Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
COPPE UFRJ

UM ESTUDO SOBRE O MECANISMO CORTICAL DO DIFFUSIVE FILLING-IN EM SILÍCIO

Genildo Nonato Santos

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Rio de Janeiro
Setembro de 2016

UM ESTUDO SOBRE O MECANISMO CORTICAL DO DIFFUSIVE FILLING-IN EM SILÍCIO

Genildo Nonato Santos

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

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

Prof. Antonio Petraglia, Ph.D.

Prof. Carlos Augusto Duque, D.Sc.

Prof. Fernando Antonio Pinto Barúqui, D.Sc.

Prof. Mário Fiorani Junior, D.Sc.

Prof. Ricardo Guerra Marroquim, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2016

Santos, Genildo Nonato

Um Estudo sobre o Mecanismo Cortical do Diffusive Filling-In em Silício/Genildo Nonato Santos. – Rio de Janeiro: UFRJ/COPPE, 2016.

XIX, 153 p.: il.; 29, 7cm.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2016.

Referências Bibliográficas: p. 79 – 84.

1. Diffusive Filling-In. 2. Silicon Retina. 3. Spiking Neural Network. 4. CMOS Neuron Models. 5. Amplitude Mode Simulation. 6. Spike Mode Simulation. I. Gomes, José Gabriel Rodríguez Carneiro. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Dedico esta Tese:

Primeiramente a Deus, que
permitiu todas as graças
alcançadas;
Aos meus pais, que sempre me
apoiaram e me deram forças;
A minha família, que sempre
esteve ao meu lado.

Agradecimentos

Agradeço ao meu orientador, o professor José Gabriel Rodríguez Carneiro Gomes pelo aprendizado e a sua constante disposição que possibilitou o desenvolvimento deste trabalho. Aos meus colegas F. Mederos, F. Duarte, A. Bides, J. Alberto e T. Brito do laboratório de processamento digital e analógico de sinais (PADS) pelas dicas de utilização das ferramentas e pelo apoio como amigos. Ao aprendizado transmitido pelos professores: A. Mesquita, A. Moreirão, A. Petraglia, M. Petraglia, A. Barúqui, R. Marroquim. Aos colegas e amigos do laboratório PADS da Universidade Federal de Rio de Janeiro pelo agradável ambiente de trabalho, que contribuíram direta ou indiretamente neste trabalho. A todos os que contribuiram diretamente e indiretamente para que eu pudesse concluir esse trabalho de Tese. À minha família, que sempre proporcionou carinho e apoio.

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

UM ESTUDO SOBRE O MECANISMO CORTICAL DO DIFFUSIVE FILLING-IN EM SILÍCIO

Genildo Nonato Santos

Setembro/2016

Orientador: José Gabriel Rodríguez Carneiro Gomes

Programa: Engenharia Elétrica

Para executar simulações de circuitos corticais em modo pulsado, isto é, considerando o sinal neural em termos de sequências de pulsos, o uso de hardware dedicado é recomendado. Muitos problemas estão envolvidos no projeto do hardware dedicado, o alto nível de complexidade de hardware para implementações de redes neurais pulsadas é um exemplo. As conexões entre os neurônios em uma rede neural pulsada se tornam impraticáveis sem a redução da quantidade de conexões. O axônio compartilhado ou SA é uma técnica de multiplexação no tempo que permite redução da quantidade de conexões. Contudo, o SA pode mudar significativamente a informação processada pela rede neural pulsada por causa de atrasos inseridos aleatoriamente no sinal. A simulação de circuitos corticais, com intuito de especificar o projeto do hardware dedicado, costuma ser conduzida em modo de amplitude, onde a informação neural é representada por uma sequência de valores escalares que descrevem as taxas de disparo dos neurônios. Ao representar sinais neurais por sequências discretas contendo taxas de disparo, a simulação em modo de amplitude não captura importantes características da dinâmica neural. Propomos uma abordagem que permite níveis mínimos de complexidade de hardware que causa baixa distorção, simulações em modo de amplitude e pulsado dos circuitos corticais que compõem o mecanismo cortical do *diffusive filling-in*. Essas simulações mostram que os resultados do modo de amplitude se aproximam bem dos resultados do modo pulsado. Utilizamos estímulos, estimados a partir de simulações de uma retina, não ideais que realçam as diferenças entre o modo de amplitude e o modo pulsado. Simulações mostram que a latência foi reduzida e sugerem que o modo de amplitude é confiável para realizar a análise de resultados de simulação em projetos de hardware customizado para simulações de circuitos corticais.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A STUDY ABOUT DIFFUSIVE FILLING-IN CORTICAL MECANISM IN SILICON

Genildo Nonato Santos

September/2016

Advisor: José Gabriel Rodríguez Carneiro Gomes

Department: Electrical Engineering

To run cortical circuit simulations in spike mode, i.e. taking into account the neural representation of information in terms of sequences of pulses (spikes), the use of customized hardware is recommended. Many problems are involved with this customized hardware design, for example, high hardware complexity is involved in the implementation of the spiking neural networks. Point-to-point connections between neurons in spiking neural networks becomes impractical without the use of hardware-level complexity reduction methods, e.g. shared axon or SA, which is a kind of time multiplexing technique. The SA technique randomly inserts delays in neural network spike sequences, which may lead to information loss. These delays are called latency. Simulation results are important for optimized hardware design. The simulations are carried out in amplitude mode. In amplitude mode information is represented by sequences of scalar values that describe neural input and output spike rates. However, by representing neural signals by conventional discrete sequences containing neural firing rates, the amplitude-mode simulation approach does not capture important features of neural dynamics. We propose a lower-latency approach which reduces to a minimum the hardware-level complexity. We carry out amplitude and spike-mode simulations of a cortical algorithm, namely the diffusive filling-in algorithm, to investigate whether the amplitude-mode results approximate well the spike-mode results. We used a non-ideal input, estimated from simulations of a retina, that highlights the differences between the amplitude and spike-mode approaches. Simulation results show that the latency was reduced and suggest that the amplitude mode is reliable for theoretical predictions, which are useful for customized hardware design for cortical circuit simulations.

Sumário

Lista de Figuras **x**

Lista de Tabelas **xvii**

Lista de Abreviaturas **xviii**

1 Introdução **1**

1.1 Motivações 2
1.2 Problemática 3
1.3 Objetivos 4
1.4 Metodologia de Trabalho 5

2 O EVS e os Estímulos para o DFI **7**

2.1 Modelo para a OPL 8
2.1.1 A OPL em uma Implementação CMOS 12
2.2 Modelo para a IPL 17
2.3 Modelo para a Camada de Células Ganglionares 19
2.4 Resultados para a OPL e a IPL 21
2.5 Conclusões 24

3 O Nervo Óptico e a Técnica AER **26**

3.1 Elementos de um Sistema AER 27
3.2 Latência em Sistemas AER 38
3.3 Complexidade de Hardware em Sistemas AER 40
3.4 Comparação entre Sistemas AER 42
3.5 Técnica AER Proposta 43
3.6 Resultados para o AER NAP 48
3.7 Conclusões 49

4 A Retina CMOS **50**

4.1 Dimensões dos Transistores MOS para o AER Proposto 54
4.2 Resultados de Reconstrução do AER Proposto 57

4.3 Resposta ao Impulso do AER Proposto 58
4.4 Estímulos para o Mecanismo Cortical 58
4.5 Conclusões 60

5 Circuitos Corticais Pulsados **63**
5.1 O Neurônio, a Representação no Tempo e o Processamento Pulsado . 64
5.2 Modelo do Diffusive Filling-In 67
5.2.1 Detalhes sobre o DFI 69
5.3 Resultados do DFI 73
5.4 Conclusões 73

6 Conclusões e Trabalhos Futuros **76**
6.1 Contribuições 77
6.2 Trabalhos Futuros 78

Referências Bibliográficas **79**

A Scripts Ocean **85**
A.1 Script Diagonal 85
A.2 Script Impulso no Espaço e no Tempo 132
A.3 Gerador de Scripts 149
A.4 Reconstrução dos Arquivos de Eventos 150

Lista de Figuras

2.1 Diagrama em blocos do EVS. É apresentada a OPL, a IPL, a camada de células ganglionares e os respectivos sinais em suas entradas e saídas. 7

2.2 Em um cabo dividido em elementos infinitesimais (x), de comprimento ΔL, existe uma corrente que flui horizontalmente pelo cabo ($I_i(x)$) e uma corrente que é conduzida verticalmente pelo elemento infinitesimal ($I_m(x)$). 8

2.3 Os componentes do modelo elétrico de um cabo são apresentados. Os elementos infinitesimais do cabo (capacitâncias e condutâncias) são conectados a partir de resistores (R_i). 9

2.4 Na parte superior da figura são mostrados o modelo elétrico de três fotorreceptores que compõem o modelo da rede de fotorreceptores, na figura da parte de baixo o modelo elétrico de três neurônios da rede de células horizontais. Fontes de corrente controlada por tensão (tensões $V_c(x)$ e $V_h(x)$) implementam as interações que ocorrem entre as duas redes. 11

2.5 Os parâmetros que definem a resposta da Eq. (2.5) foram ajustados para que os picos em frequência espacial e temporal fossem de 10 Cpd e 10 Hz. O objetivo era o ajuste para 1 Cpd e 10 Hz, contudo, devido a limitações na quantidade de pontos no espaço da nossa simulação o valor de 1 Cpd teve que ser escalado para 10 Cpd. 12

2.6 Uma unidade da OPL CMOS equivalente a do diagrama esquemático da Fig. 2.4. As fontes de corrente controladas por tensão e os resistores são substituídos por transistores MOS. Esse esquema foi adaptado de Boahen [16]. 16

2.7 Diagrama em blocos do EVS, em destaque a IPL, e diagrama esquemático do circuito retificador da IPL. 18

2.8 Na estrutura apresentada temos a matriz de neurônios com $X \cdot Y$ neurônios, cada um deles numa posição (x, y). Ao lado da matriz é apresentada a estrutura interna de um desses neurônios. A figura foi adaptada de Boahen [32]. 20
2.9 Diagrama esquemático de uma unidade da OPL. Esse diagrama esquemático ilustra a implementação do modelo elétrico, apresentado na Fig. 2.6, feita na interface de desenvolvimento Cadence Virtuoso. . 21
2.10 A implementação na interface de desenvolvimento do diagrama esquemático dos pseudo-resistores R_{hh} e R_{cc}. 22
2.11 *Script* que foi utilizado para realizar a simulação a partir da interface *Ocean Script*. O comando utilizado para alterar o valor das fontes de corrente *ik* é o *desVar*. 22
2.12 Simulação elétrica da resposta da OPL 50x1 ao impulso no espaço. O eixo horizontal representa cada unidade da OPL. O esperado seria uma resposta semelhante à resposta ao impulso espacial de um filtro DoG, resposta conhecida como chapéu mexicano. Assim o formato obtido é semelhante ao esperado. O formato está invertido devido às características de polarização elétrica dos transistores MOS que foram utilizados na implementação. 23
2.13 A resposta do modelo elétrico da OPL (50x1) ao impulso no espaço e no tempo. O impulso é aplicado à posição 26 do arranjo. 23
2.14 A resposta espaço-temporal do modelo de uma OPL para estímulos senoidais de valores variando no espaço e no tempo. A OPL foi projetada para uma resposta passa banda espaço-temporal de 10 Cpd e 10 Hz. Na figura é mostrada a previsão teórica (em cima) e o resultado da simulação elétrica (embaixo) que foi obtido ao simular o diagrama esquemático da OPL que foi projetado a partir da metodologia proposta. 24
2.15 Diagrama esquemático do modelo do retificador. O diagrama foi utilizado para a realização de simulações elétricas. 25
2.16 Resposta dos dois canais (*On* e *Off*) na saída do retificador produzidas por um estímulo que varia de acordo com o sinal de saída do diagrama esquemático da OPL, sinal Vc. É possível verificar que, apesar de certo nível de *off-set* de corrente, as respostas dos canais são separadas em relação a tensão de referência Vref. 25

3.1 Diferenças entre as topologias FD e SA. Considerando que os retângulos pontilhados são circuitos CMOS, onde cada um dos circuitos implementa três neurônios, à esquerda temos a topologia FD que necessita de n^2 conexões, enquanto na topologia SA essa quantidade pode ser reduzida para um canal de comunicação de largura de $\log_2(n)$ bits. 26
3.2 A arquitetura de comunicação AER é apresentada. Elementos de processamento, elementos codificadores e um elemento de controle possibilitam que um evento gerado na matriz do estágio transmissor seja enviado até a matriz do estágio receptor. 28
3.3 Uma ilustração da arquitetura de comunicação AER onde N eventos e_n, $n = 1, 2, \ldots, N$, são transmitidos pelo canal de comunicação. Em uma arquitetura AER, não é possível garantir que o evento e_n seja reconstruído perfeitamente em relação ao instante de tempo t. A figura foi adaptada de [12]. 29
3.4 Arquitetura dos elementos de processamento LC e D. Essa arquitetura possibilita a redução da quantidade de conexões entre a matriz de neurônios e o codificador. Um diagrama esquemático de cada elemento é mostrado. 30
3.5 A parte (a) da figura representa o diagrama esquemático das estruturas LC, no topo, e D, na parte inferior, utilizadas em Boahen [33]. Na parte (b) da figura [34] é apresentado um diagrama esquemático onde as estruturas LC, D e o neurônio formam um único bloco. . . . 31
3.6 A arquitetura de um árbitro AB que gerencia oito requisições (de x_1 até x_8). Os sinais de entrada $R_{eq}(A)$, $R_{eq}(B)$ e $A_{ck}(In)$ são respectivamente representados pelos terminais 3, 4 e 6. Os sinais de saída Ack_A, Ack_B e $R_{eq}(Out)$ são representados respectivamente pelos terminais 1, 2 e 5. 33
3.7 A CD é composta por estruturas digitais biestáveis como a que é apresentada na figura à direita. À esquerda é apresentado um exemplo de árvore binária. Os terminais dos elementos que compõem essa árvore binária estão de acordo com o que foi apresentado na Fig. 3.6. 34
3.8 Arquitetura utilizada no árbitro do tipo WTA proposto [34] (à esquerda) e o diagrama esquemático de uma CD desse árbitro (À direita). Os sinais $ReqA$ ou $ReqB$ da CD da árvore binária são equivalentes ao sinal representado pelo terminal (1) e os sinais $AckA$ ou $AckB$ são equivalentes ao terminal (2). Os outros sinais são de controle e polarização da estrutura. 34

3.9 O codificador recebe do árbitro os X sinais $A_{ck}(x)$ e os codifica em um endereço composto de $\lceil \log_2(X) \rceil$ bits. Isso ocorre porque o árbitro faz com que, dos eventos entregues ao codificador, apenas um esteja ativo por vez. 35
3.10 No codificador o vetor de saída D_X é uma função do vetor de entrada A_{ck}, sendo que a função é definida dessas operações de codificação Enc. Dependendo do código a ser gerado as operações Enc podem ocorrer de duas formas diferentes, operações de código 1 e operações de código 0. 36
3.11 Decodificador utilizado em sistemas AER arbitrados. O decodificador é projetado de maneira que cada código recebido ative somente uma de suas saídas $R_e(x)$, deixando as outras em estado de alta impedância. 37
3.12 A probabilidade de colisão, com função de uma população de neurônios e com uma taxa de disparos de 100 Spk/s. São mostradas três curvas, sendo cada uma delas para um dado T. 44
3.13 Diagrama esquemático do neurônio utilizado no sistema AER não arbitrado proposto. O neurônio apresentado na Fig. 2.8 sofre duas modificações, uma chave analógica é adicionada a sua entrada (chave analógica controlada pelo terminal 5) e apenas um sinal A_{ck} é utilizado para reiniciar o neurônio, terminal 4. Devido as alterações feitas o diagrama esquemático alcança uma complexidade de *hardware* de 8 transistores, 1 transistor a mais do que na Fig. 2.8. 45
3.14 Diagrama esquemático do gerenciador de eventos. Esse dispositivo é capaz de identificar quando um evento é gerado e se a quantidade de eventos gerados é igual ou maior que dois. 46
3.15 Lógica de comunicação utilizada, na topologia proposta, para a ligação entre os sinais de saída do gerenciador de eventos e os sinais de entrada da matriz de neurônios. 47

4.1 A figura mostra o diagrama esquemático de um dos 50 x 50 pixels do imageador que foi utilizado para realização dos testes de simulação. Na figura é possível identificar os diagramas esquemáticos da OPL, IPL e das células ganglionares. 50
4.2 A figura mostra detalhes das conexões entre os pixels do diagrama esquemático da matriz do imageador. É possível ver como são as conexões entre os fotodiodos, o pixel, os pseudo-resistores e as estruturas LC. 51
4.3 O diagrama esquemático da matriz de pixels é mostrado na figura. Um pixel e o sistema de comunicação estão destacados. 51

4.4 Detalhes dos sinais e o diagrama esquemático da estrutura LC. Os sinais de entrada são InOn e InOff e os sinais de saída são OutXOn, OutXOff, OutYOn e OutYOff. 52
4.5 A figura mostra o diagrama esquemático utilizado para a estrutura D. Os sinais de saída da estrutura LC são recebidos a partir dos sinais de entrada Inx e Iny. Os sinais de saída da estrutura D são os sinais Ox e Oy. Com x e y variando de 1 até 50. 52
4.6 Diagrama esquemático do sistema gerenciador de eventos proposto. Na figura é possível identificar as entradas de P1 até P15 e as saídas Event e Comp do sistema gerenciador de eventos. 53
4.7 A figura mostra a matriz do decodificador. Em destaque estão as entradas de P1 até P5 e as saídas de D0 até D2. 54
4.8 Análise de Monte Carlo para verificação da perda de informação (a) e descaracterização da taxa de disparo dos neurônios (b). Essa análise foi feita sobre o sistema AER proposto. 55
4.9 A reconstrução de um estímulo em diagonal aplicado ao sistema AER proposto. Como a OPL foi desabilitada toda a informação deveria passar pelo canal *On* (esquerda). No entanto, é possível perceber transmissões feitas pelo canal *Off* (direita). Apesar de algum ruído, a diagonal foi reconstruída a partir da informação de eventos aquisitada no canal *On*. 57
4.10 O imageador é estimulado por um impulso no espaço e no tempo. A resposta ao impulso do sistema AER proposto é mostrada em duas vistas. É possível perceber as semelhanças entre os perfis da resposta apresentada aqui e a resposta ao impulso de uma retina biológica apresentada por Eggermont [36], Fig. 4.11. 58
4.11 A resposta ao impulso de uma retina biológica que foi adaptada de Eggermont [36]. Essa figura é utilizada como uma previsão teórica do resultado esperado da resposta ao impulso do nosso imageador. É possível perceber semelhanças entre a posições dos vales dos gráficos apresentados aqui e os gráficos apresentados na Fig. 4.10. 59
4.12 A previsão teórica da resposta (superior, esquerda e meio), da simulação elétrica, do processamento do imageador proposto sobre um dado estímulo, coluna esquerda na linha superior. Na linha de baixo é mostrada a resposta ao impulso do imageador. 60

4.13 A diversidade no valor das amostras causada nos sinais de estímulos do DFI após o processamento realizado pelo imageador proposto, um modelo elétrico da retina (a). Se esse processamento for substituído por aquele realizado por um filtro DOG (b), a diversidade não será tão grande e assim, nesse caso, os estímulos produzidos serão menos realistas. 61

5.1 Taxa de disparo na saída de um neurônio biológico (linha pontilhada) como função de uma taxa de entrada que corresponde a um contraste normalizado [41], e a aproximação da resposta de dois modelos para essa curva, LIF (linha sólida) e IF (linha com marcadores quadrados). 65

5.2 Implementações do modo pulsado (a) e do modo de amplitude (b) de um exemplo computacional de um detector de bordas. A imagem de entrada $v_i(n)$ é obtida a partir da convolução, no domínio espacial, entre uma imagem de entrada $I(x, y)$ e um filtro passa alta $g(x, y)$ (nesse trabalho os filtros serão chamados de pesos sinápticos). No modo pulsado (a), A imagem de entrada $v_i(n)$ é utilizada como entrada de uma equação diferencial de acordo com o modelo do LIF in Eq. (5.1). No modo de amplitude (b), a equação diferencial usada corresponde ao modelo descrito pela Eq. (5.2). 66

5.3 O diagrama de blocos do algoritmo do DFI (em modo pulsado ou em modo de amplitude). Para gerar a imagem $v_{DFI}(x, y)$ a partir das imagens $v_{On}(x, y)$, $v_{FFI}(x, y)$ e do nível de limiar v_{th}, é utilizada a função de ativação $N[.]$ que foi indicada nas Figs. 5.2(a) e 5.2(b). A convolução com o peso $g_{FFI}(x, y)$ permite o espalhamento da atividade neural a partir de um pixel para os seus vizinhos. Um laço de realimentação é implementado por meio de um neurônio (o modelo). Esse laço permite que o espalhamento continue até tomar a imagem inteira. Contudo uma operação de subtração com a imagem de saída do estágio de processamento *boundary contour system* (BCS, mais detalhes a seguir) $v_{BCS}(x, y)$ faz com que o espalhamento pare. Em modo de amplitude temos que $v_{th} = v_{th1} = 0.01$. Em modo pulsado $v_{th} = v_{th1} = 1$. 67

5.4 Diagrama de blocos do BCS. A imagem de entrada é $v_{Off}(x, y)$ e a imagem de saída é $v_{BCS}(x, y)$. Todas as operações são realizadas simultaneamente ao longo dos canais orientados em 0^o, 30^o, 60^o, 90^o, 120^o, e 150^o. A convolução com o filtro DF (direcional) seguido por uma operação de limiar com o nível v_{th2} tende a manter os segmentos do contorno, em cada canal orientado, ao longo de uma direção em particular. A convolução com o filtro SC (competição espacial) seguido da operação de limiar com o nível v_{th3} remove segmentos fracos (residuais) que permaneceram do estágio de filtragem anterior, reforçando os segmentos ao longo da correta orientação. A convolução com o filtro BG (agrupamento por bipolo) permite conectar segmentos de contorno que devem ser unidos no mesmo contorno. Esses segmentos possivelmente foram separados por causa das leituras defeituosas feitas pelo sistema visual precoce. Os resultados da operação BG reforçam os resultados da operação SC e vice-versa. A saída do BCS é obtida ao serem somadas as saídas de todos os canais SC. 70

5.5 Resultado das operações de BCS (as duas colunas a direita) e DFI (as duas colunas a esquerda) em 4 diferentes instantes de tempo (10, 20, 30 e 40 mili-segundos). A primeira e a terceira colunas se referem aos resultados da implementação pulsada e as outras, aos resultados da implementação em modo de amplitude. 74

5.6 Erro médio quadrático entre os resultados da operação DFI nas implementações em modo pulsado e em modo de amplitude. 74

Lista de Tabelas

2.1 Parâmetros utilizados para o projeto da OPL. 17

3.1 Latências máximas (em ns) associadas ao processamento dos elementos de três diferentes técnicas AER (AB, WTA e NA) para um total $(X \cdot Y) = 50^2$ e $(X \cdot Y) = 100^2$ neurônios. 39

3.2 Complexidade de *hardware*, com valores dados em transistores (ou em bits para o canal), envolvida com os elementos de três diferentes técnicas AER, AB ([33]), WTA ([34]) e NA ([35]). $X \cdot Y = 50^2$ e 100^2 neurônios. 41

3.3 Sistema AER não arbitrado proposto. São mostrados: latência (em ns), probabilidade de colisão e complexidade de *hardware* (em transistores ou em bits no caso do canal de comunicação). Esses dados são comparados aos dados produzidos a partir das técnicas AER AB [33] e NA [35]. O número total de neurônios foi de $X \cdot Y = 50^2$ e $X \cdot Y = 100^2$. 48

3.4 Comparativo das taxas máximas de transferência entre as técnicas AER proposta (NAP) e duas outras, AB e WTA. A técnica proposta atinge quase o dobro da taxa máxima de transferência da técnica AB e se aproximou do resultado produzido pela técnica WTA. Os valores de taxa máxima de transferência para as técnicas WTA e AB foram retirados de [32] e [34]. As condições de simulação impostas para o NAP foram semelhantes àquelas que produziram os resultados da técnica AB e da WTA. 49

4.1 Caraterísticas da simulação elétrica do sistema AER proposto processando um estímulo em diagonal. Nesse caso a OPL foi desabilitada possibilitando a comunicação apenas pelo canal *On*. 57

Lista de Abreviaturas

I_D	Drain Current, p. 13
I_F	Forward Current, p. 13
I_R	Reverse Current, p. 13
V_P	Pinch-Off Voltage, p. 14
V_T	Threshold Voltage, p. 12
i_D	Normalized Drain Current, p. 13
i_F	Normalized Forward Current, p. 13
i_R	Normalized Reverse Current, p. 13
AB	Árvore Binária, p. 31
BCS	Boundary Contour System, p. 70
BP	Bipole Grouping, p. 71
CD	Células de Decisão, p. 32
CMOS	Complementary Metal-Oxide-Semiconductor, p. 7
Cpd	Cycles per Degree, p. 10
DFI	Diffusive Filling-In, p. 4
DF	Directional Filtering, p. 70
EKV	Enz, Krummenacher, and Vittoz, p. 13
ER	Event Representation, p. 19
EVS	Early Visual System, p. 5
Enc	Encoder Cell, p. 36

FD	Fully Dedicated, p. 4
IC	Inversion Coefficient, p. 13
IF	Integrate-and-Fire, p. 19
IPL	Inner Plexiform Layer, p. 5
LIF	Leakage Integrate and Fire, p. 64
MOS	Metal-Oxide-Semiconductor, p. 4
MRP	Mecanismo de Redução de Probabilidade, p. 43
NAND	Negative AND, p. 32
NAP	Não Arbitrado Proposto, p. 48
NA	Não Arbitrado, p. 39
OPL	Outer Plexiform Layer, p. 5
RNA	Redes Neurais Artificiais, p. 1
SA	Shared Axons, p. 4
SC	Spatial Competition, p. 71
SD	Shared Dendrite, p. 78
SS	Shared Synapse, p. 78
SoC	System on Chip, p. 26
Spk/s	Spikes por Segundo, p. 10
TDM	Time-Division-Multiplexing, p. 20
WTA	Winner Takes All, p. 32

Capítulo 1

Introdução

No sistema visual biológico a luz entra pelos olhos, a informação luminosa é convertida em um sinal elétrico, processada por um conjunto de camadas de neurônios na retina e é transmitida ao córtex visual por meio de um canal conhecido como nervo óptico. O estágio de processamento anterior ao córtex visual é conhecido como sistema visual precoce ou *Early Vision System* (EVS). Esse conjunto de camadas de processamento permite reduzir a informação redundante presente no sinal visual bem como alterar a representação da informação de níveis de intensidade para uma representação onde a informação está relacionada aos intervalos de tempos demarcados por uma sequência de eventos, pulsos elétricos, gerados pelos neurônios que estão conectados ao nervo óptico. O nervo óptico consiste de milhares de canais paralelos que conectam as saídas dos neurônios da última camada de processamento do EVS as entradas dos neurônios no córtex visual. No córtex visual circuitos corticais compostos por milhões de neurônios conectados por um número maior ainda de conexões sinápticas realizam diversas tarefas de processamento sobre esse sinal pulsado. Como exemplo, podemos citar o reconhecimento de padrões visuais e a estimativa da distância de objetos. O estudo dos circuitos corticais possibilita a compreensão de como o cérebro realiza tarefas de processamento de maneira paralela, distribuída e utilizando níveis muito baixos de energia. Cerca de 1% da energia consumida por sistemas convencionais de processamento para realizar uma tarefa equivalente [1], [2]. Esses estudos servem também de inspiração para o desenvolvimento de novos sistemas de processamento. Redes neurais artificiais (RNA) são exemplos de sistemas de processamento inspirados no funcionamento do cérebro. No estudo dos circuitos corticais experimentos sobre a estrutura fisiológica do cérebro são realizados, hipóteses são levantadas, modelos baseados nessas hipóteses são criados e simulações permitem que esses modelos sejam checados e compreendidos. O custo computacional necessário para a simulação de circuitos corticais, utilizando sistemas de processamento convencionais, se torna proibitivo à medida que esses modelos se tornam mais semelhantes aqueles encontrados na biologia [3]. A simula-

ção de modelos de circuitos corticais pode ser cerca de 9000 vezes mais lenta do que o processamento em tempo real realizado pelos circuitos corticais biológicos, caso os sinais de entrada e saída dos neurônios sejam considerados como sequências de pulsos que codificam a informação (modo pulsado ou *spike-mode*), como acontece na biologia [4]. As primeiras gerações de RNAs possuem sinais com valores escalares nas entradas e saídas dos neurônios (modo de amplitude) que representam as taxas com que os pulsos elétricos são transmitidos através da rede neural. Essa é uma estratégia para reduzir o custo computacional envolvido com as simulações das redes neurais visto que na época o poder computacional era muito limitado. Em estudos onde interessam as propriedades pulsadas dos neurônios o modo de amplitude não pode ser utilizado e o desenvolvimento de sistemas de processamento que fazem uso de arquiteturas de processamento semelhantes aquelas apresentadas pelo cérebro são imprescindíveis [5]. Esses sistemas de processamento dedicados contam com *hardware* específico que foge às arquiteturas tradicionalmente imaginadas por Turing e Neumann [6], [7] para os sistemas de computação que são utilizados nos dias atuais. Esse *hardware* específico segue uma arquitetura parecida com aquela que é apresentada pelos circuitos corticais biológicos, como por exemplo, o projeto *SpiNNaker* [8].

1.1 Motivações

A computação analógica, por explorar as propriedades físicas dos dispositivos semicondutores para realizar processamento, leva vantagem sobre a computação digital em relação à complexidade de *hardware* e o consumo de energia. Por exemplo, é muito mais econômico, em termo de *hardware*, representar uma quantidade a partir de um nível utilizando grandezas como corrente ou tensão, do que representar a mesma quantidade por um conjunto de sinais onde cada sinal é uma variável binária. Operações realizadas entre quantidades que são representadas por níveis podem ser realizadas em regimes de muito baixa corrente ou muito baixa tensão, evitando consumo de energia desnecessário [1]. Contudo, a computação analógica baseada na representação de quantidades por níveis é susceptível a ruído e ao descasamento no processo de fabricação. Na área de microeletrônica o descasamento no processo de fabricação indica o quanto as dimensões de transistores MOS iguais variam devido a imperfeições do material ao serem posicionados em locais diferentes. Por explorar propriedades físicas, que muitas vezes são fixas, sistemas analógicos de computação não são muito flexíveis à programação. Dessa maneira, criar sistemas de computação analógica que substituam os tradicionais computadores digitais [7] ainda representa um enorme desafio para os projetistas. Olhando por outro ponto de vista, os circuitos corticais biológicos realizam operações de processamento de maneira analógica,

paralela, distribuída e ainda apresentam flexibilidade de programação (plasticidade neural) [1], [4]. O desenvolvimento de um sistema de processamento que realize computação analógica baseado no funcionamento do cérebro é uma motivação para o estudo dos circuitos corticais.

1.2 Problemática

Apesar de toda uma expectativa por esses novos sistemas de computação estes estão longe de se tornarem realidade e não passam de um sonho futurista. No estágio atual da pesquisa existem outras preocupações. Uma dessas preocupações está relacionada a formas de se realizar o estudo dos circuitos corticais por meio de simulações. Como foi dito, esse estudo deve ser conduzido por meio de um *hardware* dedicado que permite simulações em modo pulsado. Previsões teóricas devem ser consideradas antes da implementação desse *hardware* dedicado. Essas previsões teóricas fornecem bases conclusivas sobre as especificações do projeto do *hardware* dedicado, como por exemplo, o tipo de neurônio a ser utilizado em determinado modelo do circuito cortical. Existem algumas formas de se fazer previsões teóricas sobre os circuitos corticais e assim obter resultados que mostram seu comportamento. O modo pulsado não pode ser substituído pelo modo de amplitude caso se esteja fazendo um estudo das propriedades pulsadas do circuito cortical. Contudo, é possível que os resultados de previsão teórica feitos em modo de amplitude tenham serventia para serem utilizados para dimensionamento do *hardware* dedicado [9], [10]. No modo de amplitude a sequência de pulsos nas entradas e na saída dos neurônios é associada a uma sequência de valores escalares que representam a taxa de disparo dos neurônios dentro de uma janela temporal. Também é possível utilizar computadores de muito alto desempenho [11], caso estes estejam disponíveis, para realizar essas previsões direto em modo pulsado. Existem casos onde às previsões não são consideradas, os detalhes de especificação do projeto são ignorados e o que é visto é um superdimensionamento no projeto do *hardware* dedicado [5]. Entre os três casos apresentados (o modo de amplitude, o uso de computadores de alto desempenho e o superdimensionamento do *hardware* dedicado), a opção mais acessível é o modo de amplitude. Utilizar o modo de amplitude, uma sequência de valores escalares representando um sinal composto por uma sequência de pulsos, é uma abordagem que desconsidera importantes características da dinâmica neural, como por exemplo, a influência dos pulsos do sinal (*spikes*) nos laços de realimentação, a influência dos *spikes* na combinação linear das entradas, o exato comportamento do neurônio logo após ele disparar e a influência do formato do *spike* na resposta do sistema. Assim, não se sabe ao certo se o modo de amplitude realmente pode fornecer resultados próximos àqueles que se obteriam caso pudéssemos utilizar sinais compostos por sequências de *spikes* nas entradas e

saídas dos neurônios (modo pulsado) mesmo se quiséssemos usar esses dados para dimensionar o *hardware* dedicado. O desenvolvimento de *hardware* para o processamento pulsado não é trivial e muitos problemas surgem ao se tentar desenvolver esse tipo de dispositivo. Implementações em *hardware* envolvendo dimensões tão grandes, milhares de neurônios conectados por milhões de conexões sinápticas, podem não estar disponíveis caso uma topologia de projeto semelhante àquela apresentada pelo cérebro seja utilizada para implementar os circuitos corticais. A topologia que imita a utilizada pelo cérebro é conhecida como completamente dedicada (*Fully Dedicated* - FD). A topologia FD é aquela onde existe um dispositivo físico representando cada elemento a ser implementado do circuito cortical [4]. Por exemplo, para cada neurônio implementado em *hardware* existe um dispositivo MOS (*Metal-Oxide-Semiconductor*) que o representa. O mesmo se aplica às sinapses, dendritos e outros elementos da rede neural. Uma maneira de se contornar esse problema é utilizar topologias onde elementos da rede neural sejam compartilhados. Como por exemplo, o dispositivo físico que implementa um axônio pode ser utilizado por mais de um elemento físico que implementa um neurônio. A essa topologia chamamos de axônio compartilhado (*Shared Axons* - SA). Contudo, topologias como a SA são baseadas em troca de informação por multiplexação no tempo, o que causa atrasos aleatórios no sinal a ser transmitido. No caso de um sistema onde a informação está relacionada ao intervalo de tempo entre pulsos consecutivos, como é o caso dos circuitos neurais, esses atrasos distorcem a informação. A latência [12] é uma medida relacionada a distorção causada por esses atrasos aleatórios ao sinal. Saber como a latência inserida por topologias de redução da quantidade de circuitos (redução da complexidade de *hardware*) como a SA influenciam o comportamento de um dispositivo que implementa um circuito cortical é uma questão importante.

1.3 Objetivos

Nosso objetivo é desenvolver um diagrama esquemático de uma retina de silício que será composta pelos estágios de processamento do EVS e implementar o algoritmo do *diffusive filling-in* (DFI) em dois modos, em modo pulsado e em modo de amplitude. O DFI é um mecanismo do córtex visual biológico ligado à reconstrução de imagens [13]. Esse circuito cortical foi escolhido por ser simples o bastante para ser implementado tanto em modo de amplitude [14], como em modo pulsado [9]. Os dois modos de implementação possibilitarão responder se o modo de amplitude pode ser utilizado para gerar resultados de previsão teórica capazes de auxiliar no dimensionamento de um *hardware* dedicado. Para tornar os resultados obtidos com as simulações do algoritmo do DFI mais realistas iremos adotar uma abordagem de tratamento dos estímulos diferente daquela que é utilizada tradicionalmente a

partir de filtros de diferenças-de-gaussianas (*Difference-of-Gaussians* - DoG). Nessa abordagem a resposta da retina é aproximada por uma filtragem com filtros DoG. Na nossa abordagem utilizaremos o diagrama esquemático da retina para fazer esse tratamento. Um ruído característico da filtragem passa-banda espacial da retina é considerado no processamento, tornando a abordagem proposta mais realista. Esse ruído característico realça as diferenças entre as duas simulações e permite testar a previsão em modo de amplitude em condições mais adversas. No modelo elétrico da retina iremos introduzir um sistema de comunicação que utiliza topologia SA. Esse sistema de comunicação, que foi proposto nesse trabalho, apresenta índices de latência menores que os sistemas de comunicação apresentados tradicionalmente, mantendo a complexidade de *hardware* equivalente. Essa abordagem é importante porque permite avaliar os efeitos causados pela técnica de redução da complexidade de *hardware* que será utilizada.

1.4 Metodologia de Trabalho

No Cap. 2, faremos uma revisão sobre o sistema visual precoce ou *Early Visual System* (EVS) . A partir dessa revisão iremos apresentar um modelo para cada camada de processamento do EVS, *Outer Plexiform Layer* (OPL), *Inner Plexiform Layer* (IPL) e camada das células ganglionares. Uma metodologia será proposta para projetar a camada da OPL, a partir de transistores MOS. Resultados sobre o funcionamento da metodologia de projeto da OPL e do funcionamento da IPL são apresentados. Os resultados mostram que tanto o modelo elétrico da OPL quanto o da IPL respondem de acordo com o esperado. No Cap. 3 é proposto um sistema de comunicação, utilizando topologia SA, que substitui o nervo óptico. Esse sistema de comunicação permite reduzir fisicamente as conexões feitas entre as saídas da retina e as entradas dos circuitos corticais do algoritmo DFI de n^2 para $\log_2(n)$, apresentando níveis de latência menores que aqueles dos sistemas de comunicação utilizados tradicionalmente e mantendo a complexidade de *hardware* equivalente. São mostrados resultados relacionados a latência e a complexidade de *hardware* do sistema de comunicação. Esses resultados comprovam que o sistema de comunicação proposto apresenta latência menor que o sistema que costuma ser utilizado e uma complexidade de *hardware* equivalente. O diagrama esquemático da retina utilizando o sistema de comunicação proposto é apresentado no Cap. 4. A resposta ao impulso do diagrama esquemático da retina e exemplos de imagens reconstruídas são apresentados. No Cap. 5 introduziremos a ideia de processamento em modo pulsado e de como funciona a dinâmica neuronal. Apresentaremos um modelo do algoritmo DFI que será utilizado nesse trabalho. Faremos duas implementações do algoritmo DFI, uma em modo de amplitude [15] e outra em modo pulsado [9]. Mostraremos exem-

plos de utilização dessas implementações e o esquema de processamento utilizado por elas. Os resultados das simulações feitas com o modelo do algoritmo DFI são apresentados. Esses resultados mostram também que as previsões feitas em modo de amplitude são muito próximas as previsões feitas em modo pulsado apesar das distorções (latência e filtragem passa-banda espacial da retina) inseridas no sinal de estímulo utilizado como entrada do algoritmo DFI. No Cap. 6 concluímos que os resultados apresentados mostram que a metodologia de projeto para a OPL funciona, que o sistema de comunicação é capaz de substituir os sistemas tradicionalmente utilizados apresentando vantagens em relação a latência e a complexidade de *hardware*. Apresentamos indícios de que o modo de amplitude é capaz de fazer previsões próximas àquelas que seriam feitas utilizando o modo pulsado para o DFI.

Capítulo 2

O EVS e os Estímulos para o DFI

Serão introduzidos a seguir os conceitos básicos que permitirão o desenvolvimento do modelo elétrico da retina (diagrama esquemático), avaliado para implementação CMOS (*Complementary Metal-Oxide-Semiconductor*) [16]. O modelo elétrico da retina possui três estágios de processamento, a OPL, a IPL e a camada das células ganglionares. Um diagrama em bloco desse esquema, que também é conhecido como EVS, pode ser visto na Fig. 2.1.

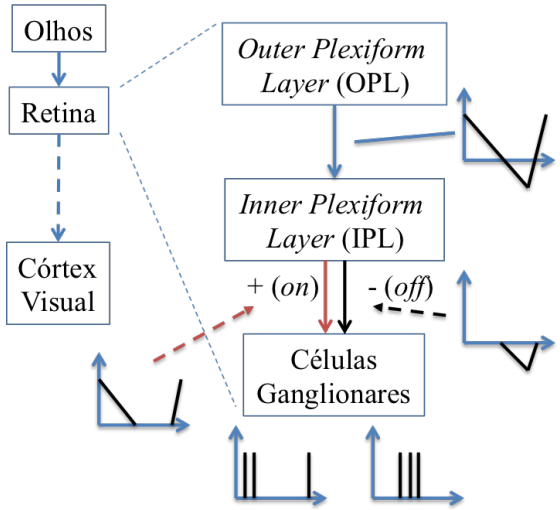


Figura 2.1: Diagrama em blocos do EVS. É apresentada a OPL, a IPL, a camada de células ganglionares e os respectivos sinais em suas entradas e saídas.

O sistema visual biológico possui ainda um estágio de transmissão de informações da retina até o córtex, conhecido como nervo óptico. Nesse capítulo serão apresentados o modelo elétrico da OPL [16], o modelo elétrico da IPL (sistema retificador [17]), e o modelo das células ganglionares [16]. Falaremos sobre o sistema de comunicação, que corresponde ao nervo óptico, no Cap. 3. O sistema de comunicação é baseado na topologia SA e permite redução na complexidade de *hardware* para uma futura implementação do modelo elétrico da retina em silício. O diagrama esquemático

da retina desenvolvido será utilizado para auxiliar no processamento dos estímulos que serão utilizados como entradas para o DFI. No Cap. 4 será apresentada uma discussão sobre os motivos que mostram que essa abordagem é mais realista do que a que tradicionalmente é utilizada com filtros DoG. Serão introduzidos os fundamentos teóricos apresentando o modelo elétrico da OPL.

2.1 Modelo para a OPL

A OPL biológica é uma camada de processamento do EVS que é formada por dois tipos de neurônios, os fotorreceptores e as células horizontais. Os fotorreceptores transduzem a intensidade luminosa da informação visual em sinal elétrico. Estudos fisiológicos revelam que os fotorreceptores são interconectados por meio de junções elétricas resistivas que permitem a difusão da corrente elétrica produzida em um dos fotorreceptores para os seus respectivos vizinhos [18]. O mesmo conceito se aplica às células horizontais. Essa estrutura composta de neurônios conectados por conexões resistivas permite espalhamento lateral de corrente tanto na rede de fotorreceptores como na rede das células horizontais. Existem conexões entre os fotorreceptores e as células horizontais que permitem difusão de corrente elétrica entre esses dois grupos de neurônios [19], [20]. Essas interações permitem uma filtragem passa-banda no domínio do espaço e do tempo do sinal transduzido que ocorre nesse estágio. Essa é uma estratégia biológica que visa reduzir a informação redundante no sinal visual. O modelo da OPL apresentado em [21] associa as difusões de corrente que ocorrem entre os fotorreceptores ou entre células horizontais na OPL as interações elétricas que ocorrem entre os elementos infinitesimais de um cabo elétrico. A Fig. 2.2 mostra um exemplo de um cabo elétrico e as correntes elétricas conduzidas por ele.

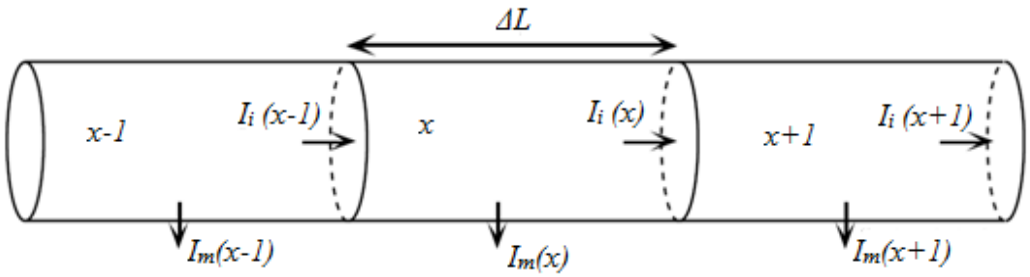


Figura 2.2: Em um cabo dividido em elementos infinitesimais (x), de comprimento ΔL, existe uma corrente que flui horizontalmente pelo cabo ($I_i(x)$) e uma corrente que é conduzida verticalmente pelo elemento infinitesimal ($I_m(x)$).

No cabo elétrico os elementos infinitesimais estão divididos horizontalmente e x representa a posição na qual o elemento infinitesimal se encontra. A corrente

lateral entre os elementos infinitesimais é indicada pelo símbolo I_i e a corrente que é conduzida verticalmente por cada elemento infinitesimal é indicada pelo símbolo I_m. O modelo elétrico desse cabo é mostrado na Fig. 2.3. Os elementos infinitesimais do cabo são formados por capacitâncias (C_m) em paralelo com uma condutância (g_m). Esses elementos são conectados entre si por meio de resistores horizontais (R_i) permitindo o espalhamento lateral de corrente. V_i é uma tensão associada ao elemento infinitesimal do cabo. $I(x)$ é a corrente que flui através dos elementos infinitesimais de condutância g_m.

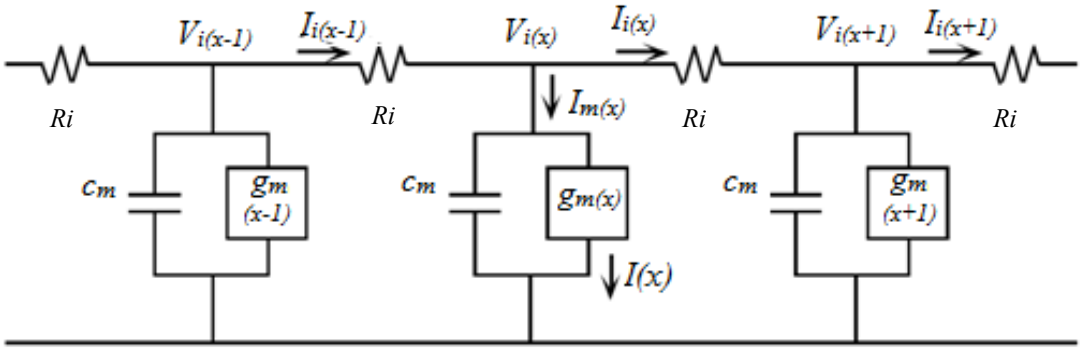


Figura 2.3: Os componentes do modelo elétrico de um cabo são apresentados. Os elementos infinitesimais do cabo (capacitâncias e condutâncias) são conectados a partir de resistores (R_i).

Como o modelo elétrico de um neurônio é um capacitor e uma condutância conectados em paralelo [21], [22] foi possível associar o modelo elétrico do elemento infinitesimal do cabo ao modelo do neurônio [23]. É possível vincular as conexões resistivas que permitem interação entre os neurônios na OPL biológica aos elementos resistivos que aparecem no modelo elétrico do cabo. Após essas duas associações é possível comparar o modelo elétrico do cabo a um conjunto de neurônios da OPL interconectados lateralmente por meio de junções elétricas resistivas. Apresentadas essas considerações seguiremos com a descrição do comportamento do modelo da rede de fotorreceptores e da rede de células horizontais por meio da Eq. (2.1), equação que descreve o comportamento de um cabo elétrico.

$$V_i(x,t) = \lambda^2 \cdot \frac{\partial^2 V_i(x,t)}{\partial x^2} - \tau \cdot \frac{\partial V_i(x,t)}{\partial t} \tag{2.1}$$

A Eq. (2.1) é uma variação da equação do cabo elétrico utilizada em [23] com $\lambda^2 = \frac{1}{R_i \cdot g_m(x)}$ e $\tau = \frac{c_m}{g_m(x)}$. Ao ser inserido um estímulo $I_p(x,t)$ na posição x do cabo e considerando que esse estímulo é um impulso no espaço e no tempo, temos que a Eq. (2.1) se torna:

$$V_i(x,t) = \lambda^2 \cdot \frac{\partial^2 V_i(x,t)}{\partial x^2} - \tau \cdot \frac{\partial V_i(x,t)}{\partial t} + \frac{1}{g_m} \cdot I_p(x,t) \tag{2.2}$$

A transformada de Fourier de $V_i(x,t)$, na Eq. (2.2), pode ser descrita como sendo $V_i(u,w)$, na Eq. (2.3). Onde $i = \sqrt{-1}$.

$$V_i(u,w) = \frac{I_p(u,w)}{g_m} \cdot \frac{1}{\lambda^2 \cdot u^2 + \tau \cdot i \cdot w + 1} \tag{2.3}$$

A resposta ao impulso do modelo elétrico do cabo é mostrado na Eq. (2.1). Assim foi possível modelar o comportamento das redes de fotorreceptores e de células horizontais utilizando a Eq. (2.1). No sistema visual cada célula horizontal recebe do fotorreceptor de mesma posição uma corrente proporcional a sua atividade. Por sua vez, cada célula horizontal drena um valor de corrente proporcional à sua atividade desse mesmo fotorreceptor. Será assumido aqui que $V_c(x,t)$ é o sinal que corresponde a atividade dos fotorreceptores e que $V_h(x,t)$ é o sinal que corresponde a atividade das células horizontais. Em termos de modelo é possível descrever esse comportamento a partir do sistema de equações apresentado na Eq. (2.4).

$$\begin{cases} V_c(u,w) = \frac{I_p(u,w) - I_h(u,w)}{g_{m_c}} \cdot \frac{1}{\lambda_c^2 \cdot u^2 + \tau_c \cdot i \cdot w + 1}, \\ V_h(u,w) = \frac{I_c(u,w)}{g_{m_h}} \cdot \frac{1}{\lambda_h^2 \cdot u^2 + \tau_h \cdot i \cdot w + 1}, \end{cases} \tag{2.4}$$

I_c é uma corrente proporcional a V_c e I_h é uma corrente proporcional a V_h. Em [16] foi proposto um modelo elétrico da OPL, Fig. 2.4, baseado no modelo apresentado na Eq. (2.4).

Utilizando os mesmos procedimentos que possibilitaram escrever o sistema de equações apresentado na Eq. (2.4) foi possível descrever o sistema de equações apresentado na Eq. (2.5) para o modelo apresentado na Fig. 2.4.

$$\begin{cases} V_c(u,w) = \frac{I_p(u,w)}{g_{ch}} \cdot \frac{e_h + l_h^2 \cdot u^2 + i \cdot \tau_h \cdot w}{(e_c + l_c^2 \cdot u^2 + i \cdot \tau_c \cdot w)(e_h + l_h^2 \cdot u^2 + i \cdot \tau_h \cdot w) + 1}, \\ V_h(u,w) = \frac{g_{hc} \cdot V_c(u,w)}{g_{ch}} \cdot \frac{1}{(e_c + l_c^2 \cdot u^2 + i \cdot \tau_c \cdot w)(e_h + l_h^2 \cdot u^2 + i \cdot \tau_h \cdot w) + 1}, \end{cases} \tag{2.5}$$

onde $l_c = \frac{1}{\sqrt{R_{cc} \cdot g_{ch}}}$, $l_h = \frac{1}{\sqrt{R_{hh} \cdot g_{hc}}}$, $\tau_c = \frac{c_{c0}}{g_{ch}}$, $\tau_h = \frac{c_{h0}}{g_{hc}}$, $e_c = \frac{g_{c0}}{g_{ch}}$ e $e_h = \frac{g_{h0}}{g_{hc}}$.

Em [21] foi possível utilizar a resposta em frequência desse sistema de equações para ajustar a curva formada pelos valores medidos da resposta em frequência da OPL em experimentos fisiológicos na retina de gatos. Os dados do experimento fisiológico mostram a característica de resposta passa-banda espacial e temporal da OPL. O pico espacial é próximo de 1 ciclo por grau (Cpd) e o pico temporal é de aproximadamente 10 Hz para a resposta da OPL. Ambos os picos atingem uma taxa de disparo máxima de 100 spikes por segundo (Spk/s). Ao derivarmos V_h na Eq. (2.5) em relação a u, para uma frequência temporal muito baixa ($w \approx 0$), foi possível encontrar onde a resposta é máxima (frequência espacial de pico u_p) igualando a expressão resultante da derivada a zero. Com esse procedimento foi possível obter a expressão mostrada na Eq. (2.6).

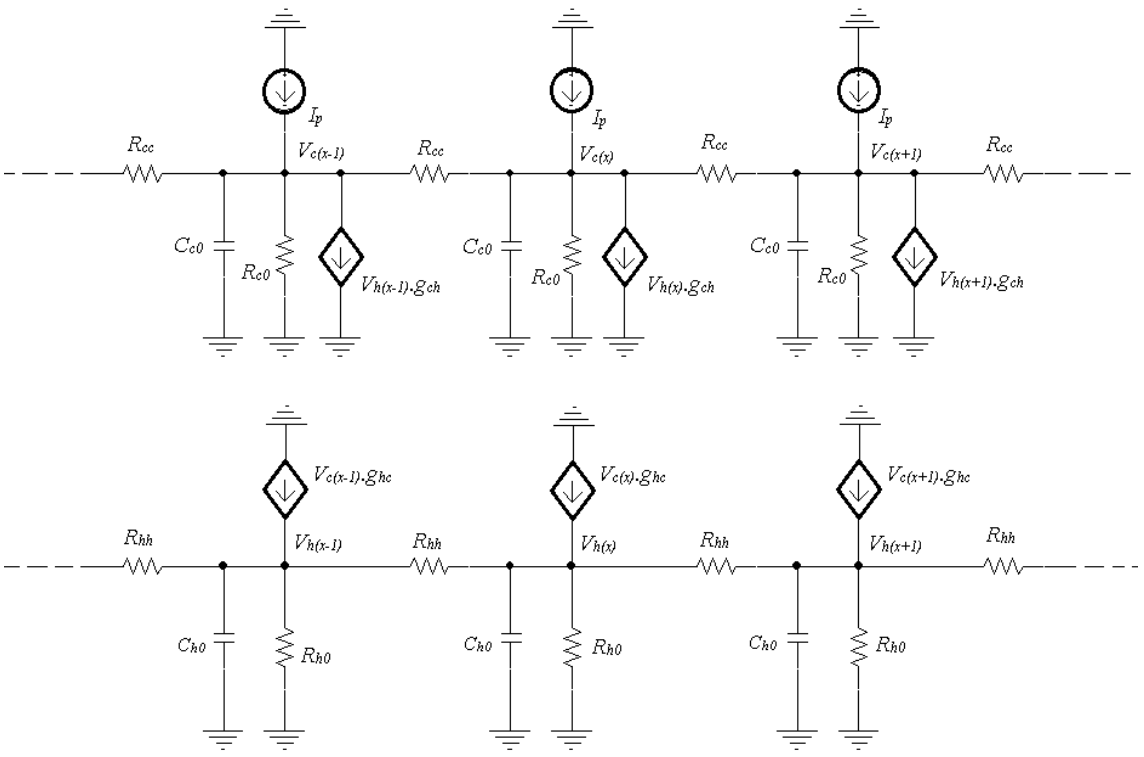


Figura 2.4: Na parte superior da figura são mostrados o modelo elétrico de três fotorreceptores que compõem o modelo da rede de fotorreceptores, na figura da parte de baixo o modelo elétrico de três neurônios da rede de células horizontais. Fontes de corrente controlada por tensão (tensões $V_c(x)$ e $V_h(x)$) implementam as interações que ocorrem entre as duas redes.

$$u_p = \frac{\sqrt{R_{cc} \cdot g_{ch}}}{l_h \cdot l_c} \tag{2.6}$$

De maneira semelhante, considerando que um estímulo possui uma frequência espacial muito baixa, $u \approx 0$, e assim encontrar uma expressão para a frequência temporal de pico (w_p). Ao assumir que $e_h \cdot e_c \ll 1$ foi possível aproximar a frequência temporal de pico pela expressão dada pela Eq. (2.7).

$$w_p \approx \frac{1}{\sqrt{\tau_c \cdot \tau_h}} \tag{2.7}$$

Os picos das respostas em frequência da Eq. (2.5) (Eqs. (2.6) e (2.7)) podem ter seus valores ajustados conforme os valores dos parâmetros R_{cc}, R_{hh}, C_{c0}, C_{h0}, g_{ch} e g_{hc}. Uma busca pelos valores dos parâmetros R_{cc}, R_{hh}, C_{c0}, C_{h0}, g_{ch} e g_{hc} que tornam os picos de frequência espacial e temporal iguais a 10 Cpd e 10 Hz, respectivamente, foi feita. Utilizaremos esse resultado de previsão teórica para comparar os resultados obtidos por simulação elétrica do diagrama esquemático da retina que será desenvolvido. Como nesse diagrama foi utilizado uma quantidade de 50 por 50 fotorreceptores será adotada essa mesma dimensão para a previsão teórica feita aqui.

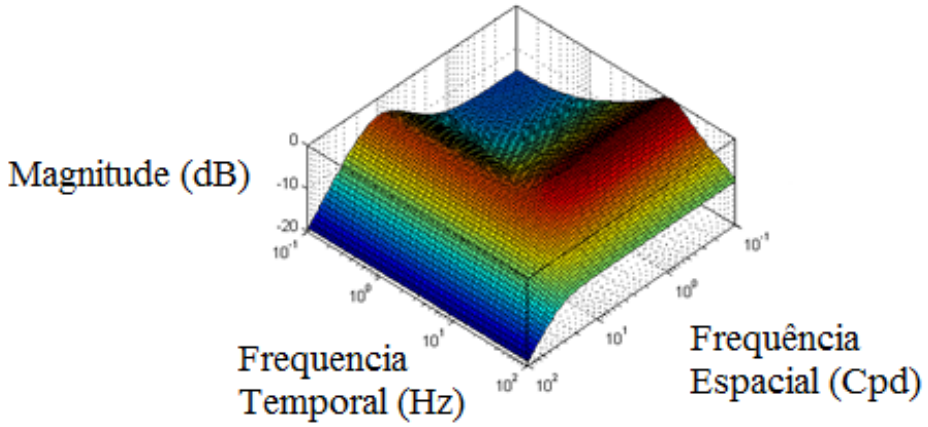


Figura 2.5: Os parâmetros que definem a resposta da Eq. (2.5) foram ajustados para que os picos em frequência espacial e temporal fossem de 10 Cpd e 10 Hz. O objetivo era o ajuste para 1 Cpd e 10 Hz, contudo, devido a limitações na quantidade de pontos no espaço da nossa simulação o valor de 1 Cpd teve que ser escalado para 10 Cpd.

A quantidade de fotorreceptores (50) que foi utilizada na simulação não permitiu uma resolução espacial suficiente para que fosse possível uma resposta em frequência espacial de 1 Cpd, assim optou-se por escalar o valor do pico de frequência espacial para 10 Cpd. O resultado dessa busca, o valor dos parâmetros, foi utilizado para plotar a superfície da Fig. 2.5.

2.1.1 A OPL em uma Implementação CMOS

Dada a fundamentação teórica sobre o modelo da OPL será mostrado como é a implementação CMOS das redes de resistores e fontes de corrente que foram utilizadas para modelar a OPL. Nesse tipo de implementação costuma-se substituir resistores e fontes de corrente por transistores MOS [24]. Para possibilitar tais implementações regimes de operação adequados devem ser impostos aos transistores MOS, contudo para alcançar níveis de consumo de energia muito baixos os transistores MOS costumam funcionar em um regime de operação conhecido como *subthreshold* [25], [26]. O regime de operação de *subthreshold* ou de inversão fraca é estabelecido em um transistor MOS quando a tensão de polarização de porta tem seu valor ajustado abaixo da tensão de limiar V_T (*threshold voltage*). Outros dois regimes, o de inversão moderada e forte, podem ser impostos para o funcionamento dos transistores MOS e esses regimes também estão associados ao valor da tensão de polarização de porta em relação a V_T. Para implementar transistores que funcionam em dife-

rentes regimes de operação os projetistas de circuito integrado costumam utilizar diferentes equações. Cada regime de operação é considerado separadamente e, tradicionalmente, não estão disponíveis equações para o regime de operação moderado. O modelo Enz, Krummenacher e Vittoz (EKV) promove uma compreensão simples das propriedades dos transistores MOS operando em todos os regimes de operação sem a necessidade de mais do que uma equação [27]. Usando o modelo EKV [27] o projetista pode escolher um regime de operação, auxiliado pelo coeficiente de inversão, para um transistor MOS de modo a alcançar características tais como a razão transcondutância pela corrente de dreno ($\frac{g_m}{I_D}$). O coeficiente de inversão (*Inversion Coefficient* - IC) é definido pela Eq.(2.8):

$$\mathrm{IC} = \frac{I_D}{I_0 \cdot (\frac{W}{L})} = \frac{I_D}{2 \cdot \mu \cdot \eta \cdot C_{OX} \cdot \phi_T^2 \cdot (\frac{W}{L})} \tag{2.8}$$

onde I_D é a corrente de dreno (*Drain Current*), I_0 é a corrente de parâmetro da tecnologia, μ é o parâmetro de mobilidade do semicondutor, η é o *slope factor*, C_{OX} é a capacitância de óxido por unidade de área, ϕ_T é a voltagem térmica e $\frac{W}{L}$ é a razão entre largura e comprimento do canal em transistores MOS. No modelo EKV, o regime de inversão forte de um transistor MOS é definido por um IC maior que 10, o moderado é definida por um IC entre 0.1 e 10 e fraco definido por um IC menor do que 0.1. Como todas as voltagens de terminais são referenciadas para o substrato local, a simetria fonte-dreno do transistor MOS é mantida e modelos desenvolvidos para operarem com transistores NMOS podem ser alterados para funcionarem com transistores PMOS sem complicação. Nesse trabalho será assumido V_{G0} como sendo a tensão entre porta e terra e V_{GB} como sendo a tensão entre porta e substrato. Similarmente V_{D0}, V_{S0}, V_{DB} e V_{SB} são definidos.

A corrente de dreno I_D pode ser obtida pela superposição da corrente direta I_F ou *forward current* (corrente provocada pela polarização do terminal de fonte) e da corrente reversa I_R ou *reverse current* (corrente provocada pela polarização do terminal de dreno). A expressão para I_D, Eq. (2.9) ou (2.10), costuma apresentar as correntes I_D, I_F e I_R na forma de correntes normalizadas. Essa normalização é feita em relação a $I_0 \cdot \frac{W}{L}$.

$$i_D = \frac{I_D}{I_0 \cdot (\frac{W}{L})} = \frac{I_F}{I_0 \cdot (\frac{W}{L})} - \frac{I_R}{I_0 \cdot (\frac{W}{L})} \tag{2.9}$$

$$i_D = i_F - i_R \tag{2.10}$$

Na Eq. (2.10), i_F e i_R denotam a corrente direta normalizada e a corrente reversa normalizada respectivamente. Nos casos em que i_F é muito maior que i_R, i_D se torna aproximadamente igual a IC. Por simplicidade essa condição costuma ser

assumida de maneira geral. No modelo EKV a utilização da função apresentada na Eq. (2.11) permite definir a corrente de dreno para qualquer nível de inversão [28]. Muitas expressões, as quais são usuais para projetos de circuitos integrados, que são definidas sobre todas as regiões de operação, podem ser desenvolvidas a partir do modelo EKV.

$$i_{F(R)} = (\ln(1 + \exp(\frac{V_P - V_{SB}(V_{DB})}{2})))^2 \tag{2.11}$$

Na Eq. (2.11) caso se queira calcular i_F é preciso considerar V_{SB} e caso se queira calcular i_R, V_{DB} deve ser considerado. $V_P = \frac{(V_{G0} - V_T)}{\eta}$ é a tensão de *pinch-off* . Os nossos modelos EKV consideraram apenas três parâmetros, η, V_T e o parâmetro β que é apresentado na Eq. (2.12):

$$\beta = \mu \cdot C_{OX} \tag{2.12}$$

Os três parâmetros (η, V_T e β) são encontrados para a tecnologia na qual será pretendido implementar a OPL a partir do seguinte procedimento. Foi realizada uma busca pelo valor de f mais próximo de zero na Eq. (2.13).

$$f(\mu, \beta, V_T) = I_{SIM} - i_F \cdot I_0 \cdot \frac{W}{L} \tag{2.13}$$

I_{SIM} é a corrente simulada de dreno na tecnologia que será utilizada na implementação e o produto ($i_F \cdot I_0 \cdot \frac{W}{L}$) é a corrente de dreno prevista pelo modelo EKV. Na busca, os valores iniciais dos parâmetros η, V_{To} e β são obtidos a partir dos valores típicos (nominais) apresentados no manual da tecnologia MOS (0.35 μm da AMS). No regime de inversão fraca a corrente de dreno I_D toma a forma descrita pela Eq. (2.14):

$$I_D = I_S \cdot e^{\frac{V_{GB} - V_{To}}{\eta \cdot v_T}} \cdot (e^{\frac{V_{DB}}{v_T}} - e^{\frac{V_{SB}}{v_T}}), \tag{2.14}$$

e assim uma relação aproximadamente linear é estabelecida entre a corrente de dreno I_D e uma função exponencial da tensão de dreno (V_D) em relação à tensão de substrato (V_B), V_{DB}. Essa função exponencial é definida como sendo a pseudo-tensão de dreno V^*_{DB} mostrada na Eq. (2.15).

$$V^*_{DB(SB)} = V_0 \cdot e^{\frac{V_{DB(SB)}}{\phi_T}} \tag{2.15}$$

Devido a essa relação nesse regime o transistor MOS funciona como sendo um tipo de resistor que aqui será chamado de pseudo-resistor R^*_{DS}. A Eq. (2.16) descreve o comportamento de $G^*_{DS} = \frac{1}{R^*_{DS}}$.

$$G^*_{DS} = \frac{I_S}{V_0} \cdot e^{\frac{V_{GB}-V_T}{\eta \cdot v_T}}. \tag{2.16}$$

As Eqs. (2.14), (2.15) e (2.16) são combinadas permitindo que seja escrita a Eq. (2.17).

$$I_D = (V^*_{DB} - V^*_{SB}) \cdot G^*_{DS} \tag{2.17}$$

Embora, de fato, não exista uma relação linear entre a corrente de dreno e a diferença de tensão entre dreno e fonte é possível descrever uma relação linear (pseudo-resistência) entre a corrente de dreno e pseudo-tensão entre dreno e fonte. Qualquer rede construída a partir de resistores lineares pode ser convertida em uma rede de pseudo-resistores formados por transistores MOS [28]. Nesse trabalho será assumido que os pseudo-resistores produzirão resultados equivalentes aqueles obtidos ao se implementar uma rede resistiva com resistores lineares. Contudo, uma análise deveria ser feita para avaliar o quanto a implementação dos pseudo-resistores alteram o resultado da rede formada com resistores lineares. A partir do modelo EKV é possível a obtenção de uma expressão para a transcondutância dada pela Eq. (2.18):

$$g_{m_{F(R)}} = \frac{1}{\phi_T} \cdot \frac{2}{\sqrt{1 + 4 \cdot i_{F(R)}} + 1} \cdot i_{F(R)} \cdot I_S \tag{2.18}$$

Existem diversas topologias de OPL CMOS. Como exemplos, temos a apresentada em [16], que realiza filtragem passa-banda espacial e temporal, e a apresentada em [17], que adiciona um mecanismo que previne efeitos de instabilidade que poderiam ocorrer no modelo apresentado em [16], denominado como sendo mecanismo de *auto-feedback*. Por simplicidade Será utilizada a topologia de menor complexidade [16], a apresentada na Fig. 2.6. Essa topologia possui comportamento regido pela Eq. (2.19).

$$\begin{cases} \frac{v_c}{R_{c0}} + C_{c0} \cdot \frac{\partial v_c}{\partial t} + g_{ch} \cdot v_h = I_p + \frac{1}{R_{cc}} \cdot \nabla^2(v_c), \\ \frac{v_h}{R_{h0}} + C_{h0} \cdot \frac{\partial v_h}{\partial t} + g_{hc} \cdot v_c = I_p + \frac{1}{R_{hh}} \cdot \nabla^2(v_h) \end{cases} \tag{2.19}$$

A versão no domínio da frequência da Eq. (2.19) é a Eq. (2.5) [16]. O símbolo $\nabla[.]$ (divergente) representa a operação de derivada parcial em relação à posição. A metodologia que foi utilizada para fazer a implementação da topologia de OPL apresentada em [16] consiste em encontrar os níveis de polarização dos transistores MOS apresentados no diagrama esquemático da Fig. 2.6. Esses níveis de polarização permitem que os transistores operem de acordo com sua função correspondente. Por exemplo, os transistores M1 e M2 devem operar como os pseudo-resistores R_{hh} e R_{cc} respectivamente, os transistores M3, M4 e M5 devem operar como as fontes de

corrente controladas por tensão g_{hc}, uma corrente de *bias* I_u e g_{ch}.

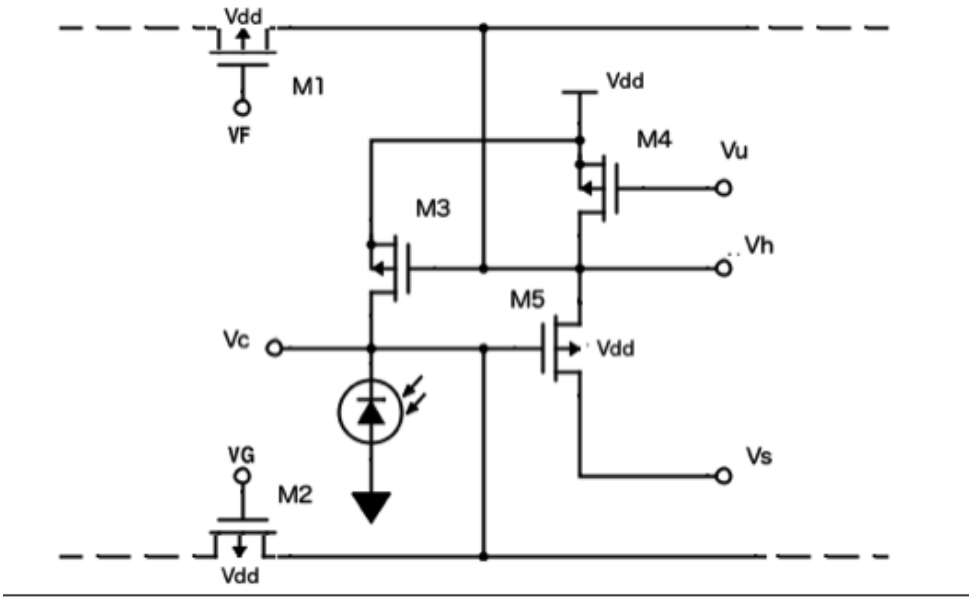


Figura 2.6: Uma unidade da OPL CMOS equivalente a do diagrama esquemático da Fig. 2.4. As fontes de corrente controladas por tensão e os resistores são substituídos por transistores MOS. Esse esquema foi adaptado de Boahen [16].

As tensões de polarização do diagrama esquemático são Vs, VG, VF e Vu. Vh e Vc são os níveis de tensão ajustados pelas tensões de polarização. A saída do circuito é uma corrente proporcional a tensão Vc. Capacitâncias parasitas presentes nos terminais de porta dos transistores MOS substituem os capacitores que originalmente faziam parte da rede de fotorreceptores e células horizontais apresentados na Fig. 2.4. Para encontrar o valor dos parâmetros Vs, VG, VF e Vu os valores de R_{cc}, R_{hh}, g_{ch} e g_{hc} foram assumidos serem os mesmos que foram utilizados para produzir o gráfico apresentado na Fig. 2.5. Quando foi feito o procedimento de cálculo desses parâmetros foram considerados os valores dos capacitores próximos daqueles que são associados aos capacitores parasitas do terminal de porta em um transistor MOS de dimensões W e L próximas as dimensões mínimas. O objetivo foi o de aproveitar esses capacitores parasitas como parte de uma futura implementação da OPL em silício. Os valores de VF e VG puderam ser encontrados a partir da Eq. (2.16) ao substituir G^*_{DS} pelos valores correspondentes aos valores calculados de $\frac{1}{R_{hh}}$ e $\frac{1}{R_{cc}}$ respectivamente. Utilizando a Eq. (2.18) é possível determinar quais correntes devem passar pelos transistores M3 e M5 para que esses tenham valores de transcondutâncias de acordo com os valores dos parâmetros g_{hc} e g_{ch} respectivamente. O transistor M4 possui uma corrente de *bias* definida. A partir da Eq. (2.14) é possível determinar os níveis de polarização para os terminais de fonte, dreno e porta que produzem as correntes que esses três transistores devem conduzir. A tensão de polarização de fonte do transistor M3 é o nível fixo Vdd. O mesmo se

aplica ao transistor M4. Como Vc e Vh não são parâmetros, restam apenas duas tensões de polarização, as tensões Vu e Vs. foi escolhido um nível de polarização contínuo para a tensão Vc. Esse nível contínuo de Vc também irá definir um valor contínuo para Vh. Como a corrente do transistor M3 já foi definida e as tensões de polarização de fonte e dreno também, resta apenas um único valor que satisfaz a condição de funcionamento de M3. Com Vh definido é possível encontrar Vu por meio das condições de funcionamento que satisfazem o transistor M4. O mesmo procedimento se aplica ao transistor M5 para que seja possível encontrar o valor de Vs. Um quadro com os valores dos parâmetros utilizados é mostrado na Tab. 2.1. Esse método é utilizado para projetar uma OPL de frequência espacial de pico de 10 Cpd, frequência temporal de pico de 10 Hz.

Tabela 2.1: Parâmetros utilizados para o projeto da OPL.

Parâmetro	Valor	Corrente
g_{ch}	$1 \cdot 10^{-9}\ \frac{A}{V}$	$\approx 57 \cdot 10^{-12}$ A
g_{hc}	$1 \cdot 10^{-9}\ \frac{A}{V}$	$\approx 57 \cdot 10^{-12}$ A
R_{cc}	$1 \cdot 10^{12}\ \Omega$	≈ 0 A
R_{hh}	$1 \cdot 10^{9}\ \Omega$	≈ 0 A
C_{c0}	$50 \cdot 10^{-15}$ F	≈ 0 A
C_{h0}	$50 \cdot 10^{-15}$ F	≈ 0 A

2.2 Modelo para a IPL

Devido à filtragem passa-banda no espaço e no tempo a resposta da OPL é composta por um sinal que apresenta amostras positivas e negativas [29]. Entre as diversas funções realizadas pela IPL está a de gerar dois outros sinais a partir do sinal resultante da filtragem da OPL. Esses dois sinais são compostos pelas amostras de valor positivo e pelas amostras de valor negativo da resposta da OPL [17], Fig. 2.7. Na figura, à esquerda, é mostrado um diagrama de conexões entre as células do sistema visual precoce. C, H, B e G são indicadores que significam cones, células horizontais, células bipolares e células ganglionares respectivamente. Em destaque, na região interna ao quadrado pontilhado, é indicada a célula bipolar que faz a operação de retificação do sinal. Na parte da direita é mostrado o diagrama esquemático do re-

tificador que foi utilizado em Zaghloul [17] e mais recentemente em Katsiamis [30]. Neurônios conhecidos como células bipolares fazem a separação das amostras positivas e negativas no processamento realizado na retina biológica. Na implementação que estamos propondo chamaremos esse estágio, que faz a separação em relação ao valor das amostras, de retificador. Na biologia, dois canais exclusivos são utilizados para transmitir os valores positivos (canal *On*) e os negativos (canal *Off*). Essa é uma estratégia da natureza para não reduzir a faixa dinâmica do canal de transmissão ao meio e ainda economizar energia. Como a taxa de disparo é sempre positiva o sistema visual teria que se manter transmitindo um valor que representaria o zero caso o mesmo canal fosse utilizado para transmitir valores negativos e positivos [16]. Outra função conhecida da IPL é a de separar as amostras de mais alta frequência temporal, presentes no sinal de saída da OPL. Contudo, como serão utilizadas apenas imagens estáticas a implementação desse estágio seria algo desnecessário.

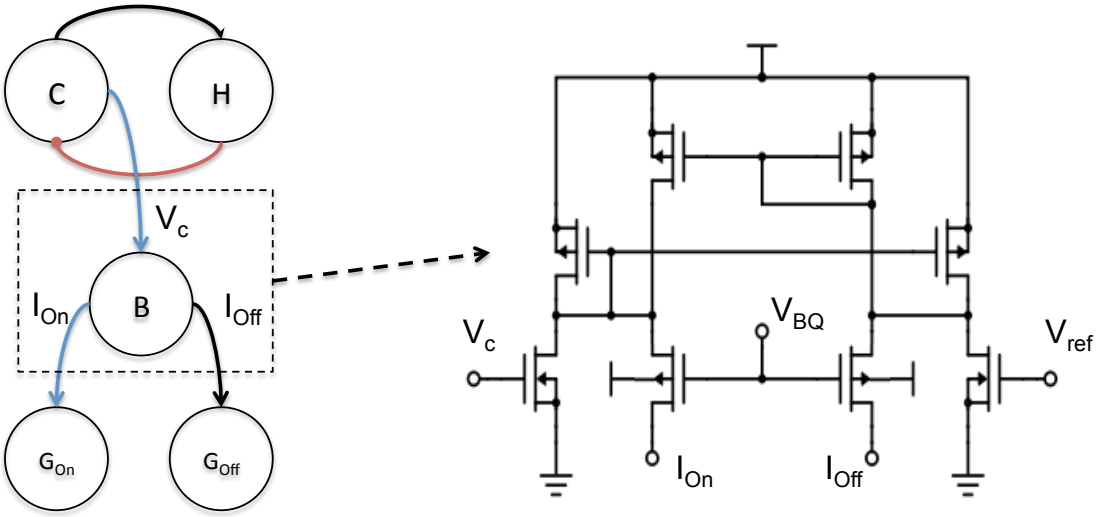


Figura 2.7: Diagrama em blocos do EVS, em destaque a IPL, e diagrama esquemático do circuito retificador da IPL.

Na Fig. 2.7 a marcação C indica cone, um tipo de fotorreceptor, a marcação H indica célula horizontal, a marcação B indica célula bipolar e a marcação G indica célula ganglionar. No retificador da figura, o terminal V_C é o terminal que recebe a imagem que sofreu o processamento da OPL. As saídas do retificador são I_{On} e I_{Off}. Os terminais V_{ref} e V_{BQ} são sinais de polarização do retificador. V_{ref} é a tensão de referência, o nível que representa o zero. Como o processamento realizado pelo retificador é baseado em operações de subtração entre espelhos de corrente, V_{BQ} é um ajuste que define o nível residual dos sinais I_{On} e I_{Off}. Esse estágio, o retificador, é bem conhecido na literatura, por esse motivo não será detalhado o projeto dos retificadores. Mais detalhes sobre o retificador podem ser encontrados em [17].

2.3 Modelo para a Camada de Células Ganglionares

As células ganglionares compõem o último estágio de processamento na retina biológica. Esses neurônios respondem por meio de pulsos elétricos às mudanças na amplitude do sinal em sua entrada, que é o sinal proveniente da saída do estágio da IPL. Experimentos fisiológicos mostram que a camada das células ganglionares altera a representação da informação visual, que antes estava representada por uma sequência de valores escalares na OPL e IPL, para uma representação pulsada. Muitos modelos para o comportamento dos neurônios biológicos têm sido propostos na literatura. Como exemplo temos o modelo *Integrate-and-Fire* (IF) [3], [22] . As entradas desse modelo são descritas por K sequências de pulsos $v_i(t)$, $i = 1, \ldots, K$. No neurônio, cada uma das K entradas é multiplicada por uma constante g_i, $i = 1, \ldots, K$. Após a multiplicação todos os sinais são somados de acordo com $s(t) = \sum_i (v_i(t, i) \cdot g(i))$, e o sinal $s(t)$ é integrado, $v(t) = \int s(t) \cdot dt$. Se $v(t)$, variável que foi chamada de tensão de membrana, ultrapassar um valor limiar, v_{th}, um pulso é gerado na saída do neurônio ($O(t)$) e $v(t)$ é ajustado para zero. Como resposta aos estímulos aplicados a eles, os neurônios geram pulsos de formato semelhante onde o intervalo temporal entre os pulsos é modulado pela intensidade do estímulo [31]. Devido ao fato de que os pulsos gerados por esses neurônios possuem formatos semelhantes, acredita-se que esses pulsos não carreguem informação alguma. É provável que eles representem apenas marcações de algum evento ocorrido e que a informação seja codificada pelo intervalo de tempo entre essas marcações. Essa estratégia biológica é conhecida na literatura como representação por eventos (*Event Representation* - ER). Nesse trabalho, mesmo sem essa certeza de como a biologia funciona, será assumida a condição de que são os intervalos de tempo entre os eventos que codificam a informação e assim possibilitar a continuidade desse trabalho. A implementação em *hardware* de um neurônio da camada das células ganglionares é mostrada no lado direito da Fig. 2.8. Um esquema da matriz de neurônios que será utilizado em nosso trabalho e um diagrama esquemático de um neurônio da camada das células ganglionares são apresentados na Fig. 2.8. O terminal 1 é a entrada do neurônio. O sinal de saída do retificador é conectado a esse terminal. O terminal 3 informa às outras partes do circuito que um determinado neurônio gerou um evento, $R_{eq}(x, y)$. O terminal 2 juntamente com o terminal 4 formam os terminais $A_{ck}(x)$ e $A_{ck}(y)$ que confirmam ao neurônio que o evento foi transmitido corretamente para as outras partes do sistema. A função lógica que controla a comunicação utiliza esses dois sinais como um sinal de *reset*. Esse sinal de *reset* serve para reiniciar o neurônio que gerou um evento.

Na camada das células ganglionares os neurônios seguem arranjos hexagonais [16]. Em chips de silício os neurônios, de uma maneira geral, são arranjados na

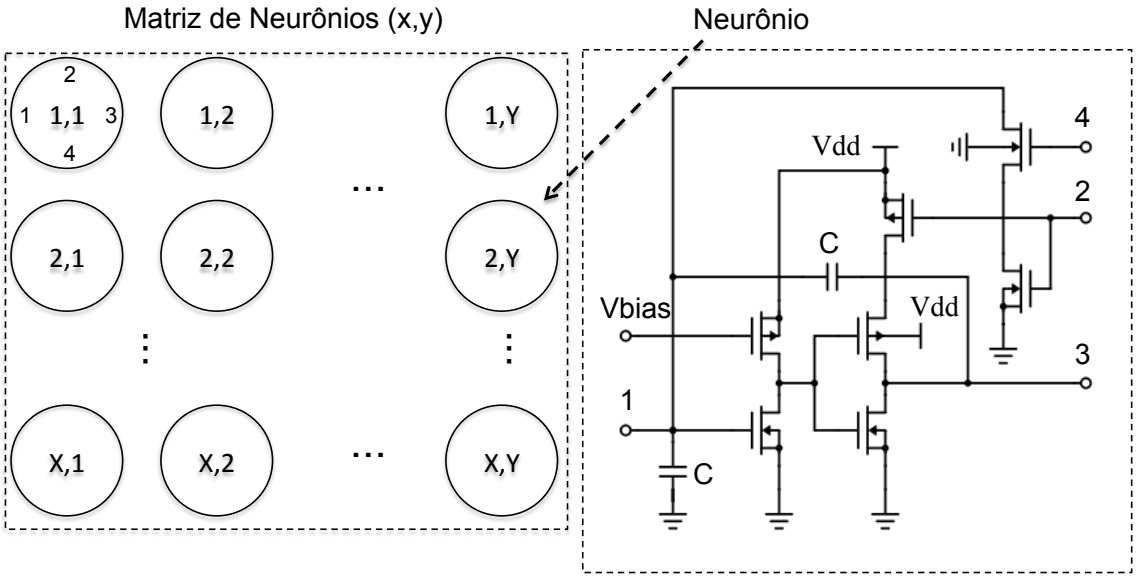


Figura 2.8: Na estrutura apresentada temos a matriz de neurônios com $X \cdot Y$ neurônios, cada um deles numa posição (x, y). Ao lado da matriz é apresentada a estrutura interna de um desses neurônios. A figura foi adaptada de Boahen [32].

forma de uma matriz onde cada elemento da matriz está localizado em uma posição (x, y). No caminho pós-retina a informação é transportada até o córtex visual por meio do nervo óptico. O nervo óptico é um emaranhado de fibras nervosas, condutoras de eletricidade, composto por canais paralelos que ligam as saídas das células ganglionares a outros neurônios. Dispositivos semicondutores não utilizam uma quantidade enorme de canais paralelos para transmitir informação. Devido ao limitado número de terminais disponíveis em um circuito integrado CMOS, a arquitetura paralela que conecta os neurônios na biologia poderia tornar o projeto de um circuito integrado CMOS inviável. Uma considerável simplificação do *hardware* poderia ser alcançada caso os neurônios pudessem transmitir informações compartilhando um mesmo canal de comunicação. A topologia SA faz uso dessa abordagem onde uma arquitetura serial de transmissão com um esquema de multiplexação no tempo do tipo TDM (*Time-Division-Multiplexing*) permite a comunicação. O AER (*Address Event Representation*) é um tipo de técnica que utiliza topologia SA. Ao contrário de sistemas síncronos de comunicação serial, como tradicionalmente se encontra nos esquemas de varredura e amostragem, da arquitetura de imageadores CMOS, no AER existe um esquema de transmissão assíncrono onde é o próprio neurônio que faz a requisição de acesso ao canal de comunicação. Esse esquema permite maior eficiência em relação às questões de economia de energia e aproveitamento da banda de transmissão do canal [26]. Mais detalhes sobre os sistemas de comunicação serão dados no Cap. 3.

2.4 Resultados para a OPL e a IPL

A Eq. (2.19) foi simulada numericamente (por meio do software Matlab) utilizando um conjunto de estímulos de dimensão 50x1 onde o valor de cada estímulo variava de acordo com uma função senoidal no espaço (x) e no tempo (t). A simulação nos permitiu obter a resposta em frequência no espaço e no tempo da OPL [16]. Essa OPL foi projetada a partir da metodologia proposta para projetos de redes resistivas MOS apresentada nesse capítulo. Na metodologia proposta foram utilizadas as Eqs. (2.16) e (2.18), que descrevem o comportamento da transcondutância e do pseudo-resistor modelados por transistores MOS, e das Eqs. (2.6) e (2.7), que descrevem os picos de resposta nas frequências espacial e temporal em uma OPL. Um sistema de equações foi obtido, a partir dessas equações, e a solução desse sistema permitiu encontrar as tensões de polarização dos transistores MOS que possibilitaram sintonizar a resposta da filtragem da OPL, espacialmente e temporalmente, nas frequências desejadas. Um diagrama esquemático de dimensões 50x1, onde cada posição é composta por uma unidade da OPL como aquela apresentada nas Figs. 2.6 e 2.9, foi desenvolvido. O fotodiodo que aparece na Fig. 2.6 foi substituído por uma fonte de corrente de valor ik, onde k é um número inteiro de 1 até 50. Essas 50x1 unidades da OPL foram conetadas, a partir dos nós Vc e Vh, por meio de transistores MOS utilizados como pseudo-resistores, Fig. 2.10.

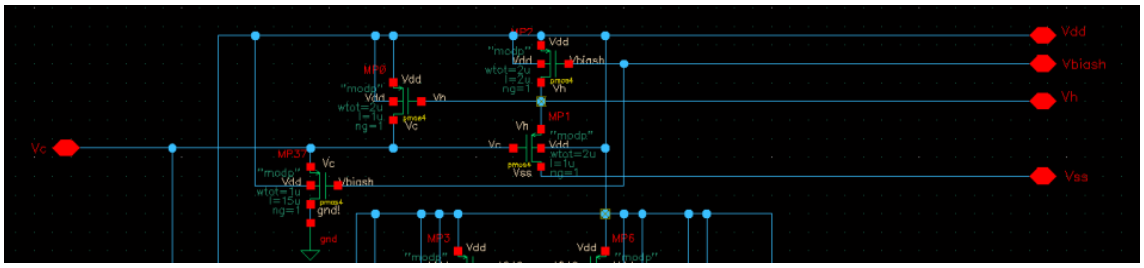


Figura 2.9: Diagrama esquemático de uma unidade da OPL. Esse diagrama esquemático ilustra a implementação do modelo elétrico, apresentado na Fig. 2.6, feita na interface de desenvolvimento Cadence Virtuoso.

O diagrama esquemático resultante da conexão das 50x1 unidades da OPL que foram conectadas a partir dos pseudo-resistores foi submetido a simulações elétricas. A simulação elétrica foi conduzida a partir do *software* de modelagem e simulação de circuitos integrados Cadence Virtuoso utilizando as ferramentas de projeto com tecnologia de fabricação de 0.35 μm da AMS. Devido às dificuldades envolvidas em gerar estímulos complexos, como por exemplo, estímulos senoidais variáveis no espaço e no tempo para as 50x1 unidades utilizando a interface gráfica, a simulação elétrica foi controlada por linhas de comando em um *script*. Para isso foi utilizada a interface de simulação *Ocean Script* e alguns arquivos de estímulos apropriados. Um exemplo do *script* utilizado para caracterizar a resposta em frequência espacial

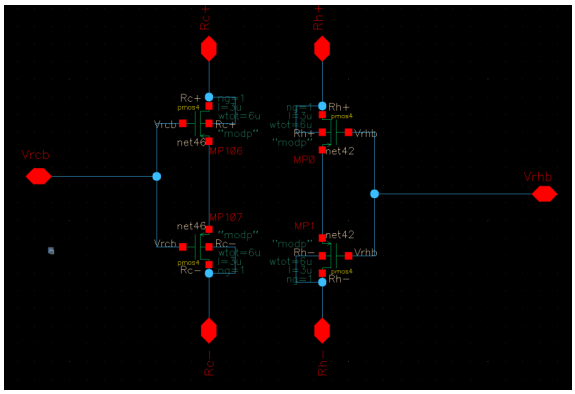


Figura 2.10: A implementação na interface de desenvolvimento do diagrama esquemático dos pseudo-resistores R_{hh} e R_{cc}.

e temporal pode ser visto na Fig. 2.11. *Scripts* completos do *Ocean Scripts* são mostrados no Apêndice A, bem como os *scripts* do Matlab que são usados para gerar os arquivos do *Ocean Scripts* e reconstruir os dados gerados por simulação elétrica.

```
; Leitura das bibliotecas

simulator( 'spectre )
design(
"/home/netware/users/humberto/netware/ProfileNT/Cadence_Simulacoes/S
netlist")
resultsDir(
"/home/netware/users/humberto/netware/ProfileNT/Cadence_Simulacoes/S
modelFile(
    '("/cds/Opus/user/AMS370/spectre/c35/mcparams.scs" "")
    '("/cds/Opus/user/AMS370/spectre/c35/cmos53.scs" "cmostm")
    '("/cds/Opus/user/AMS370/spectre/c35/res.scs" "restm")
    '("/cds/Opus/user/AMS370/spectre/c35/cap.scs" "captm")
    '("/cds/Opus/user/AMS370/spectre/c35/bip.scs" "biptm")
    '("/cds/Opus/user/AMS370/spectre/c35/ind.scs" "indtm")
)

;           Tipo de simulacao
analysis('dc ?saveOppoint t )          ;

;           Atribuicao de valores          (desVar)
    desVar("R"       10G)
    desVar("R2" 100000k)
for(k 1 20
    desVar("i1" sin(k*3.14159/25)*1n)
    desVar("i2" sin(k*2*3.14159/25)*1n)
    desVar("i3" sin(k*3*3.14159/25)*1n)
    desVar("i4" sin(k*4*3.14159/25)*1n)
    desVar("i5" sin(k*5*3.14159/25)*1n)
    desVar("i6" sin(k*6*3.14159/25)*1n)
    desVar("i7" sin(k*7*3.14159/25)*1n)
```

Figura 2.11: *Script* que foi utilizado para realizar a simulação a partir da interface *Ocean Script*. O comando utilizado para alterar o valor das fontes de corrente ik é o *desVar*.

Uma simulação para a verificação de funcionamento, que consistiu em obter a resposta ao impulso espacial do diagrama esquemático da OPL de dimensões 50x1, foi feita e o resultado do teste é apresentado na Fig. 2.12. Para gerar o impulso espacial somente a fonte de corrente na posição $k = 26$ foi acionada com uma corrente de 100 pA. A Fig. 2.13 mostra a resposta da OPL para um impulso no espaço e no tempo.

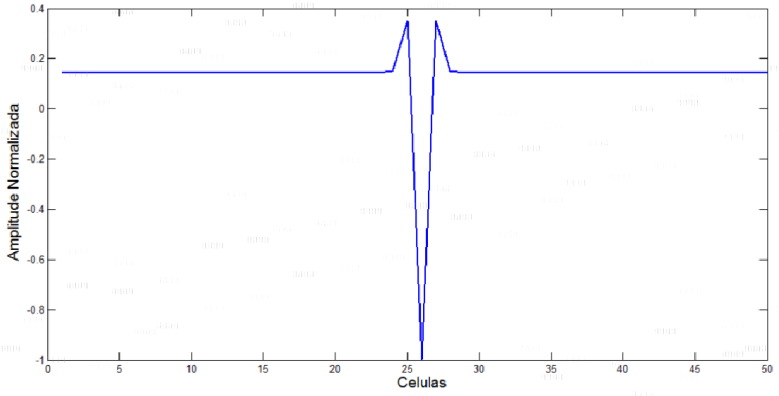


Figura 2.12: Simulação elétrica da resposta da OPL 50x1 ao impulso no espaço. O eixo horizontal representa cada unidade da OPL. O esperado seria uma resposta semelhante à resposta ao impulso espacial de um filtro DoG, resposta conhecida como chapéu mexicano. Assim o formato obtido é semelhante ao esperado. O formato está invertido devido às características de polarização elétrica dos transistores MOS que foram utilizados na implementação.

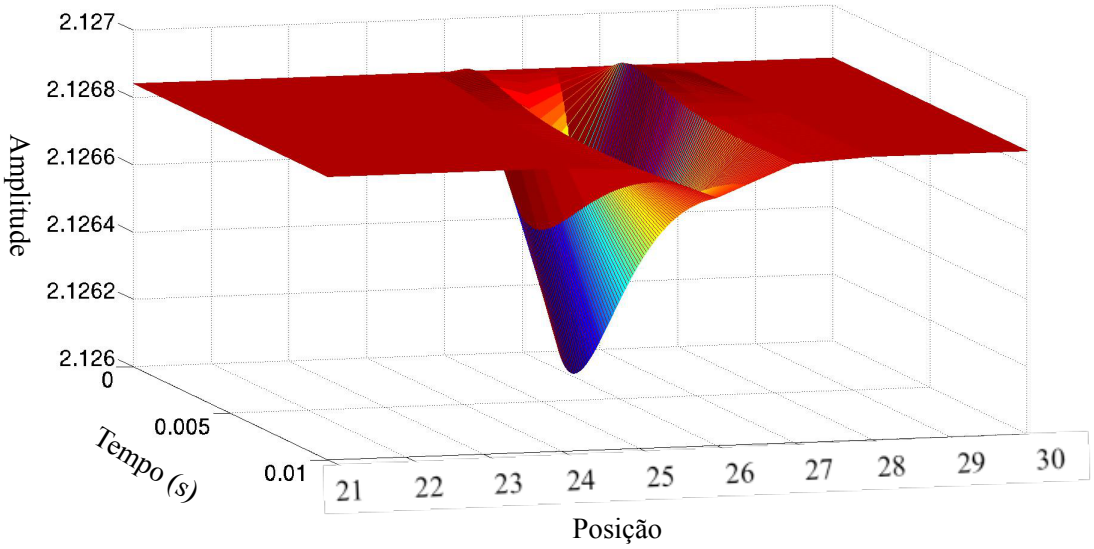


Figura 2.13: A resposta do modelo elétrico da OPL (50x1) ao impulso no espaço e no tempo. O impulso é aplicado à posição 26 do arranjo.

Os resultados da simulação elétrica (parte de baixo) e da previsão teórica (parte de cima) que foram utilizados para caracterizar a resposta em frequência espacial e temporal da OPL 50x1 são apresentados na Fig. 2.14. Os resultados apresentados pelo diagrama esquemático da OPL estão de acordo com o resultado esperado (previsão teórica). Esse resultado mostra que a metodologia proposta pode ser utilizada em projetos de OPL que utilizam transistores MOS. Os valores internos a cada quadrado na figura representam a intensidade da resposta em frequência da OPL para um par composto de uma frequência espacial e uma frequência temporal. Uma implementação de um diagrama esquemático do modelo elétrico apresentado

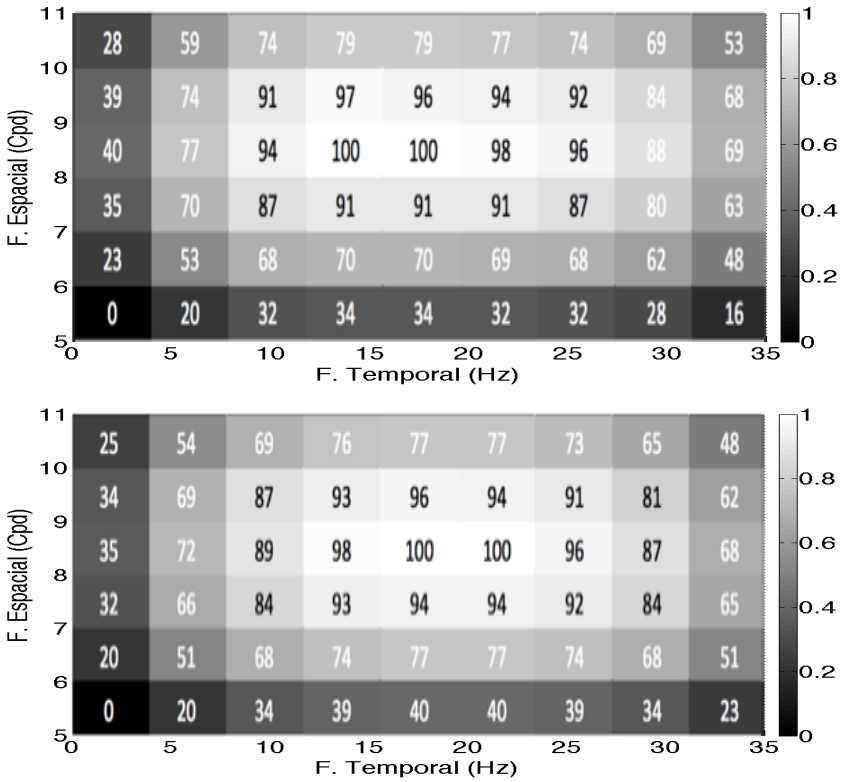


Figura 2.14: A resposta espaço-temporal do modelo de uma OPL para estímulos senoidais de valores variando no espaço e no tempo. A OPL foi projetada para uma resposta passa banda espaço-temporal de 10 Cpd e 10 Hz. Na figura é mostrada a previsão teórica (em cima) e o resultado da simulação elétrica (embaixo) que foi obtido ao simular o diagrama esquemático da OPL que foi projetado a partir da metodologia proposta.

na Fig. 2.7 foi desenvolvida e uma ilustração é mostrada na Fig. 2.15. Uma simulação elétrica foi realizada com o intuito de verificar a funcionalidade do retificador. Nessa simulação, a referência, Vref (Vr), foi ajustada para 2.25 V. A entrada do retificador foi submetida a uma tensão variando 2 até 2.5V, limites que concordam com a faixa dinâmica produzida pela saída do diagrama esquemático da OPL. As respostas em corrente produzida nos dois canais de saída do retificador, canal *On* e *Off*, são mostradas na Fig 2.16.

2.5 Conclusões

Nesse capítulo foi utilizada a metodologia de projeto de redes resistivas proposta para projetar uma OPL de frequência de pico espacial de 10 Cpd e de pico temporal de 10 Hz. Os resultados apresentados na Fig. 2.14 sugerem que a metodologia pode ser utilizada para projetos de OPL onde se queira sintonizar a resposta em frequência espacial e temporal. O diagrama esquemático de um retificador foi desenvolvido. Os

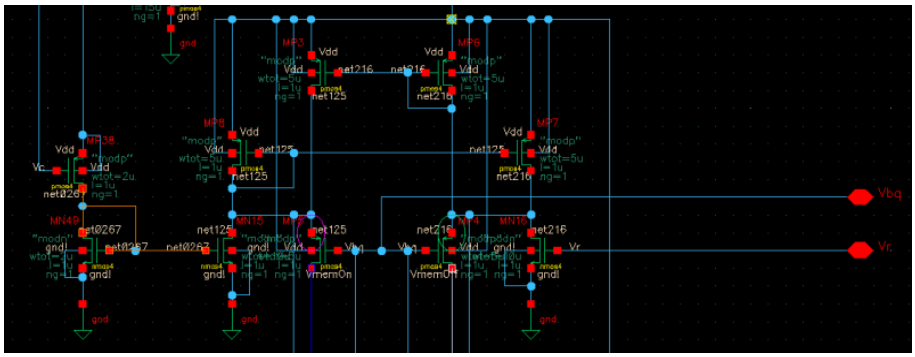


Figura 2.15: Diagrama esquemático do modelo do retificador. O diagrama foi utilizado para a realização de simulações elétricas.

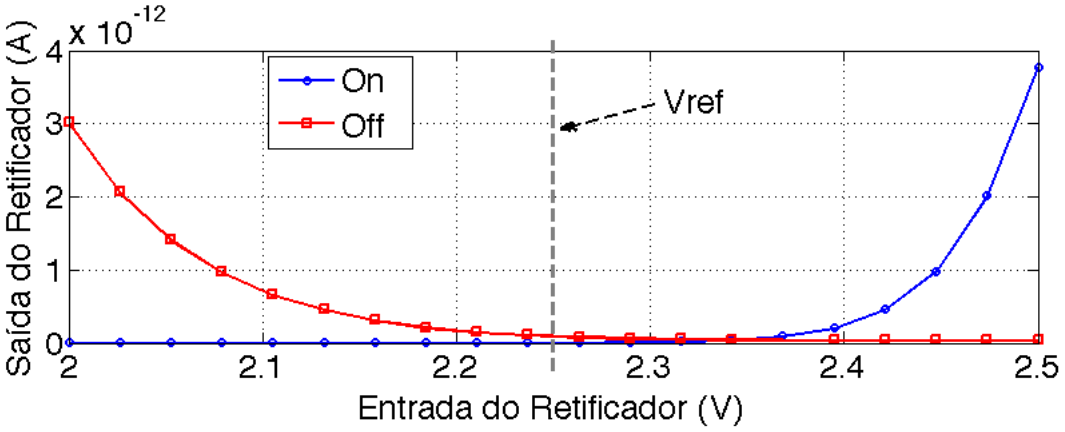


Figura 2.16: Resposta dos dois canais (*On* e *Off*) na saída do retificador produzidas por um estímulo que varia de acordo com o sinal de saída do diagrama esquemático da OPL, sinal Vc. É possível verificar que, apesar de certo nível de *off-set* de corrente, as respostas dos canais são separadas em relação a tensão de referência Vref.

resultados apresentados na Fig. 2.16 mostram que o retificador é capaz de separar entradas positivas e negativas considerando um determinado referencial de tensão. A partir da metodologia de simulação com a ferramenta *Ocean Script* utilizada foi possível realizar as simulações da OPL manipulando uma quantidade de 50 fontes de corrente. A obtenção dos resultados de simulação mostra que a metodologia de simulação é prática em casos onde existe uma quantidade muito grande de estímulos que se deseja manipular.

Capítulo 3

O Nervo Óptico e a Técnica AER

Chips neuromórficos [16] são exemplos de sistemas CMOS utilizados para implementar circuitos neurais pulsados [4], [5]. Nesses sistemas, devido à enorme quantidade de neurônios a serem implementados em *hardware*, é comum que estes neurônios sejam divididos entre dois ou mais dispositivos SoCs (*System on Chip*).

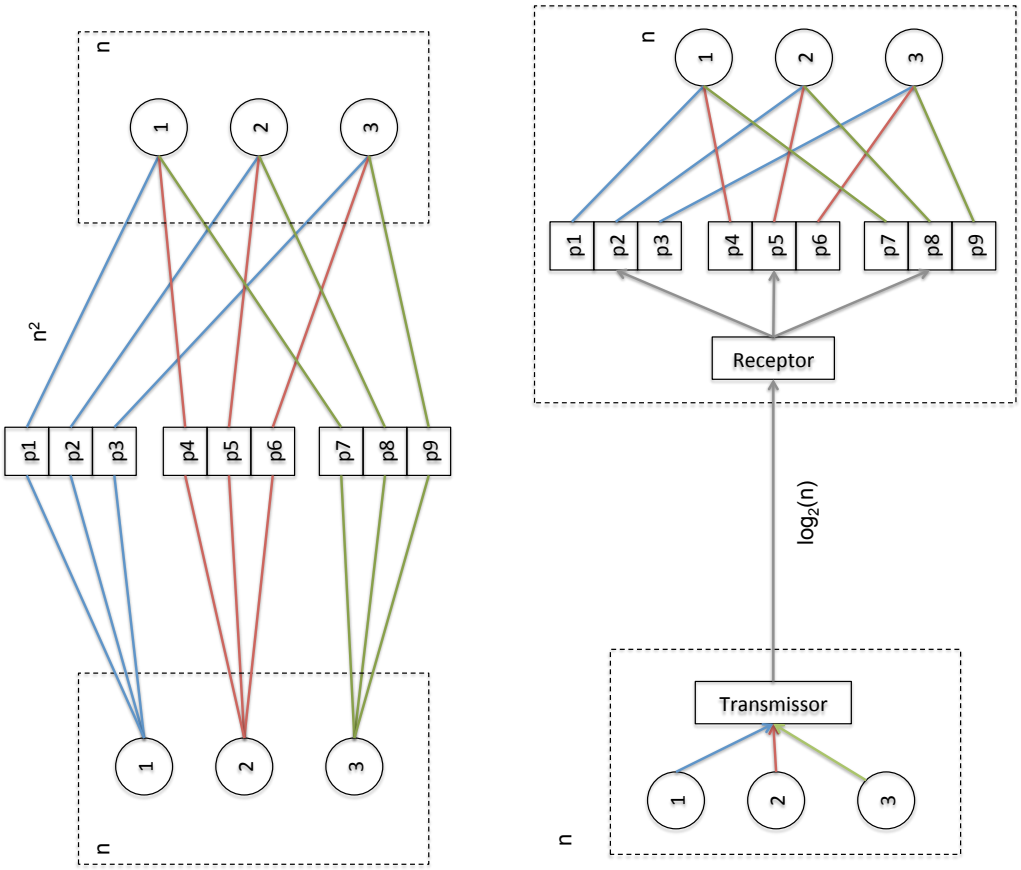


Figura 3.1: Diferenças entre as topologias FD e SA. Considerando que os retângulos pontilhados são circuitos CMOS, onde cada um dos circuitos implementa três neurônios, à esquerda temos a topologia FD que necessita de n^2 conexões, enquanto na topologia SA essa quantidade pode ser reduzida para um canal de comunicação de largura de $\log_2(n)$ bits.

Existem diversas topologias de implementação de circuitos neurais em *hardware* e, como exemplo, será explicada a topologia FD. Essa topologia implementa fisicamente cada elemento do circuito neural (dendritos, axônios, sinapses e neurônios) de maneira dedicada [24]. Na arquitetura biológica, em uma retina composta por n neurônios na camada das células ganglionares seriam necessárias n^2 conexões com os neurônios do córtex visual para que cada neurônio da retina pudesse transmitir informação para cada neurônio do córtex. Caso fosse assumida essa mesma abordagem para implementar a comunicação entre uma retina CMOS e um chip que implementasse um córtex teríamos que utilizar a mesma quantidade de conexões, conforme é mostrado na parte esquerda da Fig. 3.1. Esse exemplo torna evidente o problema de se implementar conexões entre neurônios através de canais paralelos já que os terminais disponíveis em dispositivos SoC são muito limitados. A fim de reduzir a complexidade de *hardware* necessária em implementação de circuitos CMOS, outras topologias surgiram, como por exemplo, a topologia SA [33], [4]. Na topologia SA os neurônios compartilham o mesmo axônio (um canal de comunicação) para transmitir informação. As n^2 conexões dedicadas podem ser substituídas por um canal de comunicação de até $\log_2(n)$ bits sem qualquer prejuízo, desde que a capacidade de transmissão do canal de comunicação não seja ultrapassada. Um esquema é mostrado na parte direita da Fig. 3.1. Um canal de largura de $\log_2(n)$ bits permite ao sistema de transmissão transmitir n códigos que estão relacionados a identificadores dos n neurônios. Embora a topologia SA permita uma menor complexidade de *hardware* do que a alcançada ao se utilizar uma topologia FD, o sinal a ser transmitido pelo canal de comunicação é submetido à latência relacionada aos atrasos envolvidos com a multiplexação de informação no tempo. Na topologia SA quanto maior a redução na complexidade de *hardware* permitida, maior é a quantidade de neurônios que utilizam o mesmo canal de comunicação e maiores são as distorções envolvidas com a multiplexação. O AER (*Address Event Representation*) é uma técnica que utiliza topologia SA e é capaz de reduzir complexidade de *hardware* ao permitir a comunicação por um por um canal de comunicação de até $\log_2(n)$ bits.

3.1 Elementos de um Sistema AER

Os estágios de transmissão e recepção da técnica AER são ilustrados na Fig. 3.2. Ambos os estágios são compostos por uma matriz de neurônios, alguns elementos de processamento, codificadores, decodificadores e sistemas de controle. Um canal de transferência de dados possibilita a comunicação entre os dois estágios. No estágio transmissor os neurônios da matriz fazem requisições de acesso ao canal de comunicação por meio dos elementos de processamento. Esses elementos de processamento

consistem de circuitos implementando funções lógicas que permitem que o sinal de requisição de transmissão, gerado pelos neurônios, seja entregue aos codificadores. O codificador produz códigos únicos (endereços) para as requisições feitas pelos neurônios da matriz. Esses endereços possibilitam a identificação da posição da matriz onde um evento foi gerado. Esse endereço é transmitido até o estágio receptor. No receptor o endereço é decodificado pelo elemento decodificador e a informação da posição da matriz de onde o evento foi gerado é obtido. Obtida essa informação de posição, o estágio receptor informa ao estágio transmissor, por meio dos elementos de controle, que a informação foi entregue corretamente. Algumas técnicas AER utilizam um elemento árbitro. Esse elemento gerencia as requisições feitas pelos neurônios da matriz e permite que apenas uma requisição seja atendida por vez. Sistemas AER que não possuem árbitros estão sujeitos a perda de informação quando dois ou mais neurônios tentam transmitir em um mesmo ciclo de comunicação. A Fig. 3.2 apresenta a arquitetura AER. Elementos de processamento em linha e coluna, um codificador de linha e coluna e um elemento de controle permitem que um evento seja enviado do chip A até o Chip B. No AER uma sequência de eventos $e = (e_x, e_y, e_t)$ que foram gerados na matriz do estágio de transmissão é convertida na forma de uma sequência de códigos e transmitida pelo canal de comunicação [12]. Os códigos transmitidos representam determinadas características dos eventos como a posição (x,y) do neurônio na matriz e o instante de tempo (t) no qual o evento foi gerado. A Fig. 3.3 mostra um diagrama de comunicação AER onde N eventos e_n são transmitidos por um canal de comunicação. Esses eventos são reconstruídos no estágio receptor, onde são denotados por $\hat{e}_n$.

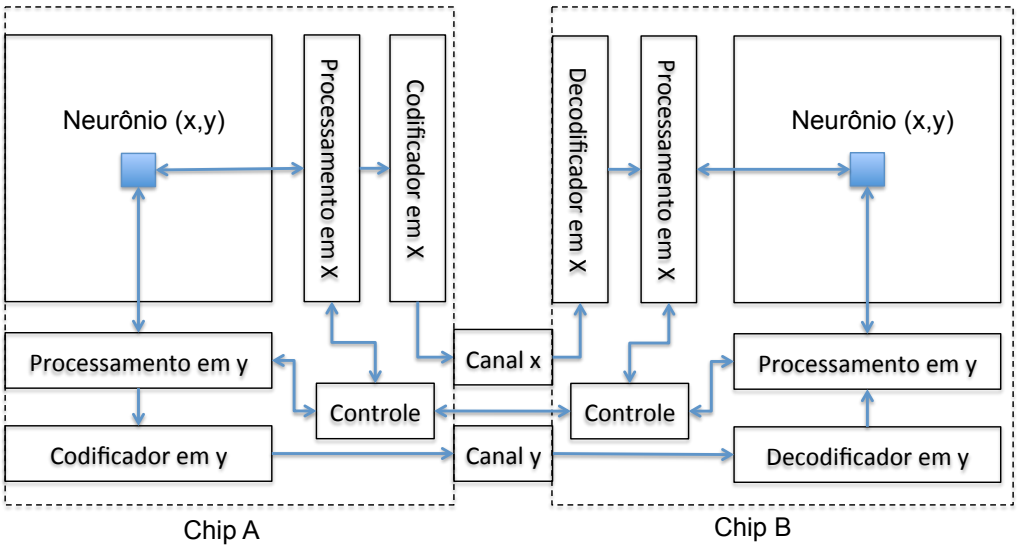


Figura 3.2: A arquitetura de comunicação AER é apresentada. Elementos de processamento, elementos codificadores e um elemento de controle possibilitam que um evento gerado na matriz do estágio transmissor seja enviado até a matriz do estágio receptor.

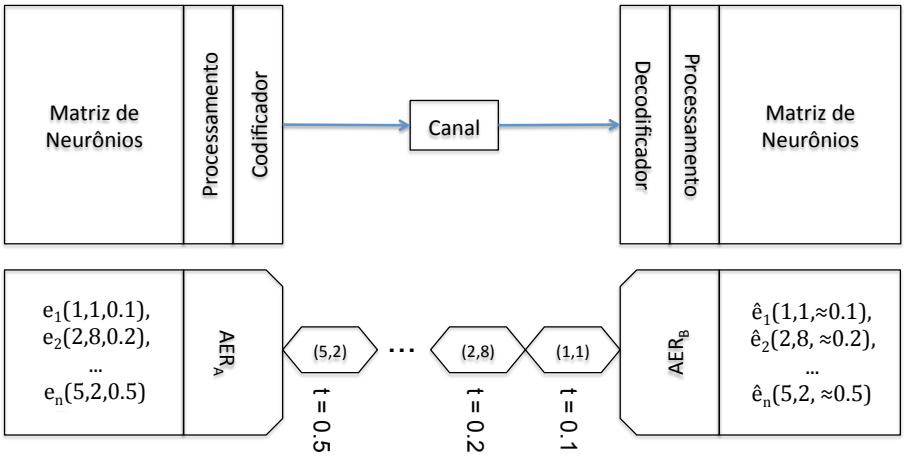


Figura 3.3: Uma ilustração da arquitetura de comunicação AER onde N eventos e_n, $n = 1, 2, \ldots, N$, são transmitidos pelo canal de comunicação. Em uma arquitetura AER, não é possível garantir que o evento e_n seja reconstruído perfeitamente em relação ao instante de tempo t. A figura foi adaptada de [12].

Dentro do estágio de transmissão, os pedidos de requisição dos neurônios $R_{eq}(x, y)$ de acesso ao canal de comunicação costumam ser tratados a nível de linhas $R_{eq}(x)$ e de colunas $R_{eq}(y)$. Essa abordagem permite reduzir o número de entradas do codificador de $(X \cdot Y)$ para $(X + Y)$, onde X e Y são as quantidades totais de linhas e colunas da matriz de neurônios. A Fig. 3.4 mostra o esquema e o diagrama esquemático dos elementos de processamento de linha e coluna (LC) e de dados (D) que permitem essa simplificação do codificador. O elemento LC é uma chave analógica que quando acionada conecta um nível zero volts à saída. O elemento D é um inversor que é mantido desativado por um transistor configurado em *pull-up* conectado à sua entrada. Caso no terminal 1 do elemento LC seja aplicado um nível de tensão suficientemente baixo, os terminais 2 e 3 desse mesmo elemento entrarão em um estado de alta impedância.
Esse estado de alta impedância na saída do elemento LC faz com que a entrada do elemento D, o terminal 1, se mantenha em nível alto devido à influência do transistor configurado em *pull-up*. O que provoca que a saída do elemento D, o terminal 2, se mantenha em nível baixo. Caso um nível alto seja colocado no terminal 1 do elemento LC, os terminais 2 e 3 entrarão em um estado de baixa impedância e farão com que a entrada do elemento D, o terminal 1, fique em nível baixo. Como o elemento D é um inversor, um nível baixo em sua entrada provoca um nível alto em sua saída, o terminal 2. Esses elementos de processamento do AER são bem conhecidos e aparecem com frequência em trabalhos relacionados a comunicação utilizando a técnica AER [24], [34] e [35]. A Fig. 3.5 exemplifica dois dos elementos LC e D apresentados na literatura.

Em [33], na estrutura LC apresentada na Fig. 3.5 (parte a no canto superior), os terminais 1, 2, 3 e 4 possuem as mesmas funcionalidades dos sinais $R_{eq}(x, y)$,

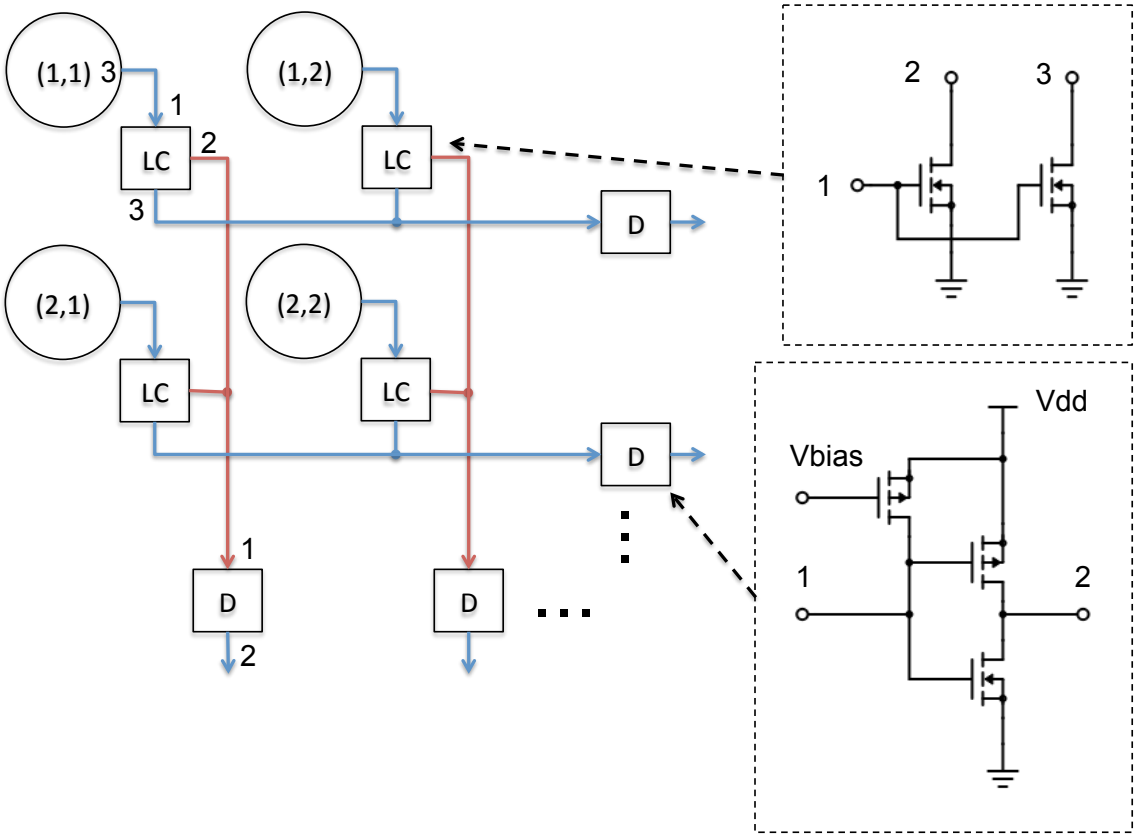


Figura 3.4: Arquitetura dos elementos de processamento LC e D. Essa arquitetura possibilita a redução da quantidade de conexões entre a matriz de neurônios e o codificador. Um diagrama esquemático de cada elemento é mostrado.

$R_{eq}(x)$, $R_{eq}(y)$ e $A_{ck}(x)$. O sinal $A_{ck}(x)$ funciona como um sinal de confirmação que informa que a transmissão foi bem-sucedida. As chamadas células de memória [33] (portas lógicas *NOT* realimentadas) armazenam os sinais de requisição da saída da matriz de neurônios ou da saída do árbitro durante um ciclo de comunicação. Essas células de memória são partes dos elementos D em sistemas AER arbitrados, Fig. 3.5 parte a no canto inferior. Como nos sistemas AER arbitrados uma requisição feita por um neurônio não será atendida imediatamente, os elementos D armazenam esse estado de requisição até que o sistema AER atenda. Exceto por essas células de memória, o funcionamento do elemento D é análogo ao que foi apresentado para o caso geral. O elemento D se conecta ao árbitro por meio do terminal 2, se conecta a matriz de neurônios por meio dos terminais 1 e 4 e se conecta ao codificador por meio do terminal 5. O terminal 6 permite que o elemento D seja ativado. Em [34], não é possível classificar exatamente quem é o elemento LC e o elemento D devido à proposta de unir todos esses elementos em um único bloco, Fig. 3.5 parte b canto superior. É possível associar a função dos pinos 1, 2, 3 e 4 aos pinos 1, 2, 4 e 3 do elemento LC (em [33]). O pino 5 corresponde a um dos sinais A_{ck}. O terminal 6 é um terminal relacionado a um controle de *reset* global. Dentro do mesmo bloco

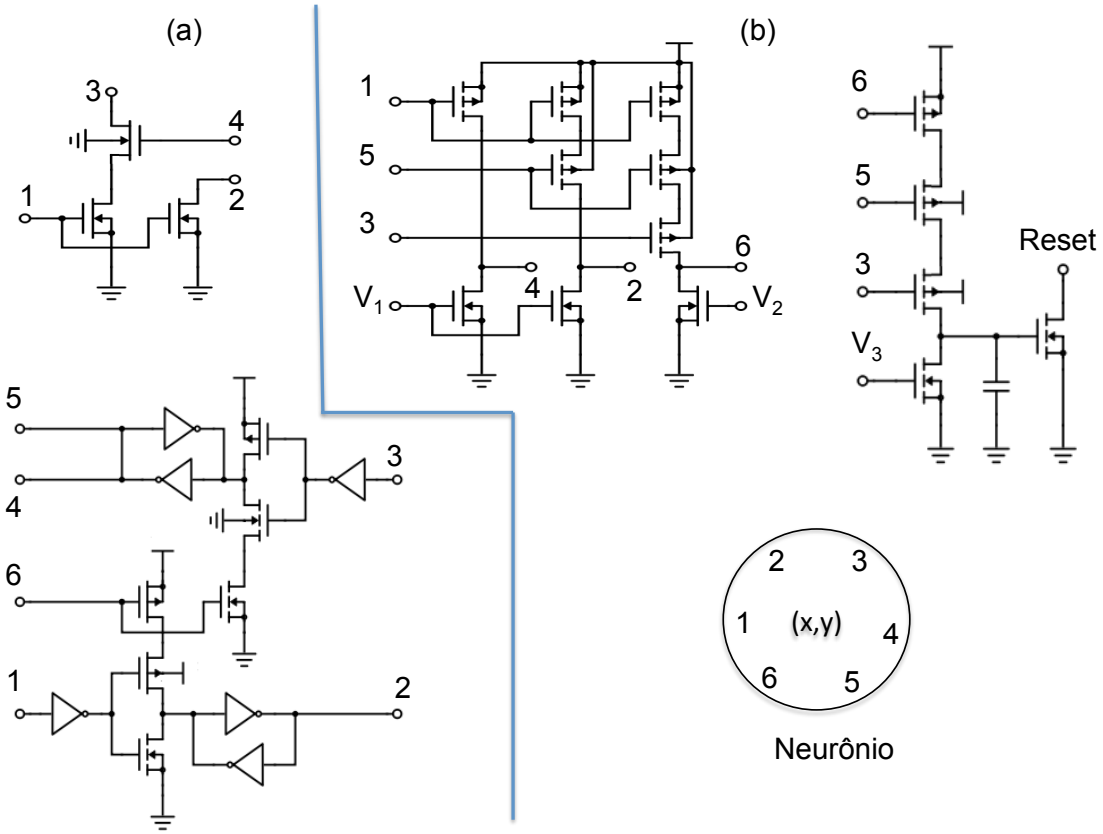


Figura 3.5: A parte (a) da figura representa o diagrama esquemático das estruturas LC, no topo, e D, na parte inferior, utilizadas em Boahen [33]. Na parte (b) da figura [34] é apresentado um diagrama esquemático onde as estruturas LC, D e o neurônio formam um único bloco.

do neurônio funciona ainda um circuito lógico que controla o *reset*, Fig. 3.5 parte b canto superior direito. Na Fig. 3.5, parte b canto inferior, todos os circuitos apresentados na parte superior e um circuito de integração e disparo, semelhante ao apresentado na Fig. 2.8, compõem o neurônio. Assim como existem sistemas AER que utilizam árbitros [24], existem sistemas AER permitem que a colisão ocorra. A esses sistemas AER foram chamados de não arbitrados. Um exemplo é apresentado em [35] onde é introduzida a ideia de dispositivos que recuperam informações que colidiram. Existem também casos onde os sistemas AER não implementam recursos para evitar ou corrigir colisões e assumem o risco da perda de informação causadas pelas colisões. A seguir serão dadas explicações sobre os árbitros. O AER arbitrado foi inicialmente proposto por Mahowald [24] e sofreu melhorias em diversos trabalhos, como por exemplo em Boahen [32]. O esquema proposto por Mahowald é conhecido atualmente por AER de código arbitrado por árvore binária (AB) [35]. Existem diversos tipos de árbitros e cada um deles possui uma característica particular. É possível citar os árbitros de funcionamento digital como exemplo, os que utilizam arbitragem feita por árvore binária [24] e os que possuem funcionamento

analógico, como é o caso dos árbitros formados por *winner-takes-all* (WTA) [34]. Será discutido a seguir o funcionamento dos árbitros do tipo AB e WTA. Na árvore binária os eventos simultâneos nas posições x das linhas competem dois a dois por meio de estruturas digitais bi-estáveis (*flip-flops*) formadas por portas lógicas do tipo *NAND* (*Negative-AND*) e que chamamos aqui de células de decisão (CD). O mesmo ocorre no processamento dos eventos nas colunas. Essas estruturas possuem entradas $R_{eq}(A)$, $R_{eq}(B)$ e $A_{ck}(In)$, saídas $R_{eq}(Out)$, $A_{ck}(A)$ e $A_{ck}(B)$ e armazenam os estados M_A e M_B. Os eventos das linhas $(x, x+1)$ /ou colunas $(y, y+1)$ são aplicados as entradas $(R_{eq}(A), R_{eq}(B))$ e podem provocar os estados [$(0,0)$, $(0,1)$,$(1,0)$ e $(1,1)$] nessas entradas. Isso causaria os estados [$(0,0)$, $(0,1)$, $(1,0)$ e (A,A)] em (M_A, M_B). Os estados $(0,0)$, $(0,1)$, $(1,0)$ e $(1,1)$ significam respectivamente que não houve evento, houve evento em $R_{eq}(B)$, houve evento em $R_{eq}(A)$ e houve uma colisão entre $R_{eq}(A)$ e $R_{eq}(B)$. O estado (A,A) indica que serão assumidos os últimos estados válidos, $(0,1)$ ou $(1,0)$, para (M_A, M_B). Os estados de (M_A, M_B) são mantidos até que o árbitro tenha um vencedor. Existe uma relação lógica entre (M_A, M_B) e a saída $R_{eq}(Out)$. Essa lógica implica em, se o estado das memórias for diferente de $(0,0)$, então o estado do sinal $R_{eq}(Out)$ se torne verdadeiro. Esse sinal $R_{eq}(Out)$ é conectado outra CDs por meio de uma de suas entradas $R_{eq}(A)$ ou $R_{eq}(B)$ e uma nova disputa é iniciada. Após as requisições dos neurônios chegarem ao árbitro ocorrem $\lceil \frac{X}{2} \rceil$ competições em $\lceil \frac{X}{2} \rceil$ CDs.

Essas competições produzirão uma mesma quantidade de vencedores, $\lceil \frac{X}{2} \rceil$. A operação $\lceil . \rceil$ indica arredondamento para cima. Esses $\lceil \frac{X}{2} \rceil$ vencedores são armazenados nas memórias associadas às respectivas CDs onde as competições ocorreram. As $\lceil \frac{X}{2} \rceil$ CDs que promoveram as primeiras disputas ativarão seus sinais $R_{eq}(Out)$. Os sinais $R_{eq}(Out)$ estão conectados dois a dois nas entradas $R_{eq}(A)$ e $R_{eq}(B)$ de $\lceil \frac{\lceil \frac{X}{2} \rceil}{2} \rceil$ CDs que promoverão a segunda disputa. Sequencialmente, $\lceil \frac{\lceil \frac{X}{2} \rceil}{2} \rceil$ competições ocorrerão entre os $\lceil \frac{X}{2} \rceil$ vencedores iniciais. Os vencedores dessa disputa serão armazenados nas memórias das respectivas CDs que promoveram a disputa e os sinais $R_{eq}(Out)$ dessas CDs serão ativados. Esse mesmo procedimento se repete até que uma última disputa ocorra e o vencedor dessa última disputa será selecionado como o vencedor final da árvore binária. Esse vencedor final ganha acesso ao canal de comunicação. O sinal de saída $R_{eq}(Out)$ da última CD avisa ao sistema controlador da comunicação que houve um vencedor final. O controlador responde ativando a entrada Ack_{In} dessa última CD. A CD como resposta a ativação de sua entrada Ack_{In} ativa uma de suas saídas, Ack_A ou Ack_B. A saída que será ativada é função do estado armazenado em sua memória (M_A, M_B). Caso esteja armazenado o estado $(1,0)$ a saída Ack_A será ativada e caso esteja armazenado $(0,1)$ a saída Ack_B será ativada. As saídas Ack_A e Ack_B estão ligadas às duas CDs que mediaram a competição anterior a última disputa. Cada sinal (Ack_A e Ack_B) está ligado à en-

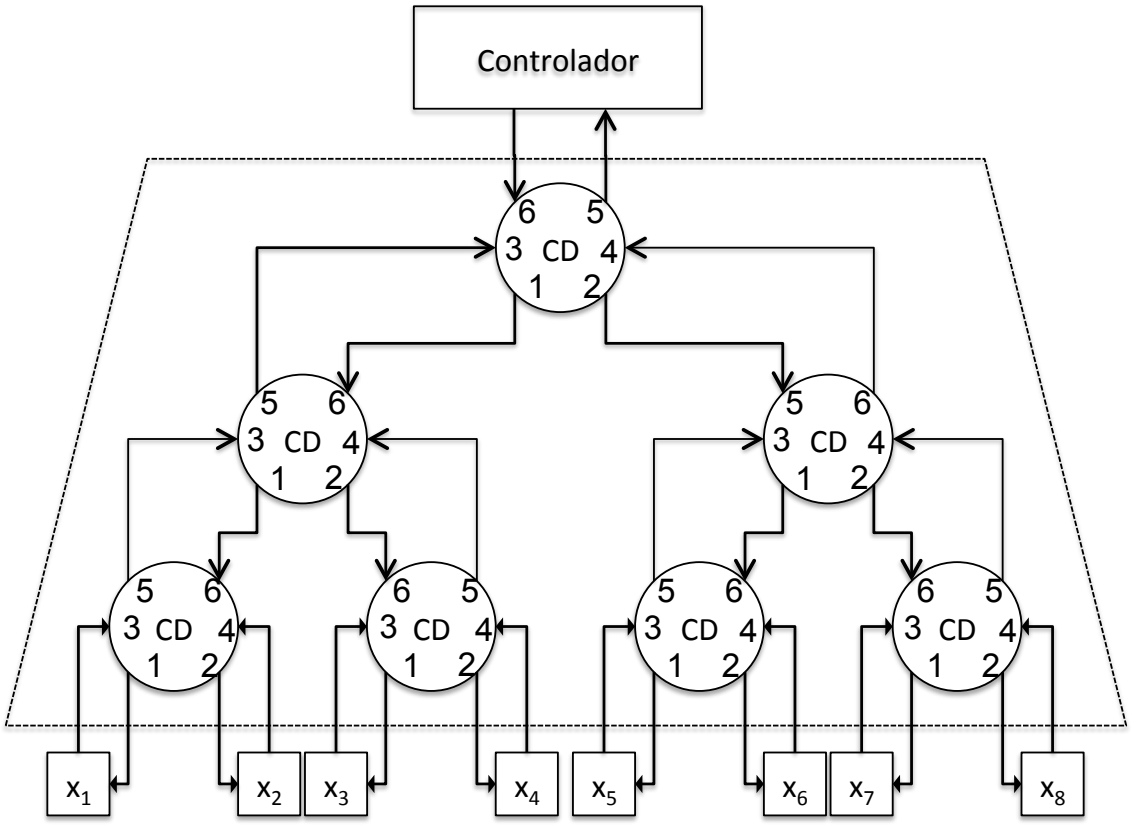


Figura 3.6: A arquitetura de um árbitro AB que gerencia oito requisições (de x_1 até x_8). Os sinais de entrada $R_{eq}(A)$, $R_{eq}(B)$ e $A_{ck}(In)$ são respectivamente representados pelos terminais 3, 4 e 6. Os sinais de saída Ack_A, Ack_B e $R_{eq}(Out)$ são representados respectivamente pelos terminais 1, 2 e 5.

trada Ack_{In} de cada uma das duas CDs. Esse processo se repete até alcançar as CDs onde as primeiras disputas aconteceram. Uma das $\lceil \frac{X}{2} \rceil$ CDs tem sua entrada Ack_{In} ativada e dependendo do valor armazenado em sua memória uma das suas duas saídas (Ack_A ou Ack_B) será ativada. Os sinais Ack_A e Ack_B são conectados diretamente ao codificador e à matriz de neurônios por meio de um elemento D como aquele apresentado na Fig. 3.5 (a).

Dessa maneira o codificador recebe apenas uma requisição por vez. A Fig. 3.6 mostra um esquema de árbitro AB para oito entradas nomeadas de x_1 até x_8 e três estágios de competição. Detalhes da CD de um árbitro AB podem ser vistos na Fig. 3.7. Os árbitros WTA utilizam circuitos analógicos que implementam operação do tipo *current conveyor*, operações que são a base para operações *winner-takes-all* [26]. Esses árbitros possuem X entradas $R_{eq}(x)$, X saídas $A_{ck}(x)$ e outros sinais que servem de polarização e sinalização. Em um árbitro WTA de X entradas ocorrem apenas uma disputa do tipo todos contra todos. Ao final dessa disputa a saída $A_{ck}(x)$ da célula vencedora é ativada. Como o árbitro WTA garante que apenas um sinal $A_{ck}(x)$ esteja ativo em sua saída, é possível utilizar um codificador de $\log_2(X)$

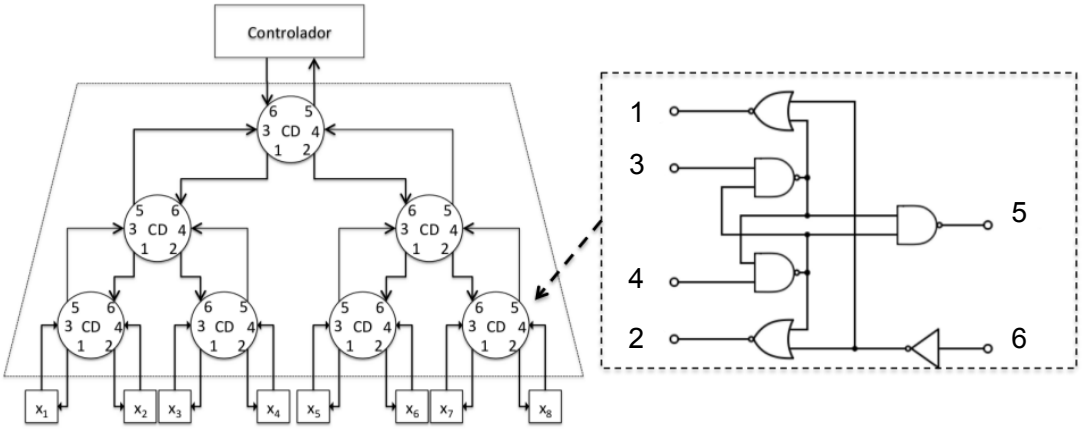


Figura 3.7: A CD é composta por estruturas digitais biestáveis como a que é apresentada na figura à direita. À esquerda é apresentado um exemplo de árvore binária. Os terminais dos elementos que compõem essa árvore binária estão de acordo com o que foi apresentado na Fig. 3.6.

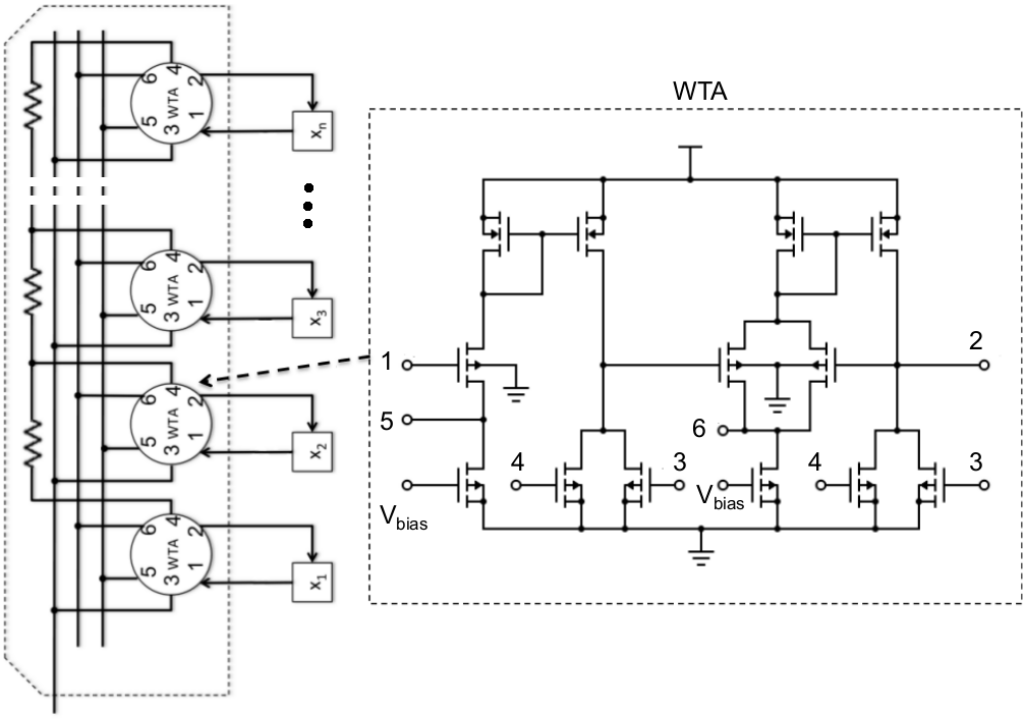


Figura 3.8: Arquitetura utilizada no árbitro do tipo WTA proposto [34] (à esquerda) e o diagrama esquemático de uma CD desse árbitro (À direita). Os sinais *ReqA* ou *ReqB* da CD da árvore binária são equivalentes ao sinal representado pelo terminal (1) e os sinais *AckA* ou *AckB* são equivalentes ao terminal (2). Os outros sinais são de controle e polarização da estrutura.

saídas. A Fig. 3.8 mostra detalhes internos do árbitro WTA e da sua célula básica. Em sistemas AER não arbitrados, nada impede que dois neurônios tentem acesso ao

canal durante o mesmo ciclo de comunicação. Como os codificadores utilizados em sistemas AER arbitrados não são capazes de gerar endereços para requisições que envolvem colisão a informação transmitida para o receptor não é uma informação válida. O que impossibilita a utilização desses codificadores em sistemas AER não arbitrados convencionais. Existe uma técnica AER que possibilita a recuperação dessa informação que foi dita como sendo inválida após ocorrer uma colisões. Essa técnica consiste em aumentar o tamanho do canal (aumento na quantidade de bits) e utilizar esse excedente para transmitir informação redundante. A partir dessa informação redundante mais o erro gerado pelo codificador a informação que era considerada perdida pode ser recuperada. Nessa técnica, para que, no máximo, k eventos que se envolveram em uma colisão sejam recuperados em um sistema composto por $(X \cdot Y)$ neurônios, seria necessário um canal de comunicação com pelo menos $(m \cdot k \cdot X)$ bits, onde $m = \lceil \log_2(X \cdot Y) \rceil$ [35].

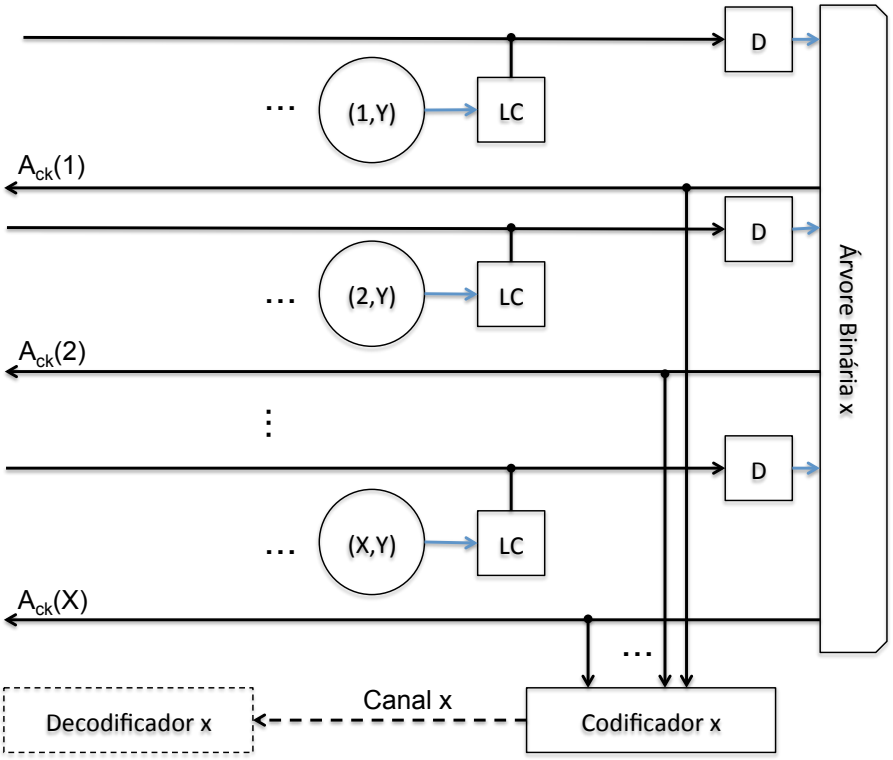


Figura 3.9: O codificador recebe do árbitro os X sinais $A_{ck}(x)$ e os codifica em um endereço composto de $\lceil \log_2(X) \rceil$ bits. Isso ocorre porque o árbitro faz com que, dos eventos entregues ao codificador, apenas um esteja ativo por vez.

Os elementos codificadores utilizados em sistemas AER arbitrados recebem, por meio de suas X entradas, os sinais $A_{ck}(x)$ referentes aos eventos gerados pela matriz de neurônios e que foram processados pelo árbitro. Esse esquema é mostrado na Fig. 3.9. Como o árbitro garante que apenas um único sinal $A_{ck}(x)$ está ativo, esse tipo de codificador possui apenas $\lceil \log_2(X) \rceil$ saídas. As saídas do codificador são conectadas diretamente ao canal de comunicação. O codificador é composto por

células de codificação (*Encoder Cell* - Enc) . Para um codificador de um sistema AER arbitrado pode haver dois tipos de células de codificação, Enc de código 1 e Enc de código 0. Essas Enc possuem um terminal de entrada (1) e um terminal de saída (2). Detalhes podem ser vistos na Fig. 3.10. As operações realizadas por essas células são descritas a seguir. Para um sinal de entrada ativo, a saída da célula de codificação pode ser ou verdadeira, caso se utilize células de código 1, ou falsa, caso se utilize células de código 0. Caso a entrada da Enc seja falsa a saída da Enc fica em alta impedância qualquer que seja o tipo de célula. A Fig. 3.10 mostra as entradas do codificador, $A_{ck}(x)$, as saídas do codificador, $D_X(x)$, e as Enc. Na figura, como apenas uma das entradas $A_{ck}(x)$ é ativada por vez, apenas as Enc de uma das colunas do codificador está em um estado diferente do de alta impedância. É essa coluna que vai determinar o código gerado na saída do codificador. Por exemplo, para uma coluna do codificador que foi ativada por um sinal $A_{ck}(x)$ se deseja gerar um código de 5 bits do tipo (1, 1, 0, 1, 0). Então é preciso escolher as CDs dessa coluna como sendo do tipo (1, 1, 0, 1, 0). Dessa maneira é possível definir o tipo de cada Enc do codificador. Foge ao escopo desse trabalho detalhar as características dos codificadores dos sistemas AER não arbitrados. Como será visto à frente, esses codificadores possuem uma alta complexidade de *hardware*, se comparado aos codificadores de sistemas AER arbitrados e ainda provocam aumento na largura do canal de comunicação.

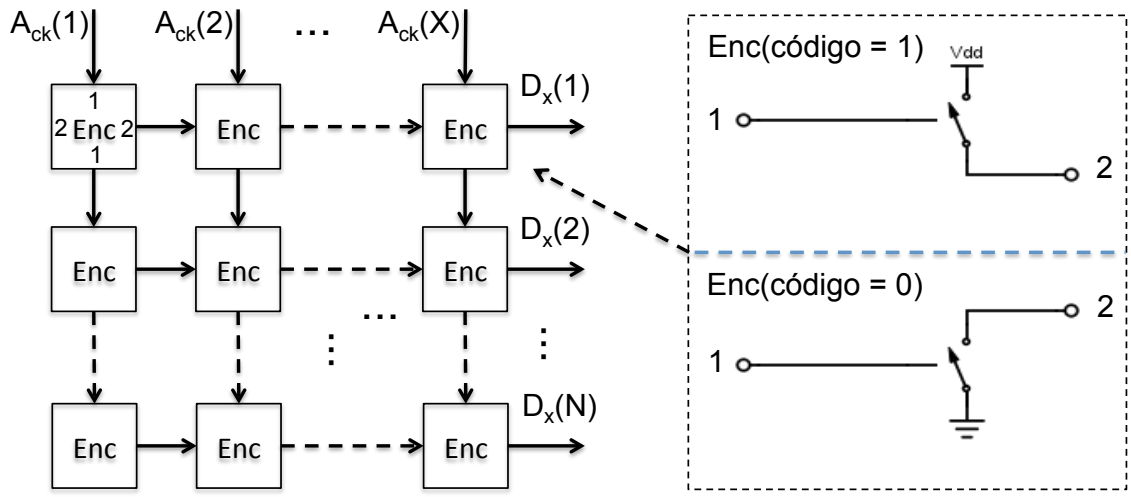


Figura 3.10: No codificador o vetor de saída D_X é uma função do vetor de entrada A_{ck}, sendo que a função é definida dessas operações de codificação Enc. Dependendo do código a ser gerado as operações Enc podem ocorrer de duas formas diferentes, operações de código 1 e operações de código 0.

De maneira complementar ao codificador utilizado em sistemas AER arbitrados o decodificador é um dispositivo de $\log_2(X)$ entradas e X saídas. Os decodificadores são formados por células de decodificação (*Decoder Cell* - Dec). Essas células

existem em dois tipos diferentes, as do tipo 1 e as do tipo 0. As Decs do tipo 1 são modeladas por chaves normalmente abertas e as do tipo 0 são modeladas por chaves normalmente fechadas. Em cada coluna do decodificador chaves do tipo 0 e do tipo 1 são colocadas em série com uma fonte de tensão, indicada por "Vdd"na Fig. 3.11. Caso o código recebido faça com que todas as chaves fiquem fechadas em apenas uma das colunas uma das saídas do decodificador ficará ativa. Nas outras colunas as saídas ficarão em estado de alta impedância. Se o código (1, 1, 0, 1, 0) for recebido na entrada do decodificador, a coluna do decodificador que deverá ser ativada deve ser composta por uma sequência de Decs do tipo (1, 1, 0, 1, 0). Um elemento de processamento D é utilizado em cada saída do decodificador para converter estados de alta impedância em estado desativado.

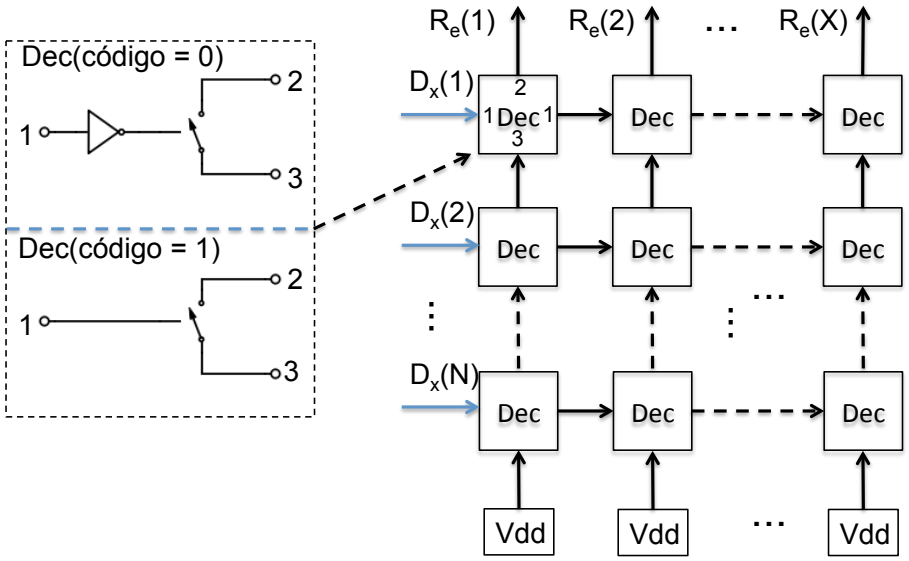


Figura 3.11: Decodificador utilizado em sistemas AER arbitrados. O decodificador é projetado de maneira que cada código recebido ative somente uma de suas saídas $R_e(x)$, deixando as outras em estado de alta impedância.

O elemento de controle é formado pelo sinal de saída da árvore binária, R_{eq} e pelo sinal de entrada do estágio transmissor, A_{ck}. O sinal R_{eq} é conectado ao estágio receptor e informa a esse estágio quando um evento será transmitido para ele. O sinal A_{ck}, por sua vez, também é conectado ao estágio receptor e serve como aviso de que o estágio receptor recebeu corretamente o evento transmitido a ele. O sinal A_{ck} é utilizado para acionar uma chave analógica que permite que as saídas $AckA$ e $AckB$ do árbitro sejam conectadas aos sistemas de *reset* dos neurônios. Como apenas um dos sinais de saída do árbitro estará ativo, apenas um neurônio será reiniciado.

3.2 Latência em Sistemas AER

A implementação de circuitos corticais utilizando técnicas AER possibilita redução na complexidade de *hardware* envolvida. Contudo, quanto maior a redução na complexidade de *hardware* maior é a distorção do sinal a ser transmitido. Por exemplo, a sequência e, nas condições (e_x, e_y, e_t), transmitida até um estágio receptor não pode ser reconstruída perfeitamente caso uma técnica AER esteja sendo utilizada com o objetivo de reduzir a complexidade de *hardware* do projeto. O árbitro, ao evitar que dois ou mais eventos que foram gerados simultaneamente sejam transmitidos também simultaneamente, insere uma distorção temporal na sequência e. Eventos gerados simultaneamente são transmitidos em sequência pelo canal de comunicação considerando a regra de preferência determinado pelo árbitro. Essa distorção é chamada aqui de latência. Será estimada a latência envolvida com duas técnicas AER arbitradas, a que utiliza árvore binária e a que utiliza árbitro do tipo WTA. Iniciaremos com o AER que utiliza árvore binária. Elementos de processamento do tipo LC e D possuem um atraso de resposta de cerca de 0.25 ns cada. Esse dado é o resultado da simulação desses dois elementos utilizando a tecnologia de 0.35 μm da AMS. Utilizando o mesmo artifício de simulação foi possível encontrar um atraso de resposta para uma CD da árvore binária de 0.33 ns. Para um codificador o atraso de resposta é de cerca de 0.8 ns. Um decodificador tem atraso de resposta também de cerca de 0.8 ns. Para o árbitro AB o atraso de resposta é dado por $(2 \cdot 0.33 \cdot \lceil \log_2(X) \rceil)$ ns, sendo que o fator logarítmico se deve as $\log_2(X)$ camadas de processamento. O tempo é dobrado pois o processamento é feito nos sentidos de ida e volta das camadas de processamento. Cada uma das CDs tem um atraso de resposta de 0.33 ns. Em um sistema AER, para que um ciclo de comunicação seja completado, uma requisição deve ser feita por um neurônio. Essa requisição precisa ser processada por um elemento LC e outro D. Um árbitro processa essa requisição e um codificador gera um código referente a essa requisição que é transmitido pelo canal de comunicação até o decodificador. O decodificador processa esse código e gera uma requisição relativa a requisição gerada no transmissor. Elementos D processam essa nova requisição e o evento é considerado entregue ao receptor. Uma resposta, de que a transmissão foi bem-sucedida, é enviada até o transmissor. Essa resposta é processada por um elemento D e então o neurônio é reiniciado. Considerando que as colisões sempre causam latência, no caso de ocorrerem k colisões a latência, t_{AB}, da árvore binária pode ser estimada pela Eq. (3.1). A última requisição a ser atendida terá que esperar k ciclos de comunicação.

$$t_{AB} = 0.66 \cdot 10^{-9} \cdot k \cdot \lceil \log_2(X) \rceil + 2.6 \cdot 10^{-9} \cdot k \tag{3.1}$$

Em um árbitro do tipo WTA a disputa ocorre de maneira todos contra todos

em apenas uma camada de processamento e em um sentido único da entrada para a saída. Essa disputa toma cerca de em 0.3 ns, valor medido por simulação utilizando a tecnologia de 0.35 μm da AMS. Caso ocorram k colisões o atraso de resposta do árbitro WTA é de cerca de $0.3 \cdot k$ ns. Como o ciclo de comunicação no sistema AER que utiliza árbitro WTA possui tempo de resposta equivalente ao ciclo de comunicação do árbitro AB, exceto pelo tempo de resposta do árbitro, temos que a latência no caso do árbitro de WTA, t_{WTA}, pode ser estimada pela Eq. (3.2).

$$t_{WTA} = 0.3 \cdot 10^{-9} \cdot k + 2.6 \cdot 10^{-9} \cdot k \tag{3.2}$$

A latência causada por um AER não arbitrado, t_{NA}, pode ser estimada caso seja desconsiderado o tempo de latência causado pelo árbitro AB ou WTA nas Eqs. (3.1) ou (3.2). A Eq. (3.3) mostra a expressão para t_{NA}.

$$t_{NA} = 2.6 \cdot 10^{-9} \cdot k \tag{3.3}$$

A Tabela 3.1 mostra o valor das latências máximas causadas pelos elementos de um sistema AER em três diferentes topologias, AB, WTA e NA (Não Arbitrado). Foram utilizadas matrizes de neurônios dimensões 50^2 e 100^2. Os tempos de atraso apresentados na tabela são estimados a partir das medidas feitas por simulação dos elementos básicos, já mencionadas. No cálculo foi considerado o ciclo de comunicação já mencionado.

Tabela 3.1: Latências máximas (em ns) associadas ao processamento dos elementos de três diferentes técnicas AER (AB, WTA e NA) para um total $(X \cdot Y) = 50^2$ e $(X \cdot Y) = 100^2$ neurônios.

Elemento	AB 50^2	WTA 50^2	NA 50^2	AB 100^2	WTA 100^2	NA 100^2
Árbitro	186	15	0	438	30	0
D+LC	0.5	0.5	0.5	0.5	0.5	0.5
Codificador	0.8	0.8	0.8	0.8	0.8	0.8
Decodificador	0.8	0.8	0.8	0.8	0.8	0.8
Transmissão	251	80	2.3	553	145	2.3
Recepção	77.5	77.5	2.6	155	155	2.6
Total	660	157.5	4.9	942	300	4.9

3.3 Complexidade de Hardware em Sistemas AER

Nesse trabalho a complexidade de *hardware* está relacionada à quantidade de transistores necessários para implementação de circuitos corticais em silício. Assumimos que esses transistores possuem dimensões W e L próximas àquelas que são indicadas mínimas pela tecnologia de fabricação (nesse caso utilizando a tecnologia 0.35 μm da AMS). Nessa condição garantimos que apenas a medida da quantidade de transistores seja um indicador para a complexidade de *hardware*. Caso a condição de trabalhar com transistores que apresentem dimensões de W e L muito maiores do que as dimensões mínimas não seja atendida, um indicador relacionado a área deve também ser adotado. Será discutido a seguir a complexidade de *hardware* envolvida com sistemas AER arbitrados AB, WTA e não arbitrados NA. Começaremos pelos sistemas AER AB. Os neurônios em sistemas AER AB possuem complexidade de *hardware* de sete transistores, Fig. 2.8. Os elementos de processamento LC e D dos sistemas AER AB possuem complexidade de *hardware* de 3 e 18 transistores respectivamente. No cálculo foi usada a Fig. 3.5 como base e foi considerado que portas lógicas inversoras são formadas por, em média, dois transistores cada. Um árbitro do tipo AB que pode arbitrar entre X entradas deve possuir $\sum_{n=1}^{\lceil \log_2(X) \rceil} \lceil \frac{X}{2^n} \rceil$ CDs em sua estrutura. A composição de cada estrutura digital biestável que forma as CDs, que é composta de seis portas lógicas, é mostrada na Fig. 3.7. Dessas seis portas uma porta é inversora e as outras são portas AND ou OR. Assumindo que cada uma das cinco portas lógicas restantes (AND ou OR) é composta por, em média, quatro transistores, é possível estimar a complexidade de *hardware* da árvore binária, C_{AB}, pela Eq. (3.4).

$$C_{AB} \approx 22 \cdot \sum_{n=1}^{\lceil \log_2(X) \rceil} \lceil \frac{X}{2^n} \rceil \tag{3.4}$$

Os codificadores de sistemas AER AB possuem X entradas e $\lceil \log_2(X) \rceil$ saídas, sendo formados por $(X \cdot \lceil \log_2(X) \rceil)$ Enc. Como as Enc são compostas de um transistor cada (Fig. 3.10), a complexidade de *hardware* do codificador é $(X \cdot \log_2(X))$ transistores. Pelos mesmos motivos a complexidade de *hardware* do decodificador é $(X \cdot \log_2(X))$ transistores. Nos sistemas AER WTA a complexidade de *hardware* do neurônio é de 21 transistores. 14 desses transistores compõem os elementos de processamento LC e D (Fig. 3.5, parte b) que são integrados ao neurônio e os outros 7 compõem o circuito do neurônio de integração e disparo (Fig. 2.8). O codificador e o decodificar utilizados nessa técnica possuem complexidade de *hardware* igual à do codificador utilizado para implementar um sistema AER AB, $(X \cdot \log_2(X))$ transistores cada um. Já nos árbitros do tipo WTA, como é mostrado na Fig. 3.8, as CDs são compostas de pelo menos 13 transistores cada. Dessa maneira a complexidade

de *hardware* do árbitro WTA, C_{WTA}, pode ser estimada pela Eq. (3.5), sendo X a quantidade de entradas do árbitro. Nesse árbitro existe uma rede resistiva de polarização. Por não ter sido informado como essa rede foi implementada em [34], Será desconsiderado o aumento na complexidade de *hardware* do árbitro provocado por essa rede de polarização. Como a complexidade de *hardware* envolvida em sistemas AER WTA, desconsiderando a complexidade de *hardware* dessa rede de polarização, ainda é maior do que a do sistema AER AB, essa desconsideração não causará mudanças nas conclusões que serão apresentadas.

$$C_{WTA} \approx 13 \cdot X \tag{3.5}$$

Tabela 3.2: Complexidade de *hardware*, com valores dados em transistores (ou em bits para o canal), envolvida com os elementos de três diferentes técnicas AER, AB ([33]), WTA ([34]) e NA ([35]). $X \cdot Y = 50^2$ e 100^2 neurônios.

Elemento	AB 50^2	WTA 50^2	NA 50^2	AB 100^2	WTA 100^2	NA 100^2
Árbitro	2500	700	0	6000	1300	0
Neurônio	17500	52500	17500	70000	210000	70000
LC	7000	0	7000	30000	0	30000
D	1800	0	1800	3600	0	3600
Codificador	300	300	3600	700	700	16800
Decodificador	300	300	3600	700	700	16800
Transmissor	32400	54500	34000	117000	214000	137200
Receptor	10718	10718	10800	50418	50418	40800
Canal	14 bits	14 bits	36 bits	16 bits	16 bits	42 bits

O canal de comunicação de um sistema AER arbitrado (AB ou WTA) tem complexidade de $(\log_2(X) + \log_2(Y))$ bits, além dos sinais de controle A_{ck}, R_{eq} de um bit cada. Na técnica AER NA cada neurônio da matriz possui complexidade de *hardware* semelhante àquela apresentada pelos sistemas AER que utilizam árvore binária, sete transistores. De maneira análoga ao caso anterior, os elementos de processamento LC e D dos sistemas AER NA possuem complexidade de *hardware* de três e 18 respectivamente. Como nessa técnica não existem árbitros, a complexidade de *hardware* é nula. Nessa técnica a complexidade de *hardware* dos codificadores e

decodificadores varia de acordo com o tipo de AER não arbitrado. Será considerado aqui um caso onde a complexidade de *hardware* do codificador (e do decodificador) é de $(4 \cdot X \cdot (\log_2(X) \cdot k))$ transistores [35]. A complexidade do canal de comunicação é igual a $(\log_2(X) \cdot k)$. Além dos dois sinais de controle A_{ck}, R_{eq}. k é a quantidade de colisões que são aceitas pelo sistema AER e que podem ser recuperadas no estágio do receptor. A Tabela 3.2 compara a complexidade de *hardware* dos elementos das três técnicas AER mencionadas na Tabela 3.1. No caso da topologia não arbitrada que aparece em ambas as tabelas o valor de k, a quantidade máxima de conflitos solucionáveis pelo decodificador, é igual a três.

3.4 Comparação entre Sistemas AER

A Tab 3.1 mostra que a latência no sistema AER NA (4.9 ns) é menor do que a latência apresentada pelo sistema AER AB tanto para o caso onde $X \cdot Y = 50^2$ quanto para o caso $X \cdot Y = 100^2$. É possível dizer o mesmo quando comparamos a latência (157.5 e 300 ns) do sistema AER WTA em relação a latência (660 e 942 ns) do sistema AER AB. Como foi visto, a latência aumenta à medida que a quantidade de neurônios aumenta na matriz do estágio transmissor. Em implementações de circuitos neurais muito grandes, baseados em representações pulsadas, níveis excessivos de latência podem descaracterizar completamente a informação a ser transmitida. Por esse motivo, algumas implementações preferem substituir a técnica AER AB pela técnica AER NA [5], mesmo que isso represente um aumento significativo em relação a complexidade de *hardware*. Na Tab. 3.2 são apresentados valores que fundamentam a argumentação. A complexidade de *hardware* do transmissor AER AB é de 32400 transistores, no caso onde $X \cdot Y = 50^2$, contra 34000 transistores apresentados pelo transmissor AER NA. No caso onde $X \cdot Y = 100^2$ o quadro é mantido, 117000 transistores do AER AB contra 137200 transistores do AER NA. Além da maior complexidade de *hardware* apresentado pelos sistemas AER NA, em relação ao AB, a complexidade do canal de comunicação também é maior, 36 bits do sistema AER NA contra 14 bits do sistema AER AB. Um canal de comunicação mais complexo implica em um receptor onde a complexidade de *hardware* é maior. Embora os sistemas AER WTA apresentem um canal de comunicação equivalente ao apresentado pelo sistema AER AB e uma latência menor, a complexidade de *hardware* de um transmissor AER WTA é quase o dobro da complexidade de *hardware* de um transmissor AER AB (campo transmissor na Tab. 3.2).

3.5 Técnica AER Proposta

Com base na Tab. 3.2, vamos propor um sistema AER não arbitrado (no sentido de não garantir que todos as colisões sejam resolvidas) de latência equivalente aquela apresentada pelos sistemas AER WTA e complexidade de *hardware* compatível com àquela apresentada pelos sistemas AER AB. Para isso será explorado o compromisso existente entre a colisão e a latência [12]. Esse compromisso implica em que, ao se permitir que algumas transmissões sejam perdidas por colisões a latência na transmissão é reduzida. Seguem as explicações que justificam a existência desse compromisso. Na prática uma colisão não acontece exatamente quando dois eventos são gerados no mesmo instante de tempo. Em um sistema AER não arbitrado, assim que um evento é gerado, existe um intervalo de tempo T que começa quando uma requisição $R_{eq}(x, y)$ é emitida pelo neurônio e termina quando esse mesmo neurônio recebe uma confirmação de atendimento a sua requisição, $A_{ck}(x, y)$. Caso um outro evento aconteça durante esse intervalo de tempo T, uma colisão ocorrerá. O intervalo de tempo T pode ser estimado através da soma dos tempos de resposta dos elementos de processamento, do codificador, do canal de comunicação e do tempo de resposta do estágio receptor. Para um sistema AER não arbitrado, T não varia em relação à quantidade de neurônios, possui valor estimado de 5 ns (estimativa obtida utilizando a tecnologia de 0.35 μm da AMS). A probabilidade P de k eventos ocorrerem dentro do intervalo de tempo T é dada pela pela Eq. (3.6) [26], [35]. A Eq. (3.6) é estimada a partir da distribuição de Poisson.

$$P = \frac{((X \cdot Y) \cdot F_n \cdot T)^k}{k!} \cdot e^{-(X \cdot Y) \cdot F_n \cdot T} \tag{3.6}$$

A Fig. 3.12 mostra a probabilidade de colisão entre dois eventos, função da quantidade total de neurônios $(X \cdot Y)$, para três diferentes intervalos de tempo T, 1, 10 e 100 ns. A frequência média de eventos produzidos por cada um dos neurônios, F_n, é de 100 Spk/s na simulação e k é a quantidade de eventos que se envolveram em uma colisão. Utilizando a Eq. 3.6 é possível estimar uma probabilidade de colisão em torno de 0.25% ($T = 5$ ns e um total de neurônios $X \cdot Y = 4000$) para um sistema AER não arbitrado. A seguir será proposto uma técnica AER não arbitrada que reduz essa probabilidade de colisão. Sistemas AER não arbitrados que são capazes de reduzir a probabilidade de ocorrência de colisões sem alterar a complexidade do canal de comunicação (em níveis equivalentes àqueles apresentados pelos sistemas AER AB), até onde foi pesquisado, não foram relatados na literatura.

Para reduzir a probabilidade de colisões P entre eventos, em um sistema AER não arbitrado introduziremos um dispositivo, que é chamado de mecanismo de redução de probabilidade (MRP), ao sistema AER não arbitrado. Serão escolhidos os elementos que compõem o MRP de tal maneira que a complexidade de *hardware* do

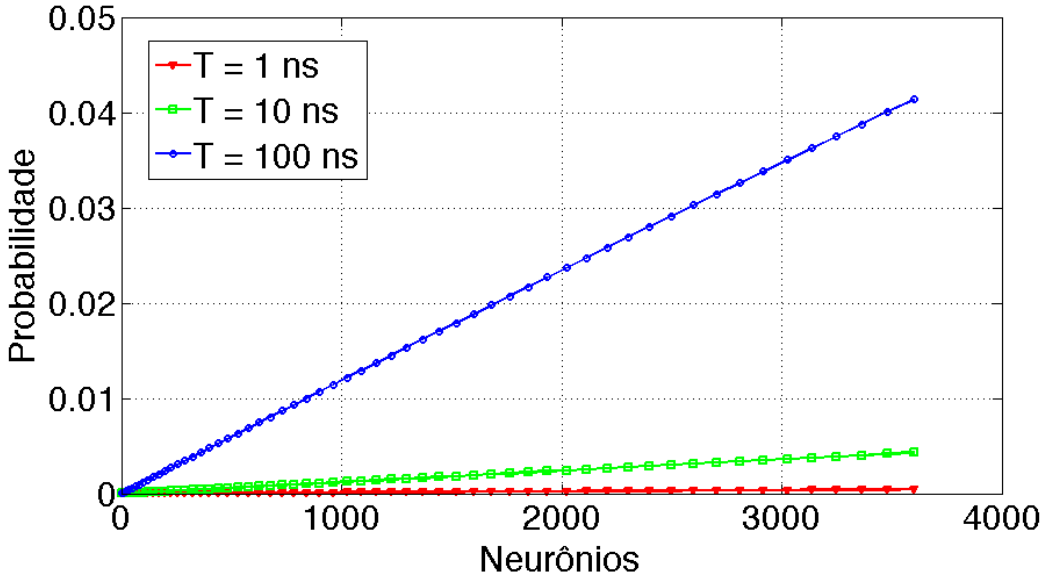


Figura 3.12: A probabilidade de colisão, com função de uma população de neurônios e com uma taxa de disparos de 100 Spk/s. São mostradas três curvas, sendo cada uma delas para um dado T.

sistema AER não arbitrado, inclusive a do canal, se mantenha equivalente àquela apresentada pelos sistemas AER arbitrados AB. A estratégia consiste em impedir que um novo evento ocorra dentro de um ciclo de comunicação de um evento. O MRP é composto por um mecanismo capaz de identificar se ocorreram um ou mais eventos (o que foi denominado de gerenciador de eventos) e alguns circuitos auxiliares que impedem que novos eventos ocorram caso um evento tenha sido identificado e um ciclo de comunicação iniciado. Assumimos nesse trabalho que um sistema AER arbitrado é aquele que é capaz de tratar todas as colisões que ocorram entre os eventos. Como o sistema proposto aqui não é capaz de evitar a colisão entre eventos que possam ocorrer no início do ciclo de comunicação, dentro da nossa definição, esse não se trata de um sistema AER arbitrado. Detalhes sobre o MRP serão apresentados a seguir.

Inicialmente será descrito o funcionamento do neurônio que é utilizado no AER não arbitrado proposto. A Fig. 3.13 consiste de um neurônio semelhante ao apresentado em [33]. Propomos a inclusão de uma chave analógica que permite ativar ou desativar a entrada do neurônio (terminal 1), conforme o controle da chave, o terminal 5 (sinal stop). O terminal 2 é o controle de *reset* geral, esse sinal é utilizado caso seja preciso reiniciar todos os neurônios da matriz de uma vez. Eventos são

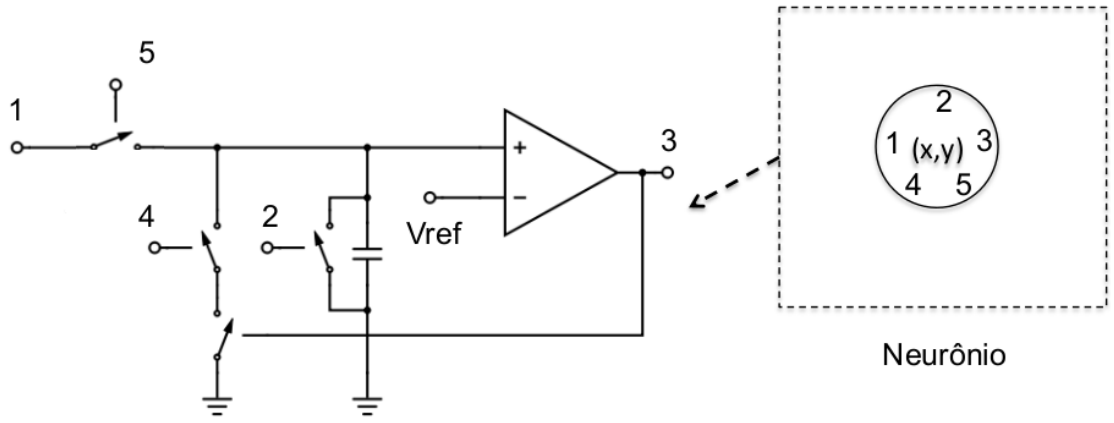


Figura 3.13: Diagrama esquemático do neurônio utilizado no sistema AER não arbitrado proposto. O neurônio apresentado na Fig. 2.8 sofre duas modificações, uma chave analógica é adicionada a sua entrada (chave analógica controlada pelo terminal 5) e apenas um sinal A_{ck} é utilizado para reiniciar o neurônio, terminal 4. Devido as alterações feitas o diagrama esquemático alcança uma complexidade de *hardware* de 8 transistores, 1 transistor a mais do que na Fig. 2.8.

gerados pelo terminal 3, sinal R_{eq}. Quando apenas um evento é gerado pela matriz de neurônios dentro de um mesmo ciclo de comunicação, o ciclo de comunicação prossegue até o final. Caso contrário, os neurônios que provocaram o evento são reiniciados antes do final do ciclo de comunicação. Isso possibilita que o mesmo sinal A_{ck} seja utilizado para reiniciar qualquer neurônio da matriz que tenha gerado o evento, terminal 4.

As explicações sobre o gerenciador de eventos serão dadas a seguir. Na Fig. 3.14 cada evento $R_{eq}(x)$ faz com que o espelho de corrente formado pelos transistores de canal N, e consequentemente o espelho formado por transistores de canal P, conduzam uma corrente de valor (k·i), onde k é a quantidade de eventos que colidiram e i é uma corrente de *bias*. Como os eventos $R_{eq}(x)$ controlam chaves conectadas as fontes de corrente de *bias* i, k eventos acionam k chaves. Quando k chaves estão ativas uma corrente total de (k·i) A é conduzida pelos espelhos de corrente. Dois comparadores de corrente indicados pelo nome Icomp recebem em suas entradas a corrente de valor (k·i) A. Os comparadores comparam essa corrente (k·i) com as quantidades fixas de corrente 0.1·i e 1.5·i A. O sinal Event indica quando um evento é gerado pela matriz de neurônios e o sinal Conf indica se a quantidade de eventos gerados, k, é maior ou igual a dois. Elementos de processamento LC e D interligam as saídas R_{eq} dos neurônios até as entradas R_{eq} do gerenciador de eventos. O arranjo composto pelos elementos LC e D é o mesmo apresentado para o sistema AER AB. As requisições feitas pelos neurônios da matriz são tratadas a nível de linha e de coluna, por isso existe a necessidade de dois gerenciadores de eventos por matriz de neurônios. Em sistemas AER arbitrados, na saída da matriz de neurônios, não

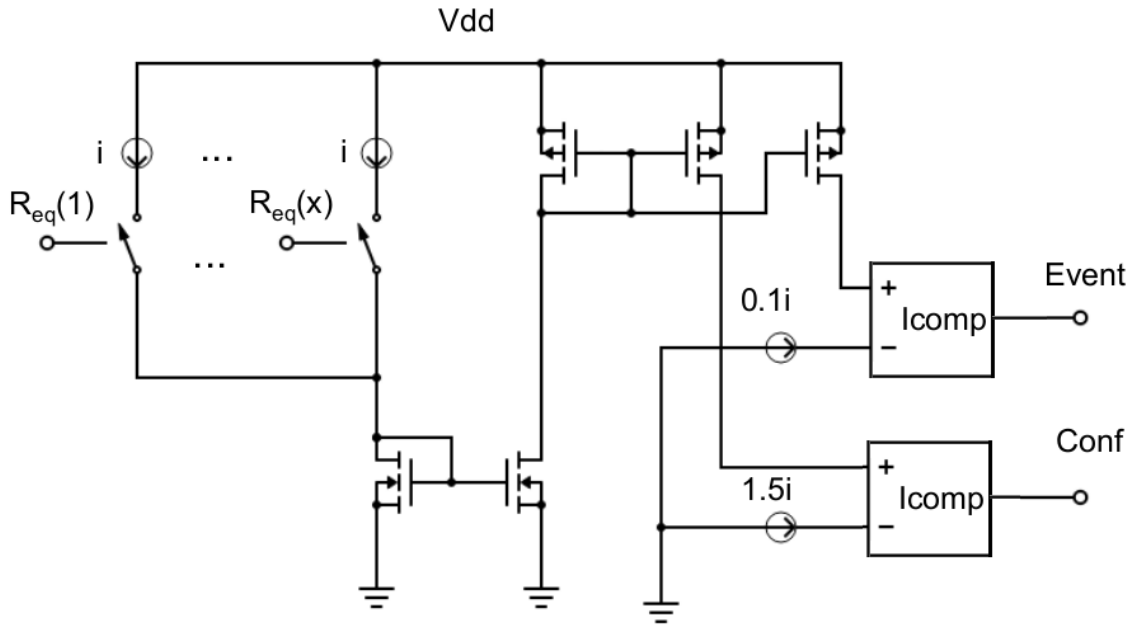


Figura 3.14: Diagrama esquemático do gerenciador de eventos. Esse dispositivo é capaz de identificar quando um evento é gerado e se a quantidade de eventos gerados é igual ou maior que dois.

existe uma regra que impeça que mais do que um evento ocorram mesmo durante um ciclo de comunicação normal. Na saída dos árbitros essa regra é imposta. Esses sistemas AER trabalham com dois tipos de sinais R_{eq} simultaneamente, um antes do árbitro e outro após o árbitro. Por esse motivo existem memórias que armazenam os estados dos sinais R_{eq} das saídas do árbitro durante o ciclo de comunicação. Como no sistema AER proposto existe apenas um tipo de sinal R_{eq}, essas memórias não são incluídas.

A Fig. 3.15 mostra como é a lógica de controle utilizada pela topologia proposta. Dois gerenciadores de evento geram os sinais Event(x), Event(y), Conf(x) e Conf(y). Esses sinais são conectados as entradas de mesmo nome do bloco que realiza a lógica de controle. Esse bloco lógico é conectado a matriz de neurônio por meio dos sinais Reset e Stop. Esses sinais são conectados aos neurônios por meio dos terminais de entrada 4 e 5 respectivamente. O bloco lógico se comunica com o estágio receptor por meio dos sinais Req e Ack. Dois codificadores iguais aos utilizados nos sistemas AER AB são utilizados para a topologia proposta. As entradas dos dois codificadores são os sinais $R_{eq}(x)$ e $R_{eq}(y)$ e as saídas são os $(\lceil \log_2(X) \rceil + \lceil \log_2(Y) \rceil)$ bits que compõem o canal de comunicação. Discutiremos agora a latência causada pelo sistema AER proposto. Como a ativação do sinal Stop reduz a zero a possibilidade de geração de novos eventos, o intervalo de tempo T é reduzido de 5 ns (um ciclo de comunicação) para 0.8 ns. Esse tempo de 0.8 ns foi simulado

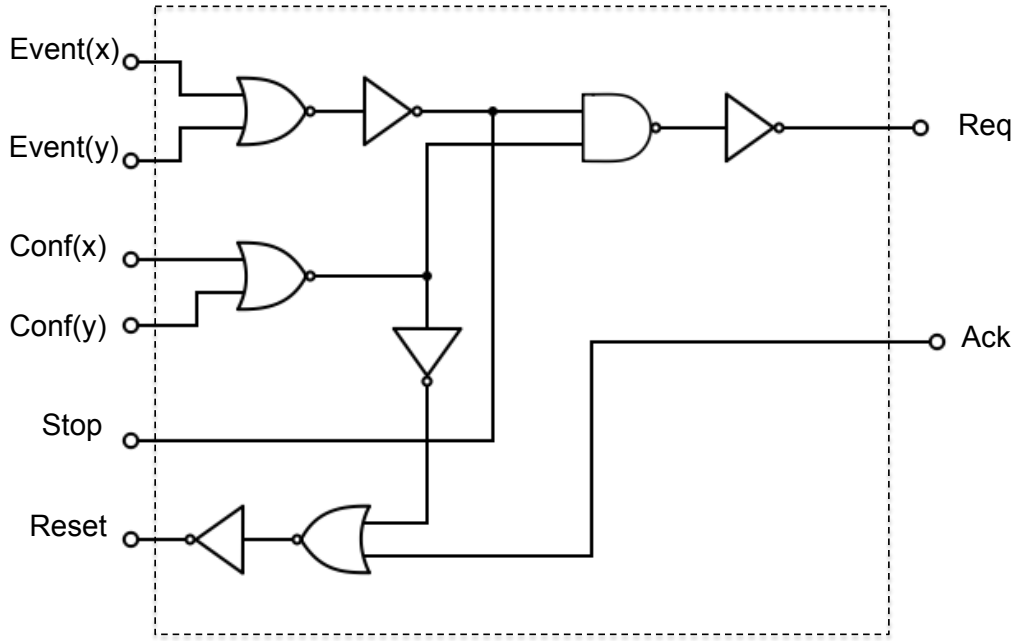


Figura 3.15: Lógica de comunicação utilizada, na topologia proposta, para a ligação entre os sinais de saída do gerenciador de eventos e os sinais de entrada da matriz de neurônios.

utilizando a tecnologia de 0.35 μm da AMS. A redução no valor do intervalo de tempo T implica em uma redução da probabilidade de ocorrer colisão. Utilizando a Eq. 3.12 foi possível estimar o valor de 0.04% para a probabilidade de ocorrer colisão P. Essa estimativa utilizou $X \cdot Y = 4000$, a mesma condição que foi utilizada na estimativa $P = 0.25\%$ para $T = 5$ ns. Entretanto, ao desabilitar a possibilidade de um evento ser gerado, um atraso é inserido e a latência na comunicação aumenta. Se durante um ciclo de comunicação do sistema AER não arbitrado proposto k eventos são evitados, $k-1$ novos ciclos de comunicação deverão ocorrer até que todos os eventos sejam transmitidos. Contudo, caso z desses eventos restantes se envolvam em colisões, serão necessários $k-z$ novos ciclos de comunicação ao invés dos $k-1$ ciclos. É possível estimar a latência do sistema AER proposto como sendo $t_P = 5 \cdot (k-z)$ ns. Essa latência é menor do que àquelas apresentadas pelos sistemas AER arbitrados AB e WTA. Passaremos agora para a questão da complexidade de *hardware* da topologia AER proposta. A técnica AER proposta tem um transistor a mais do que a que foi apresentada em [33], ou seja, tem oito transistores por neurônio. A estrutura LC é de complexidade igual a dois transistores, veja a Fig. 3.4 para detalhes. A estrutura D tem complexidade de *hardware* que segue o esquema apresentado na Fig. 3.4 e que possui complexidade de *hardware* de três transistores. Esse valor é alcançado porque não são necessárias memórias auxiliares nas estruturas D como acontece em [33]. A técnica AER proposta não possui árbitro, logo a complexidade de *hardware* envolvida com o árbitro é nula. A complexidade de

hardware envolvida com o gerenciador de eventos é de $(2 \cdot X + 17)$ transistores, ver Fig. 3.14. A estimativa feita considera que cada comparador de corrente é formado por 6 transistores. X é o número total de entradas do gerenciador. Para o codificador e o canal de comunicação a complexidade de *hardware* é igual à complexidade de *hardware* das partes correspondentes no sistema AER AB. A Tabela 3.3 mostra a análise feita sobre o sistema AER proposto e um comparativo envolvendo os sistemas AER AB ([33]) e não arbitrado [35]. O sistema AER não arbitrado proposto é indicado pela sigla NAP .

Tabela 3.3: Sistema AER não arbitrado proposto. São mostrados: latência (em ns), probabilidade de colisão e complexidade de *hardware* (em transistores ou em bits no caso do canal de comunicação). Esses dados são comparados aos dados produzidos a partir das técnicas AER AB [33] e NA [35]. O número total de neurônios foi de $X \cdot Y = 50^2$ e $X \cdot Y = 100^2$.

Elemento	AB 50^2	NA 50^2	NAP 50^2	AB 100^2	NA 100^2	NAP 100^2
Árbitro	2500	0	0	600	0	0
Anti-Colisão	0	0	232	0	0	432
Neurônio	17500	17500	20000	70000	70000	80000
Codificador	300	0	300	700	0	700
Decodificador	300	0	300	700	0	700
Transmissor	32400	34000	26364	117000	137200	102864
Canal	14 bits	29 bits	14 bits	16 bits	54 bits	16 bits
Atraso	251	2.3	<80	553	2.3	<145
Colisão	0	0.078	0.016	0	0.310	0.050

3.6 Resultados para o AER NAP

Um diagrama esquemático de um sistema AER NAP foi desenvolvido. Esse diagrama foi utilizado para transmitir eventos de uma matriz de neurônios de dimensões 50 x 50, onde 50% das fontes de corrente (estímulo dos neurônios) nas entradas do conjunto foram colocadas em um nível de 10 pA e as outras fontes foram zeradas. Foi feita uma simulação para estimar a taxa máxima de transferência permitida pelo

sistema AER proposto. Os resultados relacionados a essa simulação são apresentados na Tab. 3.4. A taxa máxima de transferência é importante para mostrar que a latência do AER NAP é equivalente aquela apresentada por um sistema AER WTA, como mostrou a previsão teórica. Isso devido ao fato de que quem limita a taxa máxima de transferência de um sistema AER arbitrado é a latência. Caso a latência seja reduzida é possível atingir níveis maiores de taxa de transferência. Na tabela é possível perceber que a taxa máxima de transferência do sistema AER proposto é quase o dobro da apresentada por um sistema AER AB e mais próximo aos valores atingidos por um sistema AER de árbitro WTA. Na simulação o correto seria por o valor de todas as fontes com o mesmo nível, contudo a limitação na capacidade computacional disponível não permitiu. Detalhes do diagrama esquemático do sistema AER proposto são apresentados no Cap. 4.

Tabela 3.4: Comparativo das taxas máximas de transferência entre as técnicas AER proposta (NAP) e duas outras, AB e WTA. A técnica proposta atinge quase o dobro da taxa máxima de transferência da técnica AB e se aproximou do resultado produzido pela técnica WTA. Os valores de taxa máxima de transferência para as técnicas WTA e AB foram retirados de [32] e [34]. As condições de simulação impostas para o NAP foram semelhantes àquelas que produziram os resultados da técnica AB e da WTA.

Caraterística	NAP	AB([32])	WTA([34])
Taxa de Transferência [Spk/s]	$5.1 \cdot 10^7$	$2.5 \cdot 10^7$	$8.3 \cdot 10^7$

3.7 Conclusões

Nesse capítulo foi proposta uma técnica AER não arbitrada que é uma alternativa as técnicas AER AB. O resultado de previsões teóricas apresentados na Tab. 3.3 mostram que o sistema AER proposto possui latência menor do que a apresentada pelo sistema AER AB e equivalente a apresentada pelo sistema AER WTA. A complexidade de *hardware* do transmissor que utiliza a técnica proposta é equivalente a apresentada pelo sistema AER AB e quase 50% menor do que a apresentada pelo sistema AER WTA. Os resultados de simulação apresentados na Tab. 3.4 confirmam que a técnica proposta é capaz de superar a técnica AER AB em relação a taxa de transferência, um indicativo concreto de que a latência envolvida com a comunicação na técnica proposta é menor. O sistema AER proposto, por não ser um sistema AER arbitrado, permite colisões. Contudo a probabilidade de colisão apresentada pelo sistema AER proposto é de 0.05% contra 0.25% de um sistema não arbitrado tradicional.

Capítulo 4

A Retina CMOS

O diagrama esquemático de um imageador de dimensões 50 x 50 foi desenvolvido para possibilitar os testes de simulação e também para auxiliar no tratamento dos estímulos que serão utilizados como entradas para o DFI. Esse diagrama esquemático é composto por uma matriz de 50 x 50 pixels. Esses pixels são compostos de uma unidade da OPL (diagrama esquemático apresentado na Fig.2.9), um retificador (diagrama esquemático apresentado na Fig. 2.15) e duas células ganglionares. O diagrama esquemático do pixel pode ser visto na Fig. 4.1, a figura mostra em destaque os diagramas esquemáticos da OPL, IPL e célula ganglionar. Cada pixel recebe estímulo de um fotodiodo, uma fonte de corrente de valor igual a Ix_y. Os símbolos x e y estão relacionados as posições dos fotodiodos dentro da matriz.

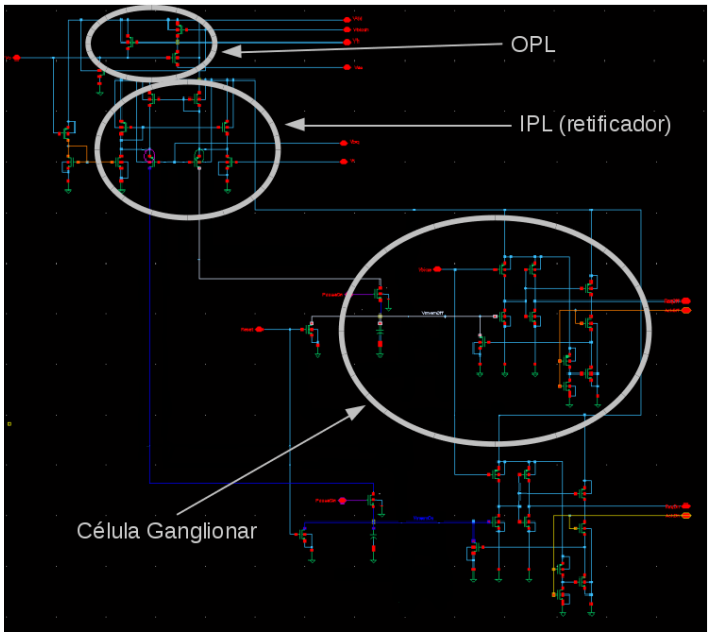


Figura 4.1: A figura mostra o diagrama esquemático de um dos 50 x 50 pixels do imageador que foi utilizado para realização dos testes de simulação. Na figura é possível identificar os diagramas esquemáticos da OPL, IPL e das células ganglionares.

As conexões feitas entre os pixels, os fotodiodos, os pseudo-resistores e as estruturas LC (diagrama esquemático apresentado na Fig. 4.4) podem ser vistas na Fig. 4.2. Uma ilustração mostrando o diagrama esquemático da matriz 50 x 50 completa e o sistema de comunicação é apresentada na Fig. 4.3.

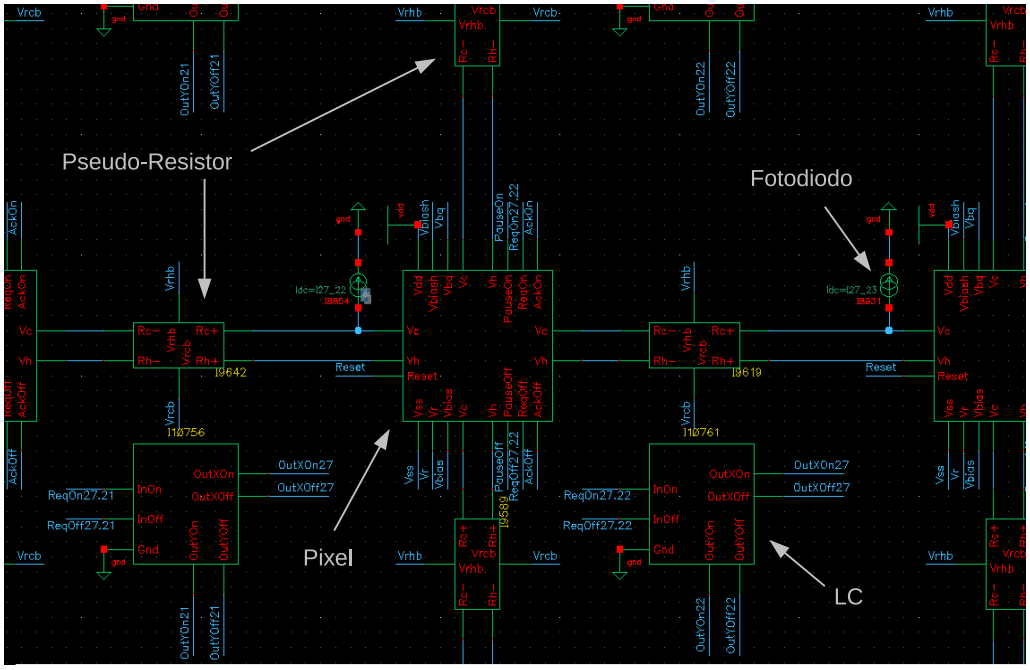


Figura 4.2: A figura mostra detalhes das conexões entre os pixels do diagrama esquemático da matriz do imageador. É possível ver como são as conexões entre os fotodiodos, o pixel, os pseudo-resistores e as estruturas LC.

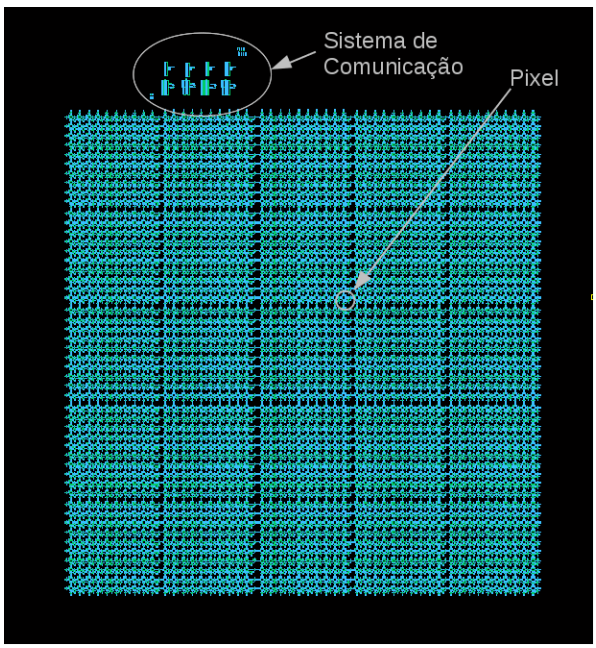


Figura 4.3: O diagrama esquemático da matriz de pixels é mostrado na figura. Um pixel e o sistema de comunicação estão destacados.

O diagrama esquemático da estrutura LC utilizada é mostrado na Fig. 4.4 e o da estrutura D é apresentado na Fig. 4.5. No diagrama esquemático a estrutura LC recebe os eventos gerados pelos terminais de saída ReqOn dos pixels pelo seu terminal de entrada InOn. A estrutura LC gera dois eventos OutXOn e OutYOn (a nível de linha e de coluna) para cada sinal ReqOn.

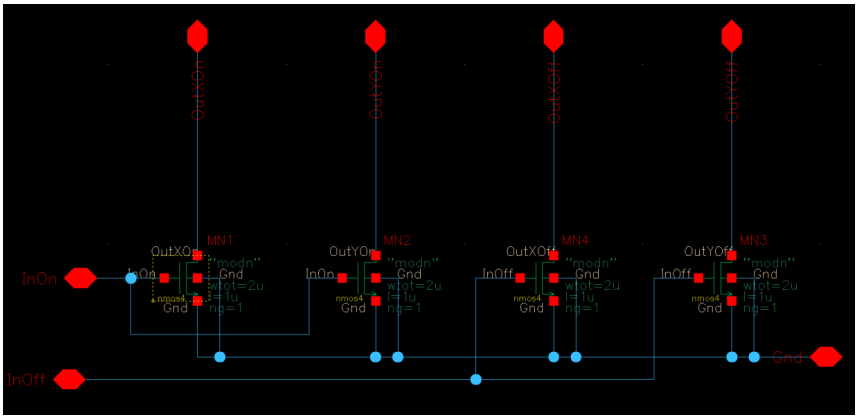


Figura 4.4: Detalhes dos sinais e o diagrama esquemático da estrutura LC. Os sinais de entrada são InOn e InOff e os sinais de saída são OutXOn, OutXOff, OutYOn e OutYOff.

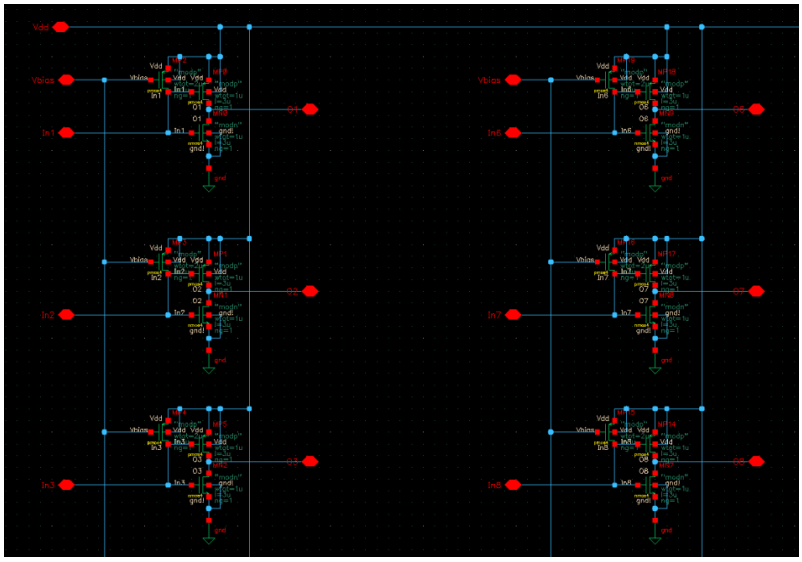


Figura 4.5: A figura mostra o diagrama esquemático utilizado para a estrutura D. Os sinais de saída da estrutura LC são recebidos a partir dos sinais de entrada Inx e Iny. Os sinais de saída da estrutura D são os sinais Ox e Oy. Com x e y variando de 1 até 50.

A saída da matriz de pixels é feita pelas estruturas LC e consiste de 50 sinais (OutXOn1 até OutXOn50) que representam as linhas da matriz e 50 sinais (OutYOn1 até OutYOn50) que representam as colunas da matriz. Esses 100 sinais de saída da matriz de pixels são transmitidos para as entradas In de 100 estruturas

D. O diagrama esquemático de uma estrutura D é apresentado na Fig. 4.5. Um sistema AER, como o que foi proposto (NAP), foi adotado para gerenciar a comunicação de dados da matriz de pixels (OPL, IPL e célula ganglionar). Esse sistema AER é composto por estruturas LC conectadas a cada pixel na matriz, por $50 + 50$ estruturas D para as linhas e colunas da saída da matriz, por dois sistemas gerenciadores de evento (linha e coluna) como o apresentado na Fig. 4.6, dois sistemas codificadores a nível de linha e de coluna (Fig. 4.7) e dois sistemas de controle (Fig. 3.15) a nível de linha e de coluna. O sistema gerenciador de eventos possui entradas de P1 até P50 e saídas Event e Comp. O sinal Event indica que um evento foi recebido por uma das entradas e o sinal Comp indica se o sistema gerenciador de eventos recebeu mais de um evento.

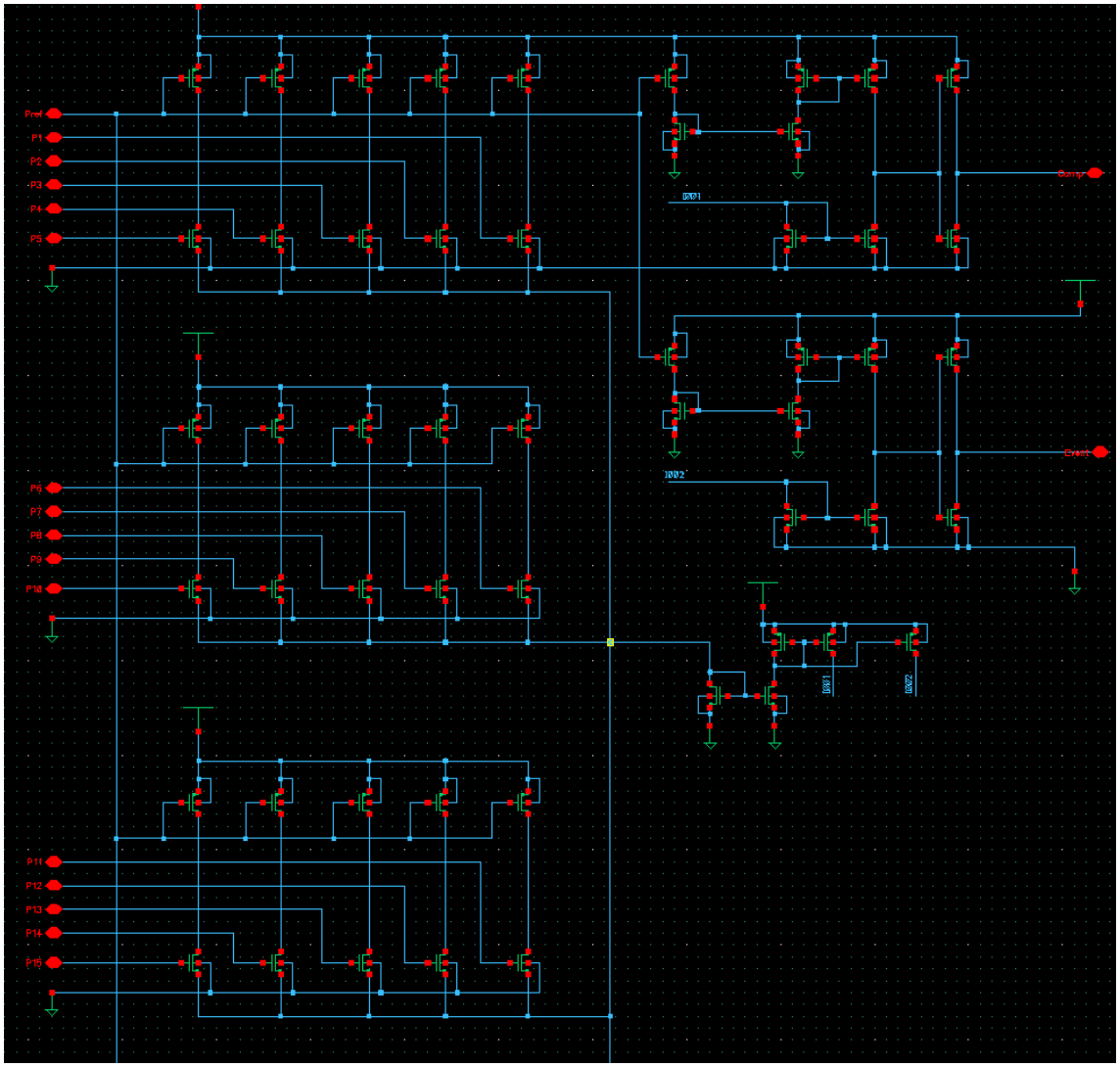


Figura 4.6: Diagrama esquemático do sistema gerenciador de eventos proposto. Na figura é possível identificar as entradas de P1 até P15 e as saídas Event e Comp do sistema gerenciador de eventos.

As entradas do decodificador são os 100 sinais de saída das estruturas D de linhas

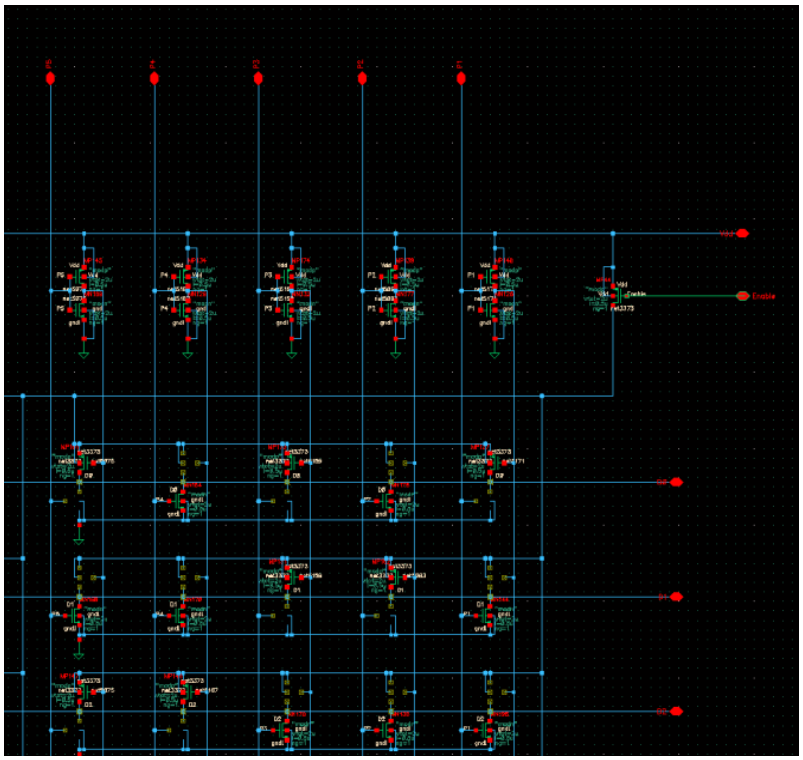


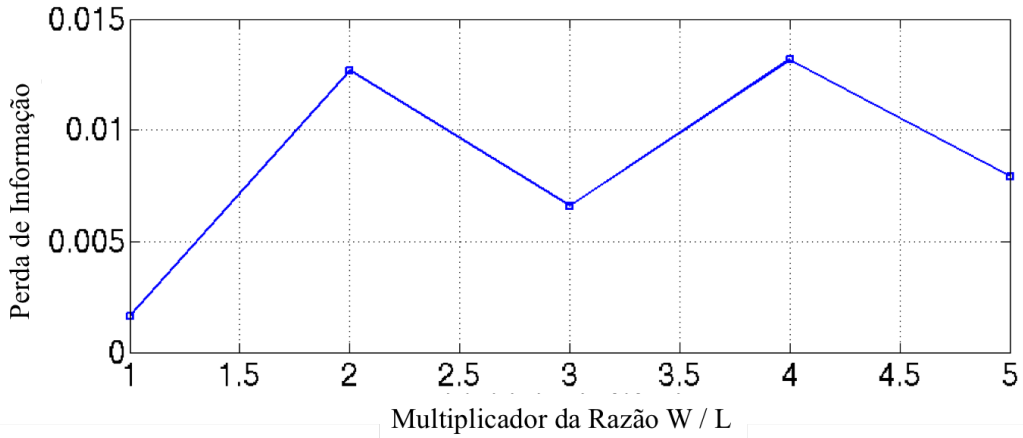
Figura 4.7: A figura mostra a matriz do decodificador. Em destaque estão as entradas de P1 até P5 e as saídas de D0 até D2.

(50 sinais) e colunas (50 sinais) e a saída do decodificador são os $2 \cdot \log_2(50)$ bits (D0 até D5) que compõem o canal de comunicação. Detalhes sobre o funcionamento do codificador podem ser encontrados no Cap. 2 ou em [33]. Todo o sistema de comunicação descrito é repetido para atender a saída do canal *Off*. Não foi implementado um diagrama esquemático para o sistema receptor. Uma operação lógica que ao verificar que o sinal *Req* está ativo, grava em um arquivo de dados os sinais de D0 a D5 do codificador é utilizada. Após gravar esses dados a operação lógica ativa o sinal *Ack*. Quando o sistema de controle de comunicação do receptor verifica que o sinal *Ack* está ativado desativa o sinal *Req* e a comunicação termina.

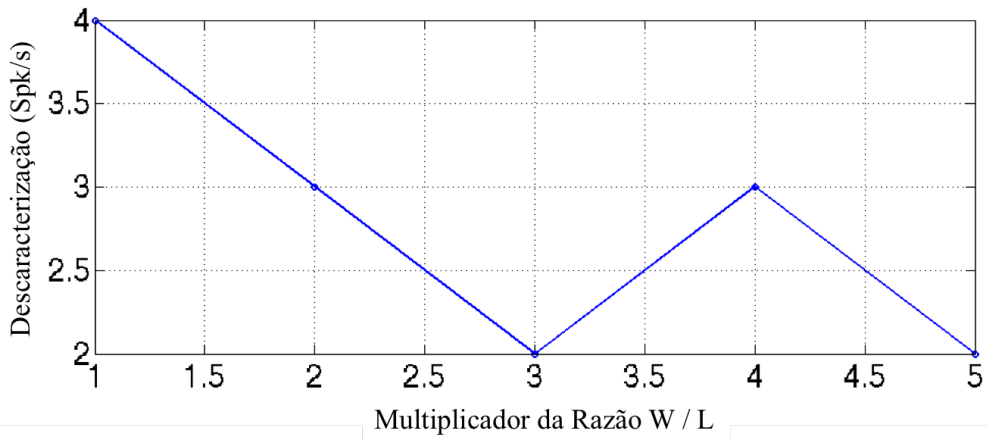
4.1 Dimensões dos Transistores MOS para o AER Proposto

Por se tratarem de sistemas de processamento digital, no projeto das estruturas LC, D e codificador não foi utilizada uma metodologia específica como a que foi utilizada para projetar a OPL a partir de transistores MOS. Foi empregada uma metodologia de projeto utilizando a ferramenta de simulação paramétrica do ambiente de projeto Cadence Virtuoso. Os parâmetros de simulação eram as dimensões W e L dos transistores. A metodologia para encontrar as dimensões dos transistores consistiu

em gerar uma família de curvas de tensão e de corrente das saídas das estruturas LC, D e codificador em função de um conjunto de dimensões W e L. Os valores de W e L no conjunto foram os números inteiros de 1 até 5 (em unidades de μm). O critério de escolha das dimensões W e L consistiu na escolha da menor dimensão que fosse capaz de produzir uma curva de resposta na saída das estruturas que tivesse a transição entre os níveis alto e baixo (3.3 e 0 V respectivamente) com um tempo de resposta igual ou menor do que aquele que foi definido para o ciclo de comunicação do sistema AER proposto, cerca de 5 ns.



(a)



(b)

Figura 4.8: Análise de Monte Carlo para verificação da perda de informação (a) e descaracterização da taxa de disparo dos neurônios (b). Essa análise foi feita sobre o sistema AER proposto.

Outro aspecto importante do dimensionamento foi a questão do descasamento no processo de fabricação. Ou seja, a medida de quanto transistores iguais sofrem variações nas suas dimensões por terem sido posicionados em locais diferentes no chip após o processo de fabricação CMOS. O descasamento no processo de fabricação CMOS pode ajudar a evitar que neurônios estimulados por entradas com níveis

de amplitude de valores iguais produzam taxas de disparo com pulsos sincronizados. Isso possibilitaria (para a técnica proposta) um aumento da probabilidade de que uma colisão fosse evitada. Contudo, níveis excessivos de descasamento podem descaracterizar as taxas de disparo de um neurônio. Ou seja, tornar a taxa de disparo de um neurônio construído em uma determinada posição do chip muito maior ou muito menor do que a de um neurônio em outra posição que é estimulado por um estímulo de mesmo valor. Para verificar um ponto de operação em que a retina CMOS possa se aproveitar dos efeitos causados pelo descasamento sem que as taxas de disparos dos neurônios sejam muito afetadas, foram feitas algumas simulações (5 simulações) alterando as dimensões do W e do L dos transistores da OPL, do retificador e dos neurônios, os mais sensíveis ao descasamento por realizarem processamento analógico. Apesar de variarmos as dimensões W e L as razões W / L desses transistores foram mantidas. Foram feitas 5 alterações nas dimensões W e L dos transistores. São multiplicados simultaneamente os W e os L originais por 1, 2, 3, 4 e 5. Para cada múltiplo de W e L foram feitas 10 simulações elétricas utilizando análise de Monte Carlo. Só foi considerado nessa análise o descasamento do processo. Foram monitorados apenas 5% dos neurônios da matriz, que foram escolhidos de forma aleatória. Essa baixa quantidade de neurônios é devido em parte ao problema de monitorar, armazenar e analisar tamanha quantidade de dados caso optássemos por utilizar todos os neurônios da matriz. 5% foi o máximo que conseguimos utilizar. O valor de corrente na entrada desses neurônios escolhidos foi de 10 pA, nos outros neurônios da matriz foi de 0 pA. A Fig. 4.8 mostra a razão entre a quantidade de eventos que colidiram e a quantidade total de eventos que ocorreram (é chamada de perda de informação essa razão) utilizando o sistema AER proposto. Era esperado que quanto maiores fossem as dimensões de W e L maior seria a perda de informação. Contudo a Fig. 4.8(b) mostra apenas uma tendência que precisa ser confirmada com mais pontos. Devido ao custo computacional envolvido com a simulação elétrica só foi possível gerar essa quantidade de pontos. É chamada de descaracterização da taxa de disparo a diferença entre a maior e a menor taxa de disparo entre os 5% dos neurônios escolhidos no monitoramento. Era esperado que houvesse uma maior descaracterização para as menores dimensões W e L. Contudo os dados que foram adquiridos por simulação elétrica não puderam comprovar esse fato e o que é possível perceber é apenas uma tendência que não é conclusiva. São utilizados os resultados apresentados nas Figs. 4.8 e 4.8(b) para escolher as dimensões W e L dos transistores. Foram escolhidos os valores de W e L que causam a menor perda de informação juntamente com a menor descaracterização da taxa de disparo. Por simplicidade, não é escopo desse trabalho considerar projetos de dispositivos CMOS utilizando o modelo Pelgrom. Contudo, existe a possibilidade que a utilização desse modelo possa melhorar os resultados obtidos aqui.

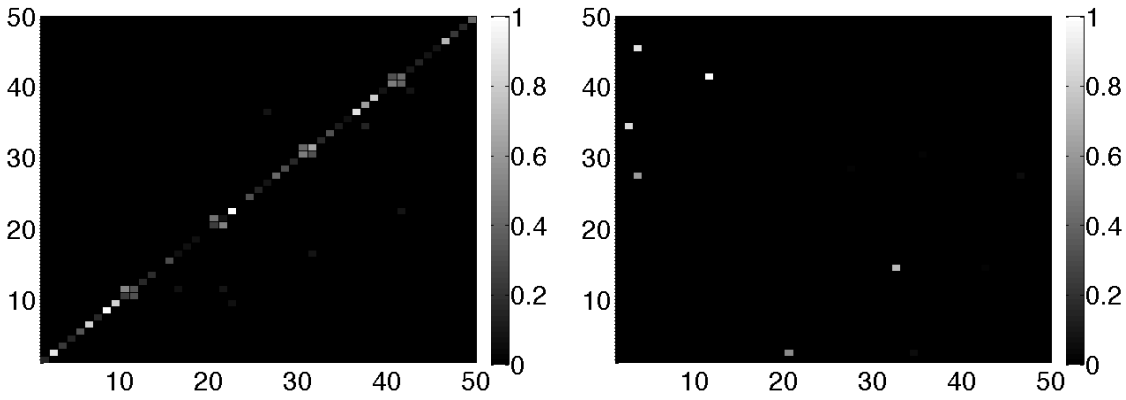


Figura 4.9: A reconstrução de um estímulo em diagonal aplicado ao sistema AER proposto. Como a OPL foi desabilitada toda a informação deveria passar pelo canal *On* (esquerda). No entanto, é possível perceber transmissões feitas pelo canal *Off* (direita). Apesar de algum ruído, a diagonal foi reconstruída a partir da informação de eventos aquisitada no canal *On*.

4.2 Resultados de Reconstrução do AER Proposto

Os valores das fontes de corrente nas entradas do imageador foram ajustados de maneira a formarem uma imagem onde apenas os pixels da diagonal possuíam intensidades diferentes de zero. Os níveis desses pixels da imagem, o valor das fontes de corrente, foram ajustados para 10 pA. Nesse caso os efeitos de filtragem causados pela OPL foram desabilitados. Assim a previsão teórica esperada seria uma imagem contendo a própria diagonal, no canal *On* (esquerda na Fig. 4.9). No canal *Off* (à direita na fig. 4.9) do imageador não deveria aparecer nenhum valor. Uma simulação elétrica foi realizada para esse conjunto e os resultados dessa simulação podem ser vistos na Fig. 4.9.

Tabela 4.1: Caraterísticas da simulação elétrica do sistema AER proposto processando um estímulo em diagonal. Nesse caso a OPL foi desabilitada possibilitando a comunicação apenas pelo canal *On*.

Caraterística	Valor
Eventos On	768.600
Eventos Off	44.300
Taxa de Transferência	51.240.000 Spk/s
Tempo de Simulação	$15 \cdot 10^{-3}$ s
Tempo Real de Simulação	1 semana
Perda de Informação	0.01

A Tab. 4.1 mostra algumas informações sobre a simulação que possibilitou a reconstrução da diagonal processada pelo sistema AER proposto. Essa informação é importante para mostrar a magnitude da complexidade computacional envol-

vida com essas simulações. O computador utilizado possui um processador Intel(R) Core(TM) i7-3770 3.40 GHz de 8 núcleos, memória *cache* de 8 gigabytes e memória RAM de 32 gigabytes.

4.3 Resposta ao Impulso do AER Proposto

Foram projetados o diagrama esquemático de uma OPL CMOS e um sistema AER de transmissão. O diagrama esquemático da OPL é formado de 50 por 50 unidades da OPL. Unidades como aquela apresentada no diagrama esquemático da Fig. 2.6. O diagrama esquemático do sistema AER desenvolvido é equivalente ao do sistema AER NAP que foi proposto no Cap. 3. O teste do sistema AER é mostrado na Fig. 4.9 e o teste da OPL CMOS é mostrado na Fig. 4.10. A resposta biológica da OPL é mostrada na Fig. 4.11, é possível perceber semelhanças entre a resposta ao impulso da OPL CMOS e a resposta da OPL biológica na posição dos vales.

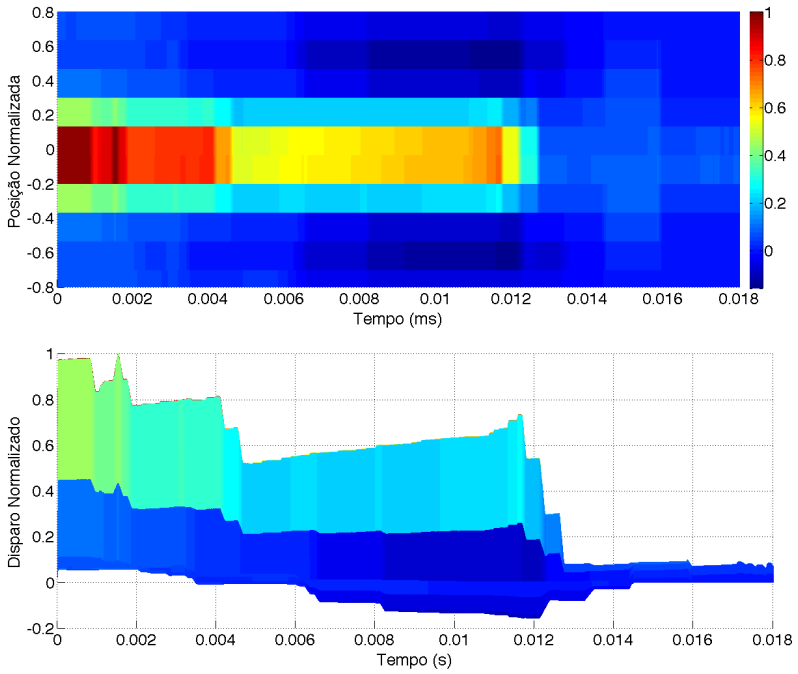


Figura 4.10: O imageador é estimulado por um impulso no espaço e no tempo. A resposta ao impulso do sistema AER proposto é mostrada em duas vistas. É possível perceber as semelhanças entre os perfis da resposta apresentada aqui e a resposta ao impulso de uma retina biológica apresentada por Eggermont [36], Fig. 4.11.

4.4 Estímulos para o Mecanismo Cortical

Durante alguns testes foi percebido que a duração das simulações elétricas se tornava proibitiva para estímulos mais complexos do que impulsos ou diagonais. Embora

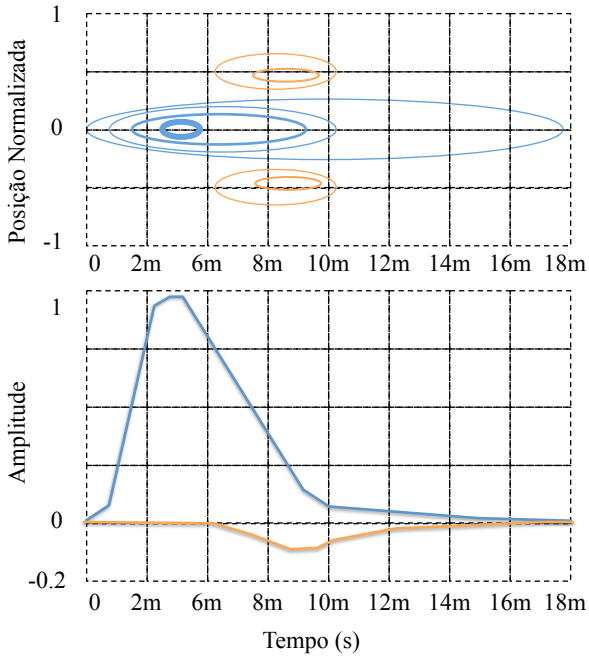


Figura 4.11: A resposta ao impulso de uma retina biológica que foi adaptada de Eggermont [36]. Essa figura é utilizada como uma previsão teórica do resultado esperado da resposta ao impulso do nosso imageador. É possível perceber semelhanças entre a posições dos vales dos gráficos apresentados aqui e os gráficos apresentados na Fig. 4.10.

nenhuma avaliação mais profunda tenha sido realizada suspeitamos que esse fato é devido a nossa capacidade computacional limitada. Por esse motivo foi resolvido adotar uma outra abordagem para gerar as previsões da retina CMOS. Foi utilizada a resposta ao impulso do nosso imageador para gerar os resultados de previsão teórica da resposta da retina CMOS. Nessa abordagem adotada o estímulo complexo (mais complexo que o impulso) foi convoluido com a resposta ao impulso da retina CMOS. Essa previsão é possível somente se considerarmos lineares as operações envolvidas. Assim, foi assumido que as operações envolvidas com essa abordagem eram lineares. Como exemplo desse processamento vamos considerar que o estímulo ao qual se quer prever a resposta do imageador é mostrado na Fig. 4.12 (em cima e à esquerda). Ao convoluirmos esse estímulo com a resposta do imageador, Fig. 4.10, é possível gerar uma previsão teórica, canal *On* (em cima e no meio) e canal *Off* (em cima e à esquerda) na Fig. 4.12. Essa previsão teórica que será utilizada como estímulo para o modelo do circuito cortical do *diffusive filling-in*.

Essa abordagem foi escolhida com o objetivo de criar entradas mais realistas para a simulação dos circuitos corticais do *diffusive filling-in* do que aquela que se costuma adotar ao se utilizar a filtragem com filtros DoG para substituir o processamento feito pela retina. Primeiro, essa abordagem insere amostras no sinal que possuem uma diversidade de valores, Fig. 4.13(a), muito maior do que a abordagem

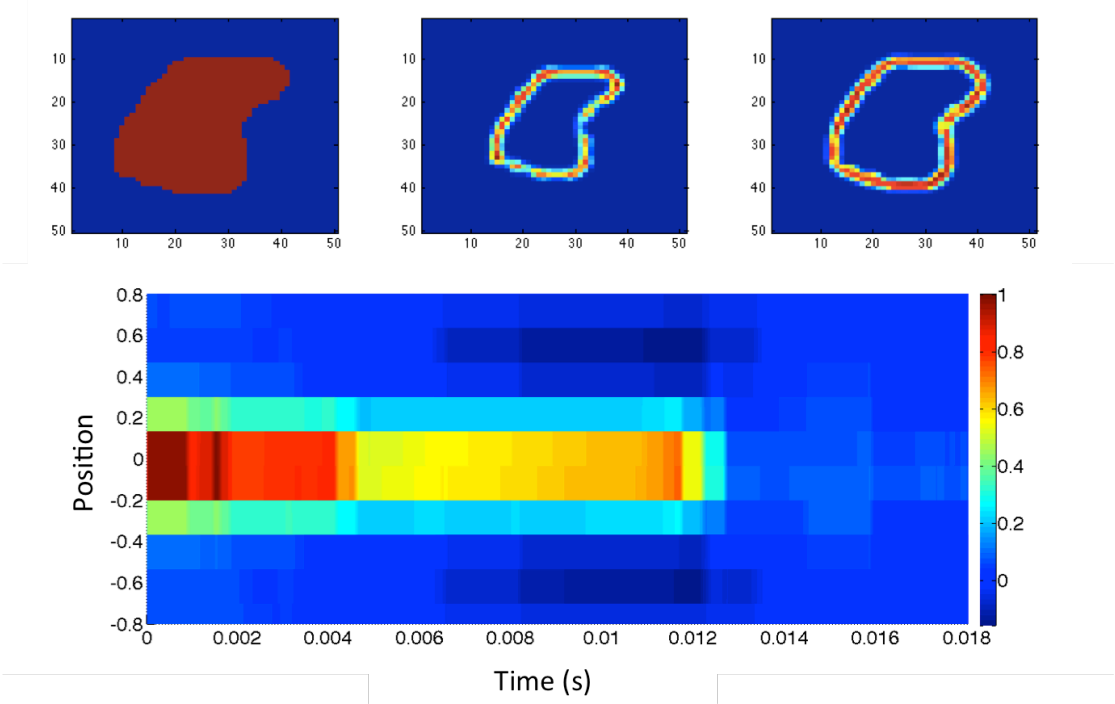
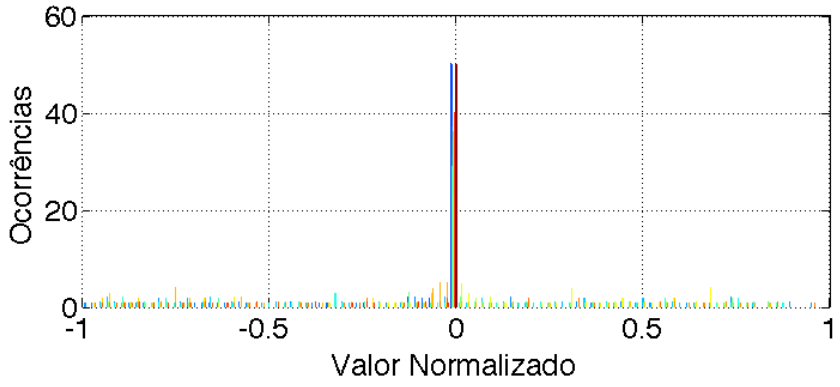


Figura 4.12: A previsão teórica da resposta (superior, esquerda e meio), da simulação elétrica, do processamento do imageador proposto sobre um dado estímulo, coluna esquerda na linha superior. Na linha de baixo é mostrada a resposta ao impulso do imageador.

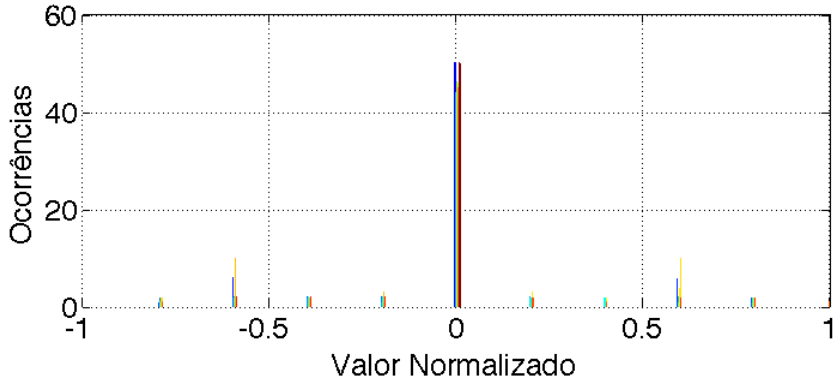
que utiliza filtragem DOG, Fig. 4.13(b). Essa maior diversidade permite maiores variações entre as sequências de pulsos na implementação pulsada. Como a implementação em modo de amplitude não é capaz de abranger tais efeitos é esperado que essa abordagem enfatize as diferenças entre os resultados obtidos pelos dois modos de implementação. Segundo, como é possível notar existe uma diferença entre os resultados apresentados nas Figs. 2.14 e 4.11. Essa diferença é em parte causada pelas distorções que envolvem os sistemas de comunicação AER, como a latência e a perda de informação. Dessa maneira, em nossas simulações vamos considerar as distorções causadas pelo sistema AER nas implementações. Como foi dito anteriormente, existe um compromisso entre essas perdas e a redução da complexidade de *hardware* do circuito neural ao qual se quer implementar.

4.5 Conclusões

O diagrama esquemático de um imageador que codifica uma imagem aplicada a sua entrada em um código pulsado foi desenvolvido. Esse imageador é composto por um estágio de processamento OPL, um estágio de processamento IPL (apenas o retificador), duas camadas de células ganglionares (uma para codificar os sinais *On* e outra para os *Off*) e um sistema de comunicação com topologia SA. Detalhes desses



(a)



(b)

Figura 4.13: A diversidade no valor das amostras causada nos sinais de estímulos do DFI após o processamento realizado pelo imageador proposto, um modelo elétrico da retina (a). Se esse processamento for substituído por aquele realizado por um filtro DOG (b), a diversidade não será tão grande e assim, nesse caso, os estímulos produzidos serão menos realistas.

diagramas esquemáticos são apresentados nesse capítulo. Resultados de simulações elétricas apresentados na Fig. 4.8 (parte a) mostram que existe um compromisso entre a perda de informação e a variação do descasamento no processo de fabricação, para uma mesma razão $\frac{W}{L}$ quanto menor as dimensões W e L menor é a perda de informação. Devido ao descasamento no processo de fabricação, uma imagem introduzida na entrada do imageador, pixels que possuem o mesmo valor apresentaram taxas de disparo diferentes na matriz de neurônios. Isso reduz a probabilidade de ocorrência de colisões. Como pixels de valores iguais causam saídas com taxa de disparo diferentes, é preciso saber quão diferentes essas saídas podem se tornar. A Fig. 4.8 (parte a) apresenta resultados de simulação que mostra essa diferença da taxa de disparo em relação ao comprimento de W e L. Esses dados auxiliam no ajuste final do dimensionamento do imageador por possibilitar redução na quantidade de perda por colisão e uma descaracterização máxima da taxa de disparo dos neurônios controlável. A Fig. 4.9 apresenta resultados de reconstrução do imageador. Uma

imagem contendo uma diagonal é inserida na entrada do imageador, o imageador responde a esse estímulos com uma sequência de pulsos (endereços) e ao utilizar um procedimento de contabilizar durante 15 ms os dados gerados pelo imageador em um histograma em duas dimensões, onde os eixos são as posições de linha e de coluna, foi possível reconstruir a diagonal apresentada na entrada. O resultado mostra que o imageador codifica corretamente os estímulos apresentados. Um impulso no espaço e no tempo foi apresentado ao imageador e como resultado a resposta do imageador a esse impulso é apresentada na Fig. 4.10. Ao comparar esse resultado com a resposta da OPL ao impulso apresentada na Fig. 4.11, apesar de algum ruído, pode-se concluir que as respostas coincidem. A Fig. 4.12 apresenta a resposta da filtragem de um estímulo utilizando um filtro que é a resposta ao impulso do imageador. É possível perceber que essa filtragem gera um conjunto de contornos fechados, resposta essa que coincide com a resposta esperada para a filtragem utilizando filtros DoG, exceto pela diversidade de valores que aparecem. Essa diversidade é destacada na Fig. 4.13, onde dois histogramas mostram as diferenças entre as diversidades produzidas por uma filtragem com um filtro DoG e a filtragem utilizando a resposta ao impulso do imageador. Em parte, essa diversidade é produzida pelas características de filtragem da OPL e em parte pelas características do sistema de comunicação que utiliza topologia SA. Esses estímulos tratados com a resposta ao impulso do imageador, por envolverem uma maior diversidade de valores, produzirão condições mais adversas para as simulações do DFI.

Capítulo 5

Circuitos Corticais Pulsados

Embora a questão do processamento de informação utilizando sequências de pulsos continue sendo um problema em aberto [37], [31] e as soluções propostas até então não resolvam por completo esse problema [38], [1] e [39], uma intensa discussão sobre topologias para a implementação em *hardware* de redes neurais pulsadas, como aquelas encontradas nos circuitos corticais biológicos, vem ganhando força no cenário acadêmico atual [3], [9], [5] e [4]. Nesse capítulo será discutido sobre os fundamentos da implementação, a nível de sistema, de circuitos corticais (redes neurais pulsadas) em dois modos, pulsado e em amplitude. Será utilizado como exemplo os circuitos corticais que compõem o mecanismo biológico do *diffusive filling-in*. Esse estudo é importante porque permitirá comparar os dois modos de implementação (modo pulsado e o modo de amplitude) e avaliar a possibilidade de utilização do modo de amplitude na previsão dos resultados que seriam obtidos pela implementação em modo pulsado [40]. O dimensionamento do *hardware* que implementa o circuito cortical a ser estudado utilizando resultados de previsão teóricos feitos a partir de simulações em modo pulsado é caro computacionalmente. O que se objetiva é descobrir se é possível utilizar o modo de amplitude para gerar esses resultados de previsão teórica que possibilitem o dimensionamento do *hardware* que implementa o circuito cortical. Na biologia, o mecanismo do DFI é utilizado pelo cérebro para reconstruir a informação de baixa frequência espacial perdida na filtragem passa-banda espacial da retina. O mecanismo do DFI repara também a informação perdida por causa de anomalias da retina como o ponto cego, uma área na retina onde não existem fotorreceptores, e a obstrução da informação luminosa causada por alguns vasos sanguíneos posicionados na frente de fotorreceptores.

5.1 O Neurônio, a Representação no Tempo e o Processamento Pulsado

Muitos modelos de neurônios biológicos têm sido propostos na literatura, no Cap. 2 foi dado o exemplo do modelo IF. Contudo esse modelo não é avaliado para implementações de circuitos corticais que envolvam laços de realimentação positiva, como é o caso do *diffusive filling-in*. O modelo IF é instável nesse tipo de configuração. Um modelo que é avaliado para situações onde implementações de circuitos corticais necessitam de laços de realimentação é o IF combinado com um período refratário. O período refratário é um intervalo de tempo, iniciado após o neurônio ter disparado, onde outros disparos não podem acontecer [3], [9]. Nesse trabalho será utilizado o modelo *leaky integrate-and-fire* (LIF) [22], ao qual corresponde ao modelo IF operando em conjunto com um artifício de adaptação por realimentação negativa. Esse artifício causa um efeito semelhante àquele que é obtido ao se utilizar o IF com o período refratário. O LIF é semelhante ao IF exceto pelo fato de que a variável de estado $v(t)$ possui um decaimento no tempo. Caso o LIF não receba estímulos o valor da variável $v(t)$ é reduzido até atingir um valor mínimo v_L. Em [22] é possível encontrar um modelo discreto do LIF com adaptação, Eq. (5.1).

$$\begin{cases} v(n+1) = A \cdot v(n) + B \cdot v_i(n) - C \cdot v_a(n), \\ v_a(n+1) = D \cdot v_o(n) - E \cdot v_a(n), \\ if\ v(n) > v_{th},\ then\ v_o(n) \leftarrow v_H,\ and\ v(n) \leftarrow v_L, \end{cases} \tag{5.1}$$

Na Eq. (5.1), os parâmetros A, B, C, D e E possuem valores positivos e devem ser ajustadas de acordo com o conjunto de dados que se quer utilizar. Nesse trabalho são utilizados $A = 0.9$, $B = 1$, $C = 0.1$, $D = 1$, e $E = 0.2$. Na equação existem duas variáveis de estado discretas ($v(n)$ e $v_a(n)$), onde $v_a(n)$ é a variável de estado que causará o efeito adaptativo (equivalente ao período refratário). No modelo as entradas são $v_i(n)$, $i = 1, \ldots, N$, e as saídas são $v_o(n)$. Um exemplo de utilização do modelo, Eq. (5.1), é mostrado na Fig. 5.1, para $v_{th} = 1$, $v_H = 1$, e $v_L = 0$). O modelo é utilizado para ajustar um conjunto de dados retirados de experimentos fisiológicos [41]. A linha pontilhada mostra a saída do neurônio biológico (na camada V2 do córtex), como uma função de uma taxa de disparos normalizada. Na figura, 0.5 corresponde a uma taxa de disparos de 120 Spk/s de uma imagem para o qual o sistema visual de um primata foi exposto. Na figura, é apresentado também o ajuste feito a partir de um modelo IF. Claramente é possível notar a diferença de comportamento entre o modelo IF e o LIF. A característica biológica de limitação na taxa de disparos na saída só é alcançada pelo modelo LIF.

Por causa do alto custo computacional necessário para realizar simulações de

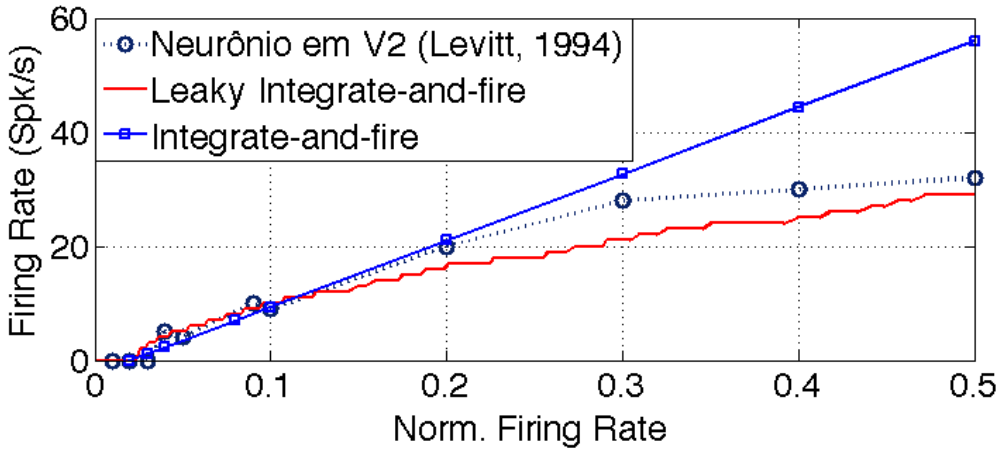
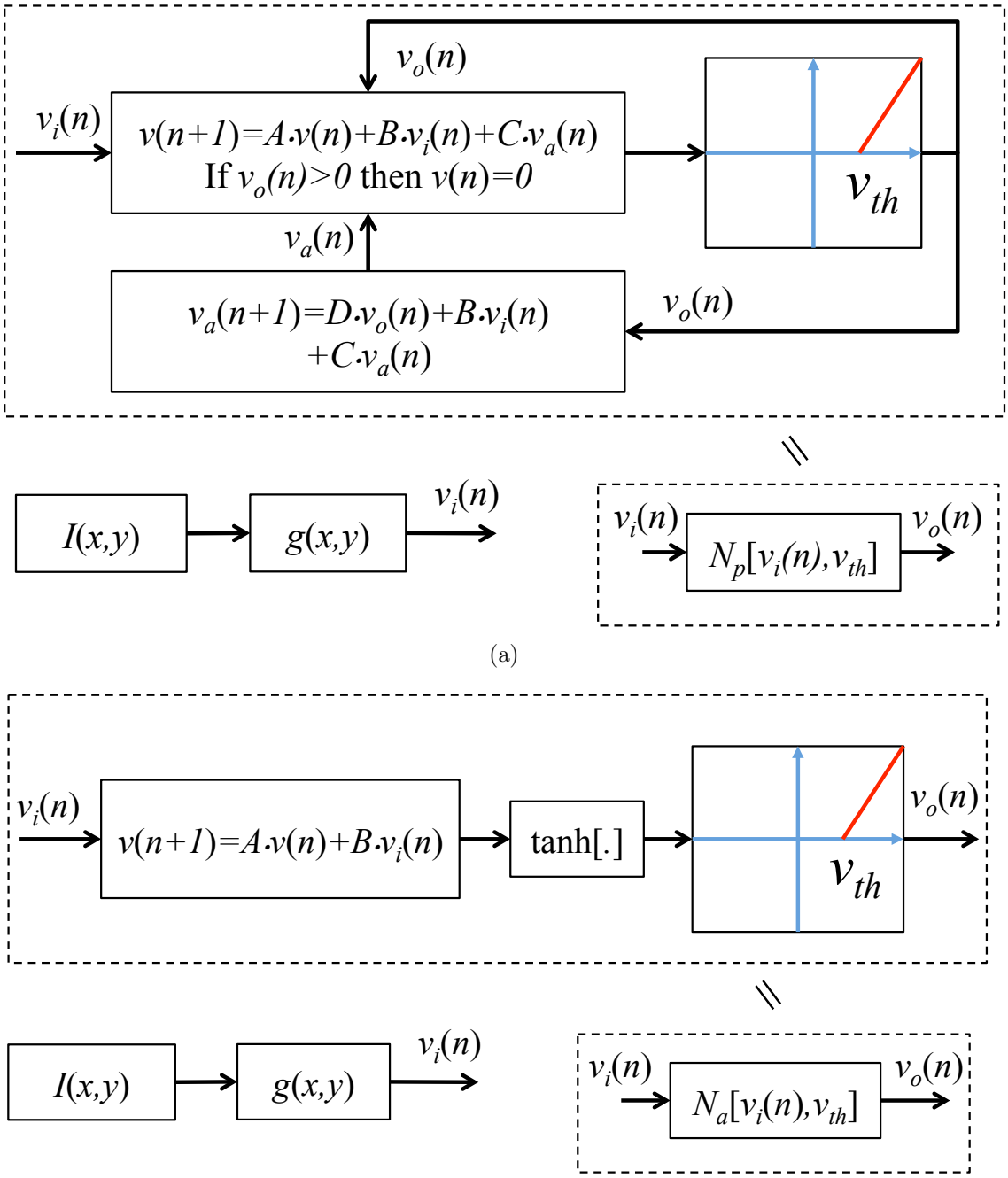


Figura 5.1: Taxa de disparo na saída de um neurônio biológico (linha pontilhada) como função de uma taxa de entrada que corresponde a um contraste normalizado [41], e a aproximação da resposta de dois modelos para essa curva, LIF (linha sólida) e IF (linha com marcadores quadrados).

circuitos corticais compostos de uma quantidade enorme de neurônios operando em modo pulsado, como o modelo apresentado na Eq. (5.1), o modelo do neurônio pulsado é substituído por um modelo simplificado, chamado de modo de amplitude. Nesse tipo de modelo uma sequência de escalares representa a taxa de disparo na saída e nas entradas de um neurônio. Embora modelos de neurônio em modo de amplitude não tenham de fato *spikes* em suas entradas e saídas, na literatura esse modo é utilizado para aproximar os resultados de previsão do modo pulsado [14]. A taxa de disparos de um neurônio pode ser definida como sendo a soma dos *spikes* gerados por um neurônio, dentro de uma janela de tempo fixa T. Não utilizar *spikes* em uma simulação reduz o custo computacional e pode determinar a viabilidade ou não de uma simulação [9]. A Eq. (5.2) mostra o modelo LIF, Eq. (5.1), descrito em modo de amplitude.

$$\begin{cases} v(n+1) = A \cdot v(n) + B \cdot v_i(n), \\ v_o(n) = [\tanh(v(n) - v_{th})]^+, \end{cases} \tag{5.2}$$

No córtex visual, circuitos biológicos processam informação por meio de operações individuais que são realizadas pelos neurônios e de acordo com as conexões estabelecidas entre eles. Em simulações em modo pulsado, as operações individuais dos neurônios são implementadas pela Eq. (5.1). Cada conexão corresponde a um peso sináptico colocado entre a saída de um neurônio e a entrada de outro. A implementação em modo pulsado possui sua estrutura mostrada na Fig. 5.2(a), e a em modo de amplitude mostrada na Fig. 5.2(b). Em ambos os diagramas de blocos, $I(x, y)$ é uma imagem de entrada, v_{th} representa uma operação de limiar, e $g(x, y)$ é um conjunto de pesos sinápticos. Na implementação em modo pulsado o valor de



(a)

(b)

Figura 5.2: Implementações do modo pulsado (a) e do modo de amplitude (b) de um exemplo computacional de um detector de bordas. A imagem de entrada $v_i(n)$ é obtida a partir da convolução, no domínio espacial, entre uma imagem de entrada $I(x, y)$ e um filtro passa alta $g(x, y)$ (nesse trabalho os filtros serão chamados de pesos sinápticos). No modo pulsado (a), A imagem de entrada $v_i(n)$ é utilizada como entrada de uma equação diferencial de acordo com o modelo do LIF in Eq. (5.1). No modo de amplitude (b), a equação diferencial usada corresponde ao modelo descrito pela Eq. (5.2).

v_{th} é 1 e na implementação em modo de amplitude é 0.1. O peso sináptico tem valor descrito por $g(x, y) = [-1\ \ -1\ \ -1;\ -1\ 8\ \ -1;\ -1\ \ -1\ \ -1]$. Os pequenos blocos

de entrada $v_i(n)$ e saída $v_o(n)$ na parte de baixo, na direita, da Figs. 5.2(a) e 5.2(b) são chamados de *estágios de ativação*, no texto serão representados esses estágios por $v_o(n) = N[v_i(n), v_{th}]$. De acordo com o modo, Figs. 5.2(a) [9] e 5.2(b) [10], é possível ter $v_o(n) = N_p[v_i(n), v_{th}]$ e $v_o(n) = N_a[v_i(n), v_{th}]$, respectivamente.

5.2 Modelo do Diffusive Filling-In

O *diffusive filling-in* (DFI) é uma operação de processamento de sinais que é executada pelo córtex visual com o objetivo de preencher regiões definidas por contornos fechados. O diagrama de blocos do DFI é mostrado na Fig. 5.3 [9]. As entradas para a operação do DFI são imagens estáticas v_{On} e v_{Off}, que são transmitidas da retina até o córtex visual, e a saída é a imagem v_{DFI}. Na imagem v_{DFI} as regiões definidas por um contorno fechado são preenchidas.

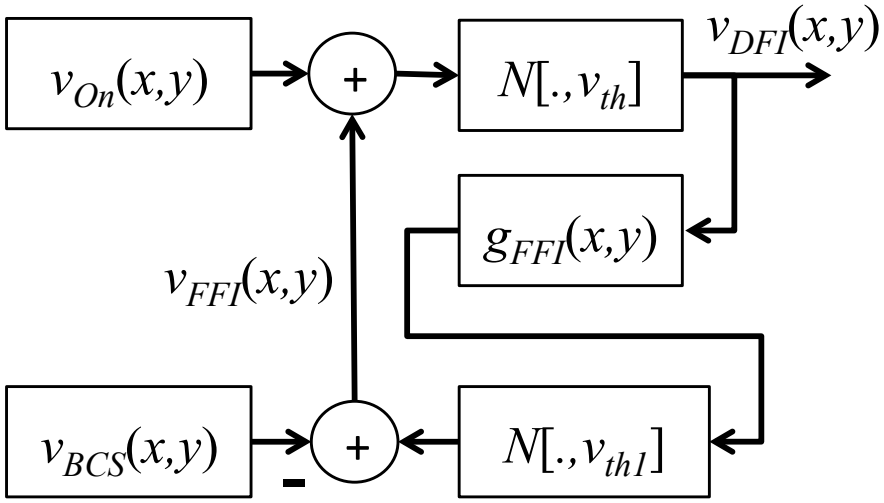


Figura 5.3: O diagrama de blocos do algoritmo do DFI (em modo pulsado ou em modo de amplitude). Para gerar a imagem $v_{DFI}(x, y)$ a partir das imagens $v_{On}(x, y)$, $v_{FFI}(x, y)$ e do nível de limiar v_{th}, é utilizada a função de ativação $N[.]$ que foi indicada nas Figs. 5.2(a) e 5.2(b). A convolução com o peso $g_{FFI}(x, y)$ permite o espalhamento da atividade neural a partir de um pixel para os seus vizinhos. Um laço de realimentação é implementado por meio de um neurônio (o modelo). Esse laço permite que o espalhamento continue até tomar a imagem inteira. Contudo uma operação de subtração com a imagem de saída do estágio de processamento *boundary contour system* (BCS, mais detalhes a seguir) $v_{BCS}(x, y)$ faz com que o espalhamento pare. Em modo de amplitude temos que $v_{th} = v_{th1} = 0.01$. Em modo pulsado $v_{th} = v_{th1} = 1$.

Por causa da filtragem espacial a nível de retina, Eq. (2.5), as imagens v_{On} e v_{Off} são compostas por contornos. Esses contornos podem ser ou não completamente fechados. O algoritmo DFI usa a informação dessas duas imagens (v_{On} e

v_{Off}) para estimar quais as regiões corretas da imagem $v_{DFI}(x, y)$ que devem ser preenchidas. No sistema visual precoce, a fronteira entre duas regiões fechadas é representada por dois contornos que circulam ambas as regiões. O contorno mais próximo à região fechada onde os pixels da imagem de entrada tinham maior valor é composto por pixels de valor positivo, enquanto no outro contorno os pixels possuem valor negativo. Cada contorno é transmitido ao córtex visual por meio de canais exclusivos, como foi dito nas explicações da OPL no Cap. 2. Um contorno pode ser utilizado para determinar os limites de preenchimento do outro contorno. Nesse trabalho vamos considerar sempre que a imagem v_{On} é que vai servir de base para o preenchimento e a imagem v_{Off} que vai definir os limites do preenchimento de v_{On}. Por causa da estrutura da retina biológica, nem todas as amostras das imagens v_{On} e v_{Off} são transmitidas ao córtex. As informações dos pixels são perdidas em relação a imagem original e nessas posições o valor dos pixels é substituído por zero. Por causa dessa perda de informação, os contornos em v_{On} e v_{Off} se tornam vazados. Assim v_{Off} não é capaz de determinar os limites de espalhamento de v_{On}. Dessa maneira as amostras perdidas em v_{Off} devem ser recuperadas. O estágio *boundary contour system*, o primeiro estágio de um total de dois estágios de processamento do algoritmo do DFI, processa a imagem v_{Off} a fim de reparar essas amostras perdidas. Nesse primeiro estágio de processamento (*boundary contour system*), a imagem v_{Off} é processada e os valores dos pixels que compõem os contornos são recuperados. Para alcançar esse resultado, a atividade dos neurônios é propagada através dos pixels vizinhos de acordo com propriedades, tais como direção e colinearidade, ao qual são comuns aos pixels que formam os contornos. A imagem v_{BCS} é o resultado da operação de processamento de sinais sobre a imagem de entrada v_{Off} que ocorre nesse estágio. No segundo estágio de processamento (*filling-in*), regiões definidas pelos contornos da imagem v_{On} são preenchidas pelo espalhamento da atividade neural entre pixels vizinhos. Para a imagem v_{On} é esperado que apenas os pixels localizados no contorno tenham seus valores diferentes de zero, assim essas são as posições iniciais de onde o espalhamento começa. Na imagem v_{FFI} na Fig. 5.3, a atividade neural é espalhada entre os pixels vizinhos ao longo de todas as direções onde os vizinhos estejam conectados. Para previnir que a atividade neural se espalhe indefinidamente, a imagem v_{Off} é utilizada. A expressão *segundo estágio*, *filling-in*, se refere ao fato de que a camada cortical em questão é, na estrutura do córtex visual, hierarquicamente superior à camada do *boundary contour system*. Ambas as camadas realizam o processamento de maneira simultânea. A partir dos dois estágios de processamento (*boundary contour system* e *filling-in*) o sistema visual reconstrói a informação de baixa frequência espacial original que havia sido removida pelos defeitos naturais e filtragem espacial envolvida com o processamento do EVS [15], [14]. Sem essa informação a percepção de superfícies sólidas seria comprometida.

Mais detalhes sobre esses dois estágios serão dados a seguir.

5.2.1 Detalhes sobre o DFI

A imagem $v_{DFI}(x, y)$ é obtida a partir da operação $v_{DFI}(x, y) = N[v_{On}(x, y) + v_{FFI}(x, y), v_{th1}]$, como é mostrado na Fig. 5.3. O valor inicial de $v_{FFI}(x, y)$ é zero, assim inicialmente $v_{DFI}(x, y)$ contém aproximadamente os mesmos contornos da imagem $v_{On}(x, y)$. Após isso, uma convolução (no domínio do espaço) entre $v_{DFI}(x, y)$ e $g_{FFI}(x, y)$ é realizada. Será utilizado o peso sináptico $g_{FFI} = [0\ 1\ 0;\ 1\ 0\ 1;\ 0\ 1\ 0]$. Essa convolução copia os valores dos pixels da imagem $v_{DFI}(x, y)$ para os seus quatro vizinhos (em cima, em baixo, a direita e a esquerda). A essa operação, será denominada de espalhamento da informação de contorno entre pixels vizinhos. Para representar o resultado da convolução em termos da atividade neural (ou em modo pulsado ou em modo de amplitude), a operação $N[g_{FFI}(x, y) * v_{DFI}(x, y), v_{th1}]$ é realizada, operação apresentada nas Figs. 5.2(a) e 5.2(b).

O laço de realimentação positiva permite que os pixels ativos em $v_{On}(x, y) + N[v_{DFI}(x, y) * g_{FFI}(x, y), v_{th1}] - v_{BCS}(x, y)$ reforcem os valores dos pixels em $v_{DFI}(x, y)$, desde que as atividades desses pixels sejam suficientemente intensas (mais intensas que v_{th}) para não serem anuladas pela operação de limiar. Após essa operação de realimentação a imagem $v_{DFI}(x, y)$ contém seus próprios contornos originais bem como os contornos formados pelos pixels vizinhos aos contornos originais. Cada vez que o algoritmo prossegue com o processamento mais largos os contornos em $v_{DFI}(x, y)$ se tornam. Se esse processo de preenchimento não for interrompido, então toda imagem será preenchida. A imagem v_{Off} é utilizada para prevenir que a atividade se espalhe além das regiões definidas pelos contornos em v_{Off}. Para interromper o processo de preenchimento, a imagem v_{Off} pode substituir a imagem v_{BCS} e ser subtraída, pixel-a-pixel, da imagem que será utilizada para gerar a imagem v_{FFI}. Tal subtração objetiva gerar valores negativos (ou nulos) nos pixels correspondentes as posições dos contornos da imagem v_{Off} na imagem v_{FFI} e assim inibir que a atividade neural se espalhe além dos contornos definidos pela imagem v_{Off}. A estrutura biológica do sistema visual precoce apresenta anomalias naturais em seus estágios de processamento. Essas anomalias são criadas pelo ponto cego e por vasos sanguíneos que cobrem os fotorreceptores, impedindo que esses recebam a informação luminosa [42]. Por causa dessas anomalias algumas posições na imagem v_{Off} correspondem a leituras defeituosas da informação. Assim a imagem v_{Off} pode conter pixels inativos em posições onde alguma atividade deveria ser detectada. Os contornos definidos por v_{Off} podem, nesse caso estarem abertos, mas precisam ser fechados para que possibilitem que o bloqueio do espalhamento da atividade seja realizado. Nós assumimos que a imagem v_{Off} sempre

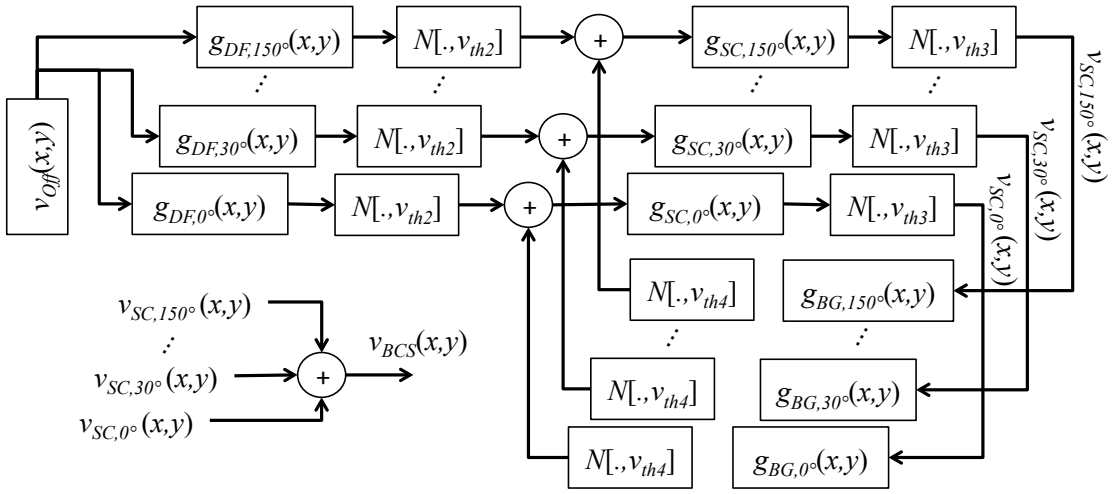


Figura 5.4: Diagrama de blocos do BCS. A imagem de entrada é $v_{Off}(x, y)$ e a imagem de saída é $v_{BCS}(x, y)$. Todas as operações são realizadas simultaneamente ao longo dos canais orientados em 0^o, 30^o, 60^o, 90^o, 120^o, e 150^o. A convolução com o filtro DF (direcional) seguido por uma operação de limiar com o nível v_{th2} tende a manter os segmentos do contorno, em cada canal orientado, ao longo de uma direção em particular. A convolução com o filtro SC (competição espacial) seguido da operação de limiar com o nível v_{th3} remove segmentos fracos (residuais) que permaneceram do estágio de filtragem anterior, reforçando os segmentos ao longo da correta orientação. A convolução com o filtro BG (agrupamento por bipolo) permite conectar segmentos de contorno que devem ser unidos no mesmo contorno. Esses segmentos possivelmente foram separados por causa das leituras defeituosas feitas pelo sistema visual precoce. Os resultados da operação BG reforçam os resultados da operação SC e vice-versa. A saída do BCS é obtida ao serem somadas as saídas de todos os canais SC.

contém pelo menos um contorno fechado que foi originalmente apresentado em uma imagem que foi capturada pela retina. Mesmo que esse contorno tenha defeitos. a operação de completamento realizada pelo *boundary completion system* é necessária para manter os contornos completamente fechados. O BCS (boundary contour system) é detalhado na Fig. 5.4. A operação do BCS é realizada sobre a imagem v_{Off} e gera, a partir dela, uma nova versão de v_{Off} (na qual os contornos se encontram completamente fechados), a imagem de saída v_{BCS}. A fim de recuperar amostras defeituosas em v_{Off}, a operação BCS usa três outras operações: filtragem direcional, competição espacial e agrupamento por bipolo.

Filtragem direcional: Nesse estágio, a imagem v_{Off} é filtrada por um banco de filtros de Gabor [43]. Cada filtro do banco $g_{DF,\theta}(x, y)$, DF - filtragem direcional (Directional Filtering), corresponde a uma direção específica θ_i, $i = 1, 2, \ldots, K$, onde K é um número inteiro de direções discretas. Nesse trabalho temos que $K = 6$, múltiplos de 30^o, de 0^o até 150^o. Todos os filtros de Gabor estão na mesma escala. Essa operação gera uma imagem para cada filtro direcional. Em todas as imagens,

os objetos ou contornos fechados terão suas bordas enfatizadas ao longo da direção associada ao filtro de Gabor que gerou a imagem. Pixels que tiverem seus valores abaixo do limiar (v_{th2}) serão ajustados para zero. Em modo pulsado temos que $v_{th2} = 1$ e em modo de amplitude $v_{th2} = 0.01$. Essa operação ajuda a eliminar segmentos residuais do contorno que não tenham a mesma direção associada ao filtro de Gabor, contudo, alguns segmentos residuais permanecem nas imagens. O procedimento de filtragem direcional é definido pela Eq. (5.3).

$$v_{FD,\theta}(x,y) = \begin{cases} v_{Off}(x,y) * g_{\theta}(x,y), \\ \quad \text{if } v_{Off}(x,y) * g_{DF,\theta}(x,y) > v_{th2} \\ 0, \text{ otherwise.} \end{cases} \tag{5.3}$$

Competição espacial: Para eliminar a presença de ainda mais contornos residuais nas imagens, as saídas do estágio de filtragem direcional são convoluidas, no domínio espacial, com o filtro $g_{SC} = [-1\ -1\ -1;\ -1\ +8\ -1;\ -1\ -1\ -1]$, o filtro de competição espacial (SC - *Spatial Competition*), e todo pixel das imagens resultantes da convolução são comparados ao valor de limiar v_{th3}. Caso os valores dos pixels sejam menores do que v_{th3}, então seus valores são ajustados para zero. De outra forma, o resultado da convolução é mantido. Para descrever uma correspondência biológica com essa operação [15], nós associamos essa operação com uma competição entre o pixel central (oito vezes) e seus vizinhos (uma vez). Se o pixel central vencer a competição, então seu valor é mantido. De outra maneira, o seu valor é ajustado para zero. A competição espacial permite assim uma simples implementação do bem conhecido detector de bordas, ao qual é composto de uma operação de filtragem com um filtro de diferença de gaussianas seguida por uma operação de limiar. $v_{th3} = 1$ no caso do modo pulsado ou $v_{th3} = 0.01$ no caso do modo de amplitude. Como resultado da competição espacial, os contornos mais fracos são eliminados e os contornos mais fortes são enfatizados. A operação da competição espacial é definida pela Eq. (5.4), ao qual também leva em consideração valores não nulos de realimentação do conjunto de imagens $v_{BG,\theta}(x,y)$ a partir da operação subsequente denotada de BG (*Bipole Grouping*). A operação de agrupamento por bipolo irá ser definida logo a seguir, mas a Eq. (5.4) pode ser entendida nesse ponto como tendo $v_{BG,\theta}(x,y) = 0$ para todo o procedimento descrito para a operação SC.

$$v_{SC,\theta}(x,y) = \begin{cases} (v_{FD,\theta}(x,y) + v_{BG,\theta}(x,y)) * \\ g_{SC}(x,y), \\ \quad \text{if } (v_{FD,\theta}(x,y) + v_{BG,\theta}(x,y)) * \\ \quad g_{SC}(x,y) > v_{th3}, \\ 0, \text{ otherwise.} \end{cases} \tag{5.4}$$

Agrupamento por Bipolo: De maneira similar ao que acontece na operação DF e SC, a operação de agrupamento de bipolo (BP) também processa imagens de acordo com a orientação (direção dominante). Uma imagem *horizontal* (associada com um filtro de Gabor horizontal) é filtrada ao longo de uma direção horizontal, uma imagem em 30^o é filtrada por um filtro de Gabor de 30^o, e assim todas as outras direções. Será focado em uma direção horizontal para a nossa explicação sendo as explicações para as outras direções similares a essa. Inicialmente, a imagem horizontal é deslocada para a direita e para a esquerda. Como o objeto na imagem horizontal corresponde a uma borda horizontal, essa operação tende a unir segmentos horizontais aos quais foram separados por leituras defeituosas no estágio de processamento do sistema visual precoce. Assim é possível recuperar esses pixels que tiveram seus valores apagados por conta dessas leituras defeituosas. Deslocar a imagem horizontal para a direita e depois para a esquerda e somar os dois resultados é equivalente a convoluir a imagem horizontal com o filtro [0 0 0; 1 0 1; 0 0 0]. Pelo fato dessas leituras defeituosas ocuparem mais do que um pixel é necessário que o deslocamento para a direita e para a esquerda seja maior do que o de um pixel. A operação que equivale deslocar a imagem horizontal 3 pixels para a direita e 3 para a esquerda (e somar as duas imagens) é a convolução da imagem horizontal com o filtro [0 0 0 0 0; 0 0 0 0 0; 1 0 0 0 1; 0 0 0 0 0; 0 0 0 0 0] e assim por diante. Uma operação de limiar restringe a saída desse estágio apenas aos pixels que responderem mais intensamente a operação. Nesse estágio é utilizado o nível v_{th4} como referência. Para implementações em modo pulsado é utilizado $v_{th4} = 2$ e em modo de amplitude $v_{th4} = 0.015$. Tipicamente, os pixels localizados entre dois segmentos horizontais tem maior magnitude de resposta do que aqueles que estão a direita ou a esquerda de um único segmento. O procedimento de agrupamento por bipolo é definido pela Eq. (5.5).

$$v_{BG,\theta}(x,y) = \begin{cases} v_{SC,theta}(x,y) * g_{SC}(x,y), \\ \quad \text{if } v_{SC,theta}(x,y) * g_{SC,\theta}(x,y) > v_{th4}, \\ 0, \text{ otherwise.} \end{cases} \tag{5.5}$$

Esses deslocamentos para a direita e para a esquerda possuem correspondência biológica: No sistema visual, estruturas específicas (circuitos neurais realizam conexões de longa distância [44]) conhecidas como bipolos [45], [46] implementam esses deslocamentos. Bipolos são circuitos neurais compostos por uma quantidade grande de neurônios, aos quais recebem estímulos pesados de outros neurônios localizados em curtas e longas distâncias. Devido a essas conexões de longa distância, mesmo se a sua entrada de hierarquia imediatamente abaixo tiver uma saída igual a zero, o bipolo pode gerar respostas diferentes de zero. Os blocos que implementam os bipolos possuem filtros que deslocam a imagem em cinco diferentes distâncias e soma

todas as imagens deslocadas. Operação que é equivalente a operação realizada pelo bipolo biológico.

Operação SC com estímulos não nulos do estágio BC: Como já foi definido na Eq. (5.4), a saída do bloco SC produz imagens compostas de segmentos de contornos colineares. O bloco BG produz imagens onde esses segmentos são aproximados. Essa informação é adicionada a saída do bloco $v_{FD,\theta}(x, y)$ e o resultado dessa soma é inserido novamente na entrada do bloco SC. Assim, a cada passo do processamento cada vez mais esses segmentos vão se aproximando até que ocorre uma união entre eles.

5.3 Resultados do DFI

Duas implementações, uma pulsada e outra em modo de amplitude, dos circuitos corticais do *diffusive filling-in*, de acordo com os esquemas apresentados nas Figs. 5.3 e 5.4 foram realizadas. O modo pulsado segue o modelo de neurônio apresentado na Eq. 5.1 e o modo de amplitude segue o modelo apresentado na Eq. 5.2. Os estímulos apresentados na Fig. 4.12 foram utilizados como entradas para ambas as implementações. Os resultados da simulação das duas implementações são mostrados na Fig. 5.5. Na simulação da implementação pulsada da operação do BCS (Fig. 5.5, coluna 1) os pixels próximos aos contornos possuem variação em excesso. Essas variações, que não aparecem no modo em amplitude (Fig. 5.5, coluna 2), podem ser interpretadas como sendo ruído. Contudo, exceto por isso, as simulações da operação DFI (Fig. 5.5, colunas 3 e 4) nas duas implementações, pulsada na coluna 3 e em amplitude na coluna 4, são similares. Na Fig. 5.6, é mostrado o erro médio quadrático (MSE) entre os resultados da operação DFI na implementação pulsada e na implementação em modo de amplitude. É possível perceber que o erro médio quadrático converge para um valor em torno de 0.01 após um intervalo de tempo de 0.4 segundos.

5.4 Conclusões

A Fig. 5.5 mostra que os resultados de simulações do DFI em modo de amplitude são próximos aos resultados de simulação que são obtidos do DFI em modo pulsado, apesar dos artefatos ruidosos que aparecem no modo pulsado. A partir desses resultados é possível concluir que o modo de amplitude pode ser utilizado para prever os resultados de simulações em modo pulsado em termos de funcionamento do algoritmo do DFI. Para o caso do DFI, o erro (MSE) é menor do que 0.1 mesmo

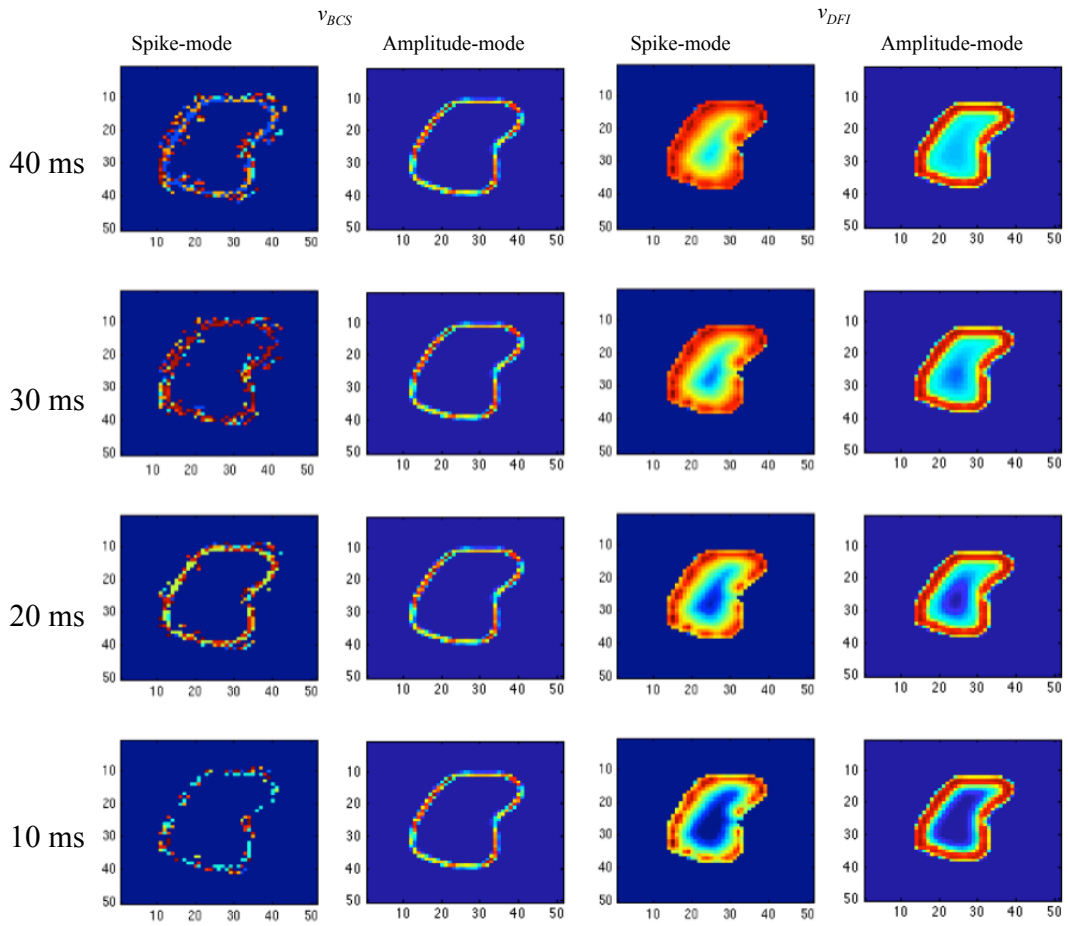


Figura 5.5: Resultado das operações de BCS (as duas colunas a direita) e DFI (as duas colunas a esquerda) em 4 diferentes instantes de tempo (10, 20, 30 e 40 mili-segundos). A primeira e a terceira colunas se referem aos resultados da implementação pulsada e as outras, aos resultados da implementação em modo de amplitude.

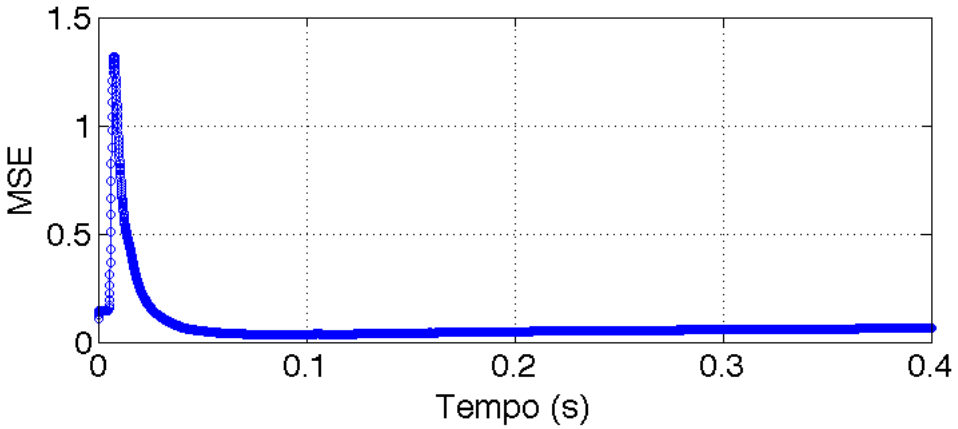


Figura 5.6: Erro médio quadrático entre os resultados da operação DFI nas implementações em modo pulsado e em modo de amplitude.

considerando as distorções inseridas pela filtragem espacial da OPL e da utilização de uma topologia de redução da complexidade de *hardware* SA.

Capítulo 6

Conclusões e Trabalhos Futuros

Foram realizadas simulações para duas implementações dos circuitos corticais do algoritmo *diffusive filling-in*, o BCS e o FFI. Uma em modo pulsado e outra e em modo de amplitude. Em ambas as simulações foram utilizadas entradas que foram processadas a partir da resposta ao impulso de um imageador (diagrama esquemático) que implementa os estágios de processamento da retina. Essa abordagem é importante porque reforça as diferenças entre as simulações das duas implementações ao inserir distorções características provocadas pela filtragem da OPL e pela topologia de projeto de *hardware* SA.

No Cap. 2 os modelos da OPL, IPL e células ganglionares são apresentados. Uma metodologia de projeto para implementar redes resistivas CMOS, utilizadas para implementação de modelos de OPL, foi proposta. A OPL da retina apresentada foi projetada a partir dessa metodologia. Resultados de simulação mostram o correto funcionamento da OPL e da IPL e assim também da metodologia proposta.

No Cap. 3 é apresentada uma técnica que permite a redução da complexidade de *hardware* mantendo a latência da comunicação menor (cerca de 50% menos) do que aquela que costuma ser utilizada para o mesmo propósito de redução da complexidade de *hardware*. O sistema de comunicação proposto possui uma perda de informação menor do que 1%, valor bem abaixo do que se é possível alcançar em sistemas não arbitrados convencionais. A validação de funcionamento da técnica proposta é ilustrada pelos resultados apresentados.

No Cap. 4 o diagrama esquemático do imageador é apresentado. Esse diagrama é composto por uma OPL, uma IPL, duas camadas de células ganglionares e um sistema de comunicação que utiliza a técnica de redução de *hardware* proposta. Resultados de reconstrução de estímulos, da resposta ao impulso, dados sobre perda de informação e de avaliação da latência são apresentados. Os estímulos que serão utilizados no DFI são processados e mostrados. Ao utilizar as entradas que foram tratadas a partir da filtragem com a resposta ao impulso do imageador foram consideradas as distorções causadas pela topologia SA.

No Cap. 5 os dois modos de implementação do DFI são apresentados, bem como o modelo do DFI. Exemplos ilustrativos de como os modos de implementação funcionam são mostrados. Resultados de simulação dos dois modos e um resultado comparativo onde se avalia o erro são apresentados. Essas simulações envolvendo os modos de amplitude e pulsado foram propostas para avaliar a capacidade modo de amplitude de prever os resultados do modo pulsado.

Os resultados apresentados sugerem que a metodologia proposta pode ser utilizada para implementação de modelos da OPL em silício, que o sistema de comunicação proposto pode ser utilizado para implementar circuitos corticais e que o modo de amplitude é capaz de gerar previsões próximas às previsões que seriam geradas para o modo pulsado na simulação do DFI. Assim esses resultados apresentados nesse trabalho podem auxiliar na implementação de circuitos corticais em silício.

6.1 Contribuições

As principais contribuições desta pesquisa de tese podem ser resumidas nos seguintes itens:

- Uma metodologia de projeto para redes resistivas utilizando transistores MOS voltado a implementações em *hardware* de modelos da OPL, Cap. 2.
- Um sistema AER não arbitrado que apresenta 50% menos latência do que o sistema AER tradicionalmente utilizado, redução na complexidade de *hardware* semelhante e uma perda de informação por colisão menor do que sistemas não arbitrados convencionais, Cap 3.
- Diagrama esquemático de um imageador que utiliza o sistema de comunicação proposto. Isso mostra que o sistema de comunicação proposto funciona corretamente e pode ser utilizado em projetos de *hardware* de circuitos neurais pulsados, Cap. 4.
- Resultados de simulações dos circuitos corticais do DFI que são indícios de que o modo de implementação em amplitude pode ser utilizado para previsões teóricas do funcionamento de implementações pulsadas mesmo quando os sinais de entrada a serem processados são mais realistas do que aqueles produzidos por uma filtragem utilizando filtros DoG, Cap.5.
- Resultados de simulação que são indícios de que a utilização da topologia SA proposta, para a redução da complexidade de *hardware*, não altera significativamente o funcionamento das implementações dos circuitos corticais, Cap. 5.

6.2 Trabalhos Futuros

As principais propostas para a continuidade desta pesquisa de tese podem ser resumidas nos seguintes itens:

- A implementação em *hardware* (*layout*) do diagrama esquemático do imageador desenvolvido para uma avaliação mais concreta.
- Diagrama esquemático do DFI utilizando topologias de redução de *hardware* SA, Dendrito Compartilhado (*Shared Dendrite* - SD) e Sinapse Compartilhada (*Shared Synapse* - SS) .
- Verificação das previsões feitas em modo de amplitude quando outras topologias de redução da complexidade de *hardware* (SD e SS) são utilizadas.
- A análise de distorções causada por outras topologias de redução da complexidade de *hardware* na implementação de circuitos corticais, como por exemplo, SD e SS.

Referências Bibliográficas

[1] CHOUDHARY, S., SLOAN, S., FOK, S., et al. "Artificial Neural Networks and Machine Learning – ICANN 2012: 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, Proceedings, Part I". cap. Silicon Neurons That Compute, pp. 121–128, Berlin, Heidelberg, Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-33269-2. doi: 10.1007/978-3-642-33269-2_16. Disponível em: <http://dx.doi.org/10.1007/978-3-642-33269-2_16>.

[2] XUE, Y. "Recent development in analog computation: a brief overview", *Analog Integrated Circuits and Signal Processing*, v. 86, n. 2, pp. 181–187, 2016. ISSN: 1573-1979. doi: 10.1007/s10470-015-0668-y. Disponível em: <http://dx.doi.org/10.1007/s10470-015-0668-y>.

[3] MEROLLA, P., ARTHUR, J., AKOPYAN, F., et al. "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm". In: *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pp. 1–4, Sept 2011. doi: 10.1109/CICC.2011.6055294.

[4] BENJAMIN, B. V., GAO, P., MCQUINN, E., et al. "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations." *Proceedings of the IEEE*, v. 102, n. 5, pp. 699–716, 2014. Disponível em: <http://dblp.uni-trier.de/db/journals/pieee/pieee102.html#BenjaminGMCCBAAMB14>.

[5] CASSIDY, A. S., GEORGIOU, J., ANDREOU, A. G. "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization", *Neural Networks*, v. 45, pp. 4 – 26, 2013. ISSN: 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2013.05.011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608013001597>. Neuromorphic Engineering: From Neural Systems to Brain-Like Engineered Systems.

[6] TURING, A. M. "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, v. 2,

n. 42, pp. 230–265, 1936. Disponível em: <http://www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf>.

[7] NEUMANN, J. V. *The Computer and the Brain*. New Haven, CT, USA, Yale University Press, 1958. ISBN: 0300007930.

[8] FURBER, S. B., GALLUPPI, F., TEMPLE, S., et al. "The SpiNNaker Project", *Proceedings of the IEEE*, v. 102, n. 5, pp. 652–665, May 2014. ISSN: 0018-9219. doi: 10.1109/JPROC.2014.2304638.

[9] CAO, Y., GROSSBERG, S. "Stereopsis and 3D surface perception by spiking neurons in laminar cortical circuits: A method for converting neural rate models into spiking models", *Neural Networks*, v. 26, pp. 75 – 98, 2012. ISSN: 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2011.10.010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608011002723>.

[10] SANTOS, N. G., GOMES, J. G. R. C. "Implementation of a biologically inspired Diffusive Filling-In algorithm for focal-plane image processing applications". In: *Proc, BRIC-CCI and CBIC, Recife, Brazil, Conference on Computational Intelligence*, Oct 2013.

[11] KUNKEL, S., SCHMIDT, M., EPPLER, J. M., et al. "Spiking network simulation code for petascale computers", *Frontiers in Neuroinformatics*, v. 8, n. 78, 2014. ISSN: 1662-5196. doi: 10.3389/fninf.2014.00078. Disponível em: <http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2014.00078/abstract>.

[12] CULURCIELLO, E., ANDREOU, A. "A comparative study of access topologies for chip-level address-event communication channels", *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, v. 14, n. 5, pp. 1266—1277, 2003. ISSN: 1045-9227. doi: 10.1109/tnn.2003.816385. Disponível em: <http://dx.doi.org/10.1109/TNN.2003.816385>.

[13] JÚNIOR, M. F., ROSA, M., GATTASS, R., et al. "Dynamic surrounds of receptive fields in primate striate cortex: a physiological basis for perceptual completion?" *Proceedings of the National Academy of Sciences*, v. 89, n. 18, pp. 8547–8551, 1992.

[14] GROSSBERG, S., KUHLMANN, L., MINGOLLA, E. "A neural model of 3D shape-from-texture: Multiple-scale filtering, boundary grouping, and surface filling-in", *Vision Research*, v. 47, n. 5, pp. 634 – 672, 2007. ISSN:

0042-6989. doi: http://dx.doi.org/10.1016/j.visres.2006.10.024. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0042698906004950>.

[15] GOVE, A., GROSSBERG, S., MINGOLLA, E. "Brightness perception, illusory contours, and corticogeniculate feedback", *Visual Neuroscience*, v. 12, pp. 1027–1052, 11 1995. ISSN: 1469-8714. doi: 10.1017/S0952523800006702. Disponível em: <http://journals.cambridge.org/article_S0952523800006702>.

[16] BOAHEN, K. A. *Retinomorphic Vision Systems: Reverse Engineering the Vertebrate Retina*. Ph.D. thesis, California Institute of Technology, Pasadena, CA, 1997.

[17] ZAGHLOUL, K. A., BOAHEN, K. "A silicon retina that reproduces signals in the optic nerve", *Journal of Neural Engineering*, v. 3, n. 4, pp. 257, 2006. Disponível em: <http://stacks.iop.org/1741-2552/3/i=4/a=002>.

[18] ZAMPIGHI, G. "Gap Junction Structure". In: De Mello, W. C. (Ed.), *Cell-to-Cell Communication*, pp. 1–28, Boston, MA, Springer US, 1987. ISBN: 978-1-4613-1917-7. doi: 10.1007/978-1-4613-1917-7_1. Disponível em: <http://dx.doi.org/10.1007/978-1-4613-1917-7_1>.

[19] SCHWARTZ, R., MARR, D. "Vision." *The Philosophical Review*, v. 94, n. 3, pp. 411, 1985.

[20] KANDEL, E. *Principles of neural science*. New York, Elsevier, 1991. ISBN: 0444015620.

[21] CHEN, E. P., FREEMAN, A. W. "A model for spatiotemporal frequency responses in the X cell pathway of the cat's retina", *Vision Research*, v. 29, n. 3, pp. 271 – 291, 1989. ISSN: 0042-6989. doi: http://dx.doi.org/10.1016/0042-6989(89)90076-X. Disponível em: <http://www.sciencedirect.com/science/article/pii/004269898990076X>.

[22] IZHIKEVICH, E. "Which model to use for cortical spiking neurons?" *Neural Networks, IEEE Transactions on*, v. 15, n. 5, pp. 1063–1070, Sept 2004. ISSN: 1045-9227. doi: 10.1109/TNN.2004.832719.

[23] JACK, J. J. B. *Electric current flow in excitable cells*. Oxford, Clarendon Press, 1975. ISBN: 0198573650.

[24] MAHOWALD, M. *VLSI analogs of neuronal visual processing: a synthesis of form and function.* Tese de Doutorado, California Institute of Technology, 1992.

[25] MEAD, C. *Analog VLSI and Neural Systems.* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0-201-05992-4.

[26] CULURCIELLO, E., ETIENNE-CUMMINGS, R., BOAHEN, K. "Arbitrated address event representation digital image sensor". In: *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, pp. 92–93, Feb 2001. doi: 10.1109/ISSCC.2001.912560.

[27] ENZ, C. C., KRUMMENACHER, F., VITTOZ, E. A. "An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications", *Analog Integrated Circuits and Signal Processing*, v. 8, n. 1, pp. 83–114. ISSN: 1573-1979. doi: 10.1007/BF01239381. Disponível em: <`http://dx.doi.org/10.1007/BF01239381`>.

[28] WANG, A., CALHOUN, B. H., CHANDRAKASAN, A. P. *Sub-threshold Design for Ultra Low-Power Systems (Series on Integrated Circuits and Systems).* Secaucus, NJ, USA, Springer-Verlag New York, Inc., 2006. ISBN: 0387335153.

[29] MARR, D. *Vision : a computational investigation into the human representation and processing of visual information.* San Francisco, W.H. Freeman, 1982. ISBN: 0-7167-1284-9. Disponível em: <`http://opac.inria.fr/record=b1079861`>.

[30] KATSIAMIS, A. G., GLAROS, K. N., DRAKAKIS, E. M. "Insights and Advances on the Design of CMOS Sinh Companding Filters", *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 55, n. 9, pp. 2539–2550, Oct 2008. ISSN: 1549-8328. doi: 10.1109/TCSI.2008.921037.

[31] GERSTNER, W., KISTLER, W. M. *Spiking neuron models : single neurons, populations, plasticity.* Cambridge, New York, Melbourne, Cambridge University Press, 2002. ISBN: 978-0-521-81384-6. Disponível em: <`http://opac.inria.fr/record=b1099891`>. Autre tirage : 2006, 2008 (4e).

[32] BOAHEN, K. A. "Point-to-Point Connectivity Between Neuromorphic Chips Using Address-Events", *IEEE Transactions on Circuits and Systems II*, v. 47, n. 5, pp. 416–34, 2000.

[33] BOAHEN, K. "A throughput-on-demand address-event transmitter for neuromorphic chips". In: *Advanced Research in VLSI, 1999. Proceedings. 20th Anniversary Conference on*, pp. 72–86, Mar 1999. doi: 10.1109/ARVLSI.1999.756038.

[34] KALAYJIAN, Z., ANDREOU, A. G. "Asynchronous Communication of 2D Motion Information Using Winner-Takes-All Arbitration", *Analog Integrated Circuits and Signal Processing*, v. 13, n. 1, pp. 103–109, 1997. ISSN: 1573-1979. doi: 10.1023/A:1008236112778. Disponível em: <http://dx.doi.org/10.1023/A:1008236112778>.

[35] BRAJOVIC, V. "Lossless non-arbitrated address-event coding". In: *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, v. 5, pp. V–825–V–828 vol.5, May 2003. doi: 10.1109/ISCAS.2003.1206440.

[36] EGGERMONT, J. J., EPPING, W. J. M., AERTSEN, A. M. H. J. "Stimulus dependent neural correlations in the auditory midbrain of the grass-frog (Rana temporaria L.)", *Biological Cybernetics*, v. 47, n. 2, pp. 103–117, 1983. ISSN: 1432-0770. doi: 10.1007/BF00337084. Disponível em: <http://dx.doi.org/10.1007/BF00337084>.

[37] THORPE, S., FIZE, D., MARLOT, C. "Speed of Processing in the Human Visual System." *Nature*, v. 381, pp. 520, 1996.

[38] STRAIN, T. J., MCDAID, L. J., MCGINNITY, T. M., et al. "An STDP Training Algotithm for a Spiking Neural Network with Dynamic Threshold Neurons", *International Journal of Neural Systems*, v. 20, n. 06, pp. 463–480, 2010. doi: 10.1142/S0129065710002553. Disponível em: <http://www.worldscientific.com/doi/abs/10.1142/S0129065710002553>. PMID: 21117270.

[39] STROMATIAS, E., MARSLAND, J. S. "Supervised learning in Spiking Neural Networks with Limited Precision: SNN/LP", *CoRR*, v. abs/1407.0265, 2014. Disponível em: <http://arxiv.org/abs/1407.0265>.

[40] SANTOS, N. G., GOMES, J. G. R. C. "A Comparison between Spike-Mode and Amplitude-Mode Implementations of the Diffusive Filling-In Algorithm, to Analog Integrated Circuits and Signal Processing", *Manuscrito submetido para publicação - SPRINGER*, 2016.

[41] LEVITT, J. B., KIPER, D. C., MOVSHON, J. A. "Receptive fields and functional architecture of macaque V2", *Journal of Neurophysiology*, v. 71,

n. 6, pp. 2517–2542, 1994. ISSN: 0022-3077. Disponível em: <http://jn.physiology.org/content/71/6/2517>.

[42] YAZDANBAKHSH, A., GROSSBERG, S. "Fast synchronization of perceptual grouping in laminar visual cortical circuits", *Neural Networks*, v. 17, n. 5-6, pp. 707–718, 2004. doi: 10.1016/j.neunet.2004.06.005. Disponível em: <http://dx.doi.org/10.1016/j.neunet.2004.06.005>.

[43] SEO, J., SHNEIDERMAN, B. "A Rank-by-Feature Framework for Interactive Exploration of Multidimensional Data", *Information Visualizationnformation Visualization*, v. 4, pp. 96 – 113, 2005/06/20/ 2005. doi: 10.1057/palgrave.ivs.9500091. Disponível em: <http://ivi.sagepub.com/content/4/2/96>.

[44] COHEN, M. A., GROSSBERG, S. "Neural dynamics of brightness perception: Features, boundaries, diffusion, and resonance", *Perception & Psychophysics*, v. 36, n. 5, pp. 428–456. ISSN: 1532-5962. doi: 10.3758/BF03207497. Disponível em: <http://dx.doi.org/10.3758/BF03207497>.

[45] BOSKING, W. H., ZHANG, Y., SCHOFIELD, B., et al. "Orientation Selectivity and the Arrangement of Horizontal Connections in Tree Shrew Striate Cortex", *The Journal of Neuroscience*, v. 17, n. 6, pp. 2112–2127, mar. 1997. ISSN: 1529-2401. Disponível em: <http://www.jneurosci.org/content/17/6/2112.abstract>.

[46] AZZI, J. A. C. B., GATTASS, R., LIMA, B., et al. "Precise visuotopic organization of the blind spot representation in primate V1", *Journal of Neurophysiology*, v. 113, n. 10, pp. 3588–3599, jun. 2015. ISSN: 1522-1598. doi: 10.1152/jn.00418.2014. Disponível em: <http://dx.doi.org/10.1152/jn.00418.2014>.

Apêndice A

Scripts Ocean

A.1 Script Diagonal

```
; 20/06/2015

ocnWaveformTool( 'wavescan )
simulator( 'spectre )

; Leitura das bibliotecas
design(  "/home/netware/../netlist")
resultsDir( "/home/netware/../spectre/schematic" )
modelFile(
    '("/cds/Opus/user/AMS370/spectre/c35/mcparams.scs" "")
    '("/cds/Opus/user/AMS370/spectre/c35/cmos53.scs" "cmostm")
    '("/cds/Opus/user/AMS370/spectre/c35/res.scs" "restm")
    '("/cds/Opus/user/AMS370/spectre/c35/cap.scs" "captm")
    '("/cds/Opus/user/AMS370/spectre/c35/bip.scs" "biptm")
    '("/cds/Opus/user/AMS370/spectre/c35/ind.scs" "indtm")
)

analysis('tran ?stop "50m" ?save "selected"  ?compression "yes"
?finalTimeOp nil  )
; Atribuicao de valores  (desVar)

desVar(   "I1_1"   0 )
desVar(   "I1_2"   0 )
desVar(   "I1_3"   0 )
desVar(   "I1_4"   0 )
```

```
desVar(   "I1_5"   0 )
desVar(   "I1_6"   0 )
desVar(   "I1_7"   0 )
desVar(   "I1_8"   0 )
desVar(   "I1_9"   0 )
desVar(   "I4_44"   0 )
desVar(   "I4_45"   0 )
desVar(   "I4_46"   0 )
desVar(   "I4_47"   0 )
desVar(   "I4_48"   0 )
desVar(   "I4_49"   0 )
desVar(   "I14_49"   0 )
desVar(   "I14_50"   0 )
desVar(   "I15_1"   0 )
desVar(   "I15_2"   0 )
desVar(   "I15_3"   0 )
desVar(   "I15_4"   0 )
desVar(   "I15_5"   0 )
desVar(   "I15_6"   0 )
desVar(   "I15_7"   0 )
desVar(   "I15_8"   0 )
desVar(   "I15_9"   0 )
desVar(   "I15_10"   0 )
desVar(   "I15_11"   0 )
desVar(   "I15_12"   0 )
desVar(   "I15_13"   0 )
desVar(   "I15_14"   0 )
desVar(   "I15_15"   0 )
desVar(   "I15_16"   0 )
desVar(   "I15_17"   0 )
desVar(   "I15_18"   0 )
desVar(   "I15_19"   0 )
desVar(   "I15_20"   0 )
desVar(   "I15_21"   0 )
desVar(   "I15_22"   0 )
desVar(   "I15_23"   0 )
desVar(   "I15_24"   0 )
desVar(   "I15_25"   20p )
desVar(   "I15_26"   0 )
```

```
desVar(   "I15_27"   0 )
desVar(   "I15_28"   0 )
desVar(   "I15_29"   0 )
desVar(   "I15_30"   0 )
desVar(   "I15_31"   0 )
desVar(   "I15_32"   0 )
desVar(   "I15_33"   0 )
desVar(   "I15_34"   0 )
desVar(   "I15_35"   0 )
desVar(   "I15_36"   0 )
desVar(   "I15_37"   0 )
desVar(   "I15_38"   0 )
desVar(   "I15_39"   0 )
desVar(   "I15_40"   0 )
desVar(   "I15_41"   0 )
desVar(   "I15_42"   0 )
desVar(   "I15_43"   0 )
desVar(   "I15_44"   0 )
desVar(   "I15_45"   0 )
desVar(   "I15_46"   0 )
desVar(   "I15_47"   0 )
desVar(   "I15_48"   0 )
desVar(   "I15_49"   0 )
desVar(   "I15_50"   0 )
desVar(   "I16_1"   0 )
desVar(   "I16_2"   0 )
desVar(   "I16_3"   0 )
desVar(   "I16_4"   0 )
desVar(   "I16_5"   0 )
desVar(   "I16_6"   0 )
desVar(   "I16_7"   0 )
desVar(   "I16_8"   0 )
desVar(   "I16_9"   0 )
desVar(   "I16_10"   0 )
desVar(   "I16_11"   0 )
desVar(   "I16_12"   0 )
desVar(   "I16_13"   0 )
desVar(   "I16_14"   0 )
desVar(   "I16_15"   0 )
```

```
desVar(   "I16_16"   0 )
desVar(   "I16_17"   0 )
desVar(   "I16_18"   0 )
desVar(   "I16_19"   0 )
desVar(   "I16_20"   0 )
desVar(   "I16_21"   0 )
desVar(   "I16_22"   0 )
desVar(   "I16_23"   0 )
desVar(   "I16_24"   0 )
desVar(   "I16_25"   20p )
desVar(   "I16_26"   0 )
desVar(   "I16_27"   0 )
desVar(   "I16_28"   0 )
desVar(   "I16_29"   0 )
desVar(   "I16_30"   0 )
desVar(   "I16_31"   0 )
desVar(   "I16_32"   0 )
desVar(   "I16_33"   0 )
desVar(   "I16_34"   0 )
desVar(   "I16_35"   0 )
desVar(   "I16_36"   0 )
desVar(   "I16_37"   0 )
desVar(   "I16_38"   0 )
desVar(   "I16_39"   0 )
desVar(   "I16_40"   0 )
desVar(   "I16_41"   0 )
desVar(   "I16_42"   0 )
desVar(   "I16_43"   0 )
desVar(   "I16_44"   0 )
desVar(   "I16_45"   0 )
desVar(   "I16_46"   0 )
desVar(   "I16_47"   0 )
desVar(   "I16_48"   0 )
desVar(   "I16_49"   0 )
desVar(   "I16_50"   0 )
desVar(   "I17_1"   0 )
desVar(   "I17_2"   0 )
desVar(   "I17_3"   0 )
desVar(   "I17_4"   0 )
```

```
desVar(   "I17_5"   0 )
desVar(   "I17_6"   0 )
desVar(   "I17_7"   0 )
desVar(   "I17_8"   0 )
desVar(   "I17_9"   0 )
desVar(   "I17_10"   0 )
desVar(   "I17_11"   0 )
desVar(   "I17_12"   0 )
desVar(   "I17_13"   0 )
desVar(   "I17_14"   0 )
desVar(   "I17_15"   0 )
desVar(   "I17_16"   0 )
desVar(   "I17_17"   0 )
desVar(   "I17_18"   0 )
desVar(   "I17_19"   0 )
desVar(   "I17_20"   0 )
desVar(   "I17_21"   0 )
desVar(   "I17_22"   0 )
desVar(   "I17_23"   0 )
desVar(   "I17_24"   0 )
desVar(   "I17_25"   20p )
desVar(   "I17_26"   0 )
desVar(   "I17_27"   0 )
desVar(   "I17_28"   0 )
desVar(   "I17_29"   0 )
desVar(   "I17_30"   0 )
desVar(   "I17_31"   0 )
desVar(   "I17_32"   0 )
desVar(   "I17_33"   0 )
desVar(   "I17_34"   0 )
desVar(   "I17_35"   0 )
desVar(   "I17_36"   0 )
desVar(   "I17_37"   0 )
desVar(   "I17_38"   0 )
desVar(   "I17_39"   0 )
desVar(   "I17_40"   0 )
desVar(   "I17_41"   0 )
desVar(   "I17_42"   0 )
desVar(   "I17_43"   0 )
```

```
desVar(   "I17_44"   0 )
desVar(   "I17_45"   0 )
desVar(   "I17_46"   0 )
desVar(   "I17_47"   0 )
desVar(   "I17_48"   0 )
desVar(   "I17_49"   0 )
desVar(   "I17_50"   0 )
desVar(   "I18_1"   0 )
desVar(   "I18_2"   0 )
desVar(   "I18_3"   0 )
desVar(   "I18_4"   0 )
desVar(   "I18_5"   0 )
desVar(   "I18_6"   0 )
desVar(   "I18_7"   0 )
desVar(   "I18_8"   0 )
desVar(   "I18_9"   0 )
desVar(   "I18_10"   0 )
desVar(   "I18_11"   0 )
desVar(   "I18_12"   0 )
desVar(   "I18_13"   0 )
desVar(   "I18_14"   0 )
desVar(   "I18_15"   0 )
desVar(   "I18_16"   0 )
desVar(   "I18_17"   0 )
desVar(   "I18_18"   0 )
desVar(   "I18_19"   0 )
desVar(   "I18_20"   0 )
desVar(   "I18_21"   0 )
desVar(   "I18_22"   0 )
desVar(   "I18_23"   0 )
desVar(   "I18_24"   0 )
desVar(   "I18_25"   20p )
desVar(   "I18_26"   0 )
desVar(   "I19_22"   0 )
desVar(   "I19_23"   0 )
desVar(   "I19_24"   0 )
desVar(   "I19_25"   20p )
desVar(   "I19_26"   0 )
desVar(   "I19_27"   0 )
```

```
desVar(   "I20_22"   0 )
desVar(   "I20_23"   0 )
desVar(   "I20_24"   0 )
desVar(   "I20_25"   20p )
desVar(   "I20_26"   0 )
desVar(   "I20_27"   0 )
desVar(   "I20_28"   0 )
desVar(   "I20_29"   0 )
desVar(   "I20_30"   0 )
desVar(   "I20_31"   0 )
desVar(   "I20_32"   0 )
desVar(   "I20_33"   0 )
desVar(   "I20_34"   0 )
desVar(   "I20_35"   0 )
desVar(   "I20_36"   0 )
desVar(   "I20_37"   0 )
desVar(   "I20_38"   0 )
desVar(   "I20_39"   0 )
desVar(   "I21_24"   0 )
desVar(   "I21_25"   20p )
desVar(   "I21_26"   0 )
desVar(   "I21_27"   0 )
desVar(   "I21_28"   0 )
desVar(   "I21_29"   0 )
desVar(   "I21_30"   0 )
desVar(   "I21_31"   0 )
desVar(   "I21_32"   0 )
desVar(   "I21_33"   0 )
desVar(   "I21_34"   0 )
desVar(   "I21_35"   0 )
desVar(   "I21_36"   0 )
desVar(   "I21_37"   0 )
desVar(   "I21_38"   0 )
desVar(   "I21_39"   0 )
desVar(   "I21_40"   0 )
desVar(   "I21_41"   0 )
desVar(   "I21_42"   0 )
desVar(   "I21_43"   0 )
desVar(   "I21_44"   0 )
```

```
desVar(   "I21_45"   0 )
desVar(   "I21_46"   0 )
desVar(   "I21_47"   0 )
desVar(   "I21_48"   0 )
desVar(   "I21_49"   0 )
desVar(   "I21_50"   0 )
desVar(   "I22_1"   0 )
desVar(   "I22_2"   0 )
desVar(   "I22_3"   0 )
desVar(   "I22_4"   0 )
desVar(   "I22_5"   0 )
desVar(   "I22_6"   0 )
desVar(   "I22_7"   0 )
desVar(   "I22_8"   0 )
desVar(   "I22_9"   0 )
desVar(   "I22_10"   0 )
desVar(   "I22_11"   0 )
desVar(   "I22_12"   0 )
desVar(   "I22_13"   0 )
desVar(   "I22_14"   0 )
desVar(   "I22_15"   0 )
desVar(   "I22_16"   0 )
desVar(   "I22_17"   0 )
desVar(   "I22_18"   0 )
desVar(   "I22_19"   0 )
desVar(   "I22_20"   0 )
desVar(   "I22_21"   0 )
desVar(   "I22_22"   0 )
desVar(   "I22_23"   0 )
desVar(   "I22_24"   0 )
desVar(   "I22_25"   20p )
desVar(   "I22_26"   0 )
desVar(   "I22_27"   0 )
desVar(   "I22_28"   0 )
desVar(   "I22_29"   0 )
desVar(   "I22_30"   0 )
desVar(   "I22_31"   0 )
desVar(   "I22_32"   0 )
desVar(   "I22_33"   0 )
```

```
desVar(   "I22_34"   0 )
desVar(   "I22_35"   0 )
desVar(   "I22_36"   0 )
desVar(   "I22_37"   0 )
desVar(   "I22_38"   0 )
desVar(   "I22_39"   0 )
desVar(   "I22_40"   0 )
desVar(   "I22_41"   0 )
desVar(   "I22_42"   0 )
desVar(   "I22_43"   0 )
desVar(   "I22_44"   0 )
desVar(   "I22_45"   0 )
desVar(   "I22_46"   0 )
desVar(   "I22_47"   0 )
desVar(   "I22_48"   0 )
desVar(   "I22_49"   0 )
desVar(   "I22_50"   0 )
desVar(   "I23_1"   0 )
desVar(   "I23_2"   0 )
desVar(   "I23_3"   0 )
desVar(   "I23_4"   0 )
desVar(   "I23_5"   0 )
desVar(   "I23_6"   0 )
desVar(   "I23_7"   0 )
desVar(   "I23_8"   0 )
desVar(   "I23_9"   0 )
desVar(   "I23_10"   0 )
desVar(   "I23_11"   0 )
desVar(   "I23_12"   0 )
desVar(   "I23_13"   0 )
desVar(   "I23_14"   0 )
desVar(   "I23_15"   0 )
desVar(   "I23_16"   0 )
desVar(   "I23_17"   0 )
desVar(   "I23_18"   0 )
desVar(   "I23_19"   0 )
desVar(   "I23_20"   0 )
desVar(   "I23_21"   0 )
desVar(   "I23_22"   0 )
```

```
desVar(   "I23_23"   0 )
desVar(   "I23_24"   0 )
desVar(   "I23_25"   20p )
desVar(   "I23_26"   0 )
desVar(   "I23_27"   0 )
desVar(   "I23_28"   0 )
desVar(   "I23_29"   0 )
desVar(   "I23_30"   0 )
desVar(   "I23_31"   0 )
desVar(   "I23_32"   0 )
desVar(   "I23_33"   0 )
desVar(   "I23_34"   0 )
desVar(   "I23_35"   0 )
desVar(   "I23_36"   0 )
desVar(   "I23_37"   0 )
desVar(   "I23_38"   0 )
desVar(   "I23_39"   0 )
desVar(   "I23_40"   0 )
desVar(   "I23_41"   0 )
desVar(   "I23_42"   0 )
desVar(   "I23_43"   0 )
desVar(   "I23_44"   0 )
desVar(   "I23_45"   0 )
desVar(   "I23_46"   0 )
desVar(   "I23_47"   0 )
desVar(   "I23_48"   0 )
desVar(   "I23_49"   0 )
desVar(   "I23_50"   0 )
desVar(   "I24_1"   0 )
desVar(   "I24_2"   0 )
desVar(   "I24_3"   0 )
desVar(   "I24_4"   0 )
desVar(   "I24_5"   0 )
desVar(   "I24_6"   0 )
desVar(   "I24_7"   0 )
desVar(   "I24_8"   0 )
desVar(   "I24_9"   0 )
desVar(   "I24_10"   0 )
desVar(   "I24_11"   0 )
```

```
desVar(   "I24_12"   0 )
desVar(   "I24_13"   0 )
desVar(   "I24_14"   0 )
desVar(   "I24_15"   0 )
desVar(   "I24_16"   0 )
desVar(   "I24_17"   0 )
desVar(   "I24_18"   0 )
desVar(   "I24_19"   0 )
desVar(   "I24_20"   0 )
desVar(   "I24_21"   0 )
desVar(   "I24_22"   0 )
desVar(   "I24_23"   0 )
desVar(   "I24_24"   0 )
desVar(   "I24_25"   20p )
desVar(   "I24_26"   0 )
desVar(   "I24_27"   0 )
desVar(   "I24_28"   0 )
desVar(   "I24_29"   0 )
desVar(   "I24_30"   0 )
desVar(   "I24_31"   0 )
desVar(   "I24_32"   0 )
desVar(   "I24_33"   0 )
desVar(   "I24_34"   0 )
desVar(   "I24_35"   0 )
desVar(   "I24_36"   0 )
desVar(   "I24_37"   0 )
desVar(   "I24_38"   0 )
desVar(   "I24_39"   0 )
desVar(   "I24_40"   0 )
desVar(   "I24_41"   0 )
desVar(   "I24_42"   0 )
desVar(   "I24_43"   0 )
desVar(   "I24_44"   0 )
desVar(   "I24_45"   0 )
desVar(   "I24_46"   0 )
desVar(   "I24_47"   0 )
desVar(   "I24_48"   0 )
desVar(   "I24_49"   0 )
desVar(   "I24_50"   0 )
```

```
desVar(   "I25_1"   0 )
desVar(   "I25_2"   0 )
desVar(   "I25_3"   0 )
desVar(   "I25_4"   0 )
desVar(   "I25_5"   0 )
desVar(   "I25_6"   0 )
desVar(   "I25_7"   0 )
desVar(   "I25_8"   0 )
desVar(   "I25_9"   0 )
desVar(   "I25_10"   0 )
desVar(   "I25_11"   0 )
desVar(   "I25_12"   0 )
desVar(   "I25_13"   0 )
desVar(   "I25_14"   0 )
desVar(   "I25_15"   0 )
desVar(   "I25_16"   0 )
desVar(   "I25_17"   0 )
desVar(   "I25_18"   0 )
desVar(   "I25_19"   0 )
desVar(   "I25_20"   0 )
desVar(   "I25_21"   0 )
desVar(   "I25_22"   0 )
desVar(   "I25_23"   0 )
desVar(   "I25_24"   0 )
desVar(   "I25_25"   20p )
desVar(   "I25_26"   0 )
desVar(   "I25_27"   0 )
desVar(   "I25_28"   0 )
desVar(   "I25_29"   0 )
desVar(   "I25_30"   0 )
desVar(   "I25_31"   0 )
desVar(   "I25_32"   0 )
desVar(   "I25_33"   0 )
desVar(   "I25_34"   0 )
desVar(   "I25_35"   0 )
desVar(   "I25_36"   0 )
desVar(   "I25_37"   0 )
desVar(   "I25_38"   0 )
desVar(   "I25_39"   0 )
```

```
desVar(   "I25_40"   0 )
desVar(   "I25_41"   0 )
desVar(   "I25_42"   0 )
desVar(   "I25_43"   0 )
desVar(   "I25_44"   0 )
desVar(   "I25_45"   0 )
desVar(   "I25_46"   0 )
desVar(   "I25_47"   0 )
desVar(   "I25_48"   0 )
desVar(   "I25_49"   0 )
desVar(   "I25_50"   0 )
desVar(   "I26_1"   0 )
desVar(   "I26_2"   0 )
desVar(   "I26_3"   0 )
desVar(   "I26_4"   0 )
desVar(   "I26_5"   0 )
desVar(   "I26_6"   0 )
desVar(   "I26_7"   0 )
desVar(   "I26_8"   0 )
desVar(   "I26_9"   0 )
desVar(   "I26_10"   0 )
desVar(   "I26_11"   0 )
desVar(   "I26_12"   0 )
desVar(   "I26_13"   0 )
desVar(   "I26_14"   0 )
desVar(   "I26_15"   0 )
desVar(   "I26_16"   0 )
desVar(   "I26_17"   0 )
desVar(   "I26_18"   0 )
desVar(   "I26_19"   0 )
desVar(   "I26_20"   0 )
desVar(   "I26_21"   0 )
desVar(   "I26_22"   0 )
desVar(   "I26_23"   0 )
desVar(   "I26_24"   0 )
desVar(   "I26_25"   20p )
desVar(   "I26_26"   0 )
desVar(   "I26_27"   0 )
desVar(   "I26_28"   0 )
```

```
desVar(   "I26_29"   0 )
desVar(   "I26_30"   0 )
desVar(   "I26_31"   0 )
desVar(   "I26_32"   0 )
desVar(   "I26_33"   0 )
desVar(   "I26_34"   0 )
desVar(   "I26_35"   0 )
desVar(   "I26_36"   0 )
desVar(   "I26_37"   0 )
desVar(   "I26_38"   0 )
desVar(   "I26_39"   0 )
desVar(   "I26_40"   0 )
desVar(   "I26_41"   0 )
desVar(   "I26_42"   0 )
desVar(   "I26_43"   0 )
desVar(   "I26_44"   0 )
desVar(   "I26_45"   0 )
desVar(   "I26_46"   0 )
desVar(   "I26_47"   0 )
desVar(   "I26_48"   0 )
desVar(   "I26_49"   0 )
desVar(   "I26_50"   0 )
desVar(   "I27_1"   0 )
desVar(   "I27_2"   0 )
desVar(   "I27_3"   0 )
desVar(   "I27_4"   0 )
desVar(   "I27_5"   0 )
desVar(   "I27_6"   0 )
desVar(   "I27_7"   0 )
desVar(   "I27_8"   0 )
desVar(   "I27_9"   0 )
desVar(   "I27_10"   0 )
desVar(   "I27_11"   0 )
desVar(   "I27_12"   0 )
desVar(   "I27_13"   0 )
desVar(   "I27_14"   0 )
desVar(   "I27_15"   0 )
desVar(   "I27_16"   0 )
desVar(   "I27_17"   0 )
```

```
desVar(   "I27_18"   0 )
desVar(   "I27_19"   0 )
desVar(   "I27_20"   0 )
desVar(   "I27_21"   0 )
desVar(   "I27_22"   0 )
desVar(   "I27_23"   0 )
desVar(   "I27_24"   0 )
desVar(   "I27_25"   20p )
desVar(   "I27_26"   0 )
desVar(   "I27_27"   0 )
desVar(   "I27_28"   0 )
desVar(   "I27_29"   0 )
desVar(   "I27_30"   0 )
desVar(   "I27_31"   0 )
desVar(   "I27_32"   0 )
desVar(   "I27_33"   0 )
desVar(   "I27_34"   0 )
desVar(   "I27_35"   0 )
desVar(   "I27_36"   0 )
desVar(   "I27_37"   0 )
desVar(   "I27_38"   0 )
desVar(   "I27_39"   0 )
desVar(   "I27_40"   0 )
desVar(   "I27_41"   0 )
desVar(   "I27_42"   0 )
desVar(   "I27_43"   0 )
desVar(   "I27_44"   0 )
desVar(   "I27_45"   0 )
desVar(   "I27_46"   0 )
desVar(   "I27_47"   0 )
desVar(   "I27_48"   0 )
desVar(   "I27_49"   0 )
desVar(   "I27_50"   0 )
desVar(   "I28_1"   0 )
desVar(   "I28_2"   0 )
desVar(   "I28_3"   0 )
desVar(   "I28_4"   0 )
desVar(   "I28_5"   0 )
desVar(   "I28_6"   0 )
```

```
desVar(   "I28_7"   0 )
desVar(   "I28_8"   0 )
desVar(   "I28_9"   0 )
desVar(   "I28_10"   0 )
desVar(   "I28_11"   0 )
desVar(   "I28_12"   0 )
desVar(   "I28_13"   0 )
desVar(   "I28_14"   0 )
desVar(   "I28_15"   0 )
desVar(   "I28_16"   0 )
desVar(   "I28_17"   0 )
desVar(   "I28_18"   0 )
desVar(   "I28_19"   0 )
desVar(   "I28_20"   0 )
desVar(   "I28_21"   0 )
desVar(   "I28_22"   0 )
desVar(   "I28_23"   0 )
desVar(   "I28_24"   0 )
desVar(   "I28_25"   20p )
desVar(   "I28_26"   0 )
desVar(   "I28_27"   0 )
desVar(   "I28_28"   0 )
desVar(   "I28_29"   0 )
desVar(   "I28_30"   0 )
desVar(   "I28_31"   0 )
desVar(   "I28_32"   0 )
desVar(   "I28_33"   0 )
desVar(   "I28_34"   0 )
desVar(   "I28_35"   0 )
desVar(   "I28_36"   0 )
desVar(   "I28_37"   0 )
desVar(   "I28_38"   0 )
desVar(   "I28_39"   0 )
desVar(   "I28_40"   0 )
desVar(   "I28_41"   0 )
desVar(   "I28_42"   0 )
desVar(   "I28_43"   0 )
desVar(   "I28_44"   0 )
desVar(   "I28_45"   0 )
```

```
desVar(   "I28_46"   0 )
desVar(   "I28_47"   0 )
desVar(   "I28_48"   0 )
desVar(   "I28_49"   0 )
desVar(   "I28_50"   0 )
desVar(   "I29_1"   0 )
desVar(   "I29_2"   0 )
desVar(   "I29_3"   0 )
desVar(   "I29_4"   0 )
desVar(   "I29_5"   0 )
desVar(   "I29_6"   0 )
desVar(   "I29_7"   0 )
desVar(   "I29_8"   0 )
desVar(   "I29_9"   0 )
desVar(   "I29_10"   0 )
desVar(   "I29_11"   0 )
desVar(   "I29_12"   0 )
desVar(   "I29_13"   0 )
desVar(   "I29_14"   0 )
desVar(   "I29_15"   0 )
desVar(   "I29_16"   0 )
desVar(   "I29_17"   0 )
desVar(   "I29_18"   0 )
desVar(   "I29_19"   0 )
desVar(   "I29_20"   0 )
desVar(   "I29_21"   0 )
desVar(   "I29_22"   0 )
desVar(   "I29_23"   0 )
desVar(   "I29_24"   0 )
desVar(   "I29_25"   20p )
desVar(   "I29_26"   0 )
desVar(   "I29_27"   0 )
desVar(   "I29_28"   0 )
desVar(   "I29_29"   0 )
desVar(   "I29_30"   0 )
desVar(   "I29_31"   0 )
desVar(   "I29_32"   0 )
desVar(   "I29_33"   0 )
desVar(   "I29_34"   0 )
```

```
desVar(   "I29_35"   0 )
desVar(   "I29_36"   0 )
desVar(   "I29_37"   0 )
desVar(   "I29_38"   0 )
desVar(   "I29_39"   0 )
desVar(   "I29_40"   0 )
desVar(   "I29_41"   0 )
desVar(   "I29_42"   0 )
desVar(   "I29_43"   0 )
desVar(   "I29_44"   0 )
desVar(   "I29_45"   0 )
desVar(   "I29_46"   0 )
desVar(   "I29_47"   0 )
desVar(   "I29_48"   0 )
desVar(   "I29_49"   0 )
desVar(   "I29_50"   0 )
desVar(   "I30_1"   0 )
desVar(   "I30_2"   0 )
desVar(   "I30_3"   0 )
desVar(   "I30_4"   0 )
desVar(   "I30_5"   0 )
desVar(   "I30_6"   0 )
desVar(   "I30_7"   0 )
desVar(   "I30_8"   0 )
desVar(   "I30_9"   0 )
desVar(   "I30_10"   0 )
desVar(   "I30_11"   0 )
desVar(   "I30_12"   0 )
desVar(   "I30_13"   0 )
desVar(   "I30_14"   0 )
desVar(   "I30_15"   0 )
desVar(   "I30_16"   0 )
desVar(   "I30_17"   0 )
desVar(   "I30_18"   0 )
desVar(   "I30_19"   0 )
desVar(   "I30_20"   0 )
desVar(   "I30_21"   0 )
desVar(   "I30_22"   0 )
desVar(   "I30_23"   0 )
```

```
desVar(   "I30_24"   0 )
desVar(   "I30_25"   20p )
desVar(   "I30_26"   0 )
desVar(   "I30_27"   0 )
desVar(   "I30_28"   0 )
desVar(   "I30_29"   0 )
desVar(   "I30_30"   0 )
desVar(   "I30_31"   0 )
desVar(   "I30_32"   0 )
desVar(   "I30_33"   0 )
desVar(   "I30_34"   0 )
desVar(   "I30_35"   0 )
desVar(   "I30_36"   0 )
desVar(   "I30_37"   0 )
desVar(   "I30_38"   0 )
desVar(   "I30_39"   0 )
desVar(   "I30_40"   0 )
desVar(   "I30_41"   0 )
desVar(   "I30_42"   0 )
desVar(   "I30_43"   0 )
desVar(   "I30_44"   0 )
desVar(   "I30_45"   0 )
desVar(   "I30_46"   0 )
desVar(   "I30_47"   0 )
desVar(   "I30_48"   0 )
desVar(   "I30_49"   0 )
desVar(   "I30_50"   0 )
desVar(   "I31_1"   0 )
desVar(   "I31_2"   0 )
desVar(   "I31_3"   0 )
desVar(   "I31_4"   0 )
desVar(   "I31_5"   0 )
desVar(   "I31_6"   0 )
desVar(   "I31_7"   0 )
desVar(   "I31_8"   0 )
desVar(   "I31_9"   0 )
desVar(   "I31_10"   0 )
desVar(   "I31_11"   0 )
desVar(   "I31_12"   0 )
```

```
desVar(   "I31_13"   0 )
desVar(   "I31_14"   0 )
desVar(   "I31_15"   0 )
desVar(   "I31_16"   0 )
desVar(   "I31_17"   0 )
desVar(   "I31_18"   0 )
desVar(   "I31_19"   0 )
desVar(   "I31_20"   0 )
desVar(   "I31_21"   0 )
desVar(   "I31_22"   0 )
desVar(   "I31_23"   0 )
desVar(   "I31_24"   0 )
desVar(   "I31_25"   20p )
desVar(   "I31_26"   0 )
desVar(   "I31_27"   0 )
desVar(   "I31_28"   0 )
desVar(   "I31_29"   0 )
desVar(   "I31_30"   0 )
desVar(   "I31_31"   0 )
desVar(   "I31_32"   0 )
desVar(   "I31_33"   0 )
desVar(   "I31_34"   0 )
desVar(   "I31_35"   0 )
desVar(   "I31_36"   0 )
desVar(   "I31_37"   0 )
desVar(   "I31_38"   0 )
desVar(   "I31_39"   0 )
desVar(   "I31_40"   0 )
desVar(   "I31_41"   0 )
desVar(   "I31_42"   0 )
desVar(   "I31_43"   0 )
desVar(   "I31_44"   0 )
desVar(   "I31_45"   0 )
desVar(   "I31_46"   0 )
desVar(   "I31_47"   0 )
desVar(   "I31_48"   0 )
desVar(   "I31_49"   0 )
desVar(   "I31_50"   0 )
desVar(   "I32_1"   0 )
```

```
desVar(   "I32_2"   0 )
desVar(   "I32_3"   0 )
desVar(   "I32_4"   0 )
desVar(   "I32_5"   0 )
desVar(   "I32_6"   0 )
desVar(   "I32_7"   0 )
desVar(   "I32_8"   0 )
desVar(   "I32_9"   0 )
desVar(   "I32_10"   0 )
desVar(   "I32_11"   0 )
desVar(   "I32_12"   0 )
desVar(   "I32_13"   0 )
desVar(   "I32_14"   0 )
desVar(   "I32_15"   0 )
desVar(   "I32_16"   0 )
desVar(   "I32_17"   0 )
desVar(   "I32_18"   0 )
desVar(   "I32_19"   0 )
desVar(   "I32_20"   0 )
desVar(   "I32_21"   0 )
desVar(   "I32_22"   0 )
desVar(   "I32_23"   0 )
desVar(   "I32_24"   0 )
desVar(   "I32_25"   20p )
desVar(   "I32_26"   0 )
desVar(   "I32_27"   0 )
desVar(   "I32_28"   0 )
desVar(   "I32_29"   0 )
desVar(   "I32_30"   0 )
desVar(   "I32_31"   0 )
desVar(   "I32_32"   0 )
desVar(   "I32_33"   0 )
desVar(   "I32_34"   0 )
desVar(   "I32_35"   0 )
desVar(   "I32_36"   0 )
desVar(   "I32_37"   0 )
desVar(   "I32_38"   0 )
desVar(   "I32_39"   0 )
desVar(   "I32_40"   0 )
```

```
desVar(   "I32_41"   0 )
desVar(   "I32_42"   0 )
desVar(   "I32_43"   0 )
desVar(   "I32_44"   0 )
desVar(   "I32_45"   0 )
desVar(   "I32_46"   0 )
desVar(   "I32_47"   0 )
desVar(   "I32_48"   0 )
desVar(   "I32_49"   0 )
desVar(   "I32_50"   0 )
desVar(   "I33_1"   0 )
desVar(   "I33_2"   0 )
desVar(   "I33_3"   0 )
desVar(   "I33_4"   0 )
desVar(   "I33_5"   0 )
desVar(   "I33_6"   0 )
desVar(   "I33_7"   0 )
desVar(   "I33_8"   0 )
desVar(   "I33_9"   0 )
desVar(   "I33_10"   0 )
desVar(   "I33_11"   0 )
desVar(   "I33_12"   0 )
desVar(   "I33_13"   0 )
desVar(   "I33_14"   0 )
desVar(   "I33_15"   0 )
desVar(   "I33_16"   0 )
desVar(   "I33_17"   0 )
desVar(   "I33_18"   0 )
desVar(   "I33_19"   0 )
desVar(   "I33_20"   0 )
desVar(   "I33_21"   0 )
desVar(   "I33_22"   0 )
desVar(   "I33_23"   0 )
desVar(   "I33_24"   0 )
desVar(   "I33_25"   20p )
desVar(   "I33_26"   0 )
desVar(   "I33_27"   0 )
desVar(   "I33_28"   0 )
desVar(   "I33_29"   0 )
```

```
desVar(   "I33_30"   0 )
desVar(   "I33_31"   0 )
desVar(   "I33_32"   0 )
desVar(   "I33_33"   0 )
desVar(   "I33_34"   0 )
desVar(   "I33_35"   0 )
desVar(   "I33_36"   0 )
desVar(   "I33_37"   0 )
desVar(   "I33_38"   0 )
desVar(   "I33_39"   0 )
desVar(   "I33_40"   0 )
desVar(   "I33_41"   0 )
desVar(   "I33_42"   0 )
desVar(   "I33_43"   0 )
desVar(   "I33_44"   0 )
desVar(   "I33_45"   0 )
desVar(   "I33_46"   0 )
desVar(   "I33_47"   0 )
desVar(   "I33_48"   0 )
desVar(   "I33_49"   0 )
desVar(   "I33_50"   0 )
desVar(   "I34_1"   0 )
desVar(   "I34_2"   0 )
desVar(   "I34_3"   0 )
desVar(   "I34_4"   0 )
desVar(   "I34_5"   0 )
desVar(   "I34_6"   0 )
desVar(   "I34_7"   0 )
desVar(   "I34_8"   0 )
desVar(   "I34_9"   0 )
desVar(   "I34_10"   0 )
desVar(   "I34_11"   0 )
desVar(   "I34_12"   0 )
desVar(   "I34_13"   0 )
desVar(   "I34_14"   0 )
desVar(   "I34_15"   0 )
desVar(   "I34_16"   0 )
desVar(   "I34_17"   0 )
desVar(   "I34_18"   0 )
```

```
desVar(   "I34_19"   0 )
desVar(   "I34_20"   0 )
desVar(   "I34_21"   0 )
desVar(   "I34_22"   0 )
desVar(   "I34_23"   0 )
desVar(   "I34_24"   0 )
desVar(   "I34_25"   20p )
desVar(   "I34_26"   0 )
desVar(   "I34_27"   0 )
desVar(   "I34_28"   0 )
desVar(   "I34_29"   0 )
desVar(   "I34_30"   0 )
desVar(   "I34_31"   0 )
desVar(   "I34_32"   0 )
desVar(   "I34_33"   0 )
desVar(   "I34_34"   0 )
desVar(   "I34_35"   0 )
desVar(   "I34_36"   0 )
desVar(   "I34_37"   0 )
desVar(   "I34_38"   0 )
desVar(   "I34_39"   0 )
desVar(   "I34_40"   0 )
desVar(   "I34_41"   0 )
desVar(   "I34_42"   0 )
desVar(   "I34_43"   0 )
desVar(   "I34_44"   0 )
desVar(   "I34_45"   0 )
desVar(   "I34_46"   0 )
desVar(   "I34_47"   0 )
desVar(   "I34_48"   0 )
desVar(   "I34_49"   0 )
desVar(   "I34_50"   0 )
desVar(   "I35_1"   0 )
desVar(   "I35_2"   0 )
desVar(   "I35_3"   0 )
desVar(   "I35_4"   0 )
desVar(   "I35_5"   0 )
desVar(   "I35_6"   0 )
desVar(   "I35_7"   0 )
```

```
desVar(   "I35_8"   0 )
desVar(   "I35_9"   0 )
desVar(   "I35_10"   0 )
desVar(   "I35_11"   0 )
desVar(   "I35_12"   0 )
desVar(   "I35_13"   0 )
desVar(   "I35_14"   0 )
desVar(   "I35_15"   0 )
desVar(   "I35_16"   0 )
desVar(   "I35_17"   0 )
desVar(   "I35_18"   0 )
desVar(   "I35_19"   0 )
desVar(   "I35_20"   0 )
desVar(   "I35_21"   0 )
desVar(   "I35_22"   0 )
desVar(   "I35_23"   0 )
desVar(   "I35_24"   0 )
desVar(   "I35_25"   20p )
desVar(   "I35_26"   0 )
desVar(   "I35_27"   0 )
desVar(   "I35_28"   0 )
desVar(   "I35_29"   0 )
desVar(   "I35_30"   0 )
desVar(   "I35_31"   0 )
desVar(   "I35_32"   0 )
desVar(   "I35_33"   0 )
desVar(   "I35_34"   0 )
desVar(   "I35_35"   0 )
desVar(   "I35_36"   0 )
desVar(   "I35_37"   0 )
desVar(   "I35_38"   0 )
desVar(   "I35_39"   0 )
desVar(   "I35_40"   0 )
desVar(   "I35_41"   0 )
desVar(   "I35_42"   0 )
desVar(   "I35_43"   0 )
desVar(   "I35_44"   0 )
desVar(   "I35_45"   0 )
desVar(   "I35_46"   0 )
```

```
desVar(   "I35_47"   0 )
desVar(   "I35_48"   0 )
desVar(   "I35_49"   0 )
desVar(   "I35_50"   0 )
desVar(   "I36_1"   0 )
desVar(   "I36_2"   0 )
desVar(   "I36_3"   0 )
desVar(   "I36_4"   0 )
desVar(   "I36_5"   0 )
desVar(   "I36_6"   0 )
desVar(   "I36_7"   0 )
desVar(   "I36_8"   0 )
desVar(   "I36_9"   0 )
desVar(   "I36_10"   0 )
desVar(   "I36_11"   0 )
desVar(   "I36_12"   0 )
desVar(   "I36_13"   0 )
desVar(   "I36_14"   0 )
desVar(   "I36_15"   0 )
desVar(   "I36_16"   0 )
desVar(   "I36_17"   0 )
desVar(   "I36_18"   0 )
desVar(   "I36_19"   0 )
desVar(   "I36_20"   0 )
desVar(   "I36_21"   0 )
desVar(   "I36_22"   0 )
desVar(   "I36_23"   0 )
desVar(   "I36_24"   0 )
desVar(   "I36_25"   20p )
desVar(   "I36_26"   0 )
desVar(   "I36_27"   0 )
desVar(   "I36_28"   0 )
desVar(   "I36_29"   0 )
desVar(   "I36_30"   0 )
desVar(   "I36_31"   0 )
desVar(   "I36_32"   0 )
desVar(   "I36_33"   0 )
desVar(   "I36_34"   0 )
desVar(   "I36_35"   0 )
```

```
desVar(   "I36_36"   0 )
desVar(   "I36_37"   0 )
desVar(   "I36_38"   0 )
desVar(   "I36_39"   0 )
desVar(   "I36_40"   0 )
desVar(   "I36_41"   0 )
desVar(   "I36_42"   0 )
desVar(   "I36_43"   0 )
desVar(   "I36_44"   0 )
desVar(   "I36_45"   0 )
desVar(   "I36_46"   0 )
desVar(   "I36_47"   0 )
desVar(   "I36_48"   0 )
desVar(   "I36_49"   0 )
desVar(   "I36_50"   0 )
desVar(   "I37_1"   0 )
desVar(   "I37_2"   0 )
desVar(   "I37_3"   0 )
desVar(   "I37_4"   0 )
desVar(   "I37_5"   0 )
desVar(   "I37_6"   0 )
desVar(   "I37_7"   0 )
desVar(   "I37_8"   0 )
desVar(   "I37_9"   0 )
desVar(   "I37_10"   0 )
desVar(   "I37_11"   0 )
desVar(   "I37_12"   0 )
desVar(   "I37_13"   0 )
desVar(   "I37_14"   0 )
desVar(   "I37_15"   0 )
desVar(   "I37_16"   0 )
desVar(   "I37_17"   0 )
desVar(   "I37_18"   0 )
desVar(   "I37_19"   0 )
desVar(   "I37_20"   0 )
desVar(   "I37_21"   0 )
desVar(   "I37_22"   0 )
desVar(   "I37_23"   0 )
desVar(   "I37_24"   0 )
```

```
desVar(   "I37_25"   20p )
desVar(   "I37_26"   0 )
desVar(   "I37_27"   0 )
desVar(   "I37_28"   0 )
desVar(   "I37_29"   0 )
desVar(   "I37_30"   0 )
desVar(   "I37_31"   0 )
desVar(   "I37_32"   0 )
desVar(   "I37_33"   0 )
desVar(   "I37_34"   0 )
desVar(   "I37_35"   0 )
desVar(   "I37_36"   0 )
desVar(   "I37_37"   0 )
desVar(   "I37_38"   0 )
desVar(   "I37_39"   0 )
desVar(   "I37_40"   0 )
desVar(   "I37_41"   0 )
desVar(   "I37_42"   0 )
desVar(   "I37_43"   0 )
desVar(   "I37_44"   0 )
desVar(   "I37_45"   0 )
desVar(   "I37_46"   0 )
desVar(   "I37_47"   0 )
desVar(   "I37_48"   0 )
desVar(   "I37_49"   0 )
desVar(   "I37_50"   0 )
desVar(   "I38_1"   0 )
desVar(   "I38_2"   0 )
desVar(   "I38_3"   0 )
desVar(   "I38_4"   0 )
desVar(   "I38_5"   0 )
desVar(   "I38_6"   0 )
desVar(   "I38_7"   0 )
desVar(   "I38_8"   0 )
desVar(   "I38_9"   0 )
desVar(   "I38_10"   0 )
desVar(   "I38_11"   0 )
desVar(   "I38_12"   0 )
desVar(   "I38_13"   0 )
```

```
desVar(   "I38_14"   0 )
desVar(   "I38_15"   0 )
desVar(   "I38_16"   0 )
desVar(   "I38_17"   0 )
desVar(   "I38_18"   0 )
desVar(   "I38_19"   0 )
desVar(   "I38_20"   0 )
desVar(   "I38_21"   0 )
desVar(   "I38_22"   0 )
desVar(   "I38_23"   0 )
desVar(   "I38_24"   0 )
desVar(   "I38_25"   20p )
desVar(   "I38_26"   0 )
desVar(   "I38_27"   0 )
desVar(   "I38_28"   0 )
desVar(   "I38_29"   0 )
desVar(   "I38_30"   0 )
desVar(   "I38_31"   0 )
desVar(   "I38_32"   0 )
desVar(   "I38_33"   0 )
desVar(   "I38_34"   0 )
desVar(   "I38_35"   0 )
desVar(   "I38_36"   0 )
desVar(   "I38_37"   0 )
desVar(   "I38_38"   0 )
desVar(   "I38_39"   0 )
desVar(   "I38_40"   0 )
desVar(   "I38_41"   0 )
desVar(   "I38_42"   0 )
desVar(   "I38_43"   0 )
desVar(   "I38_44"   0 )
desVar(   "I38_45"   0 )
desVar(   "I38_46"   0 )
desVar(   "I38_47"   0 )
desVar(   "I38_48"   0 )
desVar(   "I38_49"   0 )
desVar(   "I38_50"   0 )
desVar(   "I39_1"   0 )
desVar(   "I39_2"   0 )
```

```
desVar(   "I39_3"   0 )
desVar(   "I39_4"   0 )
desVar(   "I39_5"   0 )
desVar(   "I39_6"   0 )
desVar(   "I39_7"   0 )
desVar(   "I39_8"   0 )
desVar(   "I39_9"   0 )
desVar(   "I39_10"   0 )
desVar(   "I39_11"   0 )
desVar(   "I39_12"   0 )
desVar(   "I39_13"   0 )
desVar(   "I39_14"   0 )
desVar(   "I39_15"   0 )
desVar(   "I39_16"   0 )
desVar(   "I39_17"   0 )
desVar(   "I39_18"   0 )
desVar(   "I39_19"   0 )
desVar(   "I39_20"   0 )
desVar(   "I39_21"   0 )
desVar(   "I39_22"   0 )
desVar(   "I39_23"   0 )
desVar(   "I39_24"   0 )
desVar(   "I39_25"   0 )
desVar(   "I39_26"   0 )
desVar(   "I39_27"   0 )
desVar(   "I39_28"   0 )
desVar(   "I39_29"   0 )
desVar(   "I39_30"   0 )
desVar(   "I39_31"   0 )
desVar(   "I39_32"   0 )
desVar(   "I39_33"   0 )
desVar(   "I39_34"   0 )
desVar(   "I39_35"   0 )
desVar(   "I39_36"   0 )
desVar(   "I39_37"   0 )
desVar(   "I39_38"   0 )
desVar(   "I39_39"   0 )
desVar(   "I39_40"   0 )
desVar(   "I39_41"   0 )
```

```
desVar(   "I39_42"   0 )
desVar(   "I39_43"   0 )
desVar(   "I39_44"   0 )
desVar(   "I39_45"   0 )
desVar(   "I39_46"   0 )
desVar(   "I39_47"   0 )
desVar(   "I39_48"   0 )
desVar(   "I39_49"   0 )
desVar(   "I39_50"   0 )
desVar(   "I40_1"   0 )
desVar(   "I40_2"   0 )
desVar(   "I40_3"   0 )
desVar(   "I40_4"   0 )
desVar(   "I40_5"   0 )
desVar(   "I40_6"   0 )
desVar(   "I40_7"   0 )
desVar(   "I40_8"   0 )
desVar(   "I40_9"   0 )
desVar(   "I40_10"   0 )
desVar(   "I40_11"   0 )
desVar(   "I40_12"   0 )
desVar(   "I40_13"   0 )
desVar(   "I40_14"   0 )
desVar(   "I40_15"   0 )
desVar(   "I40_16"   0 )
desVar(   "I40_17"   0 )
desVar(   "I40_18"   0 )
desVar(   "I40_19"   0 )
desVar(   "I40_20"   0 )
desVar(   "I40_21"   0 )
desVar(   "I40_22"   0 )
desVar(   "I40_23"   0 )
desVar(   "I40_24"   0 )
desVar(   "I40_25"   0 )
desVar(   "I40_26"   0 )
desVar(   "I40_27"   0 )
desVar(   "I40_28"   0 )
desVar(   "I40_29"   0 )
desVar(   "I40_30"   0 )
```

```
desVar(   "I40_31"   0 )
desVar(   "I40_32"   0 )
desVar(   "I40_33"   0 )
desVar(   "I40_34"   0 )
desVar(   "I40_35"   0 )
desVar(   "I40_36"   0 )
desVar(   "I40_37"   0 )
desVar(   "I40_38"   0 )
desVar(   "I40_39"   0 )
desVar(   "I40_40"   0 )
desVar(   "I40_41"   0 )
desVar(   "I40_42"   0 )
desVar(   "I40_43"   0 )
desVar(   "I40_44"   0 )
desVar(   "I40_45"   0 )
desVar(   "I40_46"   0 )
desVar(   "I40_47"   0 )
desVar(   "I40_48"   0 )
desVar(   "I40_49"   0 )
desVar(   "I40_50"   0 )
desVar(   "I41_1"   0 )
desVar(   "I41_2"   0 )
desVar(   "I41_3"   0 )
desVar(   "I41_4"   0 )
desVar(   "I41_5"   0 )
desVar(   "I41_6"   0 )
desVar(   "I41_7"   0 )
desVar(   "I41_8"   0 )
desVar(   "I41_9"   0 )
desVar(   "I41_10"   0 )
desVar(   "I41_11"   0 )
desVar(   "I41_12"   0 )
desVar(   "I41_13"   0 )
desVar(   "I41_14"   0 )
desVar(   "I41_15"   0 )
desVar(   "I41_16"   0 )
desVar(   "I41_17"   0 )
desVar(   "I41_18"   0 )
desVar(   "I41_19"   0 )
```

```
desVar(   "I41_20"   0 )
desVar(   "I41_21"   0 )
desVar(   "I41_22"   0 )
desVar(   "I41_23"   0 )
desVar(   "I41_24"   0 )
desVar(   "I41_25"   0 )
desVar(   "I41_26"   0 )
desVar(   "I41_27"   0 )
desVar(   "I41_28"   0 )
desVar(   "I41_29"   0 )
desVar(   "I41_30"   0 )
desVar(   "I41_31"   0 )
desVar(   "I41_32"   0 )
desVar(   "I41_33"   0 )
desVar(   "I41_34"   0 )
desVar(   "I41_35"   0 )
desVar(   "I41_36"   0 )
desVar(   "I41_37"   0 )
desVar(   "I41_38"   0 )
desVar(   "I41_39"   0 )
desVar(   "I41_40"   0 )
desVar(   "I41_41"   0 )
desVar(   "I41_42"   0 )
desVar(   "I41_43"   0 )
desVar(   "I41_44"   0 )
desVar(   "I41_45"   0 )
desVar(   "I41_46"   0 )
desVar(   "I41_47"   0 )
desVar(   "I41_48"   0 )
desVar(   "I41_49"   0 )
desVar(   "I41_50"   0 )
desVar(   "I42_1"   0 )
desVar(   "I42_2"   0 )
desVar(   "I42_3"   0 )
desVar(   "I42_4"   0 )
desVar(   "I42_5"   0 )
desVar(   "I42_6"   0 )
desVar(   "I42_7"   0 )
desVar(   "I42_8"   0 )
```

```
desVar(   "I42_9"   0 )
desVar(   "I42_10"   0 )
desVar(   "I42_11"   0 )
desVar(   "I42_12"   0 )
desVar(   "I42_13"   0 )
desVar(   "I42_14"   0 )
desVar(   "I42_15"   0 )
desVar(   "I42_16"   0 )
desVar(   "I42_17"   0 )
desVar(   "I42_18"   0 )
desVar(   "I42_19"   0 )
desVar(   "I42_20"   0 )
desVar(   "I42_21"   0 )
desVar(   "I42_22"   0 )
desVar(   "I42_23"   0 )
desVar(   "I42_24"   0 )
desVar(   "I42_25"   0 )
desVar(   "I42_26"   0 )
desVar(   "I42_27"   0 )
desVar(   "I42_28"   0 )
desVar(   "I42_29"   0 )
desVar(   "I42_30"   0 )
desVar(   "I42_31"   0 )
desVar(   "I42_32"   0 )
desVar(   "I42_33"   0 )
desVar(   "I42_34"   0 )
desVar(   "I42_35"   0 )
desVar(   "I42_36"   0 )
desVar(   "I42_37"   0 )
desVar(   "I42_38"   0 )
desVar(   "I42_39"   0 )
desVar(   "I42_40"   0 )
desVar(   "I42_41"   0 )
desVar(   "I42_42"   0 )
desVar(   "I42_43"   0 )
desVar(   "I42_44"   0 )
desVar(   "I42_45"   0 )
desVar(   "I42_46"   0 )
desVar(   "I42_47"   0 )
```

```
desVar(   "I42_48"   0 )
desVar(   "I42_49"   0 )
desVar(   "I42_50"   0 )
desVar(   "I43_1"   0 )
desVar(   "I43_2"   0 )
desVar(   "I43_3"   0 )
desVar(   "I43_4"   0 )
desVar(   "I43_5"   0 )
desVar(   "I43_6"   0 )
desVar(   "I43_7"   0 )
desVar(   "I43_8"   0 )
desVar(   "I43_9"   0 )
desVar(   "I43_10"   0 )
desVar(   "I43_11"   0 )
desVar(   "I43_12"   0 )
desVar(   "I43_13"   0 )
desVar(   "I43_14"   0 )
desVar(   "I43_15"   0 )
desVar(   "I43_16"   0 )
desVar(   "I43_17"   0 )
desVar(   "I43_18"   0 )
desVar(   "I43_19"   0 )
desVar(   "I43_20"   0 )
desVar(   "I43_21"   0 )
desVar(   "I43_22"   0 )
desVar(   "I43_23"   0 )
desVar(   "I43_24"   0 )
desVar(   "I43_25"   0 )
desVar(   "I43_26"   0 )
desVar(   "I43_27"   0 )
desVar(   "I43_28"   0 )
desVar(   "I43_29"   0 )
desVar(   "I43_30"   0 )
desVar(   "I43_31"   0 )
desVar(   "I43_32"   0 )
desVar(   "I43_33"   0 )
desVar(   "I43_34"   0 )
desVar(   "I43_35"   0 )
desVar(   "I43_36"   0 )
```

```
desVar(   "I43_37"   0 )
desVar(   "I43_38"   0 )
desVar(   "I43_39"   0 )
desVar(   "I43_40"   0 )
desVar(   "I43_41"   0 )
desVar(   "I43_42"   0 )
desVar(   "I43_43"   0 )
desVar(   "I43_44"   0 )
desVar(   "I43_45"   0 )
desVar(   "I43_46"   0 )
desVar(   "I43_47"   0 )
desVar(   "I43_48"   0 )
desVar(   "I43_49"   0 )
desVar(   "I43_50"   0 )
desVar(   "I44_1"   0 )
desVar(   "I44_2"   0 )
desVar(   "I44_3"   0 )
desVar(   "I44_4"   0 )
desVar(   "I44_5"   0 )
desVar(   "I44_6"   0 )
desVar(   "I44_7"   0 )
desVar(   "I44_8"   0 )
desVar(   "I44_9"   0 )
desVar(   "I44_10"   0 )
desVar(   "I44_11"   0 )
desVar(   "I44_12"   0 )
desVar(   "I44_13"   0 )
desVar(   "I44_14"   0 )
desVar(   "I44_15"   0 )
desVar(   "I44_16"   0 )
desVar(   "I44_17"   0 )
desVar(   "I44_18"   0 )
desVar(   "I44_19"   0 )
desVar(   "I44_20"   0 )
desVar(   "I44_21"   0 )
desVar(   "I44_22"   0 )
desVar(   "I44_23"   0 )
desVar(   "I44_24"   0 )
desVar(   "I44_25"   0 )
```

```
desVar(   "I44_26"   0 )
desVar(   "I44_27"   0 )
desVar(   "I44_28"   0 )
desVar(   "I44_29"   0 )
desVar(   "I44_30"   0 )
desVar(   "I44_31"   0 )
desVar(   "I44_32"   0 )
desVar(   "I44_33"   0 )
desVar(   "I44_34"   0 )
desVar(   "I44_35"   0 )
desVar(   "I44_36"   0 )
desVar(   "I44_37"   0 )
desVar(   "I44_38"   0 )
desVar(   "I44_39"   0 )
desVar(   "I44_40"   0 )
desVar(   "I44_41"   0 )
desVar(   "I44_42"   0 )
desVar(   "I44_43"   0 )
desVar(   "I44_44"   0 )
desVar(   "I44_45"   0 )
desVar(   "I44_46"   0 )
desVar(   "I44_47"   0 )
desVar(   "I44_48"   0 )
desVar(   "I44_49"   0 )
desVar(   "I44_50"   0 )
desVar(   "I45_1"   0 )
desVar(   "I45_2"   0 )
desVar(   "I45_3"   0 )
desVar(   "I45_4"   0 )
desVar(   "I45_5"   0 )
desVar(   "I45_6"   0 )
desVar(   "I45_7"   0 )
desVar(   "I45_8"   0 )
desVar(   "I45_9"   0 )
desVar(   "I45_10"   0 )
desVar(   "I45_11"   0 )
desVar(   "I45_12"   0 )
desVar(   "I45_13"   0 )
desVar(   "I45_14"   0 )
```

```
desVar(   "I45_15"   0 )
desVar(   "I45_16"   0 )
desVar(   "I45_17"   0 )
desVar(   "I45_18"   0 )
desVar(   "I45_19"   0 )
desVar(   "I45_20"   0 )
desVar(   "I45_21"   0 )
desVar(   "I45_22"   0 )
desVar(   "I45_23"   0 )
desVar(   "I45_24"   0 )
desVar(   "I45_25"   0 )
desVar(   "I45_26"   0 )
desVar(   "I45_27"   0 )
desVar(   "I45_28"   0 )
desVar(   "I45_29"   0 )
desVar(   "I45_30"   0 )
desVar(   "I45_31"   0 )
desVar(   "I45_32"   0 )
desVar(   "I45_33"   0 )
desVar(   "I45_34"   0 )
desVar(   "I45_35"   0 )
desVar(   "I45_36"   0 )
desVar(   "I45_37"   0 )
desVar(   "I45_38"   0 )
desVar(   "I45_39"   0 )
desVar(   "I45_40"   0 )
desVar(   "I45_41"   0 )
desVar(   "I45_42"   0 )
desVar(   "I45_43"   0 )
desVar(   "I45_44"   0 )
desVar(   "I45_45"   0 )
desVar(   "I45_46"   0 )
desVar(   "I45_47"   0 )
desVar(   "I45_48"   0 )
desVar(   "I45_49"   0 )
desVar(   "I45_50"   0 )
desVar(   "I46_1"   0 )
desVar(   "I46_2"   0 )
desVar(   "I46_3"   0 )
```

```
desVar(   "I46_4"   0 )
desVar(   "I46_5"   0 )
desVar(   "I46_6"   0 )
desVar(   "I46_7"   0 )
desVar(   "I46_8"   0 )
desVar(   "I46_9"   0 )
desVar(   "I46_10"   0 )
desVar(   "I46_11"   0 )
desVar(   "I46_12"   0 )
desVar(   "I46_13"   0 )
desVar(   "I46_14"   0 )
desVar(   "I46_15"   0 )
desVar(   "I46_16"   0 )
desVar(   "I46_17"   0 )
desVar(   "I46_18"   0 )
desVar(   "I46_19"   0 )
desVar(   "I46_20"   0 )
desVar(   "I46_21"   0 )
desVar(   "I46_22"   0 )
desVar(   "I46_23"   0 )
desVar(   "I46_24"   0 )
desVar(   "I46_25"   0 )
desVar(   "I46_26"   0 )
desVar(   "I46_27"   0 )
desVar(   "I46_28"   0 )
desVar(   "I46_29"   0 )
desVar(   "I46_30"   0 )
desVar(   "I46_31"   0 )
desVar(   "I46_32"   0 )
desVar(   "I46_33"   0 )
desVar(   "I46_34"   0 )
desVar(   "I46_35"   0 )
desVar(   "I46_36"   0 )
desVar(   "I46_37"   0 )
desVar(   "I46_38"   0 )
desVar(   "I46_39"   0 )
desVar(   "I46_40"   0 )
desVar(   "I46_41"   0 )
desVar(   "I46_42"   0 )
```

```
desVar(   "I46_43"   0 )
desVar(   "I46_44"   0 )
desVar(   "I46_45"   0 )
desVar(   "I46_46"   0 )
desVar(   "I46_47"   0 )
desVar(   "I46_48"   0 )
desVar(   "I46_49"   0 )
desVar(   "I46_50"   0 )
desVar(   "I47_1"   0 )
desVar(   "I47_2"   0 )
desVar(   "I47_3"   0 )
desVar(   "I47_4"   0 )
desVar(   "I47_5"   0 )
desVar(   "I47_6"   0 )
desVar(   "I47_7"   0 )
desVar(   "I47_8"   0 )
desVar(   "I47_9"   0 )
desVar(   "I47_10"   0 )
desVar(   "I47_11"   0 )
desVar(   "I47_12"   0 )
desVar(   "I47_13"   0 )
desVar(   "I47_14"   0 )
desVar(   "I47_15"   0 )
desVar(   "I47_16"   0 )
desVar(   "I47_17"   0 )
desVar(   "I47_18"   0 )
desVar(   "I47_19"   0 )
desVar(   "I47_20"   0 )
desVar(   "I47_21"   0 )
desVar(   "I47_22"   0 )
desVar(   "I47_23"   0 )
desVar(   "I47_24"   0 )
desVar(   "I47_25"   0 )
desVar(   "I47_26"   0 )
desVar(   "I47_27"   0 )
desVar(   "I47_28"   0 )
desVar(   "I47_29"   0 )
desVar(   "I47_30"   0 )
desVar(   "I47_31"   0 )
```

```
desVar(   "I47_32"   0 )
desVar(   "I47_33"   0 )
desVar(   "I47_34"   0 )
desVar(   "I47_35"   0 )
desVar(   "I47_36"   0 )
desVar(   "I47_37"   0 )
desVar(   "I47_38"   0 )
desVar(   "I47_39"   0 )
desVar(   "I47_40"   0 )
desVar(   "I47_41"   0 )
desVar(   "I47_42"   0 )
desVar(   "I47_43"   0 )
desVar(   "I47_44"   0 )
desVar(   "I47_45"   0 )
desVar(   "I47_46"   0 )
desVar(   "I47_47"   0 )
desVar(   "I47_48"   0 )
desVar(   "I47_49"   0 )
desVar(   "I47_50"   0 )
desVar(   "I48_1"   0 )
desVar(   "I48_2"   0 )
desVar(   "I48_3"   0 )
desVar(   "I48_4"   0 )
desVar(   "I48_5"   0 )
desVar(   "I48_6"   0 )
desVar(   "I48_7"   0 )
desVar(   "I48_8"   0 )
desVar(   "I48_9"   0 )
desVar(   "I48_10"   0 )
desVar(   "I48_11"   0 )
desVar(   "I48_12"   0 )
desVar(   "I48_13"   0 )
desVar(   "I48_14"   0 )
desVar(   "I48_15"   0 )
desVar(   "I48_16"   0 )
desVar(   "I48_17"   0 )
desVar(   "I48_18"   0 )
desVar(   "I48_19"   0 )
desVar(   "I48_20"   0 )
```

```
desVar(   "I48_21"   0 )
desVar(   "I48_22"   0 )
desVar(   "I48_23"   0 )
desVar(   "I48_24"   0 )
desVar(   "I48_25"   0 )
desVar(   "I48_26"   0 )
desVar(   "I48_27"   0 )
desVar(   "I48_28"   0 )
desVar(   "I48_29"   0 )
desVar(   "I48_30"   0 )
desVar(   "I48_31"   0 )
desVar(   "I48_32"   0 )
desVar(   "I48_33"   0 )
desVar(   "I48_34"   0 )
desVar(   "I48_35"   0 )
desVar(   "I48_36"   0 )
desVar(   "I48_37"   0 )
desVar(   "I48_38"   0 )
desVar(   "I48_39"   0 )
desVar(   "I48_40"   0 )
desVar(   "I48_41"   0 )
desVar(   "I48_42"   0 )
desVar(   "I48_43"   0 )
desVar(   "I48_44"   0 )
desVar(   "I48_45"   0 )
desVar(   "I48_46"   0 )
desVar(   "I48_47"   0 )
desVar(   "I48_48"   0 )
desVar(   "I48_49"   0 )
desVar(   "I48_50"   0 )
desVar(   "I49_1"   0 )
desVar(   "I49_2"   0 )
desVar(   "I49_3"   0 )
desVar(   "I49_4"   0 )
desVar(   "I49_5"   0 )
desVar(   "I49_6"   0 )
desVar(   "I49_7"   0 )
desVar(   "I49_8"   0 )
desVar(   "I49_9"   0 )
```

```
desVar(   "I49_10"   0 )
desVar(   "I49_11"   0 )
desVar(   "I49_12"   0 )
desVar(   "I49_13"   0 )
desVar(   "I49_14"   0 )
desVar(   "I49_15"   0 )
desVar(   "I49_16"   0 )
desVar(   "I49_17"   0 )
desVar(   "I49_18"   0 )
desVar(   "I49_19"   0 )
desVar(   "I49_20"   0 )
desVar(   "I49_21"   0 )
desVar(   "I49_22"   0 )
desVar(   "I49_23"   0 )
desVar(   "I49_24"   0 )
desVar(   "I49_25"   0 )
desVar(   "I49_26"   0 )
desVar(   "I49_27"   0 )
desVar(   "I49_28"   0 )
desVar(   "I49_29"   0 )
desVar(   "I49_30"   0 )
desVar(   "I49_31"   0 )
desVar(   "I49_32"   0 )
desVar(   "I49_33"   0 )
desVar(   "I49_34"   0 )
desVar(   "I49_35"   0 )
desVar(   "I49_36"   0 )
desVar(   "I49_37"   0 )
desVar(   "I49_38"   0 )
desVar(   "I49_39"   0 )
desVar(   "I49_40"   0 )
desVar(   "I49_41"   0 )
desVar(   "I49_42"   0 )
desVar(   "I49_43"   0 )
desVar(   "I49_44"   0 )
desVar(   "I49_45"   0 )
desVar(   "I49_46"   0 )
desVar(   "I49_47"   0 )
desVar(   "I49_48"   0 )
```

```
desVar(   "I49_49"   0 )
desVar(   "I49_50"   0 )
desVar(   "I50_1"   0 )
desVar(   "I50_2"   0 )
desVar(   "I50_3"   0 )
desVar(   "I50_4"   0 )
desVar(   "I50_5"   0 )
desVar(   "I50_6"   0 )
desVar(   "I50_7"   0 )
desVar(   "I50_8"   0 )
desVar(   "I50_9"   0 )
desVar(   "I50_10"   0 )
desVar(   "I50_11"   0 )
desVar(   "I50_12"   0 )
desVar(   "I50_13"   0 )
desVar(   "I50_14"   0 )
desVar(   "I50_15"   0 )
desVar(   "I50_16"   0 )
desVar(   "I50_17"   0 )
desVar(   "I50_18"   0 )
desVar(   "I50_19"   0 )
desVar(   "I50_20"   0 )
desVar(   "I50_21"   0 )
desVar(   "I50_22"   0 )
desVar(   "I50_23"   0 )
desVar(   "I50_24"   0 )
desVar(   "I50_25"   0 )
desVar(   "I50_26"   0 )
desVar(   "I50_27"   0 )
desVar(   "I50_28"   0 )
desVar(   "I50_29"   0 )
desVar(   "I50_30"   0 )
desVar(   "I50_31"   0 )
desVar(   "I50_32"   0 )
desVar(   "I50_33"   0 )
desVar(   "I50_34"   0 )
desVar(   "I50_35"   0 )
desVar(   "I50_36"   0 )
desVar(   "I50_37"   0 )
```

```
desVar(   "I50_38"   0 )
desVar(   "I50_39"   0 )
desVar(   "I50_40"   0 )
desVar(   "I50_41"   0 )
desVar(   "I50_42"   0 )
desVar(   "I50_43"   0 )
desVar(   "I50_44"   0 )
desVar(   "I50_45"   0 )
desVar(   "I50_46"   0 )
desVar(   "I50_47"   0 )
desVar(   "I50_48"   0 )
desVar(   "I50_49"   0 )
desVar(   "I50_50"   0 )

desVar(   "Vbias" 2.7 )
desVar(   "Vu"  2.2 )
desVar(   "vss" 220m )
desVar(   "Vrhb" 2.35 )
desVar(   "vbq"  2.45  )
desVar(   "vr"  2.285 )
desVar(   "Vbiash" 2.65  )
desVar(   "Vrcb"  2.85 )
desVar(   "Pref"  2.2 )

option(    'error  "yes"
    'warn  "yes"
    'opptcheck  "yes"
    'topcheck  "full"
    'gmin_check  "all"
)
option( ?categ 'turboOpts    'errorLevel  "Liberal"
    'uniMode  "Turbo"
)

;Variaveis a serem salvas
save( 'v

       "/AckOn"
   "/AckOff"
```

```
   "/EventOn"
   "/EventOff"
     "/EnableOn"
     "/EnableOff"

   "/D0_YOn"
      "/D1_YOn"
     "/D2_YOn"
     "/D3_YOn"
     "/D4_YOn"
     "/D5_YOn"

     "/D0_XOn"
     "/D1_XOn"
      "/D2_XOn"
      "/D3_XOn"
     "/D4_XOn"
     "/D5_XOn"

     "/D0_YOff"
      "/D1_YOff"
     "/D2_YOff"
     "/D3_YOff"
      "/D4_YOff"
      "/D5_YOff"

     "/D0_XOff"
     "/D1_XOff"
   "/D2_XOff"
     "/D3_XOff"
      "/D4_XOff"
      "/D5_XOff"
)

temp( 27 )

monteCarlo( ?numIters "1" ?startIter "1"
    ?analysisVariation 'Mismatch ?sweptParam "None"
```

```
    ?sweptParamVals "27" ?saveData t
    ?nomRun "yes" ?append nil
    ?saveProcessParams nil
)
monteRun()

selectResult( 'tran )
plot(

     getData("/D0_XOff")
     getData("/D1_XOff")
     getData("/D2_XOff")
     getData("/D3_XOff")
     getData("/D4_XOff")
     getData("/D5_XOff")

     getData("/D0_XOn")
     getData("/D1_XOn")
     getData("/D2_XOn")
     getData("/D3_XOn")
     getData("/D4_XOn")
     getData("/D5_XOn")

     getData("/D0_YOff")
     getData("/D1_YOff")
     getData("/D2_YOff")
     getData("/D3_YOff")
     getData("/D4_YOff")
     getData("/D5_YOff")

     getData("/D0_YOn")
     getData("/D1_YOn")
     getData("/D2_YOn")
     getData("/D3_YOn")
     getData("/D4_YOn")
     getData("/D5_YOn")

     getData("/EnableOn")
     getData("/EnableOff")
```

```
     getData("/EventOn")
     getData("/EventOff")
     getData("/AckOn")
     getData("/AckOff")

)

; Salvando dados
STRING_file = strcat("/home/../spectre/schematic/DegrauResposta" )
PORT = outfile( STRING_file "w" )
ocnPrint( ?output PORT    getData("/D0_XOff")   getData("/D1_XOff")
  getData("/D2_XOff")   getData("/D3_XOff")  getData("/D4_XOff")
 getData("/D5_XOff")   getData("/D0_XOn")   getData("/D1_XOn")
 getData("/D2_XOn")   getData("/D3_XOn")   getData("/D4_XOn")
getData("/D5_XOn")    getData("/D0_YOff")   getData("/D1_YOff")
  getData("/D2_YOff")   getData("/D3_YOff")  getData("/D4_YOff")
 getData("/D5_YOff")   getData("/D0_YOn")   getData("/D1_YOn")
 getData("/D2_YOn")   getData("/D3_YOn")   getData("/D4_YOn")
getData("/D5_YOn")  getData("/EnableOn")  getData("/EnableOff")
   getData("/EventOn")  getData("/EventOff")   getData("/AckOn")
getData("/AckOff")  )
     close( PORT )
```

A.2 Script Impulso no Espaço e no Tempo

```
ocnWaveformTool( 'wavescan )
simulator( 'spectre )

;Leitura das bibliotecas
design(  "/home/../netlist")
resultsDir( "/home/../schematic" )
modelFile(
    '("/cds/Opus/user/AMS370/spectre/c35/mcparams.scs" "")
    '("/cds/Opus/user/AMS370/spectre/c35/cmos53.scs" "cmostm")
    '("/cds/Opus/user/AMS370/spectre/c35/res.scs" "restm")
    '("/cds/Opus/user/AMS370/spectre/c35/cap.scs" "captm")
```

```
    '("/cds/Opus/user/AMS370/spectre/c35/bip.scs" "biptm")
    '("/cds/Opus/user/AMS370/spectre/c35/ind.scs" "indtm")
)


analysis('tran ?stop "15m" ?save "selected"  ?compression "yes"
?finalTimeOp nil  )
; Atribuicao de valores  (desVar)

desVar(   "I1_1"   0 )
desVar(   "I1_2"   0 )
desVar(   "I1_3"   0 )
desVar(   "I1_4"   0 )
desVar(   "I1_5"   0p )
desVar(   "I1_6"   0 )
desVar(   "I1_7"   0 )
desVar(   "I1_8"   0 )
desVar(   "I1_9"   0p )
desVar(   "I1_10"   0 )
desVar(   "I1_11"   0 )
desVar(   "I1_12"   0p )
desVar(   "I1_13"   0 )
desVar(   "I1_14"   0p )
desVar(   "I1_15"   0p )
desVar(   "I1_16"   0 )
desVar(   "I1_17"   0 )
desVar(   "I1_18"   0 )
desVar(   "I1_19"   0 )
desVar(   "I1_20"   0 )
desVar(   "I1_21"   0 )
desVar(   "I1_22"   0 )
desVar(   "I1_23"   0 )
desVar(   "I1_24"   0p )
desVar(   "I1_25"   0 )
desVar(   "I1_26"   0 )
desVar(   "I1_27"   0 )
desVar(   "I1_28"   0 )
desVar(   "I1_29"   0p )
desVar(   "I1_30"   0 )
```

```
desVar(   "I1_31"   0 )
desVar(   "I1_32"   0 )
desVar(   "I1_33"   0 )
desVar(   "I1_34"   0p )
desVar(   "I1_35"   0p )
desVar(   "I1_36"   0 )
desVar(   "I1_37"   0 )
desVar(   "I1_38"   0 )
desVar(   "I1_39"   0 )
desVar(   "I1_40"   0 )
desVar(   "I1_41"   0 )
desVar(   "I1_42"   0 )
desVar(   "I1_43"   0 )
desVar(   "I1_44"   0p )
desVar(   "I1_45"   0 )
desVar(   "I1_46"   0 )
desVar(   "I1_47"   0 )
desVar(   "I1_48"   0 )
desVar(   "I1_49"   0 )
desVar(   "I1_50"   0 )
desVar(   "I2_1"   0 )
desVar(   "I2_2"   0 )
desVar(   "I2_3"   0 )
desVar(   "I2_4"   0 )
desVar(   "I2_5"   0 )
desVar(   "I2_6"   0 )
desVar(   "I2_7"   0 )
desVar(   "I2_8"   0 )
desVar(   "I2_9"   0 )
desVar(   "I2_10"   0 )
desVar(   "I2_11"   0p )
desVar(   "I2_12"   0 )
desVar(   "I2_13"   0 )
desVar(   "I2_14"   0 )
desVar(   "I2_15"   0 )
desVar(   "I2_16"   0p )
desVar(   "I2_17"   0 )
desVar(   "I2_18"   0p )
desVar(   "I2_19"   0 )
```

```
desVar(   "I2_20"   0 )
desVar(   "I2_21"   0p )
desVar(   "I2_22"   0 )
desVar(   "I2_23"   0 )
desVar(   "I2_24"   0 )
desVar(   "I2_25"   0p )
desVar(   "I2_26"   0 )
desVar(   "I2_27"   0 )
desVar(   "I2_28"   0 )
desVar(   "I2_29"   0 )
desVar(   "I2_30"   0 )
desVar(   "I2_31"   0 )
desVar(   "I2_32"   0 )
desVar(   "I2_33"   0 )
desVar(   "I2_34"   0 )
desVar(   "I2_35"   0 )
desVar(   "I2_36"   0 )
desVar(   "I2_37"   0 )
desVar(   "I2_38"   0 )
desVar(   "I2_39"   0p )
desVar(   "I2_40"   0p )
desVar(   "I2_41"   0 )
desVar(   "I2_42"   0 )
desVar(   "I6_35"   0 )
desVar(   "I6_36"   0 )
desVar(   "I6_37"   0 )
desVar(   "I6_38"   0 )
desVar(   "I6_39"   0 )
desVar(   "I6_40"   0 )
desVar(   "I6_41"   0 )
desVar(   "I6_42"   0 )
desVar(   "I6_43"   0 )
desVar(   "I6_44"   0 )
desVar(   "I6_45"   0 )
desVar(   "I6_46"   0 )
desVar(   "I6_47"   0 )
desVar(   "I6_48"   0 )
desVar(   "I6_49"   0 )
desVar(   "I6_50"   0p )
```

```
desVar(   "I7_1"   0 )
desVar(   "I7_2"   0 )
desVar(   "I7_3"   0p )
desVar(   "I7_4"   0 )
desVar(   "I7_5"   0 )
desVar(   "I7_6"   0 )
desVar(   "I15_9"   0 )
desVar(   "I15_10"   0 )
desVar(   "I15_11"   0p )
desVar(   "I15_12"   0p )
desVar(   "I15_13"   0p )
desVar(   "I15_14"   0 )
desVar(   "I15_15"   0 )
desVar(   "I15_16"   0 )
desVar(   "I15_17"   0 )
desVar(   "I15_18"   0 )
desVar(   "I15_19"   0 )
desVar(   "I15_20"   0 )
desVar(   "I15_21"   0p )
desVar(   "I15_22"   0 )
desVar(   "I15_23"   0 )
desVar(   "I15_24"   0p )
desVar(   "I15_25"   0 )
desVar(   "I15_26"   0 )
desVar(   "I15_27"   0 )
desVar(   "I15_28"   0 )
desVar(   "I15_29"   0 )
desVar(   "I15_30"   0 )
desVar(   "I15_31"   0 )
desVar(   "I15_32"   0 )
desVar(   "I15_33"   0 )
desVar(   "I15_34"   0 )
desVar(   "I15_35"   0 )
desVar(   "I15_36"   0 )
desVar(   "I15_37"   0 )
desVar(   "I15_38"   0 )
desVar(   "I15_39"   0 )
desVar(   "I15_40"   0 )
desVar(   "I15_41"   0 )
```

```
desVar(   "I15_42"   0 )
desVar(   "I15_43"   0 )
desVar(   "I15_44"   0p )
desVar(   "I15_45"   0 )
desVar(   "I15_46"   0 )
desVar(   "I15_47"   0 )
desVar(   "I15_48"   0 )
desVar(   "I15_49"   0 )
desVar(   "I15_50"   0 )
desVar(   "I16_1"   0 )
desVar(   "I16_2"   0 )
desVar(   "I16_3"   0p )
desVar(   "I16_4"   0 )
desVar(   "I16_5"   0p )
desVar(   "I16_6"   0p )
desVar(   "I16_7"   0p )
desVar(   "I16_8"   0p )
desVar(   "I16_9"   0 )
desVar(   "I16_10"   0 )
desVar(   "I16_11"   0 )
desVar(   "I16_12"   0 )
desVar(   "I16_13"   0 )
desVar(   "I16_14"   0 )
desVar(   "I16_15"   0 )
desVar(   "I16_16"   0p )
desVar(   "I16_17"   0 )
desVar(   "I16_18"   0 )
desVar(   "I16_19"   0 )
desVar(   "I16_20"   0p )
desVar(   "I16_21"   0 )
desVar(   "I16_22"   0 )
desVar(   "I16_23"   0 )
desVar(   "I16_24"   0 )
desVar(   "I16_25"   0p )
desVar(   "I16_26"   0p )
desVar(   "I16_27"   0p )
desVar(   "I16_28"   0 )
desVar(   "I16_29"   0 )
desVar(   "I16_30"   0p )
```

```
desVar(   "I16_31"   0 )
desVar(   "I16_32"   0 )
desVar(   "I16_33"   0 )
desVar(   "I16_34"   0 )
desVar(   "I16_35"   0 )
desVar(   "I16_36"   0p )
desVar(   "I16_37"   0 )
desVar(   "I16_38"   0 )
desVar(   "I16_39"   0 )
desVar(   "I16_40"   0p )
desVar(   "I16_41"   0p )
desVar(   "I16_42"   0 )
desVar(   "I16_43"   0 )
desVar(   "I16_44"   0 )
desVar(   "I16_45"   0 )
desVar(   "I16_46"   0 )
desVar(   "I16_47"   0 )
desVar(   "I16_48"   0p )
desVar(   "I16_49"   0 )
desVar(   "I16_50"   0 )
desVar(   "I17_1"   0 )
desVar(   "I17_2"   0 )
desVar(   "I17_3"   0 )
desVar(   "I17_4"   0 )
desVar(   "I17_5"   0 )
desVar(   "I22_9"   0p )
desVar(   "I22_10"   0p )
desVar(   "I22_11"   0p )
desVar(   "I22_12"   0 )
desVar(   "I22_13"   0 )
desVar(   "I22_14"   0 )
desVar(   "I22_15"   0 )
desVar(   "I22_16"   0 )
desVar(   "I22_17"   0p )
desVar(   "I22_18"   0 )
desVar(   "I22_19"   0 )
desVar(   "I22_20"   0 )
desVar(   "I22_21"   0 )
desVar(   "I22_22"   0 )
```

```
desVar(   "I22_23"   0p )
desVar(   "I22_24"   0 )
desVar(   "I22_25"   0p )
desVar(   "I22_26"   0p )
desVar(   "I22_27"   0 )
desVar(   "I22_28"   0 )
desVar(   "I22_29"   0 )
desVar(   "I22_30"   0 )
desVar(   "I22_31"   0p )
desVar(   "I22_32"   0 )
desVar(   "I22_33"   0 )
desVar(   "I22_34"   0 )
desVar(   "I22_35"   0 )
desVar(   "I22_36"   0 )
desVar(   "I22_37"   0p )
desVar(   "I22_38"   0 )
desVar(   "I22_39"   0p )
desVar(   "I22_40"   0 )
desVar(   "I22_41"   0p )
desVar(   "I22_42"   0 )
desVar(   "I22_43"   0 )
desVar(   "I22_44"   0 )
desVar(   "I22_45"   0p )
desVar(   "I22_46"   0 )
desVar(   "I22_47"   0 )
desVar(   "I22_48"   0 )
desVar(   "I22_49"   0 )
desVar(   "I22_50"   0 )
desVar(   "I23_1"   0p )
desVar(   "I23_2"   0 )
desVar(   "I23_3"   0 )
desVar(   "I23_4"   0 )
desVar(   "I23_5"   0 )
desVar(   "I23_6"   0 )
desVar(   "I23_7"   0 )
desVar(   "I23_8"   0 )
desVar(   "I23_9"   0 )
desVar(   "I23_10"   0 )
desVar(   "I23_11"   0 )
```

```
desVar(   "I23_12"   0 )
desVar(   "I23_13"   0p )
desVar(   "I23_14"   0 )
desVar(   "I23_15"   0 )
desVar(   "I23_16"   0 )
desVar(   "I23_17"   0 )
desVar(   "I23_18"   0p )
desVar(   "I23_19"   0p )
desVar(   "I23_20"   0p )
desVar(   "I23_21"   0 )
desVar(   "I23_22"   0 )
desVar(   "I23_23"   0p )
desVar(   "I23_24"   0 )
desVar(   "I23_25"   0 )
desVar(   "I23_26"   0 )
desVar(   "I23_27"   0 )
desVar(   "I23_28"   0 )
desVar(   "I23_29"   0p )
desVar(   "I23_30"   0 )
desVar(   "I23_31"   0 )
desVar(   "I23_32"   0 )
desVar(   "I23_33"   0 )
desVar(   "I23_34"   0p )
desVar(   "I23_35"   0 )
desVar(   "I23_36"   0 )
desVar(   "I23_37"   0 )
desVar(   "I23_38"   0 )
desVar(   "I23_39"   0 )
desVar(   "I23_40"   0 )
desVar(   "I23_41"   0 )
desVar(   "I23_42"   0p )
desVar(   "I23_43"   0 )
desVar(   "I23_44"   0 )
desVar(   "I23_45"   0 )
desVar(   "I23_46"   0 )
desVar(   "I23_47"   0 )
desVar(   "I23_48"   0 )
desVar(   "I23_49"   0 )
desVar(   "I23_50"   0 )
```

```
desVar(   "I24_1"   0p )
desVar(   "I24_2"   0 )
desVar(   "I24_3"   0 )
desVar(   "I24_4"   0 )
desVar(   "I24_5"   0p )
desVar(   "I24_6"   0p )
desVar(   "I24_7"   0 )
desVar(   "I24_8"   0 )
desVar(   "I24_9"   0 )
desVar(   "I24_10"   0 )
desVar(   "I24_11"   0 )
desVar(   "I24_12"   0 )
desVar(   "I24_13"   0p )
desVar(   "I24_14"   0 )
desVar(   "I24_15"   0 )
desVar(   "I24_16"   0 )
desVar(   "I24_17"   0 )
desVar(   "I24_18"   0 )
desVar(   "I24_19"   0 )
desVar(   "I24_20"   0 )
desVar(   "I24_21"   0p )
desVar(   "I24_22"   0 )
desVar(   "I24_23"   0 )
desVar(   "I24_24"   0 )
desVar(   "I24_25"   0p )
desVar(   "I24_26"   0 )
desVar(   "I24_27"   0 )
desVar(   "I24_28"   0 )
desVar(   "I24_29"   0 )
desVar(   "I24_30"   0 )
desVar(   "I24_31"   0p )
desVar(   "I24_32"   0 )
desVar(   "I24_33"   0 )
desVar(   "I24_34"   0 )
desVar(   "I24_35"   0 )
desVar(   "I24_36"   0 )
desVar(   "I24_37"   0 )
desVar(   "I24_38"   0p )
desVar(   "I24_39"   0 )
```

```
desVar(   "I24_40"   0 )
desVar(   "I24_41"   0 )
desVar(   "I24_42"   0 )
desVar(   "I24_43"   0 )
desVar(   "I24_44"   0 )
desVar(   "I24_45"   0 )
desVar(   "I24_46"   0 )
desVar(   "I24_47"   0 )
desVar(   "I24_48"   0 )
desVar(   "I24_49"   0 )
desVar(   "I24_50"   0 )
desVar(   "I25_1"   0 )
desVar(   "I25_2"   0 )
desVar(   "I25_3"   0 )
desVar(   "I25_4"   0 )
desVar(   "I25_5"   0 )
desVar(   "I25_6"   0 )
desVar(   "I25_7"   0 )
desVar(   "I25_8"   0p )
desVar(   "I25_9"   0 )
desVar(   "I25_10"   0p )
desVar(   "I25_11"   0p )
desVar(   "I25_12"   0 )
desVar(   "I25_13"   0 )
desVar(   "I25_14"   0 )
desVar(   "I25_15"   0p )
desVar(   "I25_16"   0 )
desVar(   "I25_17"   0 )
desVar(   "I25_18"   0 )
desVar(   "I25_19"   0 )
desVar(   "I25_20"   0p )
desVar(   "I25_21"   0 )
desVar(   "I25_22"   0 )
desVar(   "I25_23"   0p )
desVar(   "I25_24"   0 )
desVar(   "I25_25"   50u)
desVar(   "I25_26"   0 )
desVar(   "I25_27"   0p )
desVar(   "I25_28"   0 )
```

```
desVar(   "I25_29"   0 )
desVar(   "I25_30"   0 )
desVar(   "I25_31"   0 )
desVar(   "I25_32"   0p )
desVar(   "I25_33"   0 )
desVar(   "I25_34"   0p )
desVar(   "I25_35"   0 )
desVar(   "I25_36"   0 )
desVar(   "I25_37"   0 )
desVar(   "I25_38"   0 )
desVar(   "I25_39"   0 )
desVar(   "I25_40"   0 )
desVar(   "I25_41"   0 )
desVar(   "I25_42"   0 )
desVar(   "I25_43"   0 )
desVar(   "I25_44"   0p )
desVar(   "I25_45"   0 )
desVar(   "I25_46"   0p )
desVar(   "I25_47"   0 )
desVar(   "I25_48"   0 )
desVar(   "I25_49"   0 )
desVar(   "I25_50"   0 )
desVar(   "I26_1"   0 )
desVar(   "I26_2"   0 )
desVar(   "I26_3"   0 )
desVar(   "I26_4"   0 )
desVar(   "I26_5"   0 )
desVar(   "I26_6"   0 )
desVar(   "I26_7"   0p )
desVar(   "I26_8"   0 )
desVar(   "I26_9"   0 )
desVar(   "I26_10"   0 )
desVar(   "I26_11"   0p )
desVar(   "I26_12"   0 )
desVar(   "I26_13"   0 )
desVar(   "I26_14"   0 )
desVar(   "I26_15"   0 )
desVar(   "I26_16"   0 )
desVar(   "I26_17"   0 )
```

```
desVar(   "I26_18"   0p )
desVar(   "I26_19"   0 )
desVar(   "I26_20"   0 )
desVar(   "I26_21"   0 )
desVar(   "I26_22"   0 )
desVar(   "I26_23"   0 )
desVar(   "I26_24"   0 )
desVar(   "I26_25"   0 )
desVar(   "I26_26"   0p )
desVar(   "I26_27"   0 )
desVar(   "I26_28"   0 )
desVar(   "I26_29"   0p )
desVar(   "I26_30"   0 )
desVar(   "I26_31"   0p )
desVar(   "I26_32"   0 )
desVar(   "I26_33"   0 )
desVar(   "I30_49"   0p )
desVar(   "I30_50"   0 )
desVar(   "I31_1"   0p )
desVar(   "I31_2"   0 )
desVar(   "I31_3"   0 )
desVar(   "I31_4"   0 )
desVar(   "I31_5"   0 )
desVar(   "I31_6"   0p
desVar(   "Vbias" 2.8 )
desVar(   "Vu"  2.2 )
desVar(   "vss" 220m )
desVar(   "Vrhb" 2.35)
desVar(   "vbq"  2.3  )
desVar(   "vr"  2.27)
desVar(   "Vbiash" 2.65)
desVar(   "Vrcb"  2.8)
desVar(   "Pref"  2.2)

option(    'error  "yes"
    'warn  "yes"
    'opptcheck  "yes"
    'topcheck  "full"
    'gmin_check  "all"
```

```
)
option( ?categ 'turboOpts    'errorLevel  "Liberal"
    'uniMode  "Turbo"
)

;Variaveis a serem salvas
save( 'v

     "/D0_XOff"
     "/D1_XOff"
   "/D2_XOff"
     "/D3_XOff"
      "/D4_XOff"
      "/D5_XOff"

     "/D0_XOn"
     "/D1_XOn"
      "/D2_XOn"
      "/D3_XOn"
     "/D4_XOn"
     "/D5_XOn"

     "/D0_YOff"
      "/D1_YOff"
     "/D2_YOff"
     "/D3_YOff"
      "/D4_YOff"
      "/D5_YOff"

   "/D0_YOn"
     "/D1_YOn"
     "/D2_YOn"
     "/D3_YOn"
     "/D4_YOn"
     "/D5_YOn"

     "/EnableOn"
     "/EnableOff"
   "/EventOn"
```

```
   "/EventOff"
           "/AckOn"
   "/AckOff"
)

temp( 27 )
monteCarlo( ?numIters "1" ?startIter "1"
    ?analysisVariation 'Mismatch ?sweptParam "None"
    ?sweptParamVals "27" ?saveData nil
    ?nomRun "yes" ?append nil
    ?saveProcessParams t
)
monteRun()

;Variaveis  salvas
save( 'v

       "net72724" ; Central
       "net72742";  26-25
       "net72712"; 26-26
       "net72700"; 26-25
       "net72634"; 26-24
       "net72604"; 25-24
       "net72316" ; 24-24
       "net72388" ; 24-25
       "net72964" ; 24-26
       "/I10281/VmemOn"  ;26-25
       "/I10281/VmemOff"; 26-25
       "/I10282/VmemOn"  ;Central
       "/I10282/VmemOff" ;Central
        "/D0_XOff"
        "/D1_XOff"
       "/D2_XOff"
        "/D3_XOff"
         "/D4_XOff"
         "/D5_XOff"

        "/D0_XOn"
        "/D1_XOn"
```

```
          "/D2_XOn"
          "/D3_XOn"
         "/D4_XOn"
         "/D5_XOn"

         "/D0_YOff"
          "/D1_YOff"
         "/D2_YOff"
         "/D3_YOff"
          "/D4_YOff"
          "/D5_YOff"

        "/D0_YOn"
         "/D1_YOn"
         "/D2_YOn"
         "/D3_YOn"
         "/D4_YOn"
         "/D5_YOn"

         "/EnableOn"
         "/EnableOff"
        "/EventOn"
        "/EventOff"
            "/AckOn"
        "/AckOff"
)

temp( 27 )
monteCarlo( ?numIters "1" ?startIter "1"
    ?analysisVariation 'Mismatch ?sweptParam "None"
    ?sweptParamVals "27" ?saveData nil
    ?nomRun "yes" ?append nil
    ?saveProcessParams t
)
monteRun()

selectResult( 'tran )
plot(
           getData("net72712")
```

```
        getData("net72700")
        getData("net72634")
        getData("net72604")
        getData("net72316")
        getData("net72388")
        getData("net72964")
getData("net72724")
getData("net72742")

getData("/I10281/VmemOn")  ;Central
getData("/I10281/VmemOff") ;Central

getData("/I10282/VmemOn")  ;Central
getData("/I10282/VmemOff") ;Central

getData("/D0_XOff")
getData("/D1_XOff")
getData("/D2_XOff")
getData("/D3_XOff")
getData("/D4_XOff")
getData("/D5_XOff")

getData("/D0_YOff")
getData("/D1_YOff")
getData("/D2_YOff")
getData("/D3_YOff")
getData("/D4_YOff")
getData("/D5_YOff")

getData("/D0_YOn")
getData("/D1_YOn")
getData("/D2_YOn")
getData("/D3_YOn")
getData("/D4_YOn")
getData("/D5_YOn")

getData("/D0_XOn")
getData("/D1_XOn")
getData("/D2_XOn")
```

```
     getData("/D3_XOn")
     getData("/D4_XOn")
     getData("/D5_XOn")

     getData("/AckOn")
     getData("/AckOff")

     getData("/EventOn")
     getData("/EventOff")
     getData("/EnableOn")
     getData("/EnableOff")
)

;    Salvando dados
STRING_file = strcat("/home/../ImpulsoCentral_ComCenx" )
PORT = outfile( STRING_file "w" )
ocnPrint( ?output PORT                getData("net72712")
getData("net72700")    getData("net72634")  getData("net72604")
getData("net72316")     getData("net72388")    getData("net72964")
      getData("net72724")     getData("net72742")
getData("/I10281/VmemOn")             getData("/I10281/VmemOff")
    getData("/I10282/VmemOn")             getData("/I10282/VmemOff")
     getData("/D0_XOff")   getData("/D1_XOff")   getData("/D2_XOff")
  getData("/D3_XOff")  getData("/D4_XOff")  getData("/D5_XOff")
 getData("/D0_XOn")   getData("/D1_XOn")   getData("/D2_XOn")
getData("/D3_XOn")   getData("/D4_XOn")   getData("/D5_XOn")
 getData("/D0_YOff")   getData("/D1_YOff")   getData("/D2_YOff")
  getData("/D3_YOff")  getData("/D4_YOff")  getData("/D5_YOff")
  getData("/D0_YOn")   getData("/D1_YOn")   getData("/D2_YOn")
 getData("/D3_YOn")   getData("/D4_YOn")   getData("/D5_YOn")
getData("/EnableOn")  getData("/EnableOff")      getData("/EventOn")
getData("/EventOff")   getData("/AckOn")  getData("/AckOff")  )
close( PORT )
```

A.3 Gerador de Scripts

```
clear all;
close all;
```

```
clc;

Row = 10; Col = 10; %Vetor de Imagem
I = double(imread('Quadrado10x10'));

texto = '';
I = I*20;
p = '';
save('teste.txt','p','-ascii');
myformat = '%s %s\n';
fid = fopen('teste.txt','a');

for X=1:Row;
    for Y=1:Col;
         fonte = char(strcat('%desVar( ','"I',num2str(X),'_',num2str(Y),'"
'));
         valor = char(strcat(num2str(I(X,Y)),'p )'));
         fprintf(fid, myformat, fonte,valor);

    end;
end;

fclose(fid);
```

A.4 Reconstrução dos Arquivos de Eventos

```
clear all;
close all;
clc;

MatX = 50; %Matriz X
MatY = 50; %MAtriz Y

Dados = importdata('ImpulsoET20mEspecifico.matlab',',',1);

[TamData, CompData] = size(Dados.data);

PosVec = [];
```

```
for x=1:2:(CompData-1);
    [maxVal, maxPos] = max(Dados.data(:,x));
    PosVec = [PosVec; maxPos];
end;

PosVec = PosVec';
[maxPos, MainCol] = max(PosVec);
Data = zeros(maxPos,(round(CompData/2)+1));

y=MainCol*2-1;
for x=1:maxPos;% completa a coluna de tempo (1)
    Data(x,1)=Dados.data(x,y);
end;

maxt = max(Dados.data(:,1));

col = 2;
for n = 1:2:(CompData-1);

[maxt, maxtpos] = max(Dados.data(:,n));
x = Dados.data(1:maxtpos,n);
y = Dados.data(1:maxtpos,n+1);
xi = Data(:,1);
aux = interp1(x,y,xi);
Data(:,col) = interp1(x,y,xi);
col = col+1;

end;
DadosBackUp = Dados;
Dados = double(Data>1.5);

% Legenda para Arquivo de Dados 10x10
% time (s) (1)
% D0_XOn  D1_XOn  D2_XOn  D3_XOn  D4_XOn  D5_XOn  D0_XOff  D1_XOff
D2_XOff  D3_XOff  D4_XOff  D5_XOff (13)
% D0_YOn  D1_YOn  D2_YOn  D3_YOn  D4_YOn  D5_YOn  D0_YOff  D1_YOff
D2_YOff  D3_YOff  D4_YOff  D5_YOff (25)
% EnabOn  EnabOff  EvenOn  EvenOff  AckOn  AckOff (31)
```

```
% Legenda para Arquivo de Dados 50x50 Humberto
% time (s) (1)
% D0_XOff  D1_XOff  D2_XOff  D3_XOff  D4_XOff  D5_XOff  D0_XOn
D1_XOn  D2_XOn  D3_XOn  D4_XOn  D5_XOn (13)
% D0_YOff  D1_YOff  D2_YOff  D3_YOff  D4_YOff  D5_YOff  D0_YOn
D1_YOn  D2_YOn  D3_YOn  D4_YOn  D5_YOn (25)
% EnabOn  EnabOff  EvenOn  EvenOff  AckOn  AckOff (31)
[tempo, Colunas] = size(Dados);

DX_Off = Dados(:,2)+Dados(:,3)*2+Dados(:,4)*4
+Dados(:,5)*8+Dados(:,6)*16+Dados(:,7)*32;
DY_Off = Dados(:,14)+Dados(:,15)*2+Dados(:,16)*4
+Dados(:,17)*8+Dados(:,18)*16+Dados(:,19)*32;

DX_Off = DX_Off';
DY_Off = DY_Off';



DX_On = Dados(:,8)+Dados(:,9)*2+Dados(:,10)*4
+Dados(:,11)*8+Dados(:,12)*16+Dados(:,13)*32;
DY_On = Dados(:,20)+Dados(:,21)*2+Dados(:,22)*4
+Dados(:,23)*8+Dados(:,24)*16+Dados(:,25)*32;

DX_On = DX_On';
DY_On = DY_On';

ReqOff = Dados(:,29)';
ConfOff = Dados(:,27)';
ReqOn = Dados(:,28)';
ConfOn = Dados(:,26)';

RecImag = zeros(MatX,MatY); % Tamanho da Imagem
RecImag_ = zeros(MatX,MatY); % Tamanho da Imagem

for t = 1:tempo;

    x = DX_On(t);
    y = DY_On(t);
```

```
    x_ = DX_Off(t);
    y_ = DY_Off(t);

    if (ReqOn(t)==1);
        if (ConfOn(t)==0);
            if ((x>0)&(x<(MatX+1)))&((y>0)&(y<(MatY+1)));
                RecImag(x,y) = RecImag(x,y)+256/tempo;
            end;
        end;
    end;

    if (ReqOff(t)==1);
        if (ConfOff(t)==0);
            if ((x_>0)&(x_<(MatX+1)))&((y_>0)&(y_<(MatY+1)));

                RecImag_(x_,y_) = RecImag_(x_,y_)+256/tempo;
            end;
        end;
    end;
end;

Porcentagem_Conflito_On = 1/sum(ReqOn)*100*sum(ConfOn)
Porcentagem_Conflito_Off = 1/sum(ReqOff)*100*sum(ConfOff)
Taxa_Eventos_On = sum(ReqOn)/10e-3
Taxa_Eventos_Off = sum(ReqOff)/10e-3

figure;
I = (RecImag/max(max(RecImag)));
surf(I)
figure;
I = (RecImag_/max(max(RecImag_)));
surf(I)
```