qual a célula deve seguir.

Depois dessa idéia básica ser proposta, a utilização do SOM num sistema de equalização adaptativa foi então introduzida[Koh90b], concentrando-se na utilização da combinação do SOM com o equalizador transversal linear (LTE). Os resultados mostraram que o LTE teve uma melhor tolerância à níveis altos de Interferência Intersímbolica (ISI), porém que a estrutura LTE-SOM é melhor na compensação de não linearidades.

O primeiro esquema de combinação de um LTE e do SOM foi o paralelo[Koh90a] (vide figura 4.10), no qual foram detectadas dificuldades para achar coeficientes adequados para o equalizador. Nesta estrutura os pesos internos das células nó foram utilizadas como os limites de decisão do equalizador. A combinação LTE-SOM em paralelo mostrou um desempenho deficiente sob condições de altos níveis de ISI.

Um segundo esquema de combinação LTE-SOM foi em cascata ou série[Koh96, Koh92] (vide figura 4.11); essa estrutura mostrou ser mais estável do que a primeira. O algoritmo consiste na alimentação do sinal de entrada diretamente ao LTE e posteriormente ao SOM de forma que este último é utilizado como um detetor adaptativo e também neste caso os pesos internos das células servem de limites de decisão do equalizador.

A diferença básica entre os dois tipos de combinações, LTE-SOM, é o sinal que é utilizado para a adaptação do mapa auto-organizado. No combinação em paralelo, o sinal é a entrada do neuro-equalizador, na segunda combinação o sinal utilizado é uma estimativa da

saída da parte linear do neuro-equalizador.

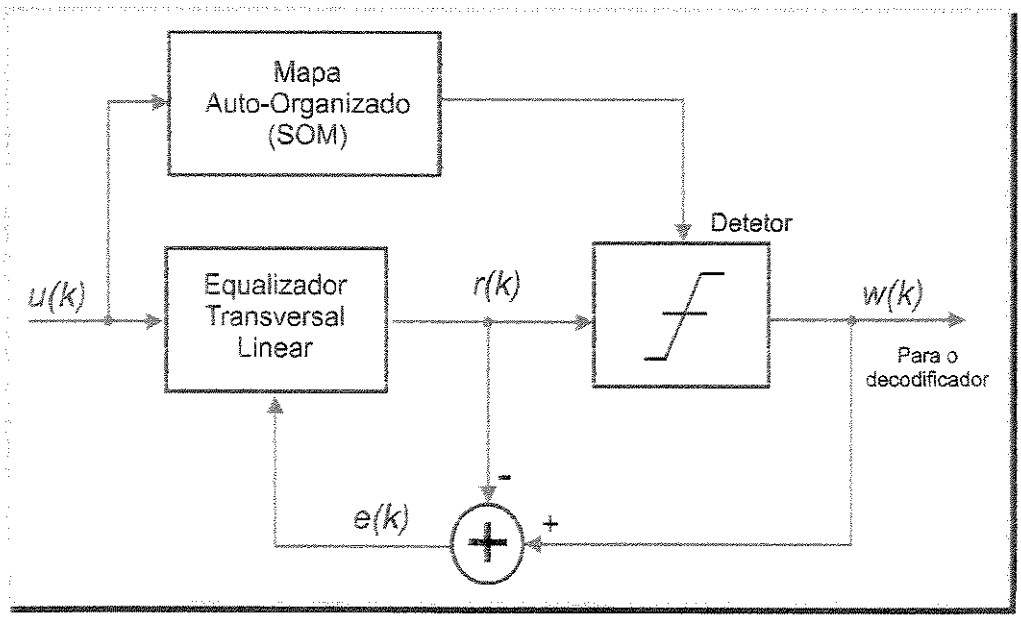


Figura 4.10 Combinação em Paralelo do Equalizador Linear com Mapa Auto-Organizado.

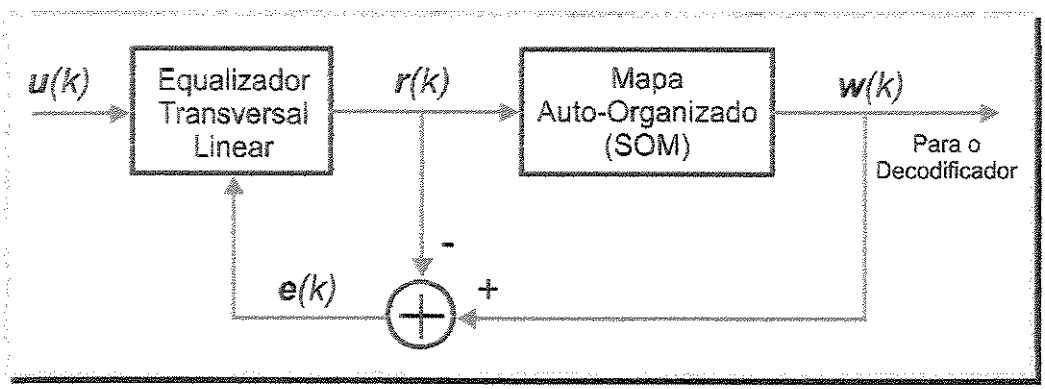


Figura 4.11 Combinação em Cascata ou Série do Equalizador Linear com o Mapa Auto-Organizado.

Por fim, a combinação do equalizador com decisão realimentada (DFE) e o SOM ligados em série[Rai91], foi proposta para obter um melhor desempenho no cancelamento de ISI. Esta última estrutura é o motivo do presente trabalho. A cada período de amostragem uma nova amostra é inserida na linha de retardo do DFE e a saída deste é alimentada ao SOM, a amostra é então classificada e realimentada ao equalizador através do equalizador de realimentação. No diagrama da figura 4.12 observa-se a estrutura de neuro-equalização DFE-SOM. Os coeficientes das derivações *(taps)* são atualizados segundo o algoritmo de gradiente LMS descrito em 4.2, cuja equação é a (4.20). A diferença obtida da subtração do sinal da saída do DFE e do sinal na saída do SOM é multiplicada pelo vetor da amostra na entrada e um fator pequeno de adaptação ou ganho. A saída do equalizador, então, é o vetor de pesos

interno da célula com melhor casamento respeito à entrada (conhecida como a célula *vencedora do processo competitivo*). A *saída do DFE é considerada como a saída desejada* do equalizador. Esta descrição pode ser acompanhada com diagrama da figura 4.13.

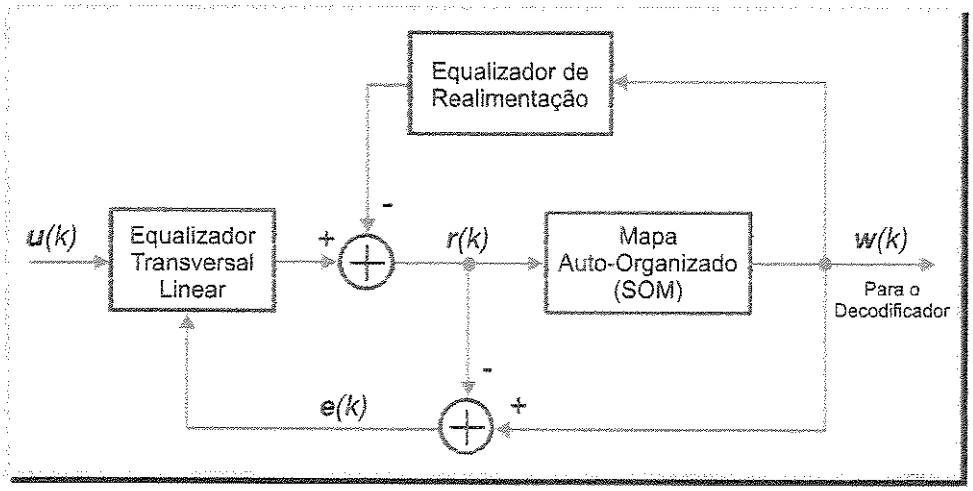


Figura 4.12 Combinação em Série do Equalizador com Decisão Realimentada e o Mapa Auto-Organizado.

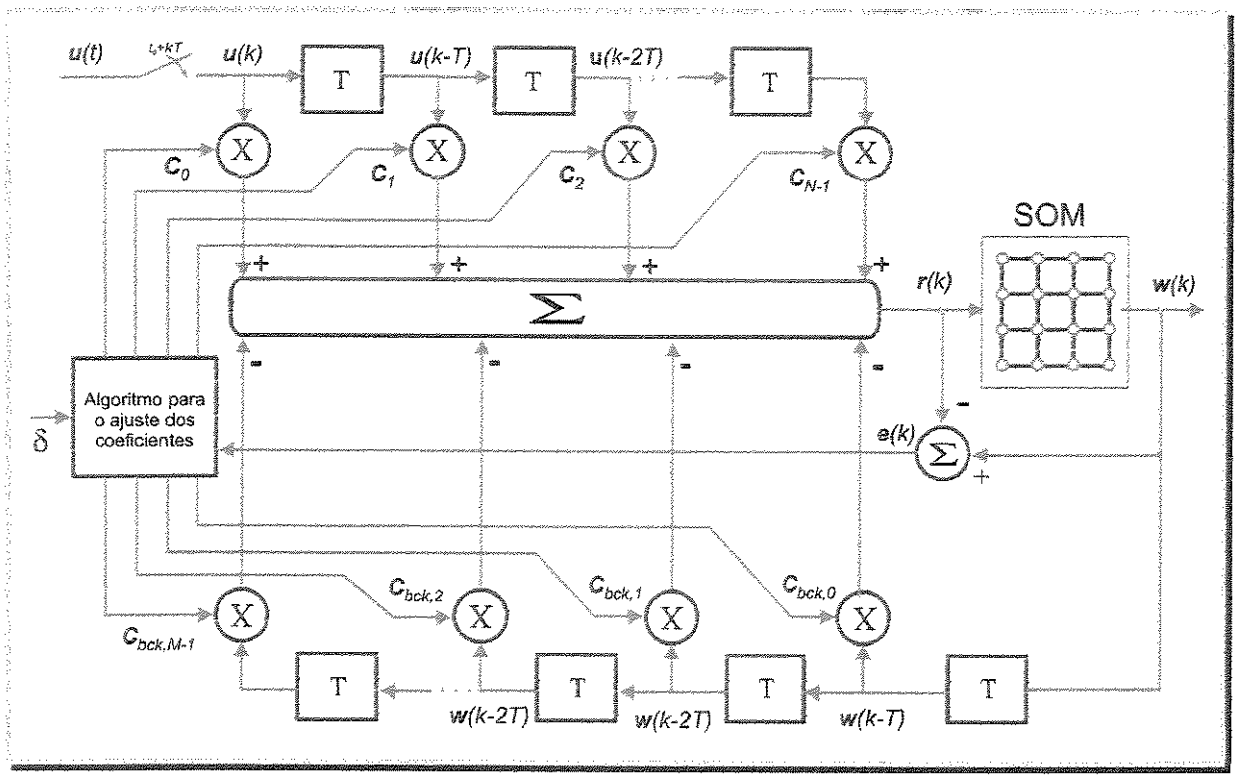


Figura 4.13 Diagrama de um Equalizador com Decisão Realimentada (DFE) combinado com o Mapa Auto-Organizado de Kohonen (SOM).

A utilização do SOM junto com o DFE ao invés do LTE tráz o beneficio de obter um melhor desempenho para o equalizador. Porém, em condições de ISI, o preço pago por isto está relacionado a erros na decisão, similar ao descrito quando se utilizado um DFE puro. Um erro na decisão causa oscilações rápidas de erro devido a realimentação dos mesmos. O tamanho dessas oscilações depende do número de derivações na linha de retardo do DFE e especialmente na linha de retardo da parte realimentada.

4.6.1 O Algoritmo de Adaptação

A topologia da rede neural auto-organizada a ser utilizada para a deteção de níveis de sinal QAM é um arranjo planar de neurônios ou células, onde cada célula corresponde a um símbolo específico da constelação.

Antes da transmissão, os pesos internos das células são iniciados para que tenham os valores ideais *(i,q)* do esquema de modulação QAM. Durante a transmissão, o sistema de aprendizado recebe, aos pares, os sinais discretos do canal de transmissão, ***u(k)***, que em geral vêm acompanhados de ruído e distorções. Como descrito no algoritmo original de Kohonen [Koh89, Koh90c, Koh96] (vide capítulo 3), determina-se a célula com melhor casamento à entrada, com uma métrica de distância.

A adaptação é projetada de forma a manter os pesos internos, bidimensionais, o mais próximo possível dos valores correntes *(i,q)* correspondentes, ainda que na presença de distorção. Conseqüentemente, os pesos internos das células são atualizados de acordo com a equação (3.4) particularizada para a equalização de sinais.

Para a célula vencedora :

$$w_j(k+1) = w_j(k) + \alpha \left[r(k) - w_j(k) \right] \tag{4.23}$$

Para as células da vizinhança topológica, N_{j^*}:

$$w_j(k+1) = w_j(k) + \beta \left[r(k) - w_j(k) \right] \tag{4.24}$$

E para as células não pertencentes à vizinhança topológica:

$$w_j(k+1) = w_j(k) \tag{4.25}$$

A vizinhança topológica consiste das células diretamente conetadas à célula vencedora, que neste caso abrange apenas um nível (vide figura 4.14), diferente do algoritmo original, onde a vizinhança topológica é função dos passos de amostragem e pode ser dependente do nível de abrangência.

Como esta equalização de sinais consiste em seguir os deslizamentos e outras deteriorações nos sinais de entrada, os coeficientes de aprendizagem ou fatores de ganho, α e β, são valores fixos e não funções do tempo monotonicamente decrescentes, como no algoritmo original.

Os pesos internos das células que representam o mapa auto-organizado se espalham num espaço bidimensional de forma que a sua distribuição (fdp) corresponda à distribuição dos vetores de entrada.

Se os valores de α e β forem selecionados apropriadamente, os pesos internos seguirão os vetores de entrada variantes no tempo, ou seja, identificarão adaptativamente o sinal de entrada[Koh90a, Koh91]. A adaptação é muito efetiva já que o espaço dos sinais transmitidos tem uma topologia bem definida e também devido à aprendizagem na vizinhança topológica. Entretanto, isto é verdadeiro somente se as distorções não forem muito severas, de modo que a topologia linear dos sinais transmitidos seja mantida.

A regra de aprendizado normalmente é aplicada simetricamente em todas as direções no arranjo de células a partir da célula vencedora. Em termos gerais, isto pode causar alguns deslizamentos em direção ao centro da constelação, nos valores dos pesos internos das células, isto devido ao fato das células localizadas nas esquinas não terem uma vizinhança topológica em todas as direções. Isto pode ser compensado pela amplificação e deslocamento específico nos pesos internos em cada célula[Koh90b], tendo um novo conjunto de equações.

Para a célula vencedora:

$$w_j(k+1) = w_j(k) + \alpha \left[V_{dc} + V_g\, r(k) - w_j(k)\right] \tag{4.26}$$

onde: V_{dc} é o fator de deslocamento e V_g o fator de amplificação, valores específicos para cada célula.

Para as células na vizinhança topológica, N_{j^*}:

$$w_j(k+1) = w_j(k) + \beta \left[V_{dc} + V_g.r(k) - w_j(k)\right] \tag{4.27}$$

E para as células que não pertencen à vizinhança topológica:

$$w_j(k+1) = w_j(k) \tag{4.28}$$

Portanto, o algoritmo é capaz de seguir as distorções, no sinal de entrada, desde que a ordem local na constelação de sinais seja preservada. Esta estrutura é capaz de compensar diversos tipos de distorções não lineares, como os descritos no capítulo 2. Assim, ainda que os valores absolutos dos níveis dos sinais mudem drasticamente, eles poderão ser seguidos.

4.6.2 A Vizinhança Topologica

A principal diferença entre a adaptação por meio dos mapas de auto-organização e os sistemas convencionais de adaptação, é a conectividade de neurônios numa vizinhança. Isto significa que depois da classificação da amostra de entrada, a adaptação não somente será realizada em cima da célula vencedora como também nas células que estão dentro da sua Vizinhança Topológica. A forma em que as células vizinhas são definidas afeta ao algoritmo de adaptação de modo que lhe proporciona uma alta probabilidade do processo de adaptação ter sucesso.

A forma convencional de definir a vizinhança, no caso particular de ter uma

topologia bidimensional quadrada, é como se mostra na figura 4.14. No entanto, existe o problema de como definir a vizinhança para as células situadas nas esquinas e como dito anteriormente, isto pode ser resolvido pela inserção de dois parâmetros, V_{dc} e V_g, específicos para cada célula. Porém isto traz como conseqüência outro problema, o de como encontrar valores apropriados, transpondo o problema de descontinuidade da vizinhança no problema da busca deste parâmetros. Uma outra alternativa é considerar a vizinhança esférica, como se ilustra na figura 4.15, porém isto também introduz o problema de como realizar o treinamento neste novo tipo de vizinhança.

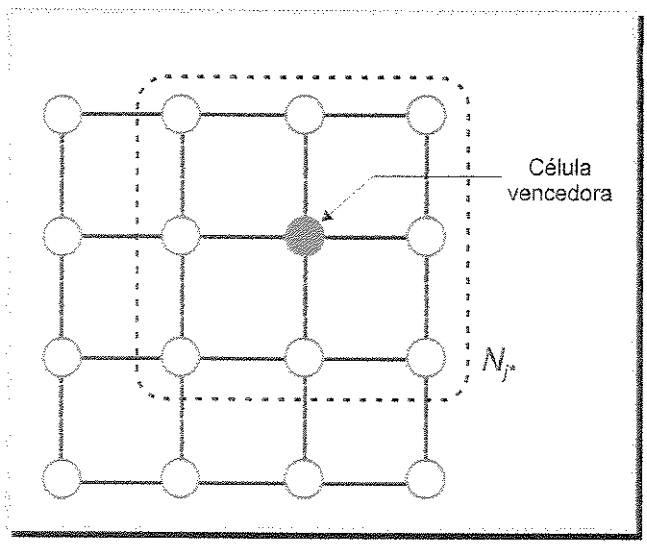


Figura 4.14 Topologia bidimensional quadrada para o Mapa Auto-Organizado de Kohonen. N_{j^*} é a vizinhança topológica.

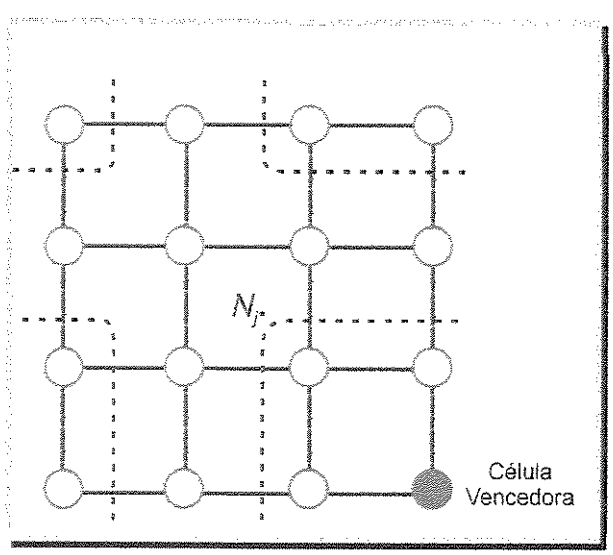


Figura 4.15 Vizinhança Esférica para o Mapa Auto-Organizado de Kohonen. N_{j^*} é a vizinhança topológica.

Capítulo 5:

Implementação em Software - Simulações e Resultados

Uma das linhas de pesquisa do Departamento de Semicondutores, Instrumentos e Fotônica (DSIF) da FEEC/UNICAMP é a implementação de estruturas de Redes Neurais Artificiais tanto em *software* quanto em *hardware* e sua aplicação em diversas áreas do conhecimento, tais como processamento digital de sinais para reconhecimento de sinais acústicos[Jar95a] e em telecomunicações para equalização adaptativa de sinais, sendo este último o assunto do presente trabalho.

Nesse sentido, como matéria de uma tese de doutorado, um primeiro trabalho de implementação da Rede Neural ou Mapa Auto-Organizado de Kohonen foi realizado[Jar97]. Essa primeira realização visou uma implementação genérica do modelo da rede neural, de forma a estudar a influência de todos os parâmetros do algoritmo. Foram estudadas várias medidas de distância para a determinação da célula ganhadora do processo competitivo que se dá a cada novo vetor apresentado à rede, analisando a sua influência na convergência do algoritmo e na formação dos mapas finais. O coeficiente de adaptação, $\alpha(t)$ (vide equação 3.4); que é uma função monotonicamente decrescente, foi implementada em forma continua no tempo e em forma discreta para a sua fácil realização física. Da mesma maneira foi estudada a diferença de a função do raio da vizinhança topológica, $N_j(t)$ (vide seção 3.5); ser implementada em forma contínua ou em forma discreta no tempo

Com base nas descrições teóricas dos capítulos anteriores e da experiência da realização genérica do modelo da rede neural, neste capítulo será apresentada a implementação em *software* da rede neural de Kohonen e sua adequação para a aplicação específica de equalização adaptativa de sinais, anteriormente proposto por Raivio[Koh90a, Koh90b, Rai91]. O diagrama do sistema de comunicação digital simulado mostra-se na figura 5.1. O sistema descrito consiste do gerador de sinais (símbolos), do canal de transmissão e do equalizador adaptativo de sinais. No gerador de sinais são originados os sinais QAM, os quais podem sofrer ou não, distorções lineares e não-lineares. O bloco canal de transmissão modela um determinado meio físico; nele é introduzido o ruído com uma dada relação sinal/ruido (SNR). O bloco equalizador adaptativo de sinais inclui ora o equalizador convencional ora o equalizador baseado no mapa auto-organizado de Kohonen. $\boldsymbol{x}(k)$, $\boldsymbol{y}(k)$ e $\boldsymbol{w}(k)$ são seqüências de símbolos.

A linguagem de programação C foi utilizada para a descrição do sistema de comunicação em estudo, um módulo para cada bloco da figura 5.1. Para o bloco gerador de sinais, o módulo em C correspondente chama-se **symsrc**; para o canal de transmissão, **awgn**; e, para o equalizador, **dfe** para convencional ou **dfe_knn** para o baseado em SOM. As listagens podem ser observadas nos anexos 1, 2, 3 e 4, respectivamente.

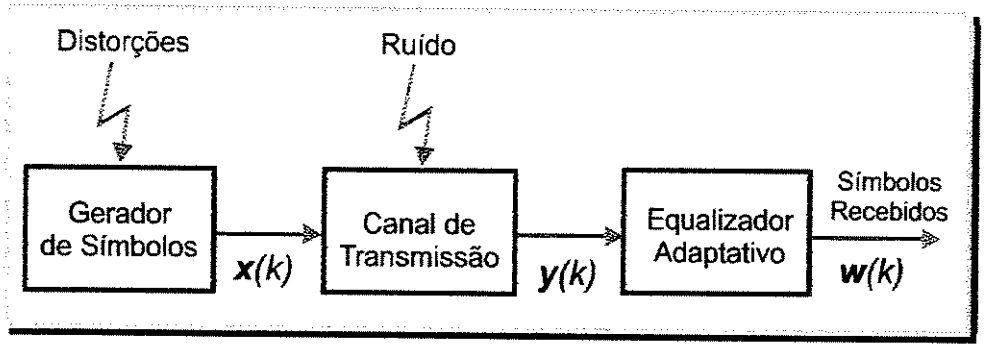


Figura 5.1 Diagrama do sistema de comunicação utilizado nas simulações.

Previamente à descrição do sistema em C foram realizadas descrições no ambiente de simulação para processamento digital de sinais, especificamente de telecomunicações, da empresa *Mentor Graphics Corp.*, com cujas ferramentas o DSIF/FEEC conta. O conjunto de ferramentas de simulação chama-se de *DSP Station*^{TM} [Men95a], no qual é possível especificar um sistema de comunicações digital quer seja utilizando um bloco pronto (parte das bibliotecas de módulos[Men95b] de que o ambiente dispõe) quer seja descrevendo blocos definidos pelo usuário através de uma linguagem chamada de DFL *(Data Flow Language)*[Men95c]. Após a descrição o sistema especificado é verificado através de um simulador, no domínio do tempo, manejado por dados *(DDSIM*^{TM} *- Dynamic Data flow SIMulator)*[Men95d]. É possível verificar a degradação sofrida no desempenho do sistema especificado pela troca de representação dos sinais (números) de ponto flutuante para ponto fixo. Depois de validar a especificação o sistema pode ser traduzido (sintetizado) para uma descrição em outra linguagem como o VHDL *(VHSIC Hardware Description Language)* e a partir dela realizar a implementação física ou até mesmo para o código de algum processador digital de sinais comercial, tal como o DSP56000 da *Motorola*[Men95e]. Não obstante a potencialidade dos recursos proporcionados neste ambiente de simulação, a implementação do sistema em estudo não pode ser completada devido à falta de alguns dos módulos de simulação da própria ferramenta.

O esquema de modulação escolhido para as simulações do sistema em estudo é o QAM-16, esquema também utilizado nos trabalhos prévios de Raivio[Koh90a, Koh90b, Rai91].

5.1 O Gerador de Sinais

Os sinais são gerados na forma de pares de números correspondendo a cada um dos níveis possíveis da constelação de sinais QAM-16. Esses números são gerados aleatoriamente de forma que cada um dos níveis tenha a mesma probabilidade de ser escolhido e formar parte do símbolo a ser transmitido num determinado instante de simulação.

Da equação (2.2), para QAM-16 então o número de níveis $L=4$ e a matriz de LxL das amplitudes a_i e b_i é:

$$(a_i, b_i)\frac{A_c}{2} = \begin{bmatrix} \left(-\frac{3}{2}Ac, \frac{3}{2}Ac\right) & \left(-\frac{1}{2}Ac, \frac{3}{2}Ac\right) & \left(\frac{1}{2}Ac, \frac{3}{2}Ac\right) & \left(\frac{3}{2}Ac, \frac{3}{2}Ac\right) \\ \left(-\frac{3}{2}Ac, \frac{1}{2}Ac\right) & \left(-\frac{1}{2}Ac, \frac{1}{2}Ac\right) & \left(\frac{1}{2}Ac, \frac{1}{2}Ac\right) & \left(\frac{3}{2}Ac, \frac{1}{2}Ac\right) \\ \left(-\frac{3}{2}Ac, -\frac{1}{2}Ac\right) & \left(-\frac{1}{2}Ac, -\frac{1}{2}Ac\right) & \left(\frac{1}{2}Ac, -\frac{1}{2}Ac\right) & \left(\frac{3}{2}Ac, -\frac{1}{2}Ac\right) \\ \left(-\frac{3}{2}Ac, -\frac{3}{2}Ac\right) & \left(-\frac{1}{2}Ac, -\frac{3}{2}Ac\right) & \left(\frac{1}{2}Ac, -\frac{3}{2}Ac\right) & \left(\frac{3}{2}Ac, -\frac{3}{2}Ac\right) \end{bmatrix} \quad (5.1)$$

onde: A_c é a distância entre os níveis de dois símbolos consecutivos.

Os valores das componentes *i* e *q* dos símbolos são normalizados de maneira que não ultrapassem a faixa de (-1 a 1):

$$|i_i| < 1, \quad |q_i| < 1 \quad (5.2)$$

Desta forma, o cálculo de A_c pode ser realizado considerando que a soma do próprio valor do nível, mais a parcela de distorção nele introduzido e mais a contribuição do ruído, recaia dentro das faixas especificadas em (5.2). Dado que os símbolos situados nas esquinas são os que podem chegar ao limite, o calculo de A_c é realizado considerando estes símbolos e expresso na seguinte equação para o caso da componente *i*:

$$\left| i_{esq} + i_{esq} f_d + \frac{i_{esq}}{f_\eta} \right| < 1 \quad (5.3)$$

onde: i_{esq} é a componente *i* de um símbolo situado num vértice (esquina) da constelação, f_d é o fator de distorção e f_η é o fator de ruído. Tendo em conta o seguinte: como a constelação é simétrica, pode-se utilizar para o cálculo o símbolo situado na esquina superior direita; a distorção de deslizamento de grade de 30% muda o valor das componentes *i* e *q* em maior grau do que a rotação de grade de 20°; e considerando a relação sinal por ruído de 8dB, a equação (5.3) assume o caso particular de:

$$\frac{3}{2}A_c + \frac{3}{2}A_c(0.3) + \frac{3}{2}A_c\left(\frac{1}{10^{\frac{8dB}{10}}}\right) < 1 \quad (5.4)$$

resultando em:

$$A_c < 0.456 \quad (5.5)$$

Considerando o valor de A_c=*0.4*, então a matriz da equação (5.2) fica assim:

$$(i_i,\ q_i)=\begin{bmatrix} (-0.6,0.6) & (-0.2,0.6) & (0.2,0.6) & (0.6,0.6) \\ (-0.6,0.2) & (-0.2,0.2) & (0.2,0.2) & (0.6,0.2) \\ (-0.6,-0.2) & (-0.2,-0.2) & (0.2,-0.2) & (0.6,-0.2) \\ (-0.6,-0.6) & (-0.2,-0.6) & (0.2,-0.6) & (-0.6,-0.6) \end{bmatrix} \quad (5.6)$$

O programa implementado gera primeiro a matriz da equação (5.6) e segundo a distorção definida pelo usuário, gera outra matriz com a transformação introduzida na matriz original. A partir desta última gera uma seqüência aleatória de símbolos em número especificado pelo usuário (10000 para as simulações aqui mostradas). A listagem do programa em C encontra-se no Anexo 1.

5.1.1 As Distorções Não-Lineares

Com o programa implementado para geração de símbolos é possível obter constelações QAM-16 sem distorção e com as distorções descritas no capítulo 2 (vide figura 2.6), porém para o presente estudo são considerados apenas 4 casos: sem distorção, colapso das esquinas, rotação da grade e deslizamento da grade. Ditas constelações podem observar-se na figura 5.2. A introdução das distorções é desde o princípio da seqüência.

No caso da distorção por colapso das esquinas, os símbolos situados nesses lugares sofrem um deslocamento horizontal e vertical em direção ao centro da constelação segundo um fator especificado pelo usuário, CCF (vide anexo 1). O valor de CCF escolhido para as simulações é de 0.8.

A distorção de rotação gira a grade, tendo como ponto de referência o centroide da constelação, num determinado ângulo, ROT, sem alterar a perpendicularidade das componentes *i* e *q* dos símbolos. O valor de ROT utilizado é de 20° (vide anexo 1)

Finalmente, no caso da distorção por deslizamento da grade consideram-se dois fatores: um de deslizamento horizontal (HGCF) que desloca a componente *i* à direita ou à esquerda; e, outro de deslizamento vertical (VGCF) que desloca a componente *q* para acima ou para abaixo. O deslizamento é proporcional ao nível do símbolo, ou seja os símbolos que ocupam os níveis maiores são afetados pelo valor integral dos fatores de deslizamento e os símbolos dos níveis inferiores por apenas uma parcela desses fatores, tornando desta maneira o centroide da constelação, ponto de referência do deslizamento. Os fatores escolhidos são: HGCF=0.3, que desliza no máximo em 30% as componentes *i* e VGCF=0.0.

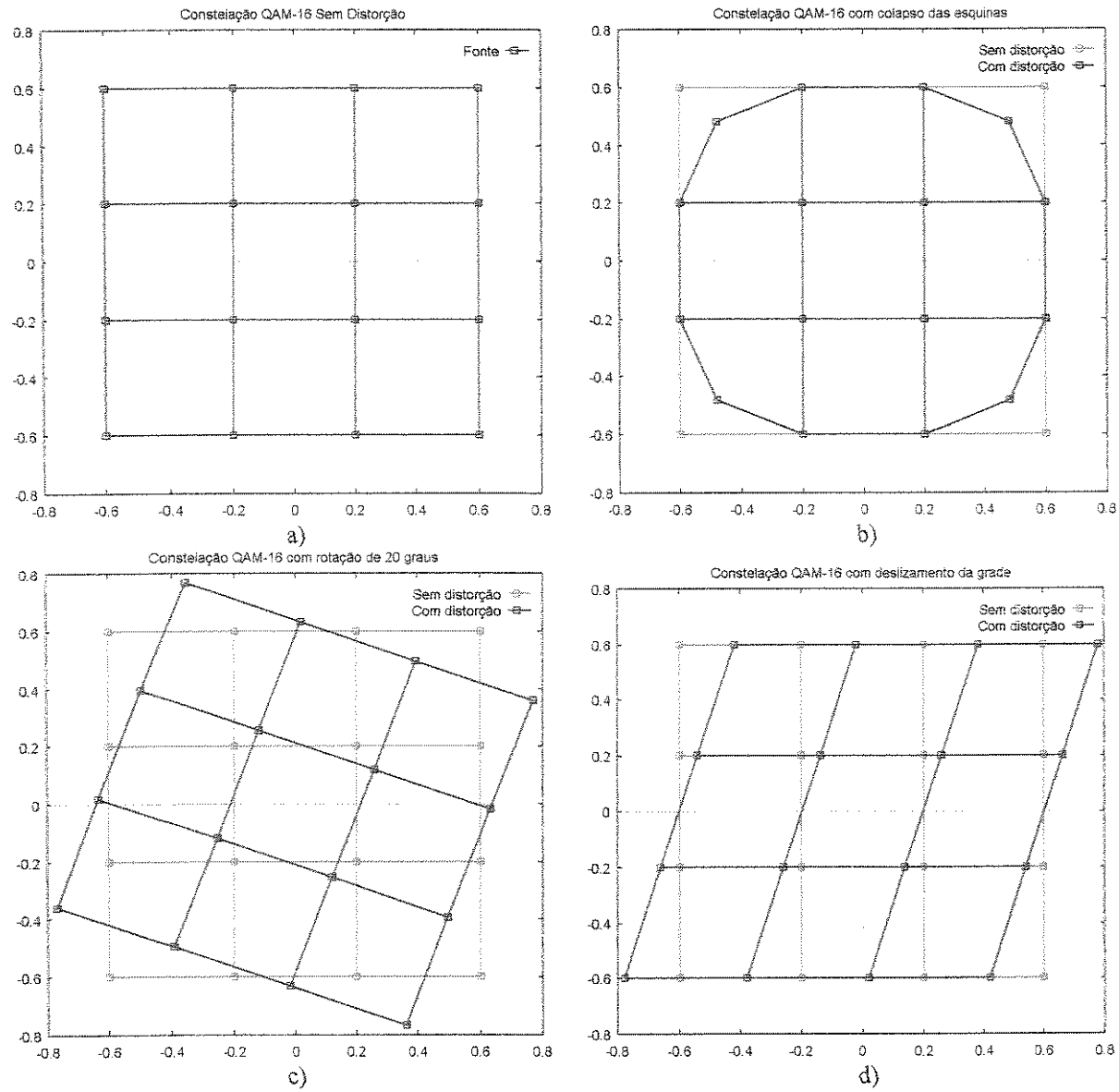


Figura 5.2 *Constelações QAM-16 geradas pelo programa* ***symsrc****:* ***a)*** *sem distorção,* ***b)*** *com colapso das esquinas (vértices),* ***c)*** com rotação da grade, ***d)*** com deslizamento da grade.

5.2 O Canal de Transmissão

O modelo de canal considerado para as simulações é o de duplo caminho, o mais simples dos modelos multicaminho para canais. Nesse modelo, além do símbolo corrente existe uma versão atrasada em τ. A resposta ao impulso do modelo de duplo caminho, cujo diagrama se mostra na figura 5.3, com espaçamento variável, τ, do *tap* pode ser expresso assim:

$$h_{duplo-caminho}(n) = \hat{a}_0\delta(n) + \hat{a}_1\delta(n-\tau) \tag{5.7}$$

onde: $\hat{a}_0$ e $\hat{a}_1$ são coeficientes complexos.

De acordo com a aplicação alguns parâmetros do modelo dinâmico do canal de duplo caminho podem ser fixados, por exemplo, para ajustar este modelo a um enlace fixo de *rádio o coeficiente* a_0, *o atraso* τ *podem ser fixados e o ganho* a_1 *e a fase* ϕ *do tap defasado,* $\hat{a}_1=a_1e^{j\phi}$, mudados de acordo as suas características estatísticas conhecidas. No presente trabalho o atraso τ é fixado ao espaçamento entre os símbolos transmitidos e a origem do tempo é fixada de forma que o primeiro símbolo a chegar tenha um atraso nulo. Para um determinado instante da transmissão o primeiro símbolo a chegar deve ter uma amplitude maior do que a versão atrasada de forma que o primeiro seja reconhecido como o símbolo desejado e o segundo como o de interferência intersimbólica.

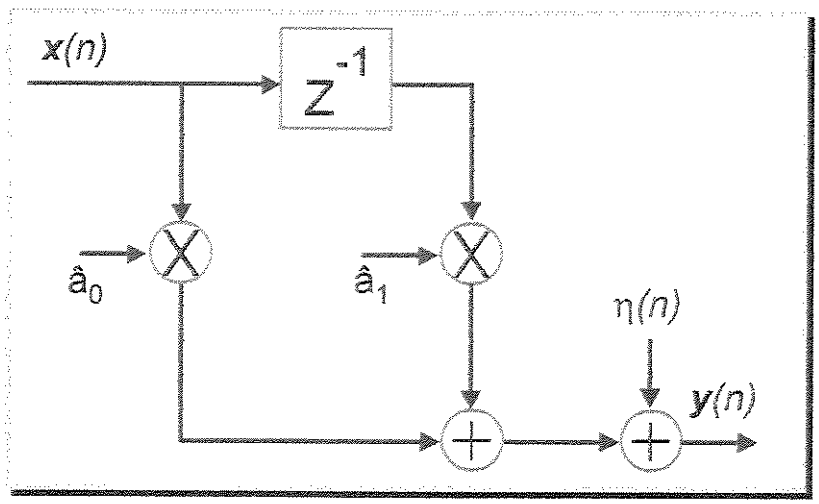


Figura 5.3 Diagrama do modelo de duplo caminho do canal de transmissão. *x(n)* é o sinal de entrada, *y(n)* é o sinal de saída e $\eta(n)$ é o sinal de ruído. z^{-1} é o atraso equivalente ao espaçamento entre símbolos.

A função de transferência do modelo do canal de duplo caminho pode ser expresso da seguinte maneira:

$$\left|H\left(e^{j\omega}\right)\right| = \hat{a}_0 + \hat{a}_1 e^{j\omega} \tag{5.8}$$

Nas simulações é considerado $\hat{a}_0=1$ e $\hat{a}_1=0.1e^{j\pi/2}$. O programa que implementa o modelo mencionado chama-se **awgn** (a listagem pode ser vista no Anexo 2), cuja operação é descrita a seguir. A seqüência de símbolos gerada pelo programa **symsrc**, serve de entrada para o presente módulo. O par de números lidos, o número complexo representando o símbolo, são introduzidos na linha de retardo da estrutura de dois *taps* do canal. Realiza-se então a multiplicação complexa dos coeficientes de cada *tap* com o elemento correspondente da entrada *x(n)* presente na linha de retardo, sendo todos os produtos somados. Neste ponto é necessário fazer o deslocamento da seqüência de entrada e deixá-lo pronto para o seguinte símbolo. Finalmente, acrescenta-se à somatória de produtos obtida, a parcela de ruído gaussiano (descrita na seção posterior), uma para cada componente (*i* e *q*), escalada segundo

a relação sinal por ruído (SNR) proporcionada pelo usuário. Na figura 5.4 observam-se todas as *constelações QAM-16 da figura 5.3 após a passagem pelo modelo de canal de duplo caminho* (são mostrados apenas 1000 amostras da saída do canal).

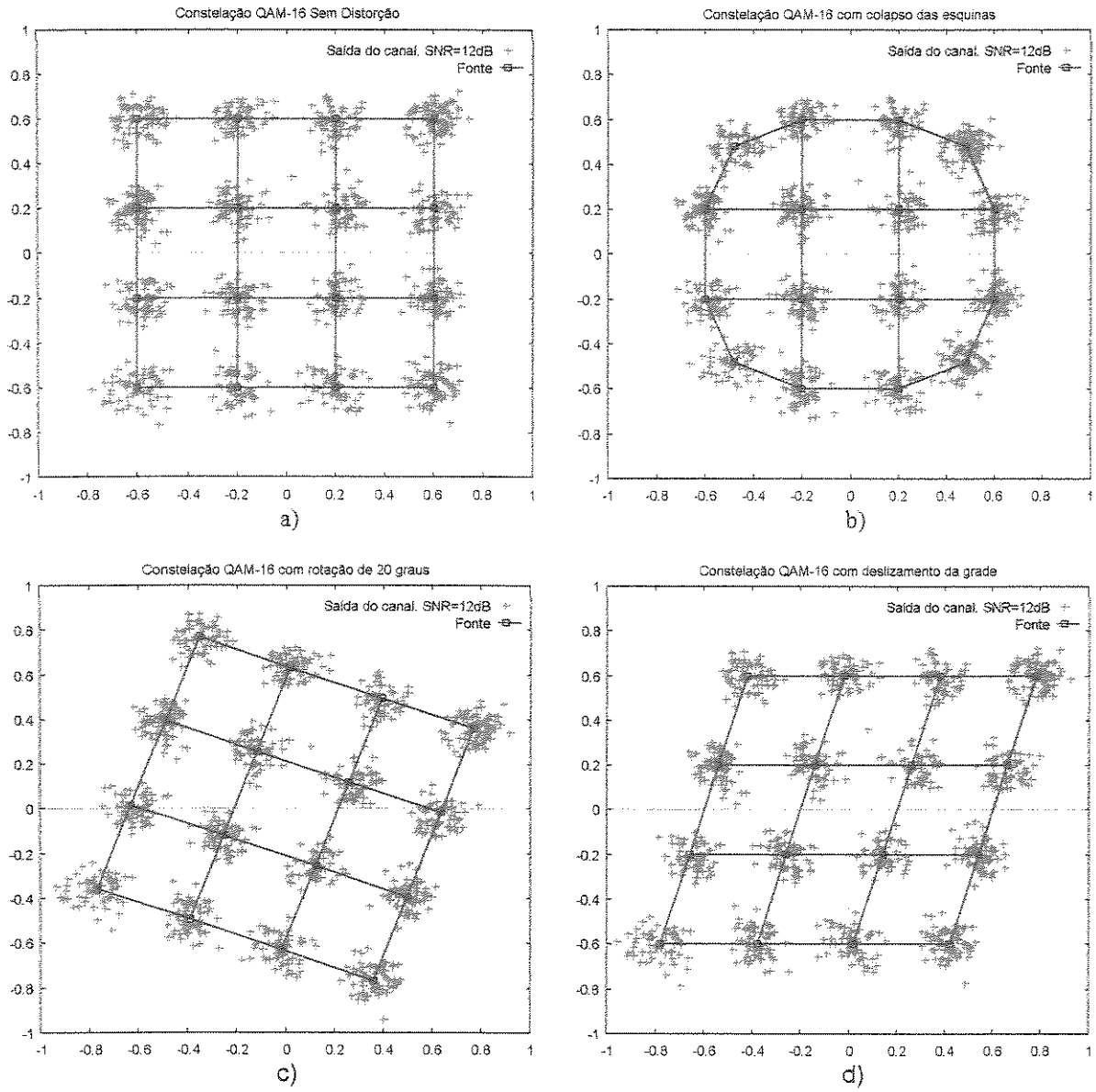


Figura 5.4 Constelações QAM-16 após o canal de duplo caminho para SNR igual a 12dB. **a)** sem distorção, **b)** com colapso das esquinas (vértices), **c)** com rotação da grade, **d)** com deslizamento da grade.

5.2.1 A Geração do Sinal de Ruído

A geração de um número aleatório com densidade normal (gaussiana) a partir de outro com densidade uniforme baseia-se na lei fundamental de transformação das

probabilidades[Pres92]. Sendo x uma variável aleatória com densidade uniforme e $y(x)$ uma *função predeterminada de x, cuja densidade de probabilidade (com média nula e variância* unitária) é:

$$p(y) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{y^2}{2}\right)} \tag{5.9}$$

Um dos métodos de transformação é o de *Box-Muller*[Pres92], que se baseia na versão bidimensional da transformação de probabilidades e cujo algoritmo foi implementado para as simulações do sistema em estudo (vide Anexo 2).

5.3 O Equalizador Convencional

O equalizador convencional utilizado para as simulações é o de Decisão Realimentada (DFE), mostrado no diagrama da figura 5.5, com dois (2) *taps* no caminho direto e um (1) *tap* no caminho de realimentação. A seqüência de entrada, *U(k)* que contém os componentes *i* e *q* dos símbolos, provêm da saída proporcionada pelo módulo **awgn** em número igual a 10000 amostras para cada um dos tipos de distorções da constelação QAM-16 descritos na seção 5.1. O programa que implementa essa estrutura, chamado **dfe** e cuja listagem completa pode ser vista no Anexo 3, gera a seqüência equalizada, *w(k)*, de símbolos.

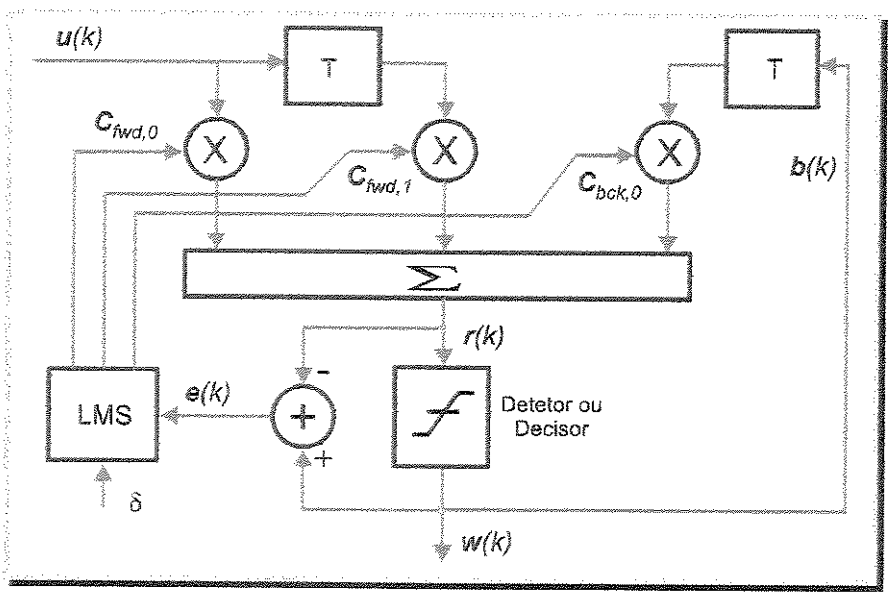


Figura 5.5 Diagrama do Equalizador Convencional com Decisão Realimentada (DFE) de 2 *taps* no caminho direto e 1 no caminho de realimentação.

O algoritmo de ajuste dos coeficientes do equalizador é o LMS e o fator de adaptação, δ, considerado para as simulações e com cujo valor o algoritmo converge é *0.0001*. Para as simulações considera-se a seqüência desejada de símbolos, *d(k)*, igual à seqüência de entrada *U(k)*.

A seguir descrevem-se as etapas do programa implementado:

1. São lidos os dois primeiros números ou os próximos dois que seguem na seqüência, que *representam um símbolo, da saída do módulo* **awgn**. *Esses números são inseridos na* linha de retardo do caminho direto do equalizador. O vetor de saída *r(k)* é iniciado em (0,0).
2. *Realiza-se a multiplicação dos números complexos* $u_i(k)$ (*símbolo de entrada na posição i-*ésima da linha de retardo direta) e do correspondente $C_{fwd,i}$ (coeficiente do *i*-ésimo *tap* do caminho direto), onde *i=0..N-1* (*N é o número de taps no caminho direto do equalizador*). Os produtos são acumulados no vetor *r(k)*.
3. De forma similar à etapa anterior, faz-se a multiplicação de $b_j(k)$ (símbolo na *j*-ésima posição na linha de retardo da realimentação), réplica da saída *w(k)*, e do respectivo coeficiente $C_{bck,j}$ (do *j*-ésimo *tap* do caminho de realimentação), onde *j=0..M-1*, sendo *M* o número de *taps* no caminho de realimentação do equalizador. Os produtos também são acumulados em *r(k)*.
4. Calcula-se o vetor de erro *e(k)* subtraindo a saída do equalizador *r(k)* do símbolo desejado *d(k)*, neste caso *u(k)*.
5. Avalia-se a taxa de erro de símbolos *(SER - Symbol Error Rate)* comparando as regiões às quais pertencem o símbolo original gerado no módulo **symsrc**, *x(k)*, e o símbolo na saída do equalizador, *r(k)*. A SER é avaliada contabilizando o número de vezes que um símbolo é recebido errado e dividido pelo número total de símbolos recebidos até o instante em curso. Entende-se por região a área delimitada pelos limites de decisão existentes entre símbolos consecutivos, no caso da constelação QAM-16 sem distorção, as áreas formadas depois de traçar retas verticais e horizontais em $-A_c$, *0* e A_c, sua implementação é descrita mais adiante.
6. Atualizam-se os coeficientes tanto do caminho direto quanto do caminho de realimentação por meio do algoritmo LMS, vide equações (4.20) e (4.21) respectivamente. Para tanto, os conjugados complexos de *u(k)* (para os *taps* do caminho direto) e de *b(k)* (para os *taps* do caminho de realimentação) são multiplicados pelo vetor de erro *e(k)*. Os resultados desses produtos são multiplicados pelo fator δ e somados aos valores passados para gerar os novos coeficientes.
7. Deslocamento das linhas de retardo e prepara-los para o seguinte símbolo de entrada. Se todos os símbolos gerados na fonte já tiverem sido lidos então o programa termina sua execução, caso contrário volta-se à etapa 1.

A função em C chamada **region** implementa a rotina de determinação da região à qual pertence o símbolo. Seu algoritmo baseia-se no estabelecimento de limites de decisão de acordo com o caso de distorção que está sendo simulado. Na figura 5.6 observa-se a representação gráfica de tais limites, no caso da constelação sem distorção os limites estão determinados pelas retas horizontais e verticais que passam em $-A_c$, *0* e A_c.

Para o caso de distorção por deslizamento da grade os limites são dados pelas retas:

$$q = i_0 \; ; \quad q = \frac{3}{2HGCF}(i - i_0); \quad para\ i_0 \in \{-A_c, 0, A_c\} \tag{5.10}$$

onde: HGCF é o fator de deslizamento horizontal. E no caso de distorção pela rotação da grade pelas seguintes retas:

$$q = \pm \frac{\cos(ROT)}{\text{sen}(ROT)}(i - i_0); \quad para\ i_0 \in \{-A_c, 0, A_c\} \tag{5.11}$$

onde: *ROT* é o angulo da rotação.

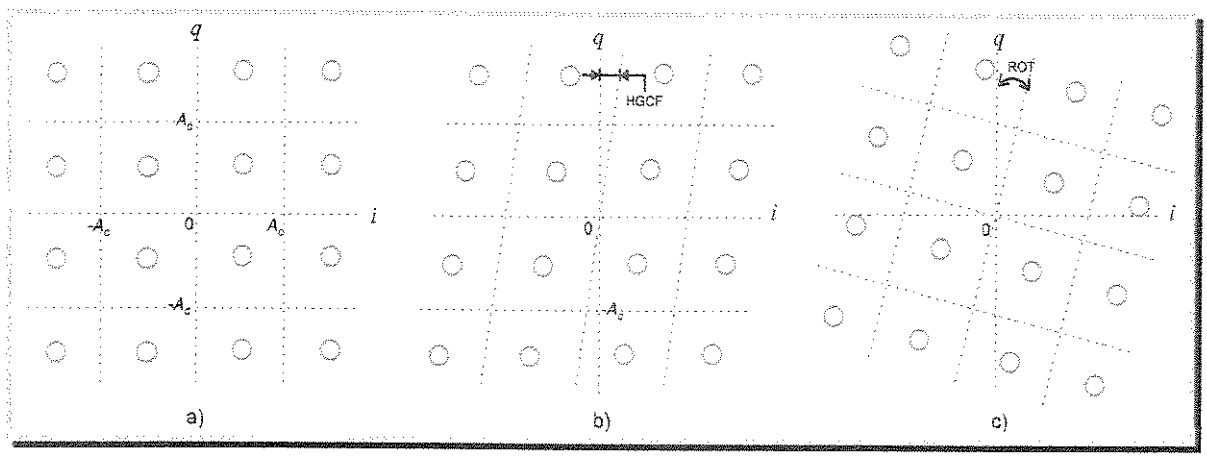


Figura 5.6 Limites de decisão implementados na rotina **region** para a determinação da região do símbolo na constelação QAM-16. **a)** sem distorção, **b)** com deslizamento da grade, **c)** com rotação da grade.

5.4 O Neuro-Equalizador

A estrutura do neuro-equalizador utilizado para as simulações é a da figura 5.7. Essa estrutura, DFE baseado em SOM, é um caso particular do equalizador mostrado na figura 4.13 e tem 2 *taps* no caminho direto e 1 *tap* no caminho da realimentação.

De forma similar ao caso do equalizador convencional, esta estrutura recebe a seqüência de símbolos (10000) do módulo **awgn** em *u(k)* para os diferentes tipos de distorções mencionadas em 5.1. O algoritmo de adaptação dos coeficientes é o LMS com o fator δ=*0.0001*.

A diferença principal em relação ao equalizador convencional utilizado nas simulações (vide figura 5.5) é a utilização como seqüência desejada, ***d(k)***, a saída da rede neural, *w(k)*. Na rede neural, o fator de ganho α utilizada para a célula ganhadora é *0.2* e o fator β utilizado para as células na vizinhança topológica da célula ganhadora é *0.01*. Os fatores de deslocamento e amplificação, V_{dc} e V_g, são 0 e 1 respectivamente, para todas as células neurônio. O programa que implementa o equalizador DFE baseado em SOM chama-se **dfe_knn** e a listagem completa está no Anexo 4.

As etapas da operação do programa **dfe_knn** são os mesmos do equalizador convencional com exceção de realizar a cada novo símbolo os seguintes passos:

3a Calcula-se a distância dos vetores de pesos internos ao vetor (símbolo) de entrada e determina-se a célula-neurônio ganhadora do processo competitivo.

3b Atualiza-se o peso da célula ganhadora segundo os fatores de α, V_{dc} e V_g.

3c Atualizam-se os pesos das células dentro da vizinhança topológica da ganhadora utilizando os fatores β, V_{dc} e V_g.

4. Calcula-se o vetor de erro *e(k)* subtraindo a saída do equalizador *r(k)* do símbolo na saída da rede neural, *w(k)*.

A função **region** é a mesma que no caso do equalizador convencional.

O custo computacional introduzido pelo uso da rede neural, comparado ao equalizador convencional, é igual ao número de células-neurônio pelo número de operações aritméticas (multiplicação, soma, subtração, valor absoluto e comparação) utilizadas para a atualização dos pesos internos.

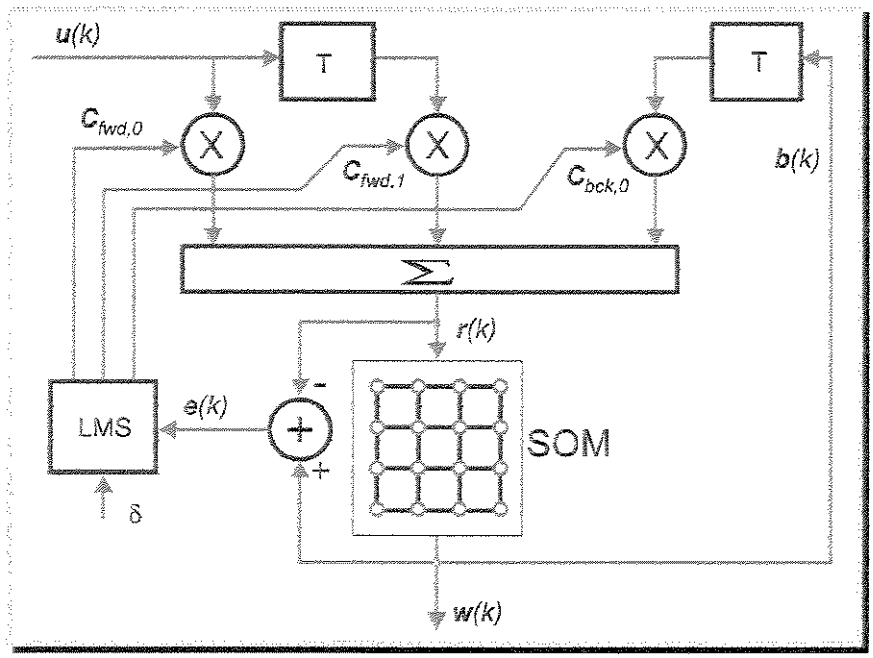


Figura 5.7 Diagrama do Equalizador Decisão Realimentada baseado em SOM (DFE-SOM) com 2 *taps* no caminho direto e 1 no caminho de realimentação.

5.5 Simulações e Resultados

Nesta seção mostram-se os resultados das simulações realizadas dos dois tipos de equalizador considerados, o DFE convencional e o DFE baseado em SOM, considerando 4 casos de constelações: sem distorção, com colapso das esquinas, com rotação da grade e com deslizamento da grade. As distorções são introduzidas ao sistema simulado desde o inicio da seqüência de símbolos.

São mostrados dois tipos de resultados, as constelações QAM-16 na saída do equalizador nos primeiros 500 símbolos e nos 500 últimos símbolos introduzidos de um total de 10000 e as taxas de erro de símbolos (SER). No caso dos gráficos das constelações, são mostrados como referência a saída do gerador de sinais e a saída do canal de transmissão.

Sendo que o objetivo principal do presente trabalho é o projeto da rede neural que dá assistência ao equalizador, considera-se apenas um caso dos parâmetros tanto da rede neural como do equalizador em si. Consideram-se $\alpha=0.2$, $\beta=0.01$, $V_{dc}=0$, $V_g=1$ e $\delta=0.0001$. A relação sinal ruído, *SNR=12 dB*. Os equalizadores, tanto no caso convencional quanto no baseado em SOM, têm 2 *taps* no caminho direto e 1 *tap* no caminho de realimentação. Nos

trabalhos prévios de Kohonen e Raivio [Koh90a, Koh92, Rai91] pode ser encontrado um estudo para a *busca dos parâmetros antes mencionados.*

Na seqüência mostram-se os resultados citados:

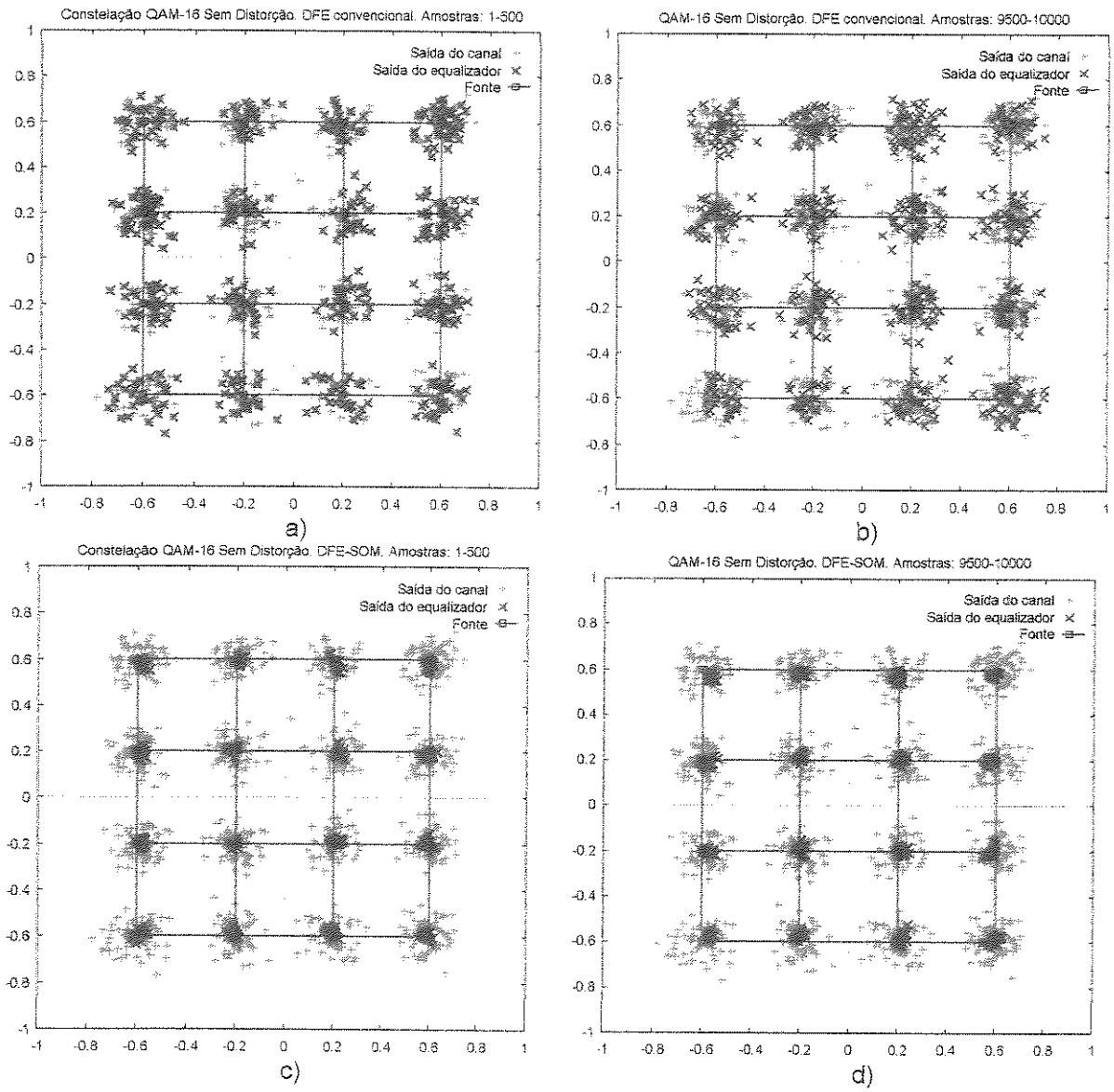


Figura 5.8 Saída dos equalizadores DFE convencional {a) e b)} e baseado em SOM {c) e d)}, para o caso da constelação QAM-16 sem distorção. *SNR=12dB*, α*=0.2*, β*=0.01*, V_{dc}*=0*, V_g*=1* e δ*=0.0001*. O equalizador é de 2 *taps* no caminho direto e 1 *tap* no caminho da realimentação.

Na figura 5.8 podem ser observadas as constelações de sinais na saída do equalizador, ***w(k)***, para o caso de não introdução de distorções na constelação. Nas figuras 5.8a e 5.8b têm-se as respostas do equalizador DFE convencional e nas figuras 5.8c e 5.8d as respostas do equalizador DFE baseado em SOM.

Observa-se que na resposta da estrutura DFE-SOM a dispersão dos símbolos

entorno do ponto ideal é menor do que na estrutura DFE convencional.

Na fase final do treinamento da estrutura DFE-SOM (figura 5.8d) percebe-se uma leve orientação dos símbolos ao centroide da constelação. Como já mencionado no capítulo 4, *nas expressões* (4.26) e (4.27), *tais orientações podem ser corrigidas pelos fatores* V_{dc} *e* V_g.

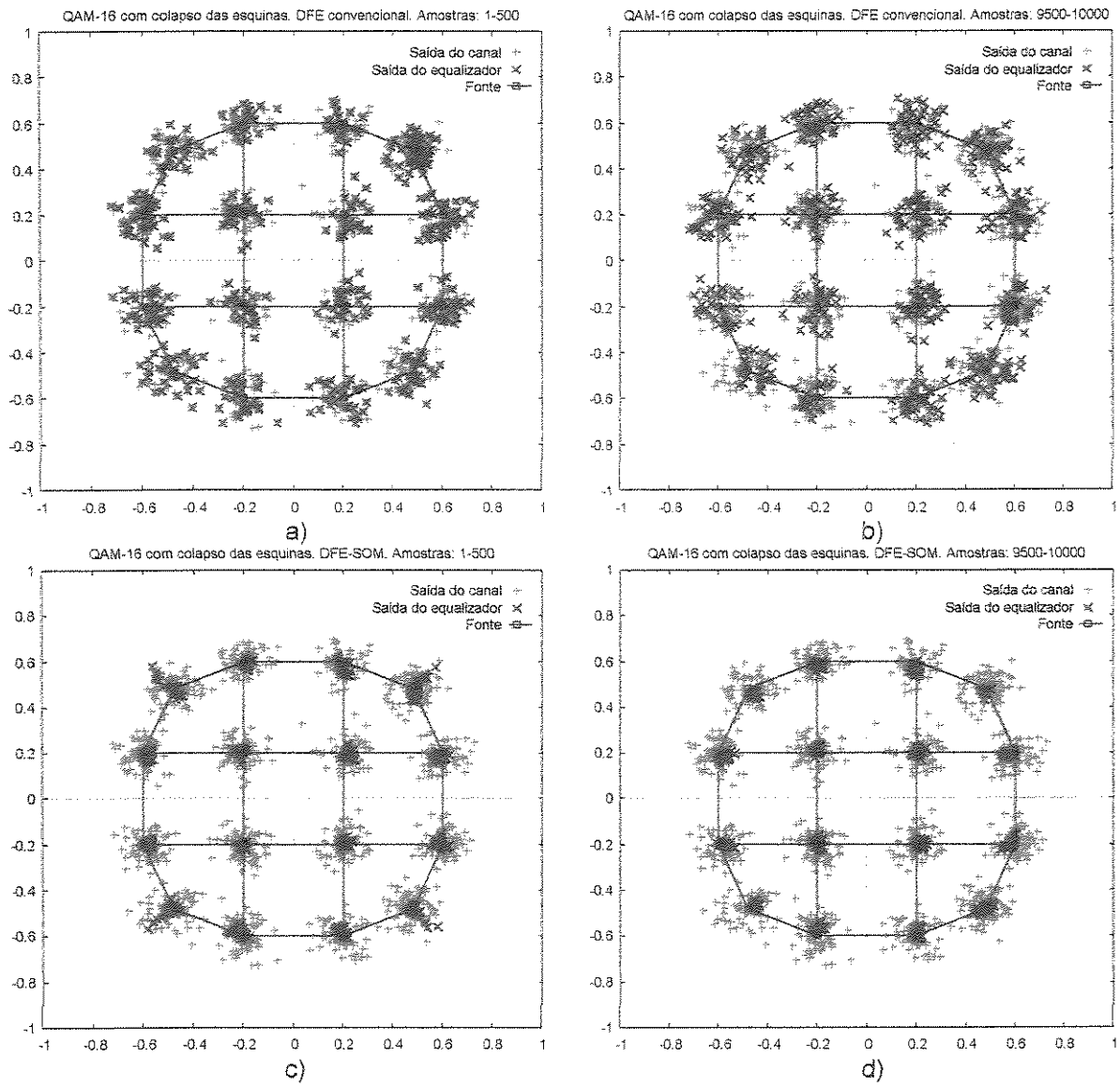


Figura 5.9 Saída dos equalizadores DFE convencional {a) e b)} e baseado em SOM {c) e d)}, para o caso da constelação QAM-16 com colapso das esquinas. *SNR=12dB,* α*=0.2,* β*=0.01,* V_{dc}*=0,* V_g*=1,* δ*=0.0001 e CCF=0.8.* O equalizador é de 2 *taps* no caminho direto e 1 *tap* no caminho da realimentação.

Os resultados para as estruturas de equalização DFE convencional (figuras 5.9a e 5.9b) e DFE baseado em SOM (figuras 5.9c e 5.9d) mostram a constelação QAM-16 nas suas saídas, para o caso de distorção por colapso dos símbolos das esquinas.

De forma similar ao caso da constelação sem distorção, observa-se que na

resposta da estrutura DFE-SOM a dispersão dos símbolos em torno do ponto ideal é menor do *que na estrutura DFE convencional.*

Na fase inicial do treinamento da estrutura DFE-SOM (figura 5.9c) observa-se a *migração dos pesos internos das células-neurônio das esquinas para os novos valores de* referência.

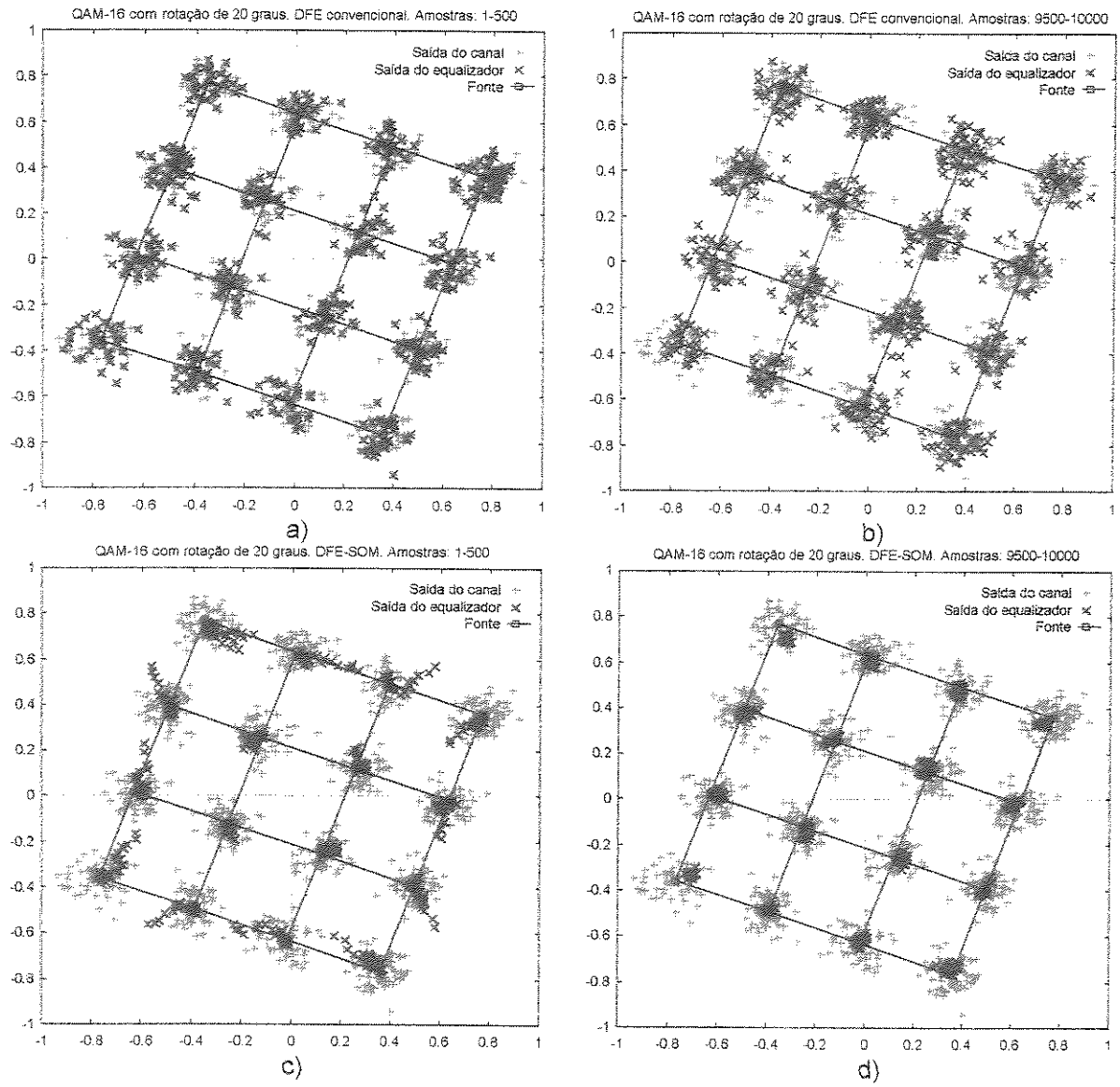


Figura 5.10 Saída dos equalizadores DFE convencional {a) e b)} e baseado em SOM {c) e d)}, para o caso da constelação QAM-16 com rotação de 20° da grade. *SNR=12dB*, α*=0.2*, β*=0.01*, V_{dc}*=0*, V_g*=1*, δ*=0.0001 e ROT=20°*. O equalizador é de 2 *taps* no caminho direto e 1 *tap* no caminho da realimentação.

Na figura 5.10 podem ser observadas as constelações de sinais na saída dos equalizadores DFE convencional (vide figuras 5.10a e 5.10b) e DFE baseado em SOM (vide figuras 5.10c e 5.10d) quando da introdução de distorção de rotação da grade em 20°.

Observa-se na fase final de treinamento, de que a resposta da estrutura DFE-SOM, *a dispersão dos símbolos em torno do ponto ideal é menor do que na estrutura DFE* convencional.

Ainda na fase final do treinamento da estrutura DFE-SOM (figura 5.10d) observa-se uma leve orientação dos símbolos das esquinas ao centroide da constelação. Isto pode ser *corrigido pelos fatores* V_{dc} *e* V_g.

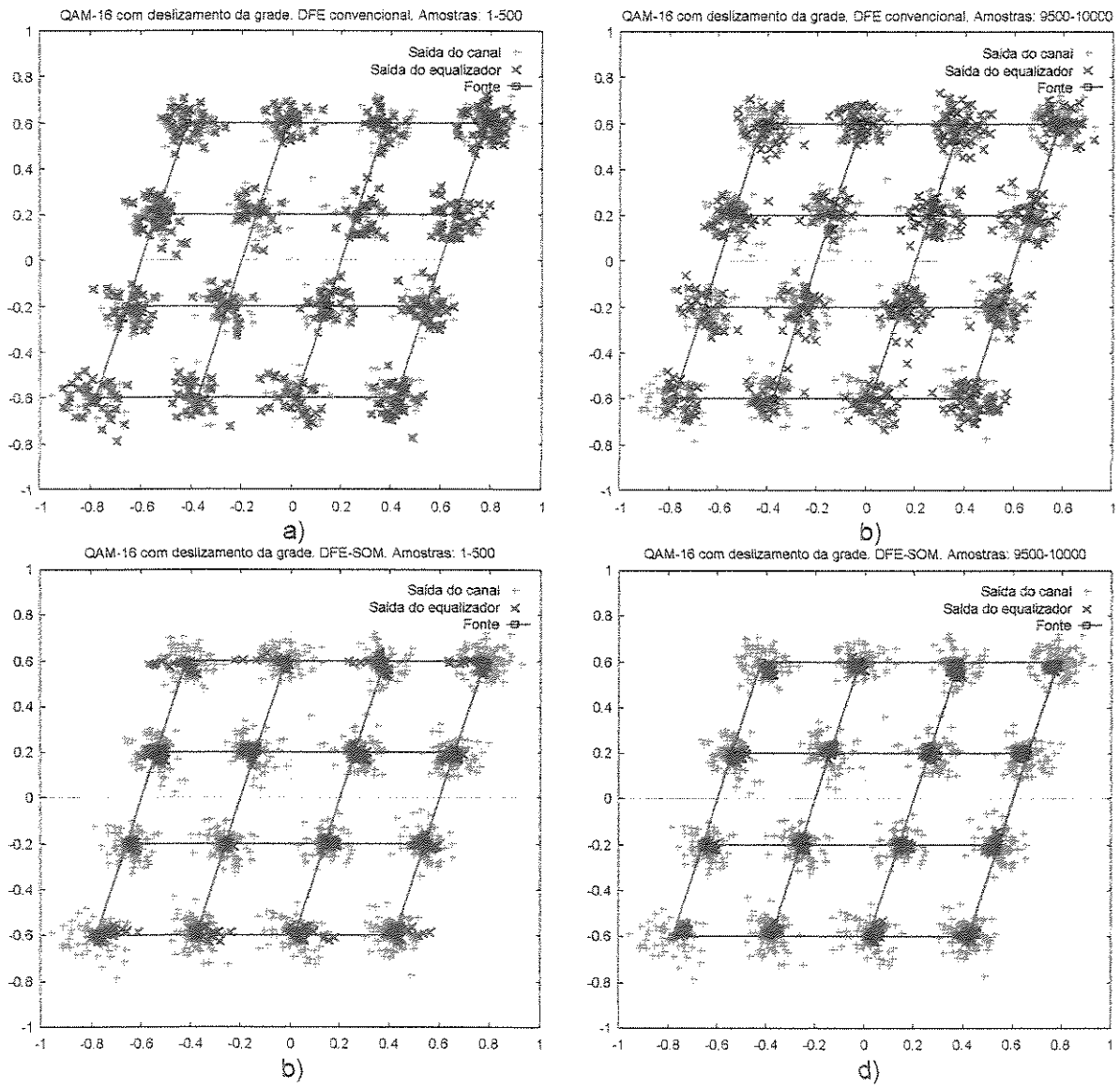


Figura 5.11 Saída dos equalizadores DFE convencional {a) e b)} e baseado em SOM {c) e d)}, para o caso da constelação QAM-16 com deslizamento da grade. *SNR=12dB,* α*=0.2,* β*=0.01,* V_{dc}*=0,* V_g*=1,* δ*=0.0001, HGCF=30% e VGCF=0%.* O equalizador é de 2 *taps* no caminho direto e 1 *tap* no caminho da realimentação.

Os resultados para as estruturas do equalizador DFE convencional (figuras 5.11a e

5.11b) e DFE baseado em SOM (figuras 5.11c e 5.11d) mostram a constelação QAM-16 nas *suas saídas, para o caso de distorção por deslizamento da grade.*

De forma similar ao caso da constelação com outras distorções, observa-se que a *resposta da estrutura DFE-SOM, na fase final do treinamento, a dispersão dos símbolos em* torno do ponto ideal é menor do que na estrutura DFE convencional.

Na fase inicial do treinamento da estrutura DFE-SOM (figura 5.11c) observa-se a migração dos pesos internos das células-neurônio das esquinas para os novos valores de referência.

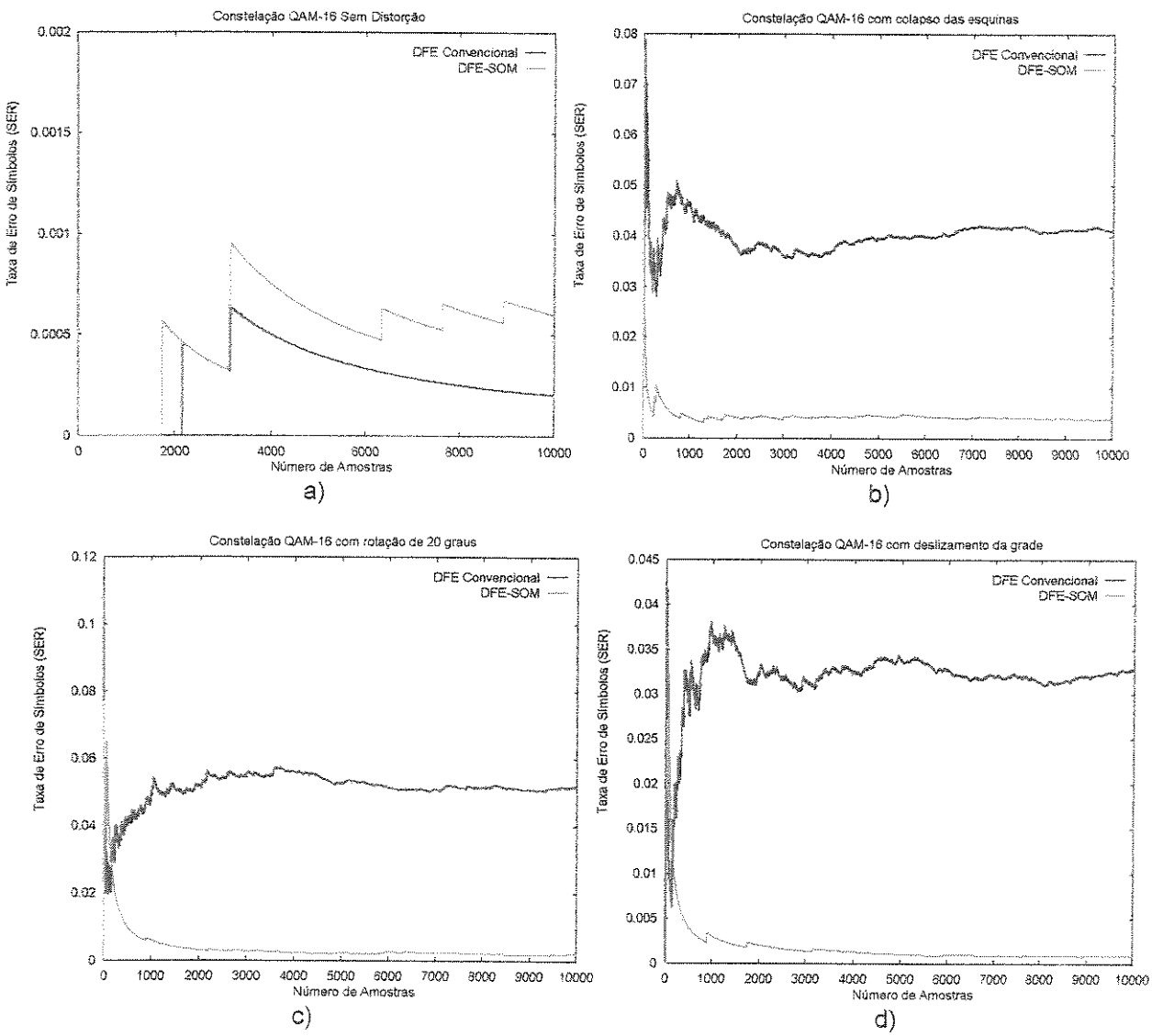


Figura 5.12 Taxa de erro de símbolos (SER) para as estruturas de equalização convencional e baseado em SOM. *SNR=12dB, α=0.2, β=0.01, V_{dc}=0, V_g=1, δ=0.0001, CCF=0.8, HGCF=30% e VGCF=0%.* Constelações QAM-16 **a)** sem distorção, **b)** com colapso dos símbolos das esquinas, **c)** com rotação da grade **d)** com deslizamento da grade.

Na figura 5.12 observam-se as curvas da taxa de erro de símbolos (SER) obtidas das simulações das estruturas de equalizador DFE convencional e DFE baseado em SOM para o caso da constelação a) sem distorção, b) com colapso das esquinas, c) com rotação da grade e d) com deslizamento da grade.

Para os casos de distorção na constelação QAM-16 considerados, de forma geral observa-se que as taxas de erro de símbolos proporcionadas pela estrutura DFE-SOM é superior ao da estrutura DFE convencional, com exceção do caso de não introdução de distorções.

Os valores SER iniciais, após a introdução da distorção, observam-se altas taxas de erro para o caso DFE-SOM. Isto se deve ao fato dos pesos internos das células-neurônio estarem começando a se adaptar aos novos valores de referência contidos nos símbolos da constelação distorcida.

Capítulo 6:

Implementação em *Hardware* - Projeto do ASIC

Neste capítulo apresenta-se o projeto do Circuito Integrado de Aplicação Específica *(ASIC - Application Specific Integrated Circuit)* de uma rede neural, que segue o modelo de Kohonen, com 4 células-neurônio, para ser aplicado na equalização adaptativa de sinais modulados segundo o esquema QAM-16, cujos resultados da implementação em *software* foram apresentados no capítulo 5.

Devido a limitações econômicas e de área de silício, foi possível integrar apenas 4 céiulas-neurônio (nós processadores) em um único *chip*, no entanto, como o seu projeto é modular é possível formar estruturas de maior nível como QAM-16.

O protótipo do ASIC foi fabricado dentro do Programa de Fabricação de Circuitos Integrados da FAPESP. Nesse programa as coordenações técnicas são realizadas diretamente entre o projetista e os centros PMU (Projeto Multi-Usuário - *MPW centers*) sejam da *EUROPRACTICE* (Béigica, Alemanha e Noruega) ou *CMP-TIMA* (França), os quais fazem a verificação prévia do projeto antes de ser enviado à *foundry*. A tecnologia de fabricação foi a CMOS 0.8μm da *foundry* AMS *(Austria Mikro Systeme)*.

6.1 Representação e Metodologia de Projeto

A representação do projeto de um sistema digital, freqüentemente é classificada pelo seu domínio e em cada domínio por seu nível de abstração[Cam92, Nag92]. Usualmente consideram-se três domínios: o comportamental (funcional), o estrutural (topológico) e o físico (geométrico). A representação das descrições de um projeto no domínio comportamental realiza-se com uma linguagem de descrição de *hardware*, no domínio estrutural com *netlists* e no domínio físico com *layouts*.

Os níveis de abstração são: de arquitetura, RTL, lógico e de dispositivo. Os domínios e níveis de abstração mencionados podem ser representados no Diagrama Y[Cam92], como é mostrado na figura 6.1.

As descrições arquiteturais estão no nível de abstração mais elevado e mostram o que os módulos fazem e não o como. Este nível de descrição é independente da tecnologia. As descrições RTL, estão no nível seguinte de abstração, mostrando o sistema em termos de unidades funcionais, registradores, multiplexadores. Diferente do arquitetural, neste nível tem-se noção da estrutura e temporização do circuito. Este nível é também independente da tecnologia. No último nível de abstração, estão as descrições do nível lógico, onde a implementação do projeto é descrito por meio de uma Rede Booleana. Estas descrições, além de manterem a arquitetura derivada do RTL, também mostram as implementações das funções booleanas. Apesar de representar tais descrições numa tecnologia genérica, normalmente elas dependem de alguma tecnologia em particular.

Na metodologia de projeto *top-down* o projetista começa modelando o sistema no

nível mais elevado de abstração e a partir deste, por meio da síntese, desce até chegar no *nível de abstração mais baixo, o de dispositivo. Costumeiramente, este processo começa com* representações do projeto no domínio comportamental, passando pelo domínio estrutural até *chegar no domínio físico. O diagrama da figura 6.2 mostra o fluxo de projeto nessa* metodologia. Na metodologia *bottom-up* o processo é inverso.

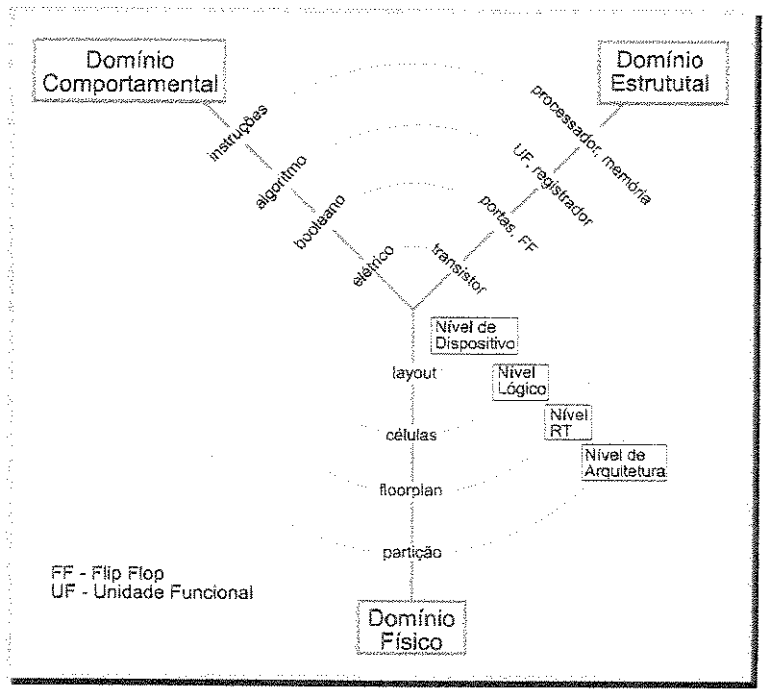


Figura 6.1 Representação dos domínios e níveis de abstração de projeto no Diagrama Y

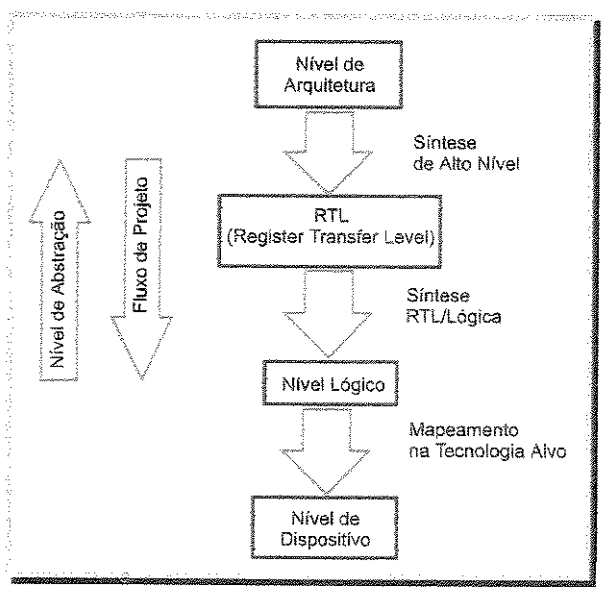


Figura 6.2 Fluxo de projeto na metodologia top-down.

A seguir descrevem-se os passos da metodologia empregada para o projeto do ASIC, sendo *top-down* para o projeto lógico e *bottom-up* para a elaboração do layout:

1. Definição da arquitetura do circuito e estabelecimento dos blocos funcionais, do nível topo e da hierarquia.
2. Descrição RTL em VHDL ou captura esquemática dos blocos componentes do ASIC. Aqui é o começo do projeto lógico.
3. Primeira verificação da validade das descrições VHDL do projeto através de simulações funcionais. Caso as especificações não serem atendidas volta-se ao passo 2.
4. Síntese lógica a partir da descrição VHDL para uma tecnologia genérica.
5. Otimização do ASIC em área de silício e temporização, logo mapeamento na tecnologia objeto.
6. Segunda verificação da validade do projeto por meio de simulações, porém desta vez incluindo os fatores temporização, *fan-out*, *fan-in*, etc. Caso as especificações iniciais não serem atingidas, volta-se para o passo 4 ou mesmo para o passo 2.
7. Elaboração do *layout* de cada um dos blocos do circuito através de ferramentas de síntese automática de *layout*. Aqui é o começo da elaboração do *layout*.
8. Verificação em cada um dos blocos das regras de projeto *(DRC - Design Rules Check)* e elétricas *(ERC - Electrical Rules Check)*. Confrontação do *layout* originado com a fonte lógica (os *netlists*) pelo processo conhecido como LVS *(Layout Vs. Schematic)*.
9. Conformação do *layout* do circuito inteiro.
10. Verificação no circuito inteiro de DRC, ERC e LVS.
11. Extração dos parâmetros parasitas do *layout*, segundo o modelo resumido *(lumped)* ou distribuído *(distribuited)* desses parâmetros.
12. Acrescentar os parâmetros extraídos no passo anterior no modelo de simulação original do circuito através do processo conhecido como de *backannotation*
13. Última verificação da validade do projeto através de simulações pós-*layout* que incluem os parâmetros parasitas. Volta-se para o passo 7 ou para os passos 4 ou 2 em caso das especificações não serem cumpridas.

No auxílio do projeto são utilizadas várias ferramentas da *Mentor Graphics Corp.*, assim têm-se: para a especificação em VHDL o *Design Architect*^{TM} [Men95f]; para a síntese lógica o *Autologic II*^{TM} [Men95g]; para as simulações iniciais, após a síntese lógica e pós-layout o *QuickSim II*^{TM} [Men95h]; e, para a elaboração do *layout*, extração de parâmetros parasitas, DRC, ERC e LVS, o conjunto de ferramentas do *IC Station*^{TM} [Men95i] e *IC Verification*^{TM} [Men95j].

Nossa experiência da primeira implementação em ASIC de uma rede neural de Kohonen[Jar97, Mac96], sob o ponto de vista de ferramentas no auxílio ao projeto e também sob a ótica de projeto de sistemas digitais, foi um fator de bastante importância para o projeto do ASIC do presente trabalho.

6.2 Arquitetura da Rede Neural

A rede neural, como já mencionado, consiste de 4 células-neurônio cada uma correspondendo a um símbolo de uma região *(cluster)* de 4 símbolos próximos da constelação QAM-16. Portanto, a estrutura do ASIC consiste de 4 nós, cada um deles sendo um processador dedicado que implementa o modelo de Kohonen, uma unidade WTA *(Winner-Takes-All)*, uma unidade de decodificação de endereços para os nós e um sistema de barramento. Na figura 6.3 mostra-se um diagrama simplificado da arquitetura da rede neural.

Primeiro a rede neural é configurada armazenando os pesos iniciais e os

parâmetros do algoritmo, nos registradores internos de cada célula. Quando um vetor-símbolo, *conjunto de dois números correspondendo às componentes i e q, chega à rede, cada uma das* células determina uma distância, baseado na distância *Manhattan* e sinaliza à unidade WTA da *disponibilidade dessa distância calculada. O WTA determina qual o menor dos valores, através* de uma árvore de decisão binária e armazena na memória qual a célula vencedora, *disponibilizando o dado da distância menor para outros grupos (clusters) de células*, sinalizando-os desta condição. A seguir o WTA aguarda pelo sinal dos outros *clusters* e somente recebe dito sinal se alguma das células do seu grupo for vencedora. A célula vencedora recebe então o sinal que indica essa condição, que por sua vez, comunica às células vizinhas, sejam de seu grupo ou de outros grupos, e aciona o processo de atualização dos pesos internos, $w_{j,i}$ e $w_{j,q}$, segundo a equação (4.26). As células que estiverem na condição de vizinhas imediatas recebem o sinal da célula vencedora e também atualizam seus pesos, porém segundo a equação (4.27). As células que não estiverem dentro da vizinhança, N_{j^*}, não sofrem alteração nos seus pesos (vide equação 4.28).

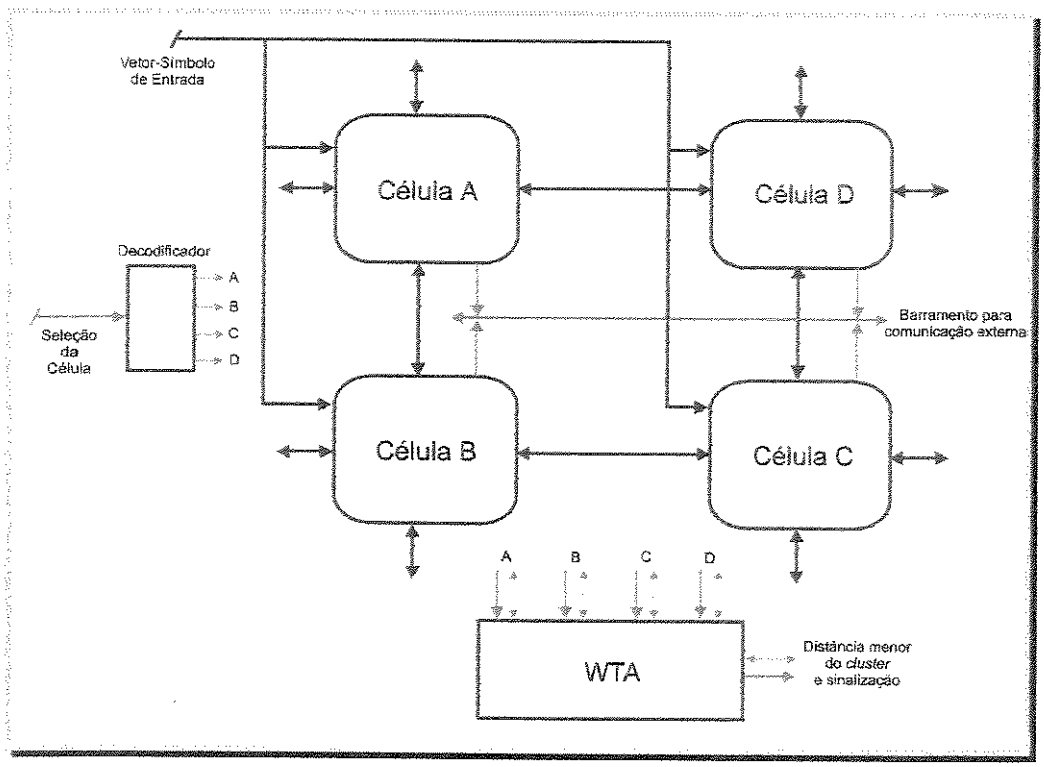


Figura 6.3 Arquitetura básica da Rede Neural implementada.

6.3 Aritmética e Representação Numérica

Para uma implementação seja digital ou analógica do algoritmo de Kohonen, as entradas $\boldsymbol{U}=(u_1, u_2, \ldots, u_n)$ e os pesos internos $\boldsymbol{W}=(w_1, w_2, \ldots, w_n)$, onde n é um número finito, estarão sempre limitadas de forma que $u_{min} \leq |u_j| \leq u_{max}$ e $w_{min} \leq |w_j| \leq w_{max}$. Pode-se considerar que os valores de u_j e w_j são de alguma forma quantizados por uma função $Q(\bullet)$, definindo-se um passo de quantização, pq, expresso assim:

$$pq = \frac{u_{\max} - u_{\min}}{2^{nb} - 1} \tag{6.1}$$

onde: *nb* é o número de bits da representação binária de ***U*** e ***W***. Os efeitos de sua quantização na realização em hardware do SOM foram investigados por Thiran et al.[Thi94, Jar97]. Para o caso bidimensional de ***U*** e ***W***, como o do presente trabalho, a função de quantização é ilustrada na figura 6.4.

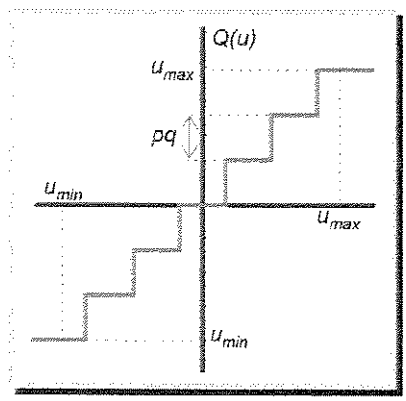


Figura 6.4 Quantização do vetor $U=(u_i, u_q)$

A função *Q(u)* ilustrada na figura 6.4 representa um mapeamento dos componentes de ***U*** em múltiplos do passo de quantização *pq*.

Segundo Thiran et al., há três fatores importantes que influem no comportamento da rede após a quantização: o número mínimo de bits, a função $\alpha(t)$ e a função de vizinhança topológica. Na aplicação do presente trabalho, como já visto, as duas últimas funções são constantes, restando somente o fator número de bits. Para uma rede com *LxL* células, supondo-se uma distribuição de probabilidade uniforme sobre um espaço quadrado *pxp*, obtem-se que o número mínimo de bits necessários para quantizar os pesos é expresso por:

$$nb > \log_2\left(1 + \frac{2L}{p}\right) \tag{6.2}$$

Para o caso particular da presente aplicação, QAM-16, *L=4* e considerando o quadrado *pxp* igual à variança do ruído (σ_η^2) derivada da contribuição da relação sinal para ruído (vide equação 2.7) máxima considerada para as simulações (8dB) e considerando a variança do sinal, unitária, temos:

$$nb > \log_2\left(1 + \frac{2.(4)}{\frac{\sigma_u^2}{10^{\frac{SNR}{10}}}}\right) \approx 6; \quad SNR = 8dB; \quad \sigma_u^2 = 1 \tag{6.3}$$

As implementações de circuitos operando em ponto flutuante demandam grandes áreas de silício motivo pelo qual foi decidido utilizar a representação numérica de ponto fixo. Considerando o cálculo em (6.3) escolheu-se para a representação numérica a seguinte: 14 bits para a interna e 12 bits para a externa, incluindo o bit de sinal (vide figura 6.5).

Com isto e aplicando a equação (6.1) para o caso de *nb=12*, tem-se o valor de passo de quantização, *pq*, de 4.88×10^{-4}.

Um número, *num*, representado no sistema numérico decimal de ponto fixo pode ser expresso assim:

$$num = \sum_{g=0}^{nbi-1} \left(d_{nbf+g} \cdot v^{g} \right) + \sum_{g=1}^{nbf} \left(d_{nbf-g} \cdot v^{-g} \right) \qquad (6.4)$$

onde: *nbi* é o número de bits da parte inteira, *nbf* é o número de bits da parte fracionária, *v* é o *radix* do sistema numérico que para o binário é 2, d_g é um digito binário (0 ou 1). Para os casos de representação em 12 e 14 bits, *nbi=0* e *nbi=2*, respectivamente e *nbf=11* bits para ambos os casos.

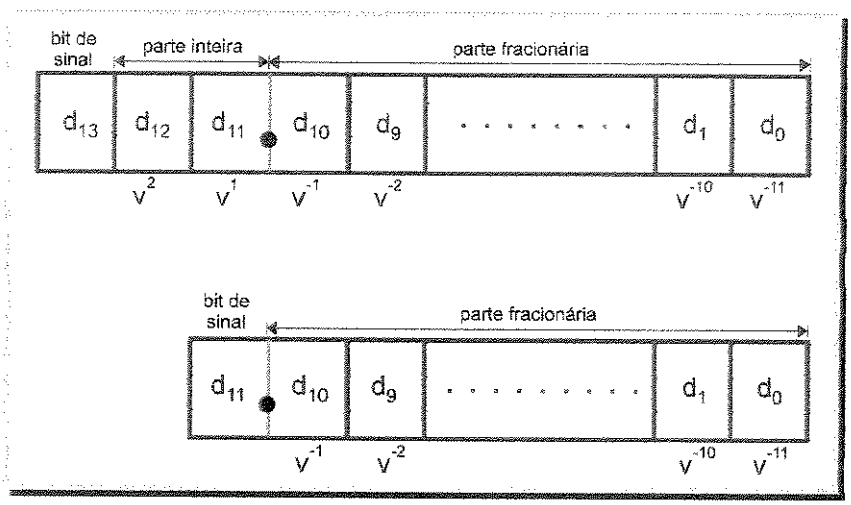


Figura 6.5 Representação numérica ponto fixo com sinal, de 12 e 14 bits.

Na seção 5.1 foi mencionado que o domínio das componentes, u_i e u_q, do vetor de entrada é delimitado por (-1 , 1) e pela característica do modelo de Kohonen, no qual os pesos internos sempre tentam seguir aos valores dos vetores de entrada, o domínio dos pesos pode ser expresso assim:

$$|u_i|, \quad |u_q| < u_{max} = 1; \quad |w_i|, \quad |w_q| < w_{max} = 1 \qquad (6.5)$$

Para o cálculo da distância é necessário avaliar o valor absoluto da diferença entre o vetor de entrada e o peso interno, portanto:

$$|u_i - w_i| < 2u_{\max}; \quad |u_i - w_i| + |u_q - w_q| < 4u_{\max} \tag{6.6}$$

A representação numérica interna possui 2 bits na parte inteira, portanto das equações (4.26) e (4.27) pode-se deduzir que:

$$|V_{dc} + V_g u_i - w_i| < 4u_{\max}; \quad |V_{dc} + V_g u_q - w_q| < 4u_{\max} \tag{6.7}$$

Onde: os valores de V_{dc} e V_g devem ser tais que se cumpra a expressão (6.7).

O domínio das constantes α e β, é [Koh90b] :

$$0 < \alpha < 1; \quad 0 < \beta < 1 \tag{6.8}$$

Para a determinação do domínio de V_g e V_{dc} pode-se seguir o seguinte raciocínio: considerem-se dois símbolos consecutivos $C_1=(C_{1,i}\ ,\ C_{1,q})$ e $C_2=(C_{2,i}\ ,\ C_{2,q})=(C_{1,i}+A_c\ ,\ C_{2,i}+A_c)$; considere-se também a restrição de que a componente, *i* ou *q*, de um determinado símbolo, não pode ultrapassar o ponto meio entre esta e a componente respectiva do símbolo consecutivo, assim:

$$C_{1,i} V_g < C_{2,i} - \frac{A_c}{2}; \quad C_{1,i} V_g < C_{1,i} + A_c - \frac{A_c}{2} \tag{6.9}$$

portanto:

$$V_g < 1 + \frac{A_c}{2 C_{1,i}} \tag{6.10}$$

Como mencionado na seção 4.6.1, os símbolos podem sofrer um deslocamento em sentido ao centroide da constelação, portanto V_{dc} e V_g devem ser valores tais que o símbolo seja deslocado em sentido oposto ao centroide. Para o caso de QAM-16 existem dois casos de valores para $C_{1,j}$ em cada quadrante da constelação, considerando símbolos do primeiro quadrante, $i>0$ e $q>0$, se $C_{1,j} = A_c/2$ o valor de $V_g < 2$ e se $C_{1,j} = 3A_c/2$ o valor de $V_g < 4/3$. De maneira análoga pode-se analisar para o caso dos outros quadrantes, portanto:

$$0 < V_g < 2 \tag{6.11}$$

De forma similar, o domínio de V_{dc} pode ser determinado assim:

$$V_{dc} + V_g C_{1,i} < C_{2,i} - \frac{A_c}{2}; \quad V_g = 1 \tag{6.12}$$

sendo que $A_c=0.4$ (vide equação 5.6) e estendendo a expressão (6.12) para símbolos de todos os quadrantes, tem-se:

$$|V_{dc}| < \frac{A_c}{2} = 0.2 \tag{6.13}$$

6.4 Arquitetura da Célula-Neurônio

A primeira realização da rede neural de Kohonen[Jar97, Mac96], visou uma implementação genérica do modelo de forma que todos os parâmetros do algoritmo pudessem ser modificados e fosse possível uma programação *on-chip* para uma determinada aplicação. Naquela versão foi possível integrar somente uma célula-neurônio num mesmo *chip*. A estrutura da célula foi tal que é possível: configurar os trechos de discretização da função de adaptação, $\alpha(t)$ (vide equação 3.4), e configurar seu valor inicial; indicar o valor inicial do raio de vizinhança topológica, $N_j(t)$ (vide seção 3.5); e, mudar os pesos internos das células a qualquer instante no processo de treinamento. Dadas as limitações de área em silício, o custo/área da realização de um protótipo em hardware digital e a necessidade da redução da complexidade, as funções $\alpha(t)$ e $N_j(t)$ tiveram de ser implementadas em forma discreta, sendo com isto suficiente projetar apenas blocos aritméticos simples (multiplicação, soma, subtração e deslocamento). A aritmética e a representação numérica foi de ponto fixo 12-bits.

No diagrama da figura 6.6 observa-se a arquitetura dessa primeira versão. O funcionamento é baseado em quatro unidades funcionais: a unidade aritmética, composta por um bloco soma, um bloco de subtração, um bloco de multiplicação e um bloco de valor absoluto; a unidade da função de adaptação; a unidade da função do raio de vizinhança; e a máquina de estados finitos.

Depois que a célula recebe a configuração dos pesos iniciais e os demais parâmetros do algoritmo, quando um vetor é apresentado à célula esta o armazena no registrador OP_1 e o peso correspondente em OP_2, calcula então a distância entre os dois vetores e o valor é disponibilizado para que a unidade WTA o leia. As diferenças entre os vetores são armazenada nos registradores $Diff_j$ para serem utilizados na atualização de pesos.

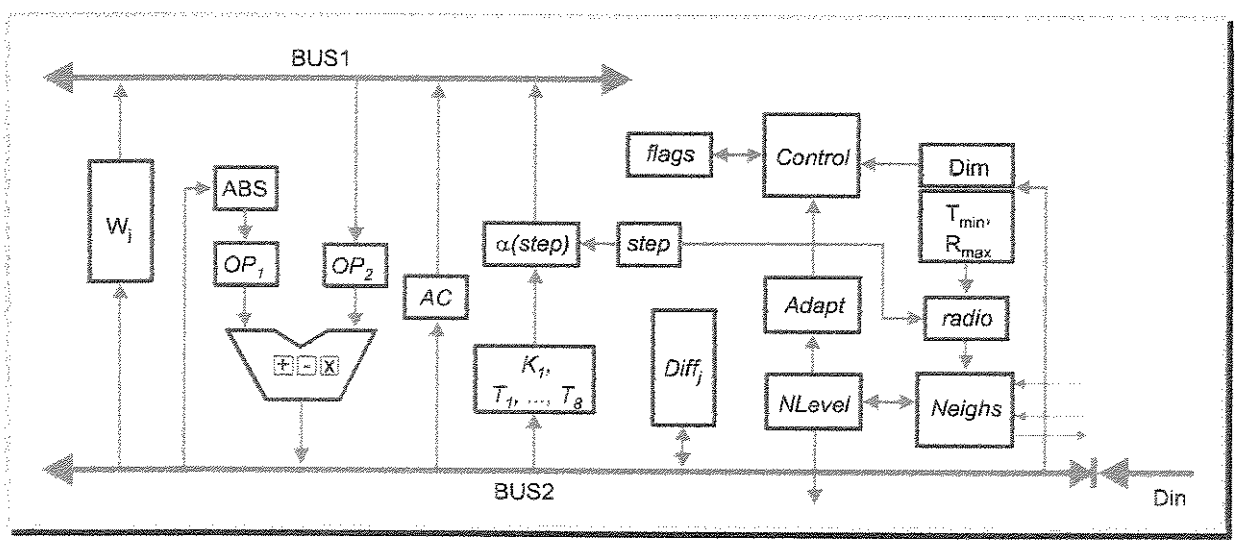


Figura 6.6 Arquitetura básica da primeira implementação em *hardware* do modelo de Kohonen[Jar97,Mac96]

Uma vez determinada a célula vencedora do processo competitivo, a unidade WTA avisa à célula vencedora enviando-lhe um sinal, esta às suas vizinhas topológicas imediatas (do primeiro nível) e estas por sua vez às suas vizinhas imediatas (do segundo nível) e assim em diante até atingir o raio máximo de vizinhança ou de propagação. Seja na condição de

vencedora ou de vizinha de algum dos níveis, a célula recebe a sinalização da condição correspondente pelo bloco *Neighs*. Este no seu turno, baseado no valor de raio de vizinhança corrente, proporcionado por *radio*, calcula o nível de vizinhança correspondente e o transmite às suas vizinhas topológicas. A função *radio* calcula o novo raio de propagação a cada novo vetor apresentado *(step)*, em função do parâmetros R_{max} e T_{min}, que indicam o raio máximo de vizinhança e passo mínimo de discretização da função de raio de vizinhança, respectivamente.

Com cada novo valor de *step* (a cada novo vetor), segundo os parâmetros K_1 (o valor inicial), T_1 a T_8 (os trechos de discretização), e nível de vizinhança *(NLevel)*, a função de adaptação, $\alpha(step)$, atualiza seu valor. Com este novo valor de α e utilizando as diferenças entre os vetores, armazenadas em $Diff_j$, os pesos internos, W_j, são atualizados por meio da unidade aritmética.

A sincronização e controle das funções antes descritas são realizados através da unidade *Control*, uma máquina de estados finitos.

6.4.1 Primeira Proposta

A primeira proposta para a arquitetura de uma célula-neurônio baseou-se na arquitetura da primeira realização [Jar97, Mac96] em *hardware*, descrita no início desta seção. A estrutura se simplifica significativamente, pois α e β (vide equações 4.26 e 4.27) são constantes e nesta estrutura não são necessárias estruturas de cálculo da função de adaptação. O mesmo acontece com a função de raio de vizinhança, que no caso é fixa. A estrutura pode ser observada no diagrama da figura 6.7.

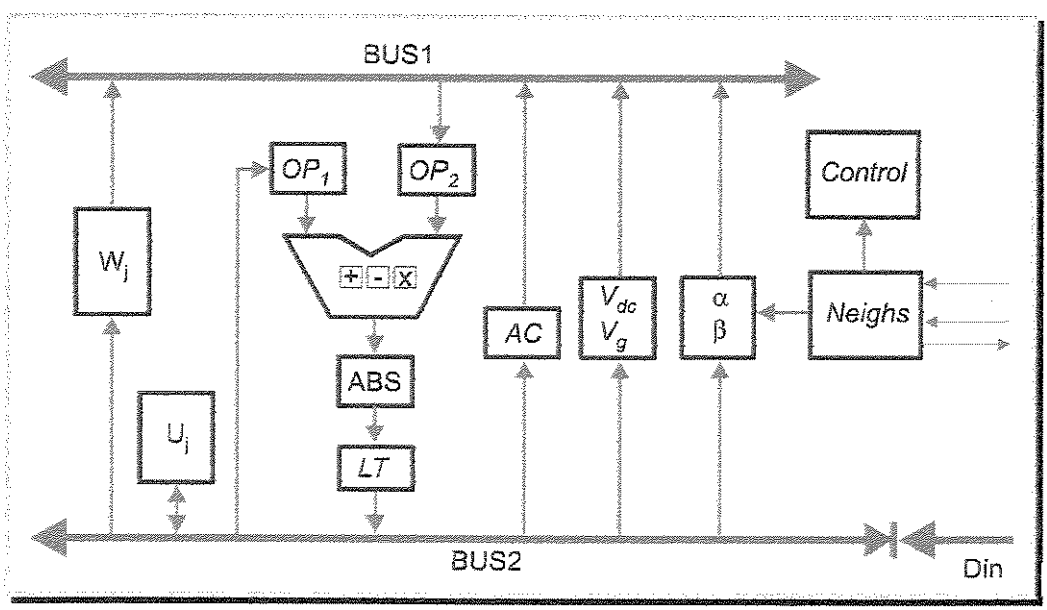


Figura 6.7 Arquitetura para a rede neural segundo a primeira realização em *hardware* do modelo de Kohonen[Jar97, Mac96]

Com relação à versão anterior as principais mudanças são:

- Os registradores $Diff_j$ são dispensáveis pois não é necessário armazenar as diferenças já que no cálculo dos novos pesos o vetor de entrada é multiplicado (amplificado) pelo fator

V_g, porém o vetor tem de ser armazenado para ser utilizado tanto no cálculo da distância *quanto na atualização dos pesos internos*,

- Incluir dois novos registradores para os coeficientes α e β no lugar da implementação da função $\alpha(step)$,
- Acrescentar registradores para os parâmetros V_g e V_{dc},
- Os resultados de algumas operações que são depois armazenadas em OP_1 ou OP_2 precisam de extração do valor absoluto, portanto desloca-se o bloco *ABS* para a saída da unidade aritmética,
- Em algumas operações o resultado da unidade aritmética precisa de ser estocado em OP_1, portanto para evitar um laço de realimentação que geraria indefinições é necessário incluir um registrador temporário *(latch)*, *LT*, na sua saída,
- A função *radio* também é dispensável e a função do bloco *Neighs* limita-se a indicar o uso de α ou β, dependendo se a célula está na condição de vencedora ou vizinha, respectivamente.

$$
\begin{aligned}
&1.\quad OP_1 \leftarrow u_i \\
&\qquad OP_2 \leftarrow w_i \\
&2.\quad LT \leftarrow \left|u_i - w_i\right| \\
&3.\quad AC \leftarrow LT \\
&\qquad OP_1 \leftarrow u_q \\
&\qquad OP_2 \leftarrow w_q \\
&4.\quad LT \leftarrow \left|u_q - w_q\right| \\
&5.\quad OP_1 \leftarrow LT \\
&\qquad OP_2 \leftarrow AC \\
&6.\quad LT = \left|u_i - w_i\right| + \left|u_q - w_q\right|; \quad (distância)
\end{aligned}
\tag{6.14}
$$

No conjunto das expressões (6.14), ← significa "atribuição a", cada linha com número significa um ciclo da máquina de estados e as atribuições sob um mesmo número de linha podem ser executados paralelamente. Esse conjunto dá uma seqüência de cálculos e atribuições para a determinação da distância *Manhattan* entre o vetor de entrada e o vetor de pesos interno, baseado na arquitetura da célula-neurônio da figura 6.7. De forma similar, no conjunto de expressões (6.15) temos a seqüência para a atualização dos pesos internos.

O conjunto de atribuições (6.15) contempla apenas a atualização da componente *i* do peso interno, sendo necessário um conjunto similar para a atualização w_q, substituindo-se u_i por u_q.

Esta primeira proposta de arquitetura não foi considerada para a implementação final por demandar um número grande de ciclos, um total *6 + 2 x 11 = 28*, em conseqüência de um baixo paralelismo, tornando ineficiente o desempenho da estrutura.

$$
\begin{aligned}
&1.\quad OP_1 \leftarrow u_i \\
&\qquad OP_2 \leftarrow V_g \\
&2.\quad LT \leftarrow V_g\ u_i \\
&3.\quad OP_1 \leftarrow LT \\
&\qquad OP_2 \leftarrow w_i \\
&4.\quad LT \leftarrow V_g\ u_i - w_i \\
&5.\quad OP_1 \leftarrow LT \\
&\qquad OP_2 \leftarrow V_{dc} \\
&6.\quad LT \leftarrow V_{dc} + V_g\ u_i - w_i \\
&7.\quad OP_1 \leftarrow LT \\
&\qquad OP_2 \leftarrow \alpha \quad ou \quad \beta \\
&8.\quad LT \leftarrow \begin{matrix}\alpha \\ \beta\end{matrix} \left[V_{dc} + V_g\ u_i - w_i\right] \\
&9.\quad OP_1 \leftarrow LT \\
&\qquad OP_2 \leftarrow w_i \\
&10.\quad LT \leftarrow w_i + \begin{matrix}\alpha \\ \beta\end{matrix} \left[V_{dc} + V_g\ u_i - w_i\right] \\
&11.\quad w_i \leftarrow LT; \quad (peso\ atualizado)
\end{aligned}
\tag{6.15}
$$

6.4.2 Segunda Proposta

A segunda arquitetura faz uso de uma estrutura chamada *SOP (Sum Of Products)* que realiza uma soma de produtos, de acorde com a seguinte expressão:

$$SOP \leftarrow MP_0 + (MP_1)(MC_1) \pm MP_2 \tag{6.16}$$

Essa estrutura ataca o problema do grande número de ciclos da arquitetura anterior, proporcionando um paralelismo maior, porém com o acréscimo de mais uma unidade de adição. No diagrama da figura 6.8 mostra-se a sua estrutura. O acesso à unidade SOP é feito com os barramentos *BUS0*, *BUS1* e *BUS2*.

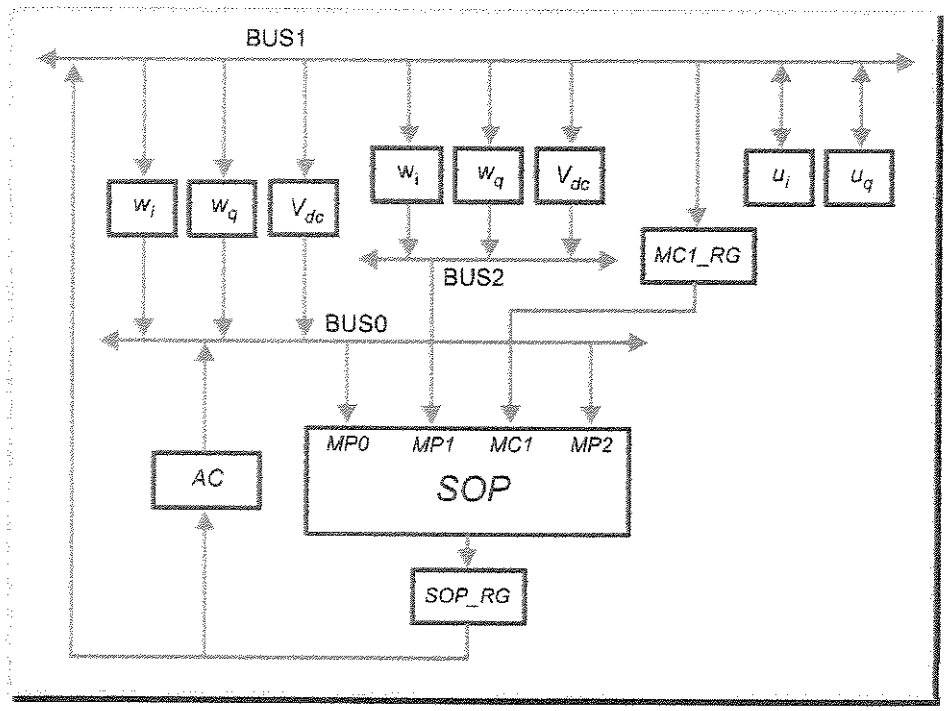


Figura 6.8 Segunda proposta. Arquitetura da célula-neurônio que utiliza a estrutura *SOP (Sum Of Products)*

As expressões (6.17) são a seqüência de atribuições para o cálculo da distância e as em (6.18), a seqüência para a atualização de pesos para o caso da componente *i* do vetor de entrada. A seqüência para a componente *q* é similar, aonde u_i é substituído por u_q e w_q por w_i.

$$
\begin{aligned}
&1.\quad MC_{1_}RG \leftarrow u_i \\
&\qquad MP_0 \leftarrow w_i \\
&\qquad MP_1 \leftarrow 1 \\
&2.\quad MP_2 \leftarrow AC = 0 \\
&3.\quad SOP_RG \leftarrow |u_i - w_i| \\
&4.\quad AC \leftarrow SOP_RG \\
&\qquad MP_0 \leftarrow w_q \\
&\qquad MC_{1_}RG \leftarrow u_q \\
&5.\quad MP_2 \leftarrow AC \\
&6.\quad SOP_RG \leftarrow |u_i - w_i| + |u_q - w_q|; \quad (distância)
\end{aligned}
\tag{6.17}
$$

$$
\begin{array}{ll}
1. & MC_1_RG \leftarrow u_i \\
 & MP_2 \leftarrow w_i \\
 & MP_1 \leftarrow V_g \\
 & MP_0 \leftarrow V_{dc} \\
2. & SOP_RG \leftarrow \left(V_{dc} + V_g\, u_i - w_i\right) \\
3. & MC_1_RG \leftarrow SOP_RG \\
 & MP_0 \leftarrow w_i \\
 & MP_1 \leftarrow \alpha \;\; ou \;\; \beta \\
 & MP_2 \leftarrow 0 \\
4. & SOP_RG \leftarrow w_i + \dfrac{\alpha}{\beta}\left(V_{dc} + V_g\, u_i - w_i\right) \\
5. & w_i \leftarrow SOP_RG; \quad (peso\ atualizado)
\end{array}
\tag{6.18}
$$

Na linha 3 das expressões (6.17) o sinal da diferença u_i-w_i controla a soma ou adição do termo MP_2 da equação (6.16); com isto implementa-se a função de valor absoluto.

Com esta segunda proposta o número de ciclos cai para *6 + 2 x 5 = 16*, porém a vantagem somente acontece na atualização de pesos. No cálculo da distância o número de estados permanece igual, isto se deve ao fato de MP_0 e MP_2 não poderem ser atribuídos paralelamente.

6.4.3 Terceira Proposta

Na figura 6.9 mostra-se a terceira proposta para a arquitetura da célula-neurônio, a qual foi implementada. Esta estrutura usa 5 barramentos, sendo que MP_0 e MP_2 já não recebem dados pelo mesmo barramento, como no caso da segunda proposta. O agrupamento fez-se de forma a minimizar o número de ciclos, tendo como resultado que: *AC* e V_{dc} compartilham o barramento *MP_0_BUS*; α, β e V_g o barramento *MP_1_BUS*; w_i e w_q o barramento *MP_2_BUS*; e, u_i e u_q o barramento *MC_1_BUS*. Esta nova organização traz o beneficio de um maior paralelismo porém o custo é a troca de uma unidade de adição mais uma unidade de adição/subtração, da segunda proposta, por duas unidades de adição/subtração. Portanto a nova expressão para (6.16) é:

$$SOP \leftarrow MP_0 \pm \left(MP_1\right)\left(MC_1\right) \pm MP_2 \tag{6.19}$$

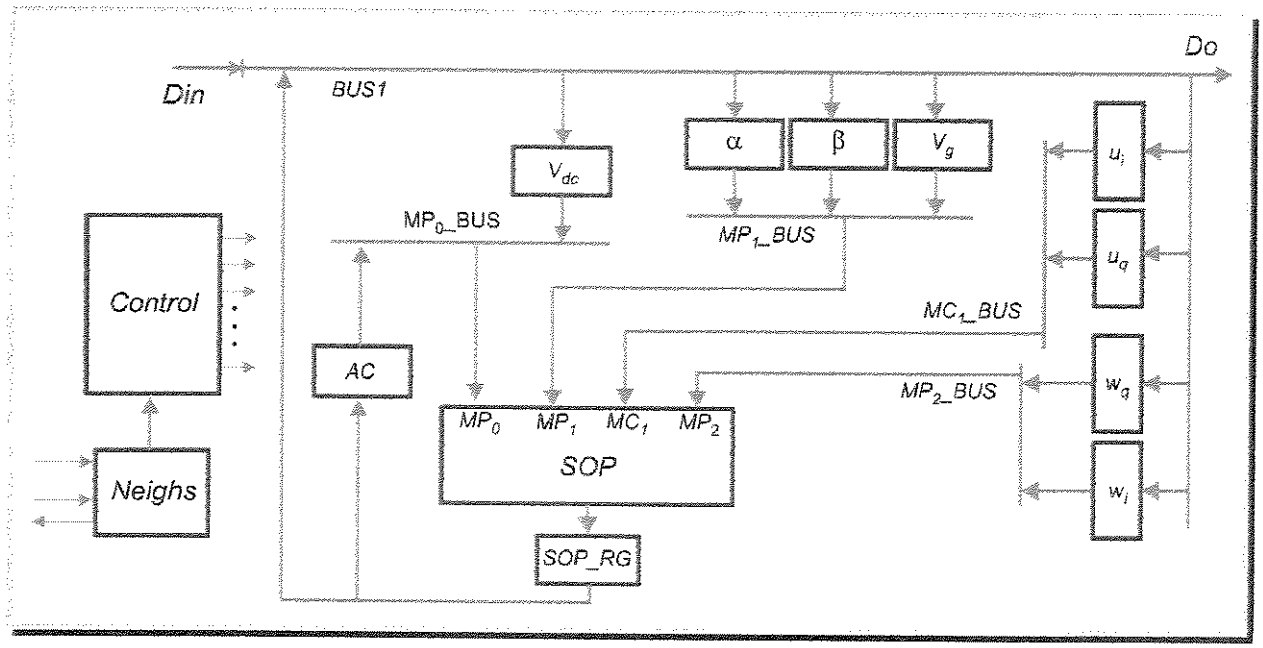


Figura 6.9 Arquitetura básica final da célula-neurônio que utiliza a estrutura *SOP (Sum Of Products)*.

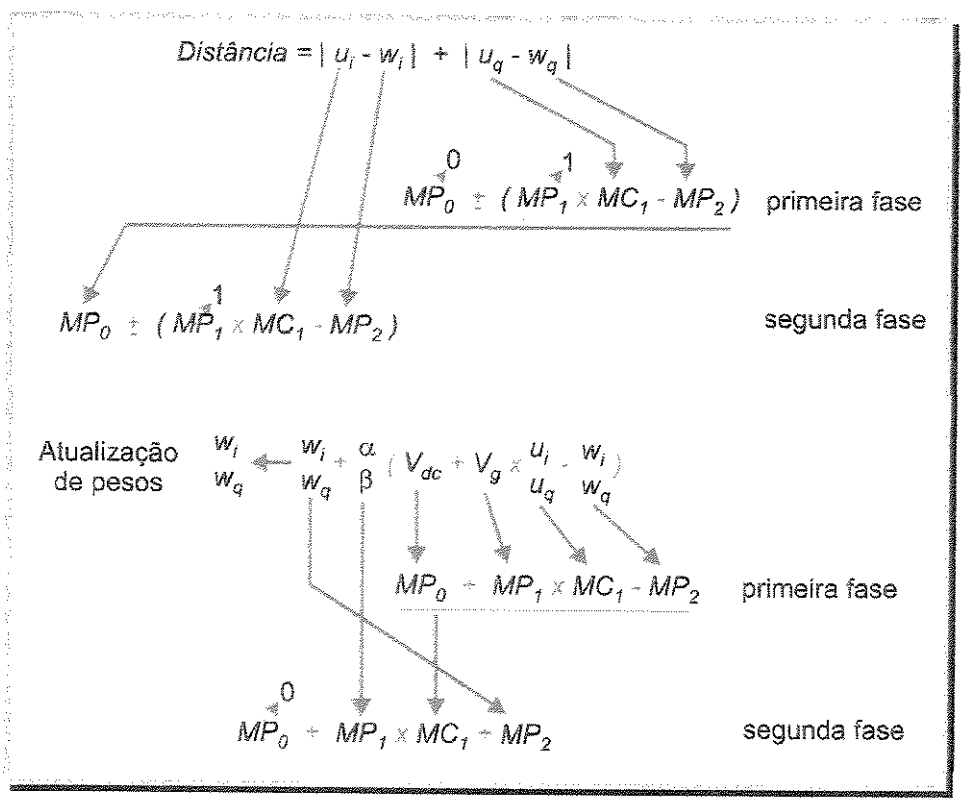


Figura 6.10 Utilização da estrutura SOP para a atualização dos pesos e cálculo da distância *Manhattan*.

$$
\begin{aligned}
&1.\quad MC_1 \leftarrow u_i \\
&\qquad MP_0 \leftarrow AC = 0 \\
&\qquad MP_1 \leftarrow 1 \\
&\qquad MP_2 \leftarrow w_i \\
&2.\quad SOP_RG \leftarrow |u_i - w_i| \\
&3.\quad MP_0 \leftarrow AC \leftarrow SOP_RG \\
&\qquad MC_1 \leftarrow u_q \\
&\qquad MP_2 \leftarrow w_q \\
&4.\quad SOP_RG \leftarrow |u_i - w_i| + |u_q - w_q| ; \quad (distância)
\end{aligned}
\tag{6.20}
$$

$$
\begin{aligned}
&1.\quad MC_1 \leftarrow u_i \\
&\qquad MP_2 \leftarrow w_i \\
&\qquad MP_1 \leftarrow V_g \\
&\qquad MP_0 \leftarrow V_{dc} \\
&2.\quad SOP_RG \leftarrow \left(V_{dc} + V_g\, u_i - w_i\right) \\
&3.\quad MC_1 \leftarrow u_i \leftarrow SOP_RG \\
&\qquad MP_0 \leftarrow 0 \\
&\qquad MP_1 \leftarrow \alpha \;\; ou \;\; \beta \\
&\qquad MP_2 \leftarrow w_i \\
&4.\quad SOP_RG \leftarrow w_i + \frac{\alpha}{\beta}\left(V_{dc} + V_g\, u_i - w_i\right) \\
&5.\quad w_i \leftarrow SOP_RG ; \quad (peso\ atualizado)
\end{aligned}
\tag{6.21}
$$

A implementação do algoritmo de Kohonen nesta estrutura consiste da utilização da estrutura SOP da expressão (6.19) em duas fases (vide figura 6.10). Na primeira avalia-se a diferença de um dos componentes do vetor de entrada, deslocado e amplificado por V_{dc} e V_g respectivamente, com o componente correspondente do vetor de pesos interno. Na segunda fase o valor anteriormente calculado é multiplicado pelo fator de adaptação α ou β e então somado ao valor da componente de peso interno respectivo.

A utilização da unidade SOP para o cálculo da distância é realizada em duas fases (vide figura 6.10). Na primeira é calculado o valor absoluto da diferença entre u_i e w_i, o qual é armazenado em *AC* e na segunda fase o valor absoluto da diferença de u_q e w_q.

Nas expressões (6.20) e (6.21) apresentam-se as seqüências de cálculo da distância *Manhattan* e da atualização de pesos, respectivamente. Nesta terceira proposta o cálculo da distância se realiza em 4 ciclos e a atualização de pesos em 2 x 5 = 10 ciclos,

resultando em um total de 14 ciclos.

6.5 Descrição do Circuito

Uma vez definida a arquitetura a ser implementada, nesta seção descrevem-se o circuito, o funcionamento e a interface da rede neural, da célula-neurônio e da unidade WTA. A especificação inicial foi realizada por meio de uma descrição em VHDL e captura esquemática. A tecnologia de implementação usada foi AMS-CMOS 0.8μm e a implementação do ASIC utilizou a biblioteca de células padrão *(standard cells)* da AMS.

6.5.1 A Rede Neural

Nos diagramas das figuras 6.11 e 6.12 mostram-se a interface e o diagrama de blocos da rede neural implementada. Na interface existem 5 tipos de sinais: os barramentos de entrada, de endereços e de dados; os sinais de configuração (escrita, leitura, reinicio, relógio, aviso, seleção); os sinais de interfaceamento com as células vizinhas; os sinais de intercâmbio com blocos WTA externos; e, sinais de saída (barramentos de dados, aviso).

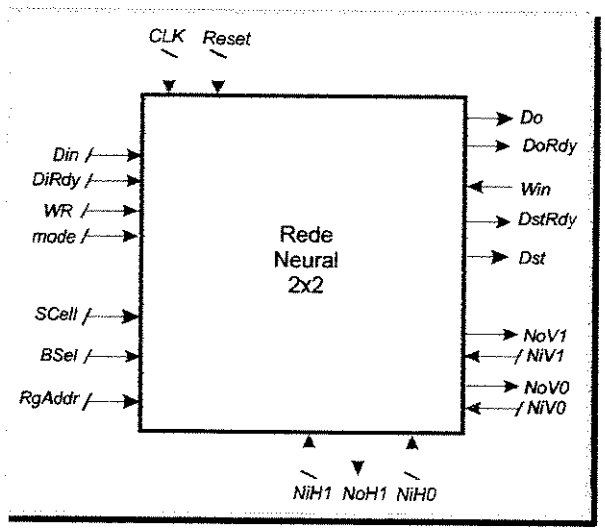


Figura 6.11 Diagrama da interface da rede neural

A seguir descrevem-se em detalhe os sinais da interface:

Barramentos de Entrada:

Din Barramento de 12 bits, transporta os dados de entrada com a representação indicada na figura 6.4. Na fase de configuração contém os dados dos pesos internos iniciais, V_{dc} e V_g. Na fase de treinamento e reconhecimento contém os dados dos vetores de entrada.

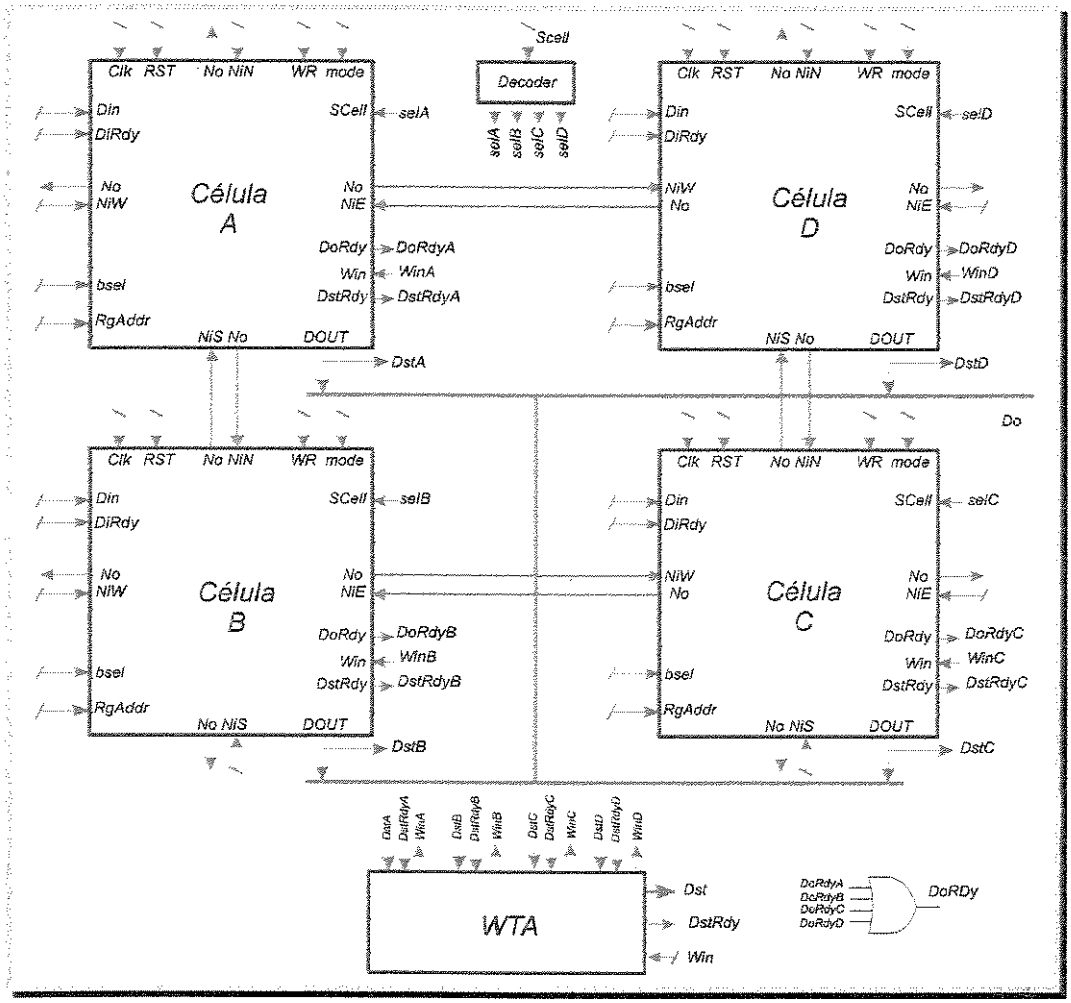


Figura 6.12 Diagrama de blocos da rede neural.

Scell Barramento de 2 bits, seleciona uma das quatro células de acordo com a tabela seguinte :

Scell	Célula selecionada
00	A
01	B
01	C
11	D

RgAddr Barramento de 3 bits, endereça os registradores α, β, V_g, V_{dc}, w_i e w_q, da célula selecionada por *Scell*.

RgAddr	Registrador selecionado
000	W_i
001	W_q
010	α
011	β
100	V_{dc}
101	V_q

BAddr Barramento de 2 bits, possibilita a seleção de um dos barramentos internos da célula selecionada por *SCell* para a sua leitura. Isto tem a finalidade de poder verificar, em qualquer momento do treinamento o reconhecimento da rede neural, o conteúdo de tais barramentos.

BAddr	Barramento selecionado
00	MP_0_BUS
01	MP_1_BUS
10	MC_1_BUS
11	MP_2_BUS

Sinais de Configuração:

DiRdy Pulso positivo que indica a disponibilidade do dado de entrada para todas as células internas. Somente é válido quando o dado é uma das componentes do vetor de entrada.

WR Pulso positivo que serve para acionar o processo de armazenamento, nos registradores endereçados por *RgAddr* e na célula selecionada por *SCell*, do dado que ingressa por *Din*.

Mode Sinal que indica o modo de operação: "0" - modo usuário, nele é possível configurar, ler e escrever nos registradores internos; "1" - modo interno, nele o controle é assumido pela máquina de estados interna.

Reset Pulso positivo que coloca a máquina de estados interna no estado inicial de espera.

BSel Permite colocar no barramento interno *BUS1* e de saída *Do*, da célula selecionada por *SCell* e no modo usuário, o valor de um dos barramentos selecionados por *Baddr (Bsel="0")* ou o barramento de entrada *Din (Bsel="1")*.

RdWg Sinal que serve para ler os valores dos registradores de pesos internos, w_i *(RdWg="0")* ou w_q *(RdWg="1")*, sempre que *BSel="0"* e estando no modo usuário.

CLK Sinal de relógio.

Sinais de Interfaceamento com as Células Vizinhas

Com a presente versão do ASIC é possível formar uma estrutura até QAM-16, portanto o número dos sinais de interfaceamento com a vizinhança é projetada de forma que o módulo de 2x2 células, contido no ASIC, possa ter interação somente com outros dois módulos similares, 2 na horizontal e 2 na vertical. Assim a célula *A* (vide figura 6.12) é destinada para os símbolos das esquinas da constelação QAM-16, portanto somente recebe sinalização de

células internas *(B* e *D)* da sua condição de pertencer à vizinhança.

NiV0, Pulso positivo recebido que indica à célula inferior da linha vertical, célula *C* no diagrama 6.12, da condição de pertença à vizinhança de uma célula vencedora de outro módulo de 2x2.

NiV1, Pulso positivo recebido que indica à célula superior da linha vertical, célula *D*, da condição de pertencer à vizinhança.

NiH0, Pulso positivo recebido que indica à célula da direita da linha horizontal, célula *C*, da condição de pertencer à vizinhança.

NiH1, Pulso positivo recebido que indica à célula da esquerda da linha horizontal, célula *B*, da condição de pertencer à vizinhança.

NoV0 Pulso positivo que é produzido pela célula *C* indicando às células de outros módulos 2x2, da sua condição de vencedora. Este sinal deverá servir de entrada para os sinais *NiH0* e *NiV0* de outros módulos.

NoV1 Pulso positivo produzido pela célula *D* avisando da sua condição de vencedora à célula de outro módulo de 2x2, que o recebe através da sua entrada *NiV1*.

NoH1 Pulso positivo gerado na célula *B* que avisa à célula vizinha de outro módulo, que o recebe na sua entrada *NiH1*, da sua condição de vencedora.

Sinais de Interfaceamento com Blocos WTA Externos

Win Pulso positivo recebido que indica que uma das células do módulo é a vencedora.

DstRdy Pulso positivo enviado pelo WTA local quando a distância menor do módulo esta disponível. Em caso do ASIC não interagir com outros módulos 2x2, este sinal pôde ser realimentado ao próprio módulo através da entrada *Win*.

Dst Barramento de 14 bits, representando dados segundo o diagrama da figura 6.4. Transporta o valor da menor distância do módulo local e o disponibiliza para blocos WTA externos.

Sinais de Saída

Do Barramento de 12 bits, que pode conter os dados do vetor de pesos interno, os dados presentes em algum barramento interno e o próprio barramento de entrada.

DoRdy Pulso positivo gerado quando o dado dos pesos internos já atualizados está disponível no barramento *Do*.

O barramento *BUS1* de cada uma das células internas acessa o barramento de saída comum, *Do*, por meio de blocos *buffer-tristate*, os quais são habilitados com o sinal de seleção *Scell*.

O sinal *DoRdy* é gerado a partir de uma função *OR* de todas as saídas *DoRdy* geradas em cada célula.

Na figura 6.13 mostra-se o esquema que foi utilizado para as simulações e que inclui as 4 células-neurônio, o bloco WTA, o decodificador de endereços, os blocos *buffer-tristate* e os PADS correspondentes para todas as entradas e saídas.

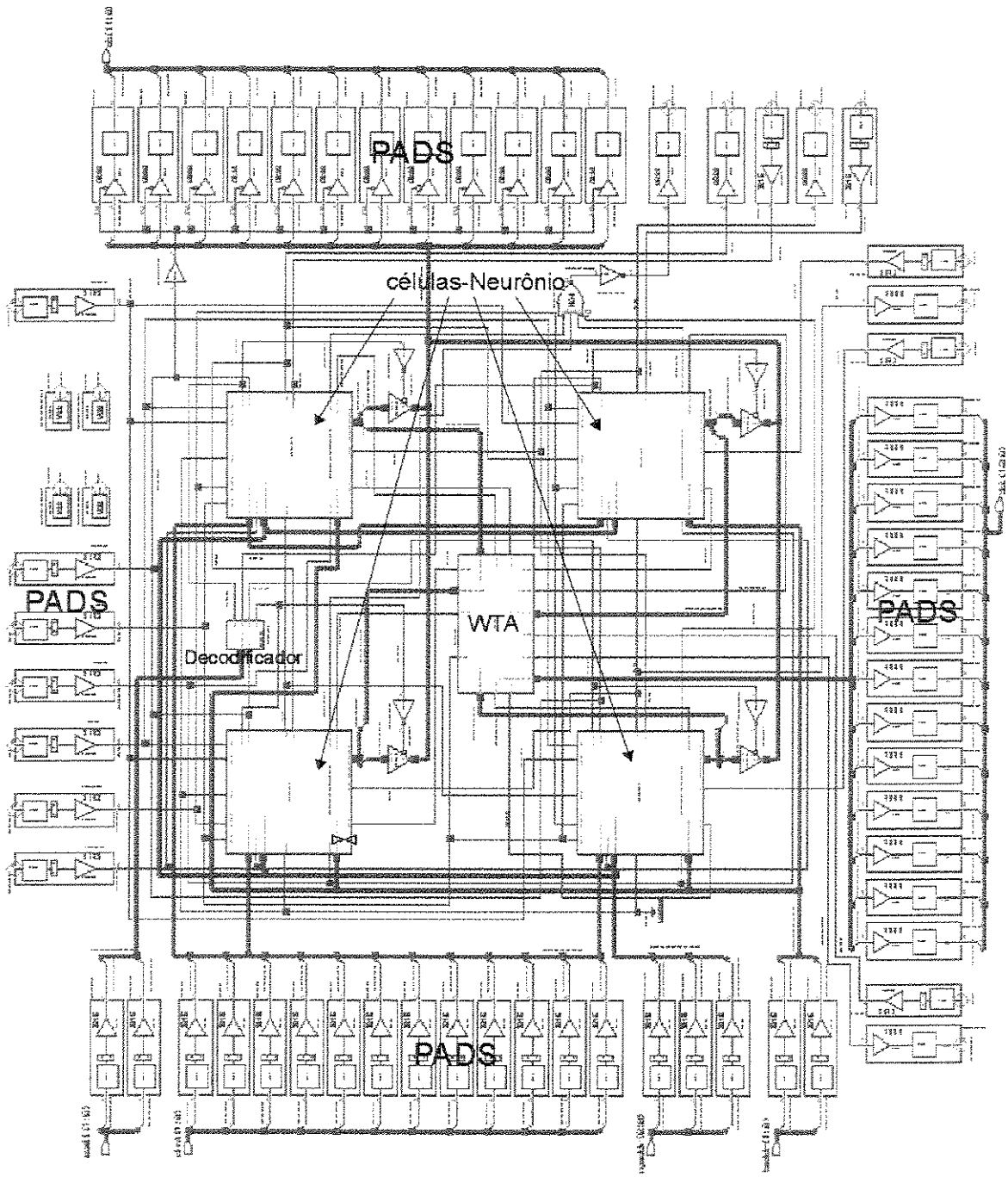


Figura 6.13 Esquema da rede neural de 2x2 células.

6.5.2 A Célula-Neurônio

A arquitetura proposta em 6.4.3 baseada na utilização da unidade Soma de Produtos *(SOP)* tem 5 barramentos principais para dar suporte a esta estrutura. O barramento

BUS1, de 14 bits, transporta os dados na fase de configuração para os registradores internos ou os dados dos pesos atualizados e dos barramentos internos, quando solicitados. O barramento MP_0_BUS é de 14 bits e leva os dados do registrador V_{dc} ou de AC para a unidade *SOP*. MP_1_BUS é de 13 bits, disponibiliza ao *SOP* os dados dos registradores V_g, α e β. Os barramentos de 14 bits MC_1_BUS e MP_2_BUS levam para a unidade *SOP* os dados das componentes *i* e *q* dos vetores de entrada e de pesos interno, respectivamente.

A interface da célula-neurônio (vide figura 6.14) é similar à da rede neural, porém há mudanças nos seguintes sinais:

SCell É um sinal de apenas 1 bit, ativo em "1" lógico, que indica se a célula foi selecionada ou não.

Dout Barramento de saída de 14 bits, reflexo do barramento interno *BUS1*.

NiN, NiS, NiE, NiW Pulsos positivos de entrada que sinalizam a condição da célula vizinha do norte (N), sul (S), leste (E) e oeste (W), respectivamente, ser a vencedora.

No Único sinal de saída para as células vizinhas do norte, sul, leste e oeste, indicando que a célula local é a vencedora.

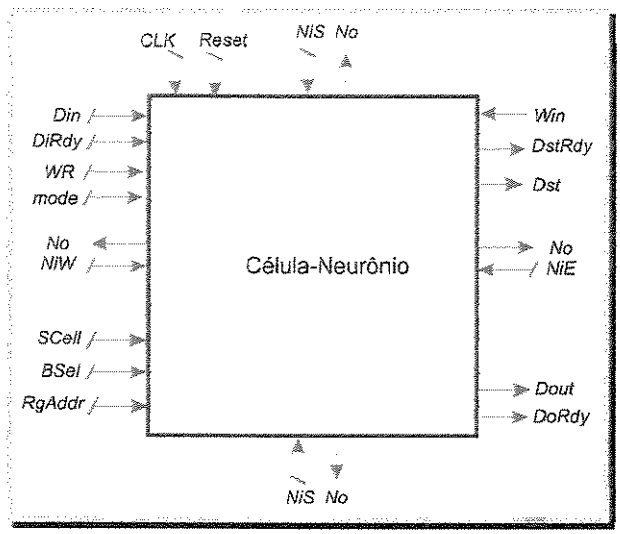


Figura 6.14 Diagrama da interface da célula-neurônio.

A compatibilidade na representação numérica entre o barramento de entrada *Din*, de 12 bits, com o barramento interno *BUS1* de 14 bits faz-se copiando o bit de sinal (vide figura 6.4), d_{11}, para os bits d_{12} e d_{13}. Isto é possível pois o domínio dos números dos vetores de entrada está restrito a $(-1 < (u_i, u_q) < 1)$. Existe o caso especial do registrador V_g, cujo domínio é como se indica na equação (6.11), sendo que o bit d_{11} contém a parte inteira. Devido à atribuição realizada nos bits d_{12} e d_{13}, estes são mascarados ao momento da armazenagem efetiva do dado em V_g.

6.5.2.1 A Unidade Soma de Produtos *(SOP - Sum Of Products)*

A unidade *SOP*, que é o bloco principal da estrutura da célula-neurônio, foi gerada com inspiração nos *Autologic Blocks*™ [Men95k], cujo esquema ilustra-se na figura 6.15. Esta unidade é composta por um multiplicador, duas unidades adição/subtração e um bloco

multiplexador. A interface é descrita a seguir:

MP_0, MP_1, MC_1 e MP_2 barramentos de entrada de 14 bits com exceção do MP_1 que é de 13 bits, correspondentes aos operandos mencionados na expressão (6.19).

Cnt1 Controla se o operando MP_2 deve ser subtraído do produto MC_1 x MP_1 ou não.

Cnt2 Habilita a operação da função de valor absoluto da diferença entre MC_1 e MP_2

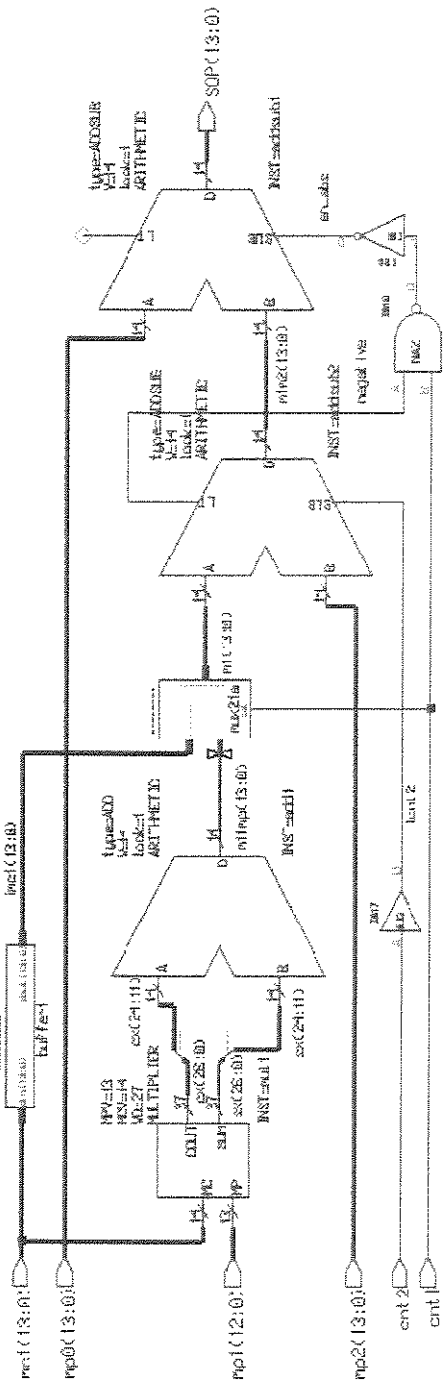


Figura 6.15 Esquema da Unidade Soma de Produtos *(SOP - Sum Of Products)*.

O bloco de multiplicação, gerado a partir dos *Autologic Blocks*^{TM}, é implementado *por meio de um bloco de multiplicação parcial e um bloco de adição. O multiplicador parcial* proporciona na sua saída dois termos de 27 bits, *Cout* e *SUM* (vide figura 6.15), baseado no *algoritmo de Booth e o procedimento na árvore de Wallace* [Men95k, Hwa79, Knu71]. *Os dois termos* depois de serem truncados a 14 bits são somados.

Nas operações mencionadas na expressão (6.20) e no diagrama da figura 6.10 em algumas fases o operando MC_1 tem de assumir o valor 1. Para tanto, na estrutura SOP incorpora-se um multiplexador que seleciona entre MC_1e a saída do multiplicador. Nos casos em que MP_1 tenha de ser multiplicado por 1, é transferido diretamente pelo multiplexador, evitando desta maneira a multiplicação.

Para os casos em que o operando *MP0* tenha de ser 0, o registrador *AC* é reiniciado antes de sua transferência pelo barramento MP_0_*BUS* para o operando MP_0.

Nos casos de cálculo da distância, na unidade SOP calcula-se o sinal da diferença MC_1-MP_2 (sendo neste caso MP_1 ignorado), com o qual controla-se o funcionamento do bloco seguinte de adição/subtração. Caso este sinal seja negativo então a operação a ser realizada é MP_0 - (MC_1 - MP_2), tendo em conseqüência o valor absoluto.

6.5.2.2 Registradores Internos

Conjuntos de registradores (i.e. α, β e V_g) estão conectados a barramentos e por meio destes à unidade SOP, como ilustrado na arquitetura da figura 6.9. Analisam-se duas formas de realizar esta conexão. Na primeira o acesso é por meio de registradores com saídas de alta impedância e na segunda os registradores são conectados por meio de multiplexadores, como pode ser observado nos diagramas da figura 6.16.

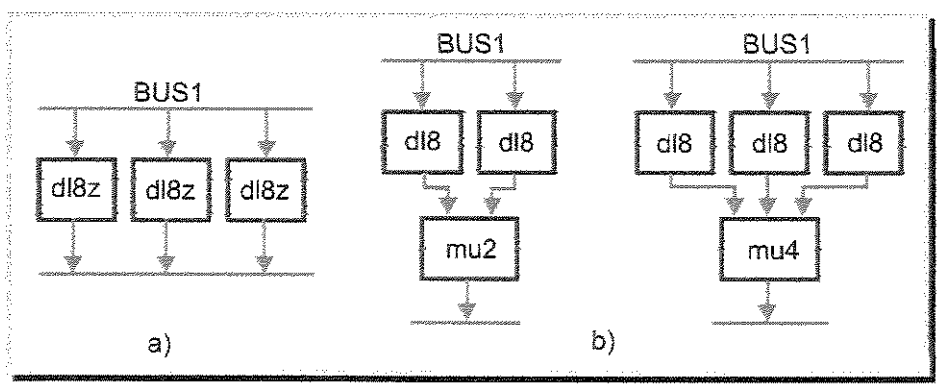


Figura 6.16 Comparação das estruturas para acesso a barramentos. a) registrador com saída em alta impedância, b) acesso via multiplexador. dl8z, dl8, mux2 e mux4 são células padrão da tecnologia AMS-CMOS 0.8µm.

As áreas das células padrão[Ams95], em unidades de *mils*^{2}, utilizadas para a análise são: *dl8z=2.14, dl8=1.78, mu2=1.5, mu4=2.78* e *mu8=6.84*. Para o caso de 14 bits segundo a opção a) da figura 6.16 tem-se 14 x dl8z=29.96 e segundo a opção b) tem-se 14 x dl8 + 2 x mu8 + mu2 = 41.38. A segunda alternativa oferece menores tempos de propagação entre os barramentos, porém a primeira opção é menor em área e além disso tem maior simplicidade na interconexão, fator bastante importante na determinação da densidade nos *layouts* dos ASICs.

A célula *dl8z* não tem entradas assíncronas de *reset* ou *preset*, sendo estas necessárias para a implementação do registrador AC. A forma de realização deste registrador é usando as células *dl8* (registrador com entrada de *reset*) em conjunto com *it2* (célula *buffer* com saída de alta impedância)

6.5.2.3 O Bloco Interface de Vizinhança

Este bloco administra os sinais vindos das células vizinhas ou da unidade WTA local determinando a utilização do fator β ou α, respectivamente. O sinal *WnFlg* na saída do *flip-flop* (vide figura 6.17) assume o valor lógico "1", quando da chegada de um pulso positivo das células vizinhas, somente se não há presença de pulso no sinal *Win*, em caso contrário assume o valor lógico "0". A máquina de estados recebe a informação da célula estar na condição de vizinha ou de vencedora no sinal *WinNg*. A ativação dos sinais *EnoA* e *EnoB*, que habilitam as saídas dos registradores α e β respectivamente, está condicionada ao sinal *Eno_AB* vindo da máquina de estados, sinal este que por sua vez depende do sinal *WinNg*. A ativação é segundo a seguinte tabela:

Eno_AB	Registrador ativado
0	α
1	β

O sinal *No* é gerado como reflexo do sinal *Win* e enviada para as células vizinhas.

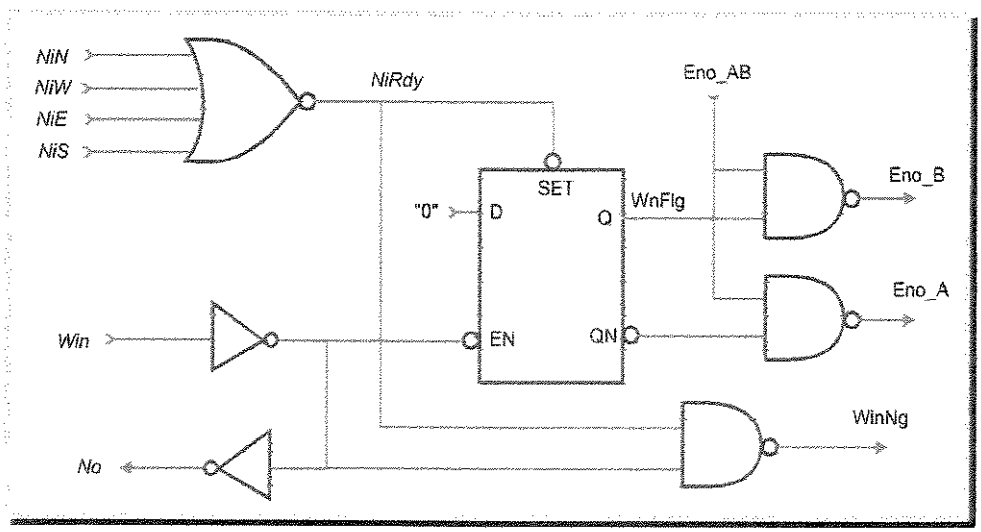


Figura 6.17 Diagrama do bloco interface de vizinhança.

6.5.2.4 Os Blocos Multiplexadores

A finalidade destes blocos é de tornar disponível no barrramento *BUS1* os dados de entrada *(Din)* ou o resultado da unidade *SOP*, dependendo do modo, *Mode*, ser de usuário ou interno, respectivamente. Também, com o propósito de testabilidade, habilitar a saída dos dados dos barramentos internos. O sinal *IBSel* é gerado internamente pela máquina de

estados. O diagrama destes blocos está na figura 6.18 e a sua operação pode ser resumida assim:

Mode	*BSel*	*iBsel*	*S0*	*S1*	Barramento selecionado
0	0	X	X	1	*BUSES*
0	1	X	0	0	*Din*
1	X	0	0	0	*Din*
1	X	1	1	0	*da unidade SOP*

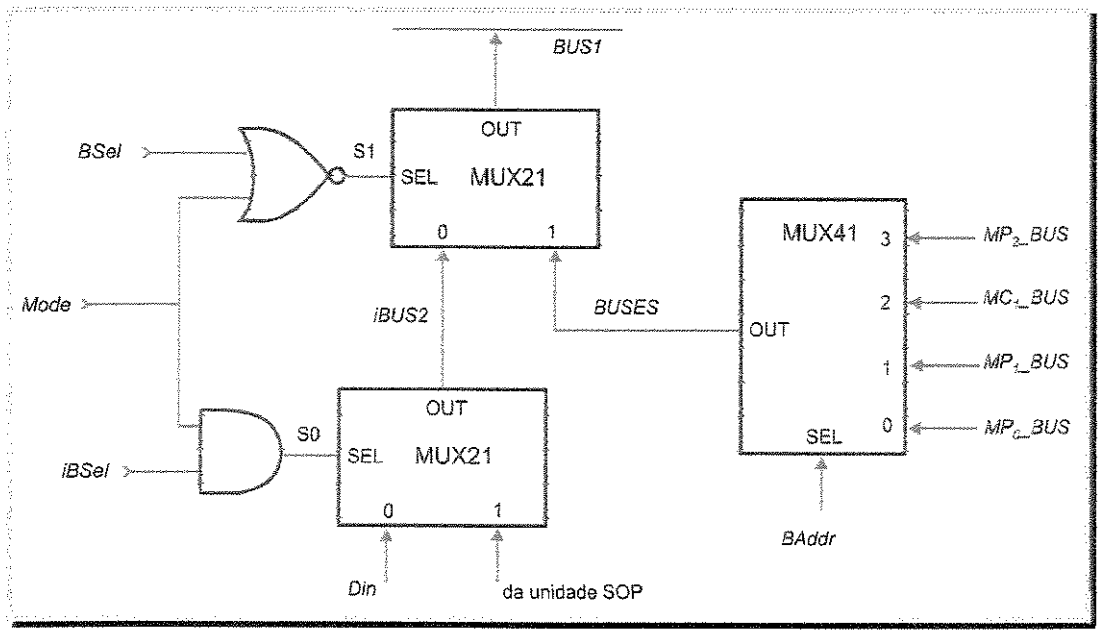


Figura 6.18 Diagrama do bloco de multiplexadores

6.5.2.5 O Bloco Decodificador de Endereços

O bloco decodificador seleciona um dos seguintes registradores: α, β, V_{dc} ou V_g e gera um pulso na saída quando o modo é de usuário em função do sinal *Scell* e *WR*. O seu diagrama mostra-se na figura 6.19 e seu funcionamento resume-se assim:

Mode	*WR*	*RgAddr*	*iST_Wi*	*IST_Wq*	*ST_Wi*	*ST_Wq*	*ST_A*	*ST_B*	*ST_Vdc*	*ST_Vg*
0	⊓	000	1	1	⊔	1	1	1	1	1
0	⊓	001	1	1	1	⊔	1	1	1	1
0	⊓	010	1	1	1	1	⊔	1	1	1
0	⊓	011	1	1	1	1	1	⊔	1	1
0	⊓	100	1	1	1	1	1	1	⊔	1
0	⊓	101	1	1	1	1	1	1	1	⊔
1	X	XXX	⊔	1	⊔	1	1	1	1	1
1	X	XXX	1	⊔	1	⊔	1	1	1	1

⊓ = pulso positivo; ⊔ = pulso negativo

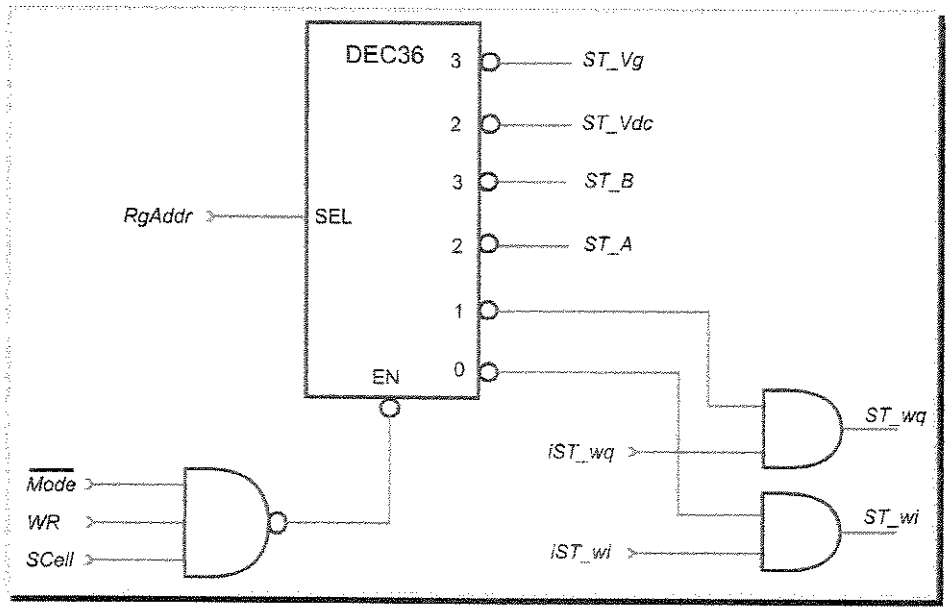


Figura 6.19 Diagrama do bloco decodificador

6.5.2.6 A Máquina de Estados

As expressões (6.20) e (6.21) correspondentes à arquitetura da terceira proposta, são implementadas pela utilização de uma máquina de estados de *Moore*[Koh78]. Esta máquina consiste de dois laços, um de 4 estados para o cálculo da distância e outro de 8 para a atualização dos pesos internos, sendo necessário um ciclo da máquina para cada estado.

Foi mencionado na seção 6.4.3 que o número de ciclos necessários para o cálculo da distância e atualização de pesos é de 14, porém na estrutura proposta é possível "adiantar" algumas atribuições enquanto se realizam as operações na unidade SOP. Com isto consegue-se economizar dois ciclos dos mencionados em 6.4.3, sendo necessários então 12 ciclos para o funcionamento completo da máquina de estados. O diagrama de estados desta máquina encontra-se na figura 6.20.

A função de cada um dos sinais na saída, agrupadas no barramento C de 16 bits, da máquina de estados é:

iST_w_i Sinal de armazenamento para o registrador w_i
iST_w_q Sinal de armazenamento para o registrador w_q
iEno_jq Sinal que habilita os dados dos registradores u_i e u_q no barramento MC_1_BUS.
ST_u_i Armazena o dado em *BUS1* no registrador u_i
ST_u_q Armazena o dado em *BUS1* no registrador u_q
ST_SOP Sinal de armazenamento para o registrador *SOP_RG*
ST_AC Sinal de armazenamento para o registrador *AC*
Rst_AC Sinal de reinicio em 0 do registrador *AC*
Eno_AC Sinal que habilita o dado em *AC* para o barramento MP_0_BUS
Eno_Bias Habilita o armazenamento do dado em *BUS1* no registrador V_{dc}.
Eno_AB Sinal habilitado quando se recebe o sinal *WinNg* do bloco de interfaceamento com a vizinhança. Serve para decidir a aplicação de α ou β na adaptação dos pesos internos.

C1_SOP Permite à unidade SOP realizar a operação MP_0 - $(MP_1 \times MC_1 \pm MP_2)$ que é utilizada para a implementar a função de valor absoluto. Corresponde ao sinal *Cnt1* da unidade SOP.

C2_SOP Permite à unidade SOP fazer a operação $MP_0 \pm (MP_1 \times MC_1 - MP_2)$. Corresponde ao sinal *Cnt2* da unidade SOP.

iBSel Permite determinar se *Din* ou a saída da unidade SOP pode ser transferida para o barramento *BUS1*.

DoRdy Pulso positivo gerado quando as duas componentes do vetor de pesos está já atualizado.

DstRdy Pulso positivo que avisa ao bloco WTA local da disponibilidade da distância *Manhattan* válida.

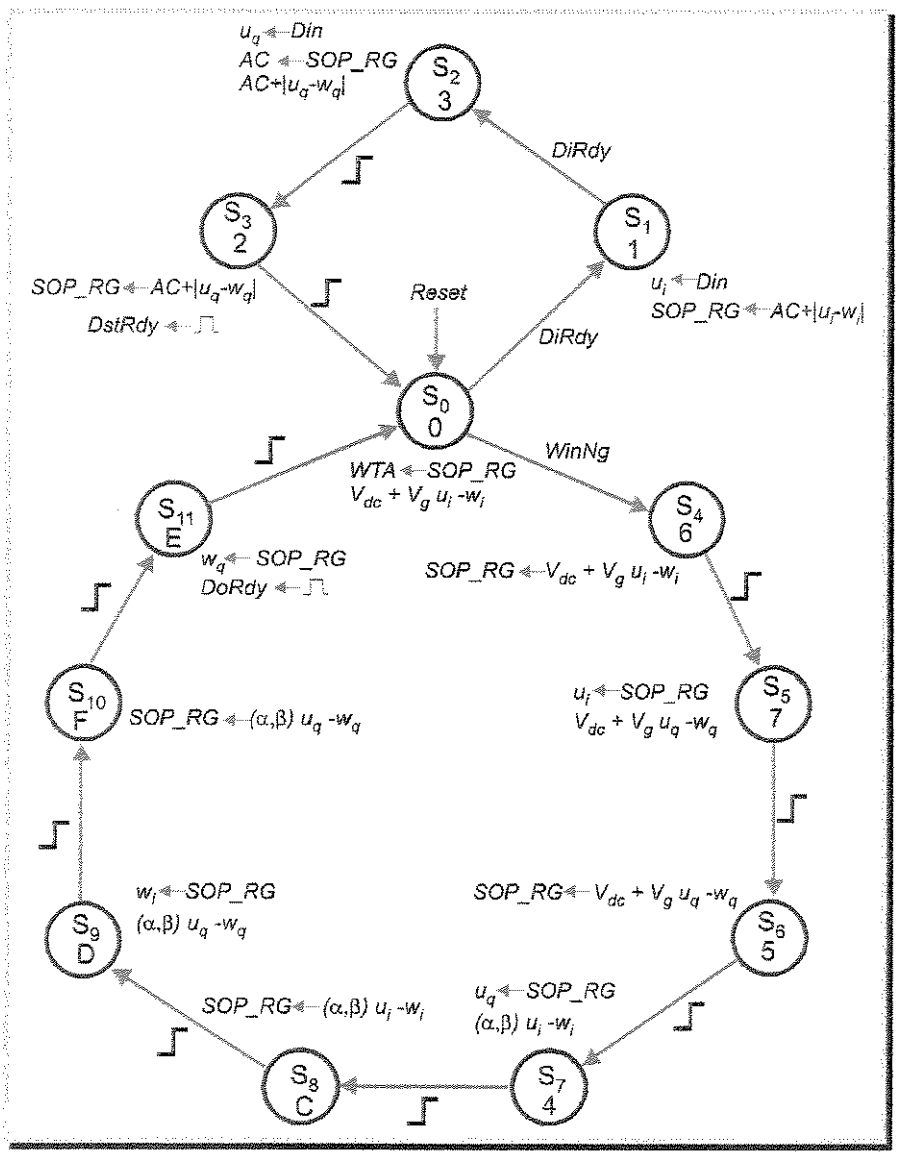


Figura 6.20 Diagrama da máquina de estados.

Na tabela seguinte encontra-se resumida a operação da máquina de estados, nela *cada linha corresponde a um estado e cada coluna a uma saída*. As *saídas com o símbolo* (•) na frente são ativas no nível lógico "0" e as restantes no nível lógico "1". A última coluna mostra *a saída no formato hexadecimal*.

Estado	Estado (HEX)	Saída - Barramento *C(15:0)*																Saída (HEX)
		DstRdy	DoRdy	IBSel	C2_SOP	C1_SOP	Eno_AB	•Eno_Bias	•Eno_AC	•Rst_AC	•ST_AC	•ST_SOP	•ST_uq	•ST_ui	•iEno_iq	•iST_wq	•iST_wi	
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
S0	0	0	0	1	1	0	0	0	1	1	1	1	1	1	0	1	1	31FB
S1	1	0	0	0	1	1	0	1	0	0	1	0	1	0	0	1	1	1A53
S2	3	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1	1	1AAF
S3	2	1	0	1	1	1	0	1	0	1	1	0	1	1	1	1	1	BADF
S4	6	0	0	1	1	0	0	0	1	1	1	0	1	1	0	1	1	31DB
S5	7	0	0	1	1	0	0	0	1	1	1	1	1	0	1	1	1	31F7
S6	5	0	0	1	1	0	0	0	1	1	1	0	1	1	1	1	1	31DF
S7	4	0	0	1	0	0	1	1	0	0	1	1	0	1	0	1	1	266B
S8	C	0	0	1	0	0	1	1	0	1	1	0	1	1	0	1	1	26DB
S9	D	0	0	1	0	0	1	1	0	1	1	1	1	1	1	1	0	26FE
S10	F	0	0	1	0	0	1	1	0	1	1	0	1	1	1	1	1	26DF
S11	E	0	1	1	0	0	1	1	0	1	1	1	1	1	1	0	1	66FD

A seguir descrevem-se em detalhe cada um dos estados da máquina, cuja explicação pode ser acompanhada com o diagrama de estados da figura 6.20:

S0 Estado de espera pelos sinais *DiRdy* ou *WinNg* e ponto de início para os laços de cálculo de distância e atualização de pesos. O sinal assíncrono *Reset* leva a máquina ao estado S0. Após a distância ter sido calculada duas operações são realizadas: disponibilizar o dado da distância, no barramento *BUS1*, para o WTA local poder lê-lo; habilitar os dados dos registradores correspondentes para adiantar a operação $V_{dc} + V_g \times u_i + w_i$. Os sinais de controle *C1_SOP* e *C2_SOP* assumem os valores "0" e "1" para poder realizar a operação indicada.

S1 Neste estado o dado do barramento *Din* é carregado em u_i e ao mesmo tempo: reiniciar o registrador *AC* com o valor 0, habilitar os dados w_i e u_i e realizar a operação *SOP_RG*←*AC*+$|u_i - w_i|$. Para a passagem ao estado seguinte é necessário que *DiRdy* apresente um pulso positivo. O sinal *C1_SOP* é ativado para a função de valor absoluto na unidade SOP.

S2 A componente *q* do vetor de entrada chega e é armazenado em u_q. O resultado anterior no registrador de saída do bloco SOP, *SOP_RG*, é transferido para o registrador *AC*. Os valores de *AC*, u_q e w_q são habilitados para a unidade SOP poder antecipar a operação AC+$|u_q - w_q|$.

S3 O valor da distância *Manhattan* final é então armazenada em *SOP_RG*, avisando ao bloco WTA local pela geração de um pulso positivo no sinal *DstRdy*.

S4 Chega-se a este estado quando o sinal *WinNg* é ativado. O resultado da operação, V_{dc} +

V_g x u_i - w_i, que foi antecipada em S0 carrega-se no registrador *SOP_RG*. *C2_SOP* continua ativado.

S5 Transfere-se o valor de *SOP_RG* para o registrador u_i e prepara-se a operação V_{dc} + V_g x u_q - w_q. O sinal *iST_ui* é ativado.

S6 *SOP_RG* recebe o valor da expressão antecipada em S5.

S7 Por fim, u_q recebe o valor da expressão V_{dc} + V_g x u_q - w_q. O sinal Eno_AB é ativado de forma que o bloco de interfaceamento com a vizinhança determina qual fator deve ser utilizado, α ou β. Antecipa-se a operação (α, β) x u_i + w_i. Também o registrador *AC* e iniciado com 0 e habilitado ao barramento *MP0_BUS*. *C1_SOP* e *C2_SOP* são desativados para se ter somente operações de soma. O sinal *iST_uq* é ativado.

S8 O resultado de (α, β) x u_i + w_i é armazenado em *SOP_RG*.

S9 A componente *i* do peso é finalmente atualizada. w_i recebe o valor da expressão (α, β) x u_i + w_i armazenado em *SOP_RG*, para tanto *iST_wi* é ativado. Paralelamente prepara-se a operação (α, β) x u_q + w_q.

S10 O resultado da operação adiantada no estado S9 é transferido ao registrador *SOP_RG*.

S11 O sinal *iST_wq* é ativado, portanto o registrador w_q recebe o valor da expressão (α, β) x u_q + w_q. e tem o ser valor atualizado. Os novos valores dos pesos internos é disponibilizado dando o sinal por meio de *DoRdy*.

No anexo 5 pode ser vista a descrição VHDL do circuito completo da máquina de estados. Essa descrição mostra quatro processos: **state_trans** que permite a determinação do seguinte na transição entre os estados da máquina, condicionada ao sinal de relógio e os sinais *DirRdy* e *WinNg*; **clkd**, permite a transição efetiva entre os estados por meio da borda de subida do sinal *CLK*; **out_decode**, decodifica os estados e produz os sinais de saída; e, **output** permite fazer *latch* dos sinais de saída, para evitar *glitches*. Na figura 6.21 tem-se o esquema gerado automaticamente a partir dessa descrição pelo processo de síntese lógica, otimização de área e de temporização e do mapeamento na tecnologia alvo.

Finalmente, na figura 6.22 temos o esquema completo para uma célula neurônio correspondente à terceira proposta de arquitetura.

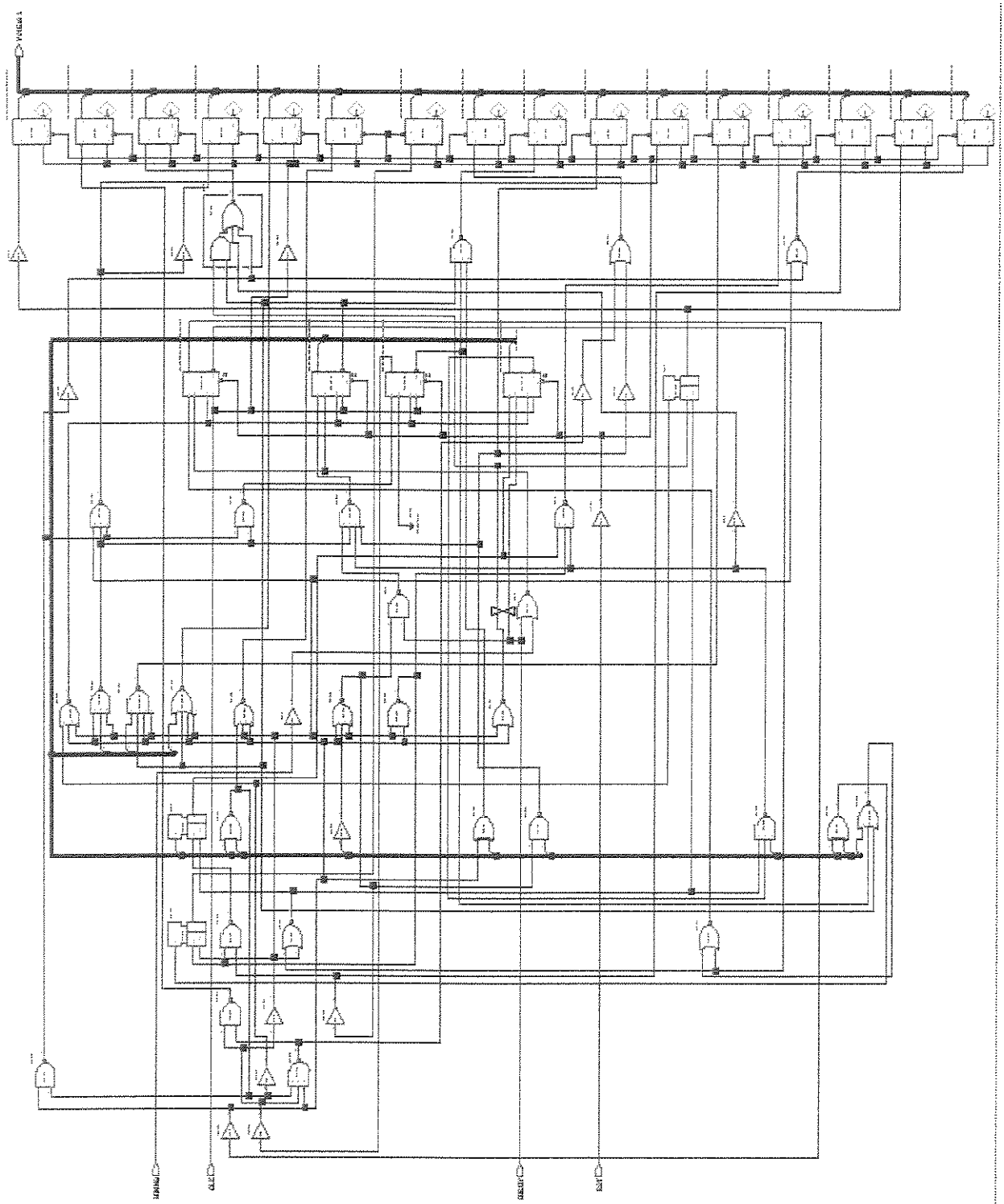


Figura 6.21 Esquema da máquina de estados gerada pelo processo de síntese automática.

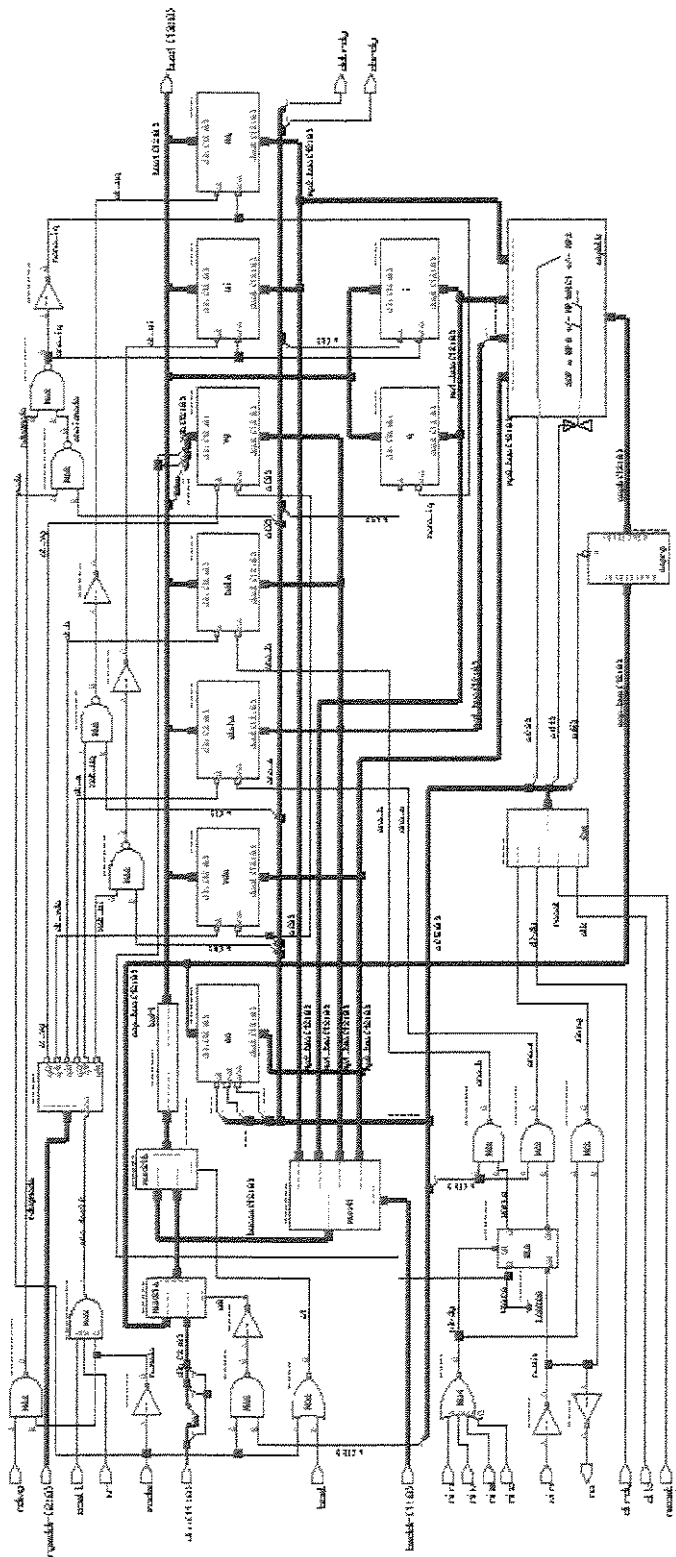


Figura 6.22 Esquema final da célula-neurônio segundo a terceira proposta de arquitetura.

6.5.3 A Unidade WTA *(Winner-Takes-All)*

A função da unidade WTA é determinar a menor distância dentre os quatro valores de distância proporcionados pelas células-neurônio A, B, C e D. A implementação para determinar a distância menor é uma arvore binária de decisão, onde para cada par de entradas determina-se a distância menor e com estas o menor valor final da distância. A célula com o menor valor de distância é sinalizada. A descrição desta unidade foi realizada em VHDL (vide anexo 5). Na figura 6.23 tem-se o diagrama simplificado da unidade WTA e na figura 6.24 uma região do esquema final gerado automaticamente pelo processo de síntese lógica, otimização e mapeamento na tecnologia objeto.

Os quatro sinais *DstRdy*, gerados nas células-neurônio, chegam ao bloco WTA quase ao mesmo tempo dado que ditas células são idênticas, havendo diferença unicamente no roteamento da interconexão entre cada uma delas e o WTA. O processo de determinação do menor valor de distância começa quando todas células tenham habilitado seu dado de distância e sinalizado por meio do *DstRdy* respectivo. Portanto com esses são alimentados a uma porta OR para o sinal *RdyX*, o qual é transmitido para blocos WTA externos no seguinte ciclo de relógio.

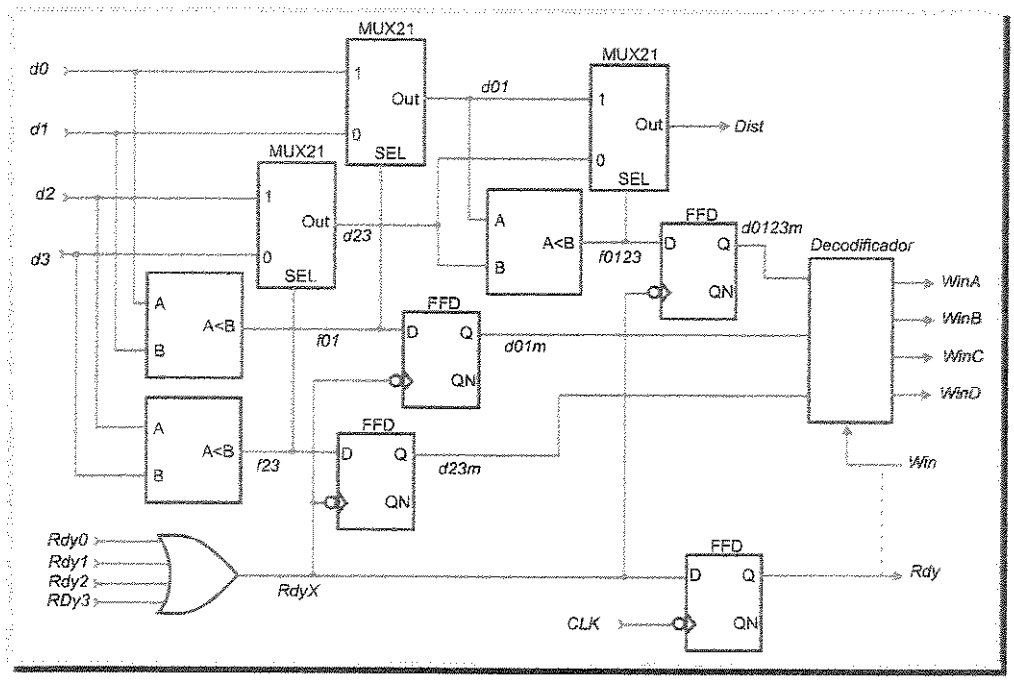


Figura 6.23 Diagrama simplificado da unidade WTA (Winner-Takes-All)

Na determinação da distância menor utilizam-se multiplexadores para cada par de entradas, assim se a entrada *d0* < *d1* então o multiplexador escolhe a entrada *d0*, caso contrário *d1*, determinando a saída *d01*. Simultaneamente, se *d2* < *d3*, um outro multiplexador escolhe *d2* caso contrário *d3*, determinando *d23*. O novo par *d01* e *d23* então é comparado, se *d01* < *d23* é escolhido *d01*, caso contrário *d23* e o valor menor é atribuído à saída *Dist*. Todas essas condições são armazenadas em registradores (*d01m*, *d23m* e *d0123m*) segundo a borda de descida do sinal *RdyX*.

O WTA então aguarda por um pulso na sua entrada *Win*, a qual pode ser realimentada pela saída *RdyX* caso não existam blocos externos de WTA. Quando o pulso chega em *Win*, gera-se numa das saídas, *Win0*, *Win1*, *Win2* ou *Win3*, dependendo das condições armazenadas em *d01m*, *d23m* e *d0123m*, um pulso positivo indicando à célula correspondente da sua condição de vencedora.

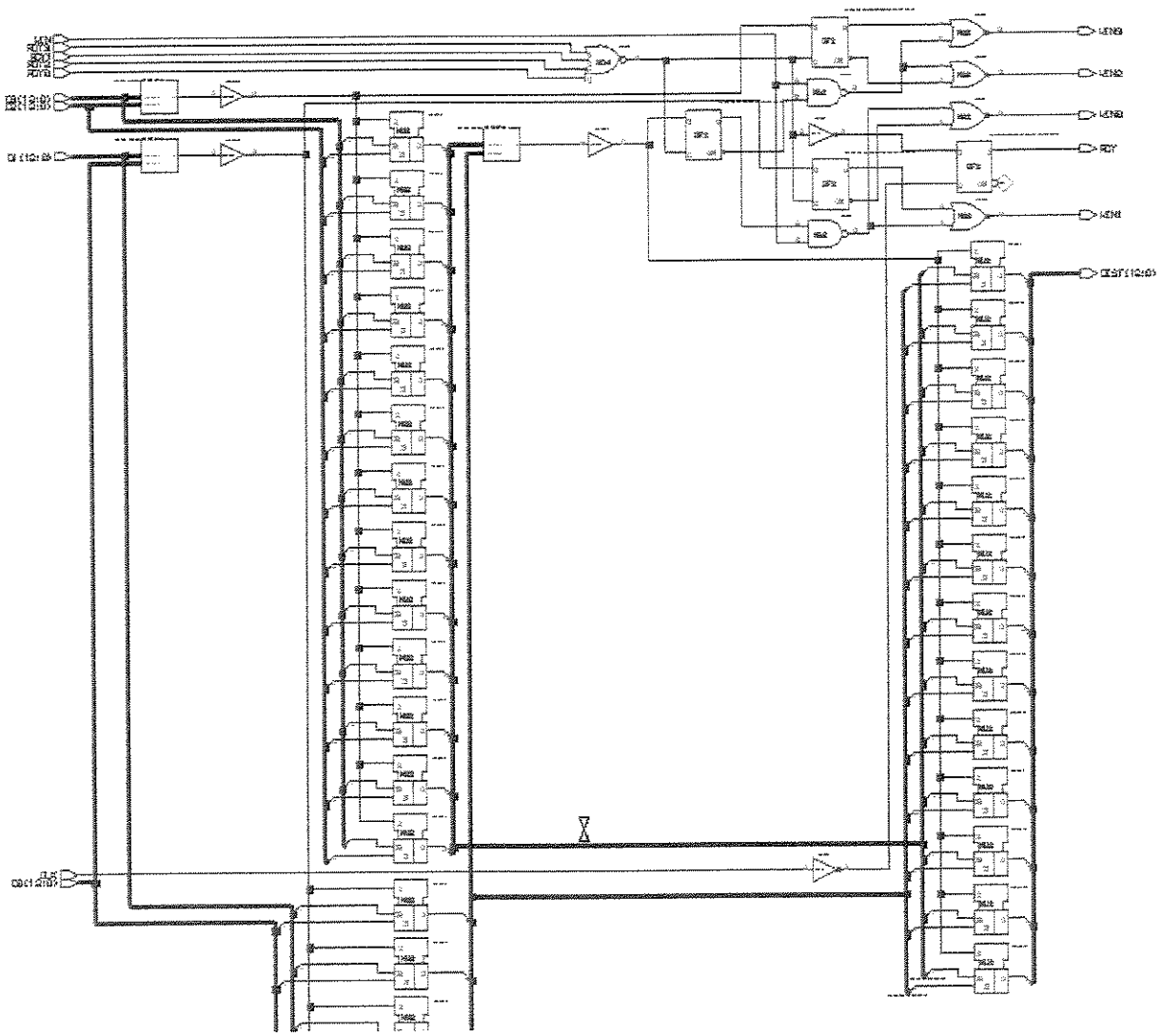


Figura 6.24 Região do esquema da unidade WTA *(Winner-Takes-All)* gerado automaticamente pelo processo de síntese, otimização e mapeamento na tecnologia objeto.

6.6 Simulações Pré-Layout

O circuito utilizado para as simulações é o da figura 6.13, o qual inclui as células de PADS onde efetivamente serão conectados os pinos do *chip*.

Como vetor de entrada para a rede neural foi utilizado a seqüência *r(k)* gerada com os programas da implementação em *software* do capítulo 5. A estrutura de equalização

utilizada nas simulações é o da figura 4.13.

Tanto nas simulações pré-layout quanto nas de pós-layout foram desenvolvidos para seu suporte *scripts* na linguagem *AMPLE*^{TM} [Men95l].

Na seqüência de figuras a seguir mostram-se os diagramas de tempo para os diferentes blocos de que está composta a rede neural.

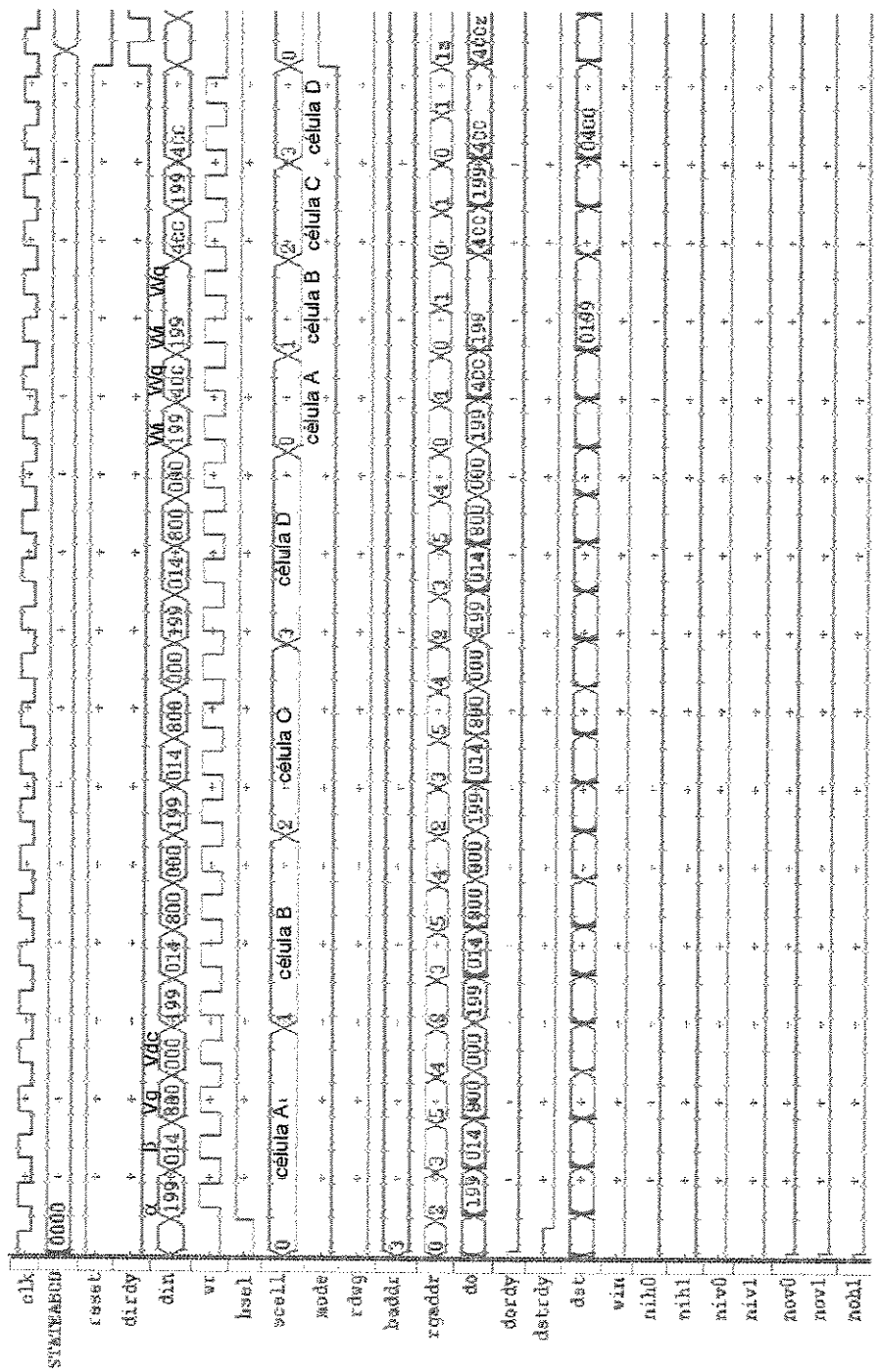


Figura 6.25 Simulação da fase configuração dos registradores internos das células-Neurônio.

Na figura 6.25 a escrita dos parâmetros α, β, V_g e V_{dc} é feita nas células A, B, C e D, nesse ordem. Após isto os valores dos pesos iniciais é armazenado em w_i e w_q de todas as células, na ordem anterior mencionada.

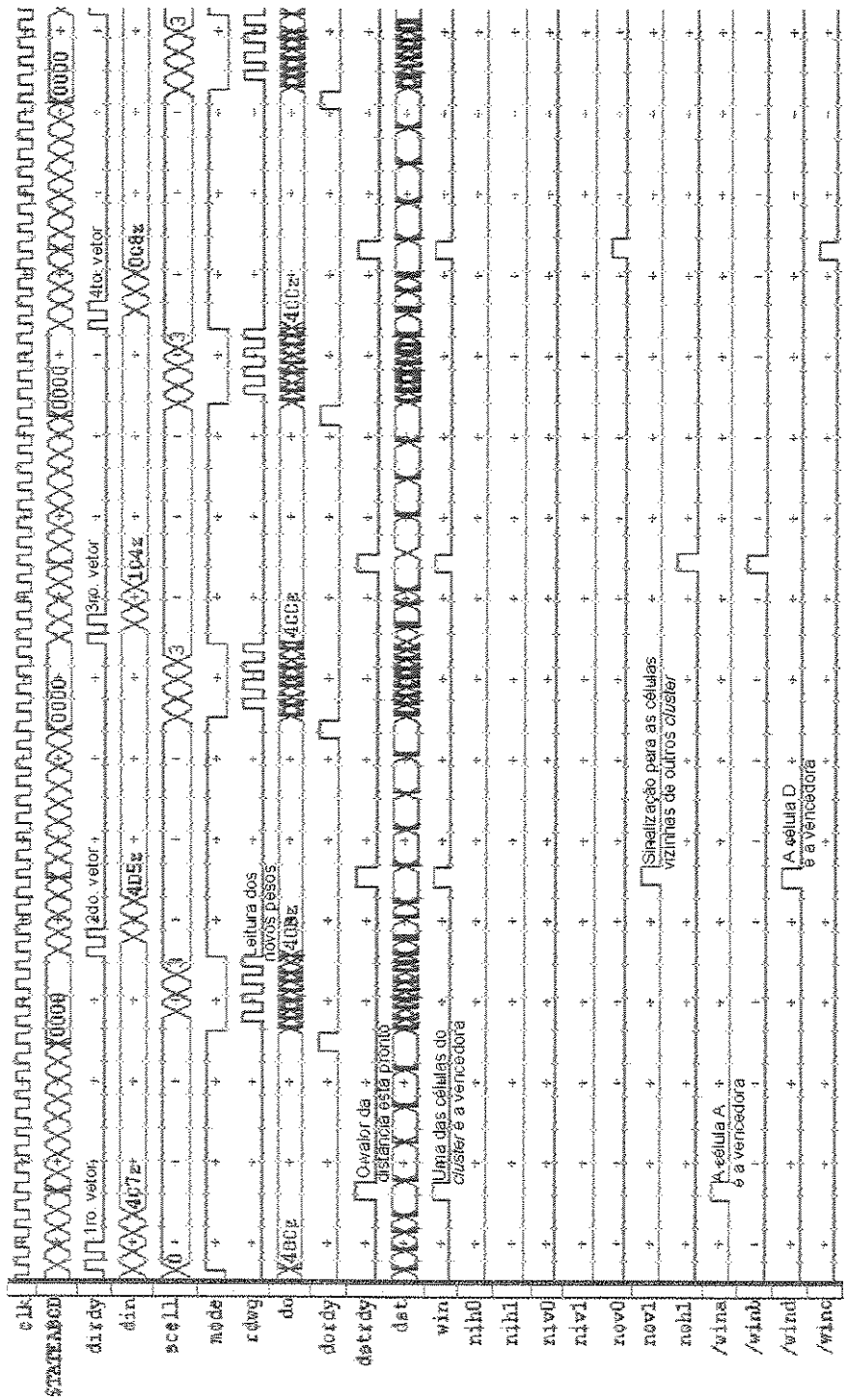


Figura 6.26 Simulação da fase de reconhecimento. Têm-se na entrada *(Din)* 4 casos de vetores.

Na figura 6.26, como primeiro caso, a célula A é a vencedora, portanto não é gerado o sinal *No*. Como segundo caso, a célula D é a vencedora, portanto gera-se um pulso na saída *NoV1*. No terceiro caso, B é a célula vencedora, gerando-se um pulso na saída *Noh1*. Como último caso, a célula C é a vencedora que gera o sinal *NoV0* que é a mesma que *NoH0*.

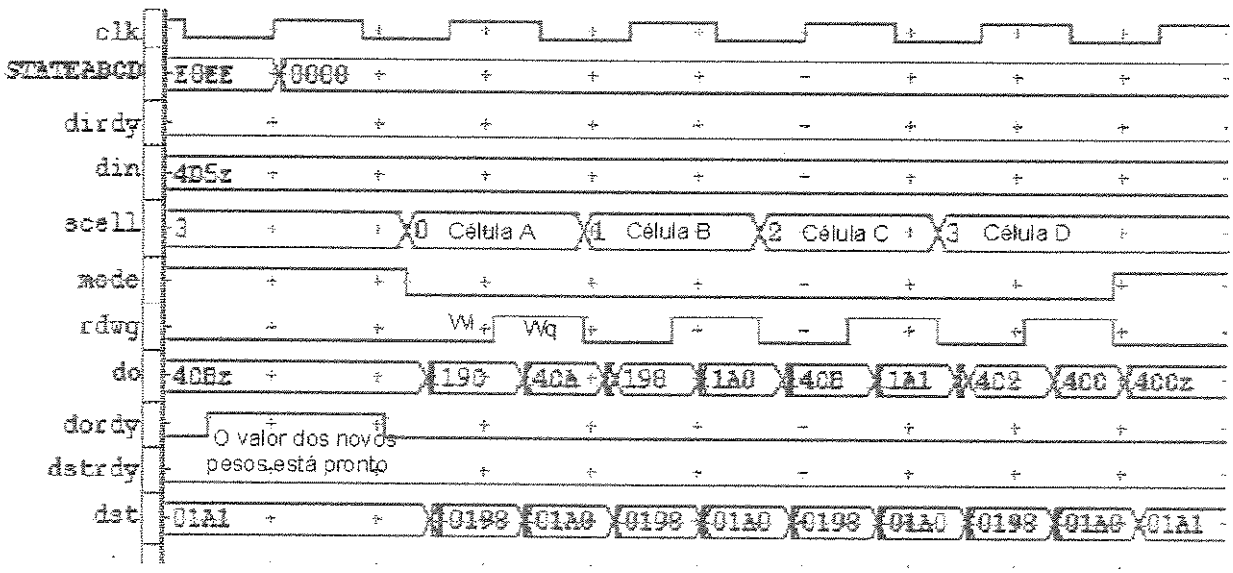


Figura 6.27 Simulação da leitura dos pesos internos já atualizados.

Na figura 6.27 os pesos internos, w_i e w_q, são lidos das células na seqüência A, B, C e D, depois de receber o pulso de aviso de disponibilidade dos pesos mediante o sinal *DoRdy*. Para cada caso o sinal *RdWg* assume primeiro o valor "0" e logo "1". O sinal *Mode* está em "0" durante todo esse período.

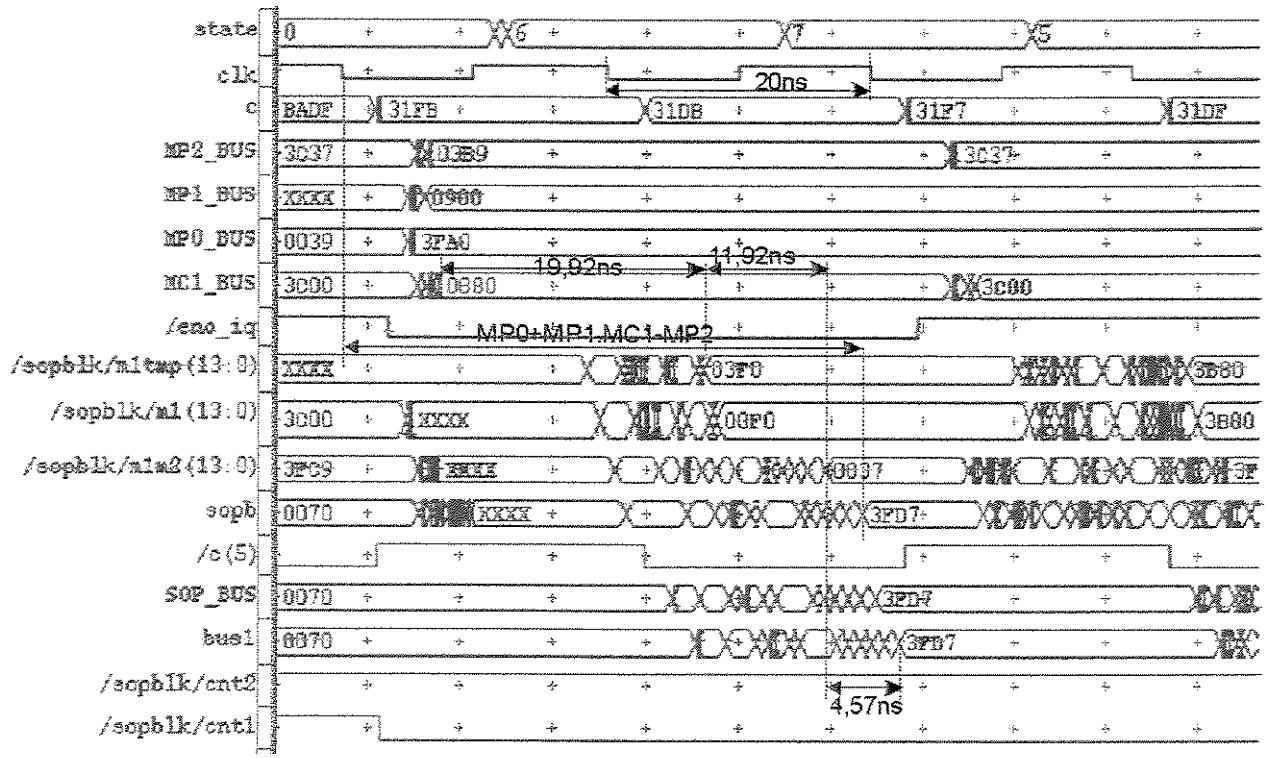


Figura 6.28 Simulação da operação da unidade SOP

A figura 6.28 mostra a operação da unidade SOP para o caso do estado S4 (6_H). SOP_RG recebe a expressão $V_{dc}+V_g u_i - w_i$ ou $MP0 + MP_1 x MC_1 - MP_2$.

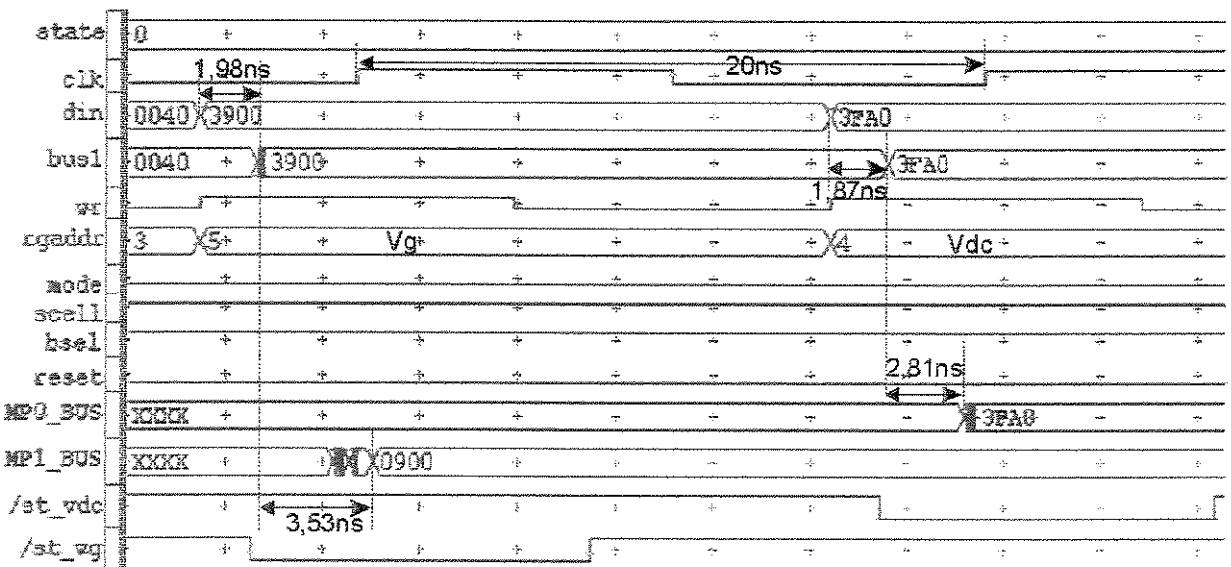


Figura 6.29 Simulação da carga dos registradores sem entrada assíncrona, casos: V_g e V_{dc}

A figura 6.29 mostra a operação de carga dos registradores V_g e V_{dc}. Para tanto nos

sinais *ST_Vg* e *ST_Vdc* produz-se um pulso negativo, reflexos do sinal *WR*. No processo, *Mode="1"*, *Scell="1"* e *BSel="1"*. Os valores da carga podem ser vistos nos barramentos *BUS1*, MP_0*_BUS* e MP_1*_BUS*.

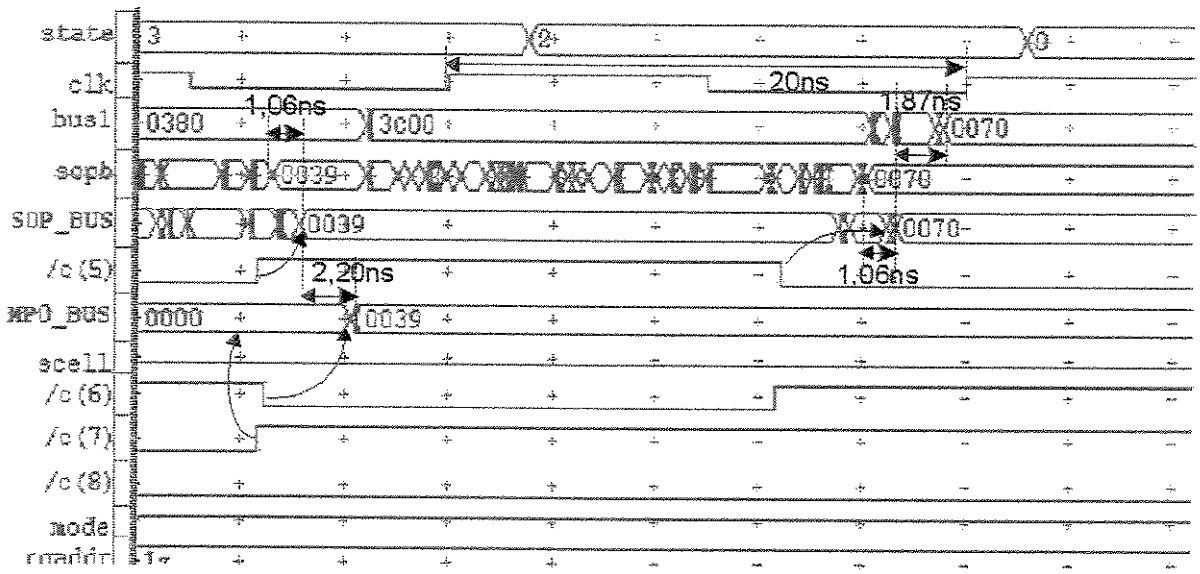
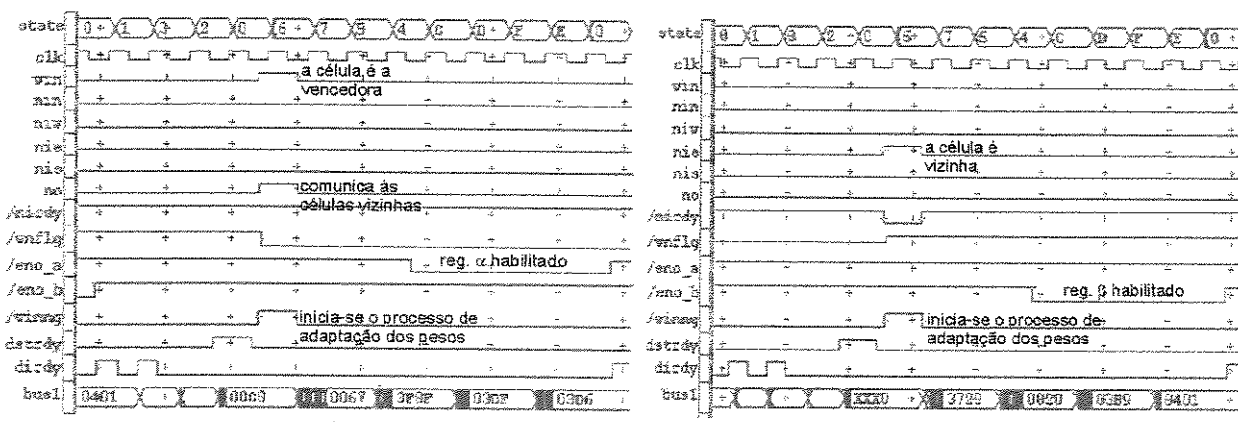


Figura 6.30 Simulação dos registradores *SOP_RG* e *AC*.

A figura 6.30 mostra a carga dos registradores *SOP_RG* e *AC* no caso do estado S3 (2_H). Na seqüência *AC* recebe o valor parcial da distância e o disponibiliza no barramento MP_0*_BUS*. *SOP_RG* recebe o valor parcial e final da distância, este último também disponível no barramento *BUS1*.



a) b)

Figura 6.31 Simulação do bloco interface de vizinhança. a):caso de célula vencedora b) caso de célula na vizinhança.

Na figura 6.31 ilustra-se o funcionamento do bloco interface de vizinhança. No caso a) a célula recebe um pulso em *Win*, do bloco WTA, e se reflete na saída No, com isto a máquina de estados passa a S4(6_H). O saída do *flip-flop*, *WnFlg* então é carregada com o nível

lógico "0" habilitando ao registrador α para a atualização de pesos. No caso b) o pulso chega de uma célula vizinha em *NiE*, *No* permanece invariável, *NiRdy* fixa o *flip-flop* em "1", portanto β é o registrador a ser usado para atualizar os pesos.

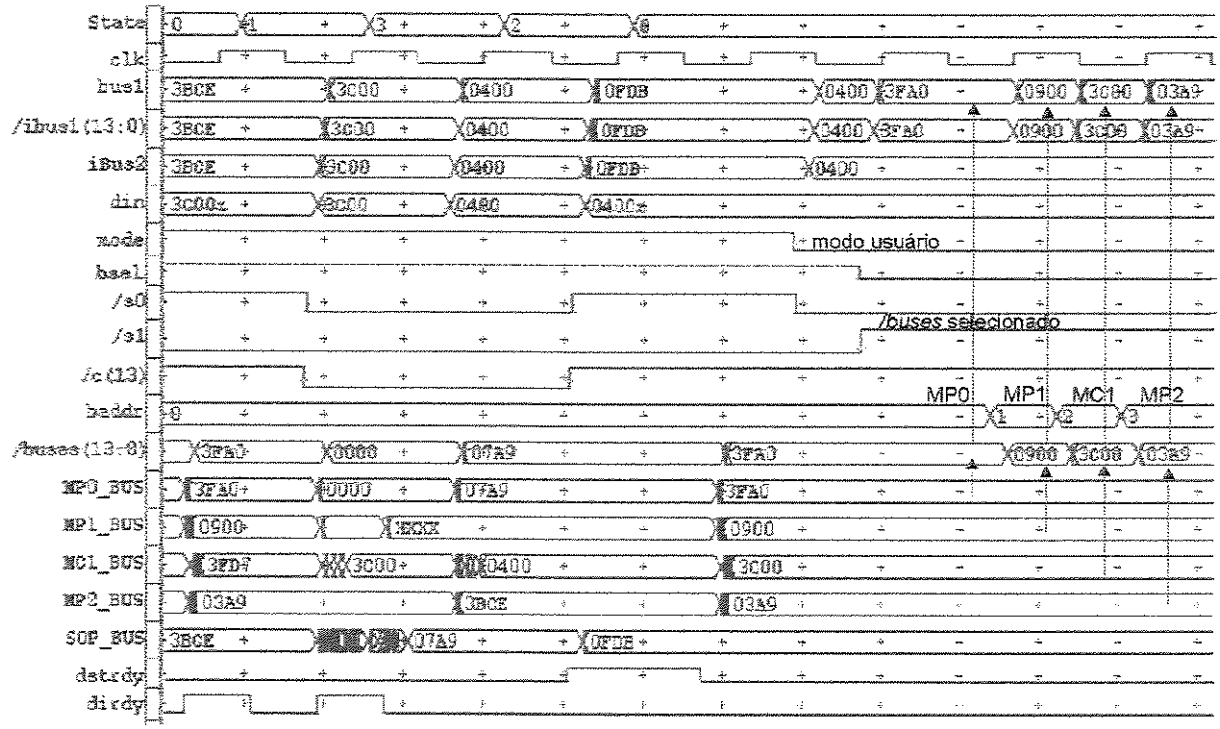


Figura 6.32 Simulação dos blocos de multiplexadores.

Na figura 6.32 mostra-se o laço de cálculo de distância completo da máquina de estados. Durante esses estados a máquina habilita o sinal *s0*, cujo controle é através de *iBSel*, para poder transferir a saída do bloco SOP para o barramento *BUS1*, o sinal *Mode="1"* e o valor de *BSel* não interessam. Após isto os barramentos internos são transferidos para *BUS1* colocando o endereço respectivo em *BAddr*. Neste caso *Mode="0"* e *BSel="0"*

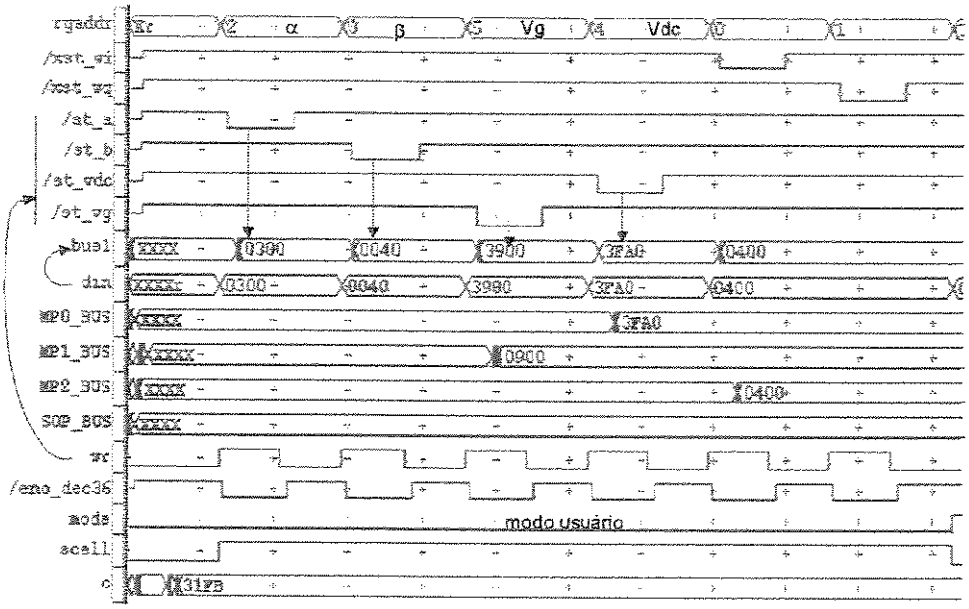


Figura 6.33 Simulação do bloco decodificador de endereços.

Na figura 6.33 observa-se a seqüência de ativações na saída do decodificador de acordo com o endereço dado em *RgAddr*. Todas as saídas do decodificador são para a escrita no registradores, portanto após a ativação de um destes sinais observa-se nos barramentos internos o valor de carga refletido neles. A seqüência é originada pelo trem de pulsos em *WR*.

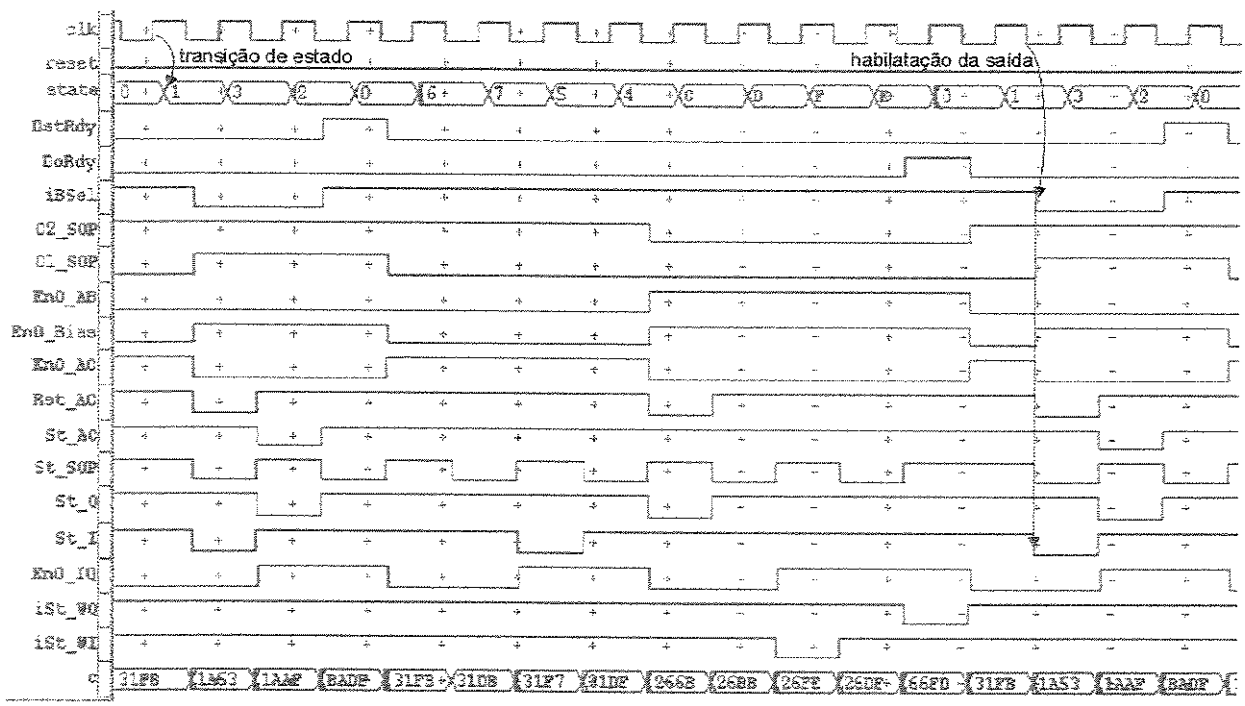


Figura 6.34 Simulações para a operação da máquina de estados.

Na figura 6.34 mostram-se as transições de estados na máquina. Para cada borda de subida do sinal de relógio, dependendo unicamente de *DirRdy* e *WinNg*, há um estado novo e para cada borda de descida a saída decodificada, *C*, dos estados.

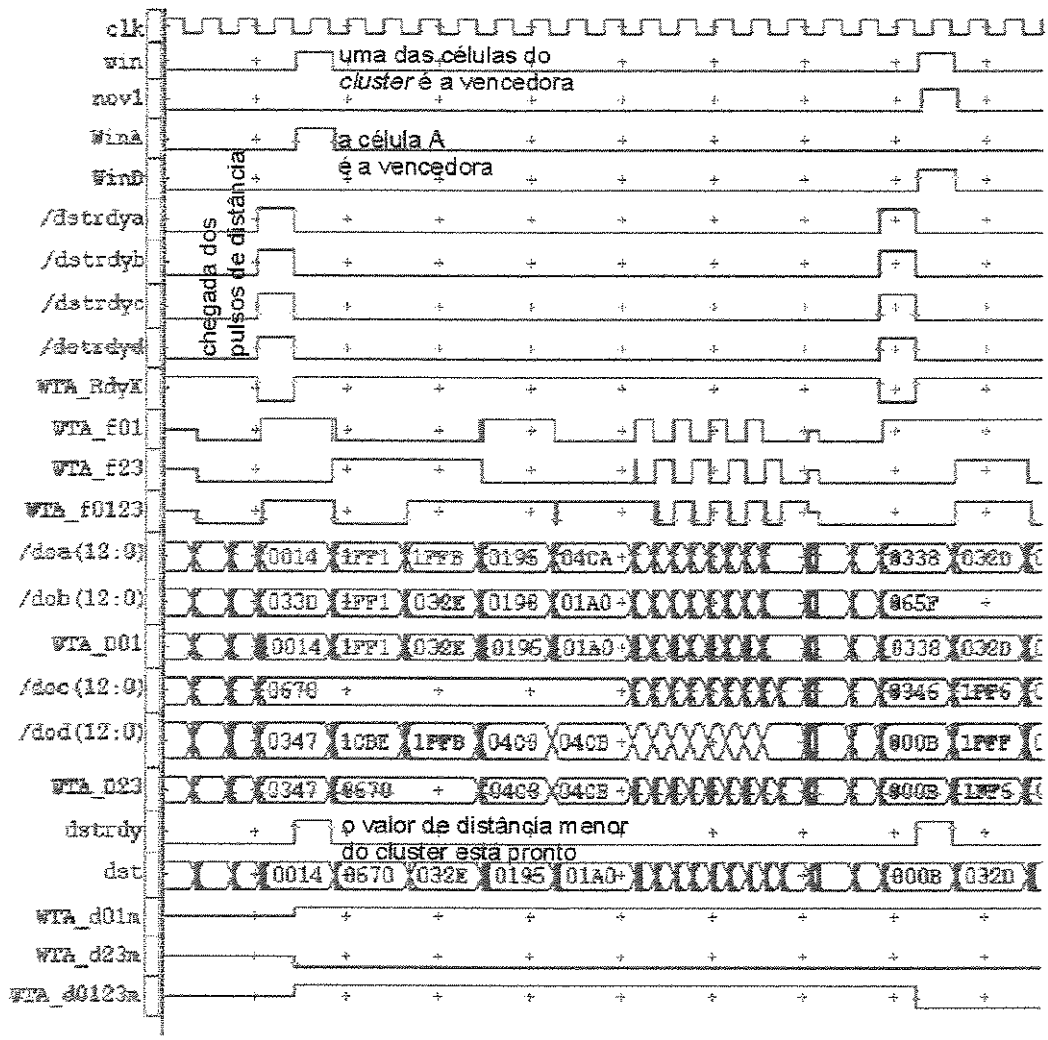


Figura 6.35 Simulações do bloco WTA (Winner-Takes-All)

Na figura 6.35 mostram-se dois casos a) A é a célula vencedora e b) D é a vencedora. Em ambos os casos o bloco WTA recebe os sinais de *DstRdy* de cada uma das células, todos coincidentes. Com isto gera-se o sinal *RdyX*, *d01m*, *d23m* e *d0123m* com os quais se decide pela célula vencedora. Como a saída *DstRdy* do WTA está conectada à entrada *Win* do WTA, gera-se um pulso numa das saídas *WinX (X=A,B,C ou D)*. No caso a) o pulso sai por *WinA* e no caso b) pela saída *WinD*.

6.7 Síntese do *Layout*

A síntese automática de *layouts* para *ASICs* é o processo da sua geração, por meio da colocação de células padrão *(standard cells)* no *layout* e a realização do roteamento entre os blocos ou células, a partir de um esquema (ou de uma *netlist*), chamado de fonte lógica. As etapas da síntese automática de *layout* são as seguintes : *floorplanning*, *placement*, roteamento e compactação. No *floorplanning* faz-se uma estimativa da área do *chip* e determina-se o número de linhas necessárias onde as células padrão irão ser alocadas, a relação da largura pelo comprimento do circuito pode ser configurada pelo usuário. Na etapa do *placement* as células padrão são efetivamente alocadas e distribuídas nas linhas reservadas na etapa anterior. No roteamento a informação da interconectividade entre as células contida na fonte lógica é utilizada para realizar a conexão física por meio das camadas de roteamento (em geral metal). Finalmente na etapa da compactação é realizado um enxugamento da área ocupada pelo *chip* de forma a preencher os espaços não utilizados nas etapas anteriores e obter um *layout* onde as distâncias entre os objetos seja a menor permitida pelas regras de projeto da tecnologia alvo. O conjunto de ferramentas *IC Station* da empresa *Mentor Graphics Corp.* suporta todas as operações mencionadas.

A tecnologia utilizada foi a AMS-CMOS 0.8μm cujas células padrão estão contidas no *Design Kit* [Ams95a] da AMS. Na figura 6.36, a título de exemplo, mostra-se uma região do *layout* final da unidade WTA. Observa-se neste *layout* a organização das células em filas e entre cada fila um espaço denominado de roteamento. As linhas verticais são de uma camada de metal e as horizontais de outra também de metal. Nos cabeçalhos e pés de cada fila têm-se linhas metálicas largas que são as trilhas de alimentação. Na vertical têm-se duas trilhas largas que unem todas as linhas de alimentação das filas. Nos cantos do *layout* são alocados os pinos do bloco; observe-se a diferença entre um pino de alimentação e outro de sinal.

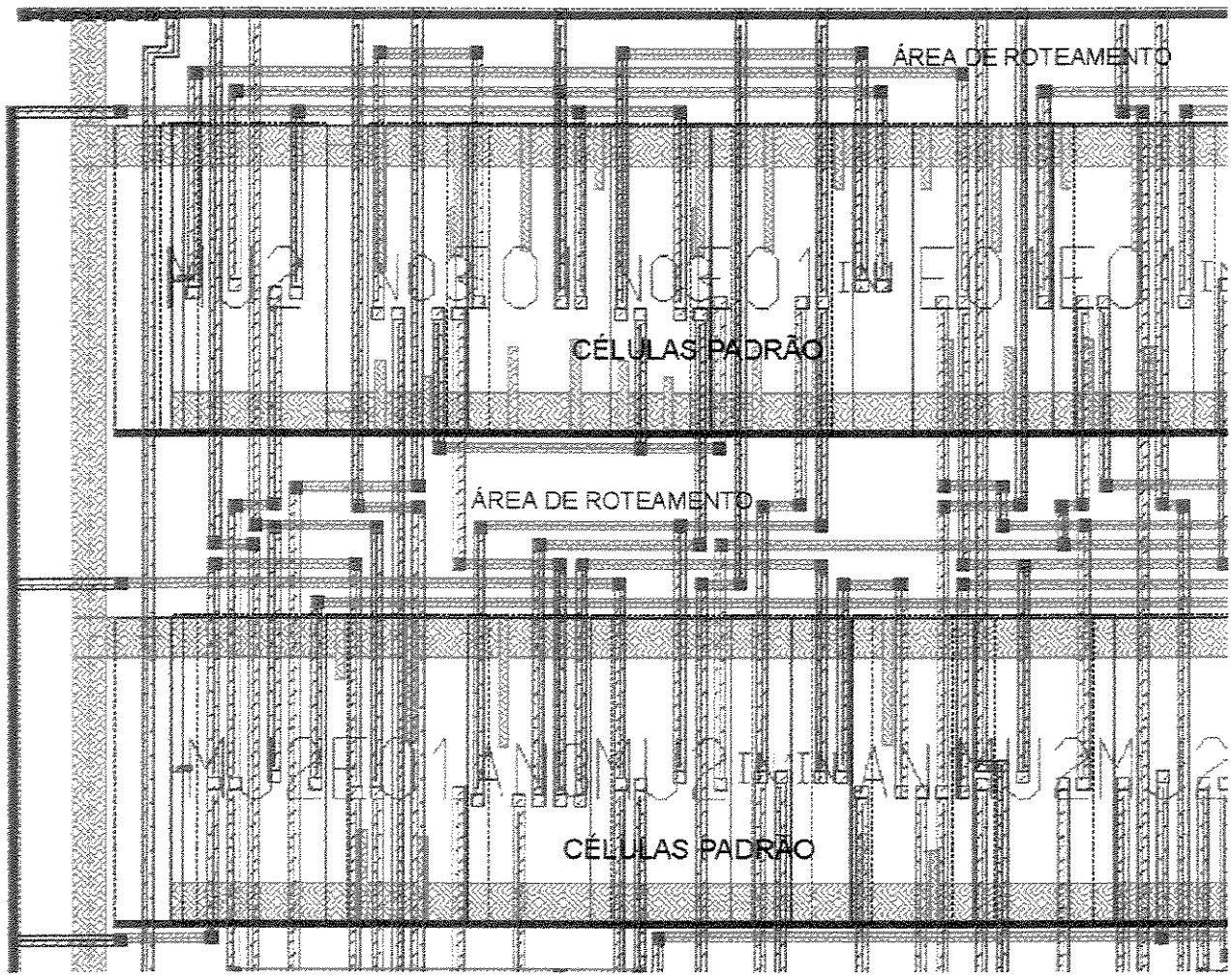


Figura 6.36 Ilustração de uma região de *layout* do bloco WTA.

Na figura 6.37 mostra-se o *layout* final do circuito fotografado após o recebimento dos protótipos encomendados à EUROPRACTICE. A área aproximada total do ASIC incluindo os PADS é de 25 mm^2.

6.8 Verificação do *Layout* - Extração de Parâmetros Parasitas

A verificação do *layout* é o processo de interpretar os dados do *layout* físico e determinar se estão dentro das regras físicas e elétricas de projeto, compara-lo com a fonte lógica (esquema) e interpretar a informação de parâmetros parasitas, os quais podem ser utilizados para ver se o circuito está de acordo às especificações. Este processo inclui: extração de conectividade, reconhecimento de dispositivos, extração de parâmetros parasitas (PEX), verificação das regras de projeto e de geometria (ERC, DRC) e comparação do *layout* com o esquema (LVS). As ferramentas da *Mentor Graphics Corp.* que dão o suporte estão no conjunto denominado *ICverify*, que por sua vez são baseadas no arquivo de regras, o qual define as regras de extração de conectividade, definições para o reconhecimento de

dispositivos, constantes de proporcionalidade para a extração de parâmetros e regras de projeto.

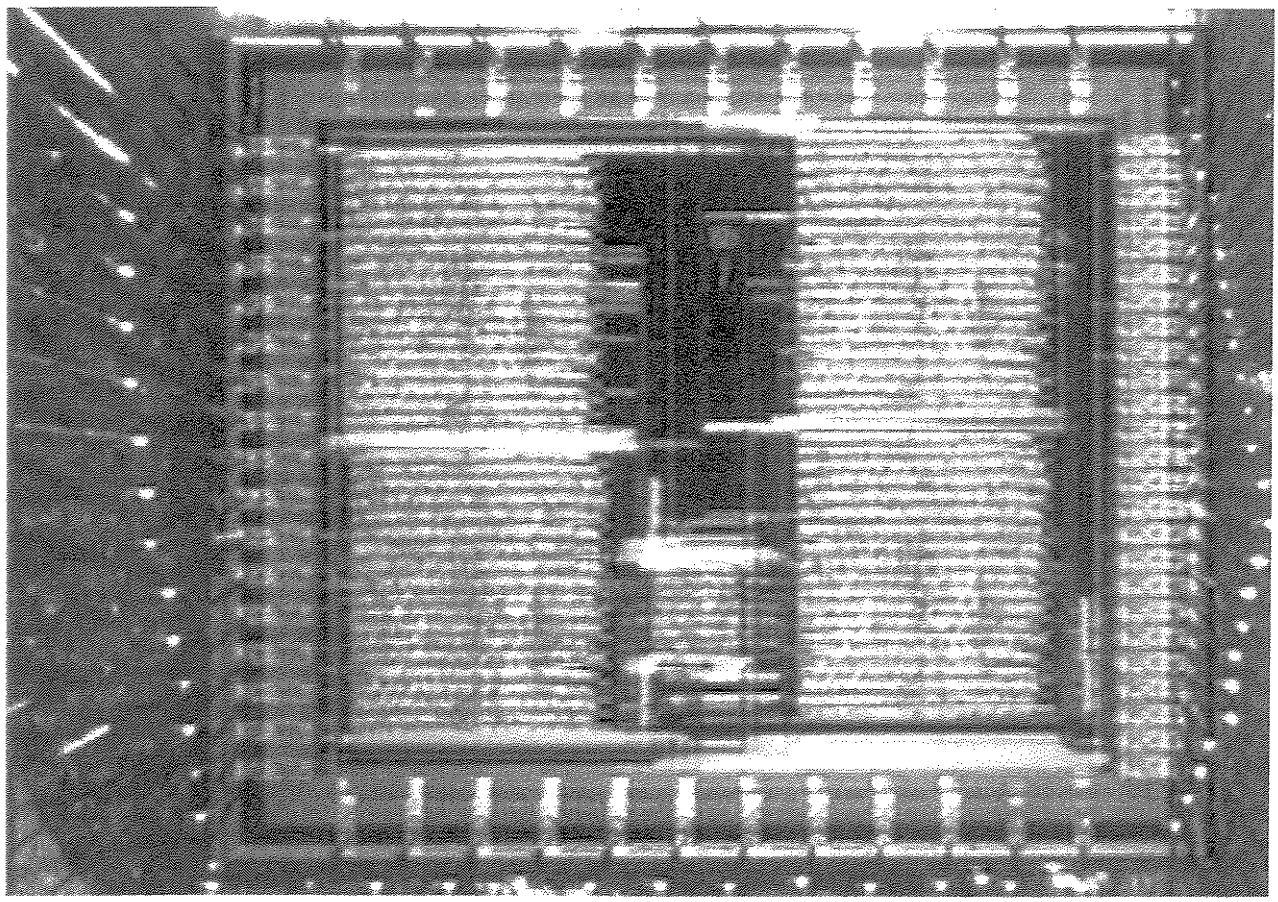


Figura 6.37 Fotografia do protótipo do circuito implementado na EUROPRACTICE. Área total do ASIC, incluindo os PADS: ~25 mm^2; área de uma célula-neuônio ~2 mm^2, tecnologia: AMS-CMOS 0.8μm; alimentação: 5V.

Os efeitos das capacitâncias e resistências parasitas não são intencionalmente projetados no *chip*, mas resultam da elaboração do *layout* e normalmente degradam o desempenho do circuito. O cálculo dos valores desses parâmetros para as *nets* do circuito baseia-se no conceito de dois tipos de camadas, as de condução e as de conexão.

As capacitâncias parasitas podem ser divididas em duas classes: a primeira chamada de intrínseca, capacitância entre uma camada de condução e a base, cujas componentes são a de corpo (proporcional à área da geometria de condução) e a de borda (proporcional ao perímetro da geometria de condução). A Segunda, conhecida como de acoplamento, é a capacitância entre duas *nets* diferentes nas camadas de condução. Seus componentes são o de superposição de cruzamento (proporcional à área comum de duas camadas que se sobrepõem), o de borda de cruzamento (proporcional ao perímetro do cruzamento) e o de corpo adjacente (entre duas *nets* diferentes e paralelas na mesma camada de condução). Os parâmetros do processo, disponíveis no *Design Kit* da tecnologia, fornecem as constantes de proporcionalidade para a capacitância.

A capacitância parasita pode ser calculada, sem incluir a resistência, para todos ou *algum dos efeitos descritos anteriormente. Esses efeitos podem ser reportados individualmente* ou ser somados para gerar a capacitância *lumped* para cada *net*, a qual pode ser retro-anotada *para o esquema do circuito.*

As resistências parasitas têm duas constituintes: a resistência de folha e a resistência de *conexão. A resistência de folha esta presente para fluxo de corrente dentro de uma camada de* condução, sendo medida pela resistividade de folha em Ω/□. A resistência de conexão está presente para camadas de conexão, tais como os *contatos* e as *vias*, que conectam camadas de condução diferentes.

O circuito da rede neural passou pelos processos de verificação de DRC, ERC, LVS e extração de parâmetros no modo *lumped*, os quais foram retro-anotados à fonte lógica para a sua resimulação.

6.9 Simulação Pós-*Layout*

Uma vez realizado o processo de extração de parâmetros parasitas, estes são armazenados numa entidade chamada de *backannotation*.

O projeto de circuitos ou sistemas dentro do ambiente das ferramentas da *Mentor Graphics Corp.* baseia-se em entidades chamadas de *viewpoints (vpt)*, através das quais é possível fazer projeto concorrente. Cada projetista pode definir um *vpt* para um mesmo circuito, através do qual o "enxerga", podendo mudar os seus parâmetros sem mudar os parâmetros originais do circuito, os quais podem estar sendo utilizados por outros projetistas. A informação contida no *backannotation* é ligada (retro-anotada) ao circuito através de um *viewpoint*. Este conceito é ilustrado no diagrama da figura 6.38.

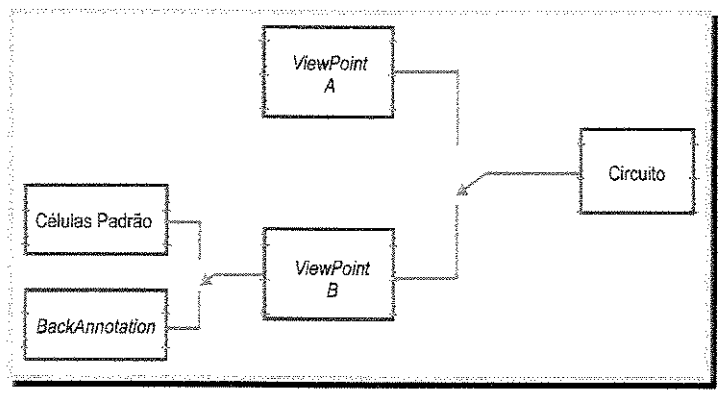


Figura 6.38 Ilustração dos conceitos de *viewpoint* e *backannotation*.

A seguir apresentam-se, a título de exemplo, os resultados da simulação antes e após a realização do *layout*, extração dos parâmetros parasitas e *backannotation*, para o caso da unidade SOP a qual têm o maior tempo de propagação e é a que determina qual o mínimo período do sinal de relógio, *CLK*.

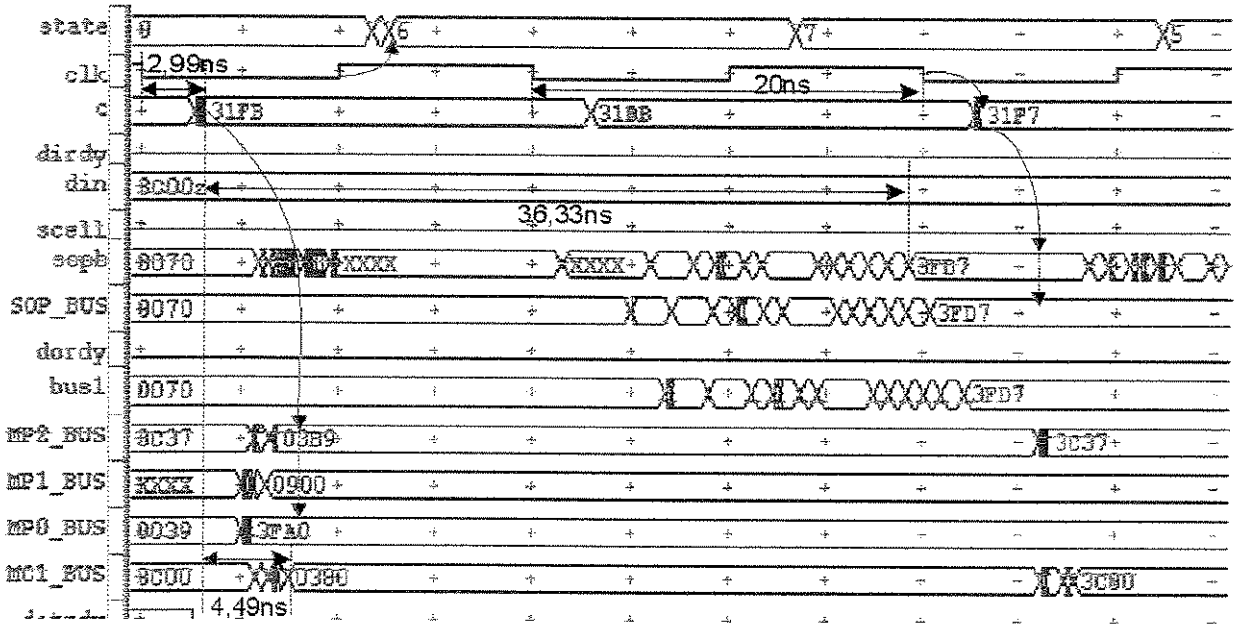


Figura 6.39 Detalhe da simulação da unidade SOP com período de relógio de 20ns sem ligar o *backannotation* ao circuito.

Na figura 6.39 mostram-se as simulações da unidade SOP sem a informação do *backannotation* ligada ao circuito, sendo o período do sinal de relógio de 20ns. Na saída da unidade SOP, no barramento SOP_BUS, tem-se o resultado da operação $V_{dc}+V_g u_i - w_i$, cujo valor é "3FD7" (vide figura 6.28).

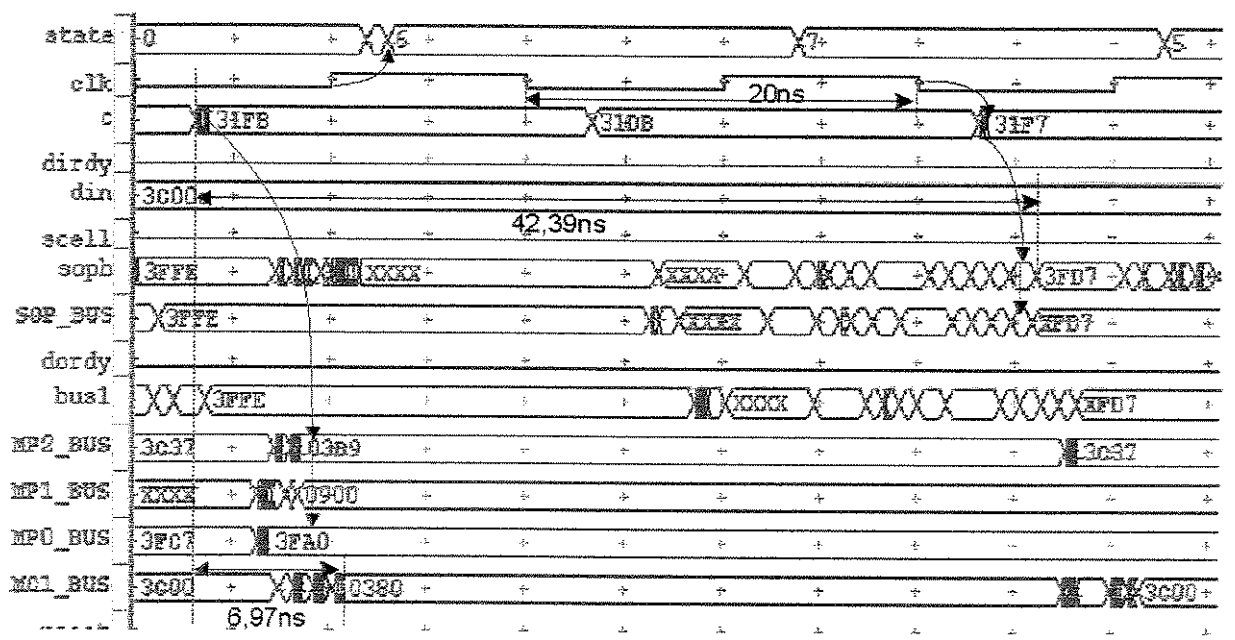


Figura 6.40 Detalhe da simulação da unidade SOP com período de relógio de 20ns com o *backannotation* ligado ao circuito.

Na figura 6.40 mostra-se a simulação da unidade SOP mostrando o mesmo trecho do que da figura 6.39, com o sinal de relógio tendo o período de 20 ns, após a informação do *backannotation* ser incorporada ao circuito. O resultado mostrado na figura 6.39 não se reproduz, no barramento SOP_BUS tem-se o valor "XFD7", indicando que o período do sinal de relógio não foi suficiente e portanto é necessário ampliá-lo ou voltar à etapa de projeto da unidade SOP e otimizá-la para se obter tempos de propagação menores.

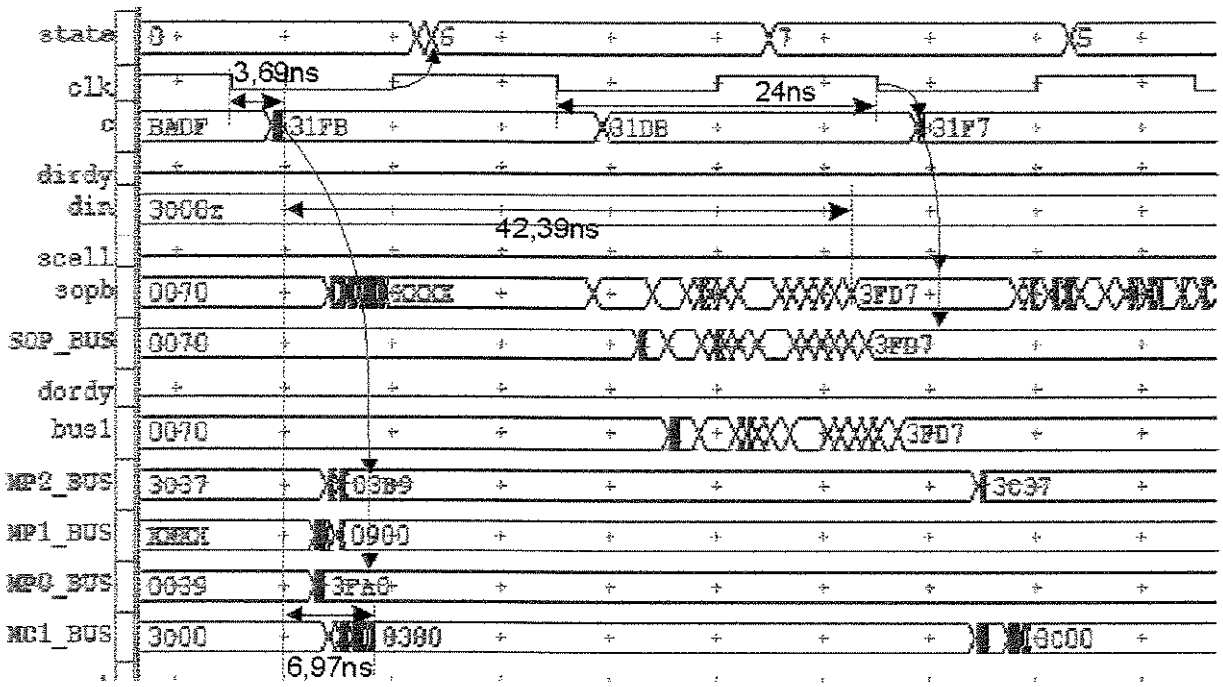


Figura 6.41 Detalhe da simulação da unidade SOP com período de relógio de 24ns com a informação do *backannotation* ligada ao circuito.

Na figura 6.41 mostra-se a simulação da unidade SOP (com o mesmo que trecho do que na figura 6.39) com o sinal de relógio de período 24 ns. O resultado da operação $V_{dc}+V_g u_i - w_i$, é o mesmo do na figura 6.39, porém com um período de relógio maior.

6.10 Características do ASIC

A seguir apresentam-se algumas características do ASIC obtidas após a simulação:

- Freqüência do sinal de relógio (valor máximo simulado) :

$$Fclk = \frac{1}{24ns} = 41{,}67\ MHz$$

- Velocidade de recepção (vide seção 6.5.2.6) :

$$Srate = \frac{1}{(12\ ciclos)(1/Fck)} = 3{,}47x10^6\ \ Símbolos/s$$

- Potência dissipada : 49,525 mW para a freqüência de 1 MHz

- Tamanho em portas equivalentes : 13690,5
- Encapsulamento : JLCC68
- Número de pinos : 65

Na tabela a seguir apresentam-se o tamanho e a potência que dissipam os principais blocos da rede neural :

Bloco	Tamanho [portas equivalentes]	Potência a 1 MHz [mW/MHz]
Rede Neural	13,690,5	49,525
Célula - Neurônio	3284	11,233
Multiplicador de 13x14 bits	1638	6,103
Bloco Soma de Produtos de 14 bits	2115,5	7,980
Bloco WTA de 13 bits - 4 entradas	380,5	1,493

Capítulo 7:

Testes

Neste capítulo apresentam-se os resultados dos testes realizados no protótipo do ASIC que implementa a rede neural de Kohonen para aplicações em equalização de sinais. A realização dos testes teve como objetivo principal a verificação funcional da implementação em *hardware*, cujo projeto foi apresentado no capítulo 6. Os resultados obtidos são comparados aos obtidos na simulação do circuito.

Os testes foram realizados com o auxílio do módulo de desenvolvimento *DSP96002ADM (Application Development Module)* da *Motorola*[Mot_a] o que demonstrou ser uma opção interessante, dada a versatilidade desse módulo. Numa aplicação em tempo real a utilização da conjunção de um DSP *(Digital Signal Processor)* comercial e do ASIC projetado seria uma boa solução, onde todo o equalizador, com exceção do módulo de decisão, seria implementado no DSP.

Para a realização do teste foi necessário o projeto de uma placa de interface com o módulo *DSP96002ADM*, que alojasse o *chip* projetado. Foram desenvolvidos rotinas na linguagem *assembly* do *DSP96002*[Mot89], para o interfaceamento e para o envio dos vetores-símbolo à rede neural.

7.1 Esquema do Teste

Com base nas simulações C feitas para a estrutura de equalizador DFE-SOM foram realizadas as simulações com a ferramenta de simulação lógica *QuickSim II* da *Mentor Graphics Corp*. Os símbolos de um dos quatro quadrantes da constelação QAM-16 do sinal *r(k)*, foram utilizados como vetores de teste do *chip* (vide figura 5.7).

Dessas simulações foram gerados dois sinais-barramentos (vetores de teste) *DSP96K_IN* e *DSP96K_OUT* que correspondem aos sinais de entrada e de saída, respectivamente (vide seção 6.5.1). A conformação desses barramentos é mostrada na seguinte tabela:

Porta A do DSP96002ADM	aD31	aD30	aD29	aD28	aD27	aD26	AD25	aD24
DSP96K_IN		Reset	CLK	DiRdy	Win	Mode	WR	RdWg
DSP96K_OUT			NoH1	Dst(12)	Dst(11)	Dst(10)	Dst(9)	Dst(8)

Porta A do DSP96002ADM	aD23	aD22	aD21	aD20	AD19	aD18	aD17	aD16
DSP96K_IN	NiH1	NiH0	NiV1	NiV0	BSel	RgAddr2	RgAddr1	RgAddr0
DSP96K_OUT	Dst(7)	Dst(6)	Dst(5)	Dst(4)	Dst(3)	Dst(2)	Dst(1)	Dst(0)

Porta A do DSP96002ADM	aD15	aD14	aD13	aD12	aD11	aD10	aD9	aD8
DSP96K_IN	Baddr 1	Baddr0	Scell1	Scell0	Din(11)	Din(10)	Din(9)	Din(8)
DSP96K_OUT	NoV1	NoV0	DstRdy	DoRdy	Do(11)	Do(10)	Do(9)	Do(8)

Porta A do DSP96002ADM	aD7	aD6	aD5	aD4	aD3	AD2	aD1	aD0
DSP96K_IN	Din(7)	Din(6)	Din(5)	Din(4)	Din(3)	Din(2)	Din(1)	Din(0)
DSP96K_OUT	Do(7)	Do(6)	Do(5)	D4(4)	Do(3)	Do(2)	Do(1)	Do(0)

Os vetores de teste gerados foram aplicados ao *chip* através da placa e rotinas de interface. Finalmente, fez-se uma comparação dos resultados das simulações no *Quicksim II* (vetor *DSP96K_OUT*) com os dados lidos do *chip* e procedeu-se a construção dos gráficos para um dos quadrantes da constelação QAM-16, em diferentes instantes da simulação. Nos diagramas das figuras 7.1, 7.2 e 7.3 mostram-se os barramentos *DSP96K_IN* e *DSP96K_OUT* para as fases de configuração, apresentação dos vetores de entrada e leitura dos pesos internos, respectivamente.

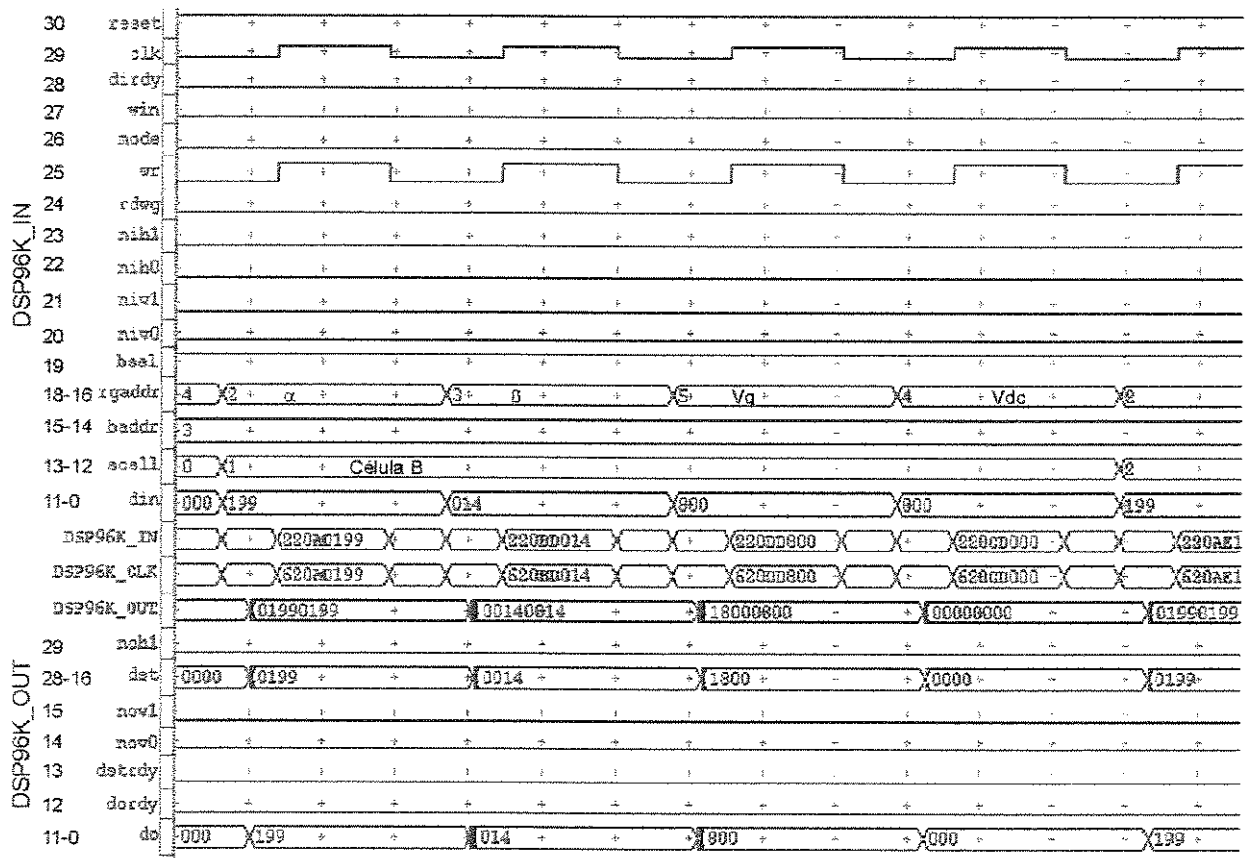


Figura 7.1 Sinais dos barramentos *DSP96K_IN* e *DSP96K_OUT* gerados na simulação lógica para sua utilização como vetores de teste na etapa de **configuração** do chip da rede neural.

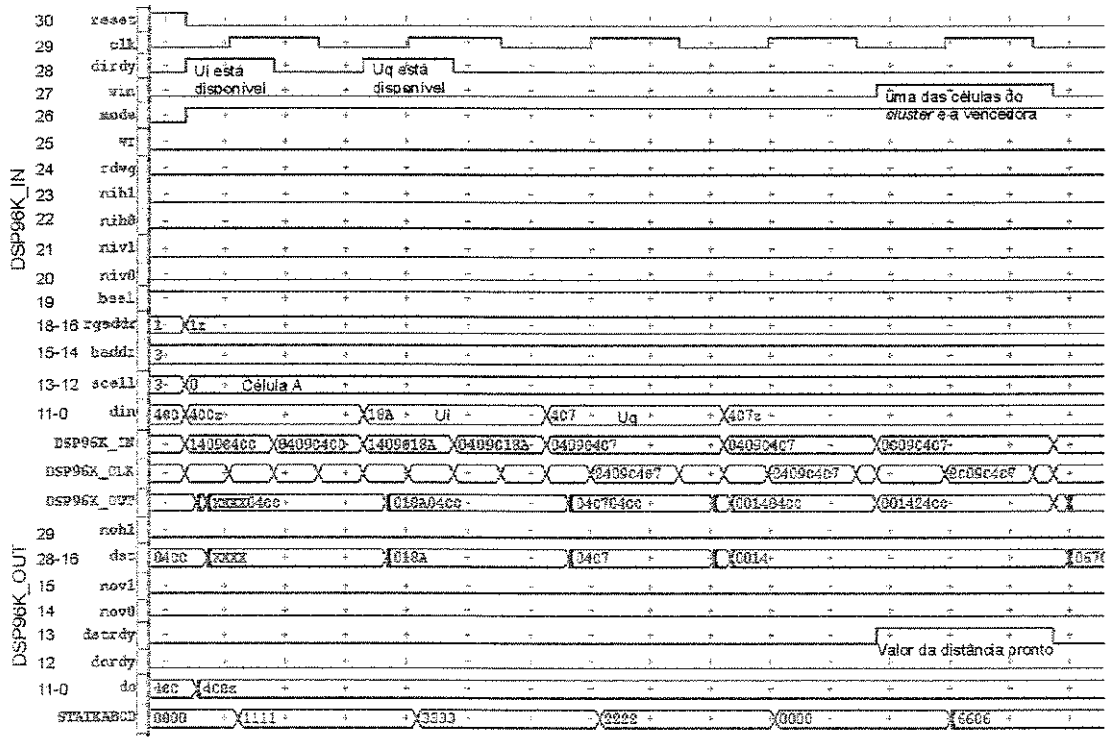


Figura 7.2 Sinais dos barramentos *DSP96K_IN* e *DSP96K_OUT* gerados na simulação lógica para sua utilização como vetores de teste na etapa de **treinamento** do chip da rede neural.

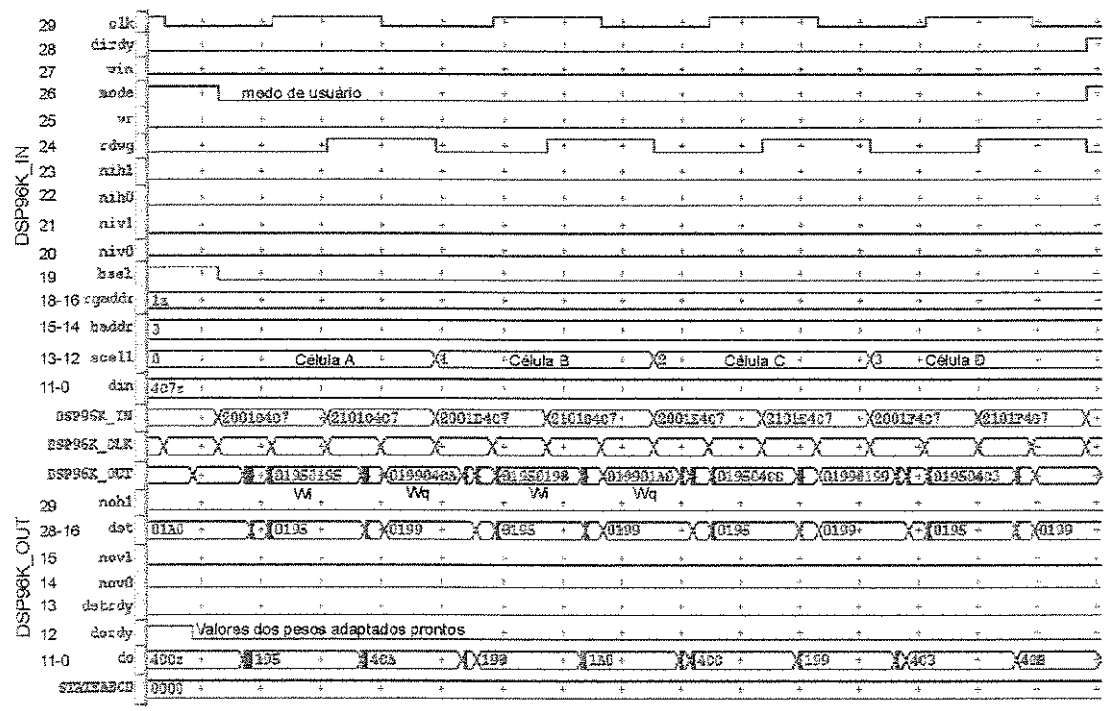


Figura 7.3 Sinais dos barramentos *DSP96K_IN* e *DSP96K_OUT* gerados na simulação lógica para sua utilização como vetores de teste na etapa de **leitura dos pesos internos** do chip da rede neural.

7.2 O Módulo DSP96002ADM da *Motorola*

O sistema de desenvolvimento de aplicações *(ADS - Application Development System)* da *Motorola* é uma ferramenta que permite projetar e realizar o teste de aplicações complexas de software para Processamento Digital de Sinais e produtos de *hardware*. O Módulo DSP96002ADM contém o DSP96002 e circuitos adicionais para o desenvolvimento dos testes de bancada.

As características principais do DSP96002ADM são:

- Tem o processador digital de sinais de 32 bits DSP96002
- Operação a 40 MHz
- Memória RAM estática de 128K palavras, expansivel para 512K
- Memória EPROM de 2K palavras, expansivel ate 64K
- Operação autônoma após desenvolvimento inicial do algoritmo.
- Suporte completo para mapas de memória de dados múltiplos
- Dois conectores de 96 pinos que permitem o acesso completo aos pinos do DSP96002
- Conector para porta de emulação *on-chip (OnCE - On Chip Emulator)*. Permite ao usuário avaliar o *software* e o *hardware* de uma aplicação sem ter um emulador especial.

Na figura 7.4 mostra-se o diagrama de interconexão da estrutura de teste do *chip* projetado baseada em PC. O computador hospedeiro (um PC no caso) contém uma placa de interface e comunicação com o conversor de comandos através de um cabo de 37 pinos. O conversor de comandos se comunica com o módulo de desenvolvimento (ADM) através de um cabo de 14 pinos. O ADM pode-se comunicar com a placa de aplicação específica (do usuário) por meio de dois portos, A e B.

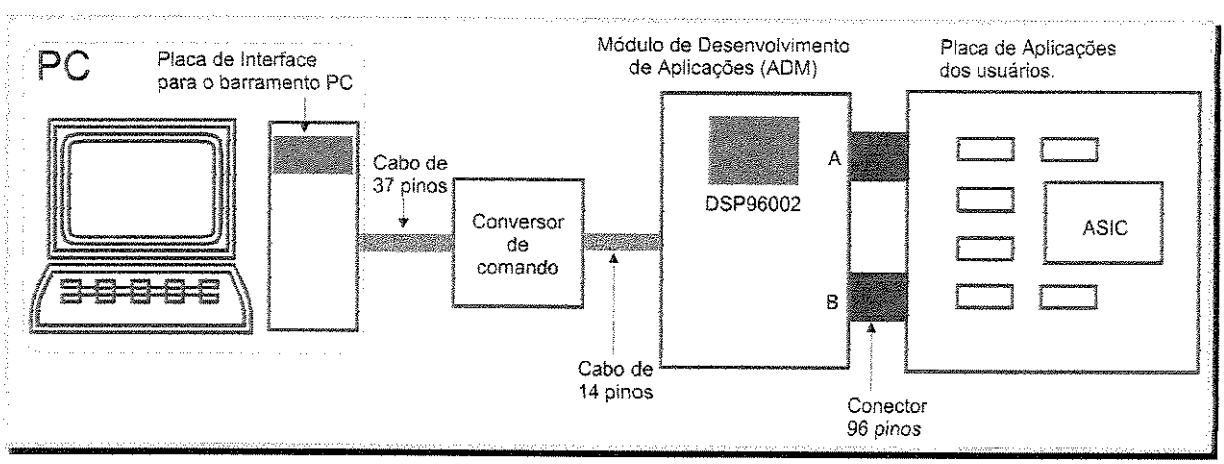


Figura 7.4 Diagrama de interconexão entre um PC e a placa que contém o *chip*.

7.3 A Placa de Interface

O projeto da placa de interface foi baseado nas informações dos vetores de teste e do módulo de desenvolvimento DSP96002ADM. Seu esquema é mostrado na figura 7.5. Seus blocos principais são: o decodificador de endereços, o *latch* de dados, o *buffer tri-state* e o chip da rede neural.

O decodificador decodifica o endereço $FFFF FFFD mais os sinais *S0*, *S1* e *TS*,

tornando disponível o sinal de habilitação na saída de U2. Os sinais *S0* e *S1* mapeiam o espaço de memória, sendo para nosso caso "00" a memória Y. O sinal *TS (Transfer Strobe)* vai para o nível "0" sempre que os sinais de endereço, *S0, S1* e de leitura/escrita (R/W) estejam estáveis. As portas de inversão de U1 servem para adequar os sinais de endereço, *S0, S1, READ* e *TS* ao nível de ativação "0" ou "1" conveniente.

O sinal na saída de U2 habilita tanto a escrita quanto a leitura dos vetores de teste de entrada e de saída, respectivamente. Quando o sinal R/W está em "1" realiza-se a leitura e quando está em "0" a escrita. A transferência do vetor de teste *DSP96K_IN* ao chip faz-se na borda de subida do sinal *WRITE*.

Os *flip-flops* U5-U8 formam o *latch* de entrada de 32 bits para o vetor de teste *DSP96K_IN* e U9-U12 o *buffer tri-state* na saída para o barramento *DSP96K_OUT*.

No anexo 8 pode-se observar a pinagem completa do *chip* além da distribuição do componentes na placa de teste. Na figura 7.6 mostra-se a foto da conjunção do módulo DSP96002ADM e da placa de interface contendo o *chip*.

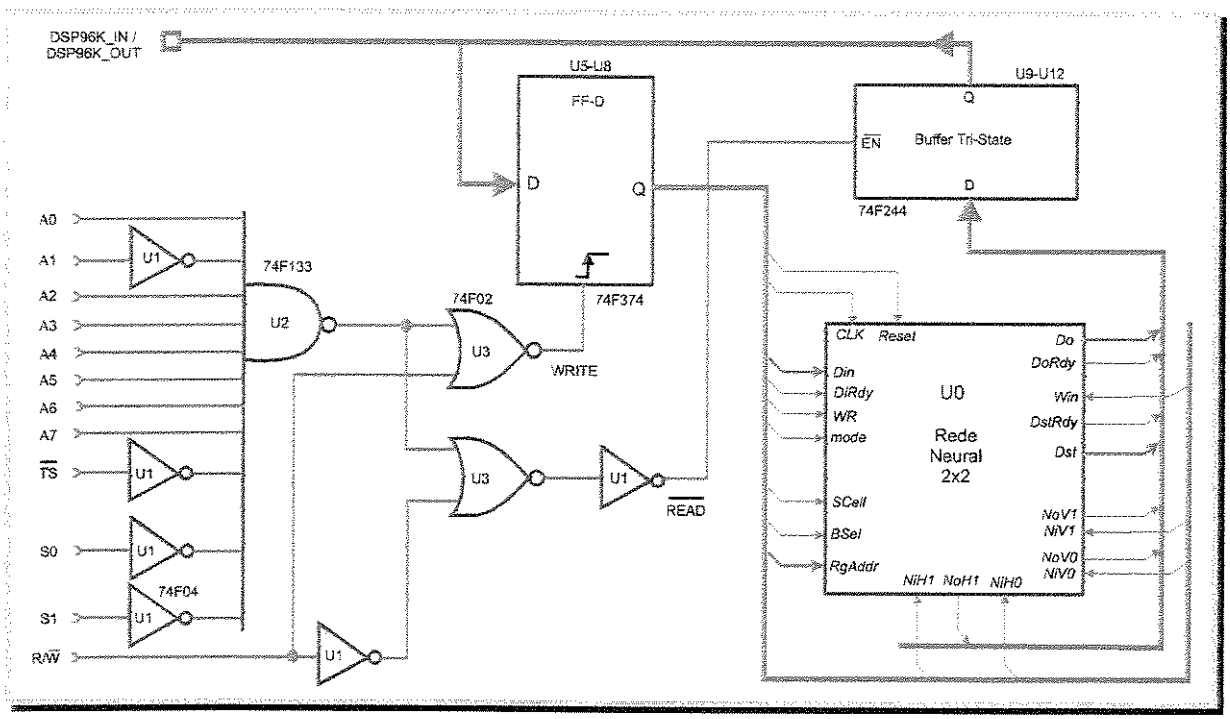


Figura 7.5 Diagrama esquemático da placa interface.

7.4 O Programa de Interface e Teste

Foram implementadas duas versões da rotina de interfaceamento entre o módulo DSP96002ADM e o *chip* na linguagem *Assembly* em função da maneira de gerar o sinal de relógio *CLK*. A primeira faz uso de um dos *timers* internos disponíveis no módulo de desenvolvimento e a segunda gera o sinal por programa.

Na primeira versão o *timer* interno tem de ser programado antes da sua habilitação num dos dois portos de saída do módulo de desenvolvimento. Esse *timer* interno pode

funcionar no modo de pulso ou trem de pulsos, cuja programação faz-se por meio do *registrador TCSR0 ou TCSR1 (dependendo do timer utilizado)*. A freqüência do sinal de relógio disponível nos pinos *TI0* ou *TI1* dos portos, depende do valor carregado no registrador *TCR0 ou TCR1*, respectivamente. É necessário também configurar o nível de prioridade da interrupção através do registrador *IPR*. A freqüência máxima que é possível gerar na saída dos *timers* é de 10 MHz, ou seja ¼ do valor do sinal de relógio interno ao módulo de desenvolvimento. Um segmento de programa em *Assembly* para a configuração do *timer* pode observar-se na figura 7.7. O programa completo do interfaceamento está no anexo 6.

O *jumper JG1* na placa de interface (vide anexo 8) que seleciona a fonte do sinal de relógio, seja do *timer* ou da palavra de teste, deve corresponder à escolha da versão da rotina de interfaceamento.

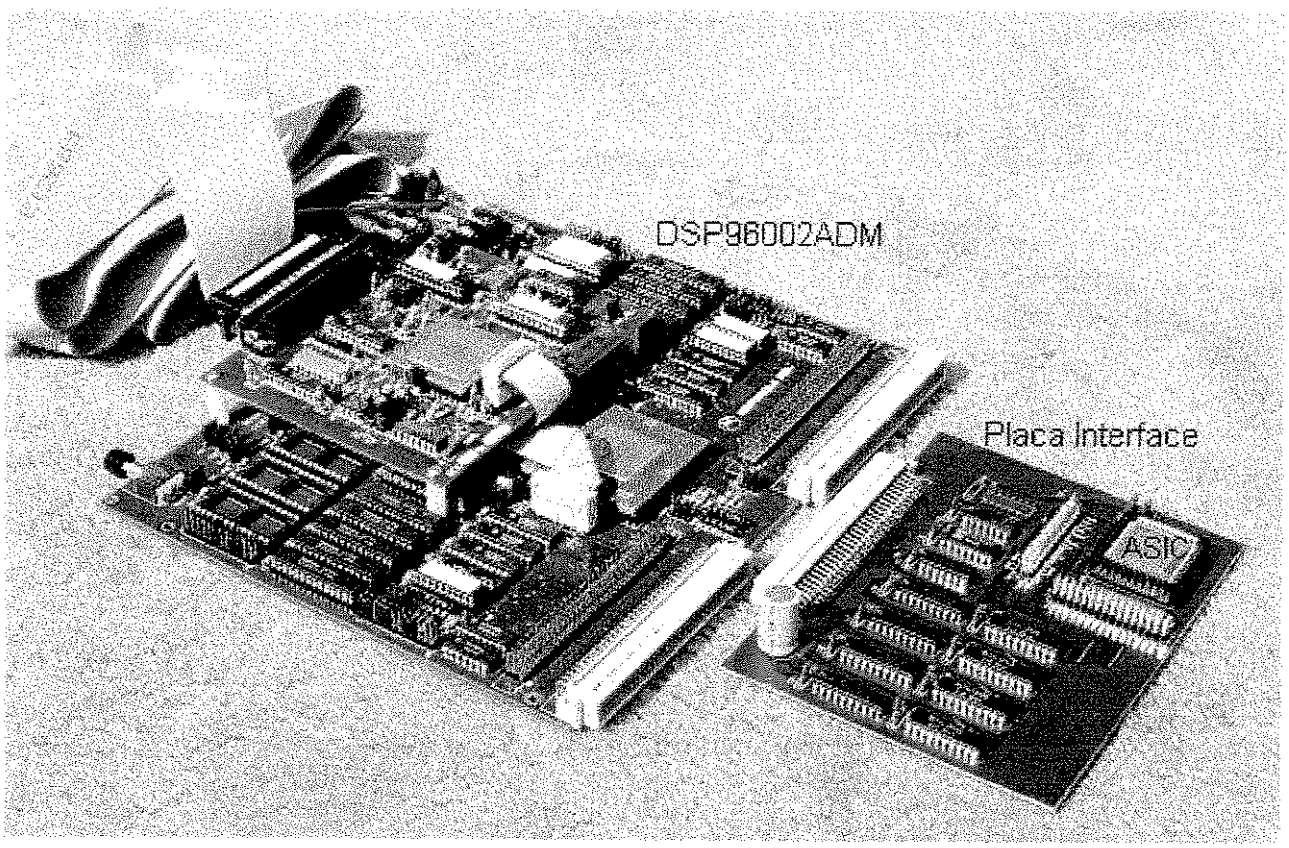


Figura 7.6 Foto do módulo de desenvolvimento DSP96002ADM e a placa de interface.

```
movep   #$4800 0000, x:TCSR0    ; Seleciona o modo de trem de pulsos
movep   #$1, x:TCR0             ; Carrega o contador para gerar 5 MHz
bset    #$18, x:IPR             ; habilita o interrupção
andi    #$CF, mr                ; retira a máscara interna no registrador de status
bset    #$1F, x:TCSR0           ; habilita o timer.
```

Figura 7.7 Segmento do programa de interfaceamento para a geração do sinal de relógio de 5MHz utilizando o *timer* 0 do módulo de desenvolvimento DSP96002ADM.

O programa de teste contém os seguintes blocos: definições de constantes para a *configuração da palavra de teste*, *configuração do chip da rede neural*, *alimentação* dos vetores-símbolo e leitura dos pesos atualizados. Na parte da configuração, realiza-se a *gravação dos parâmetros iniciais do algoritmo do modelo de Kohonen em cada uma das* células-neurônio além dos valores dos pesos iniciais. O sinal de relógio para o *chip* é gerado ao *longo do programa.*

A segunda versão da rotina opera o *chip* da rede neural numa freqüência do sinal de relógio menor e é dependente do código *Assembly*, porém é possível deter a execução em qualquer ponto da rotina de teste e, com isto, o sinal de relógio, sem perder o sincronismo, facilitando a testabilidade: por exemplo, o acesso aos registradores internos a qualquer instante.

7.5 Resultados e Discussão

As figuras 7.8 a 7.11 mostram as distribuições dos 4 símbolos da esquina superior direita da constelação QAM-16 nos casos de não distorção e na presença de três tipos de distorções não lineares: colapso das esquinas, rotação e deslizamento da grade. As figuras mostram em seqüência a evolução do ordenamento dos símbolos para diferentes instantes da operação da rede neural: 5 primeiras amostras, 10, 50, 100, 500 e 1000.

Para o caso da não presença de distorções (figura 7.8), observa-se que os pesos internos das células da rede neural mantêm a topologia da constelação sem alterações significativas.

Já no caso da distorção por colapso das esquinas (figura 7.9), as células-neurônio conseguem se ajustar à nova distribuição, a partir dos valores iniciais ideais, após as primeiras 100 amostras.

Quando a distorção é do tipo rotação da grade de 20^{o} (figura 7.10), a rede neural consegue se adaptar aos novos valores dos símbolos após as 50 primeiras amostras.

Para o caso da distorção por deslizamento da grade (figura 7.11), os pesos internos das células-neurônio conseguem de adequar à nova topologia dos símbolos, após as primeiras 50 amostras.

Dos resultados obtidos, observa-se que a estrutura de equalização DFE-SOM preserva a topologia da constelação de símbolos na presença de distorções não lineares.

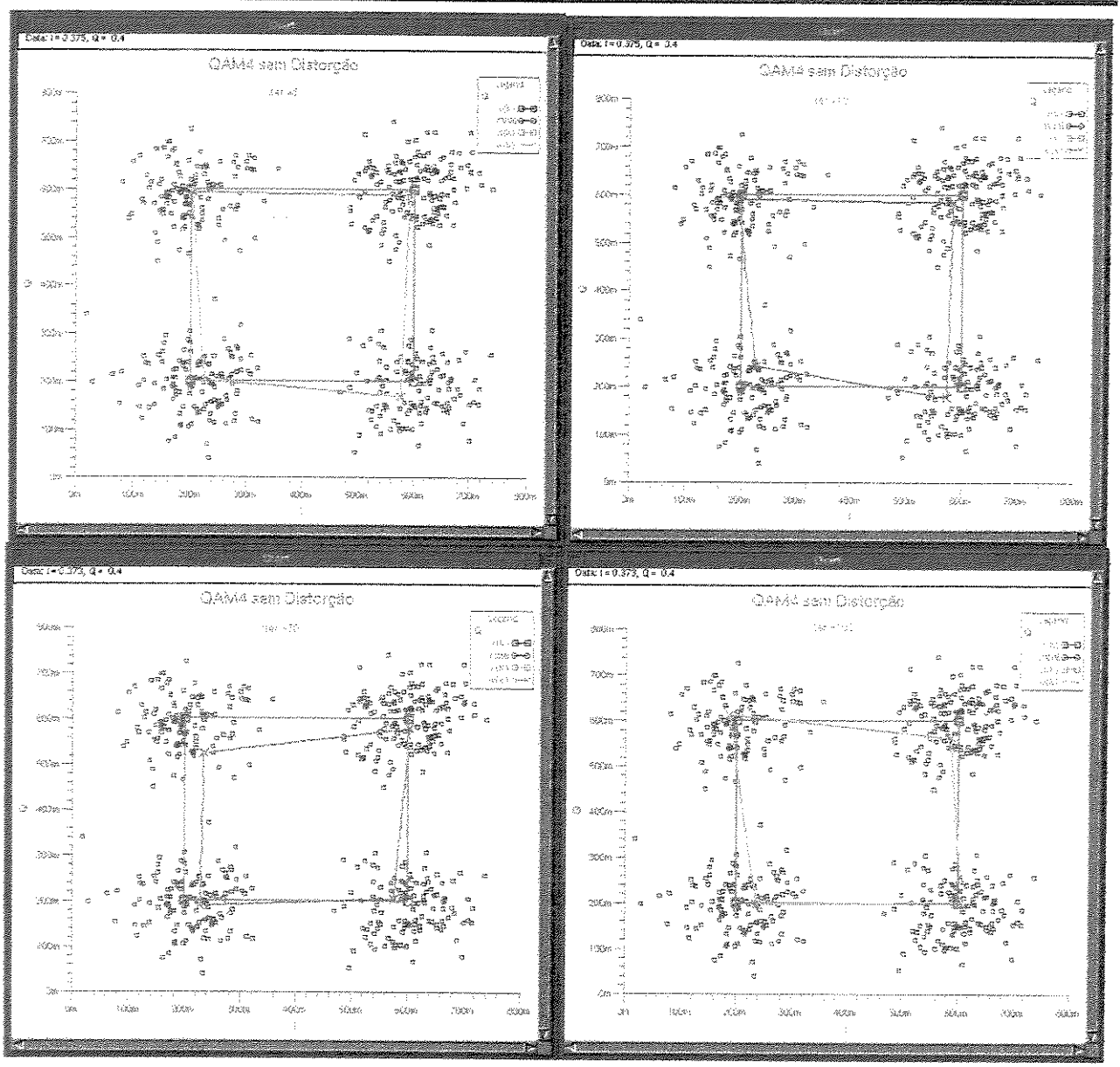


Figura 7.8 Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso de não presença de distorção. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

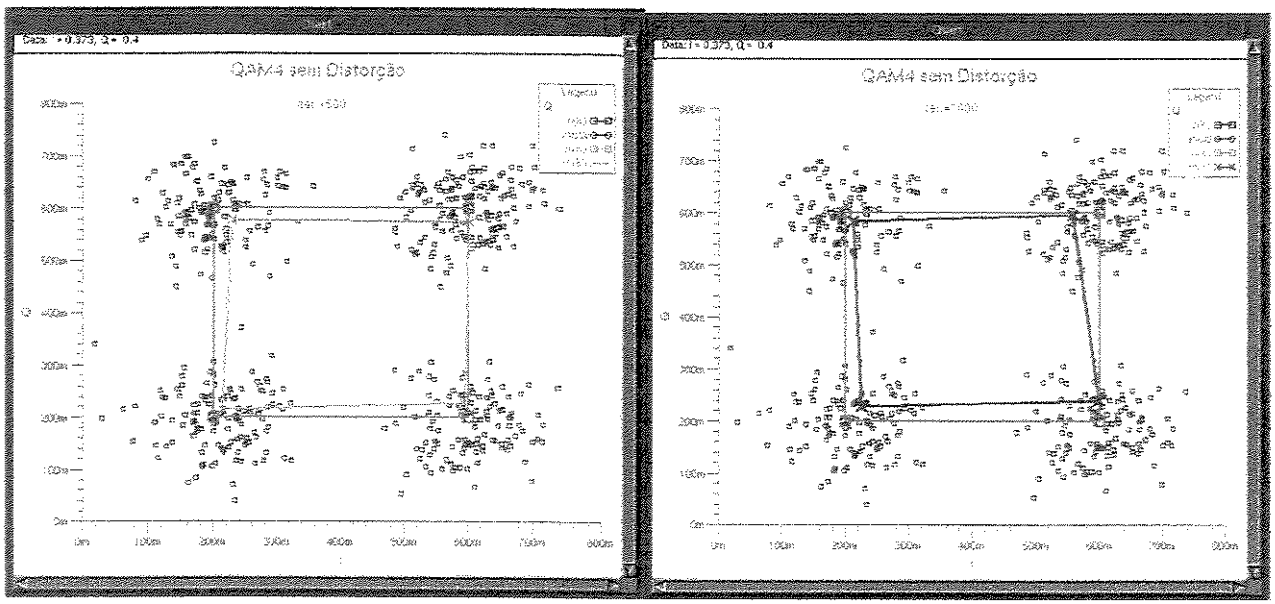


Figura 7.8 (Cont.) Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso de não presença de distorção. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

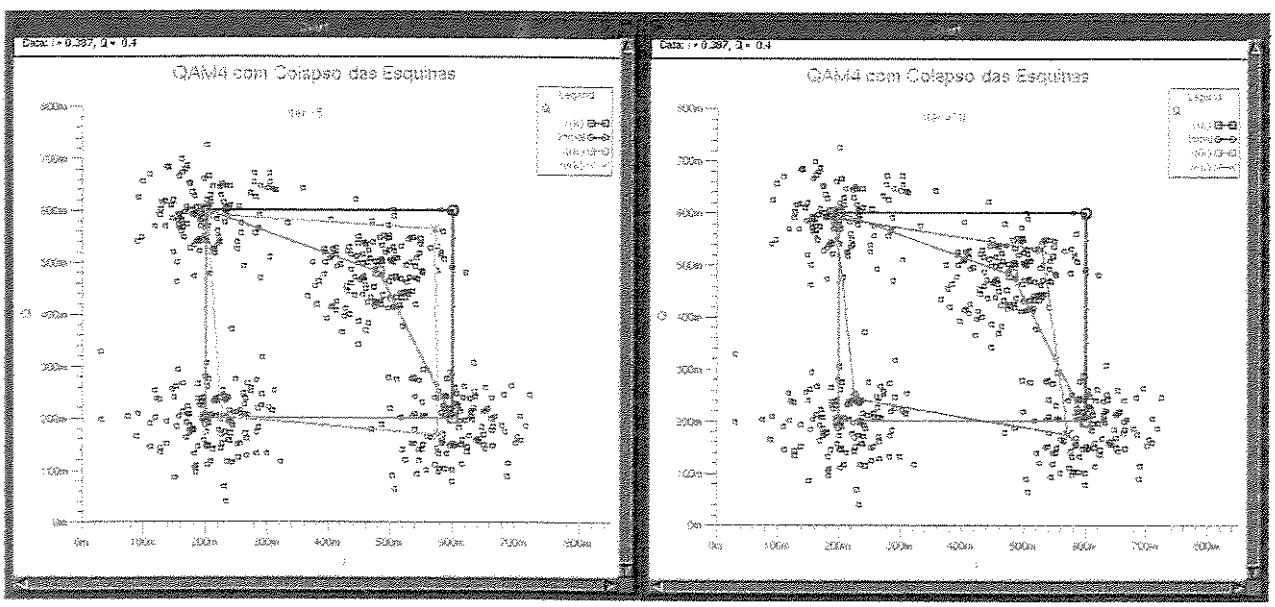


Figura 7.9 Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso da distorção por colapso das esquinas. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

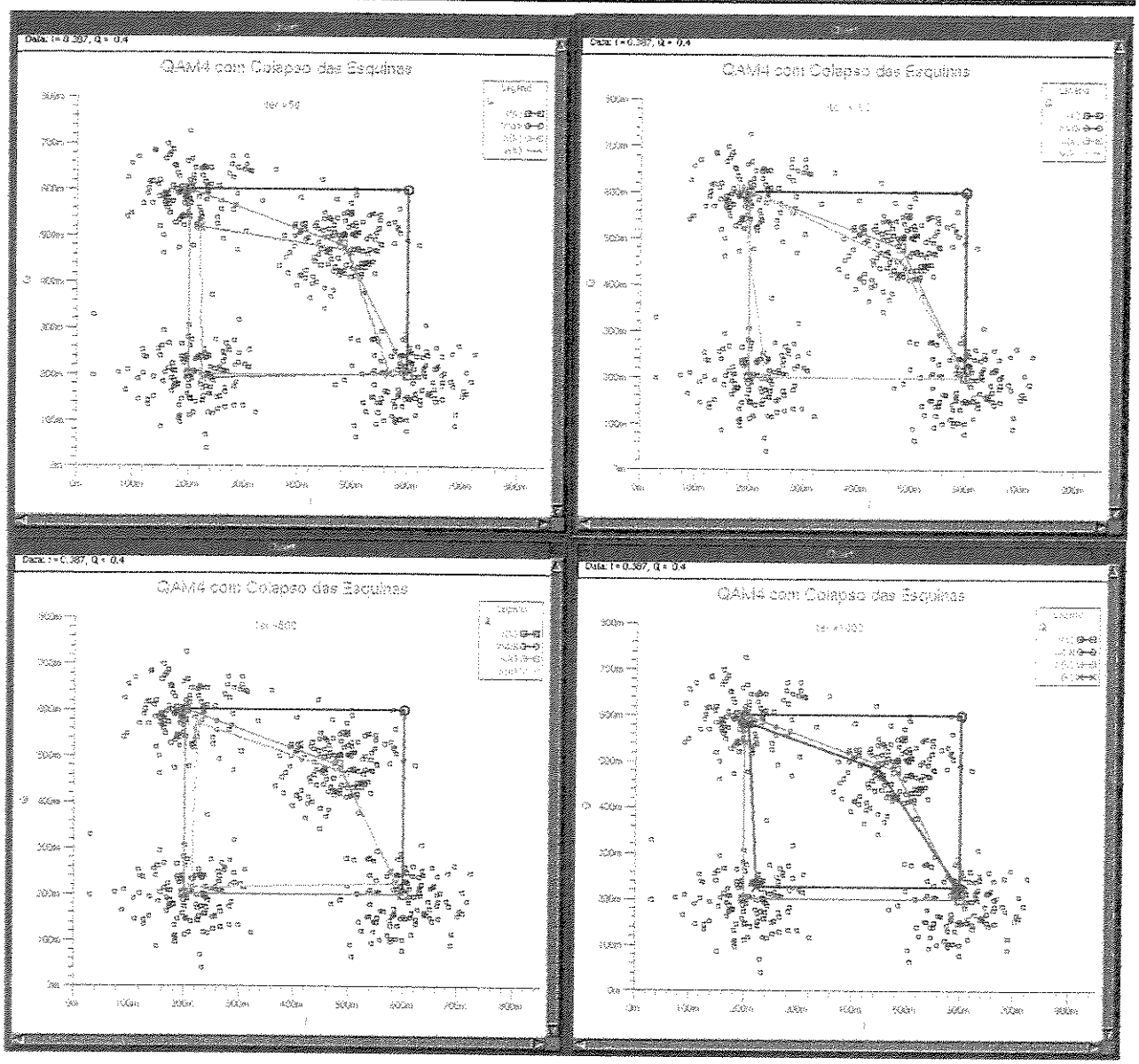


Figura 7.9 (Cont.) Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso da distorção por colapso das esquinas. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

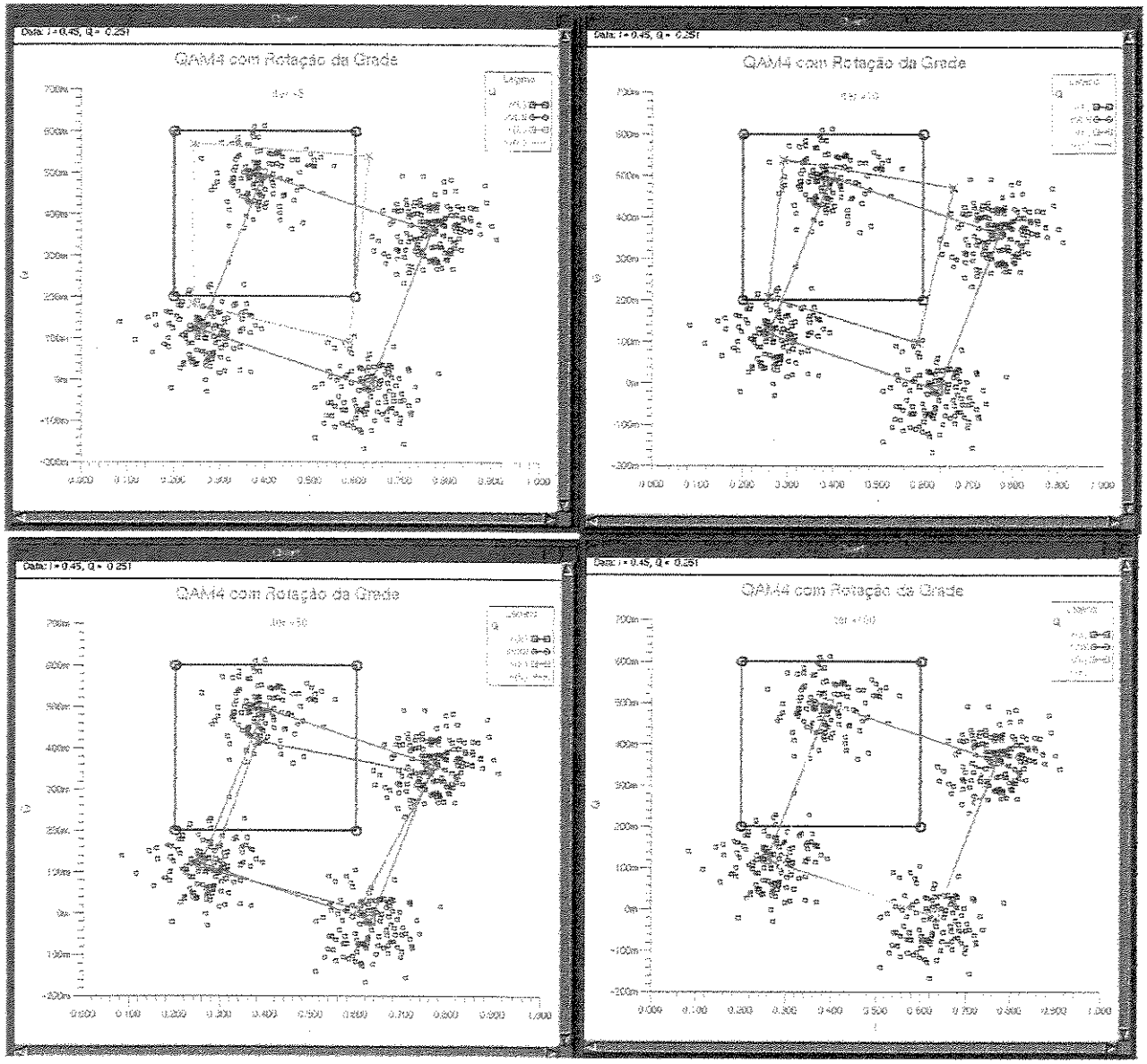


Figura 7.10 Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso da distorção por rotação da grade. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

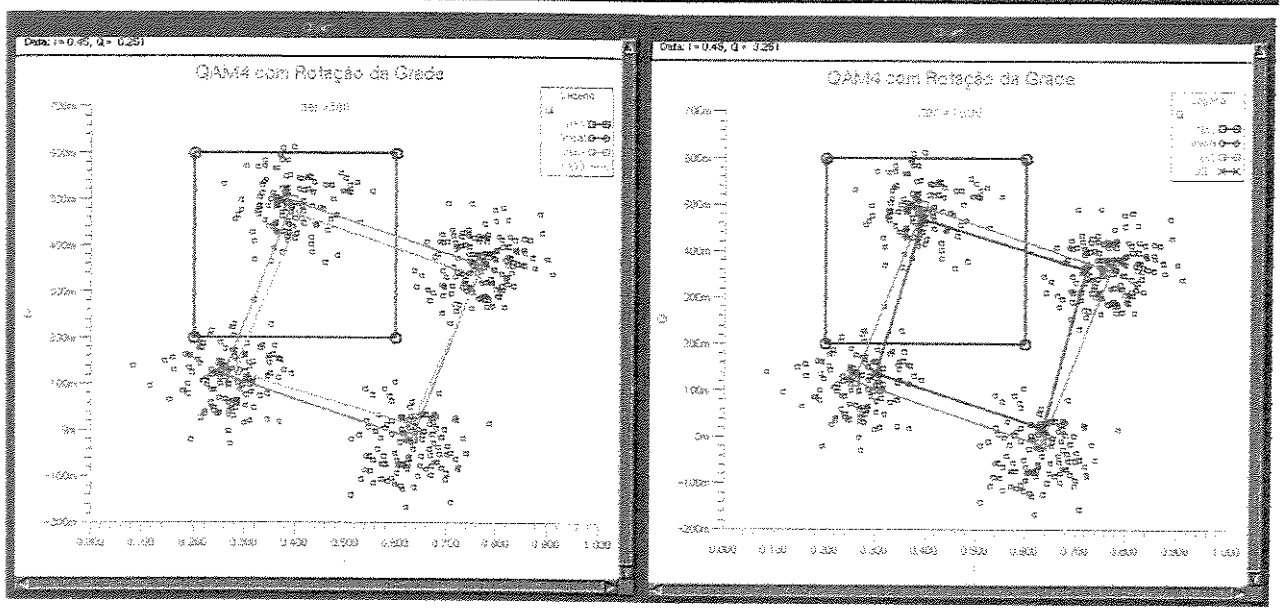


Figura 7.10 (Cont.) Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso da distorção por rotação da grade. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

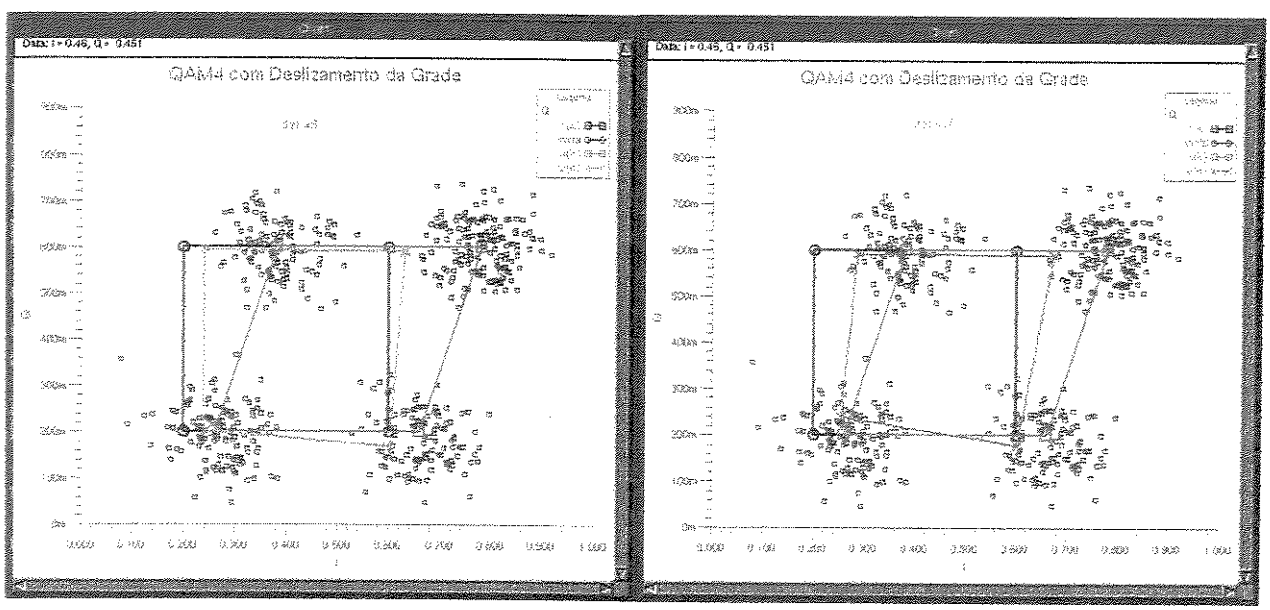


Figura 7.11 Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso da distorção por deslizamento da grade. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

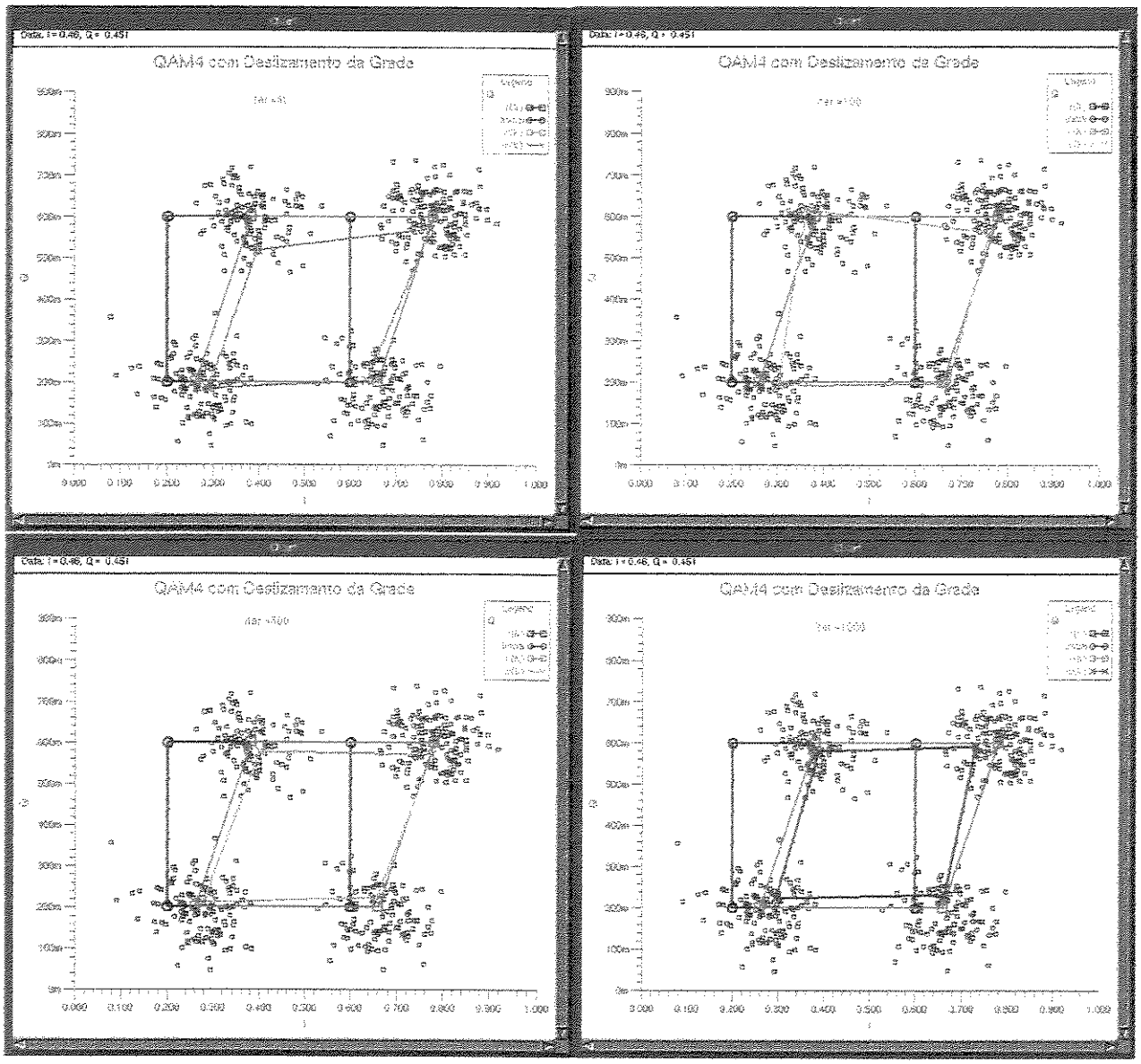


Figura 7.11 (Cont.) Distribuição do grupo dos 4 símbolos da esquina superior direita da constelação QAM-16 para o caso da distorção por deslizamento da grade. Mostra-se a seqüência para os casos de 5,10,50,100,500 e 1000 primeiros símbolos.

Conclusão

Neste trabalho foi apresentado o projeto VLSI de um ASIC implementando o mapa auto-organizado de Kohonen, utilizado no estágio de decisão de um equalizador adaptativo de sinais QAM para aplicações como comunicações digitais (rádio digital, modems), TV de alta definição (HDTV) e sistemas que utilizem modulação digital de sinais.

A metodologia *top-down* empregada mostrou-se suficiente na fase de projeto lógico do circuito, apesar de não ter sido aplicada completamente, pois houve a necessidade de realizar previamente alguns dos blocos e depois serem integrados, para se obter maior eficiência tanto em área como em desempenho do circuito. Isto resultou numa mistura de abordagens, *top-down* e *bottom-up*. A metodologia *bottom-up* empregada para a elaboração do *layout* do circuito mostrou ineficiência na integração das células-neurônio e no bloco WTA *(Winner-Takes-All)* houve desperdiço de área de silício.

No que se refere ao tempo de projeto e as ferramentas de projeto, a nossa experiência anterior, na primeira realização de circuito da rede neural de Kohonen, foi determinante para se finalizar o projeto em menor tempo, ao redor de 3 meses contando com 2 projetistas.

Na prototipação do ASIC, o tempo de retorno da *foundry* foi relativamente rápido (4 meses) se comparado com o primeiro *chip* cujo tempo de retorno foi de 7 meses. O tempo de projeto até a chegada do ASIC é crucial se considerado o contexto comercial, pois o tempo de obsolescência dos ASICs é cada vez menor.

A arquitetura das células-neurônio apresentada é diferente da primeira realização da rede de Kohonen e otimizada para aplicação específica. Esta arquitetura aproveita a estrutura de soma de produtos para realizar a atualização de pesos das células internas em apenas duas fases, diminuindo com isto o número de ciclos necessários e portanto o atraso total para a detecção do vetor-símbolo de entrada. A mesma estrutura também é utilizada no cálculo da distância.

No entanto, ainda é possível otimizar o circuito no que se refere a área de silício. Sugere-se um estudo de arquiteturas mais otimizadas de multiplicação, pois a unidade de soma de produtos ocupa uma área percentual da célula-neurônio considerável (64%). Otimizações também podem ser realizadas nas estruturas de barramentos devido às unidades *tri-state*, utilizadas no circuito, serem consumidoras de espaço e também por provocarem atrasos consideráveis tornando a resposta do circuito, mais lenta. Com estas otimizações poderá ser possível integrar um número maior de células na mesma área de silício, por exemplo 16 para a modulação QAM-16.

O fator da tecnologia de fabricação também é determinante no que se refere ao tamanho do circuito, custo, desempenho e consumo de potência. Comparando a primeira implementação da rede neural de Kohonen com a do presente trabalho, observa-se que na mesma área de silício foi possível a integração de 4 células mais o bloco WTA, sendo que na primeira implementação foi somente uma célula. A pesar de ambos os circuitos implementarem o modelo de Kohonen, os circuitos e as tecnologias de implementação são diferentes.

O teste do circuito foi realizado com o auxílio de um módulo de desenvolvimento de processamento digital de sinais da Motorola, visando a implementação completa do equalizador. Devido a limitações de operação em freqüência de tal módulo, não foi possível

realizar um teste dos limites de operação do circuito.

A linguagem de descrição de hardware VHDL foi utilizada para a descrição do circuito. Observou-se que a geração das estruturas de circuitos depende da forma de codificação e da ferramenta utilizada no processo de síntese automática. Não obstante, é necessário notar que há vantagens inerentes da utilização do VHDL como a reusabilidade e a portabilidade.

As funções de equalização são as de maior custo computacional quando utilizadas em sistemas de comunicação digital, tal como um rádio digital. Por outro lado, a inclusão da estrutura de rede neural no estágio de decisão do equalizador de sinais traz como conseqüência um aumento no custo computacional. Porém, esse incremento pode ser compensado pela inclusão de um *hardware* dedicado para a rede neural. Nessa mesma linha de raciocínio o custo computacional das funções de equalização seria diminuído se houvesse uma implementação dedicada. Não obstante, dependendo da aplicação é necessário levar em conta o custo, flexibilidade, consumo de potência, tempo de projeto para realizar um particionamento de implementações em circuitos dedicados (ASICs) e/ou processadores comerciais (DSPs).

Foi verificada a preservação topológica de sinais QAM-16 da estrutura de equalização DFE-SOM por meio das simulações de um sistema digital de comunicações e testes do ASIC implementado, para diferentes condições de distorções não lineares

Finalmente, sugere-se o estudo da aplicação da rede neural de Kohonen e do ASIC apresentado neste trabalho, para outros tipos de modulação tal como o QPSK.

Referências Bibliográficas

[Ams95] AMS - Austria Mikro Systeme International AG, *"Standard Cell Databook"*, Scholss, Premstäten, Austria, 1995.

[Ams95a] AMS - Austria Mikro Systeme, *"AMS HIT-Kit"*, versão 2.40, dezembro, 1995.

[Bel87] Bellanger M.G., *"Adaptive Digital Filters and Signal Analysis"*, Marcel Dekker, Inc., New York, 1987.

[Ber98] Bernardini A. & De Fina S., *"Optimal Decision Boundaries for M-QAM Signal Formats Using Neural Classfiers"*, IEEE Transactions On Neural Networks, Vol.9, No.2, March 1998.

[Ben87] Benedetto S. et al., *"Digital Transmission Theory"*, Prentice-Hall, Inc., New Jersey, 1987.

[Ben92] Benvenuto N., & Piazza F., "On The Complex Backpropagation Algorithm", IEEE Transactions on Signal Processing, Vol.40, No.4, pp.967-968, April 1992.

[Bez92] Bezdek J.C., Pal S.K., *"Fuzzy Models For Pattern Recognition: Methods That Search for Structures in Data"*, ed. by J.C. Bezdek, IEEE Press, 1992.

[Cam92] Camposano R., *"High-Level Synthesis"*, II Brazilian Microelectronics School, p.93-128, Gramado-RS, Brazil, March 1992.

[Car86] Carlson A. B., *"Communication Systems - An Introduction to Signals and Noise in Electrical Communications"*, McGraw-Hill Inc., Cap 14, 1986.

[Cha94] Chang P.R. Yeh B.F. & Chang C.C., *"Adaptive Packet Equalization for Indoor Radio Channel Using Multilayer Neural Networks"*, IEEE Transactions on Vehicular Technology, Vol.43, No.3, pp.773-780, Part 2, August 1994.

[Cha95] Chang P.R. & Wang B.C., *"Adaptive Decision-Feedback Equalization for Digital Satellite Channels Using Multilayer Neural Netwroks"*, IEEE Journal on Selected Areas in Communications, Vol.13, No.2, pp.316-324, February 1995.

[Che90] Chen S., Gibson G.J. & Cowan C.F.N., *"Adaptive Channel Equalization Using a Polynomial-Perceptron Strutucture"*, Procedings of the IEE, Part I, Vol. 137, pp.257-264, October 1990.

[Che93a] Chen S. & Mulgrew B., *"A Clustring Technique for Digital Communications Channel Equalization Using Radial Basis Function Networks"*, IEEE Transactions on Neural Networks, Vol.4, No.4, pp.570-579, July 1993.

[Che93b] Chen S., Mulgrew B. & McLaughlin S., *"Adaptive Bayesian Equalizer with Decision-Feedback"*, IEEE Transactions on Signal Processing, Vol.41, No.9, pp.2918-2927, September 1993.

[Cho93] Choi J.G., Bang S.H. and Sheu B.J., *"A programmable Analog VLSI Neural-Network Processor for Communications Receivers"*, IEEE Transactions on Neural Networks, Vol.4, No.3, pp.484-495, May 1993.

[Cow85] Cowan C.F.N. & Grant P.M., *"Adaptive Filters"*, Prentice-Hall Series in Signal Processing, New Jersey, 1985.

[Far60] Farley B., *"Self-Organizing Models for Learned Perception"*, in Self-Organization Systems, ed. By M. Yovits & S. Cameron, pp. 7.-27, Pergamon Press, NY, 1960.

[Gib90] Gibson G.J. & Cowan C.F.N., *"On the Decision Regions of Multilayer Perceptrons"*, Proceedings of the IEEE, Vol.78, No.10, October 1990.

[Gib91] Gibson G.J, Siu S, & Cowan C.F.N, *"The Application Of Nonlinear Structures To The Reconstruction Of Binary Signals"*, IEEE Transactions On Signal Processing, Vol. 39, No.8, pp.1877-1884, August 1991.

[Gil94] Giles C.L., Kuhn G.M. & Williams R.J., *"Dynamic Recurrent Neural Networks: Theory and Applications"*, IEEE Transactions on Neural Networks, Vol. 5, No.2, pp. 153-156, March 1994.

[Hac97] Kacioglu K., *"An Improved Recurrent Neural Network for M-PAM Symbol Detection"*, IEEE Transactions on Neural Networks, Vol.8, No.3, pp.779-783, May 1997.

[Hay84] Haykin S.S., *"Introduction to Adaptive Filters"*, MacMillan Publishing Company, cap. 1, 1988.

[Hay88] Haykin S.S., *"Digital Communications"*, John Wiley & Sons, Inc., Cap. 7, 1988.

[Hwa79] Hwang K., *"Computer Arithmetic - Principles, Architecture & Design"*, John Wiley & Sons, 1979.

[Jar95a] Jara P.M.A., Fornari J. & Damiani F., "Reconhecimento de timbres musicais através da rede neural auto-organizável de Kohonen", Proceedings da II SBC&M, 2nd Brazilian Symposium on Computer Music, pp.107-113, Canela, RS, Brasil, 1995.

[Jar95b] Jara P.M.A, Machaca L.W. Damiani F., *"Design of a 4x4 Kohonen Neural Net: VHDL Description"*, (ICCDCS-083), IEEE International Caracas Conference on Devices, Circuits and Systems, Caracas, Venezuela, Dezembro, 1995.

[Jar97] Jara P.M.A., "Projeto e Implementação VLSI de uma Rede Neural Auto-Organizável usando Síntese Automática de Alto Nível", Tese de Doutorado, DSIF/FEEC/UNICAMP, Campinas, SP, Brasil, Agosto, 1997.

[Jar98] Jara P.M.A., Machaca L.W. & Damiani F., *"Biologically-Inspired Digital Circuit for a Self-Organising Neural Network"*, Second IEEE International Caracas Conference on Devices, Circuits and Systems. Islas Margaritas, Venezuela, Março, 1998.

[Jon95] Johnson C.R.Jr., *"On the Interaction of Adaptive Filtering, Identification, and Control"*, IEEE Signal Processing Magazine, March 1995.

[Kec94] Kechriotis G., Zervas E. & Manolakos E.S., *"Using Recurrent Neural Networks for Adaptive Communication Channel Equalization"*, IEEE Transactions on Neural Networks, Vol.5, No.2, pp.267-278, March 1994.

[Knu71] Knuth D.E., *"The Art of Computer Programming"*, Vol.2, Seminumerical Algorithms, Addison-Wesley Pub. Co., 2nd printing, 1971.

[Koh78] Kohavi Z., *"Switching and Finite Automata Theory"* 2nd Edition, McGraw-Hill Inc., 1978.

[Koh89] Kohonen T., *"Self-Organization and Associative Memory"*, Springer Series in Information Sciences, Springer-Verlag, 3rd. Edition, 1989.

[Koh90a] Kohonen T., et al, *"Combining Linear Equalization and Self-Organizing Adaptation in Dynamic Discrete-Signal Detection"*, Proc. Of the International Joint Conference on Neural Networks, IJCNN-90, pp.223-228, San Diego, CA. Jun.18-21, 1990.

[Koh90b] Kohonen T., et al, *"An Adaptive Discrete-Signal Detector Based on Self-Organizing Maps"*, Proc. Of the International Joint Conference on Neural Networks, IJCNN-90, pp.223-228, San Diego, CA. Jun.18-21, 1990.

[Koh90c] Kohonen T., *"The Self-Organizing Map"*, Proceedings of the IEEE, Vol.78, No.9, September 1990.

[Koh91] Kohonen T. et al., *"Perfomance Evaluation of Self-Organizing Map Based Neural Equalizer in Dynamic Discrete-Signal Detection"*, Proceedings of the International Conference on Artificial Neural Networks, P.II 1677-1680, Helsinki, June 1991.

[Koh92] Kohonen T., et al., *"Start-Up Behaviour of a Neural Network Assisted Decision Feedback Equaliser in a Two-Path Channel"*, Proceedings of the International Conference on Communications, p.1523-1527, Chicago, June 14-18, 1992.

[Koh96] Kohonen T., et al. *"Engineering Applications of the Self-Organizing Map"*, Proceedings of the IEEE, Vol.84, No.10, October 1996.

[Lee94] Lee E.A. & Messerschmitt D.G., *"Digital Communication"* 2nd Edition, Kluwer Academic Publishers, USA, 1994.

[Lin97] Lin C.T. & Juang C.F., *"An Adaptive Neural Fuzzy Filter and its Applications"*, IEEE Transactions on Systems Man and Cybernetics, Part B: Cybernetics, Vol.27, No.4, pp.635-656, August 1997.

[Lip87] Lippman R.P., *"An Introduction to Computing with Neural Nets"*, IEEE ASSP Magazine, April, 87.

[Mac96] Machaca L.W., Jara P.M.A. & Damiani F., *"Design of a Kohonen Neural Net Cell, Neuron: VHDL Description"*, II Workshop Iberchip,EPUSP, São Paulo, Brasil, Fevereiro, 1996.

[Men95a] Mentor Graphics Corporation, *"DSP Station User's Guide™"*, Software Version 8.5_4, 1993-1995.

[Men95b] Mentor Graphics Corporation, *"DSP Station Telecom Simulation Library Reference Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95c] Mentor Graphics Corporation, *"DSP Architect DFL User's and Reference Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95d] Mentor Graphics Corporation, *"DSP Architect DDSIM User's Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95e] Mentor Graphics Corporation, *"DSP Station Motorola DSP56000 Code Generation User's Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95f] Mentor Graphics Corporation, *"Design Architect User's Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95g] Mentor Graphics Corporation, *"Synthesizing with Autologic II™"*, Software Version 8.5_4, 1993-1995.

[Men95h] Mentor Graphics Corporation, *"QuickSim II User's Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95i] Mentor Graphics Corporation, *"IC Station User's Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95j] Mentor Graphics Corporation, *"ICverify User's Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95k] Mentor Graphics Corporation, *"Autologic Blocks Manual™"*, Software Version 8.5_4, 1993-1995.

[Men95l] Mentor Graphics Corporation, *"AMPLE User's Manual™"*, Software Version 8.5_4, 1993-1995.

[Mey93] Meyer M. & Pfeiffer G., *"Multilayer Percetron Based Decision-Feedback Equalizer for Channels with Intersymbol Interference"*, IEE Proceedings, Part I, Vol.140, No.6 pp.420-424, December 1993.

[Mot_a] Motorola Incorporated, *"DSP96002ADM User's Manual"*, Nome do documento DSP96002ADMUM/AD.

[Mot89] Motorola Incorporated, *"DSP96002 User's Manual"*, 1989,

[Mul96] Mulgrew B., *"Applying Radial Basis Functions"*, IEEE Signal Processing Magazine, pp. 50-65, March 1996.

[Nag92] Nagasamy V., Berry N. & Dangelo C., *"Specification, Planning and Synthesis"*, IEEE Design & Test of Computers, Vol.9, Iss.2, p.58-68, June 1992.

[Pre92] Press W.H., et al, *"Numerical Recipes in C – The Art of Scientific Computing", Cambridge University Press,* Chap. 7, 2nd. Edition, 1992.

[Pro95] Proakis J.G., *"Digital Communications"*, 3rd Edition, McGraw-Hill, Inc., 1995.

[Pul93] Pulford G.W., Kennedy R.A. & Anderson B.D.O., *"An Upper Bound on the Perfomance of a Novel FeedFordward Perceptron Equalizer"*, IEEE Transactions on Information Theory, Vol.36, No.6, pp.1923-1929, November 1993.

[Qur85] Qureshi S.U.H., *"Adaptive Equalization"*, Proceedings of the IEEE, Vol.73, No.9, September 1985.

[Rai91] Raivio K., Simula O. & Henriksson J., *"Improving Decision Feedback Equaliser Perfomance Using Neural Network"*, Electronics Letters, Vol.27, No.23, pp.2151-2153, November 1991.

[Ros60] Rosenblatt F., *"Perceptual Generalization over Transformation Groups"*, Self Organizing Systems - Proc. of na Interdisciplinary Conf., 5-6 May 1959, pp.63-100, Pergamon Press, NY, 1960.

[Rum95] Rumelhart D.E. & McClelland J.L., *"Parallel Distribuited Processing: Explorations in the Microstrure of Cognition"*, Vol.1: Foundations. Cambridge Mass. MIT, 1995.

[Sch92] Schalkoff, R., *"Pattern Recognition: Statistical Structural and Neural Approaches"*, John Wiley & Sons, Inc., 1992.

[Siu90] Siu S. et al, *"Decision Feedback equalisation using neural network structures and perfomance comparison with the standard architecture"*, IEE Proceedings Part I, Vol.137, No.4, Aug., 1990.

[Siu93] Siu S. & Cowan C.F.N., *"Perfomance Analysis of the L(P) Norm Back-Propagation Algorithm for Adaptive Equalization"*, IEE Proceedings Part F, Vol.140, No.1, pp.43-47, February 1993.

[Thi94] Thiran P., et al., "Quantization Effects in Digitally Behaving Circuit Implementation of Kohonen Networks", IEEE Transactions on Neural Networks, Vol.5, No.3, pp.450-458,

May 1994.

[Wid88] Widrow B., et al., *"Layered Neural Nets for Pattern Recognition"*, IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP), Vol.36, pp.1109-1118, July 1988.

[Wid90] Widrow B., Lehr, M., *"30 Years of Adaptive Neural Networks: Perceptron, Madaline and BackPropagation"*, Proceedings of the IEEE, Vol.78(9), pp.1415-1442,1990.

[Wil76] Willshaw D.J. & C. von der Malsburg, *"How patterned neural connections can be set up by self-organization"*, Proc. of the Royal Society of London, B. Biological Sciences, Vol. 194, No.1117, Nov., 1976.

[Xia94] Xiang Z.J., Bi G.G. & Tho L.N, *"Polynomial Perceptrons and their Applications to Fading Channel, Equalization and Co-channel Interference Suppression"*, IEEE Transactions On Signal Processing, Vol.43, No.9, pp.2470-2480, Sept., 1994.

Anexos

Anexo 1: Listagem do programa em C para a simulação do gerador de sinais

```
/*-------------------------------------------------------------*/
/* Geração aleatória de símbolos para modulação QAM    */
/* Autor : Wilfredo Machaca Luque DSIF/FEEC/UNICAMP */
/* Primeira versão : 25-05-98                          */
/*-------------------------------------------------------------*/
/* Para compilar (Solaris, SunOS) : gcc symsrc.c -o symsrc -lm*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>  /* requerido pela função exit no Borland C++ */
#include "param.h"

main(int argc, char *argv[]) {
 int i, j, s, distor_option;
 int nsamples = 1000;        /* número de simbolos a gerar */
 float x,y,angle,angle1,angle0,radio, vsymb[NSF][NSC][Dimen];
 float ivsymb[NSF][NSC][Dimen], niv[NSF];
 FILE *fsymb, *fref;

 if (argc<2) {
  printf("Syntax : symsrc <d> <#samples> \n");
  printf("\td=0 : No Distortion \n");
  printf("\td=1 : Corner Collapse \n");
  printf("\td=2 : Escape of one quantization level of the grid \n");
  printf("\td=3 : Zoom In \n");
  printf("\td=4 : Zoom Out \n");
  printf("\td=5 : Rotate 10 \n");
  printf("\td=6 : Rotate 20 \n");
  printf("\td=7 : Rotate 30 \n");
  printf("\td=8 : Grid Collapse \n");
  exit(0);
 }
 for (i=0; i<NSF; ++i) {
  niv[i]=(float)((NSF-1)-2*i);
  for (j=0; j<NSC; ++j) {
   ivsymb[i][j][0]=(2*j-(NSC-1))/2.0*Ac;
   ivsymb[i][j][1]=((NSF-1)-2*i)/2.0*Ac; }
 }
 distor_option = atoi(argv[1]);
 if (argc==3) nsamples = atoi(argv[2]);
 printf("\tThe symsrc.io will have %d symbol samples.\n",nsamples);

 if (distor_option==0) {
  printf("\td=0 : No Distorting ...\n");
  for (i=0; i<NSF; ++i)
   for (j=0; j<NSC; ++j)
    for (s=0; s<Dimen; ++s)
     vsymb[i][j][s]=ivsymb[i][j][s];
 }
 else if (distor_option==1) {
  printf("\td=1 : Corner Collapsing ...\n");
  for (i=0; i<NSF; ++i)
   for (j=0; j<NSC; ++j)
    for (s=0; s<Dimen; ++s) {
     vsymb[i][j][s]=ivsymb[i][j][s];
     if ( ((i==0)||(i==NSF-1)) && ((j==0)||(j==NSC-1)))
      vsymb[i][j][s]=ivsymb[i][j][s]*CCF;
    }
 }
 else if (distor_option==2) {
  printf("\td=2 : Escaping of one quantization level of the grid ...\n");
  for (i=0; i<NSF; ++i)
   for (j=0; j<NSC; ++j)
    for (s=0; s<Dimen; ++s)
     vsymb[i][j][s]=ivsymb[i][j][s];
  vsymb[0][NSC-1][0]=ivsymb[0][NSC-1][0]*(1.0-HQLG);
  vsymb[0][NSC-1][1]=ivsymb[0][NSC-1][1]*(1.0-VQLG);
 }
 else if (distor_option==3) {
  printf("\td=3 : Zooming In ...\n");
  for (i=0; i<NSF; ++i)
   for (j=0; j<NSC; ++j)
    for (s=0; s<Dimen; ++s)
     vsymb[i][j][s]=ivsymb[i][j][s]*ZIF;
 }
 else if (distor_option==4) {
  printf("\td=4 : Zooming Out ...\n");
  for (i=0; i<NSF; ++i)
   for (j=0; j<NSC; ++j)
    for (s=0; s<Dimen; ++s)
     vsymb[i][j][s]=ivsymb[i][j][s]*ZOF;
 }
 else if ((distor_option==5) || (distor_option==6) || (distor_option==7)) {
  if (distor_option==5) angle0=ROT10;
  if (distor_option==6) angle0=ROT20;
  if (distor_option==7) angle0=ROT30;
  printf("\td=%d : Rotating %2.2f degrees ...\n",distor_option,angle0*180/PI);
  for (i=0; i<NSF; ++i)
   for (j=0; j<NSC; ++j) {
    x=ivsymb[i][j][0]; y=ivsymb[i][j][1]; radio=sqrt(x*x+y*y);
    if (x<0.0) angle1=PI; else angle1=0.0;
    angle=angle1+atan(y/x)-angle0; printf("angle=%f\n",angle*180/PI);
    vsymb[i][j][0]=radio*cos(angle);
    vsymb[i][j][1]=radio*sin(angle);
   }
 }
 else if (distor_option==8) {
  printf("\td=8 : Grid Collapsing ...\n");
  for (i=0; i<NSF; ++i)
   for (j=0; j<NSC; ++j) {
    x=HGCF*(NSF-1)*Ac/2.0/(NSF-fabs(niv[i]));
    y=VGCF*(NSC-1)*Ac/2.0/(NSC-fabs(niv[j]));
    if (ivsymb[i][j][1]>0.0) vsymb[i][j][0]=ivsymb[i][j][0]+x;
    else vsymb[i][j][0]=ivsymb[i][j][0]-x;
    if (ivsymb[i][j][0]>0.0) vsymb[i][j][1]=ivsymb[i][j][1]+y;
    else vsymb[i][j][1]=ivsymb[i][j][1]-y;
   }
 }
 else printf("%d, Not Implemented Case !!\n",distor_option);
 fref = fopen(Fnref,"w");
 for (i=0; i<NSF; ++i)
  for (j=0; j<NSC; ++j) fprintf(fref,"%f %f\n",vsymb[i][j][0],vsymb[i][j][1]);

 srand(SEED);
```

```
 srand(SEED);
 fsymb = fopen(Fnsrc,"w");
 for (s=0; s<nsamples; ++s) {
  /* i=random()%NSC; j=random()%NSF; */ i=random(NSC); j=random(NSF);
  /*printf("\tsample # %d, i=%d, j=%d\n",s,i,j);*/
  fprintf(fsymb,"%f %f\n",vsymb[i][j][0],vsymb[i][j][1]);
 }
 fclose(fsymb); fclose(fref);
 printf("\tsymsrc.io was succesfully created !\n");
} /*main*/
```

Conteúdo do arquivo "PARAM.H"

```
/*-------------------------------------------------------------------*/
/* Definições para os programas symsrc, awgn, dfe e dfe_knn          */
/* Autor : Wilfredo Machaca Luque DSIF/FEEC/UNICAMP (27-12-98)  */
/*-------------------------------------------------------------------*/

#define Dimen 2   /* dimensão vetor peso/entrada */
#define PI 3.1416  /* Constante PI */
#define CCF 0.8   /* Fator de Corner Collapse */
#define ZOF 1.2   /* Fator de Zoom Out */
#define ZIF 0.8   /* Fator de Zoom In */
#define HGCF 0.3  /* Porcentagem para o Grid Collapse Factor horizontal */
#define VGCF 0.0  /* Porcentagem para o Grid Collapse Factor vertical */
#define HQLG 0.2  /* Porcent. para o Escape of Quant. level horizontal*/
#define VQLG 0.2  /* Porcentagem para o Escape of Quant. level vertical */
#define HCCF 0.2  /* Porcent. para o Corner Collapse Factor horizontal */
#define VCCF 0.2  /* Porcentagem para o Corner Collapse Factor vertical */
#define ROT10 PI/18 /* Rotação de 10 graus */
#define ROT20 PI/9  /* Rotação de 20 graus */
#define ROT30 PI/12 /* Rotação de 15 graus */

#define NSF 4     /* número de filas de simbolos na constelação */
#define NSC 4     /* número de colunas de simbolos na constelação */
#define Ac 0.4    /* distância entre dois niveis (de i ou de q) consecutivos */
#define SEED 69069 /* Semente para a geração de números aleatórios */
#define DELTA 0.001 /* Ganho de laço (Loop Gain) para o algoritmo LMS */

#define Fawgn "awgn16.io"   /*seq. se símb. saida canal two-path AWGN*/
#define Fnrk  "dfe16rk.io"  /*seq. de simbolos na saida do DFE*/
#define Fnek  "dfe16ek.io"  /*seq. de simbolos Erro=Desejado-Yn */
#define FWGINI "wg16.ini"   /* Pesos iniciais da rede neural*/
#define FPARAM "param16.cfg" /* Parâm. para o algoritmo de aprendizado */
#define Fnsrc  "src16.io"   /*seq. de simbolos enviados */
#define Fcnvber "berc16.io" /* Arq. saída  Taxa de Erro de Bits (BER) */
#define Fknnber "berk16.io" /* Arq. saida  Taxa de Erro de Bits (BER) */
#define Fnref  "ref16.io"   /* Arq. saída para a referencia do gnuplot */
#define FPLOT  "wg16.io"    /* Buffer for GNUPLOT */
```

Anexo 2: Listagem do programa em C para a simulação do canal AWGN

```
/*-------------------------------------------------------------------*/
/* Simula um canal AWGN (Additive White Gaussian Noise) Two-Path */
/* Autor : Wilfredo Machaca Luque DSIF/FEEC/UNICAMP  (28-05-98) */
/*-------------------------------------------------------------------*/
/* Para compilar (Solaris, SunOS) : gcc awgn16.c -o awgn -lm */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>  /* requerido pela função exit no Borland C++ */
#include "param.h"

#define NTAPS 2   /* Número de TAPS */
#define Fout "awgn16.io"   /*seq. de simbolos na saida do canal*/

float noise();
float gaussnoise();
void multcomplx();

main(int argc, char *argv[]) {
 int f;
 float ndisp, xn[NTAPS][Dimen], rs[Dimen], sum[Dimen];
 float cfwd[NTAPS][Dimen]={1.0,0.0,0.0,0.1};  // h(n) = 1 + 0.1 e^(j*PI/2)
 FILE *fxn, *fyn;

 if (argc!=2) { printf("Syntax : awgn <SNR>\n"); exit(0); }
 ndisp = pow(10.0,atoi(argv[1])/10.0); /* SNR=10*Log(Pot.Sinal/Pot.Ruido) */

 for (f=0; f<NTAPS; ++f) { xn[f][0]=0.0; xn[f][1]=0.0; }
 fxn = fopen(Fnsrc,"r");
 fyn = fopen(Fawgn,"w");
 while (fscanf(fxn,"%f",&xn[0][0])!=EOF) {
  fscanf(fxn,"%f",&xn[0][1]);
  sum[0]=0.0; sum[1]=0.0;
  /* Produto Xn * Cfwd */
  for (f=0; f<NTAPS; ++f) {
   /* printf("Multiplying ... f=%d\n",f); */
   multcomplx(&xn[f],&cfwd[f],&rs);
   sum[0]+=rs[0]; sum[1]+=rs[1];
  }
  /* Rotação de Xn */
  for (f=NTAPS-1; f>0; --f) {
   /* printf("Rotating Xn... f=%d\n",f); */
   xn[f][0]=xn[f-1][0]; xn[f][1]=xn[f-1][1];
  }
  /* Adição do ruido */
  sum[0]+=gaussnoise()*(NSF-1)*Ac/2.0/ndisp;
  sum[1]+=gaussnoise()*(NSF-1)*Ac/2.0/ndisp;
  fprintf(fyn,"%f %f\n",sum[0],sum[1]);
 }
 fclose(fxn); fclose(fyn);
 printf("10^(SNR/10)=%f \n\tawgn.io was succesfully created !\n",ndisp);

} /*main*/

void multcomplx(c1,c2,result)
 float *c1, *c2, *result;
{
 *(result+0)=*(c1+0) * *(c2+0) - *(c1+1) * *(c2+1);
 *(result+1)=*(c1+0) * *(c2+1) + *(c1+1) * *(c2+0);
}

long int seed0, seed1=0x100;
```

```
} /* gera um número aleatório entre (0..1) */

float gaussnoise() {
 static int iset=0; static float gset;
 float fac,rsq,v1,v2;

 if (iset==0) {
  do {
   v1 = (float)(rand()&RAND_MAX)/RAND_MAX;
   v2 = (float)(rand()&RAND_MAX)/RAND_MAX;
   v1=2.0*v1-1.0; v2=2.0*v2-1.0;
   rsq=v1*v1+v2*v2;
  } while (rsq>=1.0 || rsq==0); /* printf("v1=%f  v2=%f\n",v1,v2); */
  fac = sqrt(-2.0*log(rsq)/rsq);
  gset=v1*fac;
  iset=1;
  return v2*fac;
 }
 else {
  iset=0; return gset;
 }
} /* devolve número aleatório (0..1) com distribuição gaussiana */
```

Anexo 3: Listagem do programa em C para a simulação do equalizador DFE convencional

```
/*-------------------------------------------------------------------------*/
/* Simula um equalizador DFE (Decision Feedback Equalizer) adaptativo  */
/* Algoritmo de adaptação dos coeficientes : LMS (Least-Mean-Square)    */
/* Autor : Wilfredo Machaca Luque DSIF/FEEC/UNICAMP (02-06-98)          */
/*-------------------------------------------------------------------------*/
/* Para compilar (Solaris, SunOS) : gcc dfe.c -o dfe -lm                */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>  /* requerido pela função exit no Borland C++ */
#include "param.h"

#define FWDTAPS 2  /* 5 Número de TAPS no caminho direto */
#define BCKTAPS 1  /* 4 Número de TAPS no caminho da realimentação */

float niv[NSF];
int distor;
int region();
void multcomplx();

main(int argc, char *argv[]) {
 int ct=0, ctber=0, ctplot=1900, f, g; float lg;
 float uk[FWDTAPS][Dimen], bk[BCKTAPS+1][Dimen];
 float src[Dimen], tmp[Dimen], ek[Dimen], rk[Dimen];
 float cfwd[FWDTAPS][Dimen]={1.0,0.0, 0.0,0.0};//, 0.0,0.1, 0.0,-0.1, 0.0,0.1};
 float cbck[BCKTAPS][Dimen]={0.0,0.0};//, 0.0,-0.1, 0.0,0.1, 0.0,-0.1};
 FILE *fuk, *frk, *fek, *fsrc, *fber;

 if (argc<2) {
  printf("Syntax : dfe [distor# [loop_gain [#plots]]]\n"); exit(0); }
 else {
  distor=atoi(argv[1]);
  if (argc>=3) {
   lg=atof(argv[2]);
   if (argc>=4) ctplot=atoi(argv[3]);
  }
  else lg=DELTA;
 }
 printf("DFE - Decision Feedback Equalizer.\n");
 printf("\tUsing : Loop Gain=%f;  Distortion#=%d ...\n\n",lg,distor);

 for (g=0; g<NSF; ++g) niv[g]=(float)((NSF-1)-2*g);
 /* Inicialização dos vetores Xk e Bk */
 for (f=0; f<FWDTAPS; ++f) for (g=0; g<Dimen; ++g) uk[f][g]=0.0;
 for (f=0; f<BCKTAPS+1; ++f) for (g=0; g<Dimen; ++g) bk[f][g]=0.0;

 fuk = fopen(Fawgn,"r"); frk = fopen(Fnrk,"w"); fek = fopen(Fnek,"w");
 fsrc = fopen(Fnsrc,"r"); fber = fopen(Fcnvber,"w");

 srand(SEED);
 while (fscanf(fuk,"%f",&uk[0][0])!=EOF) {
  fscanf(fuk,"%f",&uk[0][1]); ++ct;
  for (g=0; g<Dimen; ++g) { rk[g]=0.0; fscanf(fsrc,"%f",&src[g]); }

  /* Produto Uk * Cfwd */
  for (f=0; f<FWDTAPS; ++f) {
   /* printf("Multiplying ... f=%d\n",f); */
   multcomplx(&uk[f],&cfwd[f],&tmp);
   for (g=0; g<Dimen; ++g) rk[g]+=tmp[g];
  }
  /* Produto Bk * Cbck */
  for (f=0; f<BCKTAPS; ++f) {
   /* printf("Multiplying ... f=%d\n",f); */
   multcomplx(&bk[f+1],&cbck[f],&tmp);
   for (g=0; g<Dimen; ++g) rk[g]-=tmp[g];
  }

  /* Cálculo do BER (Bit-Error-Rate) */
  /* printf("region src=%d, dfe=%d\n",region(&src),region(&rk)); */
  if (region(&src)!=region(&rk)) ctber+=1;
  /* printf("ctber=%d, ct=%d %f\n",ctber,ct,(float)ctber/ct); */
  fprintf(fber,"%f\n",(float)ctber/ct);

  /* Cálculo de Ek */
  for (g=0; g<Dimen; ++g) ek[g]=uk[0][g]-rk[g];

  /* Atualização dos coeficientes */
  for (f=0; f<FWDTAPS; ++f) {
   tmp[0]=uk[f][0]; tmp[1]=-uk[f][1];
   multcomplx(&ek,&tmp,&tmp);
   for (g=0; g<Dimen; ++g) cfwd[f][g]+=tmp[g]*lg;
  }
  for (f=0; f<BCKTAPS; ++f) {
   tmp[0]=bk[f+1][0]; tmp[1]=-bk[f+1][1];
   multcomplx(&ek,&tmp,&tmp);
   for (g=0; g<Dimen; ++g) cbck[f][g]+=tmp[g]*lg;
  }

  /* Rotação de Xk */
  for (f=FWDTAPS-1; f>0; --f) {
   /* printf("Rotating Xk... f=%d\n",f); */
   for (g=0; g<Dimen; ++g) uk[f][g]=uk[f-1][g];
  }

  /* Rotação de Bn */
  for (f=BCKTAPS; f>0; --f) {
```

```
    /* printf("Rotating Bk... f=%d\n",f); */
    for (g=0; g<Dimen; ++g) bk[f][g]=bk[f-1][g];
   }
   for (g=0; g<Dimen; ++g) bk[0][g]=rk[g];

   /* Geração dos arquivos dfe_ek.io e dfe_rk.io */
   for (g=0; g<Dimen; ++g) {
    fprintf(frk,"%f ",rk[g]); fprintf(fek,"%f ",ek[g]);
   }
   fprintf(frk,"\n"); fprintf(fek,"\n");
   if (!(ct % ctplot)) {
    fclose(frk); fclose(fek); fclose(fber);
    printf("%d symbols read !",ct); getchar();
    frk = fopen(Fnrk,"w"); fek = fopen(Fnek,"w"); fber=fopen(Fcnvber,"a");
   }
  }
  fclose(fuk); fclose(frk); fclose(fek); fclose(fber); fclose(fsrc);
  printf("\nDFE outputs were succesfully created !\n");

} /*main*/

int region(sym)
 float *sym;
{
 float i=*(sym+0),q=*(sym+1), radio, angle,angle0,angle1, x, y;
 if (distor>4 && distor<8) { /* Distortion is rotation */
  if (i<0.0) angle1=PI; else angle1=0.0;
  if (distor==5) angle0=ROT10;
  if (distor==6) angle0=ROT20;
  if (distor==7) angle0=ROT30;
  angle=angle1+atan(q/i)+angle0; radio=sqrt(i*i+q*q);
  i=radio*cos(angle); q=radio*sin(angle);
 }
 if (distor==8) { /* Distortion is grid collapse */
  if (q>Ac) x=-HGCF*3.0/2.0*Ac;
  else if (q>0.0) x=-HGCF*3.0/2.0*Ac/3.0;
  else if (q>-Ac) x=HGCF*3.0/2.0*Ac/3.0;
  else x=HGCF*3.0/2.0*Ac;
  i=i+x; q=q+y;
 }
// fprintf(fcorr,"%f %f\n",i,q);
 if (i>Ac)
  if (q>Ac) return 4;
  else if (q>0.0) return 8;
  else if (q>-Ac) return 12;
  else return 16;
 else if (i>0.0)
  if (q>Ac) return 3;
  else if (q>0.0) return 7;
  else if (q>-Ac) return 11;
  else return 15;
 else if (i>-Ac)
  if (q>Ac) return 2;
  else if (q>0.0) return 6;
  else if (q>-Ac) return 10;
  else return 14;
 else
  if (q>Ac) return 1;
  else if (q>0.0) return 5;
  else if (q>-Ac) return 9;
  else return 13;
}

void multcomplx(c1,c2,result)
 float *c1, *c2, *result;
{
 *(result+0)=*(c1+0) * *(c2+0) - *(c1+1) * *(c2+1);
 *(result+1)=*(c1+0) * *(c2+1) + *(c1+1) * *(c2+0);
}
```

Anexo 4: Listagem do programa em C para a simulação do equalizador baseado em SOM

```
/*-------------------------------------------------------------------*/
/* Simula um equalizador DFE (Decision Feedback Equalizer) adaptativo */
/* Algoritmo de adaptação dos coeficientes : LMS (Least-Mean-Square)  */
/* Regra de decisão : Algoritmo de Kohonen (Rede Neural)              */
/* Autor : Wilfredo Machaca Luque DSIF/FEEC/UNICAMP  (02-06-98)       */
/*-------------------------------------------------------------------*/
/* Para compilar (Solaris, SunOS) : gcc dfe_knn.c -o dfe_knn -lm      */
/* Sintaxe: dfe_knn [Distortion LoopGain [Step]]]                     */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>  /* requerido pela função exit no Borland C++ */
#include "param.h"

#define Nneu  4   /* Número de filas/colunas de células (neurônios) */
#define ALPHA 0.2
#define BETA 0.01
#define FWDTAPS 2  /* 5 Número de TAPS no caminho direto */
#define BCKTAPS 1  /* 4 Número de TAPS no caminho da realimentação */
#define Fchip "dfe4rk.io"  /*seq. de símbolos para o chip qam_cell4*/
#define Fcr "corr.io"

float wg[Nneu][Nneu][Dimen];  /* Vetor de pesos */
float parvec[Nneu][Nneu][Dimen]; /* Vetor dos parâm. do algoritmo */
float niv[NSF];
int distor;

int region();
float distance();
float winner();
void adapt();
void multcomplx();
void plot_wgmat_to_file();
void read_input_vectors();
FILE *fcorr;

main(int argc, char *argv[]) {
 int ct=0, ctber=0, ctplot=1900, f,c,g,fil,col;
 float lg, dista;
 float uk[FWDTAPS][Dimen], bk[BCKTAPS+1][Dimen];
 float src[Dimen], tmp[Dimen], ek[Dimen], rk[Dimen];
 float cfwd[FWDTAPS][Dimen]={1.0,0.0, 0.0,0.0};//, 0.0,0.1, 0.0,-0.1, 0.0,0.1};
 float cbck[BCKTAPS][Dimen]={0.0,0.0};//, 0.0,-0.1, 0.0,0.1, 0.0,-0.1};
 FILE *fuk, *frk, *fek, *fsrc, *fber, *fci;

 if (argc<2) {
  printf("Syntax : dfe_knn [distor# [loop_gain [#plots]]]\n"); exit(0); }
 else {
  distor=atoi(argv[1]);
  if (argc>=3) {
```

```
    lg=atof(argv[2]);
    if (argc>=4) ctplot=atoi(argv[3]);
  }
  else lg=DELTA;
}
printf("SOM-Based Decision Feedback Equalizer.\n");
printf("\tUsing : Loop Gain=%f;  Distortion#=%d ...\n\n",lg,distor);

for (g=0; g<NSF; ++g) niv[g]=(float)((NSF-1)-2*g);
/* Inicialização dos vetores Xk e Bk */
for (f=0; f<FWDTAPS; ++f) for (g=0; g<Dimen; ++g) uk[f][g]=0.0;
for (f=0; f<BCKTAPS; ++f) for (g=0; g<Dimen; ++g) bk[f][g]=0.0;

/* Lê os pesos iniciais e parâmetros do algoritmo de aprendizado */
read_input_vectors();

fuk = fopen(Fawgn,"r"); frk = fopen(Fnrk,"w"); fek = fopen(Fnek,"w");
fsrc = fopen(Fnsrc,"r"); fber = fopen(Fknnber,"w"); fci=fopen(Fchip,"w");
fcorr = fopen(Fcr,"w");

srand(SEED);
while (fscanf(fuk,"%f",&uk[0][0])!=EOF) {
  fscanf(fuk,"%f",&uk[0][1]); ++ct;
  for (g=0; g<Dimen; ++g) { rk[g]=0.0; fscanf(fsrc,"%f",&src[g]); }

  /* Produto Xk * Cfwd */
  for (f=0; f<FWDTAPS; ++f) {
    /* printf("Multiplying ... f=%d\n",f); */
    multcomplx(&uk[f],&cfwd[f],&tmp);
    for (g=0; g<Dimen; ++g) rk[g]+=tmp[g];
  }
  /* Produto bk * Cbck */
  for (f=0; f<BCKTAPS; ++f) {
    /* printf("Multiplying ... f=%d\n",f); */
    multcomplx(&bk[f+1],&cbck[f],&tmp);
    for (g=0; g<Dimen; ++g) rk[g]-=tmp[g];
  }

  /* Algoritmo de Kohonen */
  dista=winner(&fil,&col,&rk);
  /* Muda os pesos da célula ganhadora */
  adapt(&wg[fil][col],&rk,&parvec[fil][col],ALPHA);
  /* Muda os pesos das celulas vizinhas */
  if (fil-1>=0) adapt(&wg[fil-1][col],&rk,&parvec[fil-1][col],BETA);
  if (fil+1<Nneu) adapt(&wg[fil+1][col],&rk,&parvec[fil+1][col],BETA);
  if (col-1>=0) adapt(&wg[fil][col-1],&rk,&parvec[fil][col-1],BETA);
  if (col+1<Nneu) adapt(&wg[fil][col+1],&rk,&parvec[fil][col+1],BETA);

  /* Cálculo do BER (Bit-Error-Rate) */
  if ( region(&src)!=region(&wg[fil][col]) ) ctber+=1;
  fprintf(fber,"%f\n",(float)ctber/ct);
  /* printf("rk=(%f,%f) (f,c)=(%d,%d) src=(%f,%f) region=%d ctber=%d",
  rk[0],rk[1],fil,col,src[0],src[1],region(&src),ctber); getchar(); */

  /* Cálculo de En */
  for (g=0; g<Dimen; ++g) ek[g]=wg[fil][col][g]-rk[g];

  /* Atualização dos coeficientes */
  for (f=0; f<FWDTAPS; ++f) {
    tmp[0]=uk[f][0]; tmp[1]=-uk[f][1];
    multcomplx(&ek,&tmp,&tmp);
    for (g=0; g<Dimen; ++g) cfwd[f][g]+=tmp[g]*lg;
  }
  for (f=0; f<BCKTAPS; ++f) {
    tmp[0]=bk[f+1][0]; tmp[1]=-bk[f+1][1];
    multcomplx(&ek,&tmp,&tmp);
    for (g=0; g<Dimen; ++g) cbck[f][g]+=tmp[g]*lg;
  }

  /* Rotação de Xk */
  for (f=FWDTAPS-1; f>0; --f) {
    /* printf("Rotating Xn... f=%d\n",f); */
    for (g=0; g<Dimen; ++g) uk[f][g]=uk[f-1][g];
  }

  /* Rotação de Bk */
  for (f=BCKTAPS; f>0; --f) {
    /* printf("Rotating Bk... f=%d\n",f); */
    for (g=0; g<Dimen; ++g) bk[f][g]=bk[f-1][g];
  }
  for (g=0; g<Dimen; ++g) bk[0][g]=wg[fil][col][g];

  /* Geração dos arquivos dfe4rk.io, dfe_ek.io e dfe_rk.io */
  for (g=0; g<Dimen; ++g) {
    fprintf(fek,"%f ",ek[g]);
    fprintf(frk,"%f ",wg[fil][col][g]); //rk[g]);
  }
  f=region(&rk);
  if (f==3 || f==4 || f==7 || f==8)
    for (g=0; g<Dimen; ++g) fprintf(fci,"%f ",rk[g]);
  fprintf(fci,"\n"); fprintf(frk,"\n"); fprintf(fek,"\n");
  if (!(ct % ctplot)) {
    fclose(frk); fclose(fek); fclose(fber);
    plot_wgmat_to_file(&wg);
    printf("%d symbols read !",ct); getchar();
    frk = fopen(Fnrk,"w"); fek = fopen(Fnek,"w"); fber=fopen(Fknnber,"a");
  }
}
fclose(fuk); fclose(frk); fclose(fek); fclose(fber);
fclose(fsrc); fclose(fci); fclose(fcorr);
printf("\ndfe.io was succesfully created !\n");

} /*main*/

int region(sym)
  float *sym;
{
  float i=*(sym+0),q=*(sym+1), radio, angle,angle0,angle1, x, y;
  if (distor>4 && distor<8) { /* Distortion is rotation */
    if (i<0.0) angle1=PI; else angle1=0.0;
    if (distor==5) angle0=ROT10;
    if (distor==6) angle0=ROT20;
    if (distor==7) angle0=ROT30;
    angle=angle1+atan(q/i)+angle0; radio=sqrt(i*i+q*q);
    i=radio*cos(angle); q=radio*sin(angle);
  }
  if (distor==8) { /* Distortion is grid collapse */
    if (q>Ac) x=-HGCF*3.0/2.0*Ac;
    else if (q>0.0) x=-HGCF*3.0/2.0*Ac/3.0;
    else if (q>-Ac) x=HGCF*3.0/2.0*Ac/3.0;
    else x=HGCF*3.0/2.0*Ac;
    i=i+x; q=q+y;
  }
//  fprintf(fcorr,"%f %f\n",i,q);
  if (i>Ac)
    if (q>Ac) return 4;
    else if (q>0.0) return 8;
    else if (q>-Ac) return 12;
    else return 16;
  else if (i>0.0)
```

```
    if (q>Ac) return 3;
    else if (q>0.0) return 7;
    else if (q>-Ac) return 11;
    else return 15;
   else if (i>-Ac)
    if (q>Ac) return 2;
    else if (q>0.0) return 6;
    else if (q>-Ac) return 10;
    else return 14;
   else
    if (q>Ac) return 1;
    else if (q>0.0) return 5;
    else if (q>-Ac) return 9;
    else return 13;
}

void multcompix(c1,c2,result)
  float *c1, *c2, *result;
{
  *(result+0)=*(c1+0) * *(c2+0) - *(c1+1) * *(c2+1);
  *(result+1)=*(c1+0) * *(c2+1) + *(c1+1) * *(c2+0);
}

float distance(v1,v2)
  float *v1, *v2;
{
  int i=Dimen; float ac=0, d;
  while (i--) {
    d = *v1++ - *v2++;
    ac += fabs(d);
  }
  return(ac);
}

void adapt(ww,vv,pv,factor)
  float *ww, *vv, *pv, factor;
{
  int i;
  for (i=0; i<=Dimen-1; ++i)
    *(ww+i)+=factor*( *(pv+0) + *(pv+1)*(*(vv+i)) - *(ww+i) );
}

float winner(fi,ci,pvec)
  int *fi, *ci; float *pvec;
{
  float dist, tmp;
  int i,j,fil=0,col=0;

  tmp = distance(pvec,&wg[0][0]);
  for (i=0; i<=Nneu-1; ++i)
    for (j=0;j<=Nneu-1; ++j) {
      dist = distance(&wg[i][j],pvec);
      if (dist<tmp) {
        tmp=dist;
        fil=i; col=j;
      }
    }
  *fi=fil; *ci=col;
  return(dist);
}

void plot_wgmat_to_file(mm)
  float *mm;
{
  int k,j,i;  FILE *ftmp;
  ftmp = fopen(FPLOT,"w");
  for (k=0; k<=Nneu-1; ++k)
    for (j=0; j<=Nneu-1; ++j) {
      for (i=0; i<=Dimen-1; ++i)
        fprintf(ftmp,"%1.4f ",*(mm+k*Nneu*Dimen+j*Dimen+i));
      fprintf(ftmp,"\n");
    }
  fclose(ftmp);
}

void read_input_vectors(void)
{ /* read Initial Wgs, Params. for learning algorithm */
  FILE *fwgmat, *fparam;
  int j,i,k;
  fwgmat = fopen(FWGINI,"r"); fparam = fopen(FPARAM,"r");
  for (j=0; j<=Nneu-1; ++j)
    for (k=0; k<=Nneu-1; ++k)
      for (i=0; i<=Dimen-1; ++i) {
        fscanf(fwgmat,"%f",&wg[j][k][i]);
        fscanf(fparam,"%f",&parvec[j][k][i]);
      }
  fclose(fwgmat); fclose(fparam);
}
```

Anexo 5: Listagem das descrições VHDL fonte para o projeto do chip

Listagem para a máquina de estados

```
library IEEE;
use IEEE.std_logic_1164.all;

ENTITY control IS
  PORT(
    clk, DiRDY, WinNg, rst : IN std_logic;
    y: OUT std_logic_vector (15 DownTo 0) );
END control;

ARCHITECTURE rtl OF control IS
  TYPE states IS (s0,s1,s3,s2,s6,s7,s5,s4,s12,s13,s15,s14);
  SIGNAL state : states := s0;
  SIGNAL nxt_state : states := s0;
  SIGNAL y_tmp : std_logic_vector(15 DownTo 0);
BEGIN
  -- Process to Latch state on rising edge of "clk"
  clkd: PROCESS (clk, rst)
  BEGIN
    IF (rst='1') THEN
      state <= s0;
    ELSIF (clk'EVENT AND clk = '1' AND clk'LAST_VALUE = '0') THEN
      state <= nxt_state;
    END IF;
  END PROCESS clkd;
```

```
-- Process to determine next state
state_trans: PROCESS (state,DiRDY,WinNg)
BEGIN
  nxt_state <= state;
  CASE state IS
    WHEN s0 => IF (DiRDY='1') THEN
                 nxt_state <= s1;
               ELSIF (WinNg='1') THEN
                 nxt_state <= s6;
               ELSIF (DiRDY='0') THEN
                 nxt_state <= s0;
               ELSIF (WinNg='0') THEN
                 nxt_state <= s0;
               END IF;
    -- Distance
    WHEN s1 => IF (DiRDY='1') THEN
                 nxt_state <= s3;
               ELSIF (DiRDY='0') THEN
                 nxt_state <= s1;
               END IF;
    WHEN s3 => nxt_state <= s2;
    WHEN s2 => nxt_state <= s0;
    -- Winner and Neighboors
    WHEN s6 => nxt_state <= s7;
    WHEN s7 => nxt_state <= s5;
    WHEN s5 => nxt_state <= s4;
    WHEN s4 => nxt_state <= s12;
    WHEN s12 => nxt_state <= s13;
    WHEN s13 => nxt_state <= s15;
    WHEN s15 => nxt_state <= s14;
    WHEN s14 => nxt_state <= s0;
  END CASE;
END PROCESS state_trans;

-- Process to determine output value
out_decode: PROCESS (state)
BEGIN
  CASE state IS
    -- Wait either WinNg or DiRDY signal
    WHEN s0  => y_tmp <= "0011000111111011"; --31FB
    -- Distance.....5432109867543210
    -- Active level  ++++++------
    WHEN s1  => y_tmp <= "0001101001010011"; --1A53
    WHEN s3  => y_tmp <= "0001101010101111"; --1AAF
    WHEN s2  => y_tmp <= "1011101011011111"; --BADF
    -- Active level          ++++++------
    -- Winner and Neighs.  .5432109867543210
    WHEN s6  => y_tmp <= "0011000111011011"; --31DB
    WHEN s7  => y_tmp <= "0011000111110111"; --31F7
    WHEN s5  => y_tmp <= "0011000111011111"; --31DF
    WHEN s4  => y_tmp <= "0010011001101011"; --266B
    WHEN s12 => y_tmp <= "0010011011011011"; --26DB
    WHEN s13 => y_tmp <= "0010011011111110"; --26FE
    WHEN s15 => y_tmp <= "0010011011011111"; --26DF
    WHEN s14 => y_tmp <= "0110011011111101"; --66FD
  END CASE;
END PROCESS out_decode;

-- Process to latch output on falling edge of "clk"
output: PROCESS (clk, rst)
BEGIN
  IF (rst = '1') THEN
    --s0..5432109876543210
    y <= "0011000111111011";
  ELSIF (clk'EVENT AND clk='0' AND clk'LAST_VALUE='1') THEN
    y <= y_tmp;
  END IF;
END PROCESS output;
END  rtl ;
```

Listagem para o bloco de registradores

```
library IEEE, ARITHMETIC;
use IEEE.std_logic_1164.all;
use ARITHMETIC.std_logic_arith.all;

Entity b14_regz_vhdl IS
        Port(   din   : In STD_LOGIC_VECTOR( 13 downto 0);
                st    : In STD_LOGIC;
                eno: In STD_LOGIC;
                dout  : OUT std_logic_vector( 13 downto 0 ) BUS
             );
End b14_regz_vhdl;

Architecture RTL Of b14_regz_vhdl Is
        signal tmp   : STD_LOGIC_VECTOR( 13 downto 0 );
Begin
        latch : Process (st,din)
        Begin
                If (st='1') Then
                  tmp <= din;
                End If;
        End Process latch;

        tri_buffer : BLOCK (eno = '1')
        Begin
                dout <= GUARDED std_logic_vector (tmp) ;
        End BLOCK tri_buffer;

End RTL;
```

Listagem para o bloco WTA

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_1164_extensions.all;

ENTITY wta4_b13 Is
```

```
 PORT (
  d0, d1, d2, d3 : IN STD_LOGIC_VECTOR(12 DOWNTO 0);
  rdy0, rdy1, rdy2, rdy3 : IN STD_LOGIC;
  win        : IN STD_LOGIC;
  dist : OUT STD_LOGIC_VECTOR(12 DOWNTO 0);
  rdy        : OUT STD_LOGIC;
  clk  : IN STD_LOGIC;
  win0, win1, win2, win3 : OUT STD_LOGIC  );

 Signal f01, f23, f0123, d01m, d23m, d0123m : STD_LOGIC;
 Signal rdyx, win01, win23 : STD_LOGIC;
 Signal d01, d23, d0123 : STD_LOGIC_VECTOR(12 DOWNTO 0);

End wta4_b13;

Architecture rtl Of wta4_b13 is

Begin

 rdyx <= rdy0 OR rdy1 OR rdy2 OR rdy3;

 f01 <= (d0 < d1);
 f23 <= (d2 < d3);
 f0123 <= (d01 < d23);

 set_flags : Process (rdyx)
 Begin
  If falling_edge(rdyx) Then
   d01m <= f01;
   d23m <= f23;
   d0123m <= f0123;
  End If;
 End Process set_flags;

 mux01 : Process(f01, d0, d1)
 Begin
  Case f01 Is
   When '0' => d01 <= d1;
   When '1' => d01 <= d0;
   When Others => NULL;
  End Case;
 End Process mux01;

 mux23 : Process(f23, d2, d3)
 Begin
  Case f23 Is
   When '0' => d23 <= d3;
   When '1' => d23 <= d2;
   When Others => NULL;
  End Case;
 End Process mux23;

 mux0123 : Process(f0123, d01, d23)
 Begin
  Case f0123 Is
   When '0' => dist <= d23;
   When '1' => dist <= d01;
   When Others => NULL;
  End Case;
 End Process mux0123;

 rdy_ffd : Process (clk)
 Begin
  If falling_edge(clk) Then
   rdy <= rdyx;
  End If;
 End Process rdy_ffd;

 win01 <= win AND d0123m;
 win23 <= win AND not d0123m;

 win0 <= win01 AND d01m;
 win1 <= win01 AND not d01m;
 win2 <= win23 AND d23m;
 win3 <= win23 AND not d23m;

End rtl;
```

Anexo 6: Listagem do programa em Assembly para o interfaceamento entre o módulo DSP96002ADM (Motorola) e a placa de teste do chip.

```
;*** DO.L Register is used as 32-bit control word, composed as following:
;D(31) D(30) D(29) D(28)   D(27) D(26) D(25) D(24)   D(23) D(22) D(21) D(20)   D(19) D(18)   D(17)   D(16)
;       Reset CLK  DiRdy   Win  Mode  WR   RdWg   NiH1  NiH0  NiV1  NiV0   BSel  RgAd2 RgAd1 RgAd0
;D(15)  D(14)  D(13)  D(12)   D(11) D(10) D(9  D(8)    D(7)  D(6)  D(5)  D(4)   D(3)  D(2)  D(1)  D(0)
;BAdr1 BAdr0 SCel1 SCel0   Din11                      Din7                       Din3              Din0


ZERO    equ $00000000
QAMPORT  equ $FFFFFFFD
Rst     equ $1E
Clk     equ $1D
DiRdy   equ $1C
Mode    equ $1A
WR      equ $19
RdWg    equ $18
Bsel    equ $13
DinMsk   equ $FFFFF000
RgAdrMsk equ $FFF8FFFF
SCellMsk equ $FFFFCFFF
BAddrMsk equ $FFFF3FFF


; Alpha, Beta, Vdc and Vg are stored starting in x:$200. Same values for all the
cells !
;  %=Binary $Hex. `=Dec
       org x:$200
Wi      dc $00000000 ; Defines Bits for Wi, Wq Registers Addressing
Wq      dc $00010000
Alpha   dc $00020000 ; Defines Bits for Alpha, Beta, Vg, Vdc Registers
Addressing
Beta    dc $00030000
Vdc     dc $00040000
Vg      dc $00050000
CellA   dc $00000000 ; Defines Bits for Cells Addressing
```

```
CellB   dc $00001000
CellC   dc $00002000
CellD   dc $00003000
MP2_Bus  dc $0000C000
MC1_Bus  dc $00008000
MP1_Bus  dc $00004000
MP0_Bus  dc $00000000
AlphaV   dc $00000300 ; $199
BetaV    dc $00000040 ; $014
VdcV     dc $00000000 ; 0
VgV      dc $00000800 ; $800
WiA      dc $00000C00 ; ($199,$4CC) = (-0.2,0.2)
WqA      dc $00000400
WiB      dc $00000C00 ; ($199,$199)= (-0.2,-0.2)
WqB      dc $00000C00
WiC      dc $00000400 ; ($4CC,$199)= (0.2,-0.2)
WqC      dc $00000c00
WiD      dc $00000400 ; ($4CC,$4CC)= (0.2,0.2)
WqD      dc $00000400
DINi     ds 1 ; -> Dini=$CCD
DINq     ds 1 ; -> Dinq=$400
WgRet    ds 8

         org p:$200

Start0
         move #DinMsk,d4.l
         move #RgAdrMsk,d5.l
         move #SCellMsk,d6.l
         move #BAddrMsk,d7.l

Cfg_Call bsr Config          ; Return with the last D0.L control word sent to
QAM_CELL chip

Start1   bset #Mode,d0.l     ; Sets FSM mode, leave USER mode
         and d4.l,d0.l       ; applies DinMsk to Do
         move x:DINi,d2.l
         or d2.l,d0.l        ; Wi value to Do

DiRdyBegin do #2,DiRdyLp
         bset #DiRdy,d0.l    ; Sets DiRdy
         movep d0.l,y:QAMPORT
         bset #Clk,d0.l      ; Sets CLK
         movep d0.l,y:QAMPORT
         bclr #DiRdy,d0.l    ; Resets DiRdy
         movep d0.l,y:QAMPORT
         bclr #Clk,d0.l      ; Resets CLK
         movep d0.l,y:QAMPORT
DiRdyLp
         and d4.l,d0.l       ; applies DinMsk to D0
         move x:DINq,d2.l
         or d2.l,d0.l        ; Wq value to D0
         movep d0.l,y:QAMPORT

Clk_Begin do #11,clk_Lp
         bset #Clk,d0.l      ; Sets CLK
         movep d0.l,y:QAMPORT
         bclr #Clk,d0.l      ; Reset CLK
         movep d0.l,y:QAMPORT
clk_Lp
         bclr #Bsel,d0.l     ; Selects BUSES bus
         bclr #Mode,d0.l     ; Sets USER mode, leave FSM mode
         and d7.l,d0.l       ; Clean the BAddr sector
         move x:MP2_Bus,d1.l ; Prep. D1 to select MP2_Bus
         or d1.l,d0.l        ; Puts BAddr in D0
         move #CellA,r0      ; points to CellA
         move #WgRet,r1      ; points to WgRet. Memory area for udpated weights

RdWg_Begin do #4,RdWg_Lp
         move x:(r0)+,d1.l
         and d6.l,d0.l       ; Clean the SCell sector
         or d1.l,d0.l        ; Put Scell in D0
         bclr #RdWg,d0.l     ; Resets RdWg
         movep d0.l,y:QAMPORT
         rep #4
         nop
         movep y:QAMPORT,x:(r1)+ ; Reads Wi from QAM_CELL4 chip
         bset #RdWg,d0.l     ; Sets RdWg
         movep d0.l,y:QAMPORT
         rep #4
         nop
         movep y:QAMPORT,x:(r1)+ ; Reads Wq from QAM_CELL4 chip
RdWg_Lp
         debugal

;******** Routines ***************
Config   move #ZERO,d0.l
         movep d0.l,y:QAMPORT
         bset #Rst,d0.l
         movep d0.l,y:QAMPORT  ; Resets the QAM_CELL4 chip
         bclr #Rst,d0.l
         movep d0.l,y:QAMPORT

         bset #Bsel,d0.l     ; Selects DIN bus
         move #CellA,r2      ; point to CellA
         move #WiA,r3        ; point to WiA
Cfg_all  do #4,Cfg_Lp
         move x:(r2)+,d1.l   ; Prepares D1 for selecting Cell ?
         and d6.l,d0.l       ; clean the SCell sector
         or d1.l,d0.l        ; Put Scell in D0

         move #Alpha,r0      ; point to Alpha
         move #AlphaV,r1     ; point to AlphaV
cfg_rgs  do #4,CRg_Lp
         move x:(r0)+,d2.l   ; Prepares D2 for addressing Alpha, Beta, Vdc, or Vg
regs.
         move x:(r1)+,d3.l   ;      D3 to set Alpha, Beta, Vdc or Vg values at D0
         and d4.l,d0.l       ; Cleans the DIN sector
         and d5.l,d0.l       ; Cleans the RgAddr sector
         or d3.l,d2.l        ; Alpha, Beta, Vdc or Vg value to D0
         or d2.l,d0.l        ;              address to D0
         movep d0.l,y:QAMPORT  ; Write in DIN bus. Select Din bus, Cell ?, and ?
reg
         bset #WR,d0.l
         movep d0.l,y:QAMPORT  ; Generate a pulse in WR
         bclr #WR,d0.l
         movep d0.l,y:QAMPORT
CRg_Lp
         move #Wi,r0         ; point to Wi
Cfg_wg   do #2,CWg_Lp
         move x:(r0)+,d2.l   ; Prepares D2 for addressing Wi and Wq regs.
         move x:(r3)+,d3.l   ;      D3 to set Wi and Wq values at D0
         and d4.l,d0.l       ; Cleans the DIN sector
         and d5.l,d0.l       ; Cleans the RgAddr sector
         or d3.l,d2.l        ; Wi or Wq value to D0
         or d2.l,d0.l        ;      address to D0
         movep d0.l,y:QAMPORT  ; Write in DIN bus. Select Din bus, Cell ?, and ?
reg
         bset #WR,d0.l
         movep d0.l,y:QAMPORT  ; Generate a pulse in WR
```

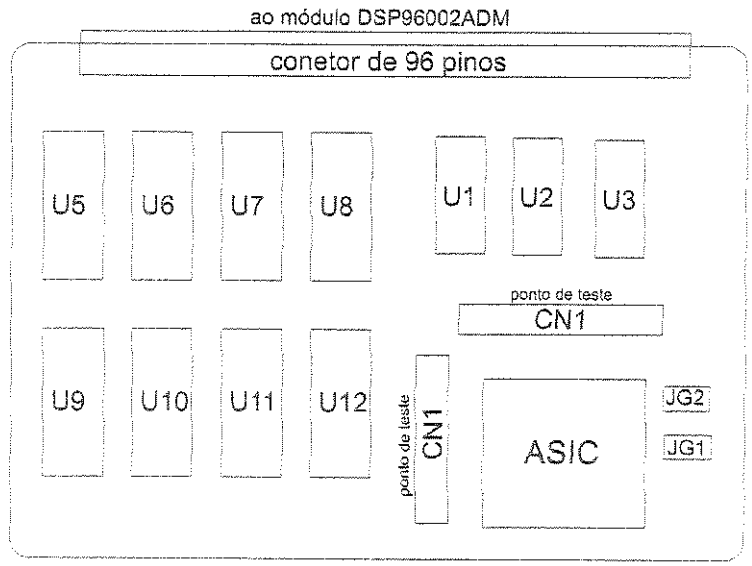
```
      bclr #WR,d0.l
      movep d0.l,y:QAMPORT
CWg_Lp  nop
Cfg_Lp  rts

      end
```

Anexo 7 Diagramas de pinagem do chip - *Floorplan* do PCB da placa de interface.

	A	B	C	D	E	F	G	H	J	K	L
1	X	NC 60	VDD 58	WR 56	DtRdy 54	SCell(0) 52	RdWg 50	NC 48	Do(11) 46	Dst(12) 44	X
2	Din(3) 61	RgAddr(0) 62	PVDD 59	Bsel 57	Reset 55	CLK 53	SCell(1) 51	Mode 49	Do(1) 47	Dst(11) 45	Dst(9) 43
3	RgAddr(1) 63	RgAddr(2) 64				Ref.				Dst(10) 42	Dst(7) 41
4	Din(2) 65	Din(7) 66								Dst(8) 40	Dst(5) 39
5	Din(0) 67	Din(1) 68				(*Socket* para JLCC 68) Vista superior				Dst(6) 38	Dst(3) 37
6	Din(10) 1	Din(11) 2				NC=Não conectado				Dst(4) 36	Dst(1) 35
7	Din(4) 3	Din(5) 4								Dst(2) 34	Dst(0) 33
8	Din(6) 5	Din(8) 6								Do(10) 32	Do(9) 31
9	Din(9) 7	BAddr(0) 8								Do(8) 30	Do(7) 29
10	BAddr(1) 9	NiV0 11	NoH1 13	DstRdy 15	NiV1 17	NiH1 19	PVSS 21	Do(0) 23	Do(3) 25	Do(6) 28	Do(5) 27
11	X	NC 10	NoV0 12	NoV1 14	DoRdy 16	Win 18	NiH0 20	VSS 22	Do(2) 24	Do(4) 26	X

Diagrama de pinagem



Floorplanning da placa de teste