SENSIBILIDADE DE PRODUTOS INTERNOS À FABRICAÇÃO EM CIRCUITOS CMOS PARA QUANTIZAÇÃO VETORIAL EM SISTEMAS DE COMPRESSÃO DE IMAGENS NO PLANO FOCAL

Marcos José Carvalho de Mello

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Antonio Petraglia, Ph.D.

Prof. José Gabriel Rodríguez Carneiro Gomes, Ph.D.

Prof. José Vicente Calvano, D.Sc.

Prof. Antonio Carneiro de Mesquita Filho, Dr. d'État

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2007

MELLO, MARCOS JOSÉ CARVALHO DE

Sensibilidade de Produtos Internos à Fabricação em Circuitos CMOS para Quantização Vetorial em Sistemas de Compressão de Imagens no Plano Focal

[Rio de Janeiro] 2007

XII, 71 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1.Microeletrônica 2.Circuitos Analógicos 3.Compressão de Imagens

I.COPPE/UFRJ II.Título (série)

Dedico este trabalho à minha avó Juracy (*in memoriam).*

Agradecimentos

Em primeiro lugar gostaria de agradecer à minha esposa Paula, por estar ao meu lado mesmo nos momentos mais difíceis. Por ela tudo isso valeu a pena.

Agradeço à minha mãe Marly, que sempre primou pela qualidade da minha educação.

Obrigado ao professor Antonio Petraglia, por ter me proporcionado esta oportunidade de crescimento e por sua demonstração sincera de interesse. Não poderia deixar de acrescentar que exerceu sua função de orientador com maestria e perfeição durante todo o curso.

Obrigado ao professor José Gabriel Rodríguez Carneiro Gomes, por estar sempre disponível nos momentos em que precisei tirar dúvidas, mesmo à noite e nos fins de semana. Deixo registrado que sou profundo admirador de sua inteligência e dedicação.

Agradeço aos professores da banca Antonio Carneiro de Mesquita Filho e José Vicente Calvano, pelo interesse e pela atenção dispensada a esta dissertação.

À Itautec S.A., representada aqui pelos Srs. Paulo Luis Falcão da Motta, Alberto Nogueira Júnior e Robson Luiz da Silva Leal, que me apoiaram e permitiram flexibilização de horários nos dias em que precisei me ausentar do trabalho.

Meu muito obrigado aos amigos do CEFET-RJ, em especial ao Anderson de Oliveira Mello e a sua esposa Leila C. de Brito Mello, a quem sou grato pela amizade e carinho que sempre demonstraram.

Sobretudo não poderia esquecer de agradecer a Deus, porque sem Sua vontade nada disso teria se realizado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SENSIBILIDADE DE PRODUTOS INTERNOS À FABRICAÇÃO EM CIRCUITOS CMOS PARA QUANTIZAÇÃO VETORIAL EM SISTEMAS DE COMPRESSÃO DE IMAGENS NO PLANO FOCAL

Marcos José Carvalho de Mello

Março / 2007

Orientadores: Antonio Petraglia

José Gabriel Rodríguez Carneiro Gomes

Programa: Engenharia Elétrica

Nesta dissertação apresentamos um circuito elétrico que será utilizado como modelo para a análise da sensibilidade de produtos internos aos erros na implementação do *hardware* analógico. Este modelo se baseia em simulações feitas com o Spice, e pode ser utilizado para se desenvolver sistemas de compressão de imagem com sensores CMOS. Este modelo é comparado com modelos anteriores e se mostrou equivalente ao modelo com precisão de 4 bits. Para 6 bits de precisão, os resultados diferem das previsões teóricas feitas pelo modelo anterior. Os resultados aplicados à compressão de imagens também são mostrados.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SENSITIVITY OF INNER PRODUCTS TO CMOS CIRCUITS IMPLEMENTATION FOR VECTOR QUANTIZATION IN FOCAL PLANE IMAGE COMPRESSION SYSTEMS

Marcos José Carvalho de Mello

March/2007

Advisors: Antonio Petraglia

José Gabriel Rodríguez Carneiro Gomes

Department: Electrical Engineering

We present an electrical circuit which will be used as a model for the analysis of the sensitivity of inner products to analog hardware implementations errors. This model is based on Spice simulations, and it can be used to develop image compression systems with CMOS sensors. This model is compared with previous models, and it is shown to be equivalent to the previous model for 4-bit precision. For 6-bit precision, the results differ from the theoretical predictions made by the previous model. Image compression results are also presented.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Em sistemas de imagem CMOS (*Complementary Metal-Oxide Silicon*) convencionais, todas as amostras analógicas dos sensores ópticos são convertidas em amostras digitais por conversores A/D. Depois, elas são armazenadas em uma memória intermediária, que normalmente é acessada por um algoritmo de compressão de imagens, antes do armazenamento definitivo. O *hardware* pode ser bastante simplificado se a compressão da imagem for feita diretamente no plano focal[1], antes da conversão A/D. Nos últimos anos, várias pesquisas vêm sendo feitas neste sentido [22, 33]. Para que o circuito de compressão da imagem caiba dentro de uma pequena área dentro do bloco de *pixels*, o algoritmo deve ser o mais simples possível ao mesmo tempo em que se tenha uma taxa e distorção ótimas, dada a limitação de complexidade. Em trabalho anterior [14] foi proposto um esquema com blocos de 4×4 *pixels*, luminância[2] média codificada através de DPCM (*differential pulse-code*

[1]Em uma câmera, o plano focal é a região onde a imagem é projetada pela lente. Em câmeras digitais, este é o local onde são colocados os sensores de imagem.

[2]A luminância representa a razão entre a intensidade luminosa emitida por uma superfície e a área desta mesma superfície projetada sobre um plano perpendicular à sua direção.

modulation), e quantização vetorial (VQ). Em vez de serem utilizados mapas auto-organizáveis (SOM – *self-organizing maps*), este quantizador vetorial foi projetado com *perceptrons* de múltiplas camadas (MLP – *multilayer perceptrons*), para redução da complexidade.

Também anteriormente [14], foi sugerido um modelo teórico para análise da degradação gerada na compressão bem como erros[3] decorrentes do processo de fabricação do *hardware*. Outros modelos estatísticos para fabricação de MLPs também foram propostos [34], mas eles não puderam ser utilizados para análise conjunta da taxa e compressão, distorção e complexidade, o que se torna necessário para o desenvolvimento de sistemas de compressão de imagens.

Nesta dissertação iremos demonstrar o uso de sinapses implementadas com apenas um transistor, para a construção de circuitos analógicos que realizem a operação de produtos internos, que serão utilizados na implementação de MLPs.

No Capítulo 2 iremos fazer um breve resumo sobre quantizadores vetoriais, bem como uma pequena descrição das redes neurais, apresentando o modelo básico do neurônio e os *perceptrons* de múltiplas camadas.

O Capítulo 3 introduz o conceito de compressão de imagens no plano focal e mostra os dois principais sensores de imagem utilizados em câmeras digitais, os sensores CCDs (*Charge-Coupled Devices*) e o sensores CMOS, apresentado as vantagens e desvantagens de cada tipo.

O Capítulo 4 aborda a teoria sobre os circuitos denominados *current conveyors*. São mostradas as características principais dos CCI e CCII (*current con-*

[3]Ao longo desta dissertação, todas as vezes que utilizarmos o termo "erro", estaremos indicando tão somente os desvios que ocorrem nos parâmetros do circuito durante a sua fabricação, e não, como poderia se pensar, defeitos do processo.

veyors de primeira e segunda geração, respectivamente) e suas representações através de blocos *nullator/norator*. São também demonstradas a implementação e simulação com o Spice de um *current conveyor* a ser utilizado em nosso projeto.

Temos no Capítulo 5 um estudo sobre a sensibilidade dos *perceptrons* de múltiplas camadas e dos quantizadores vetoriais restritos em entropia (ECVQ) aos erros de implementação introduzidos pelo processo de fabricação, nas aplicações de compressão de imagens.

A avaliação da viabilidade da construção e integração de circuitos analógicos que efetuam a compressão de imagens no plano focal motivou a implementação de um modelo elétrico, através do Spice, para a realização da operação de um produto interno (necessário para a construção de MLPs), bem como os efeitos dos erros introduzidos pelo processo de fabricação na qualidade da imagem comprimida. Os resultados que foram obtidos através das simulações são mostrados no Capítulo 6. Neste capítulo também são apresentadas algumas sugestões de continuidade para o trabalho.

Finalmente, no Capítulo 7 apresentamos as conclusões deste trabalho.

Capítulo 2

Quantização Vetorial

A quantização vetorial (VQ)[1] é uma técnica bastante conhecida para compressão de dados [13], onde o codificador trabalha com vetores M-dimensionais $\mathbf{x}(n), n = 1, \ldots, M$, mapeando cada vetor em um índice $j(n)$ que será posteriormente utilizado para reconstrução dos vetores originais, mantidos certos limites de precisão. Na maioria das aplicações de compressão de dados utilizando quantização vetorial, o sistema de codificação de blocos é implementado inteiramente através de *hardware* digital, o que significa que cada componente do vetor $\mathbf{x}(n)$ é uma representação digital da amostra analógica original. A conversão analógico-digital (A/D) de cada componente de $\mathbf{x}(n)$ é um estágio de compressão implementado por um banco de M quantizadores escalares (SQ). Na realidade, um VQ digital possui dois estágios de compressão como mostrado na Figura 2.1. Se houver disponibilidade de acesso direto aos vetores analógicos, então o VQ pode codificá-los diretamente. A Figura 2.2 ilustra este conceito. Este último possui uma menor complexidade para

[1]Neste texto, a notação VQ será utilizada simultaneamente para representar as expressões "quantização vetorial" e "quantizador vetorial", sendo que o contexto mostrará qual foi a representação usada.

sua implementação, uma vez que não necessitamos do banco de conversores A/D e nem de um meio de armazenamento intermediário das amostras digitais.

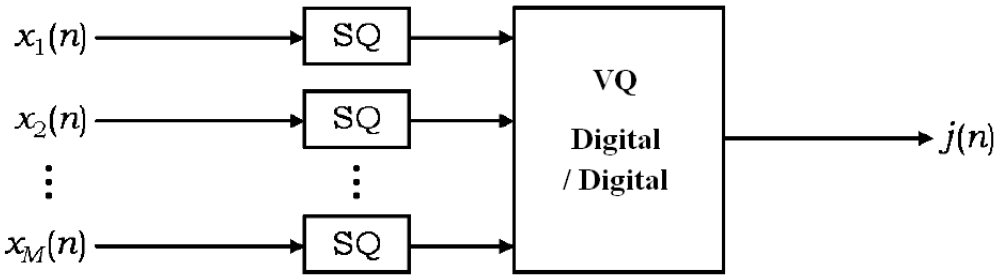


Figura 2.1: VQ com vetores de entrada digitais

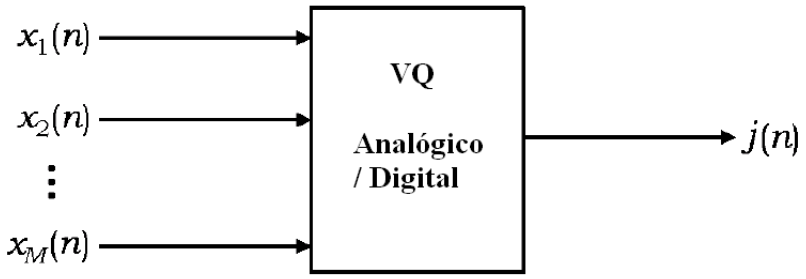


Figura 2.2: VQ com vetores de entrada analógicos

Um quantizador nada mais é do que um aproximador. Um exemplo de quantizador escalar é mostrado na Figura 2.3. Aqui, qualquer número menor ou igual a -2 é aproximado para -3. Números maiores que -2 e menores ou iguais a 0 são aproximados para -1. Números maiores que 0 e menores ou iguais a 2 são aproximados para 1, e qualquer número maior que 2 é aproximado para 3. Note que os valores para os quais os números são aproximados são representados por 2 bits. Este é, portanto, um quantizador de 2 bits, unidimensional.

Um exemplo de quantizador vetorial é mostrado na Figura 2.4. Cada par

de números que aponta para uma coordenada dentro de uma região particular é aproximado por um ponto (marcado na figura por um ponto escuro) associado com aquela região. Note que existem 16 regiões que podem ser representadas por 4 bits. Este é um quantizador de 4 bits, bidimensional.

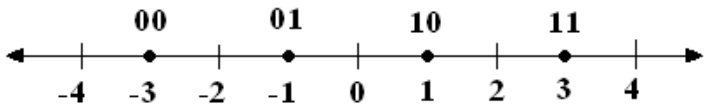


Figura 2.3: Exemplo de quantizador escalar

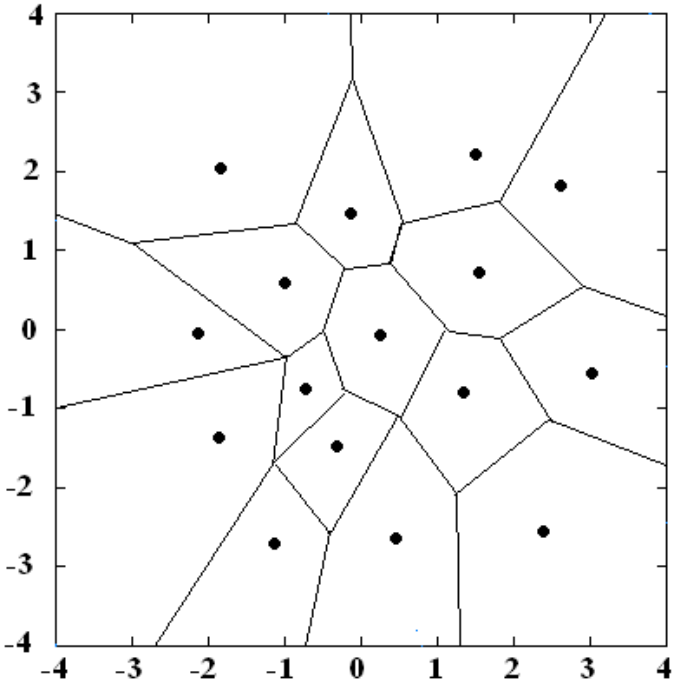


Figura 2.4: Exemplo de quantizador vetorial

Nos dois exemplos anteriores, os pontos são chamados centróides e as regiões definidas pelas bordas são chamadas de células de quantização. O conjunto de centróides é chamado de dicionário.

Um VQ completo é composto por duas operações. A primeira é o codificador, e a segunda é o decodificador. O codificador lê o vetor de entrada e procura o centróide que oferece a menor distorção. Neste caso, a menor distorção é encontrada calculando a distância Euclideana entre o vetor de entrada e cada um dos centróides no dicionário. A distância Euclideana é definida conforme mostra a Equação (2.1), onde x_j é o j-ésimo componente do vetor de entrada $\mathbf{x}$, e y_{ij} é o j-ésimo componente do centróide y_i.

$$d(x, y_i) = \sqrt{\sum_{j=1}^{k} (x_j - y_{ij})^2} \tag{2.1}$$

Uma vez que o centróide mais próximo é encontrado, seu índice é enviado através do canal (o canal pode ser uma memória para armazenamento ou um meio de comunicação, por exemplo). Quando o decodificador recebe o índice do centróide, ele o substitui pelo centróide correspondente. A Figura 2.5 mostra o diagrama de blocos da operação do codificador e do decodificador.

O desenvolvimento de sistemas como os da Figura 2.2, com entradas analógicas, foi estudado tanto por projetistas de circuitos, quanto por pesquisadores nas áreas de compressão de dados e de redes neurais [4, 13].

A Seção 2.1 descreve as Redes Neurais utilizando MLPs e sua utilização em sistemas de codificação de blocos.

2.1 Redes Neurais - MLPs

Desde o início do estudo das redes neurais artificiais, seus pesquisadores têm sido motivados pelo modo como o cérebro humano consegue processar as informações

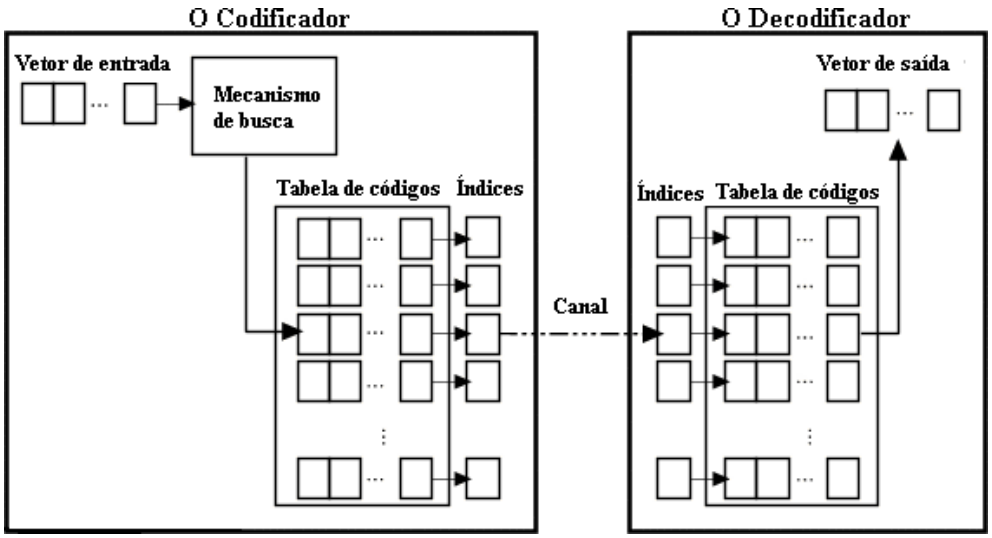


Figura 2.5: O codificador e o decodificador em um VQ

de forma bastante diferente dos computadores digitais convencionais. De um modo geral, podemos dizer que o cérebro humano é uma central de processamento de informações extremamente complexa, capaz de realizar funções de forma não-linear e paralela. Sua estrutura básica constituinte é o neurônio. Juntos, os neurônios têm a capacidade de realizar várias funções (como por exemplo, a percepção dos nossos sentidos físicos, atividades motoras, reconhecimento de objetos) [18].

Uma das propriedades que mais intriga os pesquisadores é a capacidade que o cérebro humano possui de reorganizar constantemente sua estrutura. Através da experiência adquirida e da percepção de novas informações, os neurônios criam novas ligações com seus vizinhos, através das sinapses, estabelecendo regras próprias que, ao longo do tempo, constituem o processo de aprendizagem do indivíduo.

Podemos então oferecer a seguinte definição de uma rede neural [18]:

Uma rede neural é um processador paralelamente distribuído, e constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se

assemelha ao cérebro em dois aspectos:

- *O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.*
- *Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.*

2.1.1 Modelo do Neurônio

O modelo matemático de um neurônio artificial pode ser descrito através das seguintes equações:

$$\begin{aligned} u_k &= \sum_{j=1}^{m} w_{kj} x_j \\ y_k &= \varphi(v_k) \end{aligned} \tag{2.2}$$

onde

$$v_k = (u_k + b_k) \tag{2.3}$$

Nesta equação, $x_1, x_2, \ldots, x_m$ são os sinais de entrada; $w_{k1}, w_{k2}, \ldots, w_{km}$ são os pesos sinápticos ou multiplicadores do neurônio k; v_k é chamado de campo local induzido ou potencial de ativação, e é dado pela soma $u_k + b_k$; u_k é a saída do somador no qual é composto o produto interno $\mathbf{x}^T\mathbf{w}$; b_k é o *bias*; $\varphi(\cdot)$ é a função de ativação; e y_k é a saída do neurônio. A Figura 2.6 mostra o modelo de um neurônio, que forma a base para o projeto de redes neurais artificiais.

A função de ativação $\varphi(.)$ define a saída do neurônio. Podemos identificar três tipos básicos:

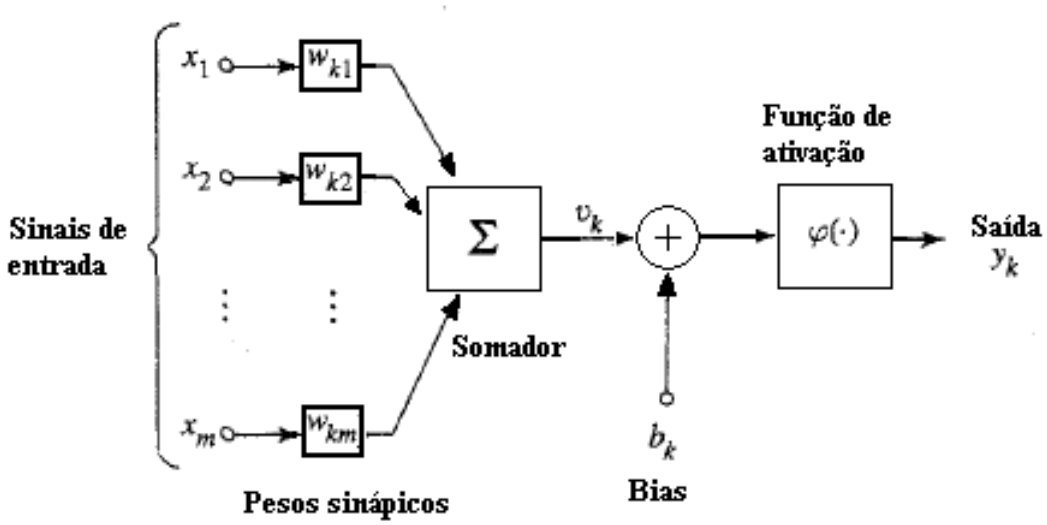


Figura 2.6: Modelo de um neurônio artificial

1. Função de Limiar

   Esta função de ativação, também chamada função de *Heaviside*, é expressa pela Equação

$$\varphi(v) = \begin{cases} 1 & \text{se} \quad v \geq 0 \\ 0 & \text{se} \quad v < 0 \end{cases} \tag{2.4}$$

   e sua forma característica é mostrada na Figura 2.7. O neurônio que utiliza esta função também recebe o nome de modelo de *McCulloch-Pitts* por causa do trabalho realizado por estes pesquisadores [25].

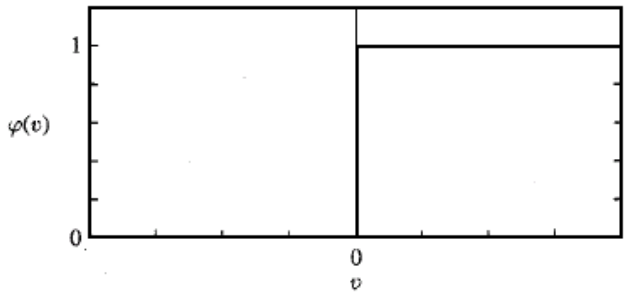


Figura 2.7: Função de limiar ou de Heaviside

2. *Função Linear por Partes*

   A função linear por partes é descrita pela Equação

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v + \frac{1}{2}, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{2.5}$$

   cuja forma característica pode ser vista na Figura 2.8.

   O fator de amplificação na região linear de operação é igual a 1. Se a região linear de operação for mantida sem que o neurônio entre em saturação, temos um combinador linear.

3. *Função Sigmóide*

   A função sigmóide é a forma mais comum de função de ativação utilizada na construção de redes neurais artificiais. Um exemplo de função sigmóide é a função logística , definida por:

$$\varphi(v) = \frac{1}{1 + e^{-av}} \tag{2.6}$$

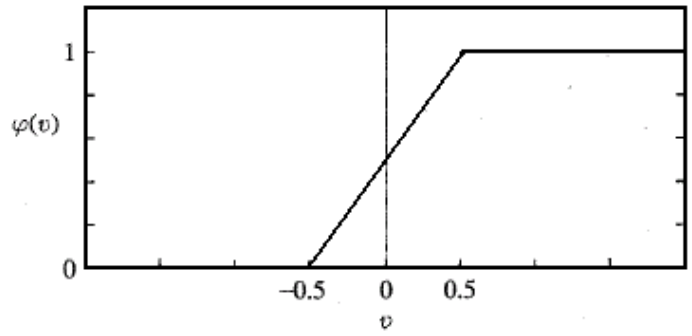


Figura 2.8: Função linear por partes

Nesta equação, a determina a inclinação da função sigmóide. Variando-se o parâmetro a, obtemos funções sigmóides com diferentes inclinações, como podemos ver na Figura 2.9. Outra forma de função sigmóide é a tangente hiperbólica, definida na Equação (2.7):

$$\varphi(v) = \tanh(v) \tag{2.7}$$

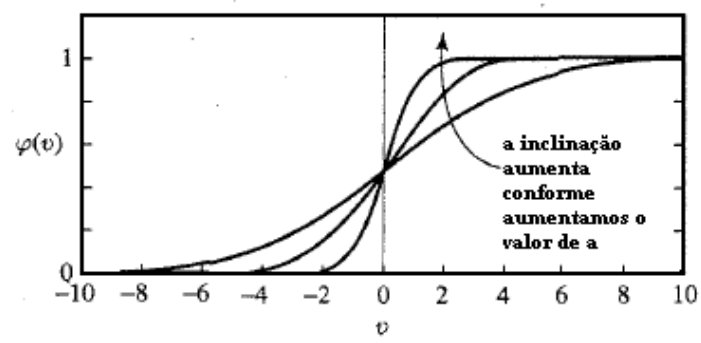


Figura 2.9: Função sigmóide

Na próxima seção iremos definir o que são os *perceptrons* de múltiplas camadas (MLPs) bem como mostrar suas principais características.

2.2 *Perceptrons* de Múltiplas Camadas – MLPs

O elemento mais simples de uma rede neural artificial, usado para classificação de padrões linearmente separáveis, é o *perceptron*. Em sua forma básica ele consiste de um neurônio único, com pesos multiplicadores (sinapses) ajustáveis e um *bias* [18]. Quando temos um *perceptron* que é constituído de apenas um neurônio, ficamos limitados à classificação de padrões em apenas duas classes. Para que possamos ser capazes de realizar classificações com mais de duas classes, precisamos aumentar a camada de saída do *perceptron* de forma a ter mais de um neurônio. Como este *perceptron* está limitado a uma única camada, ele é denominado *perceptron* de camada única (SLP – *single layer perceptron*) e as classes devem ser linearmente separáveis para que o mesmo funcione de forma adequada.

Já as redes neurais de múltiplas camadas (MLP – *multilayer perceptron*) consistem em um conjunto de unidades sensoriais que correspondem à camada de entrada, uma ou mais camadas intermediárias ou ocultas de neurônios, e uma camada de saída também de neurônios. O sinal de entrada se propaga para frente através da rede, camada por camada, e por isso classificamos este tipo de rede como *feedforward*. O MLP, diferentemente do SLP, pode realizar a classificação entre classes que não são linearmente separáveis.

A Figura 2.10 apresenta um MLP típico, onde o vetor de entrada $\mathbf{x}(n)$ é mapeado no vetor de saída $\mathbf{b}_R(n)$. Sua estrutura é composta por $\mathcal{K}$ camadas. A k-ésima camada tem como vetor de entrada a saída da camada anterior $(k-1)$, e desta forma podemos definir a operação de um MLP através da Equação (2.8)

$$\begin{aligned} \mathbf{u}_k(n) &= \mathbf{W}_k^T \mathbf{o}_{k-1}(n) + \mathbf{b}_k \\ \mathbf{o}_k(n) &= \varphi(\mathbf{u}_k(n)) \end{aligned} \tag{2.8}$$

onde: $\mathbf{W}_k$ é a matriz de pesos sinápticos; $\mathbf{b}_k$ é o vetor de *bias* da k-ésima camada; $\mathbf{o}_k(n)$ é a saída da k-ésima camada; $\varphi(\cdot)$ é a função de ativação; e $\mathbf{u}_k$ é o vetor de potencial de ativação.

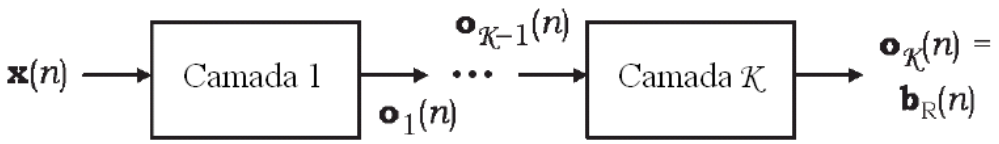


Figura 2.10: Diagrama de um MLP

Como em nosso trabalho estamos focando o uso de MLPs para a construção de VQs, podemos dizer que as saídas $\mathbf{b}_R(n)$ são os índices dos centróides desejados. Note que estes índices estão representados em formato binário.

Capítulo 3

Sensores de Imagem

3.1 Compressão de Imagens no Plano Focal

Nos últimos anos, a pesquisa sobre sensores de imagem do tipo CMOS tem aumentado em comparação aos sensores CCD. Atualmente muitas câmeras digitais ainda utilizam sensores CCD, mas existem vários estudos que demonstram as vantagens dos sensores CMOS com relação à velocidade, tamanho, modo de endereçamento e consumo [11].

Uma das tecnologias básicas dos sensores de imagem do tipo CMOS são os Sensores de Pixel Ativo, APS (*active-pixel sensor*). A característica principal de um APS é o uso de pelo menos um elemento ativo dentro de cada *pixel* [27]. As amostras analógicas de cada *pixel* são endereçadas por colunas, e depois lidas por conversores A/D fora da matriz de *pixels*. Nos APS, normalmente cada coluna possui seu próprio conversor A/D, mas também existem implementações de apenas um conversor A/D de alta velocidade para a matriz inteira [35].

Desde meados dos anos 90, duas tecnologias importantes surgiram na área de

pesquisa de sensores CMOS: o *digital-pixel* (DPS) e o *smart-pixel*. O DPS possui o elemento ativo dentro do *pixel*, onde cada um dos *pixels* possui seu próprio conversor A/D (Figura 3.1(a)). Um DPS se comporta externamente como uma memória digital, ou seja, um *pixel* pode ser lido diretamente do sensor através do endereçamento de sua linha e coluna. O uso de um conversor A/D para cada *pixel* permite a captura de imagens com taxas da ordem de 10.000 quadros por segundo sem compressão [19] e, portanto, é necessário o uso de grandes quantidades de memória para armazenar os dados de um DPS antes que a compressão da imagem possa ser feita.

A tecnologia *smart-pixel* permitiu o processamento de sinais no nível do *pixel*, em um sistema CMOS [9]. Com o *smart-pixel*, o processamento pode ser feito de forma analógica ou de forma mista (processamento analógico e digital) e, portanto, aplicado antes da conversão A/D, ou até mesmo sem os conversores A/D. Dentro da matriz de *pixels*, este processamento analógico requer que o *pixel* obtenha informações de seus vizinhos. Estes sistemas também são chamados de processadores de imagem no plano focal. Com estes, várias operações de processamento de imagem no plano focal (detecção de borda [5], detecção de movimento [12], filtros passa-alta e filtros passa-baixa [23]) foram implementadas com bons resultados.

Também no plano focal podemos realizar compressão de imagem, conforme sugerido em [33]. Alguns autores propuseram implementação preditiva para realizar compressão de dados diretamente ao nível do *pixel* [20, 21, 22], através da realização de operações com *pixels* vizinhos, de forma a "prever" o valor do *pixel* e codificar apenas o valor escalar residual. O objetivo principal nestas aplicações de compressão é fazer o codificador pequeno o bastante para ser implementado no próprio sensor, e ao mesmo tempo robusto, de tal forma que não seja sensível a variações do processo

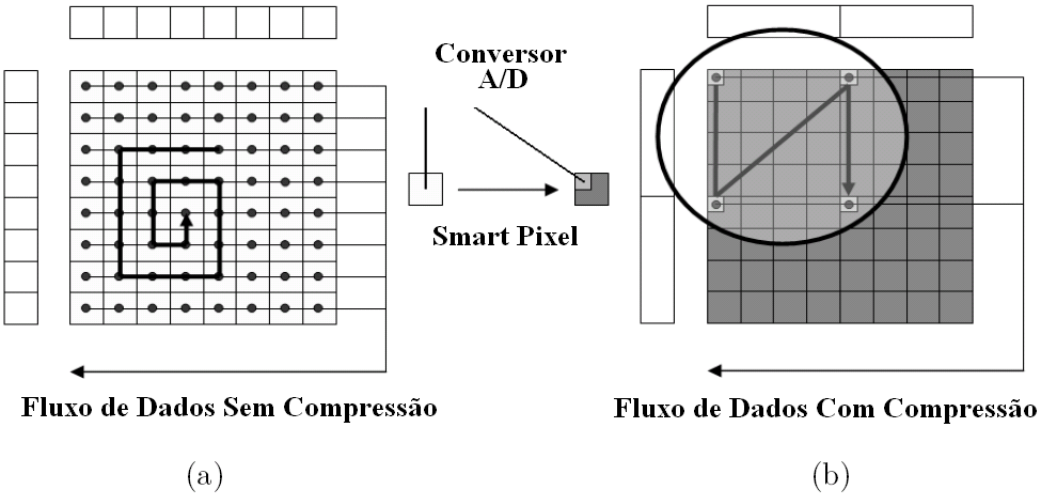


Figura 3.1: a) Sistema DPS; b)Codificação em blocos no plano focal

de fabricação [6]. Com isso podemos obter ao mesmo tempo alta resolução e boa qualidade de imagem.

A Figura 3.1(b) mostra a codificação em blocos, ao nível do *pixel*, no plano focal. Nós podemos interligar todos os *pixels* dentro de cada bloco de tamanho $p \times q$ (considerando uma imagem com resolução $P \times Q$), onde cada bloco pode ser representado por um código binário com B bits. Desta forma podemos implementar o VQ no plano focal. Este mapeamento apresenta muitas vantagens em termos de processamento de sinal, sendo a mais importante a eliminação dos conversores A/D, do *buffer* de memória e do algoritmo de compressão de imagem digital. Ao mesmo tempo, são necessárias a leitura e o armazenamento de apenas $(P \times Q \times B)/(p \times q)$ bits por imagem (Figura 3.1). O mapeamento também permite a redução por um fator pq em relação à complexidade de endereçamento, comparado com uso de quantizadores escalares usando um conversor A/D por *pixel*.

3.2 Sensores de Imagem CCD

Conforme mencionado na Seção 3.1, um dos principais tipos de sensores de imagem é o CCD (*Charged Coupled Device*). Os CCDs são encontrados nos dias atuais, não só em câmeras digitais, como também em equipamentos de fac-símile, copiadoras, *scanners* e celulares.

Os sensores CCD surgiram em 1970 e funcionam a partir da geração de carga elétrica proporcional à luz incidente, que é armazenada para leitura posterior [2]. Neste tipo de sensor, a leitura é feita de forma seqüencial (Figura 3.2), onde um conversor A/D lê a primeira fonte de cada linha. Logo em seguida, todas as cargas elétricas remanescentes em cada linha são deslocadas para para seu vizinho imediato. O processo continua até que todas as cargas sejam lidas e digitalizadas [17].

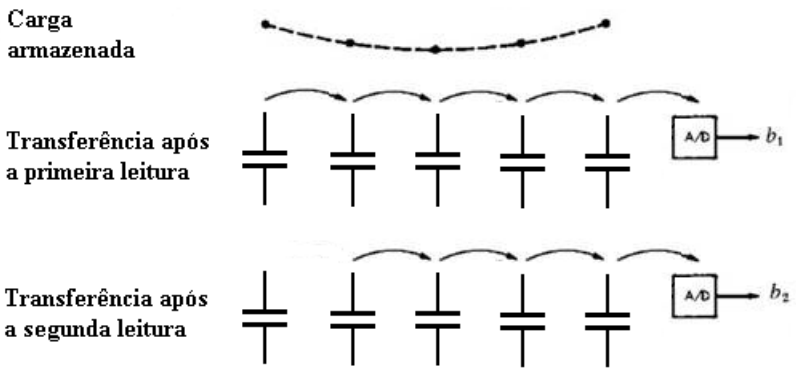


Figura 3.2: Processo de leitura no CCD

Como resumo, podemos apresentar as seguintes vantagens, que determinam a escolha da utilização de um sensor do tipo CCD:

- Produz pouco ruído na imagem;
- Possui boa sensibilidade;

- Captura imagens com alta qualidade;
- Permite o aproveitamento máximo da área de exposição.

Como desvantagens, podemos citar:

- Consumo elevado;
- Custo;
- Dificuldade de integração com outros circuitos;

3.3 Sensores de Imagem CMOS

Os sensores CMOS são formados também por elementos sensíveis à radiação incidente de luz. Porém, diferentemente dos CCDs, cada *pixel* é construído com fotodiodos e transistores. Uma das grandes vantagens dos sensores CMOS em relação aos CCDs é a possibilidade de se endereçar individualmente um *pixel* de interesse, para processamento [8].

Podemos classificar dois tipos de sensores CMOS:

- Sensor de *Pixel* Passivo - é composto basicamente por um fotodiodo e uma chave, que pode ser implementada através de transistores (Figura 3.3). Um único amplificador é utilizado para toda a matriz de sensores (Figura 3.4).
- Sensor de *Pixel* Ativo - apresenta como diferença básica em relação ao sensor de *pixel* passivo, a inclusão de um amplificador em cada célula da matriz de fotosensores (Figura 3.5).

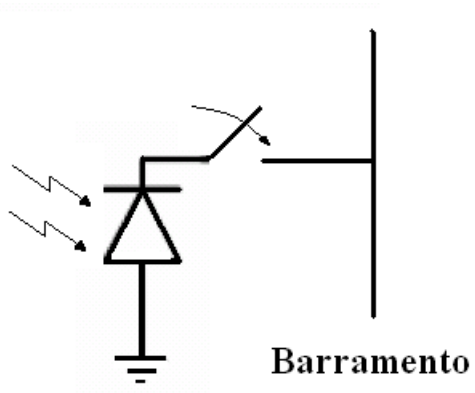


Figura 3.3: Estrutura do pixel PPS

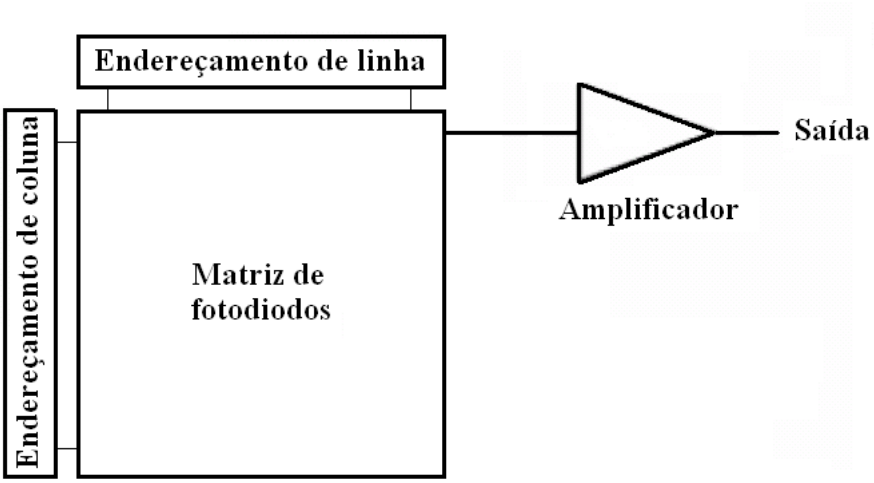


Figura 3.4: Arquitetura da matriz PPS

Dentre as diversas estruturas que implementam os sensores ativos, podemos destacar a célula 3T, por causa da sua implementação com poucos transistores. A estrutura básica de um *pixel* baseada em três transistores é mostrada na Figura 3.6.

Como resumo, para os sensores CMOS podemos citar as seguintes características:

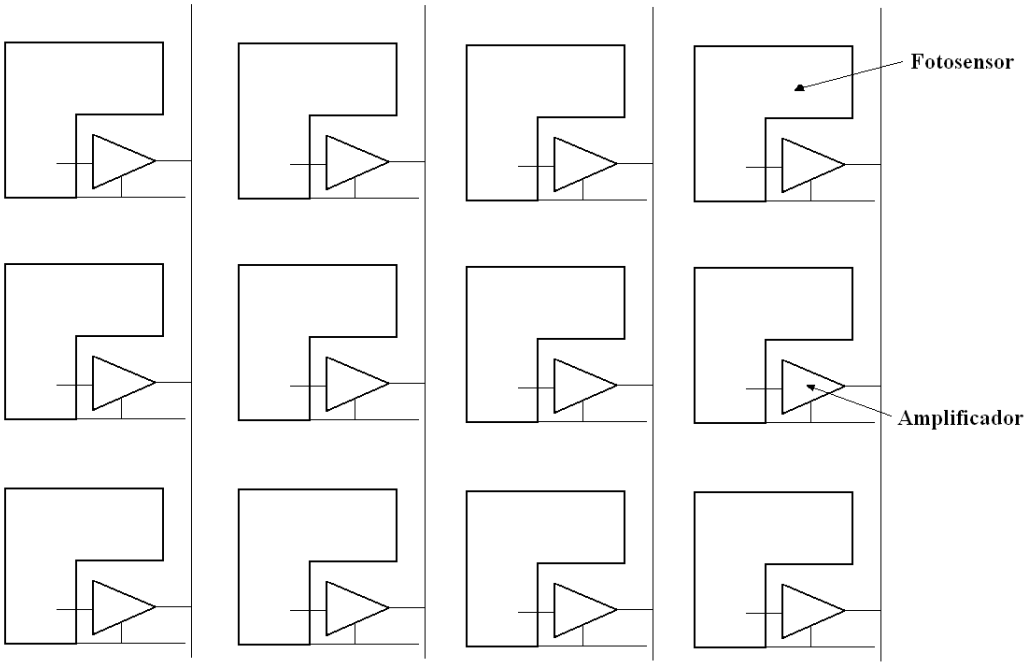


Figura 3.5: Esquema de um sensor APS

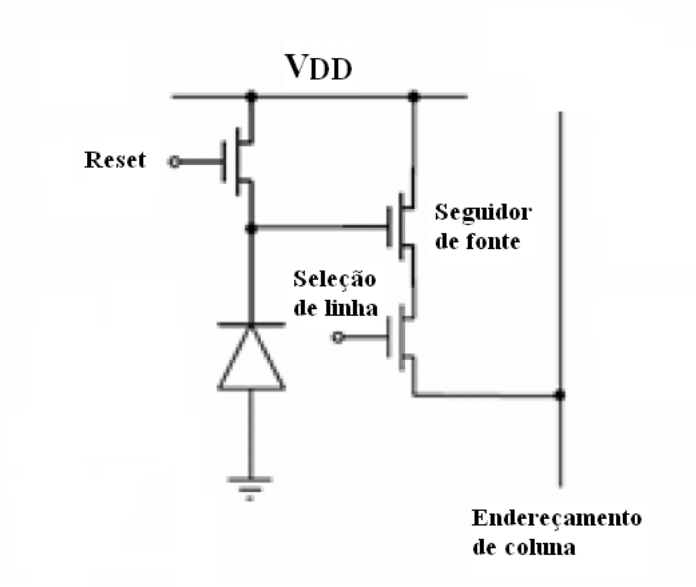


Figura 3.6: Esquema básico da célula 3T

- Consumo muito menor que os sensores do tipo CCD;
- Possuem tecnologia de fabricação semelhante a outros componentes CMOS;
- Existem atualmente diversos fabricantes desta tecnologia, diminuindo com isso os custos de fabricação;
- Permitem velocidades de leitura maiores, uma vez que os *pixels* podem ser lidos de forma independente, não sendo necessária a leitura do *frame* inteiro para se acessar o dado desejado;
- Têm como desvantagem o fato de apresentarem maior ruído na imagem que o CCD e, como são necessários outros circuitos que são integrados dentro da área da matriz dos *pixels*, a sensibilidade à luz também é reduzida.

Capítulo 4

Current Conveyors

4.1 Introdução

Sedra e Smith [29, 31, 32] introduziram o conceito de *current conveyor* como um dispositivo de 3 portas híbrido (corrente/tensão). O *current conveyor* é um dispositivo bastante versátil para aplicações em processamento analógico de sinais e foi projetado para ser usado no lugar do amplificador operacional.

Em projetos de circuitos analógicos, freqüentemente, há a necessidade de amplificadores com características voltadas para processamento de sinais em modo de corrente (*current-mode approach*). A abordagem de corrente considera que a informação é vista através do enfoque de correntes que variam no tempo, e propõe uma nova forma de implementação de circuitos integrados analógicos. Como vantagens deste tipo de abordagem podemos citar, em primeiro lugar, que ela não requer alto ganho de tensão, de forma que amplificadores de alta performance não são necessários. Também não necessita de elementos passivos de alta precisão, e seu projeto pode ser feito quase que inteiramente com transistores. Também demonstra

alta performance em termos de velocidade, banda-passante e precisão [10].

A primeira idéia apresentada pelos autores Sedra e Smith foi identificada inicialmente como *current conveyor* de primeira geração, ou simplesmente CCI.

O CCI funciona da seguinte forma (Figura 4.1): o valor de tensão que aparece no nó X tem o mesmo valor da tensão que é aplicada no nó Y. Com as correntes, por outro lado, o esquema é exatamente ao contrário, ou seja, a corrente que sai pelo nó Y tem o mesmo valor da corrente que entra pelo nó X. Temos também este mesmo valor de corrente saindo pelo nó Z.

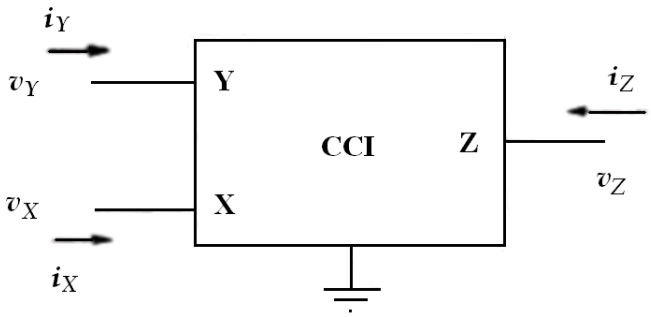


Figura 4.1: Representação em bloco do *current conveyor* (CCI)

Podemos ter a corrente i_Z no mesmo sentido ou no sentido oposto da corrente i_X. Ficou convencionado que, na representação matricial da Equação (4.1), o sinal positivo (+) significa que as correntes estão no mesmo sentido, enquanto que para correntes em sentidos opostos, o sinal é negativo (-). No primeiro caso temos um CCI 'positivo' (chamado CCI+), enquanto que para o outro caso temos um CCI 'negativo' (ou CCI-).

$$\begin{bmatrix} i_Y \\ v_X \\ i_Z \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & \pm 1 & 0 \end{bmatrix} \begin{bmatrix} v_Y \\ i_X \\ v_Z \end{bmatrix} \tag{4.1}$$

Os nós X e Y devem ter impedância baixa, enquanto o nó Z deve ter impedância alta. A Tabela 4.1 resume estas características.

Tabela 4.1: Características principais de um CCI

Nó do CCI	**Impedância**
X	baixa (idealmente 0)
Y	baixa (idealmente 0)
Z	alta (idealmente ∞)

Conforme podemos verificar, fica claramente explícito que a tensão no nó X não possui nenhuma relação com a corrente que flui por este mesmo nó. Do mesmo modo, a corrente no nó Y não depende da tensão que é aplicada neste nó.

Para melhor visualizarmos a relação entre as tensões e as correntes nos nós de um CCI, podemos representá-lo através de componentes denominados *nullator* e *norator* (Figura 4.2). O *nullator* é um dispositivo que é caracterizado por tensão zero e corrente zero. É usado para representar um curto-circuito virtual entre os nós (Figura 4.3). No caso do *norator*, a corrente que entra em um terminal é igual à corrente que sai pelo outro terminal (Figura 4.4).

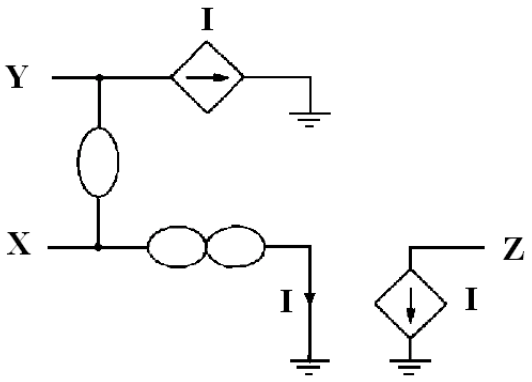


Figura 4.2: CCI representado com *nullator/norator*

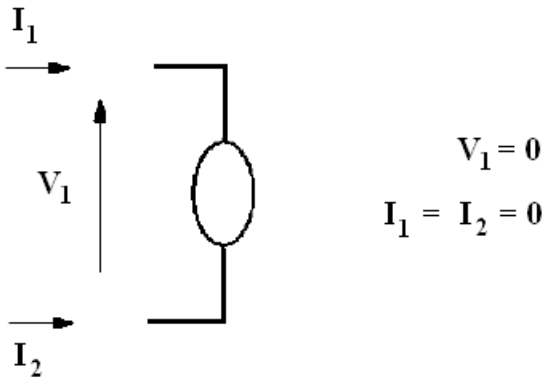


Figura 4.3: Representação do *nullator*

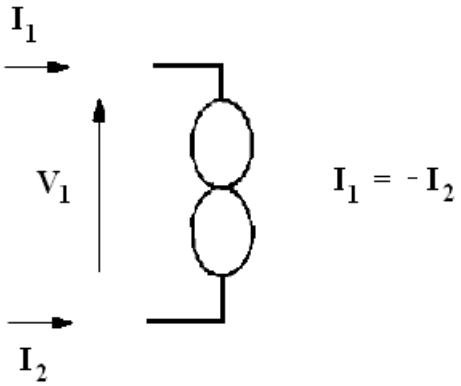


Figura 4.4: Representação do *norator*

Para aumentar a versatilidade do *current conveyor*, uma segunda versão foi criada alguns anos depois, onde nenhuma corrente flui pelo nó Y. Esta versão foi denominada *current conveyor* de segunda geração, ou, simplesmente CCII. Utilizando o mesmo diagrama em blocos da Figura 4.1, o funcionamento do CCII é descrito como na equação:

$$\begin{bmatrix} i_Y \\ v_X \\ i_Z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & \pm 1 & 0 \end{bmatrix} \begin{bmatrix} v_Y \\ i_X \\ v_Z \end{bmatrix} \tag{4.2}$$

Em um CCII, o terminal Y apresenta impedância infinita. Da mesma forma que no CCI, observamos que a tensão que aparece no nó X é a mesma do nó Y. Ainda, podemos verificar que a corrente no terminal X é 'copiada' para o terminal Z.

A Tabela 4.2 mostra as características principais do CCII, e a Figura 4.5 mostra o diagrama simplificado de um CCII representado através de dispositivos *nullator/norator*.

De forma semelhante ao CCI, com o CCII também podem ocorrer duas situações: o CCII+, que tem as correntes i_Z e i_X fluindo no mesmo sentido; e o CCII-, quando estas mesmas correntes estão em sentidos opostos.

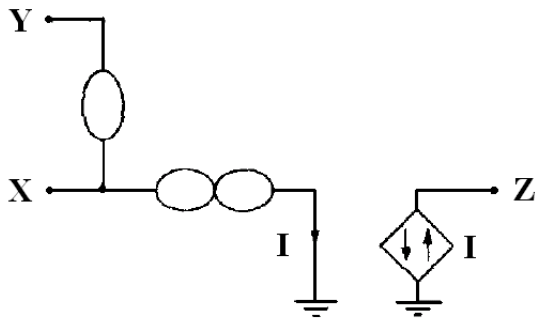


Figura 4.5: CCII representado com *nullator/norator*

4.2 Implementação do *Current Conveyor*

Um CCII pode ser visto como um transistor ideal (que pode ser bipolar ou MOS)[30]. Para compreendermos esta situação, tomemos como exemplo o transistor da Figura 4.6. Se o transistor for ideal, o V_{GS} é zero. Portanto, a mesma tensão que é aplicada no terminal G (*Gate*) deverá produzir uma outra tensão de mesmo valor no terminal S (*Source*). Devemos lembrar que a impedância de entrada no terminal G é alta (exatamente como no nó Y do CCII), enquanto que a impedância no terminal S é baixa (como no nó X do CCII). O valor da corrente no terminal D (*Drain*) deverá ter o mesmo valor da corrente que do terminal S. A impedância 'vista' pelo terminal D é alta (como no nó Z do CCII). Concluímos que o transistor ideal se comporta

Tabela 4.2: Características principais de um CCII

Nó do CCII	**Impedância**
X	baixa (idealmente 0)
Y	alta (idealmente ∞)
Z	alta (idealmente ∞)

como um CCII. Observando o transistor ideal através de sua representação com *nullator/norator*, esta comparação fica ainda mais evidente (Figura 4.7).

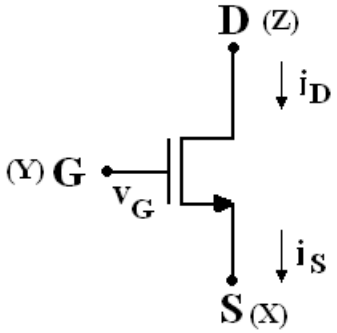


Figura 4.6: O transistor NMOS como um CCII

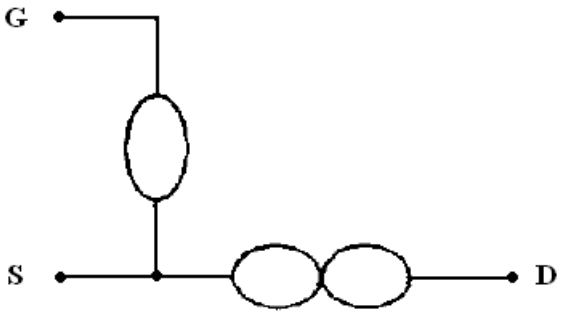


Figura 4.7: Representação do transistor com *nullator/norator*

A Figura 4.8 apresenta o diagrama esquemático, utilizado para simulação no SPICE, do *current conveyor* com um único transistor apresentado na Figura 4.6. A função do transistor M2 é apenas fornecer corrente para polarização do circuito, de forma a utilizarmos tanto correntes de entrada com valores positivos quanto correntes de entrada com valores negativos.

Esta implementação do *current conveyor*, apesar de simples, apresenta um incoveniente: a impedância de entrada tem valor elevado para as nossas necessidades.

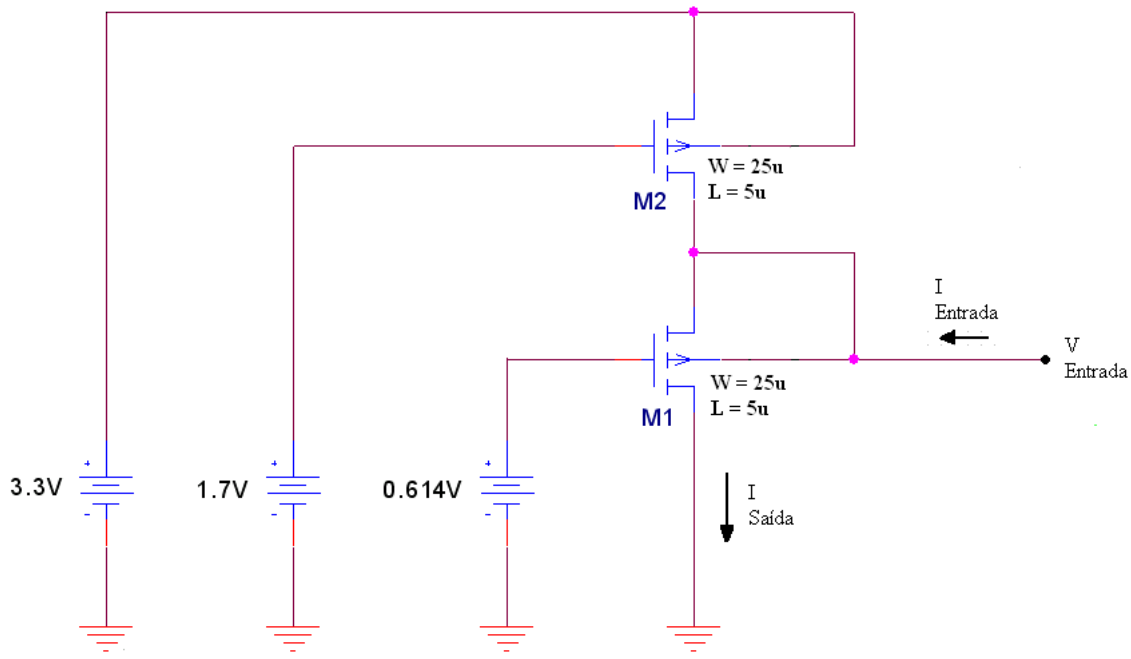


Figura 4.8: Diagrama esquemático para simulação no SPICE do *current conveyor* com transistor, apresentado na Figura 4.6

Observando a Figura 4.9, nota-se que a variação da corrente na entrada do circuito provoca uma variação na tensão de entrada do *current conveyor* bastante alta. A impedância de entrada é de cerca de 6,5 $K\Omega$.

Uma outra idéia para a implementação de um *current conveyor* é mostrada na Figura 4.10. Devido à realimentação negativa, este *current conveyor* apresenta uma impedância em seu nó de entrada (X) mais baixa que o anterior[1]. Em circuitos utilizando transistores não ideais, esta característica é bastante desejada, uma vez que necessitamos manter a tensão no nó X constante e independente das variações de corrente neste mesmo nó.

Da mesma forma que no *current conveyor* padrão, a corrente I_Z é igual à corrente I_X. Entretanto, a tensão V_X é uma função da corrente I_Y (Equação (4.3)),

[1]Esta impedância fica reduzida a apenas algumas dezenas de ohms.

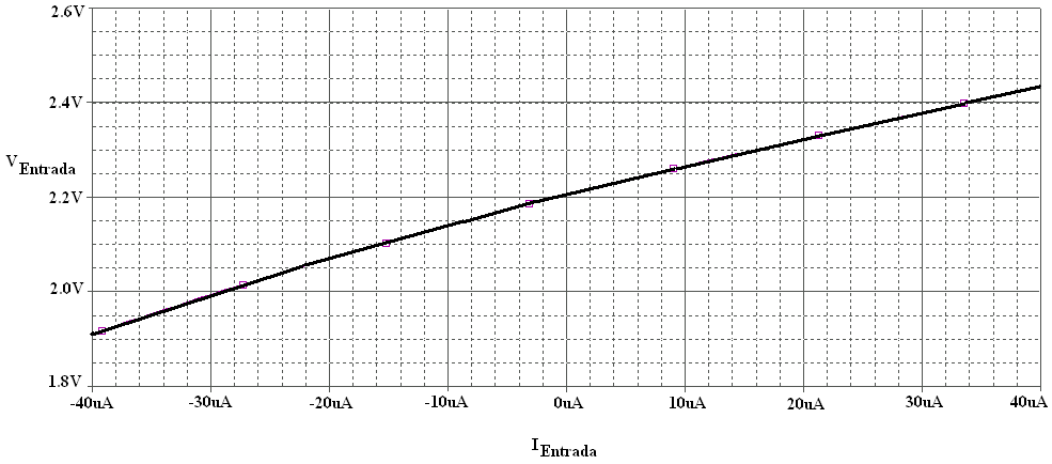


Figura 4.9: Simulação com o Spice: relação V-I na entrada do *current conveyor* da Figura 4.8

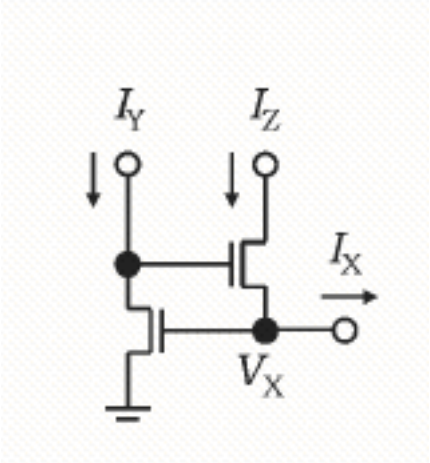


Figura 4.10: *Current conveyor* utilizando dois transistores

e por este motivo ele é denominado *current conveyor* controlado por corrente.

$$\begin{cases} I_Z = I_X \\ V_X = f(I_Y) \end{cases} \tag{4.3}$$

A Figura 4.11 mostra o diagrama completo do nosso *current conveyor* uti-

lizado para simulações com o Spice. Da mesma forma que o *conveyor* anterior, o transistor M2 oferece corrente para a polarização do transistor M3, sendo o transistor M1 responsável pela polarização adequada do transistor M4.

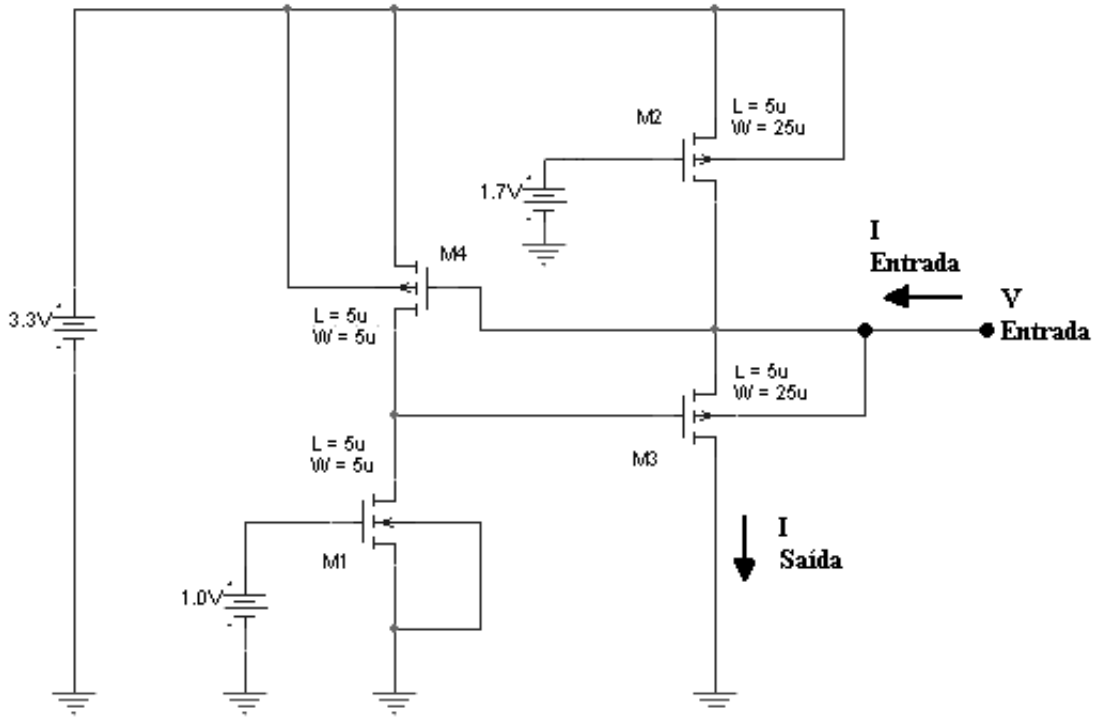


Figura 4.11: Diagrama esquemático do *current conveyor* completo

Fazendo a corrente de entrada variar entre -40 μA e +40 μA, observamos na Figura 4.12 que a variação de tensão é de aproximadamente 3,3 mV, o que nos leva a concluir que a impedância de entrada do circuito é 41,3 Ω.

Tendo chegado a este resultado, passaremos então ao projeto das sinapses que efetuarão o nosso produto interno. Antes porém, é conveniente apreciarmos alguns ensaios a respeito da sensibilidade dos MLPs e ECVQs à introdução de erros aleatórios.

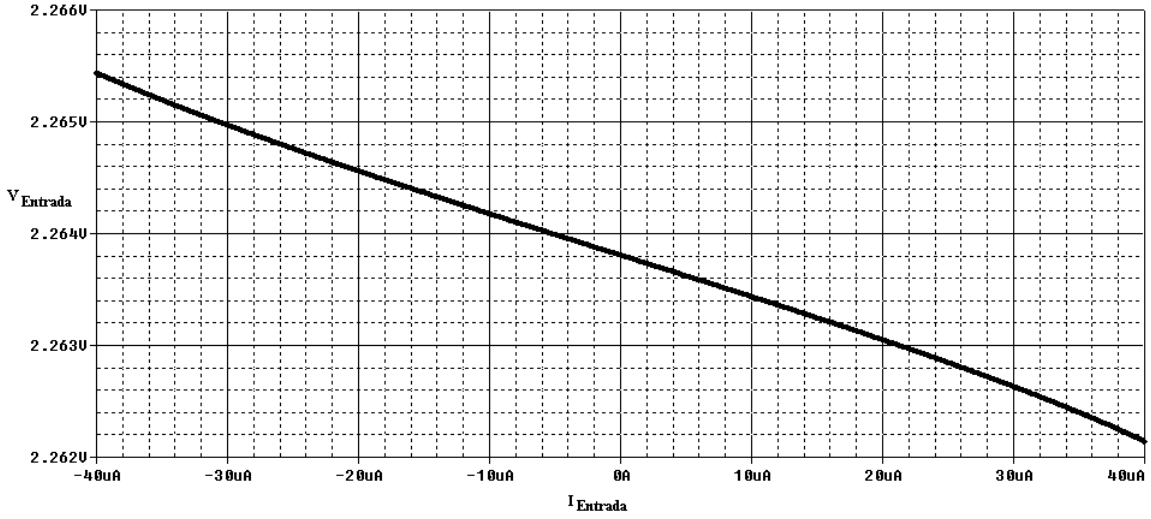


Figura 4.12: Simulação com o Spice: impedância de entrada do *current conveyor* da Figura 4.11

Capítulo 5

Estudo da Sensibilidade de MLPs Aplicados à Compressão de Imagens no Plano Focal

Uma das maiores preocupações na construção de um VQ utilizando *perceptrons* de um MLP é a relação entre o desempenho em termos de taxa e distorção[1] do algoritmo de compressão, e as restrições de complexidade[2] impostas pelo tamanho do *pixel* e pela área do sensor.

Tomando por base que o processo de fabricação CMOS introduz erros no algoritmo de compressão, é necessário levar em consideração a sua sensibilidade a estas variações. A sensibilidade dos MLPs aos erros de realização de seus coeficientes já foi estudada em [34], para o caso de funções de ativação contínuas. Para o caso

[1] A distorção é dada pelo erro médio quadrático entre os vetores $\mathbf{x}(n)$ e os vetores reconstruídos $\hat{\mathbf{x}}(n)$. É definida por $D = \frac{1}{N} \sum_n \|\mathbf{x}(n) - \hat{\mathbf{x}}(n)\|^2$.

[2] Definimos a complexidade de um sistema de processamento no plano focal através da contagem do número transistores utilizados em sua implementação.

de funções de ativação não-contínuas, abordagens estatísticas utilizando simulações de Monte Carlo tornam-se mais viáveis e foram usadas anteriormente no estudo da sensibilidade aos erros de implementação em MLPs usados em classificadores, mas não em algoritmos de compressão.

Neste capítulo serão descritas algumas simulações estatísticas realizadas para comparar a sensibilidade de MLPs e quantizadores vetoriais restritos em entropia[3] (ECVQs) aos erros de implementação do algoritmo de compressão de imagens [16]. O foco principal será o esquema de VQ baseado em MLPs ou ECVQs, que comprime os vetores $\mathbf{x}(n)$ gerando uma seqüencia aleatória com entropia[4] H, onde um decodificador deverá ser capaz de reconstruir os dados originais a partir das estimativas $\hat{\mathbf{x}}(n)$, com uma dada distorção D. Relações entre a sensibilidade dos pares (H, D) e a complexidade, tanto para MLPs quanto para ECVQs, obtidas através de simulações de Monte Carlo, permitem distinguir os projetos viáveis de codificadores analógicos no plano focal. Tanto no lado do codificador como no lado do decodificador serão utilizadas funções descontínuas.

[3] A restrição de entropia possibilita a redução da taxa de dados, mantendo-se aproximadamente fixa a distorção. Esta restrição foi proposta em [7] pela primeira vez. O método de ECVQ baseia-se em uma extensão do algoritmo clássico LBG. O seu excelente desempenho em taxa e distorção é conseguido ao custo de uma complexidade muito elevada.

[4] A definição de entropia é dada pela equação $H = -\sum_k p_k \log_2 p_k$, onde p_k é a probabilidade de ocorrência do centróide k.

5.1 Modelos para Análise de Sensibilidade

5.1.1 MLP com Função de Ativação Tangente Hiperbólica

Nesta seção apresentamos na Figura 5.1, como exemplo ilustrativo, um MLP que utiliza como função de ativação a tangente hiperbólica (HT-MLP). Este HT-MLP irá mapear vetores de dados com quatro dimensões, $\mathbf{x}(n)$, para palavras binárias $\mathbf{b}(n)$ de comprimento igual a 8.

Cada componente do vetor de entrada $\mathbf{x}(n) = [x_1(n)\ x_2(n)\ x_3(n)\ x_4(n)]^T$ da Figura 5.1 é multiplicado por um fator de ganho, e um termo de polarização (*bias*) é somado ao resultado dessa multiplicação. O *Kernel PCA* [28] é feito através de uma multiplicação matriz-vetor aplicada às saídas da função de ativação do tipo tangente hiperbólica. Logo em seguida, o vetor de *bias* é somado com esse resultado , gerando o vetor $\mathbf{f}(n) = [f_1(n)\ f_2(n)\ f_3(n)\ f_4(n)]^T$.

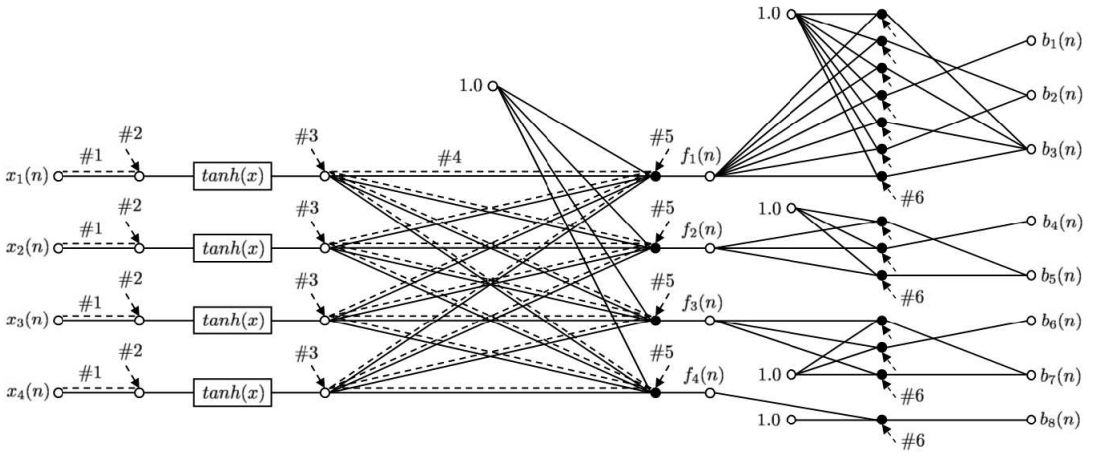


Figura 5.1: Estrutura típica de um HT-MLP usado para a compressão dos vetores $\mathbf{x}(n)$. As linhas pontilhadas representam coeficientes que são afetados por erros de implementação

O espaço dos vetores $\mathbf{f}(n)$ (também chamados de vetores de "características"

ou *features*), ao invés de ser particionado através de linhas retas, é particionado através de curvas arbitrárias, conforme desejado para a implementação de um VQ. Cada componente de $\mathbf{f}(n)$ é quantizada por um SQ com resolução baixa. Os escalares $f_1(n)$, $f_2(n)$, $f_3(n)$ e $f_4(n)$ são codificados com resoluções de 3, 2, 2 e 1 bit, respectivamente, realizando então um VQ de 8 bits. Os SQs são implementados pela subtração entre a variável de entrada e os limiares constantes que definem os intervalos de quantização. A subtração é seguida por comparadores que definem códigos binários com tamanhos iguais a 7, 3, 3 e 1. Tais códigos binários são convertidos para os códigos de saída com tamanhos 3, 2, 2 e 1, realizando-se multiplicações matriz-vetor, em que os vetores constantes têm valores discretos, como em um conversor A/D do tipo *flash*.

Os ganhos das funções do tipo tangente hiperbólica, os termos de *bias* das funções tangente hiperbólica, a matriz de *Kernel PCA* e o vetor de *bias* de *Kernel PCA* são calculados durante o projeto do MLP. Os limiares de quantização foram obtidos através de métodos de otimização não-lineares, com o objetivo de minimizar a função custo $J = D + \lambda H$ (onde λ é um multiplicador de Lagrange que controla o compromisso entre a minimização de taxa e a minimização de distorção [14]). A codificação de entropia é feita fora do bloco de *pixels*, de forma que a saída binária $\mathbf{b}(n)$ do MLP tem comprimento fixo.

Os vetores $\mathbf{x}(n)$ usados para o teste do *perceptron*, foram obtidos a partir de um conjunto com sete imagens para teste. Os vetores de teste foram comprimidos em palavras binárias, de índices $\mathbf{b}(n)$, com 8 bits. Estima-se a entropia H a partir do histograma dos índices binários da seqüencia $\mathbf{b}(n)$. Os vetores reconstruídos $\hat{\mathbf{x}}(n)$ foram gerados pelo decodificador, através de uma seleção do dicionário de códigos

do VQ, produzido a partir de um conjunto de imagens de treino. O erro médio quadrático D entre $\mathbf{x}(n)$ e $\hat{\mathbf{x}}(n)$ é calculado (com n variando de 1 até N, onde N é o número de blocos de *pixels* das imagens de teste).

Codificadores MLP com diversas propriedades de taxa e de distorção foram projetados com λ variando de 10^{-4} a 10^{-1}. Por exemplo, nas curvas do conjunto de teste $\mathcal{X}_T$ da Figura 5.5, o MLP com máxima SQNR (*Signal to Quantization Noise Ratio*) [16] é um HT-MLP com H = 4,4 bits por vetor. A sua distorção D é tal que $10\log_{10}(D_0/D)$ = 9,0 dB. Foi utilizado $\lambda = 1{,}2\times10^{-4}$, assumindo-se ausência de erros de implementação. Outros MLPs, baseados em outros tipos de funções de ativação, também foram projetados aplicando-se algumas mudanças nos passos do projeto do HT-MLP. Os parâmetros dos MLPs são pré-calculados e mantidos constantes durante as simulações numéricas da sensibilidade. A única modificação será a inclusão das perturbações aleatórias.

5.1.2 VQ com Restrição de Entropia

A estrutura típica de um ECVQ que pode ser utilizado para codificação em blocos, é mostrada no exemplo da Figura 5.2. Este ECVQ associa palavras binárias de comprimento variável $\mathbf{b}(n) = [b_1(n)\ b_2(n)\ ...\ b_L(n)]^T$, com comprimentos até $L = 15$ neste exemplo, aos vetores de dados $\mathbf{x}(n)$ com quatro componentes.

Neste ECVQ, K produtos internos de tamanho 5 são calculados entre o vetor de entrada $[||\mathbf{x}(n)||^2\ \ \mathbf{x}(n)]$ e os vetores constantes $[1\ \ -2\mathbf{c}_k]$, $k = 1, \ldots, K$. Adiciona-se então, um termo constante de *bias* ao resultado de cada produto. Com estas operações, obtém-se os custos $j_k(n) = d_k(n) + \lambda l_k$ de usar cada um dos vetores de código $\mathbf{c}_k$ (do dicionário de códigos) como representação para $\mathbf{x}(n)$. O termo d_k é a

distância Euclideana entre $\mathbf{x}(n)$ e $\mathbf{c}_k$, e l_k é o comprimento da palavra binária que representa $\mathbf{c}_k$ no canal de comunicação.

Para o cálculo de $||\mathbf{x}(n)||^2$, são utilizados quatro circuitos para a realização das funções quadráticas e um CCII (*second generation current conveyor*). Para a geração dos $j_k(n)$, K CCIIs são necessários.

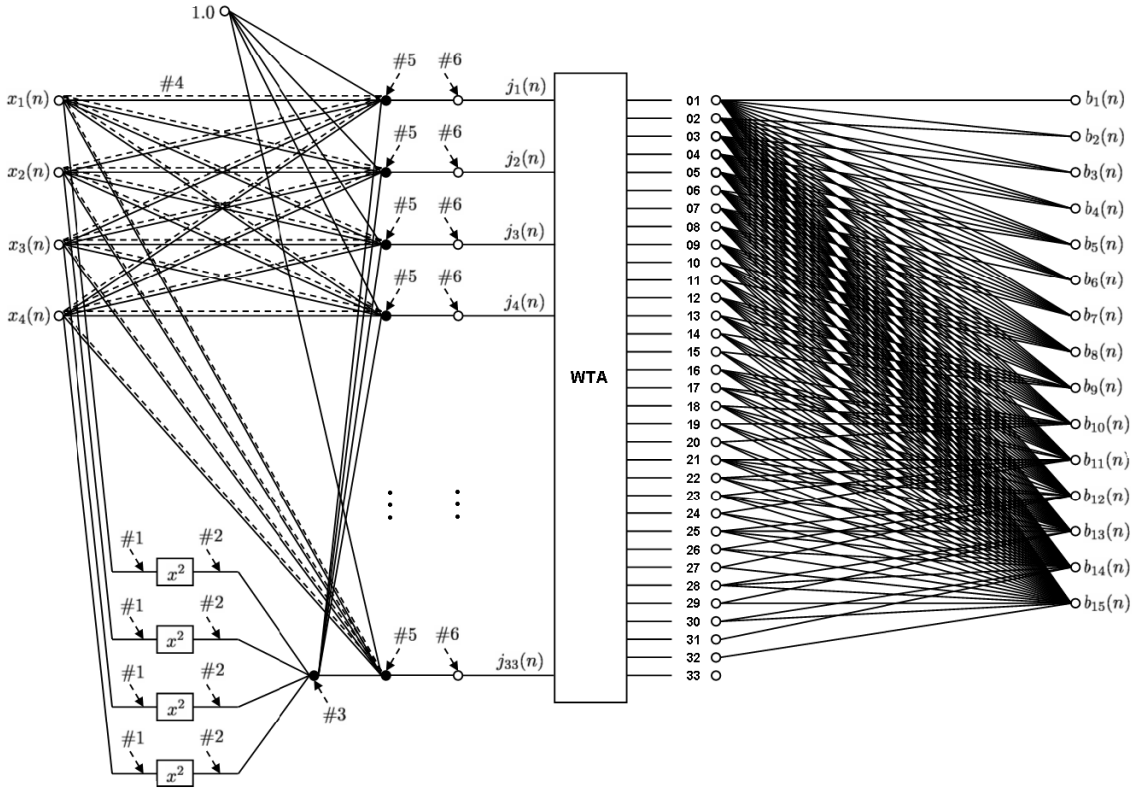


Figura 5.2: Estrutura típica de um ECVQ usado para a compressão dos vetores $\mathbf{x}(n)$

Cada $j_k(n)$ é aplicado a um circuito WTA (*winner-takes-all*). O menor valor então é selecionado e o circuito WTA gera um vetor binário $\mathbf{e}_k(n)$ que é processado por uma matriz de conexões de ganhos iguais 1, para gerar o vetor de saída binário $\mathbf{b}(n)$, cujo comprimento é variável. As componentes de $\mathbf{e}_k(n)$ são iguais a zero,

exceto aquela que se encontra na posição correspondente ao menor valor de $j_k(n)$, onde o seu valor é 1.

Utilizando o algoritmo proposto em [7] para o projeto de ECVQs, o dicionário que contém os vetores $\mathbf{c}_k$, e os comprimentos l_k, foram otimizados conjuntamente de forma a minimizar $J = D+\lambda H$. Para testar o ECVQ, os vetores $\mathbf{x}(n)$ foram determinados a partir do mesmo conjunto de testes utilizado no HT-MLP, comprimindo-os em uma seqüencia de vetores binários $\mathbf{b}(n)$. Vários ECVQs foram projetados com λ entre 10^{-4} e 10^{-1}, com H's e D's calculados da mesma forma utilizada pelo modelo anterior (HT-MLP).

Pode-se citar, como exemplo, o ECVQ com a segunda menor SQNR nas curvas do conjunto de teste $\mathcal{X}_T$ da Figura 5.5 que tem $H = 0{,}9$ bits por vetor. A sua distorção D é tal que $10\log_{10}(D_0/D) = 5{,}9$ dB. O projeto foi feito com $\lambda = 0{,}06$, assumindo a ausência de erros de implementação. O dicionário tem tamanho $K = 33$ e o índice binário mais longo tem comprimento $L = 15$. Assim como no caso do HT-MLP, os parâmetros dos ECVQs são pré-calculados e mantidos constantes durante as simulações numéricas de sensibilidade. Isso significa que não houve treinamento após as perturbações terem sido aplicadas aos parâmetros de cada ECVQ.

5.1.3 Implementação das Funções

A função tangente hiperbólica pode ser sintetizada a partir de um par diferencial com transistores CMOS (conforme apresentado em [26]). O circuito foi modelado como um bloco não-ideal, contendo a função tangente hiperbólica ideal, um erro de *offset* de entrada aditivo, um erro de *offset* de saída aditivo e, um erro de ganho.

Um CCII soma, em modo de corrente, o conjunto dos resultados das multiplicações efetuadas pelas sinapses (implementadas através um único transistor CMOS [24]), realizando assim a operação do produto interno.

Concentrados nos círculos pretos das Figuras 5.1 e 5.2, os circuitos dos CCIIs foram modelados como blocos não ideais contendo um CCII ideal, erros de *offset* na leitura da entrada, erros de *offset* na realização dos termos de *bias*, erros de *offset* na leitura dos termos de *bias*, erros de *offset* de saída, e erros das sinapses. Estes últimos foram divididos em dois tipos:

- erro do fator multiplicativo, análogo ao erro de ganho da tangente hiperbólica;
- erro de *offset* na saída da sinapse.

A função de comparação também foi apresentada utilizando-se um par diferencial de transistores CMOS [26], operando como comparador. O circuito do comparador foi modelado como uma função sinal, ideal, com erros de *offset* somados ao valor do limiar e à leitura da entrada.

Para implementação da função quadrática (Figura 5.2), utiliza-se o circuito de três transistores proposto por [3]. O modelo foi formado pela função quadrática ideal, um erro de *offset* aditivo na entrada, e um erro de *offset* aditivo na saída.

Os CCIIs são necessários em três casos:

- Para somar os resultados dos blocos com funções quadráticas. Como todos os ganhos são unitários, não há produto interno envolvido, e estes CCIIs foram modelados pelos seus erros de *offset* na leitura das entradas e pelos seus erros de *offset* na saída;

- Para implementar os produtos internos dentro da multiplicação matriz-vetor que calcula $j_1(n), j_2(n), \ldots, j_{33}(n)$. Os CCIIs e suas conexões sinápticas foram foram modeladas como mencionado nos parágrafos anteriores;

- Para somar os resultados binários da operação WTA (*winner-takes-all*) na saída do codificador de comprimento variável. Por envolverem dados exclusivamente digitais, estas adições foram consideradas ideais.

Para a função WTA, temos conexões paralelas de CCIIs básicos utilizando dois transistores [1]. O modelo não-ideal considera apenas o erro de *offset* na leitura de cada entrada $j_1(n), j_2(n), \ldots, j_{33}(n)$, por causa da simetria desta operação em relação às suas entradas. Não há erro de *offset* na saída.

5.2 Resultado das Simulações

Para as simulações de Monte Carlo, os mesmos valores de erro são utilizados para todos os blocos de *pixels*. Isto porque os efeitos de gradiente do processo de fabricação criam uma correlação grande entre os erros de implementação que ocorrem em blocos vizinhos. Os erros foram modelados então como variáveis aleatórias com distribuição normal com média 0 e desvio padrão σ_p. O desvio padrão representa a precisão que pode ser alcançada com as técnicas do projeto analógico e com o processo de fabricação considerados. Para nosso caso usamos $\sigma_p = 2^{-E}$, onde E representa, em bits, a precisão equivalente obtida por circuitos analógicos em um dado processo de fabricação, e assume valores $E = 6$, 8, 10, 12, 14, e 16 bits.

Para cada MLP e para cada ECVQ foram realizadas 50 iterações, para as quais os respectivos valores médios de H e D foram armazenados.

Foram testados ao todo 60 MLPs e 17 ECVQs com diferentes valores de λ. Além das funções de ativação do tipo tangente hiperbólica, outras funções também foram consideradas, como as funções de ativação de base radial, polinomiais e lineares. Maiores detalhes estão disponíveis em trabalho anterior [14], incluindo todos os modelos de perturbação para entradas, dicionários e saídas de tamanho genéricos .

As Figuras 5.3 e 5.5 mostram as curvas de desempenho, em taxa e distorção (H, D) ideais (pontilhadas), que foram obtidas através da avaliação dos 77 sistemas sobre o conjunto de treino $\mathcal{X}_R$, para o qual o dicionário havia sido projetado, e em seguida sobre o conjunto de teste $\mathcal{X}_T$, usando o mesmo dicionário calculado a partir de $\mathcal{X}_R$, sem a aplicação de perturbação alguma aos parâmetros dos codificadores.

5.2.1 Perda de Qualidade Limitada

Definimos a perda de qualidade média do decodificador (para codificadores MLP ou ECVQ) como sendo $\Delta S(E) = S - \bar{S}(E)$, onde $\bar{S}(E)$ é a SQNR média do par codificador/decodificador depois de 50 realizações em um experimento de Monte Carlo. A entropia e SQNR ideais do codificador sem erros de implementação são representadas como H e S, respectivamente.

Definindo P como sendo a perda de qualidade máxima desejada para um codificador, $E_{\Delta S<P}$ é encontrado de forma que $\Delta S(E) < P$, para esse codificador. Consideramos que, se um par codificador/decodificador apresenta $E_{\Delta S<P} < B$ bits sobre o conjunto de teste $\mathcal{X}_T$, então a sensibilidade deste par é suficientemente baixa para permitir uma implementação analógica robusta com precisão equivalente a B bits.

Conforme apresentado na Figura 5.3(a), observamos os pares (H, S) calcu-

lados para MLPs (curvas pontilhadas inferiores) e ECVQs (curvas pontilhadas superiores), assumindo codificadores com parâmetros ideais, isto é, sem perturbação aplicada. Os sistemas para os quais $E_{\Delta S<0,3} < 8$ bits são mostrados com círculos. Com 8 bits de precisão, conclui-se que a maioria das MLPs pode ser realizada com perda menor que $P = 0{,}3$ dB na SQNR, enquanto que as realizações dos ECVQs sofrem uma degradação maior. Na Figura 5.3(b) temos os resultados para B também igual a 8 bits, mas com $P = 3$ dB.

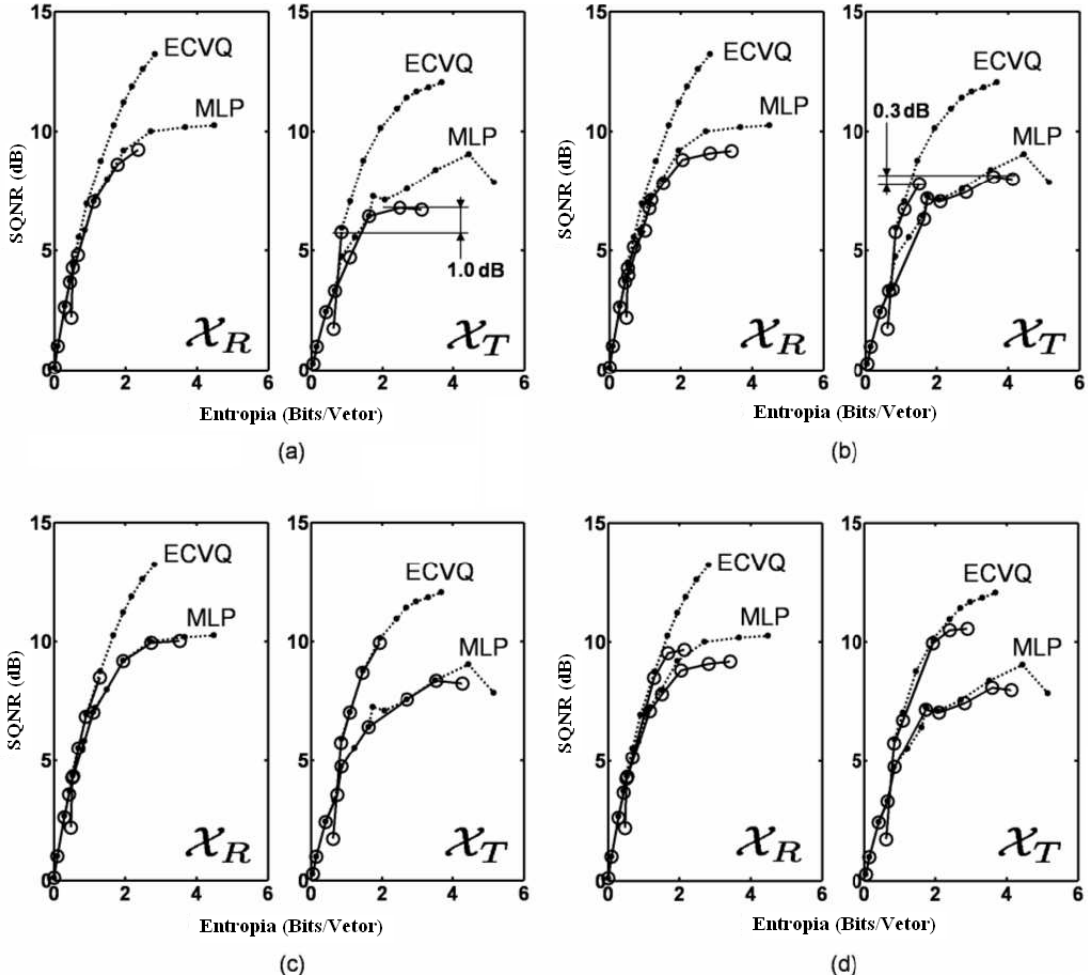


Figura 5.3: Perda de qualidade limitada

Temos também na Figura 5.3(c) os resultados das simulações para $P = 0{,}3$

dB, e na Figura 5.3(d), os resultados das simulações para $P = 3$ dB. Ambos com precisão equivalente a $B = 10$ bits. Deduz-se daí que para implementações com precisão de até 8 bits os MLPs são mais robustos do que os ECVQs em uma faixa bastante ampla de pontos (H, S). Já para precisões maiores (10 bits ou mais), os ECVQs têm desempenho melhor.

5.2.2 Sensibilidade × Complexidade

Naturalmente, por terem suas estruturas fixas durante o treinamento, os MLPs têm complexidade limitada. No treinamento, os métodos de otimização resultam em uma solução com J sendo um mínimo local num conjunto de codificadores com estrutura fixa. Já o ECVQ permite minimização local de J sem restrições de complexidade[5]. Como os MLPs têm menos parâmetros para sofrer erros na implementação, é razoável esperar que eles sejam mais robustos para realização em com circuitos analógicos, como indicado nas Figuras 5.3(a) e 5.3(d).

A tolerância percentual máxima (ξ) [16] dos coeficientes é apresentada na Figura 5.4(a) em função da complexidade θ, para sistemas que possuem $\Delta S < 0{,}5$ dB com precisão de implementação equivalente a 10 bits. Nesta figura são usados os vetores do conjunto de teste $\mathcal{X}_T$. Dentre os MLPs, 11 sistemas têm $\xi = 4{,}7$ (6 bits), porque eles alcançam $P < 0{,}5$ dB com tolerância maior do que 4,7%. Nem todos os 11 sistemas têm os mesmos requerimentos de tolerância, o que poderia ser confirmado por uma simulação com $E = 4$. Entretanto, esta simulação não foi feita porque $E = 6$ é suficiente para representar a precisão dos processos CMOS atuais.

As retas da Figura 5.4(a) foram produzidas de acordo com a Equação (5.1),

[5]Para uma mesma taxa mínima H, um ECVQ alcança SQNR igual ou maior do que a do MLP.

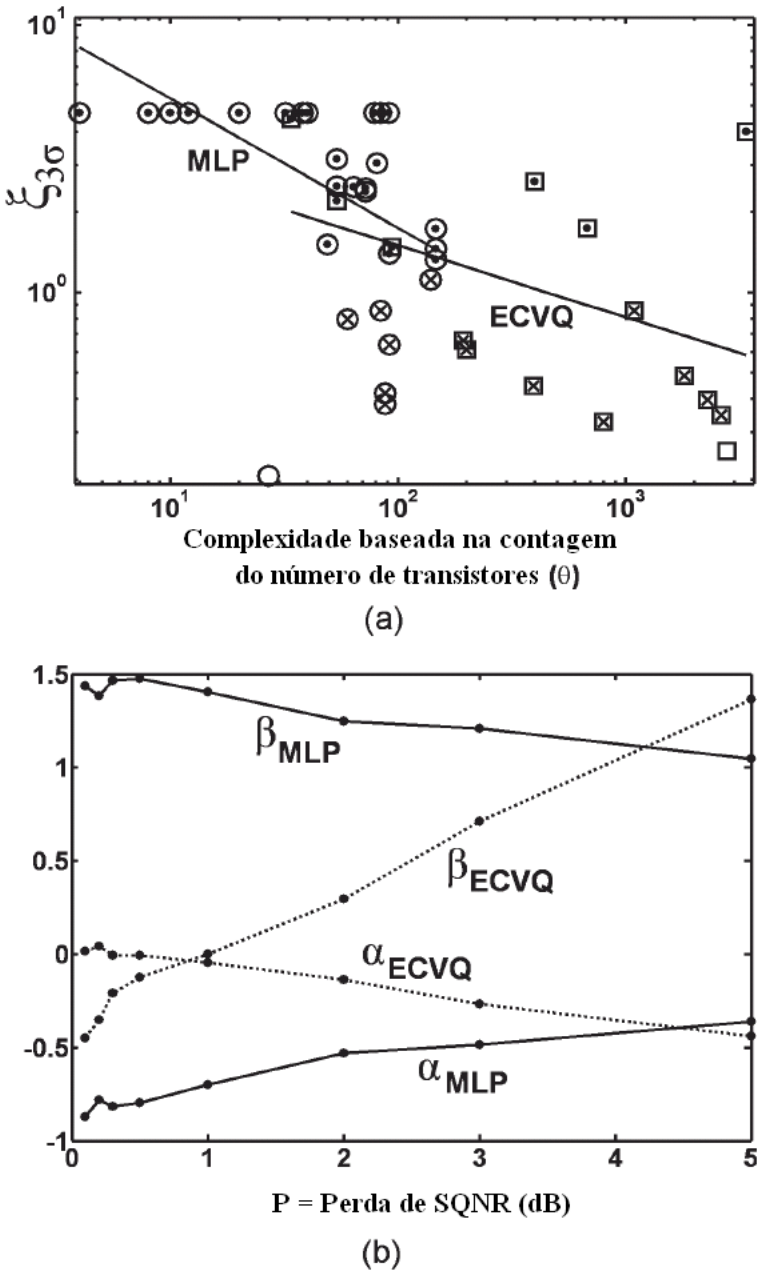


Figura 5.4: (a) Tolerância × complexidade: $E \leq 8$ (pontos), $8 < E \leq 10$ (cruzes), MLPs (círculos), e ECVQs (quadrados); (b) funções $\alpha(P)$ e $\beta(P)$

que mostra a relação entre ξ e θ, desenvolvida por ajuste de mínimos quadrados entre $\log(\xi)$ e $\log(\theta)$.

$$\log(\xi) = \alpha(P)\log(\theta) + \beta(P) \tag{5.1}$$

As funções $\alpha_{MLP}(P)$, $\alpha_{ECVQ}(P)$, $\beta_{MLP}(P)$ e $\beta_{ECVQ}(P)$, mostradas na Figura 5.4(b) foram obtidas variando-se a perda máxima aceitável de SQNR (P) de 0,1 dB até 5,0 dB.

Para um dado tipo de fabricação em *hardware*, é desejável a escolha de sistemas que possuam ξ fracamente dependente de θ (ou seja, $|\alpha|$ deveria ser pequeno) ou então sistemas que tenham *offset* de tolerância β alto (o que significa que a sensibilidade é baixa para configurações simples).

Os MLPs se tornam menos sensíveis à medida em que a perda máxima é aumentada, embora não possuam um $|\alpha|$ baixo, especialmente para θ baixo. Além disso, os MLPs apresentam β alto para qualquer P.

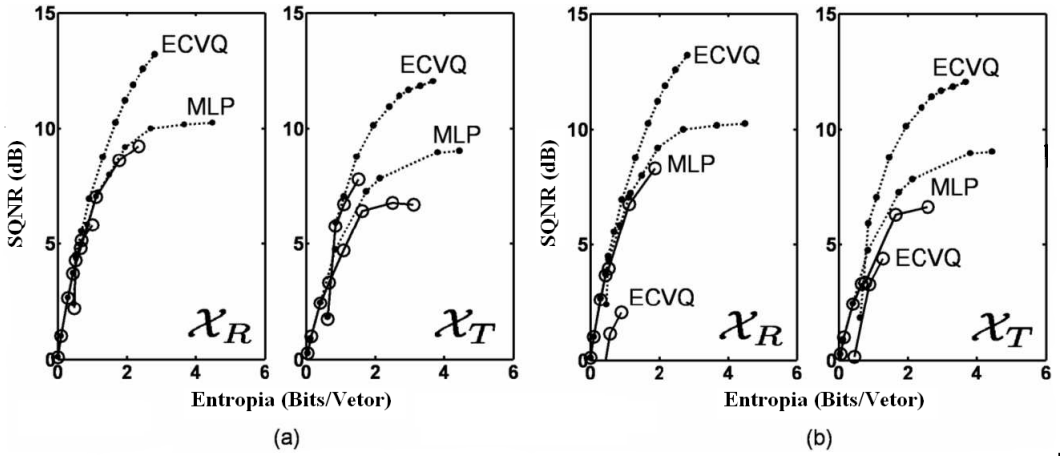


Figura 5.5: Perda livre com precisão E igual a (a) 8 bits e (b) 6 bits

5.2.3 Perda de Qualidade Livre

Para este caso, foram considerados todos os codificadores que haviam sido simulados com precisão igual a 6 ou 8 bits, sem limitar a sua perda de SQNR. Para precisão $E = 8$ bits, a Figura 5.5(a) mostra que os MLPs alcançam SQNR menor, enquanto que sua complexidade é inferior à dos ECVQs. Já com 6 bits, os MLPs são consistentemente melhores em ambas as situações (Figura 5.5(b)).

Todos os resultados apresentados aqui nesta seção poderiam ser previstos a partir das seguintes observações:

- Como os MLPs têm complexidade mais baixa, têm portanto menos parâmetros a serem perturbados pelos erros de implementação (Figuras 5.3 e 5.5);
- A otimização dos MLPs é restrita pelos limites estabelecidos para a complexidade, fazendo com que o valor mínimo J seja menos condicionado aos parâmetros obtidos a partir do processo de otimização (Figura 5.4).

Já que não é possível calcular de forma simples, nestes problemas, as derivadas do custo Lagrangeano com relação aos parâmetros, fica como sugestão um tratamento puramente teórico do compromisso entre sensibilidade e complexidade.

Capítulo 6

Implementação do Produto Interno

Neste capítulo, será proposto um circuito para o cálculo de produtos internos entre vetores com componentes analógicas. Neste circuito, usa-se somente um transistor por multiplicação. Serão apresentados resultados para a compressão de imagens usando MLPs implementados a partir do circuito proposto. Estes resultados são baseados em simulações elétricas no Spice incluindo análise de Monte Carlo.

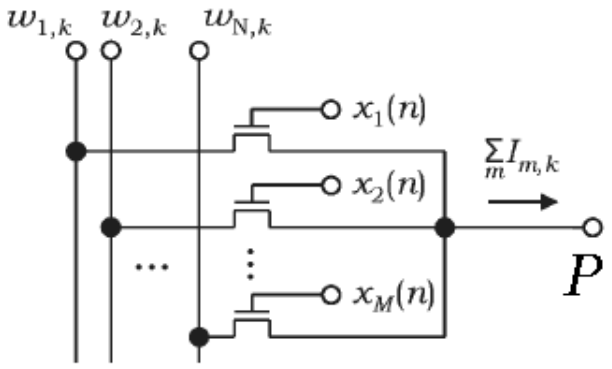


Figura 6.1: Produto interno utilizando transistores MOS

Sendo $\mathbf{x}(n)$ um vetor de variáveis e $\mathbf{w}_k$ o vetor de constantes (correspondentes aos pesos sinápticos do MLP), ambos com comprimento M, então o produto interno $\mathbf{x}(n)^T\mathbf{w}_k$ pode ser obtido através de um circuito como o da Figura 6.1 [15].

Através da relação $I_D \times V_{GS}$ (Equação (6.1)) em um transistor MOS operando na região de triodo[1], é fácil demonstrar que, se mantivermos V_{DS} constante, a corrente I_D de cada transistor será proporcional ao produto $V_{DS}V_{GS}$.

$$I_D = \mu_n C_{ox} \frac{W}{L} \left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2} \right] \tag{6.1}$$

Sendo:

μ_n $\rightarrow$ mobilidade dos elétrons

C_{ox} $\rightarrow$ capacitância por unidade de área no *gate*

W $\rightarrow$ largura do canal

L $\rightarrow$ comprimento do canal

V_T $\rightarrow$ tensão de *threshold*

Desenvolvendo a Equação (6.1) temos:

$$I_D = \mu_n C_{ox} \frac{W}{L} \left[V_{GS}V_{DS} - V_T V_{DS} - \frac{V_{DS}^2}{2} \right]$$

[1]O transistor opera na região de triodo quando $V_{GS} \geq V_T$, $V_{GD} \geq V_T$ ou $V_{DS} \leq V_{sat}$, onde $V_{sat} = V_{GS} - V_T$.

Como μ_n, C_{ox}, W, L, V_T e V_{DS} são constantes, a equação fica na forma:

$$I_D = a(V_{GS}V_{DS}) + b$$

onde a e b são constantes e, conseqüentemente, I_D será proporcional ao produto $V_{DS}V_{GS}$ como queríamos demonstrar.

Olhando novamente a Figura 6.1, verificamos que a corrente que sai pelo nó P é a soma das correntes I_D de cada um dos transistores e, portanto, o produto interno desejado.

6.1 Implementação das Sinapses

A Figura 6.2 mostra o diagrama elétrico de uma sinapse utilizando apenas um transistor MOS.

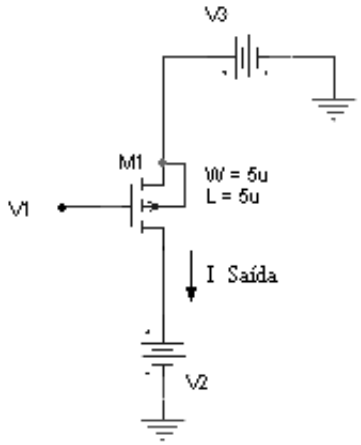


Figura 6.2: Sinapse utilizando um único transistor MOS

Neste circuito, a fonte de tensão V_1, ligada ao *gate*, representa as componentes

Tabela 6.1: Resultado do treinamento da rede neural - pesos sinápticos para os MLPs

	Peso Sináptico			
Circuito 1	0,54	0,73	0,23	0,36
Circuito 2	-0,72	0,49	-0,37	0,34
Circuito 3	-0,13	-0,43	0,42	0,79
Circuito 4	-0,42	0,23	0,80	-0,36

$x_j(n)$. Para que o circuito opere adequadamente, seu valor deve ficar compreendido entre 0 e 1 V. Como ficou convencionado que as entradas $x_j(n)$ terão valores que variam de -1 a 1, chega-se, através de aritmética simples, ao valor de 0,5 V, equivalente à entrada nula.

A fonte V_2, para efeitos de simulação, corresponde ao circuito equivalente do *current conveyor*, cuja tensão DC medida em sua entrada tem valor de 2,264 V, apresentando uma variação muito pequena (da ordem da alguns mV, conforme visto no Capítulo 4).

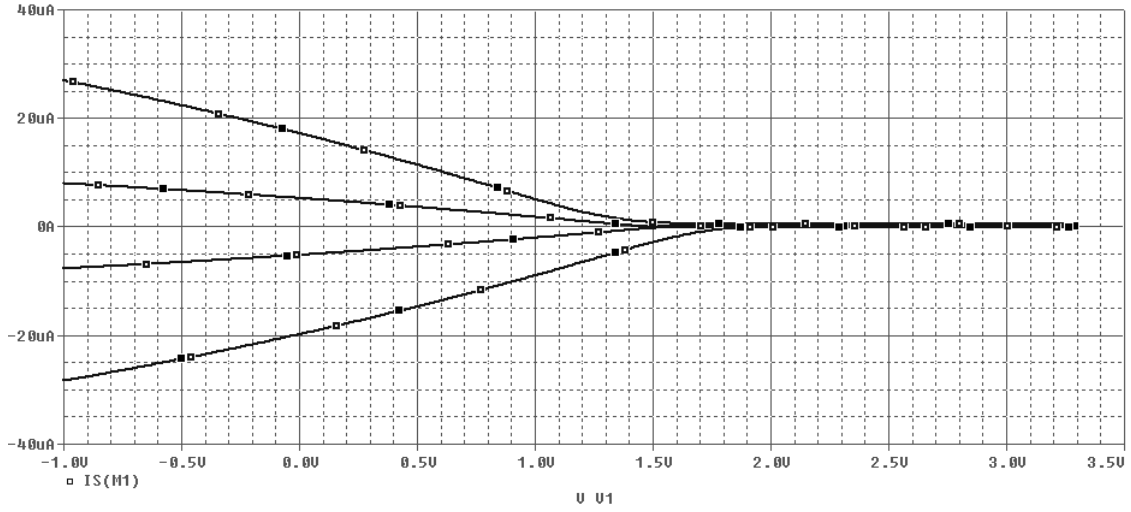
A fonte V_3 é fixa, e deve ser escolhida de forma a representar os pesos sinápticos w_{jk}. Para que o transistor opere na região de triodo, e conseqüentemente não saia de sua região linear, foi estabelecido o limite inferior $V_3 > 1{,}8\ V$.

Quatro produtos internos foram implementados neste trabalho. Seus coeficientes foram obtidos através do treinamento de uma rede neural, conforme calculado em [14], e seus valores mostrados na Tabela 6.1.

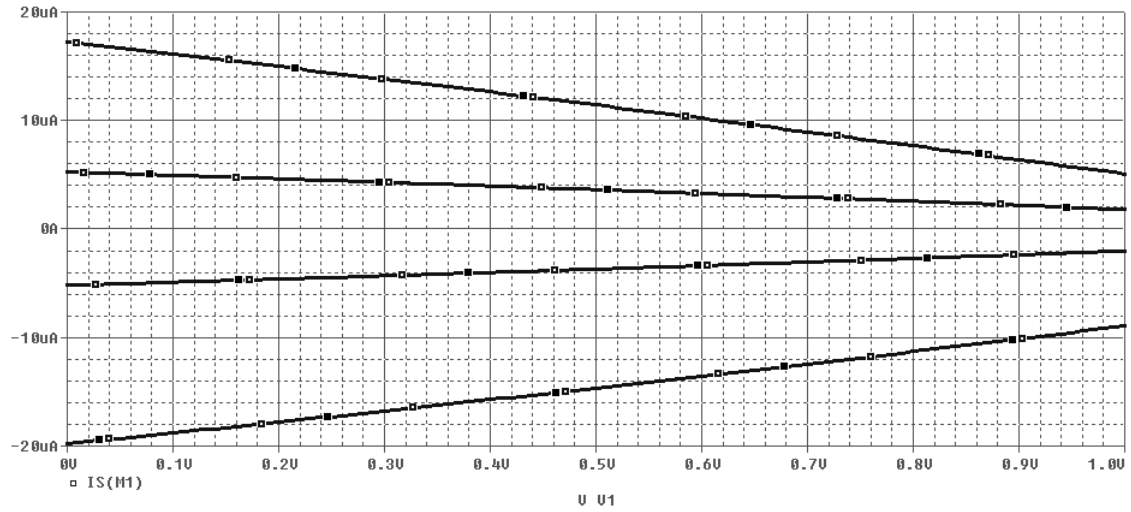
A Figura 6.3 mostra a corrente no terminal *source* do transistor em função

da tensão de *gate* (V_1), para quatro diferentes tensões da fonte V_3 (1,80 V; 2,13 V; 2,34 V e 2,72 V), correspondentes aos pesos sinápticos w_{jk}: 0,80; 0,23; -0,13 e -0,72.

Deve-se notar que, dentro da região de triodo, cada um dos valores de V_3 gera uma curva cuja inclinação (derivada) é proporcional ao seu peso sináptico w_{jk} correspondente. Como o maior peso sináptico a ser implementado neste trabalho



(a)



(b)

Figura 6.3: Corrente de dreno da sinapse da Figura 6.2, como função da tensão de *gate*, para quatro valores de V_3

Tabela 6.2: Tensão de dreno (V_D) para implementação das sinapses

Circuito 1		Circuito 2		Circuito 3		Circuito 4	
Peso sináptico	V_D	Peso sináptico	V_D	Peso sináptico	V_D	Peso sináptico	V_D
0,54	1,95 V	-0,72	2,72 V	-0,13	2,34 V	-0,42	2,52 V
0,73	1,84 V	0,49	1,98 V	-0,43	2,53 V	0,23	2,13 V
0,23	2,13 V	-0,37	2,49 V	0,42	2,02 V	0,80	1,80 V
0,36	2,05 V	0,34	2,07 V	0,79	1,81 V	-0,36	2,48 V

tem valor 0,8, associamos este valor à tensão limite de 1,8 V, cuja curva tem uma derivada de -12,02 $\mu A/V$. Conseqüentemente, o peso sináptico de valor unitário terá uma curva com inclinação de -15,03 $\mu A/V$, que foi adotada como referência para todos os outros valores de sinapses.

Observar também que, quando $V_3 < V_2$, a curva tem derivada negativa, e quando $V_3 > V_2$, a curva tem derivada positiva. Isto permite que sejam criadas sinapses tanto com valores positivos quanto com valores negativos. A Tabela 6.2 mostra as tensões de dreno utilizadas para implementar cada uma das sinapses.

6.2 Modelo Elétrico da Simulação

A Figura 6.4 mostra o circuito completo que implementa o produto interno do vetor sináptico $\mathbf{w}$ =[0,54 0,73 0,23 0,36] (Circuito 1). Observe que as tensões de *source* (V_{11}, V_{12}, V_{13} e V_{14}) dos transistores que implementam as sinapses (M11, M12, M13 e M14) são [1,95 V; 1,84 V; 2,13 V; 2,05 V] respectivamente (Tabela 6.2). As fontes V_{21}, V_{22}, V_{23} e V_{24}, ligadas ao *gate* destes mesmos transistores são, conforme mencionado na sessão anterior, as variáveis de entrada $x_j(n)$. O valor de tensão correspondente a 0,5 V representa a entrada nula. Valores de tensão de 0 V

até 0,5 V correspondem à entradas de valor negativo (entre -1 e 0), e valores de 0,5 V até 1,0 V a entradas de valor positivo (entre 0 e 1).

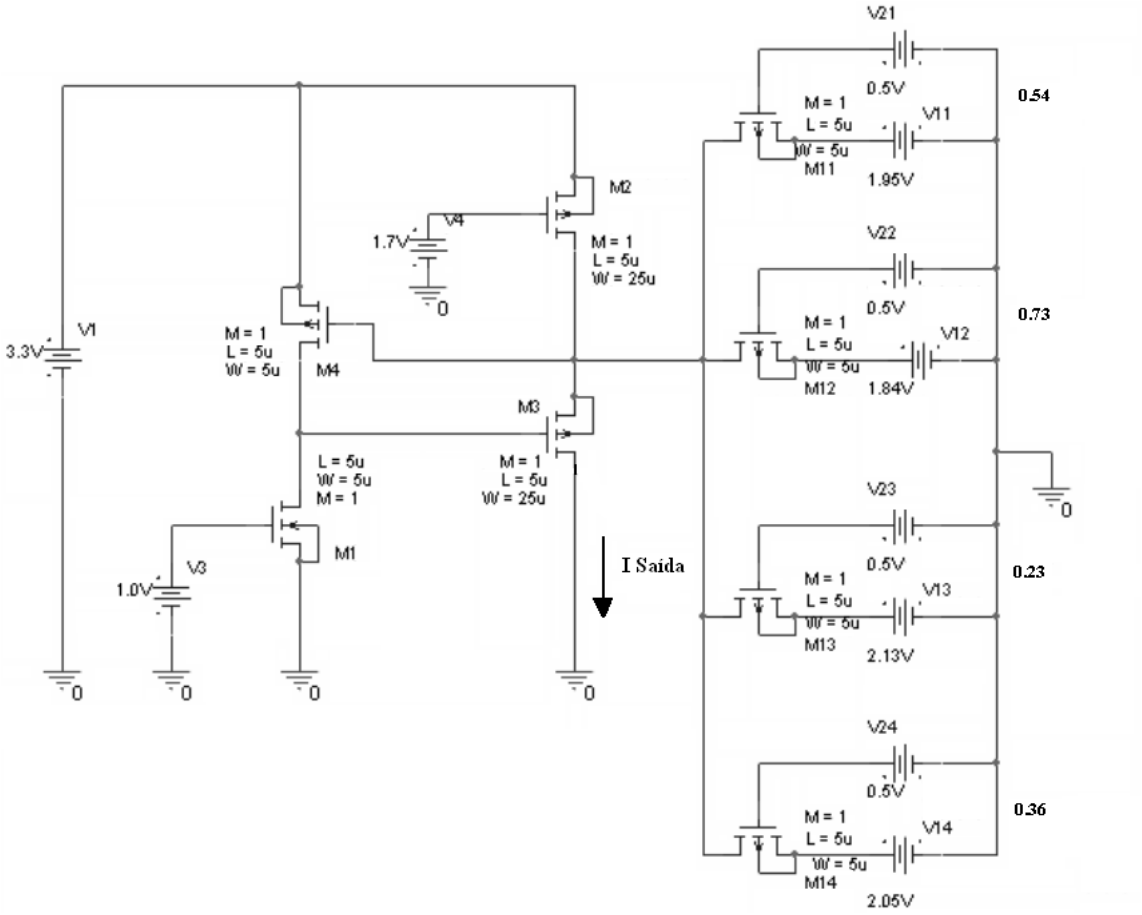


Figura 6.4: Circuito utilizado como modelo para simulação no Spice

Para a simulação no Spice, foram utilizados os parâmetros BSIM3 do processo CMOS TSMC 0.35 μm.

Assumindo que na fabricação do dispositivo são inseridos erros (com distribuição Gaussiana) na razão entre largura/comprimento (W/L), foram executadas simulações de Monte Carlo (1000 simulações) para cada produto interno projetado.

A Figura 6.5 apresenta a distribuição obtida para a corrente I_D através de M_3, onde o produto interno para a entrada $\mathbf{x} = [-1\ 1\ 1\ 1]$ foi estimado. Para esta

distribuição o valor esperado é de 33,66 μA, e o desvio padrão 0,49 μA.

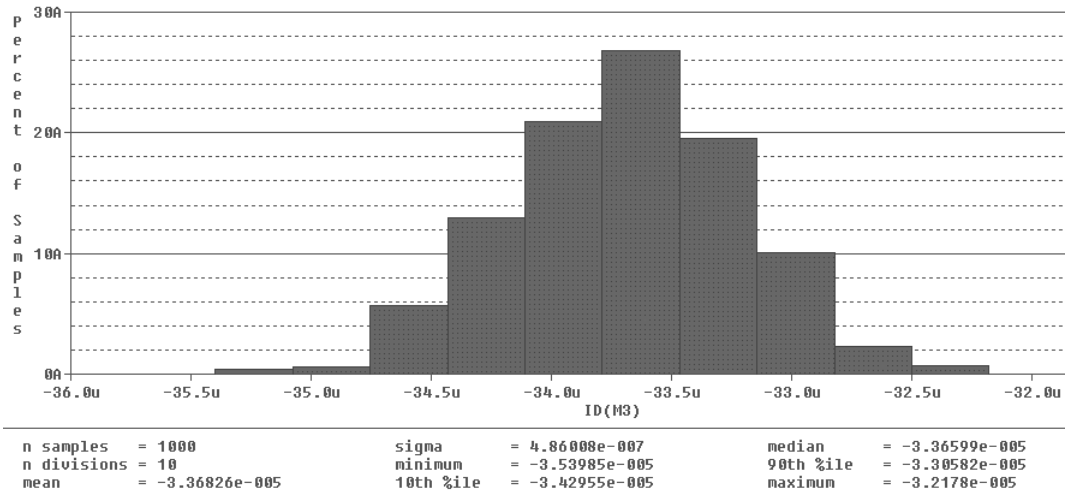


Figura 6.5: Exemplo da distribuição da corrente de saída

A Tabela 6.3 mostra os resultados para os quatro produtos internos desejados, obtidos para várias entradas diferentes. Note que $\bar{I}_{D0}$ é o *offset* médio da corrente de dreno (I_D) quando a entrada é nula. Este valor deve ser subtraído do valor médio da corrente de dreno ($\bar{I}_D$), produzido para cada entrada, para obtermos o valor de saída desejado. Quando o sinal de entrada varia entre -1 e 1, a sinapse de referência, que é a sinapse de valor unitário, produz uma variação na corrente de saída de 15 μA. Através desta relação obtivemos o fator de normalização de 7,5 (mostrado na última coluna).

Para calcularmos o produto interno entre o vetor das sinapses e um vetor de entrada genérico, expandimos o vetor de entrada como uma combinação linear dos vetores de entrada da Tabela 6.3. Os coeficientes e os desvios padrão obtidos da tabela são utilizados para calcular o resultado da operação.

Tabela 6.3: Resultado das simulações de Monte Carlo (Spice) da Figura 6.4

Circuito 1

Entradas	$\bar{I}_D$	σ	$\bar{I}_D - \bar{I}_{D0}$	Saída	Ideal	σ/7,5
0, 0, 0, 0	27,05	0,51				
-1, 1, 1, 1	33,66	0,49	6,61	0,881	0,78	0,065
1, -1, 1, 1	30,76	0,49	3,71	0,495	0,40	0,065
1, 1, -1, 1	38,34	0,48	11,29	1,505	1,40	0,064
1, 1, 1, -1	36,26	0,47	9,21	1,228	1,14	0,063

Circuito 2

Entradas	$\bar{I}_D$	σ	$\bar{I}_D - \bar{I}_{D0}$	Saída	Ideal	σ/7,5
0, 0, 0, 0	63,36	0,50				
-1, 1, 1, 1	72,04	0,48	8,68	1,157	1,18	0,064
1, -1, 1, 1	53,89	0,48	-9,47	-1,263	-1,24	0,064
1, 1, -1, 1	66,80	0,48	3,44	0,459	0,48	0,064
1, 1, 1, -1	56,23	0,47	-7,13	-0,951	-0,94	0,063

Circuito 3

Entradas	$\bar{I}_D$	σ	$\bar{I}_D - \bar{I}_{D0}$	Saída	Ideal	σ/7,5
0, 0, 0, 0	47,28	0,51				
-1, 1, 1, 1	54,23	0,49	6,95	0,927	0,91	0,065
1, -1, 1, 1	58,80	0,48	11,52	1,536	1,51	0,064
1, 1, -1, 1	46,02	0,48	-1,26	-0,168	-0,19	0,064
1, 1, 1, -1	40,49	0,49	-6,79	-0,905	-0,93	0,065

Circuito 4

Entradas	$\bar{I}_D$	σ	$\bar{I}_D - \bar{I}_{D0}$	Saída	Ideal	σ/7,5
0, 0, 0, 0	53,58	0,52				
-1, 1, 1, 1	61,84	0,49	8,26	1,101	1,09	0,065
1, -1, 1, 1	52,13	0,49	-1,45	-0,193	-0,21	0,065
1, 1, -1, 1	43,52	0,50	-10,06	-1,341	-1,35	0,067
1, 1, 1, -1	60,90	0,49	7,32	0,976	0,97	0,065

Todos os valores em μA.

6.3 Resultados em Compressão de Imagens

O sensor de imagem é dividido em blocos idênticos de 4×4 *pixels*. Do mesmo modo, para comprimirmos uma imagem, nós a dividimos também em blocos de 4×4 *pixels*. Dentro de cada bloco, a luminância de cada *pixel* foi calculada através das amostras RGB. Depois, o valor médio da luminância foi calculado e a diferença em relação ao bloco imediatamente à esquerda foi computada. Esta diferença foi quantizada com quatro bits, e o valor previsto da luminância foi subtraído de todos os *pixels*. Uma multiplicação matriz-vetor de 16×4 foi usada para calcular os primeiros quatro componentes principais das 16 amostras. Os sinais de cada um destes componentes foram transmitidos diretamente sem codificação. Os vetores 4-D sem sinal têm uma distribuição Laplaciana e foram comprimidas por uma rede neural que foi treinada para mapear o vetor $\mathbf{x}(n)$ no vetor $\hat{\mathbf{x}}(n)$ com uma entropia e uma distorção mínimas. Para treinamento da rede neural, foram utilizados vetores 4-D extraídos de uma base para treinamento contendo 21 imagens.

Adotamos que todo o processamento para gerar os vetores $\mathbf{x}(n)$ foi feito de forma ideal, e consideramos apenas o erro introduzido pela implementação do *hardware* do quantizador vetorial. No exemplo utilizado nesta dissertação, a rede neural é do tipo MLP com duas camadas. A primeira camada multiplica a matriz de pesos sinápticos pelo vetor $\mathbf{x}(n)$, produzindo $\mathbf{f}(n) = \mathbf{W}\mathbf{x}(n)$. A segunda camada divide o espaço vetorial de entrada realizando a quantização escalar de cada componente de $\mathbf{f}(n)$. É importante notar que esta operação é de fato a quantização vetorial uma vez que o centróide da célula 4-D inteira é usado para a reconstrução, em vez do valor quantizado de cada componente.

A implementação do hardware adiciona erros em $\mathbf{W}\mathbf{x}(n)$, como descrito na

Seção 6.2. Também são adicionados erros na quantização de $\mathbf{f}(n)$, uma vez que os comparadores são imprecisos. O modelo para a aleatoriedade em $\mathbf{Wx}(n)$ foi obtido através das simulações no Spice, onde utilizamos análises estatísticas (Monte Carlo) para obter o valor esperado e o desvio padrão em $\mathbf{Wx}(n)$, como mostrado na Seção 6.2. A aleatoriedade na quantização escalar foi modelada pela adição do erro Gaussiano com desvio padrão igual a 2^{-B} aos valores limites, onde B é a implementação da precisão dos limites. Consideramos dois níveis de precisão: 6 bits e 4 bits.

Na Tabela 6.3, comparamos o modelo do erro (Spice) desenvolvido na Seção 6.2 com o modelo (Teórico) obtido em [14]. Nesta comparação, os dois modelos foram utilizados para comprimir uma imagem de teste ("BikeSmall") que não havia sido incluída na base de treinamento. A imagem comprimida é mostrada na Figura 6.6.

As colunas para a entropia (H, em bits por vetor) e para a distorção (D, erro médio quadrático) são apenas para quantização vetorial dos vetores $\mathbf{x}(n)$. Na quantização vetorial perfeita, os vetores $\mathbf{x}(n)$ ficam sem compressão, e na VQ taxa zero, os vetores $\mathbf{x}(n)$ são inteiramente descartados (substituídos pelo centróide da densidade Laplaciana da base de treinamento). As colunas para a taxa (R, em bits por *pixel*) e para a qualidade (PSNR, $10\log_{10}(255^2/E)$, onde E é o erro médio quadrático entre a imagem original e a imagem reconstruída), são para a compressão da imagem inteira, e levam em consideração o DPCM e o sinal. Também foi incluída uma coluna chamada MOS (*mean opinion score*) que foi obtida através de testes subjetivos de qualidade com 23 pessoas. A cada uma das pessoas foi solicitado que atribuísse uma nota de 0 a 10 (estes limites correspondem à pior qualidade possível

e à melhor qualidade possível, respectivamente).

A cada vez que uma multiplicação matriz-vetor foi executada, um erro de saída foi adicionado ao resultado, de acordo com $\sigma/7{,}5$ da Tabela 6.3. Consideramos dois casos: para "erro fixo", o mesmo erro foi adicionado a todos os blocos do sensor. Para "erro aleatório", um erro de saída diferente foi adicionado para cada bloco do sensor. Em ambos os casos, notamos que uma PSNR semelhante foi obtido com precisão de 4 bits, enquanto que para 6 bits observamos algumas discrepâncias. Os

Tabela 6.4: Comparação entre o modelo simplificado (teórico) de [14] e o modelo realista (Spice) proposto aqui, em termos de taxa e distorção na compressão da imagem "BikeSmall"

Sistema	H (bpv)	D ($\times 10^{-3}$)	R (bpp)	PSNR (dB)	MOS
VQ Perfeita	∞	0,0	∞	27,64	8,6
VQ Taxa zero	0,0	29,7	0,46	22,99	2,1
Erro Fixo					
6b Teórico	4,26	6,0	0,72	**26,31**	**6,8**
4b Teórico	2,89	10,0	0,64	**25,69**	**6,0**
6b Spice	4,48	11,1	0,73	25,41	6,2
4b Spice	3,95	10,6	0,70	**25,50**	**5.7**
Erro Aleatório					
6b Teórico	4,58	6,2	0,74	**26,25**	**7,1**
4b Teórico	4,35	12,6	0,73	**25,19**	**4,9**
6b Spice	6,05	16,0	0,83	24,72	4,5
4b Spice	4,54	14,6	0,74	**24,86**	**4,7**

valores da coluna MOS confirmam as conclusões obtidas pela PSNR.

Como um exemplo dos resultados obtidos com o sistema em consideração, podemos ver na Figura 6.6 a compressão da imagem através do modelo teórico de

Figura 6.6: Resultados da compressão da imagem com o Modelo Teórico 4b (acima) e o Modelo do Spice 4b (abaixo), ambos contendo erro de saída fixo

4 bits e o modelo do Spice de 4 bits, ambos com erro de saída fixo. Notamos que, enquanto ambos os modelos prevêem alguma degradação (na maioria efeito de bloco e algum ruído de quantização de alta freqüência), esta previsão é similar nos dois modelos. Isto está de acordo com os resultados da Tabela 6.3, o que demonstra que o modelo do Spice é equivalente ao modelo teórico de 4 bits.

6.4 Sugestões para Continuidade deste Trabalho

Antes da integração do circuito, torna-se necessário o projeto de circuitos complementares que possibilitem, na bancada, a medição dos resultados com valores mais altos do que estes, com correntes da ordem de μA. Pode-se utilizar, como sugestão, espelhos com transistores MOS, projetados para elevar esta corrente que será aplicada posteriormente a um *buffer* de saída. Veja na Figura 6.7 o esquema de um espelho de corrente típico.

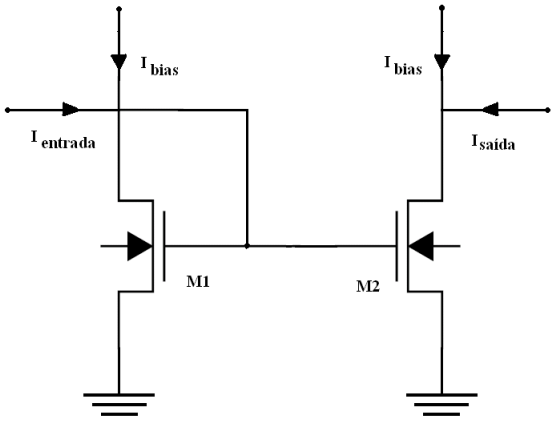


Figura 6.7: Diagrama de um espelho de corrente

Podem-se também utilizar circuitos que convertam o resultado em valores de tensão. Para esta escolha, obviamente, deve-se manter a solução que se apresente a mais simples possível e que, ao mesmo tempo, não seja nova fonte de introdução de erros.

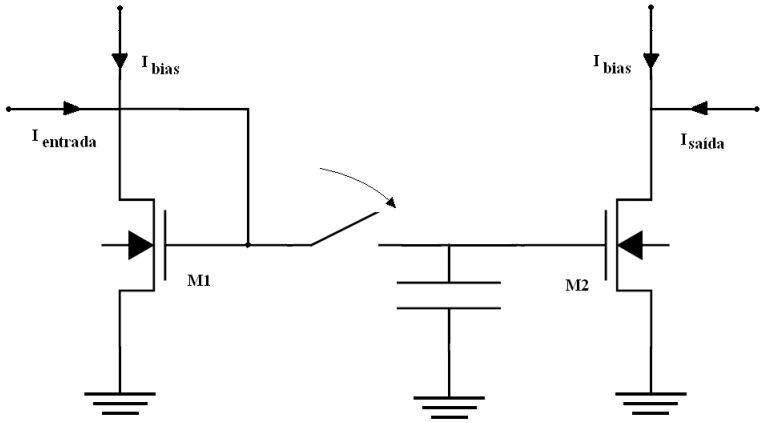


Figura 6.8: Esquema simplificado de uma memória de corrente

Quando as entradas são iguais a zero na Figura 6.4, a saída do circuito apresenta uma corrente, devido à polarização, que equivale à saída nula. Para que o circuito apresente o resultado correto do produto interno, este valor deve ser subtraído da corrente de saída logo após sua medição. Memórias de corrente podem ser utilizadas, junto com circuitos seqüenciais que garantam que o circuito opere adequadamente, para armazenamento desta corrente com o circuito em repouso, e posteriormente subtraída da corrente de saída quando em operação. A idéia básica para uma memória de corrente é mostrada na Figura 6.8. Seu funcionamento pode ser descrito da seguinte forma: inicialmente a chave encontra-se aberta e a carga do capacitor é nula. Logo após a corrente de entrada ser aplicada ao circuito, a chave

é então fechada e, após um intervalo de tempo T, a tensão no capacitor atinge um valor máximo V_c que permanece constante mesmo após a abertura da chave. Neste momento a corrente de entrada pode ser retirada porque a tensão V_c, aplicada ao *gate* do transistor M2, é tal que, como no espelho de corrente, o valor da corrente de saída se mantenha com o mesmo valor que a corrente de entrada, desde que os transistores sejam iguais.

Outros modelos de multiplicadores também podem ser utilizados para a construção das sinapses [23]. Atualmente estuda-se como continuação deste trabalho a possibilidade do uso de sinapses implementadas através de espelhos de corrente.

Capítulo 7

Conclusão

Longe de esgotar o estudo sobre o tema, nesta dissertação propusemos um circuito elétrico que poderá ser utilizado nas aplicações de compressão de imagens no plano focal, bem como obtivemos um novo modelo para observar os erros que são inseridos no sistema, devido a imprecisões no cálculo do produto interno causados pelo processo de fabricação. Tal modelo foi obtido através de várias simulações utilizando o Spice, levando em consideração as variações aleatórias no processo de fabricação da tecnologia CMOS 0.35 μm. Como resultado, podemos predizer com mais confiabilidade a performance do circuito após sua implementação em *chip* do que com o modelo teórico anterior.

Com este estudo, podem-se tirar as seguintes conclusões:

1. A qualidade da compressão da imagem não é afetada de modo significante pela precisão do limiar da camada de saída;

2. A qualidade da compressão é aproximadamente a mesma do sistema equivalente com 4 bits de precisão - ambos os modelos, tanto o obtido de [14] quanto o modelo apresentado aqui, apresentam praticamente os mesmos resultados,

que são aqueles obtidos de um sistema de compressão de imagens de baixa precisão;

3. A qualidade de compressão equivalente ao modelo teórico de 6 bits pode ser obtida melhorando-se a precisão da razão W/L dos transistores, o que de outra forma, iria aumentar a área do circuito e conseqüentemente reduzir a resolução do sensor.

Referências Bibliográficas

[1] A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasović, R. E. Jenkins, e K. Strohbehn. Current-mode subthreshold MOS circuits for analog and VLSI neural systems. *IEEE Trans. Neural Networks*, 2(2):205–213, 1991.

[2] W. S. Boyle e G. E. Smith. Charge-coupled semiconductor devices. *Bell Systems Technical Journal*, 49(3):587–593, 1970.

[3] K. Bult e H. Waalinga. A class of analog CMOS circuits based on the square-law characteristic of an MOS transistor in saturation. *IEEE J. Solid-State Circuits*, 22(3):357–365, 1987.

[4] G. Cauwenberghs e V. Pedroni. A low-power CMOS analog vector quantizer. *IEEE J. Solid-State Circuits*, 32(8):1278–1283, 1997.

[5] G. Cauwenberghs e J. Waskiewicz. Analog VLSI cellular implementation of the boundary contour system. *Anais Neural Information Processing Systems*, pp. 657 – 663, Denver, CO, EUA, 1998.

[6] T. Chen, P. Catrysse, A. El Gamal, e B. Wandell. How small should pixel size be? *Anais SPIE*, volume 3965, pp. 451–459, 2000.

[7] P. A. Chou, T. Lookabaugh, e R. M. Gray. Entropy-constrained vector quantization. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37(1):31–42, 1989.

[8] K. G. de Lima. *Estruturas APS Resistentes à Radiação para Aplicações Espaciais*. COPPE/UFRJ, Dissertação de Mestrado, 2006.

[9] S. Espejo, A. Rodríguez-Vazquez, R. Domínguez-Castro, J. L. Huertas, e E. Sánchez-Sinencio. Smart-pixel cellular neural networks in analog current-mode CMOS technology. *IEEE J. Solid-State Circuits*, 29(8):895–905, 1994.

[10] G. Ferri e N. C. Guerini. *Low-Voltage Low-Power CMOS Current Conveyors*. Kluwer Academic Publishers, 2003.

[11] E. R. Fossum. Active pixel sensors: are CCD's dinosaurs? *Anais SPIE*, volume 1900, pp. 2–14, 1993.

[12] E. Funatsu. Artificial retina large scale integration with on-sensor projection function for high-speed motion detection. *Optical Engineering*, 41(11):2709–2718, 2002.

[13] A. Gersho e R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer, Boston, EUA, 1992.

[14] J. G. R. C. Gomes. *Mixed-Signal Multilayer Perceptron Implementation of Low-Complexity Vector Quantizers for Image Compression*. Universidade da Califórnia, EUA, Tese de Doutorado, 2004.

[15] J. G. R. C. Gomes, M. J. C. Mello, H. L. Haas, e A. Petraglia. New error sensitivity model for the analog hardware implementation of inner products. *Anais IEEE Int. Conf. Image Processing*, Atlanta, GA, EUA, 2006.

[16] J. G. R. C. Gomes, A. Petraglia, e S. K. Mitra. Sensitivity analysis of multilayer perceptrons applied to focal-plane image compression. *IET Circuits, Devices & Systems*, 1(1):79–86, 2007.

[17] W. B. Green. *Digital Image Processing*. Van Nostrand Reinhold Company, New York, 1983.

[18] S. Haykin. *Neural Networks - A Comprehensive Foundation*. Prentice-Hall, New Jersey, NJ, 2^a Edição, 1999.

[19] S. Kleinfelder, S. Lim, X. Liu, e A. El Gamal. A 10000 frames/s CMOS digital pixel sensor. *IEEE J. Solid-State Circuits*, 36(12):2049–2059, 2001.

[20] W. D. Leon, S. Balkir, e K. Sayood. An evolvable predictor for lossless image compression. *Anais IEEE Int. Symp. Circuits and Systems*, pp. IV.731–IV.734, Scottsdale, AZ, EUA, 2002.

[21] W. D. Leon, S. Balkir, e K. Sayood. A CMOS imager with pixel prediction for image compression. *Anais IEEE Int. Symp. Circuits and Systems*, pp IV.776–IV.779, Bangcoc, Tailândia, 2003.

[22] W. D. Leon, S. Balkir, K. Sayood, e M. W. Hoffman. Charge-based prediction circuits for focal plane image compression. *Anais IEEE Int. Symp. Circuits and Systems*, pp. IV.936–IV.939, Vancouver, Canadá, 2004.

[23] G. Liñán. *Diseño de Chips Programables de Señal Mixta con Bajo Consumo de Potencia para Sistemas de Vision en Tiempo Real*. Universidade de Sevilha, Espanha, Tese de Doutorado, 2002.

[24] G. Linán-Cembrano, A. Rodríguez-Vázquez, R. Carmona-Galán, F. Gimenez-Garrido, S. Espejo, e R. Domínguez-Castro. A 1000 FPS at 128x128 vision processor with 8-bit digitized I/O. *IEEE J. Solid-State Circuits*, 39(7), 2004.

[25] W.S. McCulloch e W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[26] C. Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, Reading, MA, EUA, 1989.

[27] S. K. Mendis et al. CMOS active pixel image sensors for highly integrated imaging systems. *IEEE J. Solid-State Circuits*, 32(2):187–197, 1997.

[28] B. Schölkopf, A. Smola, e K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[29] A. S. Sedra. *A New Approach to Active Network Synthesis*. Tese de Doutorado, Universidade de Toronto, Canadá, 1969.

[30] A. S. Sedra, G.W. Roberts, e F. Gohh. The current conveyor – history, progress and new results. *IEE Proceedings*, volume 137, 1990.

[31] A. S. Sedra e K. C. Smith. A second generation current conveyor and its applications. *IEEE Trans. on Circuit Theory*, CT-17:132–134, 1970.

[32] K. C. Smith e A. S. Sedra. The current conveyor – a new circuit building block. *Proceedings of the IEEE*, volume 56, pp. 1368–1369, 1968.

[33] R. Tawel. Real-time focal-plane image compression. *Anais Data Compression Conf.*, pp. 401–409, Snowbird, UT, EUA 1993.

[34] X. Zeng e D. S. Yeung. Sensitivity analysis of multilayer perceptron to input and weight perturbations. *IEEE Trans. Neural Networks*, 12(6):1358–1366, 2001.

[35] Z. Zhou, B. Pain, e E. R. Fossum. CMOS active pixel sensor with on-chip successive approximation analog-to-digital converter. *IEEE Trans. Electron Devices*, 44(10):1759–1763, 1997.