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

IMAGEADOR CMOS COM COMPRESSÃO DE IMAGENS NO PLANO FOCAL BASEADA NO ALGORITMO EMBEDDED ZEROTREE WAVELET

Bruno Bastos Cardoso

Dissertação de Mestrado 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 Mestre em Engenharia Elétrica.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Rio de Janeiro
Março de 2013

IMAGEADOR CMOS COM COMPRESSÃO DE IMAGENS NO PLANO FOCAL BASEADA NO ALGORITMO EMBEDDED ZEROTREE WAVELET

Bruno Bastos Cardoso

DISSERTAÇÃO 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 MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

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

Prof. Fernando Antônio Pinto Barúqui, D.Sc.

Prof. Estêvão Coelho Teixeira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2013

Bastos Cardoso, Bruno

Imageador CMOS com Compressão de Imagens no Plano Focal Baseada no Algoritmo Embedded Zerotree Wavelet/Bruno Bastos Cardoso. – Rio de Janeiro: UFRJ/COPPE, 2013.

XIV, 102 p.: il.; 29, 7cm.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2013.

Referências Bibliográficas: p. 85 – 86.

1. Imageador CMOS. 2. Compressão de Imagens. 3. Plano Focal. I. Gomes, José Gabriel Rodríguez Carneiro. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Ao Léo, cão amigo e
companheiro, que descanse em
paz ao lado do Deus.

Agradecimentos

Agradeço aos meus pais e à minha família pelo apoio durante esta jornada.

Ao meu orientador José Gabriel Rodríguez Caneiro Gomes por todo o auxílio e paciência ao longo destes anos.

Ao CNPq pelo financiamento do meu curso e do projeto.

Aos professores e colaboradores da COPPE/UFRJ, não apenas aos quais tive o prazer de ser aluno, mas a todos os que contribuíram para a minha formação.

Aos colegas e professores do Laboratório de Processamento Analógico e Digital de Sinais pela companhia, ajuda e incentivo.

Agradeço também aos colegas de turma pelo espírito de grupo, dedicação e solidariedade nas horas de dificuldade.

E agradeço também aos amigos da vida diária pela amizade, confiança e companhia.

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

IMAGEADOR CMOS COM COMPRESSÃO DE IMAGENS NO PLANO FOCAL BASEADA NO ALGORITMO EMBEDDED ZEROTREE WAVELET

Bruno Bastos Cardoso

Março/2013

Orientador: José Gabriel Rodríguez Carneiro Gomes

Programa: Engenharia Elétrica

Apresenta-se, nesta dissertação, um projeto de um circuito de compressão de imagens no plano focal de câmeras digitais. A compressão é realizada pela implementação em hardware analógico de um algoritmo com perdas baseado na teoria de *wavelets* e que emprega estruturas de árvores de zeros para classificar e descartar informação. O algoritmo permite atingir uma determinada taxa de compressão através de um processo iterativo que reduz progressivamente a resolução da imagem.

Um sensor de imagens com 32 x 32 *pixels* contendo os circuitos para o processamento da imagem foi projetado em tecnologia CMOS 0.35 μm da AMS. Simulações elétricas que levam em conta as variações do processo de fabricação CMOS de acordo com os parâmetros fornecidos pelo fabricante foram obtidas e comparadas com a simulação computacional do algoritmo.

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

CMOS IMAGE SENSOR WITH FOCAL PLANE IMAGE COMPRESSION BASED ON THE EMBEDDED ZEROTREE WAVELET ALGORITHM

Bruno Bastos Cardoso

March/2013

Advisor: José Gabriel Rodríguez Carneiro Gomes

Department: Electrical Engineering

In this work, we present the design of a focal plane image compression circuit for digital cameras. The compression is obtained by the analog hardware implementation of a lossy algorithm based on the wavelet theory and which employs zerotree structures to classify and discard data. The algorithm allows meeting a certain compression rate through an iterative process which progressively reduces image resolution.

A 32 x 32 pixels image sensor containing the image processing circuits was designed in AMS CMOS 0.35 μm technology. Electrical simulations which consider the CMOS fabrication process variations in accordance to the parameters provided by the foundry were obtained and compared with the algorithm's computer simulation.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Câmeras Eletrônicas Digitais

Câmeras são instrumentos bastante populares e de grande contribuição para áreas como lazer e indústria. Desde a concepção das câmeras analógicas originais de imagens estáticas, conhecidas como câmeras de halogeneto de prata pelo emprego desta substância, a tecnologia óptica sofreu um grande avanço com o aprimoramento da fabricação de lentes, mas o processo de fotografia só realmente passou por uma revolução com a invenção das câmeras eletrônicas digitais [1].

A tecnologia das câmeras de halogeneto de prata consiste basicamente de uma lente capaz de projetar a luz incidente sobre uma película fotossensível, o chamado filme fotográfico. Para ser possível a visualização da imagem, no entanto, é necessário que esta película passe primeiramente por um conjunto de etapas para a revelação da mesma. Por este motivo, não é possível determinar a qualidade de uma fotografia imediatamente após esta ter sido tirada. Além disto, o processo de revelação tem a chance de danificar permanentemente o filme. Além da possibilidade de isto se tornar uma fonte de frustração para o fotógrafo, também a elaboração de um novo conjunto de fotografias requer a reposição do material fotossensível após o seu uso. Com a mão de obra e os materiais necessários para a revelação, são gerados custos que encarecem e complicam o processo.

Por não contarem com estes problemas, é fácil perceber por que as câmeras eletrônicas, sobretudo as digitais, se tornaram tão populares. A tecnologia eletrônica consiste em uma lente capaz de projetar a luz incidente sobre um sensor de imagem, que é um dispositivo que converte um sinal óptico em um sinal elétrico proporcional à luz incidente. No caso das câmeras eletrônicas analógicas, este sinal é gravado em uma fita magnética. Já nas câmeras eletrônicas digitais, o sinal é convertido em um sinal digital e armazenado em uma unidade de memória digital. Em ambos os casos, o dispositivo de armazenamento tem a vantagem de poder ser acessado a qualquer

momento.

Não apenas uma memória ou fita magnética hoje armazenam mais fotografias do que um rolo de filme teria a capacidade de conter, as câmeras eletrônicas mais modernas também possuem uma tela de cristal líquido onde a fotografia tirada pode ser imediatamente observada e avaliada por sua qualidade. Fotografias podem ser copiadas e reimpressas quantas vezes forem necessárias, sem risco de danificação do material original. Também não há custo de revelação ou necessidade de compra de filmes fotográficos, o que torna as câmeras eletrônicas financeiramente mais atraentes para os usuários.

O fato das câmeras digitais terem ultrapassado as analógicas na preferência do mercado se deve principalmente à evolução sofrida pelos processadores e tecnologias de armazenamento. Os dispositivos de memória digitais têm vantagens como maior capacidade de armazenamento, maior velocidade de acesso, maior tempo de vida útil, menor tamanho e menor custo de fabricação. Estas vantagens também influenciam na tendência do mercado, permitindo uma maior integração entre a câmera e outros dispositivos, o que acabou por transformar o comércio de câmeras não-eletrônicas e eletrônicas não-digitais em um nicho pequeno do mercado de câmeras.

1.2 Sensores de Imagem

Sensores de imagem são utilizados essencialmente em câmeras eletrônicas digitais, podendo ser classificados sobretudo pela faixa do espectro luminoso na qual são projetados para operar, pela sua resolução e pela sua tecnologia de fabricação.

A faixa de comprimento de onda da luz para o qual o sensor é projetado depende tanto do material fotossensível (geralmente o silício) quanto dos filtros ópticos empregados sobre estes. Usualmente os filtros ópticos utilizados são, no caso dos sensores das câmeras eletrônicas próprios para fotografias coloridas, arranjos de três cores conhecidos como mosaicos de Bayer. Por sua vez, os sensores de silício para imagens em tons de cinza não requerem filtros para ajustar as suas respostas espectrais à equivalência do olho humano (de 380 a 780 nm), já que estas se dão dominantemente nesta mesma faixa [1]. Quando estes sensores são utilizados em aplicações dentro de áreas técnicas cujas faixas de interesse estejam fora do espectro visível (como por exemplo o imageamento medicinal), torna-se necessária a utilização de outro tipo de semicondutor ou uma filtragem luminosa específica [1].

A resolução de um sensor de imagem representa a capacidade de definição de uma imagem que ele pode produzir. Esta característica depende do número de *pixels* (*picture elements*, nome dado às menores unidades capazes de formar uma cor em uma imagem digital) presentes no sensor. Um maior número de *pixels* assegura que detalhes não serão perdidos na captação da imagem. Outro fator também

determinante para a qualidade de um sensor de imagem é a razão entre o tamanho da área do elemento fotossensível e o tamanho da área do *pixel*, denominada fator de preenchimento ou *fill factor*. Um fator de preenchimento maior também contribui para a definição da imagem, pois garante que toda a incidência luminosa sobre o *pixel* seja captada. Idealmente, uma câmera teria um fator de preenchimento próximo de um e o maior número de *pixels* possível.

As principais tecnologias utilizadas para a construção de sensores de imagem são a CMOS (*Complementary Metal-Oxide Semiconductor*) e a CCD (*Charge-Coupled Device*). Dominante no mercado, a tecnologia CCD vem perdendo espaço para a tecnologia CMOS devido aos avanços tecnológicos em sua fabricação, destacando-se entre estes: a diminuição do tamanho dos *pixels* e a melhoria da razão sinal-ruído na leitura de suas saídas, um menor consumo de potência e um menor custo de fabricação [1].

A tecnologia CMOS conta com a vantagem sobre a tecnologia CCD de permitir maior integração entre o sensor de imagem e eventuais circuitos de pós-processamento. Estes circuitos são normalmente projetados para se conseguir uma qualidade de imagem bem próxima das obtidas pelos sensores CCD e para a realização de outros tipos de processamento na imagem. Além disto, dependendo dos tipos de substrato e sensor utilizados, sensores CMOS podem ser mais resistentes ou mesmo imunes ao efeito de *blooming*, um borrão de luz (geralmente vertical) que aparece na imagem obtida e que é causado por um excesso de exposição luminosa, mais frequente nos sensores CCD [1].

Existem outros efeitos indesejáveis que também ocorrem em ambos os tipos de sensores. Por exemplo, como as respostas individuais dos *pixels* de um sensor dependem das variações do processo que inevitavelmente ocorrem em sua fabricação, a não-uniformidade destas fatalmente torna o nível de saída de um *pixel* desbalanceada em relação à dos seus pares devido a diferenças de ganho e *offset*. Este problema é conhecido como ruído de padrão fixo (*Fixed-Pattern Noise* ou FPN). Outros problemas incluem a interferência causada pela corrente de escuro (*dark current*), que é a corrente sobre o fotodetector que ocorre mesmo quando não há incidência luminosa, e o ruído 1/f no circuito. Tanto o problema do FPN quanto o da corrente de escuro são mais pronunciados nos sensores CMOS do que nos sensores CCD.

1.2.1 Fotodetectores

O principal elemento do sensor de imagem é o fotodetector, que é uma região constituída de material semicondutor com o objetivo de gerar uma corrente elétrica (fotocorrente) proporcional ao número de fótons incidentes nesta região. Quando esta incidência de fótons ocorre, os elétrons da banda de valência do semicondutor

recebem energia e são excitados para a banda de condução, deixando uma lacuna. Os pares elétron-lacuna gerados, quando uma tensão externa é aplicada ao semicondutor produzem, então, a fotocorrente.

Os fotodetectores mais conhecidos são: fotocondutores, fotodiodos, fotodiodos PIN (*pinned photodiode* ou PPD), fotogates e fototransistores. Para os sensores de imagem CMOS, geralmente utiliza-se fotodiodos ou fotodiodos PIN, dependendo do compromisso estabelecido entre o *fill factor*, maior para uma mesma área quando se emprega fotodiodos comuns, e o grau desejado de imunidade a ruídos, melhor nos fotodiodos PIN [1].

1.2.2 *Active Pixel Sensor*

Um *Active Pixel Sensor*, ou simplesmente APS, é um sensor de imagem que consiste de um circuito integrado contendo uma matriz de fotodetectores e, para cada fotodetector, um amplificador ativo. A adição do amplificador ativo diretamente à saída do fotodetector diminui os efeitos de injeção de carga que geralmente ocorrem nos sensores de imagem com *pixel* passivo.

Pixels contendo um APS geralmente empregam fotodiodos ou fotogates. Configurações clássicas conhecidas são a 3-T, que utiliza fotodiodos, e a 4-T, que utiliza fotodiodos PIN. Ambas recebem seus nomes de acordo com a quantidade mínima de transistores para seguir seu princípio de funcionamento.

A estrutura 3-T clássica, mostrada na Figura 1.1 (a), consiste em um fotodiodo e um circuito de leitura com três transistores: um para realizar o *reset* do fotodiodo, gerando um acúmulo de cargas sobre o fotodetector e criando sobre este uma tensão de valor conhecido; um transistor configurado como seguidor de fonte, que age como um amplificador; e um transistor configurado como chave para a seleção individual do *pixel* em uma coluna de *pixels* com saídas interligadas, para a realização da sua leitura.

A estrutura 4-T funciona de maneira similar à 3-T, e pode ser vista na Figura 1.1 (b). O seu transistor extra (chamado de transistor de transmissão ou TX) é utilizado para o controle da região semicondutora intrínseca da junção PIN, que por sua vez é responsável por separar o nó de sensibilidade luminosa do nó de integração de cargas. Ocupa uma região física maior do que a estrutura 3-T, mas possui uma menor corrente de escuro e imunidade ao ruído de *reset* [1].

Normalmente são realizadas duas leituras de valores das correntes: uma durante um período de *reset*, em que o número de cargas sobre o fotodetector é mantido constante e, consequentemente, a tensão sobre este; e outra após um tempo conhecido, denominado tempo de integração, quando a exposição à luz já drenou uma quantidade de cargas do fotodetector proporcional à incidência luminosa. Os valo-

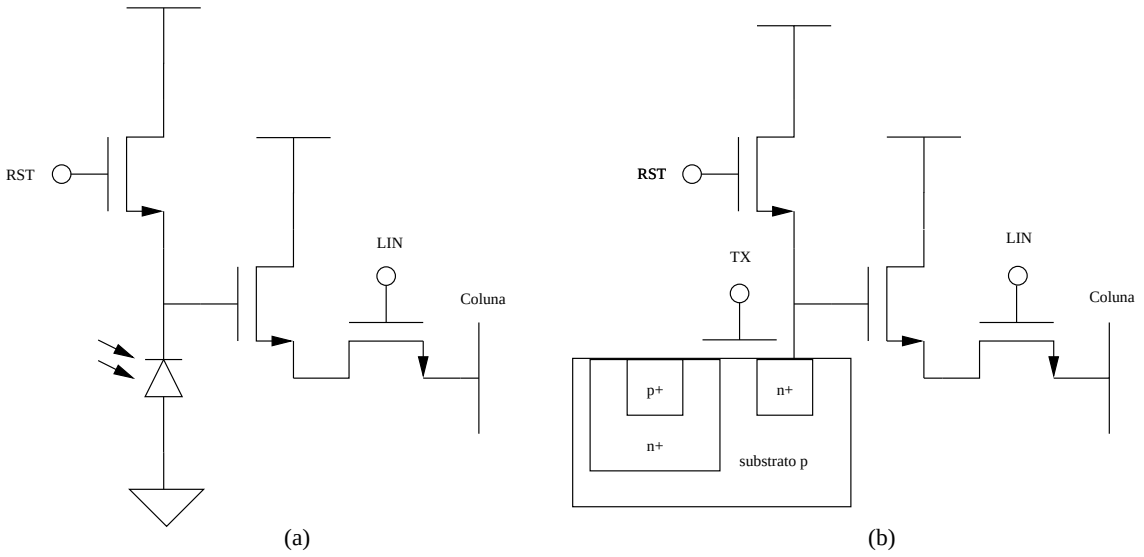

Figura 1.1: Típicos sensores APS: 3-T e 4-T.

res obtidos são comumente subtraídos um do outro por um circuito posterior, num processo denominado *Correlated Double Sampling* ou CDS. A principal utilidade desta operação é eliminar *offsets* causados pelo FPN, pela corrente de escuro e por outros efeitos [1].

1.3 Compressão de Imagens

O objetivo de se comprimir uma imagem é a redução da irrelevância e da redundância dos dados da mesma, de forma que seu armazenamento ou transmissão sejam feitos de uma forma mais eficiente. Por consequência, considera-se o melhor algoritmo de compressão aquele que produz a melhor qualidade de imagem para uma determinada taxa de *bits* ou taxa de compressão.

A compressão de uma imagem pode ser com ou sem perda de dados. A área de aplicação geralmente define onde um ou o outro tipo de algoritmo é o mais adequado, já que a compressão com perdas geralmente apresenta um maior potencial de compressão em relação à sem perdas.

Embora a compressão com perdas possa gerar distorções em uma imagem, quando feita de forma cuidadosa pode ocorrer da perda de fidelidade à imagem original ser tão pequena a ponto de ser imperceptível, ou apenas possível de ser detectada por olhos treinados. Uma imagem assim comprimida é denominada visualmente sem perdas, e este é geralmente o caso para imagens naturais (aquelas obtidas em uma fotografia).

Uma outra característica importante para um algoritmo de compressão de imagens, tanto os com perda quanto os sem perda, é a sua escalabilidade, isto é, o quanto um eventual descarte de dados da imagem (como por exemplo a perda de

alguns *bits* no final da transmissão da imagem) afeta a qualidade final da mesma. Em outras palavras, um algoritmo com alta escalabilidade permite uma imagem comprimida ser reconstruída, mesmo que com menor fidelidade, apenas por uma parte da sequência de dados obtidos. Esta característica é desejável por proporcionar ao decodificador a oportunidade de formar uma amostra da imagem antes mesmo do término de seu envio ou ao codificador a opção de variar a qualidade da imagem comprimida pela quantidade de dados enviados, de forma a satisfazer a um limite na banda de transmissão. Um algoritmo capaz de comprimir uma imagem em uma sequência de *bits* que possua uma ordem de importância com a característica descrita é denominado de codificação progressiva (*progressive coding*) ou codificação embutida (*embedded coding*).

A maneira mais utilizada para se medir a qualidade da reconstrução de uma imagem comprimida com perdas é através do cálculo da sua PSNR (*Peak Signal-to-Noise Ratio*), que determina a razão entre a máxima potência possível para um sinal e a potência do ruído em um dado sinal. A PSNR é usualmente dada na escala de decibéis, e para imagens monocromáticas ou coloridas representadas no sistema de cores RGB (e suas transformações lineares) é calculada pela Equação (1.1). Imagens coloridas representadas em outros sistemas de cor precisam ser convertidas.

$$\mathrm{PSNR} = 20 \log_{10} \left(\frac{P_{max}}{\sqrt{\mathrm{MSE}}} \right). \tag{1.1}$$

Quando se trata de calcular a PSNR de uma imagem, a potência máxima do sinal é fornecida pelo valor máximo possível que um *pixel* pode alcançar, como por exemplo, o valor 255 quando este é representado por oito *bits*. Já a potência do sinal ruidoso é dada pelos erros introduzidos na compressão e é determinada através do erro médio quadrático (*Mean Squared Error* ou MSE), calculado pela raiz quadrada da soma das diferenças quadráticas entre cada par de valores fornecido pelos *pixels* originais e pelos *pixels* obtidos da reconstrução da imagem. Em imagens monocromáticas, o MSE é dado pela Equação (1.2). Quando há mais de uma cor na imagem, existem diferentes maneiras de se calcular o MSE, mas o recomendado é que o mesmo seja feito para cada cor independentemente das outras e que, então, seja calculada uma média ponderada entre estes valores [2].

$$\mathrm{MSE} = \frac{1}{\mathrm{lin} \cdot \mathrm{col}} \sum_{i=0}^{lin-1} \sum_{j=0}^{col-1} \left[P_{\mathrm{original}}(i,j) - P_{\mathrm{restaurado}}(i,j) \right]^2. \tag{1.2}$$

1.3.1 Compressão de Imagens no Plano Focal

Câmeras eletrônicas digitais transformam uma imagem obtida em uma sequência de *bits*, geralmente na taxa de oito ou mais a cada *pixel*. Por esta razão, o total de

bits lidos na saída de uma matriz de *pixels* acaba por ocupar um grande espaço em memória. Como este espaço necessário cresce de forma multiplicativa com o aumento das dimensões desta matriz, torna-se interessante a implementação de técnicas de compressão para compensar a necessidade de maiores velocidades tanto para a leitura do sensor de imagem quanto para a transmissão dos dados resultantes deste aumento.

A tecnologia CMOS, por permitir o processamento de sinais no plano focal do próprio imageador através da inclusão de circuitos para um propósito diferente no mesmo chip, vem sendo muito utilizada neste contexto. Técnicas de *layout* ajudam a reduzir as capacitâncias parasitas, permitindo maior velocidade na leitura da resposta do circuito. Mais importante ainda, alguns algoritmos de compressão desenvolvidos para a computação e que geralmente requerem instruções com múltiplos ciclos em processadores podem ser implementados dentro do próprio *pixel*, garantindo um maior paralelismo e economizando tempo de processamento. O fato destes circuitos se localizarem no plano focal, diretamente após as saídas de leitura dos sensores APS também contribui para a redução da injeção de ruído nos mesmos.

Trabalhos anteriores nesta área ([3]-[6]) geralmente apresentam resultados interessantes do ponto de vista qualitativo, com sensores conseguindo obter imagens com uma boa nitidez, e também no sentido da compressão de dados, apresentando reduções significativas no número de *bits* de saída. O crescente número de trabalhos publicados para este assunto também indica que o campo é promissor para a indústria.

1.4 Proposta do Trabalho

Neste trabalho é proposto um imageador CMOS para aplicação em fotografia padrão e na faixa de luz visível, com uma compressão de imagens no plano focal realizada através de um algoritmo com perdas denominado *Embedded Zerotree Wavelet* ou EZW, originalmente proposto em [7]. Neste algoritmo, uma determinada taxa de compressão pode ser visada, e rodadas iterativas de compressão são realizadas até que se atinja a melhor qualidade possível para o tamanho em *bits* desejado para a codificação da imagem.

O imageador é composto de uma matriz de 32 x 32 *pixels*, cada um contendo um APS e mais os circuitos necessários para a implementação do algoritmo citado. É feita a decomposição de uma imagem obtida através de uma *wavelet* (ver Capítulo 2) em cinco escalas de resolução. Este trabalho realiza a implementação dos circuitos responsáveis por esta decomposição com os mesmos operando em modo de corrente (ver Capítulo 4), o que dispensa o uso de matrizes de capacitores e permite *pixels* menores em uma implementação multiresolução.

A maior parte dos circuitos de processamento além desta decomposição foram implementados para operar em modo de corrente devido ao menor número de transistores necessários em relação à operação em modo de tensão. Em contrapartida, todas as entradas e saídas, com a exceção de duas correntes de polarização externas necessárias para espelhamentos internos, são digitais para facilitar a comunicação com um eventual microcontrolador externo. O imageador conta com um conversor analógico-digital para realizar esta transição. A tecnologia escolhida para o projeto do circuito limita a tensão de alimentação para um máximo de 3,3 V, já que a operação em modo de corrente torna desnecessária uma tensão de alimentação maior. Além disto, uma menor tensão de alimentação contribui para reduzir o consumo de potência do circuito para uma mesma corrente.

O projeto dos circuitos foi feito de forma a permitir que estes executem o algoritmo de compressão utilizado da maneira mais paralelizada o possível, de forma que o resultado relativo a cada *pixel* esteja disponível após um breve intervalo de tempo (referente ao carregamento das suas capacitâncias) posterior à sua amostragem. Por esta razão, a maior parte dos circuitos opera de maneira estática, com a exceção dos registradores de deslocamento, que são responsáveis por realizar a comunicação externa e operam de maneira dinâmica (controlados por um sinal de *clock*).

Um microcontrolador é capaz de selecionar individualmente cada *pixel* ao fornecer seu endereço de linha e coluna. Esta característica facilita a implementação do algoritmo, já que a natureza inconstante da ordem de leitura dos *pixels* imposta pelo EZW (capaz até de tornar desnecessária a leitura de alguns *pixels*) inviabiliza a realização deste controle por um circuito interno ao chip. Logo, este trabalho transfere ao programa gravado no microcontrolador a tarefa de tanto decidir a ordem de leitura dos *pixels* quanto de verificar se um dado *pixel* deve ser lido, que deve seguir a previsão do algoritmo utilizado.

Um *clock* externo é necessário para o sincronismo das estruturas para comunicação serial. Os sinais de controle do circuito são todos projetados para serem manipulados por um microcontrolador externo e os *pixels* são enviados um por vez após a sua codificação através de uma saída digital serial. Os tamanho da palavra de codificação originada por um *pixel* compõe uma sequência de *bits* cujo tamanho é variável, podendo ser composta de dois ou de oito *bits* por *pixel* escaneado.

O algoritmo utilizado foi reproduzido e testado por simulações através do *software* MathWorks MATLAB. O imageador proposto foi projetado em tecnologia CMOS AMS 0.35 μm através do *software* CADENCE Custom IC Design Tools e suas simulações elétricas e estatísticas foram obtidas pelo simulador CADENCE Spectre para a confirmação dos resultados.

Ao longo deste trabalho, espera-se que o leitor possa compreender todo o processo

de decisão que envolve a compressão realizada pelo algoritmo, bem como identificar qual etapa do mesmo é realizada por cada circuito apresentado. Ao final serão apresentados os resultados obtidos e as conclusões do autor.

1.5 Organização

No Capítulo 1, foi apresentada uma introdução sobre os conceitos de imageadores e compressão de imagens. Também foi apresentado o imageador proposto por este trabalho, bem como sua contextualização.

O Capítulo 2 apresenta a transformada *wavelet* como uma ferramenta matemática, assim como o conceito de hierarquia de sub-bandas. Também apresenta a disposição dos *pixels* na implementação física do circuito de acordo com os níveis de resolução dos coeficientes por estes implementados e mostra as ligações necessárias entre estes *pixels* para a realização da transformada.

O Capítulo 3 fornece uma breve descrição do algoritmo EZW. Define uma árvore de zeros e apresenta a lógica da codificação envolvendo os coeficientes obtidos por uma transformada *wavelet*, oferecendo um exemplo para sua aplicação. Ao final, são mostradas as ligações físicas implementadas entre os *pixels* de diferentes escalas de resolução e mesma orientação de detalhes segundo as árvores de hierarquia.

No Capítulo 4 são mostrados os esquemáticos dos circuitos e fornecidos os parâmetros utilizados neste trabalho, juntamente com a descrição da operação do circuito. Também são apresentadas as respectivas simulações realizadas para parâmetros típicos e de Monte Carlo.

O Capítulo 5 realiza a comparação entre os resultados obtidos pelas simulações computacionais e os obtidos por simulações elétricas do circuito projetado, considerando variações estatísticas referentes ao processo de fabricação e ao descasamento entre transistores pelo método de Monte Carlo.

O Capítulo 6 apresenta uma conclusão sobre este trabalho e sugere algumas melhorias possíveis de serem implementadas em futuros trabalhos.

O Apêndice A fornece os códigos, escritos em linguagem MATLAB, utilizados para as simulações computacionais do algoritmo e para suporte nas simulações elétricas.

Capítulo 2

Transformadas *Wavelet*

Em 1909, o matemático húngaro Alfréd Haar apresentou o primeiro estudo conhecido sobre sua teoria de *wavelets* [8]. Ela, no entanto, só veio a se tornar conhecida por este nome alguns anos depois, quando diversos outros trabalhos foram desenvolvidos tendo sua teoria como base.

As transformadas *wavelet* são ferramentas matemáticas que permitem decompor ou quebrar sinais hierarquicamente em suas partes constituintes, tornando possível analisar os dados em diferentes domínios de frequência com a resolução de cada componente amarrada à sua escala.

Os algoritmos de decomposição em *wavelets* possuem a capacidade de processar dados em diferentes escalas ou resoluções. Oferecem uma técnica elegante para representar os níveis de detalhes presentes, seja a função de interesse uma imagem, uma curva, etc. Um algoritmo de decomposição em *wavelets* descreve uma função em termos de uma forma grosseira, acrescida de uma ou mais formas progressivamente mais ricas em detalhes. Seu objetivo é exatamente o de permitir tanto a visualização do todo quanto a das partes e suas singularidades.

As transformadas *wavelet* são largamente utilizadas na compressão de dados, já que sua aplicação realiza uma transformação dos dados de forma a descrevê-los em uma outra base matemática onde a correlação entre os mesmos torna-se explícita. A escolha do algoritmo que melhor se adapta aos dados nos permite, portanto, codificar a resposta obtida de forma que as resoluções de maior nível de detalhes tenham uma palavra de codificação de tamanho menor do que as demais, obtendo-se assim uma melhor taxa de compressão.

Alguns exemplos de algoritmos que utilizam transformadas *wavelet* são o *Joint Photographic Experts Group* 2000 (JPEG 2000) e o *Embedded Zerotree Wavelet* (EZW), este último descrito resumidamente neste trabalho.

2.1 Transformadas *Wavelet* de Haar Contínuas

As *wavelets* contínuas não são o foco deste trabalho, mas como foram a forma de apresentação original da teoria, são apresentadas por uma breve descrição a seguir. Maiores detalhes sobre as mesmas podem ser encontrados em diversas referências na literatura, como por exemplo em [9].

As *wavelets* contínuas são ondas de curta duração, com sua energia concentrada em um intervalo de tempo finito. Possuem certas propriedades matemáticas e são definidas no espaço $L^2(\mathbb{R})$ [9], onde $\mathbb{R}$ é o conjunto dos números reais. As *wavelets* de Haar são consideradas o tipo de *wavelet* mais simples que existe. Uma *wavelet* denominada mãe, mostrada na parte esquerda da Figura 2.1, é responsável por gerar uma base de Haar, o que é feito ao se dilatar, comprimir e/ou transladar esta função, conforme o exemplo da parte direita da Figura 2.1.

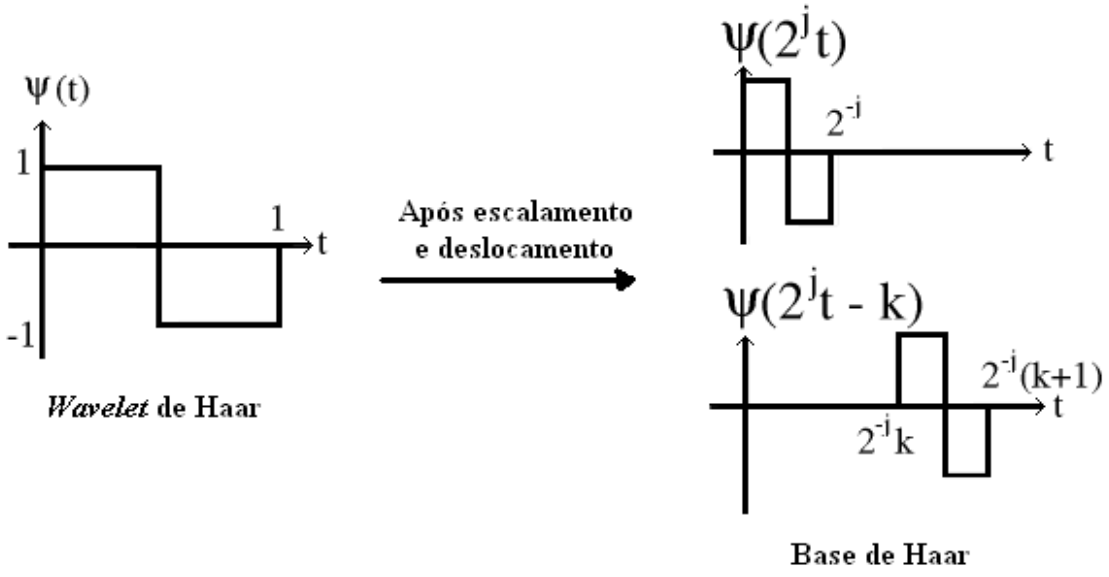

Figura 2.1: *Wavelet* de Haar contínua e exemplo de base gerada.

As outras *wavelets* de Haar geradas por estas modificações na função mãe são conhecidas como *wavelets* bebês, e um conjunto destas gera uma base de Haar. Um sinal contínuo ou contínuo por partes pode ser decomposto na base escolhida através da transformada *wavelet* contínua. Esta transformada permite a exploração das propriedades estatísticas do sinal na frequência, de maneira similar à transformada de Fourier, e também no tempo e no espaço.

2.2 Transformadas *Wavelet* de Haar Discretas

As transformadas *wavelet* de Haar discretas (abreviadas como HDWT) compartilham as propriedades de suas contrapartes contínuas adaptadas para o tempo dis-

creto. São utilizadas predominantemente na engenharia e na ciência da computação para auxiliar na compressão de dados, pois são computacionalmente simples de serem calculadas e oferecem bons resultados.

A transformada *wavelet* aplicada a um arranjo de dados não realiza compressão alguma deste, apenas tende a separar seu valor médio de sua dispersão estatística. Uma compressão sem perdas pode sim ser então realizada através da utilização de um código de entropia, mas existem alguns algoritmos com perdas que exploram ainda mais as propriedades das *wavelets*. Portanto, o entendimento de como realizar a transformada de uma imagem é essencial para o entendimento do porquê de alguns algoritmos de compressão com perdas favorecerem a manutenção de alguns dados e descartarem outros.

É importante evidenciar também que a transformada *wavelet* pode ser calculada de diversas maneiras, importando apenas que todas elas, ao final, forneçam o mesmo resultado. Ao longo deste capítulo, será apresentada a teoria conforme descrita em [10].

Na implementação discreta de uma transformada *wavelet* de Haar, um conjunto de matrizes quadradas denominadas matrizes de Haar são associadas às *wavelets*. Estas matrizes possuem somente elementos unitários (positivos ou negativos) e nulos, além da propriedade de que sua inversa é sempre a metade de sua transposta. As matrizes de Haar são identificadas pelo valor das suas dimensões, sendo a de duas e a de quatro, respectivamente nomeadas H_2 e H_4, mostradas na Equação (2.1).

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}. \tag{2.1}$$

Das matrizes de Haar são derivadas as matrizes utilizadas nas transformadas, que também são matrizes quadradas cujas dimensões têm o mesmo valor da matriz original, mas com seus elementos modificados de forma a torná-las ortogonais (satisfazem a condição de ter o seu inverso igual à sua transposta). As matrizes usadas nas transformadas de Haar de dimensões dois e quatro, respectivamente nomeadas W_2 e W_4, são mostradas na Equação (2.2).

$$W_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad W_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}. \tag{2.2}$$

Tipicamente, as matrizes apresentadas na Equação (2.2) sofrem alterações em suas dimensões, com a inclusão de elementos de valor nulo. Esta modificação tem o intuito de permitir a realização da transformada de um vetor ou de uma matriz com dimensões maiores que a sua - que por tal característica são incompatíveis para uma multiplicação matricial - sem precisar que seja realizado o seu fracionamento.

2.2.1 Transformadas *Wavelet* de Haar Discretas Unidimensionais

Sendo as propriedades das matrizes do conjunto W_k muito similares entre si, o foco deste trabalho se dará nas operações realizadas com a matriz onde $k = 2$ já que esta é aquela com a transformada mais simples de ser calculada.

A transformada *wavelet* de um vetor de dados V_n de tamanho $n = 2$ por uma matriz W_2 pode ser realizada de acordo com a Equação (2.3).

$$W_2V_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} v_1 + v_2 \\ v_1 - v_2 \end{bmatrix} = \begin{bmatrix} L1 \\ H1 \end{bmatrix} = V_2'. \quad (2.3)$$

Onde os vetores $L1$ e $H1$ são dados, respectivamente, pelas Equações (2.4) e (2.5):

$$L1 = \frac{1}{\sqrt{2}} \begin{bmatrix} v_1 + v_2 \end{bmatrix}. \quad (2.4)$$

$$H1 = \frac{1}{\sqrt{2}} \begin{bmatrix} v_1 - v_2 \end{bmatrix}. \quad (2.5)$$

O procedimento difere um pouco quando é o caso de $n > 2$. Nestas condições, V_n precisa possuir um tamanho n tal que $\frac{n}{2} \in \mathbb{N}$ (onde $\mathbb{N}$ é o conjunto dos números naturais) para tornar possível realizar a transformada de V_n utilizando-se uma expansão da matriz W_2. Em outras palavras, n precisa ser múltiplo de 2 para que esta operação seja possível.

Para se calcular a matriz $E(W_2)_n$ de expansão de W_2, divide-se primeiramente W_2 em dois vetores $R(1)$ e $R(2)$ que são, respectivamente, a primeira e a segunda linha de W_2. Em seguida, gera-se duas matrizes diagonais com $\frac{n}{2}$ colunas, denominadas $D(1)_{\frac{n}{2}\times n}$ e $D(2)_{\frac{n}{2}\times n}$, cujos elementos são matrizes menores de tamanho 1×2, fazendo-se com que estas tenham dimensões $\frac{n}{2} \times n$ no total. As matrizes menores 1×2 que não pertencem à diagonal são matrizes nulas e as que pertencem são, respectivamente para $D(1)_{\frac{n}{2}\times n}$ e $D(2)_{\frac{n}{2}\times n}$, os vetores $R(1)$ e $R(2)$. Por fim, é feita a concatenação vertical destas, com $D(1)_{\frac{n}{2}\times n}$ acima de $D(2)_{\frac{n}{2}\times n}$, gerando a matriz $E(W_2)_n$, também ortogonal pelas suas propriedades. A sequência de operações

pode ser acompanhada pela sequência de equações que vai da Equação (2.6) até a Equação (2.9), onde $0_{1\times 2}$ é definido como o vetor com uma linha e duas colunas cujos elementos têm valor zero.

$$W_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \rightarrow \begin{array}{l} R(1)_{1\times 2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix}, \\ R(2)_{1\times 2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \end{bmatrix}. \end{array} \tag{2.6}$$

$$\begin{aligned} R(1) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix} \rightarrow D(1)_{\frac{n}{2}\times n} &= \frac{1}{\sqrt{2}} \begin{bmatrix} R(1)_{1\times 2} & 0_{1\times 2} & \cdots & 0_{1\times 2} \\ 0_{1\times 2} & R(1)_{1\times 2} & \cdots & 0_{1\times 2} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times 2} & \cdots & 0_{1\times 2} & R(1)_{1\times 2} \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & 1 \end{bmatrix}. \end{aligned} \tag{2.7}$$

$$\begin{aligned} R(2) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \end{bmatrix} \rightarrow D(2)_{\frac{n}{2}\times n} &= \frac{1}{\sqrt{2}} \begin{bmatrix} R(2)_{1\times 2} & 0_{1\times 2} & \cdots & 0_{1\times 2} \\ 0_{1\times 2} & R(2)_{1\times 2} & \cdots & 0_{1\times 2} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times 2} & \cdots & 0_{1\times 2} & R(2)_{1\times 2} \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & -1 \end{bmatrix}. \end{aligned} \tag{2.8}$$

$$E(W_2)_n = \begin{bmatrix} D(1)_{\frac{n}{2}\times n} \\ D(2)_{\frac{n}{2}\times n} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & -1 \end{bmatrix}. \tag{2.9}$$

Finalmente, a matriz $E(W_2)_n$ obtida na Equação (2.9) pode ser usada no lugar de W_2 para calcular a transformada de V_n, conforme mostra a Equação (2.10).

$$
\begin{aligned}
E(W_2)_n V_n &= \begin{bmatrix} D(1)_{\frac{n}{2}\times n} \\ D(2)_{\frac{n}{2}\times n} \end{bmatrix} V_n \\
&= \begin{bmatrix} D(1)_{\frac{n}{2}\times n} V_n \\ D(2)_{\frac{n}{2}\times n} V_n \end{bmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \cdots \\ v_{n-1} \\ v_n \end{bmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} v_1 + v_2 \\ \vdots \\ v_{n-1} + v_n \\ v_1 - v_2 \\ \vdots \\ v_{n-1} - v_n \end{bmatrix} = \begin{bmatrix} L1_{\frac{n}{2}} \\ H1_{\frac{n}{2}} \end{bmatrix} = V'_n.
\end{aligned} \tag{2.10}
$$

Onde os vetores $L1_{\frac{n}{2}}$ e $H1_{\frac{n}{2}}$ são dados, respectivamente, pelas Equações (2.11) e (2.12).

$$
L1_{\frac{n}{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} v_1 + v_2 \\ \vdots \\ v_{n-1} + v_n \end{bmatrix}. \tag{2.11}
$$

$$
H1_{\frac{n}{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} v_1 - v_2 \\ \vdots \\ v_{n-1} - v_n \end{bmatrix}. \tag{2.12}
$$

A transformada inversa é obtida pela inversão da matriz W_2, restaurando o vetor V_2 original sem perda de informação. O exemplo é mostrado na Equação (2.13). O mesmo ocorre para o caso geral entre a matriz $E(W_2)_n$ e um vetor V'_n qualquer que tenha sido obtido pela transformada em questão, como mostra a Equação (2.14).

$$
\begin{aligned}
W_2^{-1}V_2' &= W_2^T V_2' \\
&= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} v_1 + v_2 \\ v_1 - v_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \qquad (2.13)
\end{aligned}
$$

$$
\begin{aligned}
E(W_2)_n^{-1}V_n' &= E(W_2)_n^T V_n' \\
&= \begin{bmatrix} D(1)^T_{\frac{n}{2}\times n} & D(2)^T_{\frac{n}{2}\times n} \end{bmatrix}\begin{bmatrix} L1_{\frac{n}{2}} \\ H1_{\frac{n}{2}} \end{bmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} v_1 + v_2 \\ v_3 + v_4 \\ \vdots \\ v_{n-1} + v_n \\ v_1 - v_2 \\ v_3 - v_4 \\ \vdots \\ v_{n-1} - v_n \end{bmatrix} \\
&= \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix}. \qquad (2.14)
\end{aligned}
$$

2.2.1.1 Decomposição em Sub-Bandas

Observando-se novamente a Equação (2.10), nota-se que os produtos matriciais $D(1)_{\frac{n}{2}\times n}V_n$ e $D(2)_{\frac{n}{2}\times n}V_n$, respectivamente nomeados $L1_{\frac{n}{2}}$ e $H1_{\frac{n}{2}}$, são os responsáveis por gerar dois tipos de respostas bem diferentes. $L1_{\frac{n}{2}}$ é sempre um vetor cujos elementos são o valor médio, multiplicado por um fator $\sqrt{2}$, dos elementos dos vetores obtidos pela divisão de V_n em $\frac{n}{2}$ partes de tamanho $k = 2$. Já $H1_{\frac{n}{2}}$ é um vetor cujos elementos equivalem à metade das diferenças, também multiplicadas por um fator $\sqrt{2}$, entre os elementos destes mesmos blocos.

Um elemento de $L1_{\frac{n}{2}}$, portanto, tem o valor do seu módulo maior quando o par de elementos de V_n que o gera tem o mesmo sinal, enquanto que um elemento de $H1_{\frac{n}{2}}$ tem o valor dos seus módulo maior quando o par de elementos de V_n que o gera tem

o sinal contrário. Por esta razão, a resposta em frequência da transformada pode ser dividia em duas bandas de acordo com a Figura 2.2. As matrizes $L1_{\frac{n}{2}}$ e $H1_{\frac{n}{2}}$ passam a ser denominadas sub-bandas de, respectivamente, baixa e de alta frequência, e os seus elementos passam a receber o nome de coeficientes de sub-banda.

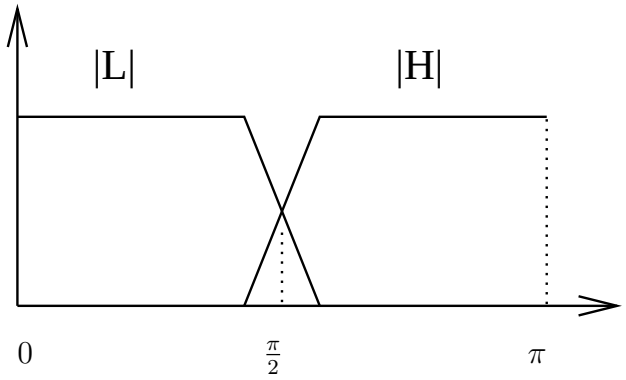

Figura 2.2: Divisão em frequência da banda pela transformada.

Percebe-se que a informação de correlação entre os valores dos elementos de V_n que faziam parte do mesmo bloco de $k = 2$ elementos foi separada pela transformada na forma dos coeficientes de $L1_{\frac{n}{2}}$. As propriedades estatísticas entre os elementos do vetor original, no entanto, podem ser ainda mais explicitadas pela manipulação destes resultados obtidos. Através do mesmo processo de aplicação da transformada, os coeficientes de $L1_{\frac{n}{2}}$ podem ser agrupados novamente em novos blocos de tamanho $k = 2$ para a exploração do equivalente à correlação entre médias locais no vetor V_n.

Enfim, pode-se realizar a transformada de $L1_{\frac{n}{2}}$ da mesma maneira feita anteriormente com V_n. Com este segundo nível de processamento, obtém-se como resultado os coeficientes de $L2_{\frac{n}{4}}$ e $H2_{\frac{n}{4}}$ conforme é mostrado na Equação (2.15).

$$E(W_2)_{\frac{n}{2}} L1_{\frac{n}{2}} = \begin{bmatrix} D(1)_{\frac{n}{4}\times\frac{n}{2}} \\ D(2)_{\frac{n}{4}\times\frac{n}{2}} \end{bmatrix} L1_{\frac{n}{2}} = \begin{bmatrix} D(1)_{\frac{n}{4}\times\frac{n}{2}} L1_{\frac{n}{2}} \\ D(2)_{\frac{n}{4}\times\frac{n}{2}} L1_{\frac{n}{2}} \end{bmatrix}$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} v_1 + v_2 \\ v_3 + v_4 \\ \vdots \\ v_{n-3} + v_{n-2} \\ v_{n-1} + v_n \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} v_1 + v_2 + v_3 + v_4 \\ \vdots \\ v_{n-3} + v_{n-2} + v_{n-1} + v_n \\ v_1 + v_2 - (v_3 + v_4) \\ \vdots \\ v_{n-3} + v_{n-2} - (v_{n-1} + v_n) \end{bmatrix}$$

$$= \begin{bmatrix} L2_{\frac{n}{4}} \\ H2_{\frac{n}{4}} \end{bmatrix}. \tag{2.15}$$

Onde os vetores $L2_{\frac{n}{4}}$ e $H2_{\frac{n}{4}}$ são dados, respectivamente, pelas Equações (2.16) e (2.17).

$$L2_{\frac{n}{4}} = \frac{1}{2} \begin{bmatrix} v_1 + v_2 + v_3 + v_4 \\ \vdots \\ v_{n-3} + v_{n-2} + v_{n-1} + v_n \end{bmatrix}. \tag{2.16}$$

$$H2_{\frac{n}{4}} = \frac{1}{2} \begin{bmatrix} v_1 + v_2 - (v_3 + v_4) \\ \vdots \\ v_{n-3} + v_{n-2} - (v_{n-1} + v_n) \end{bmatrix}. \tag{2.17}$$

A aplicação desta nova transformada equivale a dividir a sub-banda $L1_{\frac{n}{2}}$ em outras duas sub-bandas $L2_{\frac{n}{4}}$ e $H2_{\frac{n}{4}}$, conforme é mostrado na Figura 2.3. Por representar o mesmo tipo de variação que a sub-banda $H1_{\frac{n}{2}}$ mas que ocorre em um intervalo de frequência menor, $H2_{\frac{n}{4}}$ é considerada como sendo uma escala de detecção de detalhes de menor resolução.

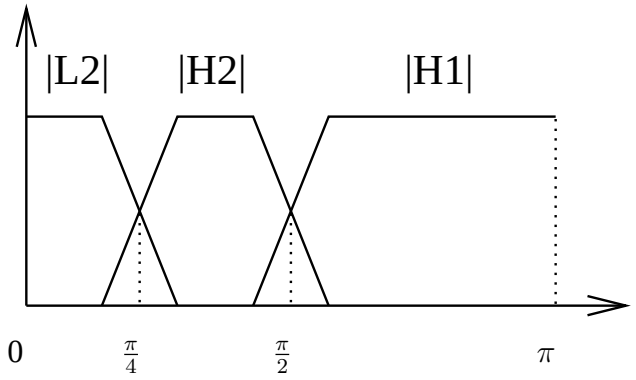

Figura 2.3: Divisão em frequência da banda $L1$ através da segunda transformada.

De uma maneira geral, esta operação pode continuar a ser realizada para a divisão da sub-banda de mais baixa frequência até o limite imposto pela dimensão do vetor V_n, o que pode deixar a sub-banda de mais baixa frequência no final com um

único coeficiente nos casos em que o valor de n seja uma potência de dois. Quando isto ocorre, significa que toda a correlação entre os elementos de V_n possível de ser explorada pela transformada estará sendo representada pela média dos elementos de V_n contida no único coeficiente da sub-banda. Além disto, o conjunto de vetores $H1_{\frac{n}{2}}$, $H2_{\frac{n}{4}}$, ..., possuirá toda a informação que distingue a imagem do seu valor médio.

A restauração da matriz original continua sendo realizada pela Equação (2.14), mas agora esta deve ser feita em um número de passos idêntico ao número de transformadas realizadas, sempre no sentido de restauração das escalas de maior resolução. Por exemplo, em uma decomposição de três níveis, utiliza-se $L3_{\frac{n}{8}}$ e $H3_{\frac{n}{8}}$ para restaurar $L2_{\frac{n}{4}}$, e então $L2_{\frac{n}{4}}$ e $H2_{\frac{n}{4}}$ para restaurar $L1_{\frac{n}{2}}$, concluindo o processo com $L1_{\frac{n}{2}}$ e $H1_{\frac{n}{2}}$ para restaurar V_n.

2.2.2 Transformadas *Wavelet* de Haar Discretas Bidimensionais

As transformadas *wavelet* de Haar discretas bidimensionais funcionam de forma muito semelhante às suas contrapartes unidimensionais, com os dados de entrada agora sendo apresentados no formato de uma matriz quadrada $M_{n\times n}$ ao invés de no formato de um vetor. Esta nova apresentação possui uma característica vantajosa que é a possibilidade de se explorar a correlação espacial entre os seus elementos tanto no eixo vertical quanto no eixo horizontal.

A transformada bidimensional utiliza a mesma série de matrizes W_k da transformada unidimensional. Novamente, o foco da teoria apresentada neste trabalho será dado para $k = 2$ devido à sua maior simplicidade e por ser a maneira implementada. Assim, considerando que a matriz $M_{n\times n}$ simbolize uma imagem, a separação de detalhes ficará disposta em três orientações: uma vertical, uma horizontal e uma diagonal.

Novamente, para ser possível a aplicação da transformada, o valor n das dimensões de $M_{n\times n}$ devem ser tais que $\frac{n}{2} \in \mathbb{N}$. A geração da matriz $E(W_2)_n$ é feita pelo mesmo processo do caso unidimensional, e o cálculo da nova transformada bidimensional é realizado pela expressão $E(W_2)_n M_{n\times n} E(W_2)_2^T$, conforme mostra a Equação (2.18).

$$
\begin{aligned}
E(W_2)_n M_{n\times n} E(W_2)_n^T &= \begin{bmatrix} D(1)_{\frac{n}{2}\times n} \\ D(2)_{\frac{n}{2}\times n} \end{bmatrix} M_{n\times n} \begin{bmatrix} D(1)^T_{\frac{n}{2}\times n} & D(2)^T_{\frac{n}{2}\times n} \end{bmatrix} \\
&= \begin{bmatrix} D(1)_{\frac{n}{2}\times n} M_{n\times n} \\ D(2)_{\frac{n}{2}\times n} M_{n\times n} \end{bmatrix} \begin{bmatrix} D(1)^T_{\frac{n}{2}\times n} & D(2)^T_{\frac{n}{2}\times n} \end{bmatrix}
\end{aligned}
$$

$$= \begin{bmatrix} D(1)_{\frac{n}{2}\times n} M_{n\times n} D(1)^T_{\frac{n}{2}\times n} & D(1)_{\frac{n}{2}\times n} M_{n\times n} D(2)^T_{\frac{n}{2}\times n} \\ D(2)_{\frac{n}{2}\times n} M_{n\times n} D(1)^T_{\frac{n}{2}\times n} & D(2)_{\frac{n}{2}\times n} M_{n\times n} D(2)^T_{\frac{n}{2}\times n} \end{bmatrix}$$
$$= \begin{bmatrix} LL1_{\frac{n}{2}\times\frac{n}{2}} & HL1_{\frac{n}{2}\times\frac{n}{2}} \\ LH1_{\frac{n}{2}\times\frac{n}{2}} & HH1_{\frac{n}{2}\times\frac{n}{2}} \end{bmatrix} = M'_{n\times n}. \tag{2.18}$$

Onde as matrizes $LL1_{\frac{n}{2}\times\frac{n}{2}}$, $HL1_{\frac{n}{2}\times\frac{n}{2}}$, $LH1_{\frac{n}{2}\times\frac{n}{2}}$ e $HH1_{\frac{n}{2}\times\frac{n}{2}}$ são dadas, respectivamente, pelas Equações (2.19), (2.20), (2.21) e (2.22).

$$LL1_{\frac{n}{2}\times\frac{n}{2}} = D(1)_{\frac{n}{2}\times n} M_{n\times n} D(1)^T_{\frac{n}{2}\times n}$$
$$= \frac{1}{2}\begin{bmatrix} m_{1,1}+m_{1,2}+m_{2,1}+m_{2,2} & \cdots & m_{1,n-1}+m_{1,n}+m_{2,n-1}+m_{2,n} \\ \vdots & \ddots & \vdots \\ m_{n-1,1}+m_{n-1,2}+m_{n,1}+m_{n,2} & \cdots & m_{n-1,n-1}+m_{n-1,n}+m_{n,n-1}+m_{n,n} \end{bmatrix}. \tag{2.19}$$

$$HL1_{\frac{n}{2}\times\frac{n}{2}} = D(1)_{\frac{n}{2}\times n} M_{n\times n} D(2)^T_{\frac{n}{2}\times n}$$
$$= \frac{1}{2}\begin{bmatrix} m_{1,1}+m_{2,1}-(m_{1,2}+m_{2,2}) & \cdots & m_{1,n-1}+m_{2,n-1}-(m_{1,n}+m_{2,n}) \\ \vdots & \ddots & \vdots \\ m_{n-1,1}+m_{n,1}-(m_{n-1,2}+m_{n,2}) & \cdots & m_{n-1,n-1}+m_{n,n-1}-(m_{n-1,n}+m_{n,n}) \end{bmatrix}. \tag{2.20}$$

$$LH1_{\frac{n}{2}\times\frac{n}{2}} = D(2)_{\frac{n}{2}\times n} M_{n\times n} D(1)^T_{\frac{n}{2}\times n}$$
$$= \frac{1}{2}\begin{bmatrix} m_{1,1}+m_{1,2}-(m_{2,1}+m_{2,2}) & \cdots & m_{1,n-1}+m_{1,n}-(m_{2,n-1}+m_{2,n}) \\ \vdots & \ddots & \vdots \\ m_{n-1,1}+m_{n-1,2}-(m_{n,1}+m_{n,2}) & \cdots & m_{n-1,n-1}+m_{n-1,n}-(m_{n,n-1}+m_{n,n}) \end{bmatrix}. \tag{2.21}$$

$$HH1_{\frac{n}{2}\times\frac{n}{2}} = D(2)_{\frac{n}{2}\times n} M_{n\times n} D(2)^T_{\frac{n}{2}\times n}$$
$$= \frac{1}{2}\begin{bmatrix} m_{1,1}+m_{2,2}-(m_{1,2}+m_{2,1}) & \cdots & m_{1,n-1}+m_{2,n}-(m_{1,n}+m_{2,n-1}) \\ \vdots & \ddots & \vdots \\ m_{n-1,1}+m_{n,2}-(m_{n-1,2}+m_{n,1}) & \cdots & m_{n-1,n-1}+m_{n,n}-(m_{n-1,n}+m_{n,n-1}) \end{bmatrix}. \tag{2.22}$$

O cálculo da transformada inversa bidimensional é similar ao caso unidimensional, sendo dado pela expressão $E(W_2)_n^{-1} M_{n\times n} (E(W_2)_n^{-1})^T$, conforme é mostrado na Equação (2.23) para o caso geral.

$$
\begin{aligned}
&E(W_2)_n^{-1} M'_{n\times n} (E(W_2)_n^{-1})^T \\
&\quad = E(W_2)_n^T M'_{n\times n} E(W_2)_n \\
&\quad = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \cdots & 0 & 1 & \cdots & 0 \\ 1 & \cdots & 0 & -1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 1 \\ 0 & \cdots & 1 & 0 & \cdots & -1 \end{bmatrix} \begin{bmatrix} LL1_{\frac{n}{2}\times\frac{n}{2}} & HL1_{\frac{n}{2}\times\frac{n}{2}} \\ LH1_{\frac{n}{2}\times\frac{n}{2}} & HH1_{\frac{n}{2}\times\frac{n}{2}} \end{bmatrix} E(W_2)_n \\
&\quad = \frac{1}{\sqrt{2}} \begin{bmatrix} m_{1,1}+m_{1,2} & \cdots & m_{1,n-1}+m_{1,n} & m_{1,1}-m_{1,2} & \cdots & m_{1,n-1}-m_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ m_{n,1}+m_{n,2} & \times & m_{n,n-1}+m_{n,n} & m_{n,1}-m_{n,2} & \cdots & m_{n,n-1}-m_{n,n} \end{bmatrix} \\
&\quad \times \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{bmatrix} \\
&\quad = \begin{bmatrix} m_{1,1} & \cdots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,n} \end{bmatrix} = M_{n\times n}.
\end{aligned}
\tag{2.23}
$$

2.2.2.1 Decomposição em Sub-Bandas para Duas Dimensões

No caso unidimensional, como existia somente uma direção para a qual era possível ocorrer variações de sinal (ao longo do vetor V_n), apenas uma sub-banda de alta frequência era gerada para cada transformada realizada. Como será visto, o mesmo não ocorre para a transformada de duas dimensões, onde para a *wavelet* utilizada é possível a existência de três sub-bandas de alta frequência — já que em um bloco de tamanho 2×2 retirado de $M_{n\times n}$, existem três direções possíveis para seguir de um elemento a um outro qualquer: a horizontal, a vertical e a diagonal.

A observação da matriz $M'_{n\times n}$ resultante da transformada realizada na Equação (2.18) mostra quatro regiões distintas. Estas regiões podem ser separadas em matrizes menores que foram chamadas de $LL1_{\frac{n}{2}\times\frac{n}{2}}$, $HL1_{\frac{n}{2}\times\frac{n}{2}}$, $LH1_{\frac{n}{2}\times\frac{n}{2}}$ e $HH1_{\frac{n}{2}\times\frac{n}{2}}$.

Sendo $G_{n\times n}$ uma matriz quadrada de dimensões de valor n par, tem-se que o

produto $D(1)_{\frac{n}{2}\times n}G_{n\times n}$ fornece a média, multiplicada por um fator $\sqrt{2}$, em cada um dos pares de elementos pertencentes a uma mesma coluna de $G_{n\times n}$; o produto $G_{n\times n}D(1)^T_{\frac{n}{2}\times n}$ fornece a média, multiplicada por um fator $\sqrt{2}$, em cada um dos pares de elementos pertencentes a uma linha de $G_{n\times n}$; o produto $D(2)_{\frac{n}{2}\times n}G_{n\times n}$ fornece a diferença, dividida por um fator $\sqrt{2}$, em cada um dos pares de elementos pertencente a uma mesma coluna de $G_{n\times n}$; e o produto $G_{n\times n}D(2)^T_{\frac{n}{2}\times n}$ fornece a diferença, também dividida por um fator $\sqrt{2}$, em cada um dos pares de elementos pertencentes a uma mesma linha de $G_{n\times n}$.

Portanto, para a matriz $LL1_{\frac{n}{2}\times\frac{n}{2}}$ que é obtida pelo produto $D(1)_{\frac{n}{2}\times n}M_{n\times n}D(1)^T_{\frac{n}{2}\times n}$, ocorre que os seus coeficientes são o dobro da média entre os quatro elementos dos blocos 2×2 obtidos do desmembramento de $M_{n\times n}$. Estes coeficientes terão os seus módulos maiores quando todos os elementos do respectivo bloco 2×2 possuírem o mesmo sinal, o que torna $LL1_{\frac{n}{2}\times\frac{n}{2}}$ o equivalente bidimensional da sub-banda $L1_{\frac{n}{2}}$.

Já no caso da matriz $HL1_{\frac{n}{2}\times\frac{n}{2}}$, obtida pelo produto $D(1)_{\frac{n}{2}\times n}M_{n\times n}D(2)^T_{\frac{n}{2}\times n}$, seus coeficientes são exatamente a diferença entre as médias dos elementos de uma mesma coluna do respectivo bloco 2×2 de $M_{n\times n}$. Os módulos destes coeficientes aumentam quando os elementos pertencentes a uma mesma coluna do seu bloco têm mesmo sinal, mas também quando os elementos de uma mesma linha têm sinais contrários. Portanto, $HL1_{\frac{n}{2}\times\frac{n}{2}}$ é definida como o equivalente bidimensional da sub-banda $H1_{\frac{n}{2}}$ no sentido horizontal.

Similarmente, $LH1_{\frac{n}{2}\times\frac{n}{2}}$, obtida pelo produto $D(2)_{\frac{n}{2}\times n}M_{n\times n}D(1)^T_{\frac{n}{2}\times n}$ possui como coeficientes a diferença entre as médias dos elementos de uma mesma linha do respectivo bloco 2×2 de $M_{n\times n}$. Os módulos destes coeficientes aumentam quando os elementos pertencentes a uma mesma linha do bloco têm mesmo sinal e quando os elementos de uma mesma coluna têm sinais contrários. $LH1_{\frac{n}{2}\times\frac{n}{2}}$ é definida como o equivalente bidimensional da sub-banda $H1_{\frac{n}{2}}$ no sentido vertical.

Por último, $HH1_{\frac{n}{2}\times\frac{n}{2}}$ é obtida pelo produto $D(2)_{\frac{n}{2}\times n}M_{n\times n}D(2)^T_{\frac{n}{2}\times n}$ e tem como valor dos seus coeficientes as diferenças entre as médias dos elementos de uma mesma diagonal do respectivo bloco 2×2 de $M_{n\times n}$, sendo o equivalente bidimensional da sub-banda $H1_{\frac{n}{2}}$ no sentido diagonal.

Da mesma maneira que no caso unidimensional, pode-se aplicar sucessivamente a transformada em $LL1_{\frac{n}{2}\times\frac{n}{2}}$ para uma maior separação da informação de correlação espacial, até o limite em que as dimensões da sub-banda de mais baixa frequência não sejam mais uma potência de dois positiva.

Para ilustrar como é o aspecto visual das sub-bandas no caso bidimensional, realizou-se a transformada numa matriz que representa uma imagem. A imagem original é mostrada na Figura 2.4(a). Seu aspecto quando decomposta em sub-bandas é mostrado na Figura 2.4(b) para uma escala de resolução e na Figura 2.4(c)

para duas. Na Figura 2.4(d) observa-se a disposição das sub-bandas para as duas escalas de resolução. Quanto mais próximos de zero são os coeficientes, mais escuros os seus respectivos pontos na imagem.

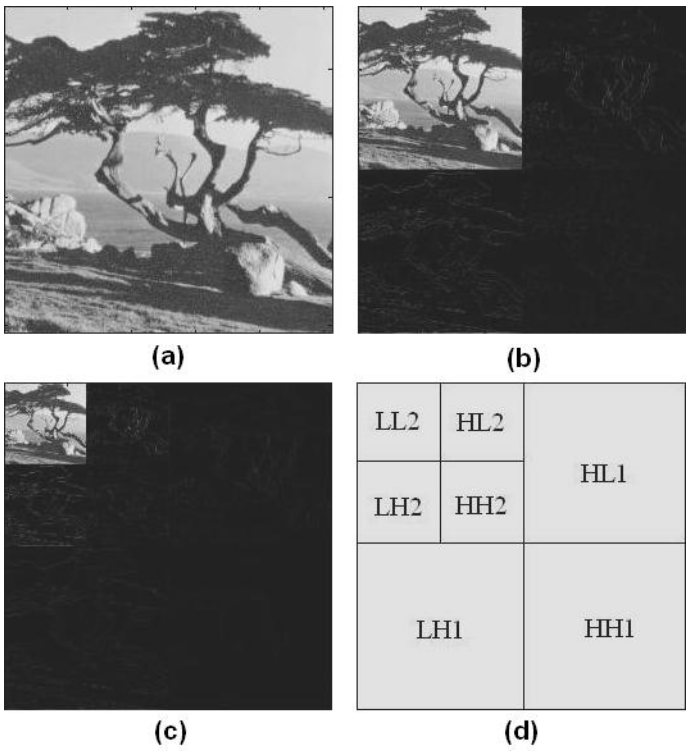

Figura 2.4: Aspecto visual da imagem *tree*.jpg (a) após a primeira aplicação da transformada *wavelet* (b), após a segunda aplicação (c) e a discriminação das sub-bandas por sua localização (d).

2.2.3 Decomposições Computacionalmente Eficientes

O procedimento até aqui mostrado para a transformada *wavelet* teve uma abordagem puramente matemática, o mais didática possível. No entanto, o cálculo destas transformadas geralmente não é realizado exatamente desta maneira, que é considerada computacionalmente pouco eficiente devido às operações de multiplicação e realocação de matrizes de grandes dimensões.

Uma maneira mais eficiente e comumente utilizada na literatura [10] pode ser empregada para a obtenção dos resultados desejados. De fato, as simulações realizadas neste trabalho pela ferramenta matemática MATLAB foram todas produzidas desta outra maneira.

A convolução bidimensional entre uma matriz $M_{n\times n}$ e um operador $F_{t\times t}$ é definida pela Equação (2.24). Elementos inexistentes de $M_{n\times n}$ que eventualmente sejam referenciados são considerados de valor nulo.

$$
\begin{aligned}
M_{n\times n} * F_{t\times t} &= \begin{bmatrix} m_{1,1} & \cdots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,n} \end{bmatrix} * \begin{bmatrix} f_{1,1} & \cdots & f_{1,t} \\ \vdots & \ddots & \vdots \\ f_{t,1} & \cdots & f_{t,t} \end{bmatrix} \\
&= \sum_{i=1}^{t}\sum_{j=1}^{t} \begin{bmatrix} m_{1+(1-i),1+(1-j)}f_{i,j} & \cdots & m_{1+(1-i),n+(t-j)}f_{i,j} \\ \vdots & \ddots & \vdots \\ m_{n+(t-i),1+(1-j)}f_{i,j} & \cdots & m_{n+(t-i),n+(t-j)}f_{i,j} \end{bmatrix}
\end{aligned} \tag{2.24}
$$

Um exemplo simples para $n = 1$ e $t = 2$ pode ser observado na Equação (2.25), caso o leitor prefira entender o resultado da operação de uma maneira mais intuitiva. Ela equivale a inverter horizontalmente e verticalmente o operador $F_{2\times2}$ e deslizá-lo como uma máscara sobreposta a $m_{1,1}$, multiplicando por este seus elementos. Note que, no caso em que $n = 1$, a matriz resultante tem suas dimensões iguais a t.

$$
\begin{aligned}
M_{1\times1} * F_{2\times2} &= \begin{bmatrix} m_{1,1} \end{bmatrix} * \begin{bmatrix} f_{1,1} & f_{1,2} \\ f_{2,1} & f_{2,2} \end{bmatrix} \\
&= \begin{bmatrix} m_{1,1}f_{1,1} & m_{1,1}f_{1,2} \\ m_{1,1}f_{2,1} & m_{1,1}f_{2,2} \end{bmatrix}
\end{aligned} \tag{2.25}
$$

Desta maneira, pode-se entender o resultado de $M_{2\times2} * F_{2\times2}$ como uma soma dos resultados da convolução de $F_{2\times2}$ com cada elemento de $M_{2\times2}$ individualmente, seguida de uma soma que considere os deslocamentos dos resultados de acordo com o posicionamento original do elemento de $M_{2\times2}$. Observe esta operação na Equação (2.26). Note que esta matriz resultante terá sempre dimensões de valor $n + t - 1$, que é um aumento de $t - 1$ em relação ao tamanho original de $M_{n\times n}$, resultante da "sobra"do deslocamento da máscara.

$$
\begin{aligned}
M_{2\times2} * F_{2\times2} &= \begin{bmatrix} m_{1,1}f_{1,1} & m_{1,1}f_{1,2} & 0 \\ m_{1,1}f_{2,1} & m_{1,1}f_{2,2} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & m_{1,2}f_{1,1} & m_{1,2}f_{1,2} \\ 0 & m_{1,2}f_{2,1} & m_{1,2}f_{2,2} \\ 0 & 0 & 0 \end{bmatrix} \\
&+ \begin{bmatrix} 0 & 0 & 0 \\ m_{2,1}f_{1,1} & m_{2,1}f_{1,2} & 0 \\ m_{2,1}f_{2,1} & m_{2,1}f_{2,2} & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & m_{2,2}f_{1,1} & m_{2,2}f_{1,2} \\ 0 & m_{2,2}f_{2,1} & m_{2,2}f_{2,2} \end{bmatrix} = \\
&\begin{bmatrix} m_{1,1}f_{1,1} & m_{1,1}f_{1,2} + m_{1,2}f_{1,1} & m_{1,2}f_{1,2} \\ m_{1,1}f_{2,1} + m_{2,1}f_{1,1} & m_{1,1}f_{2,2} + m_{1,2}f_{2,1} + m_{2,1}f_{1,2} + m_{2,2}f_{1,1} & m_{1,2}f_{2,2} + m_{2,2}f_{1,2} \\ m_{2,1}f_{2,1} & m_{2,1}f_{2,2} + m_{2,2}f_{2,1} & m_{2,2}f_{2,2} \end{bmatrix}
\end{aligned} \tag{2.26}
$$

Como consequência da disposição dos elementos de $M_{n\times n} * F_{2\times 2}$, pode-se realizar uma subamostragem (*downsampling*) destes a um intervalo duplo (meia frequência), descartando-se todos os elementos que pertençam a uma linha ou coluna ímpar, conforme exemplificado pela Equação (2.27) para uma matriz quadrada $U_{n\times n}$ qualquer.

$$U^{2\downarrow}_{\frac{n}{2}\times\frac{n}{2}}(l,c) = U_{n\times n}(2l, 2c). \tag{2.27}$$

Desta maneira, garante-se que um dado elemento de $M_{n\times n}$ aparecerá sempre em um e somente em um elemento de $M_{n\times n} * F_{2\times 2}$, o que já fornece um resultado bem similar ao encontrado nas sub-bandas resultantes da transformada de $M_{n\times n}$, bastando acrescentar um fator multiplicativo de $\frac{1}{2}$ ao operador $F_{2\times 2}$ e escolher seus coeficientes de forma a igualar os resultados obtidos.

Quando $F_{2\times 2}$ é escolhido de forma que $M_{n\times n} * F_{2\times 2}$ seja igual à matriz $LL1_{\frac{n}{2}\times\frac{n}{2}}$ por exemplo, $F_{2\times 2}$ recebe o nome de filtro LL. Da mesma maneira, são denominados filtro HL, filtro LH e filtro HH quando produzem resultados respectivamente iguais às matrizes $HL1_{\frac{n}{2}\times\frac{n}{2}}$, $LH1_{\frac{n}{2}\times\frac{n}{2}}$ e $HH1_{\frac{n}{2}\times\frac{n}{2}}$. Seus valores são fornecidos na Equação (2.28).

$$\begin{aligned}
\text{Filtro LL} &= \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, & \text{Filtro HL} &= \frac{1}{2}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \\
\text{Filtro LH} &= \frac{1}{2}\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, & \text{Filtro HH} &= \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.
\end{aligned} \tag{2.28}$$

A restauração da matriz $M_{n\times n}$ (ou de uma sub-banda LL de maior resolução que uma atual) pode ser realizada de maneira bem simples. Primeiro, realiza-se um aumento da taxa (*upsampling*) para o dobro da frequência para as sub-bandas $LL1_{\frac{n}{2}\times\frac{n}{2}}$, $HL1_{\frac{n}{2}\times\frac{n}{2}}$, $LH1_{\frac{n}{2}\times\frac{n}{2}}$ e $HH1_{\frac{n}{2}\times\frac{n}{2}}$. Uma matriz quadrada $U_{n\times n}$ tem os elementos de sua sobreamostragem definidos de acordo com a Equação (2.29).

$$U^{2\uparrow}_{2n\times 2n}(l,c) = \begin{cases} U_{n\times n}(\frac{l}{2}, \frac{c}{2}) & \text{se l e c forem pares} \\ 0 & \text{do contrário} \end{cases} \tag{2.29}$$

Em seguida, convolui-se as matrizes $LL1^{2\uparrow}_{n\times n}$, $HL1^{2\uparrow}_{n\times n}$, $LH1^{2\uparrow}_{n\times n}$ e $HH1^{2\uparrow}_{n\times n}$ com seus respectivos filtros invertidos tanto horizontalmente quanto verticalmente. A Equação (2.30) ilustra esta operação, com a apóstrofe denotando esta operação de inversão, o que na prática altera somente os filtros HL e LH. A matriz $M_{n+1\times n+1}$ obtida possui uma linha e uma coluna a mais (a primeira, em ambos os casos) em relação à matriz $M_{n\times n}$ original devido à convolução. Ambas devem ser descartadas para assim obter-se a matriz $M_{n\times n}$ original.

$$M_{n+1\times n+1} = 2 \times (LL1^{2\uparrow}_{n\times n} * \text{Filtro LL'} + HL1^{2\uparrow}_{n\times n} * \text{Filtro HL'} + LH1^{2\uparrow}_{n\times n} * \text{Filtro LH'} + HH1^{2\uparrow}_{n\times n} * \text{Filtro HH'}). \quad (2.30)$$

A efetiva implementação destes filtros neste trabalho difere um pouco da literatura utilizada [10]. Optou-se por dividir por dois o fator multiplicativo do filtro LL, visando garantir que os elementos da convolução de $M_{n\times n}$ com o filtro LL sejam efetivamente médias entre quatro valores de $M_{n\times n}$. Além disto, os filtros HL e LH foram implementados de maneira invertida, o que apenas inverte o sinal dos resultado das respectivas sub-bandas. Os novos filtros são mostrados na Equação (2.31), e a nova maneira de se restaurar a imagem original, na Equação (2.32).

$$\text{Filtro LL} = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \qquad \text{Filtro HL} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix},$$
$$\text{Filtro LH} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \qquad \text{Filtro HH} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \quad (2.31)$$

$$M_{n+1\times n+1} = 4 \times LL1^{2\uparrow}_{n\times n} * \text{Filtro LL'} + 2 \times (HL1^{2\uparrow}_{n\times n} * \text{Filtro HL'} + LH1^{2\uparrow}_{n\times n} * \text{Filtro LH'} + HH1^{2\uparrow}_{n\times n} * \text{Filtro HH'}). \quad (2.32)$$

2.2.4 Implementação Física da Decomposição em *Wavelet*

Conhecendo-se os fatores que multiplicam os elementos de $M_{n\times n}$ e a relação de quais dentre estes são necessários para a geração de cada coeficiente das sub-bandas, é possível partir para uma implementação física da transformada que seja capaz de operar com os sinais fornecidos diretamente pelos sensores APS de cada *pixel* para a transformada de maior resolução e com os resultados obtidos por esta e pelas transformadas seguintes para outras resoluções mais grosseiras.

Pela observação da matriz $M'_{n\times n}$ na Equação (2.18), verifica-se que para a determinação dos coeficientes de cada nova escala, é necessário realizar operações que utilizam sempre quatro operandos fisicamente próximos. Mesmo que estas operações realizadas sejam simples (somas, subtrações e produtos), chega-se à conclusão de que um sensor de imagens que implemente este tipo de processamento possuirá um grande número de ligações entre os seus *pixels*.

Uma questão importante seria então definir a localização dos circuitos referentes à determinação de cada coeficiente de cada sub-banda na matriz de *pixels*, sem deixar de levar em consideração que o critério adotado pode elevar o número de

ligações.

Uma tentativa inicial seria manter todos os circuitos de processamento contidos em um único *pixel*, o que reduziria o número de interligações para um valor igual ao número de *pixels* menos um. Mas como os fotodiodos precisam ser igualmente espaçados para se manter uma resolução de captura uniforme ao longo do imageador, tem-se por consequência que o aumento da área ocupada por um *pixel* aumenta a área de todos os *pixels* em relação a uma distribuição mais equitativa.

Por esta razão, neste trabalho propõe-se que a distribuição destes circuitos seja realizada entre todos os *pixels* do imageador da seguinte maneira: primeiramente, cada *pixel* torna-se responsável por fornecer o resultado do cálculo de um coeficiente de uma das matrizes de sub-banda utilizadas, recebendo seu nome de acordo com este. Como nesta implementação será realizada uma transformada com cinco níveis de resolução através de uma matriz de 32 x 32 *pixels*, isto significa que existirá um *pixel* para cada uma das sub-bandas de menor resolução (LL5, HL5, LH5 e HH5), e um número quatro vezes maior de *pixels* para cada escala de resolução imediatamente maior para as altas frequências (HL4, LH4, HH4). As denominações continuam de maneira progressiva para as resoluções maiores - 16 *pixels* para as denominações HL3, LH3 e HH3; 64 para HL2, LH2 e HH2; 256 para HL1, LH1 e HH1 - totalizando o mesmo número de 1024 *pixels* contidos no sensor.

Os *pixels* que determinarem um coeficiente da transformada de qualquer sub-banda em uma dada resolução menor do que a maior resolução possível também serão responsáveis pelo cálculo de pelo menos um coeficiente de cada uma das sub-bandas com resolução maior que sejam necessários à obtenção deste. Por exemplo, o *pixel* que determinar um coeficiente da sub-banda $HL3$ também determinará um coeficiente de $LL2$ que seja necessário para se obter este coeficiente de $HL3$, além de um coeficiente de $LL1$ necessário a obter o coeficiente de $LL2$ utlizado.

Em seguida, tenta-se posicionar da maneira mais próxima o possível os *pixels* que realizam as transformadas para a obtenção das sub-bandas de maior nível de resolução, a fim de se minimizar, na média, o tamanho das trilhas necessárias para as suas interligações. Isto contribui para diminuir as resistências dos fios, que podem sofrer descasamento pelas variações de processo e alterar ligeiramente os resultados do cálculo dos coeficientes da sub-banda $LL1$, o que propagaria o erro para as escalas seguintes.

A Figura 2.5 ilustra o posicionamento dos *pixels* cujos valores amostrados dos seus APS estão envolvidos no cálculo dos coeficientes das sub-bandas de maior nível de detalhes, com setas indicando os quatro *pixels* pertencentes a um mesmo bloco 2×2. Todos os quatro *pixels* pertencentes ao mesmo bloco recebem uma cópia da corrente de saída de cada sensor APS dos outros três e as utiliza, juntamente com a sua própria amostra, para realizar o cálculo do seu coeficiente.

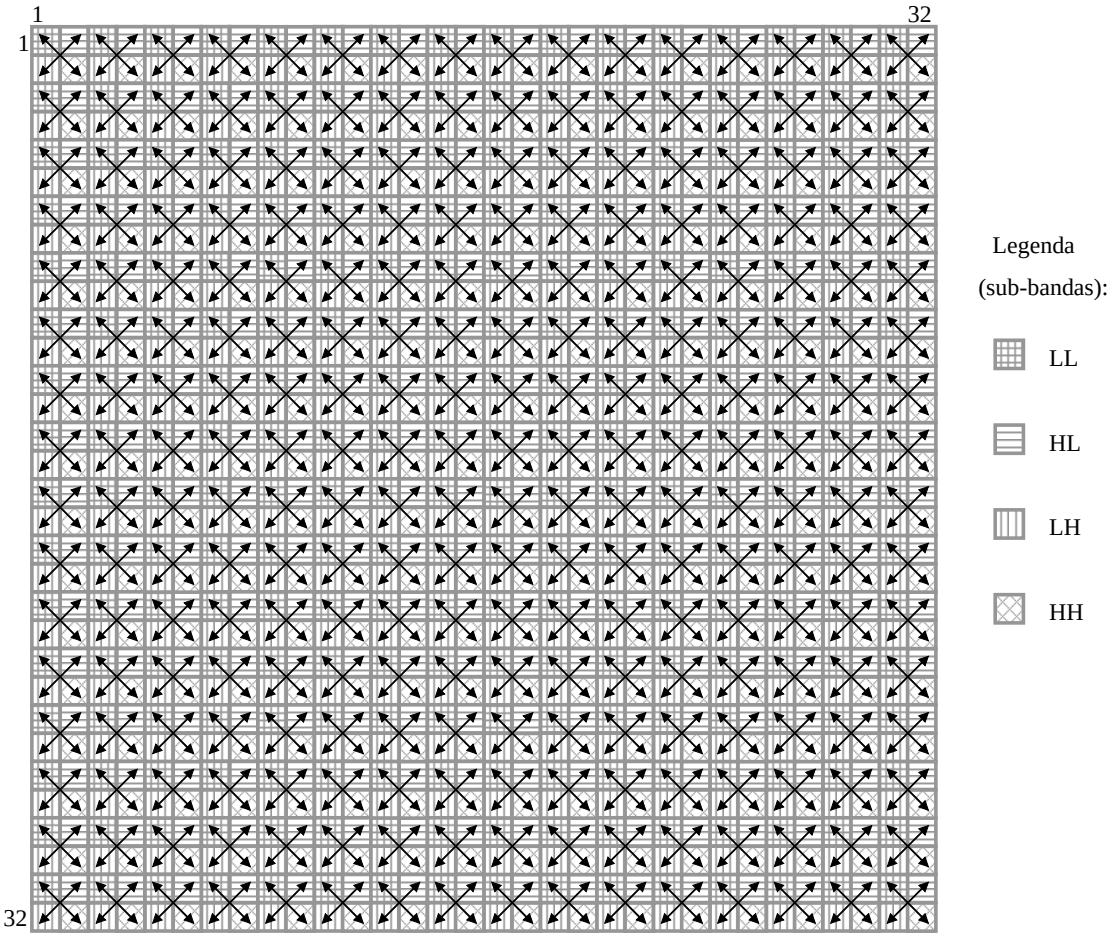

Figura 2.5: Disposição dos circuitos implementados para o cálculo da transformada *wavelet* em uma matriz de 32 x 32 *pixels* referentes à primeira transformada.

Da mesma maneira, na Figura 2.6 observa-se o posicionamento dos *pixels* que geram os coeficientes obtidos pela segunda transformada. É notável que os *pixels* pertencentes a um mesmo bloco 2×2 agora estão ao dobro da distância de um em relação ao outro quando se compara com a etapa anterior. Os sinais compartilhados entre os quatro elementos deste novo bloco são agora as correntes que representam os coeficientes obtidos pela transformada anterior (o equivalente aos elementos da matriz de $LL1$), no lugar das correntes de saída dos sensores APS.

Assim se prossegue para a obtenção dos coeficientes resultantes da terceira transformada (Figura 2.7), da quarta transformada (Figura 2.8) e da quinta (Figura 2.9), com o distanceamento entre os *pixels* pertencentes a um mesmo bloco 2×2 sempre tendo a distância que os separa dobrada. Na última escala de menor detalhamento, efetivamente se implementa o *pixel* com o único coeficiente referente à sub-banda $LL5$ no elemento da primeira linha e primeira coluna do imageador.

Uma conclusão lógica deste arranjo é que os *pixels* são estruturalmente diferentes entre si e portanto têm tamanhos variáveis uns em relação aos outros. Os *pixels* de maior escala de resolução serão responsáveis pelo processamento mais simples (o que

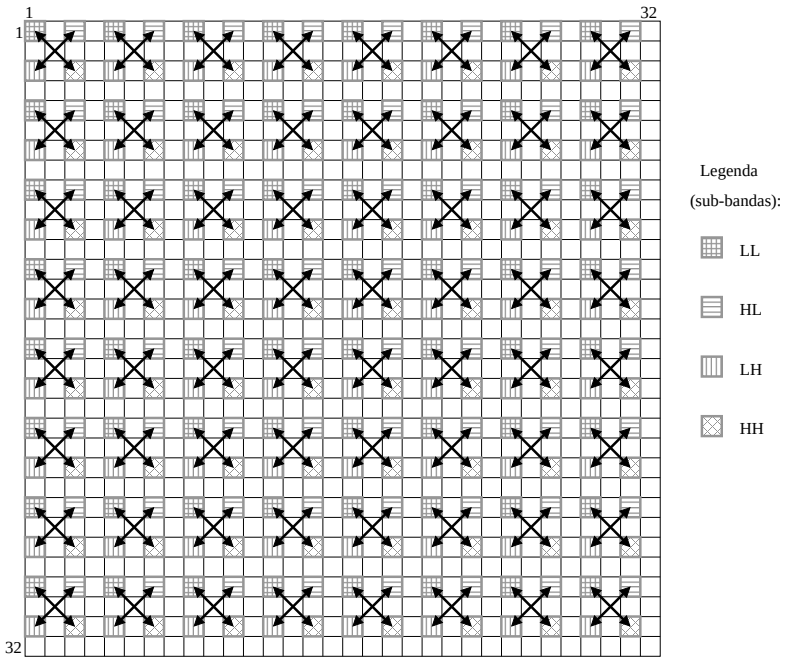

Figura 2.6: Disposição dos circuitos implementados para o cálculo da transformada *wavelet* em uma matriz de 32 x 32 *pixels* referentes à segunda transformada.

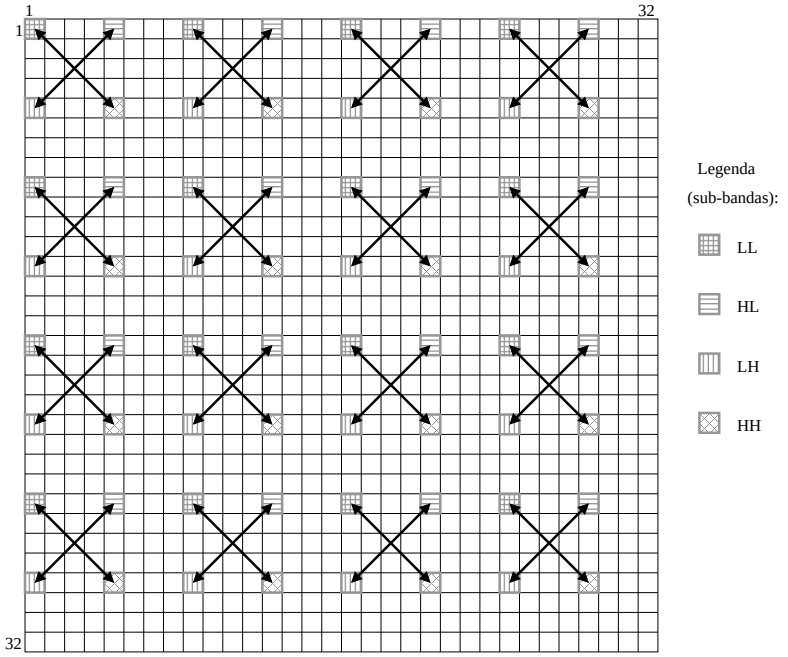

Figura 2.7: Disposição dos circuitos implementados para o cálculo da transformada *wavelet* em uma matriz de 32 x 32 *pixels* referentes à terceira transformada.

sugere um menor espaço ocupado) e como $\frac{3}{4}$ do espaço físico do sensor de imagem é ocupado por este tipo de *pixel*, permite-se que um *pixel* responsável pelo cálculo de coeficiente de escala de detalhes mais grosseira "invada"esta região excedente. Maiores detalhes sobre a estrutura de cada *pixel* são vistos no Capítulo 4.

Apesar de a transformada *wavelet* ter como função apenas a representação de

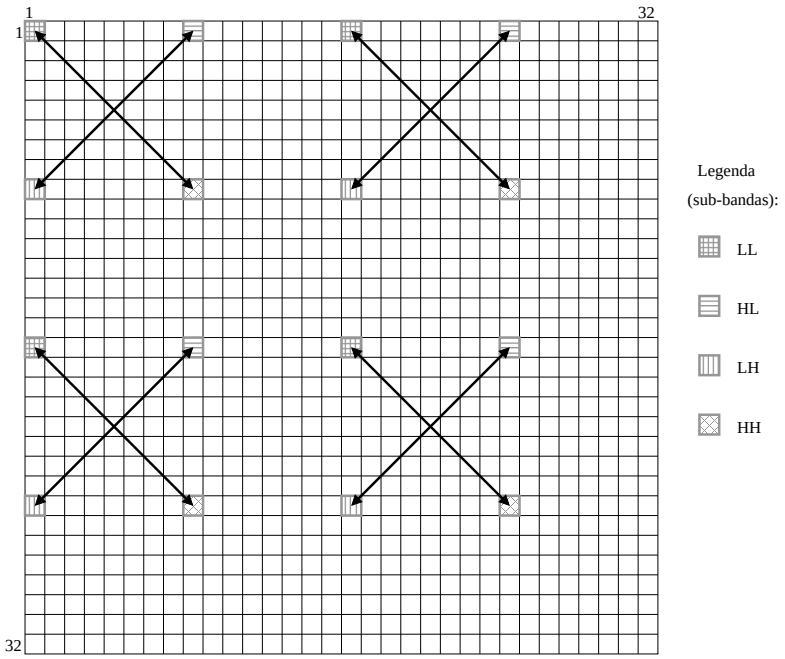

Figura 2.8: Disposição dos circuitos implementados para o cálculo da transformada *wavelet* em uma matriz de 32 x 32 *pixels* referentes à quarta transformada.

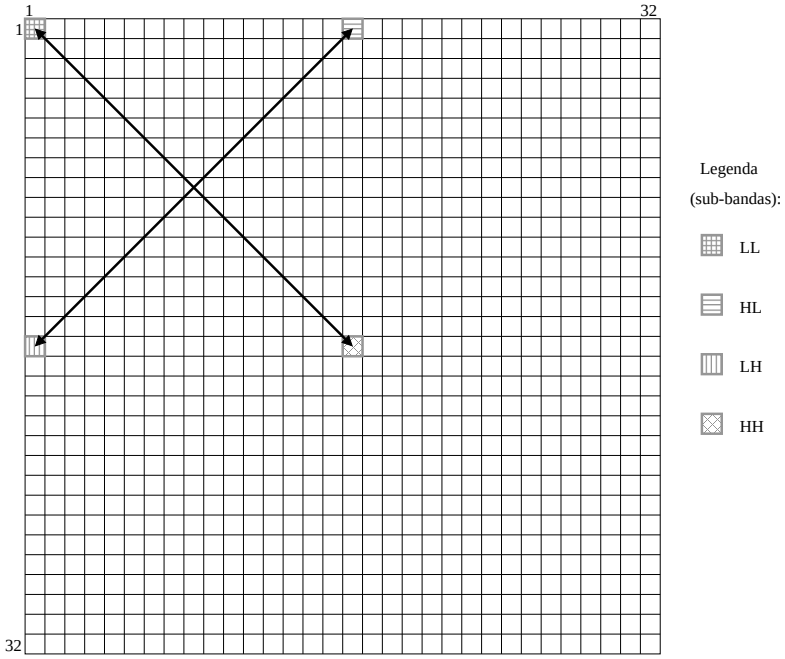

Figura 2.9: Disposição dos circuitos implementados para o cálculo da transformada *wavelet* em uma matriz de 32 x 32 *pixels* referentes à quinta transformada.

uma imagem em um ambiente multiresolução onde as suas propriedades estatísticas estejam explicitamente separadas, uma compressão sem perdas desta imagem pode ser realizada pela utilização de um código de entropia. Existem ainda alguns algoritmos com perdas que exploram as propriedades das transformadas para atingir uma taxa de compressão ainda maior, como o que será visto no próximo capítulo.

Capítulo 3

O Algoritmo *Embedded Zerotree Wavelet*

O algoritmo *Embedded Zerotree Wavelet* [7] foi criado por Jerome M. Shapiro em 1993 no intuito de desenvolver um método de comprimir imagens capaz de obter sua melhor qualidade possível para uma dada taxa de compressão, de maneira que todas as codificações da mesma imagem feitas com uma taxa de *bits* menor estejam totalmente incluídas no início da sequência de *bits* maior.

O algoritmo EZW contém as seguintes características: aplicação de uma transformada *wavelet* discreta, que fornece uma representação compacta da imagem em multiresolução; uma codificação em árvores de zeros, que permite a predição de coeficientes de valor insignificante ao longo das escalas de resolução da imagem decomposta; uma etapa de aproximação sucessiva, a fim de facilitar a codificação progressiva dos coeficientes a serem transmitidos; um protocolo de priorização de transmissão, que organiza os coeficientes da transformada de imagem em ordem de importância pela sua resolução, pela sua localização e pela sua magnitude; um código de entropia, para uma compressão adicional, sem perdas, após a etapa de quantização; a capacidade de terminar a qualquer momento seu processamento, assim que a taxa de compressão desejada for exatamente atingida.

3.1 Árvores de Zeros

Para se obter uma maior taxa de compressão pelo algoritmo apresentado, de maneira que a imagem comprimida seja representada com boa qualidade visual e de modo eficiente, uma nova estrutura de dados denominada árvore de zeros é definida. Esta estrutura funciona pela organização dos coeficientes de todas as escalas de alta frequência para uma mesma orientação em uma ou mais árvores hierárquicas.

Nestas árvores, os coeficientes da escala com o nível de detalhes mais grosseiro

ocupam as posições referentes às raizes, sendo que cada um destes coeficientes é "pai"de um conjunto formado por quatro coeficientes da escala imediatamente superior à sua em nível de detalhes e de mesma orientação. Por sua vez, cada um dos quatro "filhos"também tem como filhos quatro coeficientes da escala imediatamente superior à sua em nível de detalhes e de mesma orientação. Esta organização continua por classificar hierarquicamente os coeficientes de maneira sucessiva por este critério, até que se atinja a escala de maior nível de detalhes para a transformação realizada.

Desta maneira, tem-se que o número de árvores obtidas é sempre igual ao total do número de coeficientes referentes à escala de menor nível de detalhes das três sub-bandas HL, LH e HH. Para um dado pai, os elementos do conjunto de todos os seus filhos, e sucessivamente os dos filhos dos seus filhos, são chamados de seus descendentes. Similarmente, um pai é chamado de ancestral em relação a cada coeficiente que seja um descendente seu. Coeficientes que pertençam à escala de maior nível de detalhes (e que portanto não possuam descendentes) recebem o nome de folhas. Coeficientes que não são nem raízes e nem folhas são denominados galhos.

A relação entre quais coeficientes são pais de quais coeficientes é espacial e explora a correlação que existe entre a ocorrência da detecção de algum detalhe em um ponto para uma dada escala grosseira e a ocorrência da detecção do mesmo em uma escala de detalhes mais fina. A Figura 3.1 mostra alguns exemplos destas dependências em uma imagem decomposta em três escalas de maneira similar à Figura 2.4(c). Observando-se esta imagem pela visão de uma árvore hierárquica, tem-se que as setas apontam dos pais para os filhos.

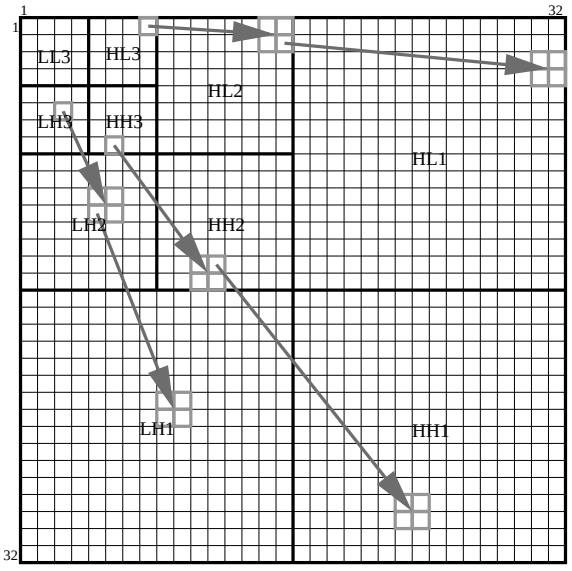

Figura 3.1: Exemplos de relação entre pais e filhos ao longo das sub-bandas.

Como os coeficientes obtidos para as sub-bandas de alta frequência são todos determinados por uma diferença entre a combinação de somas dois a dois entre um total de quatro coeficientes pertencentes a uma mesma sub-banda com uma escala de detalhes mais fina (ou mesmo da imagem original), e assumindo-se que estes coeficientes são independentes estatisticamente uns dos outros, presume-se que a distribuição probabilística do resultado desta operação tenha seu valor médio em zero e apresente uma simetria em relação a este ponto, já que estes resultados podem ser positivos ou negativos.

Assim, pode-se definir que um dado coeficiente de valor x é insignificante em relação a um limiar T se $|x| < T$. A idéia da árvore de zeros se baseia em evidências empíricas para assumir que se um coeficiente de uma sub-banda de alta frequência em uma escala grosseira de detalhes for insignificante em relação a um dado limiar T, então todos os coeficientes descendentes deste na árvore de hierarquia têm alta probabilidade de também serem insignificantes em relação a T. Coeficientes insignificantes são considerados como tendo valor nulo, e utilizam símbolos especiais para serem representados, não necessitando portanto terem seus valores quantizados.

Para a classificação de significância destes coeficientes, a ordem de varedura dos mesmos é realizada nas escalas de detalhe no sentido das mais grosseiras para as mais finas. Define-se que um coeficiente é um elemento de uma árvore de zeros para um dado limiar T quando ele e todos os seus filhos forem insignificantes. Este coeficiente é definido como uma raiz de árvore de zeros para este mesmo limiar se ele não possuir nenhuma raiz de árvore de zeros como ancestral. Um coeficiente que seja uma raiz de árvore de zeros deve ser codificado com um símbolo especial para sinalizar que a insignificância dos seus descendentes é previsível, e portanto os mesmos têm seus valores presumidamente nulos. Caso este coeficiente seja insignificante mas não pertença a uma árvore de zeros, ou seja, caso ele possua pelo menos um filho significativo, o mesmo é codificado com um outro símbolo especial. Um resumo do processo de classificação dos coeficientes é mostrado no fluxograma da Figura 3.2.

A vantagem da utilização de árvores de zeros é notória. Mesmo para valores de T baixos, como em imagens naturais (em oposição às geradas artificialmente) geralmente existem grandes superfícies com poucos detalhes ou mesmo nenhum, um grande número de coeficientes acaba por ser classificado como insignificante. Se os símbolos que representam as duas possibilidades de codificação destes elementos insignificantes forem dados por poucos *bits*, então ocorrerá uma grande redução no espaço necessário para se armazenar uma imagem. Não apenas isto, a classificação de um coeficiente como raiz de uma árvore de zeros acaba também por tornar desnecessário a transmissão ou armazenamento de todos os seus descendentes (eles serão considerados de valor nulo), gerando uma compressão ainda maior. O custo para os coeficientes significativos é de apenas um *bit* acrescentado à sua quantização normal,

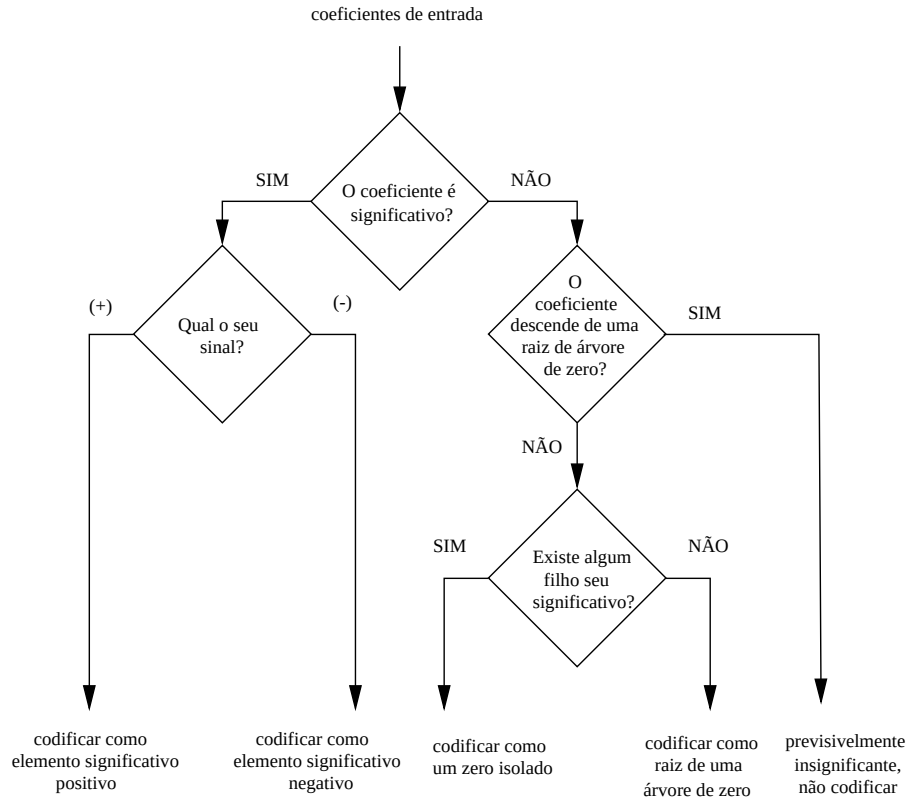

Figura 3.2: Fluxograma de codificação dos coeficientes em uma árvore de zeros.

necessário para distinguí-los dos coeficientes insignificantes.

3.1.1 Exemplo de Aplicação de Árvore de Zeros

É oferecido como um exemplo ao leitor a Figura 3.3 referente à aplicação da árvore de zeros para as sub-bandas de orientação HL de uma imagem decomposta em três níveis de resolução. Os valores referentes a este exemplo se encontram na Tabela 3.1.

Tabela 3.1: Valores para exemplo de aplicação de árvore de zeros.

Coeficiente	Valor	Coeficiente	Valor
$HL3_{1,1}$	8	$HL1_{2,3}$	-1
$HL2_{1,1}$	-5	$HL1_{2,4}$	0
$HL2_{1,2}$	2	$HL1_{3,1}$	-5
$HL2_{2,1}$	-1	$HL1_{3,2}$	-2
$HL2_{2,2}$	6	$HL1_{4,1}$	1
$HL1_{1,1}$	-2	$HL1_{4,2}$	-2
$HL1_{1,2}$	-4	$HL1_{3,3}$	2
$HL1_{2,1}$	-5	$HL1_{3,4}$	2
$HL1_{2,2}$	3	$HL1_{4,3}$	1
$HL1_{1,3}$	2	$HL1_{4,4}$	0
$HL1_{1,4}$	1		

Considere um limiar $T = 3$. O primeiro passo é comparar o valor do coeficiente $HL3_{1,1}$ com T. Como o coeficiente tem o seu módulo maior que o limiar e tem

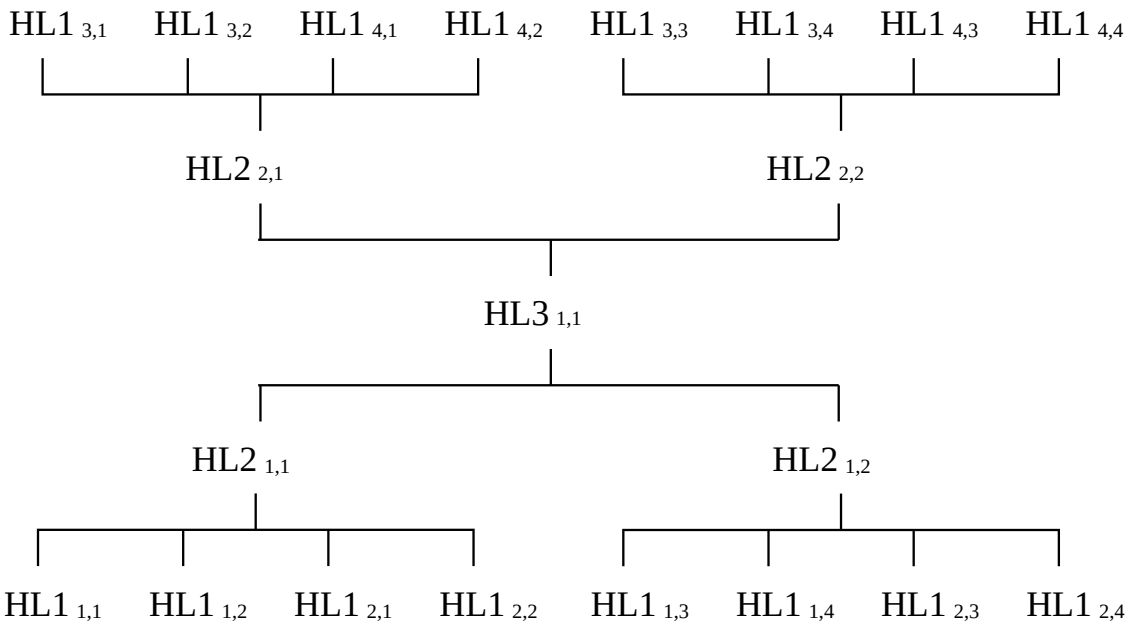

Figura 3.3: Exemplo de aplicação de árvore de zeros em sub-bandas de orientação HL.

valor maior do que zero, $HL3_{1,1}$ é codificado como um elemento significativo e positivo, com o valor do seu módulo quantizado acrescentado logo depois. Em seguida, compara-se, em qualquer ordem, os coeficientes $HL2_{1,1}$, $HL2_{1,2}$, $HL2_{2,1}$ e $HL2_{2,2}$ com T.

Verifica-se que os coeficientes $HL2_{1,1}$ e $HL2_{2,2}$ são ambos significativos, mas respectivamente negativo e positivo, sendo codificados de maneira similar a $HL3_{1,1}$. Os coeficientes $HL2_{1,2}$ e $HL2_{2,1}$, no entanto, são ambos insignificantes, e ainda não há informação suficiente para classificá-los como uma raiz de árvore de zeros ou um zero isolado. Portanto, o próximo passo é verificar $HL1_{1,3}$, $HL1_{1,4}$, $HL1_{2,3}$ e $HL1_{2,4}$ para $HL2_{1,2}$ e $HL1_{3,1}$, $HL1_{3,2}$, $HL1_{4,1}$ e $HL1_{4,2}$ para $HL2_{2,1}$ para descobrir se entre cada grupo existe ao menos um elemento significativo.

No caso de $HL2_{1,2}$ temos que $|HL1_{3,1}| > T$, e conclui-se que $HL2_{1,2}$ deve ser codificado como um zero isolado, sem ter seu sinal ou quantização de seu módulo acrescentados a esta codificação. Já $HL2_{2,1}$ não possui nenhum filho com módulo maior do que T, e deve ser codificado como uma raiz de árvore de zeros, também sem ter seu sinal ou quantização de seu módulo acrescentados a esta.

O passo final é verificar os coeficientes da sub-banda $HL1$, com a exceção dos coeficientes $HL1_{3,1}$, $HL1_{3,2}$, $HL1_{4,1}$ e $HL1_{4,2}$, que são presumidamente insignificantes e portanto não serão codificados. Note que todos os coeficientes de $HL1$ são folhas da árvore, e como tais não podem ser raízes de árvore de zeros, sendo classificados somente como zeros isolados caso seus módulos sejam inferiores ao limiar T.

Por fim, considere que cada um destes coeficientes do exemplo seja codificado

utilizando um dicionário com palavras de tamanho constante e igual a 8 *bits*, incluído nestes o *bit* de sinal. Como existem 21 coeficientes, seria necessário um total de 168 *bits* para o armazenamento dos mesmos. No exemplo mostrado, existe um total de 6 coeficientes significativos (positivos ou negativos), 1 coeficiente que é uma raiz de árvore de zeros, 4 coeficientes presumidamente insignificantes que não devem ser codificados e 10 coeficientes que são zeros isolados.

Utilizando-se como prefixo na codificação um *bit* para determinar se um coeficiente é significativo ou não, e um outro *bit* adicional para determinar se os coeficientes insignificantes são raízes de árvores de zero ou zeros isolados (num total de dois *bits* para os coeficientes insignificantes), além do mesmo dicionário de codificação de sinal e magnitude utilizado para o caso sem árvores de zeros quando o mesmo for aplicável, calcula-se um total de 76 *bits* para o armazenamento dos dados referentes ao exemplo, uma taxa de compressão de aproximadamente 2.2.

3.2 Aproximações Sucessivas

Um dos objetivos do algoritmo consiste em se obter exatamente uma determinada taxa de compressão. A utilização de árvores de zeros permite ao algoritmo realizar uma compressão poderosa, sem no entanto ser capaz de controlar a taxa da mesma. Felizmente, a manipulação do valor do limiar T de maneira iterativa fornece a possibilidade de se aproximar deste objetivo.

Após a codificação dos coeficientes de uma imagem utilizando-se um limiar T qualquer, o total de *bits* necessários para se representar a mesma pode ser facilmente determinado, bem como a sua taxa de compressão. Imagine que, para um dado valor do limiar T, chegue-se a uma taxa de compressão ligeiramente inferior à desejada. Neste caso, basta aumentar um pouco o valor deste limiar e realizar a recodificação dos coeficientes. Como dependendo da maneira de implementação do algoritmo pode ocorrer que a variação de T possa ser realizada somente de maneira quantizada, o valor ideal para o limiar será aquele ao qual o menor incremento possível de seu valor geraria uma taxa de compressão melhor do que a desejada. Isto porque esta taxa atual — abaixo da alvejada — ainda pode ser melhorada, conforme será visto na seção seguinte.

3.3 Ordem de Transmissão dos Coeficientes

A última etapa do algoritmo consiste na sequência de envio dos coeficientes já codificados, para sua transmissão ou armazenamento. Mesmo que seja improvável que com a escolha de um determinado limiar T feita através de um processo iterativo atinja-se exatamente a taxa de compressão desejada, é possível organizar a ordem

de transmissão dos coeficientes das sub-bandas de forma que os mais importantes sejam escaneados no início e os menos importantes no final. Quando a ordem de escaneamento é definida desta maneira, o código ganha uma alta escalabilidade: pode-se determinar que um número dentre os últimos coeficientes sejam simplesmente descartados, não necessitando serem transmitidos ou armazenados, para que se atinja exatamente a taxa de compressão desejada e sem perda de sincronismo para o receptor.

Uma ordem de importância está implícita na própria maneira de se restaurar uma imagem decomposta por uma *wavelet*. Os coeficientes da sub-banda de mais baixa frequência (LL), por representarem a própria imagem que se quer reconstituir em uma resolução menor, adquirem maior importância entre todos os coeficientes, e devem ser escaneados primeiro. A sequência dos coeficientes deste grupo não é importante, desde que o receptor tenha conhecimento da mesma. Por exemplo, pode-se transmitir da primeira à última linha, da esquerda para a direita.

Em seguida, vêm os coeficientes das sub-bandas de resolução imediatamente superior, referentes às escalas mais grosseiras em nível de detalhes. Em cada sub-banda, a ordem de escaneamento dos seus coeficientes deve ser conhecida pelo receptor, preferencialmente sendo em todas igual à ordem da sub-banda LL por uma questão de simplificação. Quanto à ordem entre as sub-bandas em si, como de maneira geral as sub-bandas cuja orientação de detalhes seja nas diagonais (HH) têm menos energia em comparação às sub-bandas cujas orientações de detalhes sejam na horizontal (HL) e na vertical (LH), os seus coeficientes devem ser os últimos a serem escaneados, de forma a causar uma menor perda em detalhes na restauração final da imagem caso os mesmos sejam descartados. A preferência sobre qual das sub-bandas restantes deve ser a primeira da sequência não faz diferença, bastando ser, como sempre, conhecida pelo receptor.

A preferência por esta ordenação é porque, com apenas estes coeficientes, já é possível restaurar uma imagem com o dobro da resolução em comparação à obtida apenas pela sub-banda LL. Exatamente por esta questão, a ordem de transmissão ou armazenamento segue o sentido dos coeficientes das sub-bandas de menor resolução para as de maior, com a sub-banda HH sendo sempre a última a ser transmitida entre as sub-bandas de uma mesma escala de detalhes. A Figura 3.4 mostra a sugestão de ordenação proposta por Shapiro [7]. O sentido das setas se dá na direção dos últimos coeficientes a serem escaneados.

3.4 Implementação Física do Algoritmo

Na implementação realizada por este trabalho, as árvores de hierarquia entre os coeficientes das sub-bandas de alta frequência e mesma orientação estão dispostas

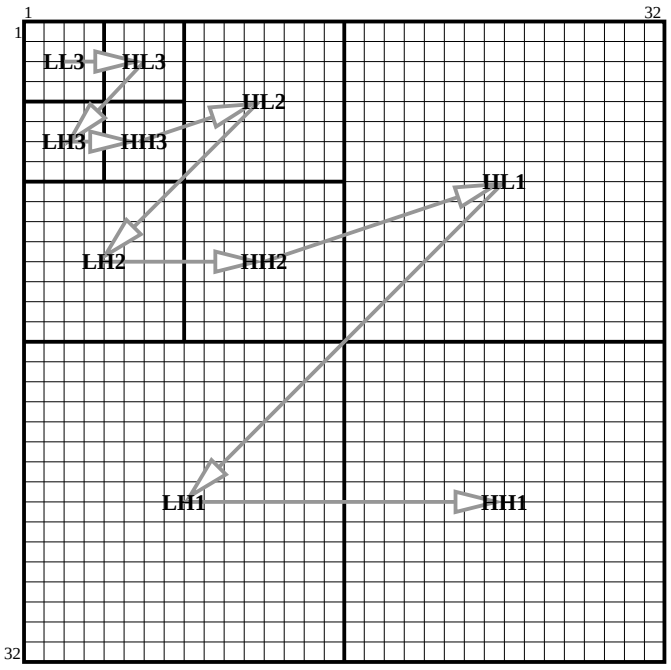

Figura 3.4: Sugestão de ordem do escaneamento das sub-bandas.

de maneira diferente da representada pela Figura 3.1, já que a própria disposição dos sinais que representam estes coeficientes na matriz é diferente e segue a maneira apresentada nas Figuras 2.5, 2.6, 2.7, 2.8 e 2.9.

Devido à existência de apenas um coeficiente para as sub-bandas de menor resolução nesta implementação, existirá uma única árvore para cada tipo de orientação de detalhamento possível. A estas árvores daremos os respectivos nomes de árvore HL, LH e HH. As Figuras 3.5 a 3.8 ilustram simultaneamente estas três árvores, com as setas partindo de cada pai para o seus respectivos quatro filhos. Para uma melhor visualização das àrvores, a representação das mesmas foi quebrada em uma imagem para cada nível de hierarquia. A Figura 3.5 ilustra a relação hierárquica entre os coeficientes de sub-bandas de alta frequência e mesma orientação de detalhes obtidos pela aplicação da primeira e da segunda transformada *wavelet*, enquanto que a Figura 3.6 faz o mesmo para os coeficientes obtidos pela aplicação da segunda e da terceira transformada, a Figura 3.7 o faz para os coeficientes obtidos pela terceira e pela quarta e a Figura 3.8 o faz para os coeficientes obtidos pela quarta e pela quinta.

Um valor de corrente analógico equivalente ao limiar T é fornecido a cada *pixel* do imageador para a comparação com o valor do sinal do coeficiente contido no mesmo, gerando um sinal equivalente à classificação deste coeficiente em significativo ou insignificante. As ligações físicas equivalentes às mostradas nas Figuras 3.5 até 3.8 entre os *pixels* são implementadas neste trabalho para permitir que um *pixel* possa ler estes sinais referentes à significância dos coeficientes dos seus filhos. Isto permite

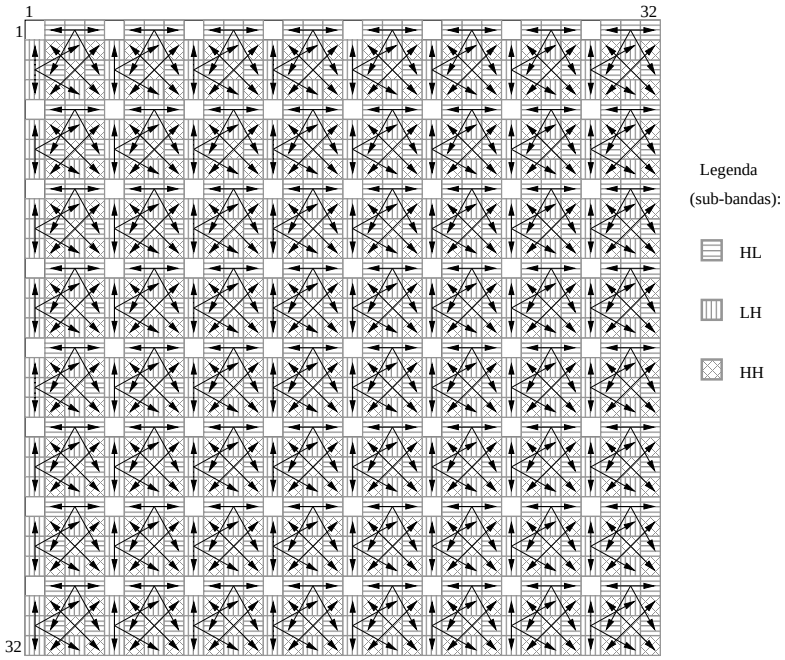

Figura 3.5: Hierarquização entre os coeficientes das sub-bandas de alta frequência e de mesma orientação de detalhes, obtidos pela aplicação da primeira e da segunda transformada.

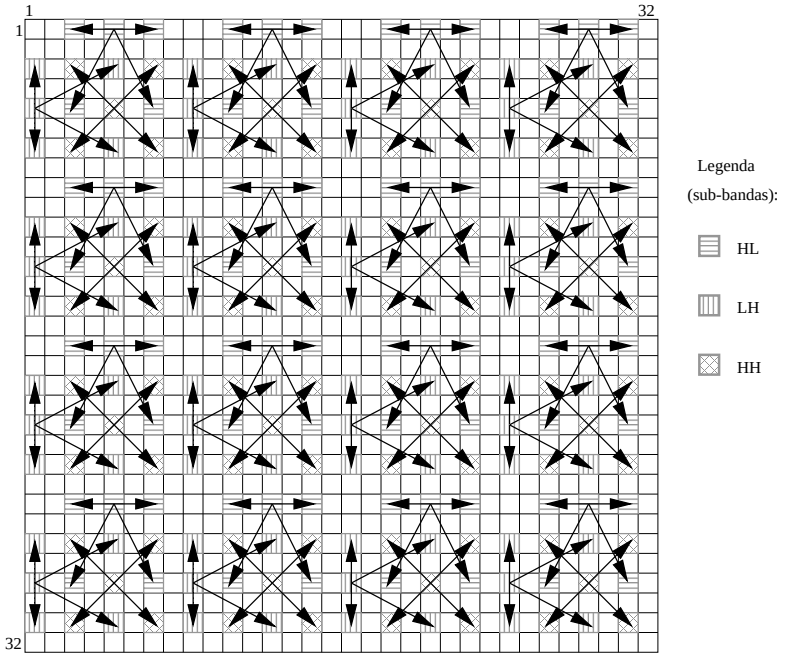

Figura 3.6: Hierarquização entre os coeficientes das sub-bandas de alta frequência e de mesma orientação de detalhes, obtidos pela aplicação da segunda e da terceira transformada.

a classificação de cada coeficiente como um zero isolado ou como uma raiz de uma árvore de zeros pelo próprio *pixel*.

Apesar de esta implementação permitir que um *pixel* possa se auto-classificar como raiz de uma árvore de zeros mesmo este tendo um pai também classificado

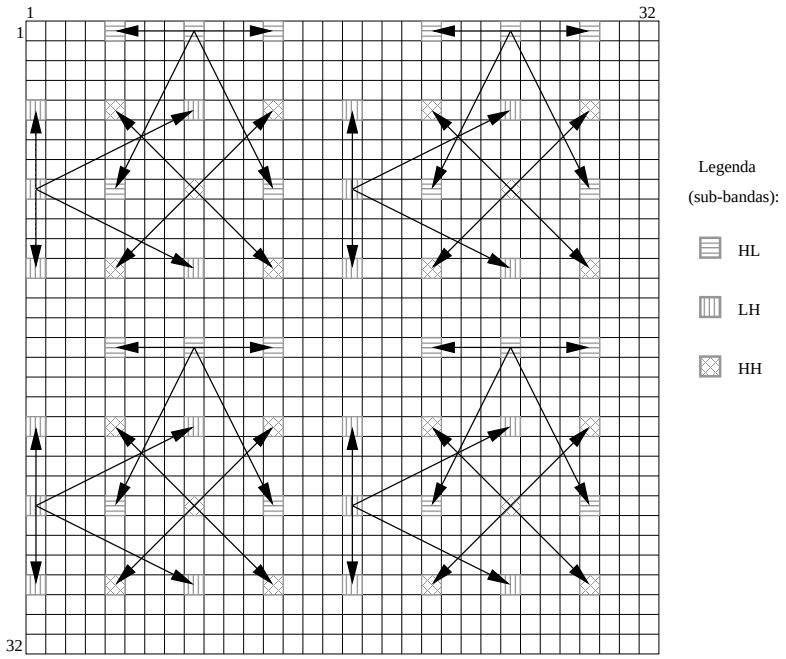

Figura 3.7: Hierarquização entre os coeficientes das sub-bandas de alta frequência e de mesma orientação de detalhes, obtidos pela aplicação da terceira e da quarta transformada.

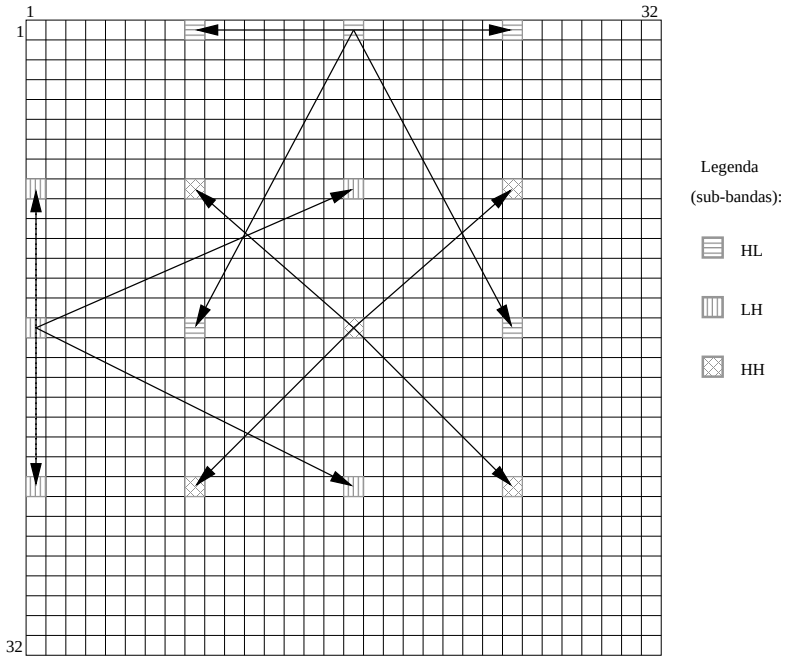

Figura 3.8: Hierarquização entre os coeficientes das sub-bandas de alta frequência e de mesma orientação de detalhes, obtidos pela aplicação da quarta e da quinta transformada.

como uma raiz de uma árvore de zeros, como o imageador é projetado de forma que um microcontrolador externo seja capaz de acessar individualmente cada *pixel* seu, pode-se delegar ao microcontrolador a tarefa de controle da ordem da leitura, fazendo com que o mesmo "pule"a leitura de todos os descendentes das raízes de

árvores de zeros que, conforme foi visto, não devem ser codificados.

Por fim, como as ligações são construídas de maneira paralela, para uma mesma amostragem do imageador basta uma mudança no valor do limiar T para que os circuitos que realizam a comparação deste com os coeficientes alterem o resultado da significância dos mesmos de imediato, o que também pode acabar mudando a classificação destes para zeros isolados ou raízes de árvores de zeros. Logo, é interessante que o microcontrolador externo também seja o responsável por determinar o valor do limiar T, pois assim este pode realizar as aproximações sucessivas necessárias ao algoritmo. Para tanto, um conversor digital-analógico é incluído no imageador.

Capítulo 4

Circuitos Usados e Simulações

Neste capítulo serão apresentados os circuitos utilizados neste trabalho. Os circuitos serão detalhados através da visualização dos seus respectivos esquemáticos utilizados, juntamente de uma breve descrição sobre o seu funcionamento. Para os circuitos de maior complexidade, será mostrada também a sua simulação elétrica para parâmetros típicos e uma simulação de Monte Carlo realizada com dez rodadas de variações estatísticas, todas elas realizadas levando-se em consideração tanto as possíveis variações de processo quanto os erros de descasamento entre os transistores.

Todos os circuitos foram projetados na tecnologia CMOS 0.35 μm da AMS (*austriamicrosystems*) através do *software* CADENCE Custom IC Design Tools e simulados através do CADENCE Spectre utilizando-se os modelos de transistores fornecidos pelo fabricante. A versão da tecnologia utilizada foi para uma tensão de alimentação $V_{DD} = 3.3$ V, que será omitida dos esquemáticos mostrados.

4.1 Visão Geral do Circuito

A Figura 4.1 mostra uma visão geral do circuito implementado.

Um sinal de *clock*, omitido na figura, é comum aos três registradores de deslocamento (*shift register*) mostrados (o de seleção, o de entrada e o de saída) e responsável pelo sincronismo dos mesmos. Cada um destes registradores possui o seu respectivo sinal de carregamento (*load*), todos estes também omitidos, para permitir ou bloquear o carregamento ou o envio das respectivas sequências de *bits*. As taxas de transmissão ou recepção são sempre de um *bit* por batimento de *clock*. Estes registradores são também a única parte dinâmica do circuito; todo o resto opera de modo estático.

Em uma entrada digital serial denominada SEL ocorre o carregamento da sequência de *bits* que seleciona o *pixel* cujo coeficiente será individualmente lido. A transmissão do coeficiente, devidamente codificado em uma palavra binária de tamanho variável, ocorre pela saída digital serial de nome OUT. O receptor pode

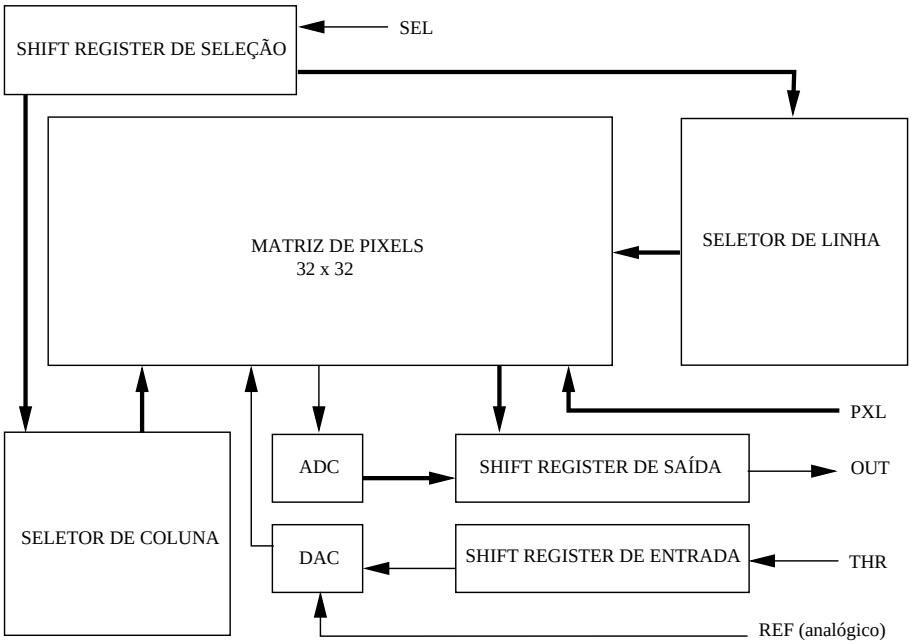

Figura 4.1: Visão geral do circuito implementado, com os barramentos representados por um traçado mais grosso e os fios por um traçado mais fino.

identificar o tamanho de cada palavra enviada pela leitura do seu primeiro *bit*, que define se a mesma tem tamanho dois ou oito *bits*.

Uma corrente de referência para o conversor digital-analógico é injetada pela entrada REF. Uma corrente múltipla desta referência é copiada dentro de cada *pixel* do imageador para ser utilizada na comparação com o valor do limiar T, conforme visto no Capítulo 3. A multiplicidade desta corrente é definida por uma sequência de *bits* armazenada no registrador de entrada, obtida pela entrada digital serial THR do mesmo.

O sinal de entrada PXL na verdade se refere a quatro sinais distintos que são enviados a cada *pixel* da matriz: três sinais de controle e uma segunda corrente de referência injetada (que é copiada para cada *pixel*) a ser utilizada nos circuitos para a determinação das árvores de zeros. Os três sinais de controle são os responsáveis pelo *reset* do *pixel* e pelas suas duas amostragens a serem realizadas pelo circuito de *correlated double sampling*.

4.2 Estrutura dos *Pixels*

Os *pixels* utilizados neste trabalho, em função da diferença entre as operações para as quais são designados, acabam sendo estruturalmente diferentes entre si.

A Figura 4.2 ilustra, através de um diagrama em blocos, a estrutura projetada para o único *pixel* de baixa frequência utilizado nesta implementação, que no caso se localiza na primeira linha e na primeira coluna da matriz de *pixels*. O mesmo é

composto inicialmente de um sensor APS seguido de um circuito para a realização da diferença entre duas amostragens suas pelo método *correlated double sampling*. Logo, a saída deste último bloco contém o sinal equivalente ao elemento da primeira linha e da primeira coluna da matriz $M_{32\times32}$ que representa a imagem amostrada.

Este sinal é representado na figura pelo nome $PXL_{1,1}$ já que o *pixel* em questão se encontra fisicamente na posição (1,1) da matriz. Por este sinal ser necessário aos outros três *pixels* vizinhos para o cálculo de um coeficiente da primeira transformada, ele deve ser transmitido aos mesmos para ser copiado.

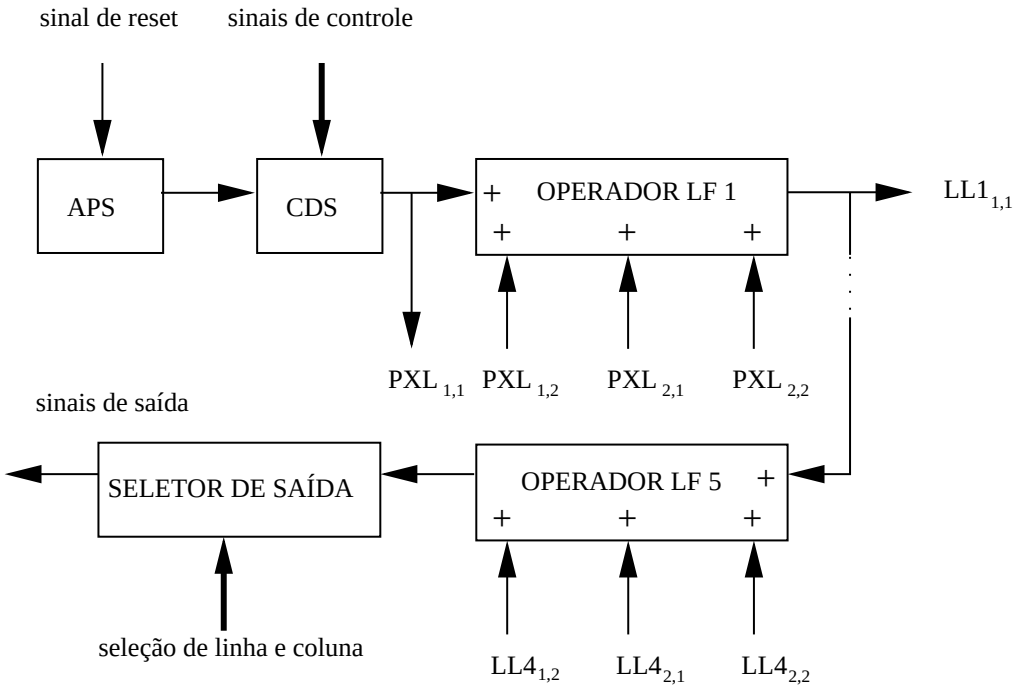

Figura 4.2: Aspecto estrutural do *pixel* com o coeficiente da sub-banda de baixa frequência. Os traçados mais grossos indicam um barramento no lugar de um único fio.

O bloco seguinte, nomeado operador LF1 (baixa frequência ou *Low Frequency*), realiza o cálculo do coeficiente $LL1_{1,1}$ através da média entre o sinal $PXL_{1,1}$ e os sinais recebidos dos *pixels* vizinhos. Ao sinal de saída deste bloco foi dado o nome do coeficiente calculado, e o mesmo deve ser transmitido aos *pixels* que irão utilizá-lo para o cálculo dos coeficientes da segunda transformada. Este padrão é seguido até o bloco operador LF5, utilizando-se os devidos coeficientes para cada transformada.

O sinal de saída do bloco operador LF5 é o único coeficiente da sub-banda $LL5$. Por ter sido obtido apenas pela soma de valores positivos, tem necessariamente o seu valor também positivo e é assumidamente significativo. O seu valor ainda é dado na forma de uma corrente, o que permite que o coeficiente seja copiado para um circuito de seleção representado pelo último bloco da sequência. Este bloco é responsável pelo corte do sinal quando o *pixel* em questão não for o selecionado; do contrário,

permite o seu envio a um codificador analógico-digital, externo à matriz de *pixels*, juntamente com o envio dos sinais do prefixo de sua codificação (lembrando que no caso $LL5$ o coeficiente é obrigatoriamente positivo e significativo) ao registrador de deslocamento de saída.

A estrutura dos *pixels* sofre uma ligeira mudança para aqueles que implementam os coeficientes das sub-bandas de alta frequência. A Figura 4.3 exemplifica a estrutura dos *pixels* cujos coeficientes pertençam à sub-banda $HL1$. Como existe um conjunto deles com esta mesma estrutura, considere que estamos nos referindo a um *pixel* fisicamente localizado na linha l e coluna c da matriz de *pixels*, de forma que os valores possíveis para l e c obedecem à lógica da Figura 2.5.

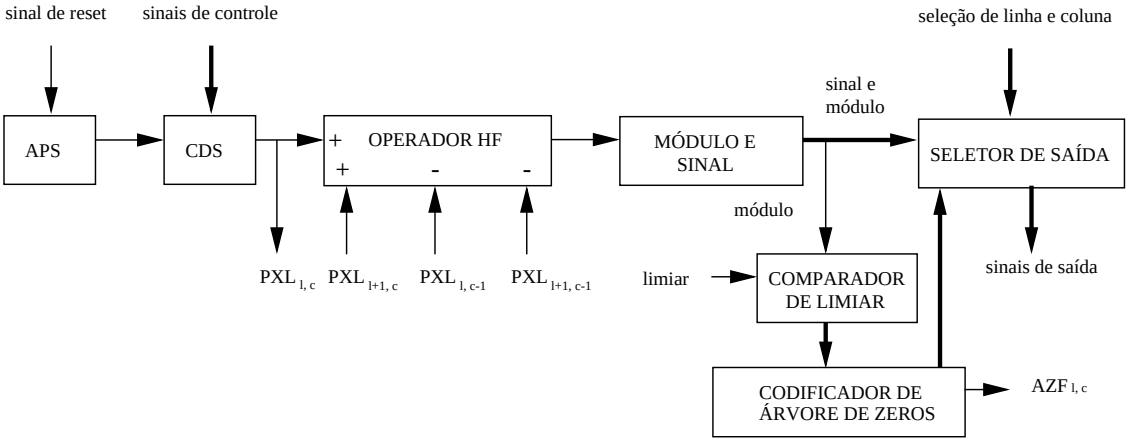

Figura 4.3: Aspecto estrutural dos *pixels* com coeficientes pertencentes à sub-banda de mais alta resolução e orientação HL. Os traçados mais grossos indicam um barramento no lugar de um único fio.

A estrutura destes *pixels* é igual à do *pixel* que calcula o coeficiente da sub-banda $LL5$ até a saída do seu bloco CDS, onde então o bloco operador LF1 é substituído pelo bloco operador HF (alta frequência ou *High Frequency*). Este operador HF é capaz de calcular um coeficiente da transformada pertencente a qualquer uma das três sub-bandas de alta frequência. O que define a orientação dos detalhes da sub-banda obtida é a escolha de quais sinais transmitidos ao operador HF serão somados e quais serão subtraídos. A ordem destas ligações, juntamente com as que formam a árvore de hierarquia entre as sub-bandas de mesma orientação, são as únicas diferenças estruturais entre os *pixels* cujos coeficientes pertencem a sub-bandas de mesma resolução de detalhamento.

Como os coeficientes obtidos pelo bloco operador HF podem ser positivos ou negativos, o circuito do bloco seguinte é responsável por obter o módulo do seu coeficiente e seu sinal. Ambos os sinais são enviados ao bloco de seleção tal como no caso do coeficiente da sub-banda $LL5$, mas o módulo deste coeficiente também é enviado e copiado em um outro bloco denominado comparador de limiar.

O bloco comparador de limiar realiza a comparação da corrente referente ao módulo do coeficiente obtido com uma corrente referente ao limiar T e gera um sinal referente à significância deste coeficiente, que é enviado ao bloco codificador de árvore de zeros. Este último bloco gera dois sinais para o bloco seletor de saída: um sobre a insignificância ou não do módulo do coeficiente obtido e o outro que diz respeito ao mesmo ser uma raiz de árvore de zeros, o que nunca é o caso já que os coeficientes da sub-banda $HL1$ são sempre folhas. Além disto, também é necessário sinalizar ao respectivo coeficiente hierarquicamente superior da sub-banda $HL2$ se o mesmo possui um filho insignificante ou significativo. Este sinal é representado pelo nome de $AZF_{l,c}$ (Árvore de Zeros Filial).

Os *pixels* das sub-bandas de resolução intermediária têm suas estruturas como sendo um misto das duas apresentadas. A Figura 4.4 mostra o exemplo do *pixel* com o coeficiente da sub-banda $HL5$. Como nele são calculados cinco coeficientes de cinco transformadas diferentes, o mesmo possui quatro blocos operadores LF e um bloco operador HF.

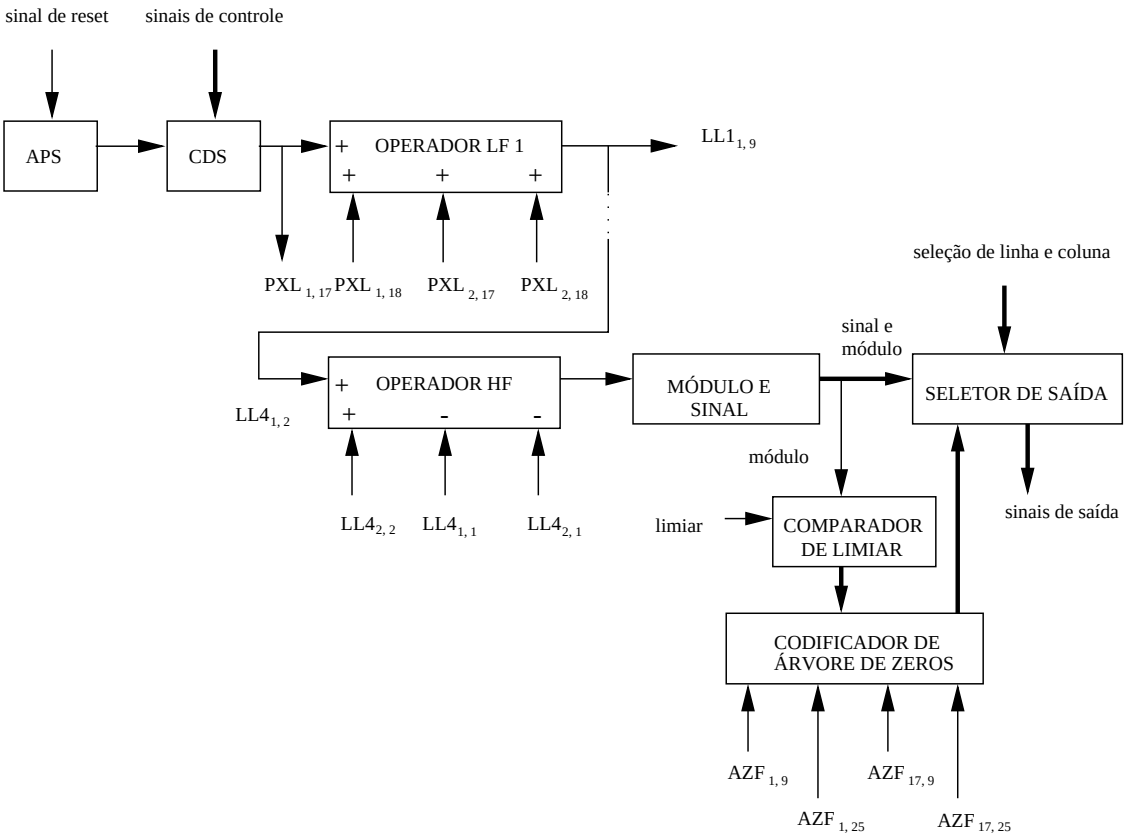

Figura 4.4: Aspecto estrutural do *pixel* com o coeficiente da sub-banda HL5. Os traçados mais grossos indicam um barramento no lugar de um único fio.

Uma outra modificação nestes *pixels* diz respeito ao bloco codificador de árvore de zeros. Este bloco agora recebe o sinal referente à significância dos seus quatro filhos na árvore hierárquica e os utiliza, juntamente com o seu próprio, para definir

sua codificação. Como o coeficiente em questão é a raiz de uma árvore hierárquica, não é necessário transmitir seu sinal de significância para nenhum outro *pixel*; se o mesmo fosse de uma frequência entre $HL5$ e $HL1$, o bloco codificador seria um misto dos dois, transmitindo um sinal AZF de significância e recebendo quatro outros.

4.3 Circuitos de Operações Básicas

Os circuitos a seguir desempenham as operações básicas realizadas neste trabalho, geralmente sendo incluídos em outros circuitos mais complexos.

4.3.1 Espelhos de Corrente

A geração dos coeficientes das transformadas decorre de um elevado número de operações básicas (somas, subtrações, produtos e divisões). Pela simplicidade de se realizar estas operações em modo de corrente e pelo menor número de transistores necessários, foi decidido pela implementação dos circuitos através deste modo.

O circuito básico utilizado para este modo de operação é o espelho de corrente simples [11], mostrado na Figura 4.5(a) para o caso de um espelho formado por transistores MOSFET (*Metal Oxide Semiconductor Field Effect Transistor*) do tipo N (NMOS) e na Figura 4.5(b) para o caso de um espelho formado por transistores MOSFET do tipo P (PMOS).

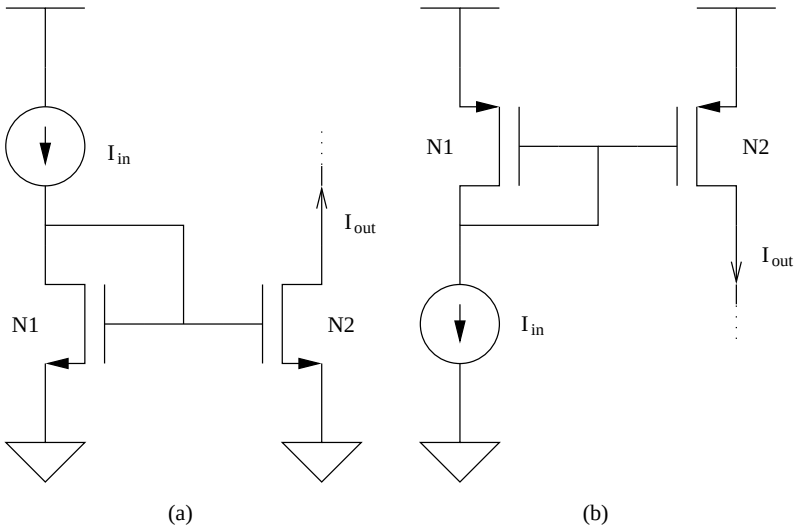

Figura 4.5: Diagramas esquemáticos dos espelhos de corrente simples NMOS e PMOS.

Um espelho de corrente consiste em N1 transistores de entrada e N2 transistores de saída, todos de mesmo tamanho (mesmos valores para todos os W, mesmos valores para todos os L, ambos definidos a seguir). Todos os transistores devem operar na região de saturação. Quando eles o fazem, a corrente I_d que passa pelo terminal

de dreno (*drain*) de cada transistor é controlada principalmente pela diferença de tensão entre os seus terminais de fonte (*source*) e de porta (*gate*), como mostra a Equação (4.1).

$$I_d = \frac{K}{2}\frac{W}{L}(V_{gs} - V_{th})^2(1 + \lambda V_{ds}) \tag{4.1}$$

Nesta equação, K é a constante de transcondutância, W a largura do transistor, L o seu comprimento, V_{th} a tensão de limiar (threshold) e λ o fator de modulação de comprimento do canal. V_{th} é positivo para transistores de canal N (NMOS) e negativo para transistores de canal P (PMOS).

Para o caso do espelho de corrente, a configuração é tal que a diferença de tensão entre fonte e porta é igual para todos os transistores. Portanto, quando se despreza o fator λ e as variações de valor para W e L na Equação (4.1), obtém-se a relação entre as correntes de entrada I_{in} e de saída I_{out} fornecida pela Equação (4.2).

$$I_{out} = \frac{N_1}{N_2} I_{in} \tag{4.2}$$

Para que se possa desprezar o fator λ e as variações nos valores de W e L, é necessário que tanto W quanto L possuam valores maiores do que o mínimo permitido pela tecnologia. As dimensões dos transistores dos espelhos de corrente foram estudadas em [12], e os parâmetros W=1 μm e L=2 μm foram escolhidos por apresentarem uma baixa variação na corrente prevista e um custo em área razoável.

4.3.2 Inversores

Os inversores utilizados neste trabalho têm duas funções básicas: inverter um sinal lógico e funcionar como um *buffer* de tensão, geralmente após um comparador de corrente. A Figura 4.6(a) mostra o esquemático do inversor e a Figura 4.6(b) o símbolo utilizado para representá-lo nos outros esquemáticos deste trabalho.

Idealmente, uma tensão de entrada V_{in} tende a retirar um dos dois transistores M1 ou M2 da região de corte e colocá-lo na saturação, o que é impedido pelo outro transistor que permanece cortado. Para compensar, a tensão do terminal de dreno, que é a própria tensão V_{out}, é modificada de maneira a colocar o transistor na região ôhmica com uma corrente muito próxima de zero, com mesma ordem de grandeza da corrente de corte. A corrente de dreno de um transistor na região ôhmica é aproximada pela Equação (4.3).

$$I_d = \frac{K}{2}\frac{W}{L}V_{ds}\left(V_{gs} - V_{th} - \frac{V_{ds}}{2}\right) \tag{4.3}$$

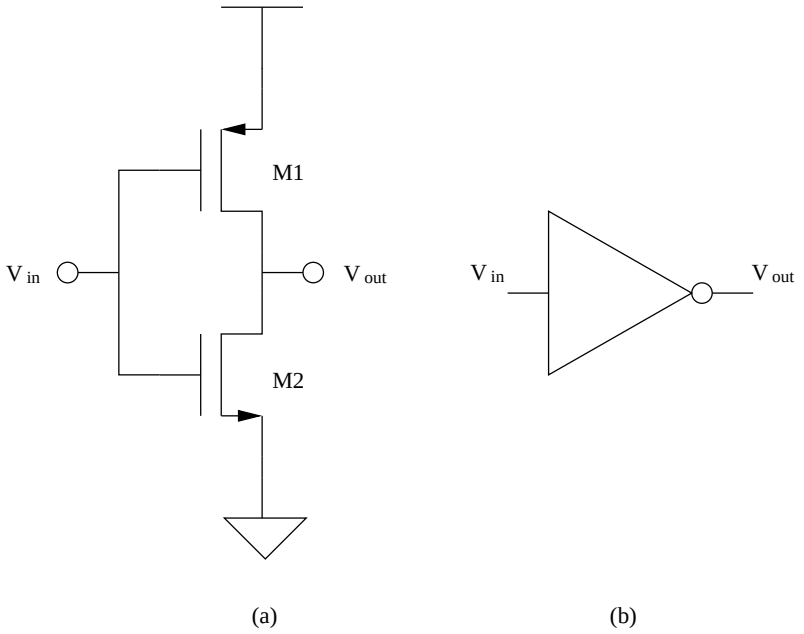

Figura 4.6: Inversor CMOS e símbolo.

O inversor, portanto, não funciona bem no intervalo em que a tensão V_{in} tende a colocar ambos os transistores na região de saturação. A Tabela 4.1 mostra o valor de V_{out} esperado para os intervalos de V_{in} que permitem o funcionamento do circuito como um inversor.

Tabela 4.1: Lógica de sinais de um inversor.

V_{in} (V)	V_{out} (V)
$\approx$(3,3-V_{th}) até 3,3	0
0 até $\approx V_{th}$	3,3

Neste trabalho, duas versões de inversores são utilizadas. A versão mínima, que ocupa menor área, tem as dimensões de seus transistores mostrada na Tabela 4.2 e é utilizada nos circuitos dentro de cada *pixel*. Esta versão apresenta uma velocidade de resposta desigual em relação ao valor lógico do sinal de entrada V_{in} devido à diferença entre as constantes de transcondutância dos transistores do tipo P e do tipo N. Uma versão balanceada, que compensa a diferença entre estas constantes, é utilizada nos circuitos fora da matriz de *pixels*. As dimensões desta versão balanceada são fornecidas pela Tabela 4.3.

Tabela 4.2: Dimensões dos transistores de um inversor mínimo.

Transistor	W(μm)	L(μm)
M1	0,4	0,35
M2	0,4	0,35

Tabela 4.3: Dimensões dos transistores de um inversor mínimo balanceado.

Transistor	W(μm)	L(μm)
M1	1,2	0,35
M2	0,4	0,35

4.3.3 Portas Lógicas NAND

Para a realização da lógica de seleção dos *pixels*, portas lógicas NAND estáticas de duas e cinco entradas são utilizadas. O diagrama esquemático de uma porta NAND estática de duas entradas V_A e V_B é dado pela Figura 4.7(a), e o seu símbolo pela Figura 4.7(b).

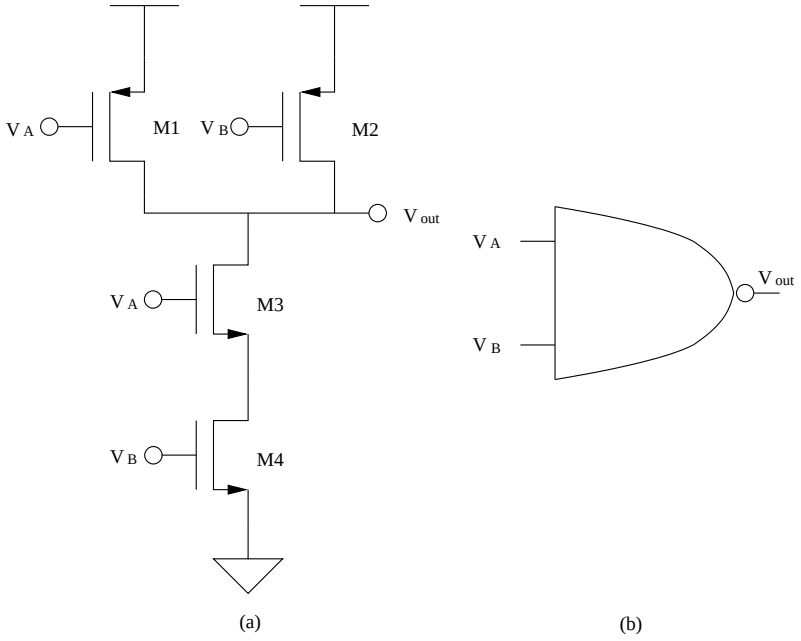

Figura 4.7: Porta NAND CMOS estática de duas entradas e símbolo.

O funcionamento desta porta lógica é semelhante ao do inversor, com o acréscimo de um nó extra, ligado ao terminal de fonte de M3 e ao terminal de dreno de M4, cuja tensão pode variar sem afetar a saída V_{out} enquanto pelo menos um destes dois transistores for mantido cortado. A Tabela 4.4 mostra o valor de V_{out} para a lógica de sinais de entrada V_A e V_B.

Tabela 4.4: Lógica de sinais de uma porta NAND.

V_A (V)	V_B (V)	V_{out} (V)
$\approx$(3,3-V_{th}) até 3,3	$\approx$(3,3-V_{th}) até 3,3	0
0 até $\approx V_{th}$	qualquer	3,3
qualquer	0 até $\approx V_{th}$	3,3

As dimensões dos transistores para as portas de duas entradas implementadas são dadas pela Tabela 4.5. São usadas as dimensões mínimas para os transistores

das portas localizadas dentro dos *pixels*, sem balanceamento, já que a área utilizada é um fator de relevância muito maior do que a velocidade do circuito.

Tabela 4.5: Dimensões dos transistores de uma porta NAND de duas entradas.

Transistor	W(μm)	L(μm)
PMOS (M1 e M2)	0,4	0,35
NMOS (M3 e M4)	0,4	0,35

As portas NAND de cinco entradas são semelhantes às de duas, mas contendo cinco transistores PMOS em paralelo e cinco transistores NMOS em série. Estas portas são usadas na decodificação da sequência de *bits* do seletor de *pixel*, que por sua vez é desbalanceado na sua velocidade de resposta devido ao número de inversores da lógica do circuito depender da linha ou coluna referenciada. Foi decidido então por aumentar um pouco o W dos transistores NMOS para se atenuar o pior caso de velocidade (quando os cinco transistores NMOS não estão cortados), pois a decisão não ocorria em um aumento de área necessária devido à diferença de espaço ocupado entre os transistores PMOS e os NMOS dos inversores balanceados. A Tabela 4.6 exibe as dimensões obtidas.

Tabela 4.6: Dimensões dos transistores de uma porta NAND de cinco entradas.

Tipo de transistor	W(μm)	L(μm)
PMOS	0,4	0,35
NMOS	1,2	0,35

4.3.4 Chaves Complementares

As chaves analógicas CMOS utilizadas para a transmissão de um valor de tensão apresentam a configuração complementar, ilustrada na Figura 4.8. A configuração complementar, que utiliza um transistor do tipo P (M1) e um transistor do tipo N (M2), evita a queda de V_{th} na tensão transmitida que ocorreria com o uso de uma chave simples.

A chave está fechada quando $\phi = 3.3$ V e $\bar{\phi} = 0$ V, e aberta quando $\phi = 0$ V e $\bar{\phi} = 3.3$ V. Quando ela se encontra fechada, tem-se que $V_{out} = V_{in}$, e quando ela está aberta, V_{out} mantém seu valor dado pelo acúmulo de cargas do estado anterior.

Os transistores M3 e M4 funcionam como transistores *dummy*, agindo para reduzir os efeitos de injeção de carga. Como esta injeção ocorre somente nos nós passivos do circuito devido à alta impedância dos mesmos, duas versões para a chave complementar são implementadas nos circuitos deste trabalho: uma com os transistores M3 e M4 e outra sem os mesmos.

As dimensões dos transistores são dadas pela Tabela 4.7.

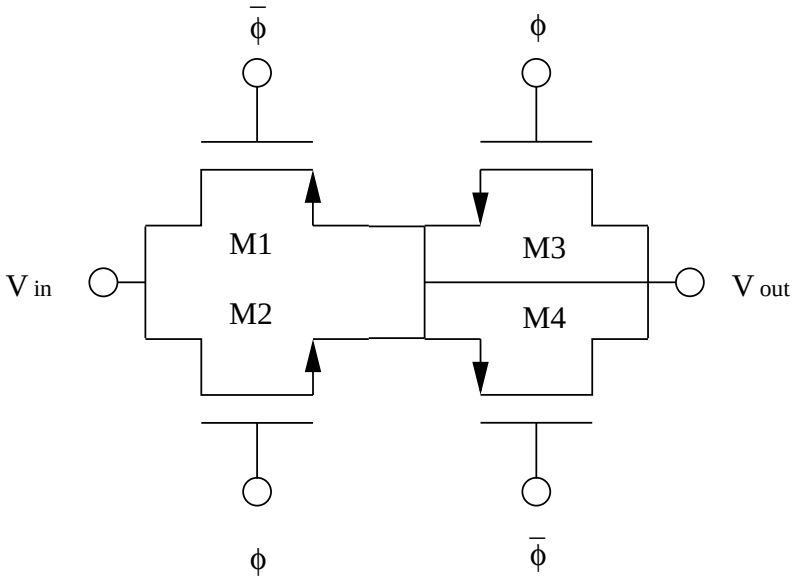

Figura 4.8: Chave complementar CMOS com transistores *dummy*.

Tabela 4.7: Dimensões dos transistores das chaves complementares CMOS.

Transistor	W(μm)	L(μm)
M1	1	0,35
M2	1	0,35
M3	0,5	0,35
M4	0,5	0,35

4.4 Sensor APS e circuito de *Correlated Double Sampling*

O circuito de leitura utilizado foi inicialmente proposto em [13] e posteriormente modificado em [15] para permitir a manipulação da corrente dentro do próprio *pixel*, sacrificando um pouco da linearidade mas eliminando a necessidade de um *current conveyor* [14]. Nele é utilizado um fotodiodo como elemento fotossensível e sua arquitetura é muito próxima da célula 3-T convencional. Seu diagrama esquemático é mostrado na Figura 4.9.

O circuito que forma o sensor APS é composto pelo fotodiodo e pelos transistores M1 e M2. Analogamente às células 3-T, uma tensão V_{pd} é gerada no nó de integração de cargas do sensor APS pelo acúmulo das mesmas em sua capacitância equivalente, dada principalmente pela soma da capacitância C_{pd} do fotodiodo com a capacitância C_{gs} entre os terminais de porta e fonte do transistor M2. Esta tensão diminui devido à drenagem das cargas causada pela corrente do fotodiodo I_{pd}, proporcional à potência da incidência luminosa. O transistor M1, através do sinal V_{rst}, eleva a tensão V_{pd} para seu estado inicial de aproximadamente $3,3 - V_{th}$ volts durante um tempo de *reset* t_{rst}, quando então M1 é cortado para dar início ao período de

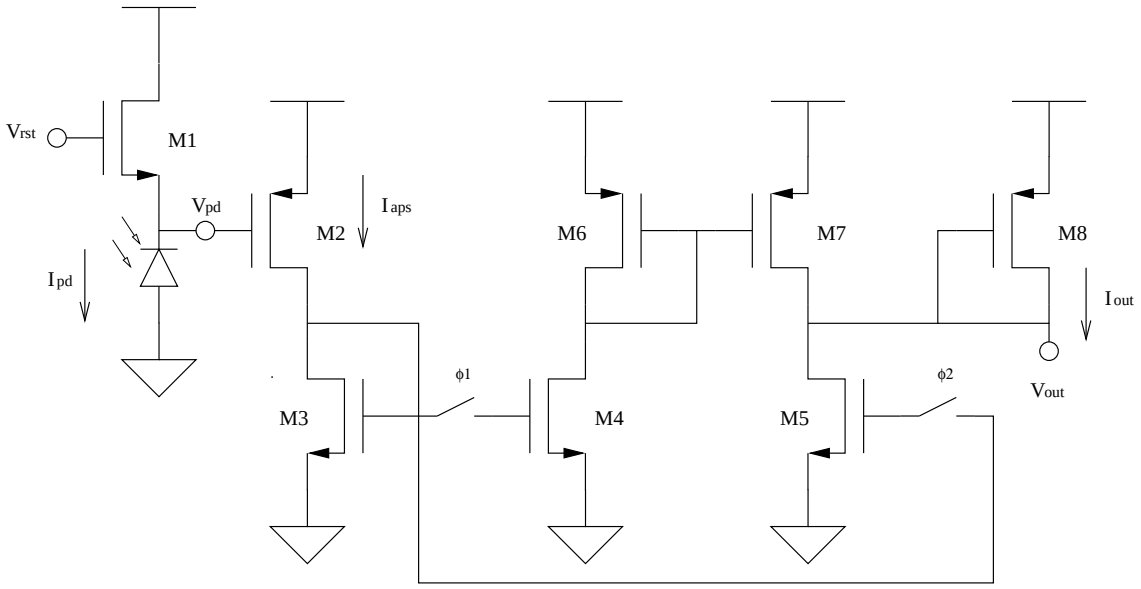

Figura 4.9: Sensor APS e circuito CDS utilizados.

integração das cargas.

A corrente de saída I_{aps} do sensor APS aumenta à medida em que V_{pd} diminui, conforme a Equação (4.3) é a corrente de dreno do transistor M2, que opera na região ôhmica. Esta corrente é inversamente proporcional à tensão V_{pd} e, portanto, diretamente proporcional ao quadrado da potência luminosa incidente.

O fotodiodo utilizado tem dimensões de 10 μm $\times$ 10 μm e assumiu-se em [15], para fins de projeto e simulação, que a corrente máxima no fotodiodo é de 400 pA para uma potência luminosa de aproximadamente 12 $\frac{W}{m^2}$.

A amostragem do sinal é realizada após um tempo de integração t_{int} pelo circuito de *correlated double sampling*, composto pelos transistores M3 a M8.

Este circuito funciona retendo a mesma quantidade de cargas do terminal porta de M3 referente ao valor de corrente $I_{aps} = I_{aps1}$ no início do tempo de integração no terminal porta de M4, e a mesma quantidade de cargas do terminal de porta de M3 referente ao valor de corrente $I_{aps} = I_{aps2}$ no final do tempo de integração no terminal porta de M5. Para isto, os sinais $\phi 1$ e $\phi 2$ são inicialmente acionados para fechar as suas chaves correspondentes, e então desligados ao se atingir os respectivos tempos t_{smp} e $t_{smp} + t_{int}$. Estas chaves foram projetadas de forma a possuírem transistores *dummy* entre si mesmas e M4 ou M5, de forma a haver uma pequena capacitância para absorver as cargas injetadas.

A corrente de dreno de M7 tem, por espelhamento, seu valor igual à de M4, que vale I_{aps1}. Como $I_{aps2} > I_{aps1}$, o transistor M8 gera uma corrente de dreno $I_{out} = I_{aps2} - I_{aps1}$ para compensar M5. Esta corrente força o nó do seu terminal porta a ter um valor de tensão V_{out}, que pode ser usado para a cópia da corrente de dreno de M8.

As dimensões utilizadas para os transistores são fornecidas pela Tabela 4.8.

Tabela 4.8: Dimensões dos transistores para o sensor APS e o circuito de CDS.

Transistor	W(μm)	L(μm)	Transistor	W(μm)	L(μm)
M1	1	2	M5	2	2
M2	1	3,7	M6	1	2
M3	2	2	M7	1	2
M4	2	2	M8	1	2

A Figura 4.10 mostra os resultados da simulação elétrica transiente obtida para o circuito a parâmetros típicos com t_{int} = 100 μs e I_{pd} = 400 pA. A Figura 4.11 mostra a variação dos resultados para I_{out} obtida por dez rodadas de Monte Carlo, usando-se os mesmos sinais de controle da simulação da Figura 4.10.

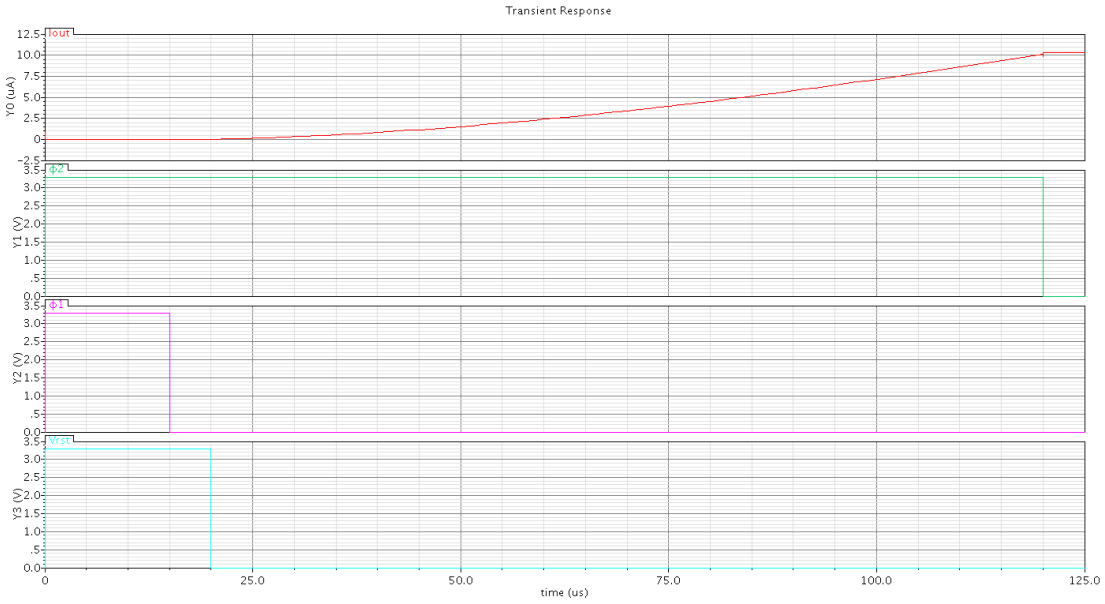

Figura 4.10: Simulação elétrica do conjunto APS e CDS a parâmetros típicos.

Variações na saída do circuito com a potência luminosa incidente comuns a todos os fotodiodos podem ser compensadas através da modificação do tempo t_{int} de integração, que pode ser ajustado de maneira a otimizar a faixa dinâmica de leitura dos mesmos.

4.5 Operador de Baixa Frequência

O esquemático deste circuito denominado operador de baixa frequência é ilustrado na Figura 4.12. A função deste circuito é calcular um único coeficiente da sub-banda de baixa frequência espacial, daí o seu nome.

Quatro correntes I_1 a I_4 são geradas através das respectivas tensões V_1 a V_4 nos respectivos terminais porta dos transistores M1 a M4. Estas correntes são as

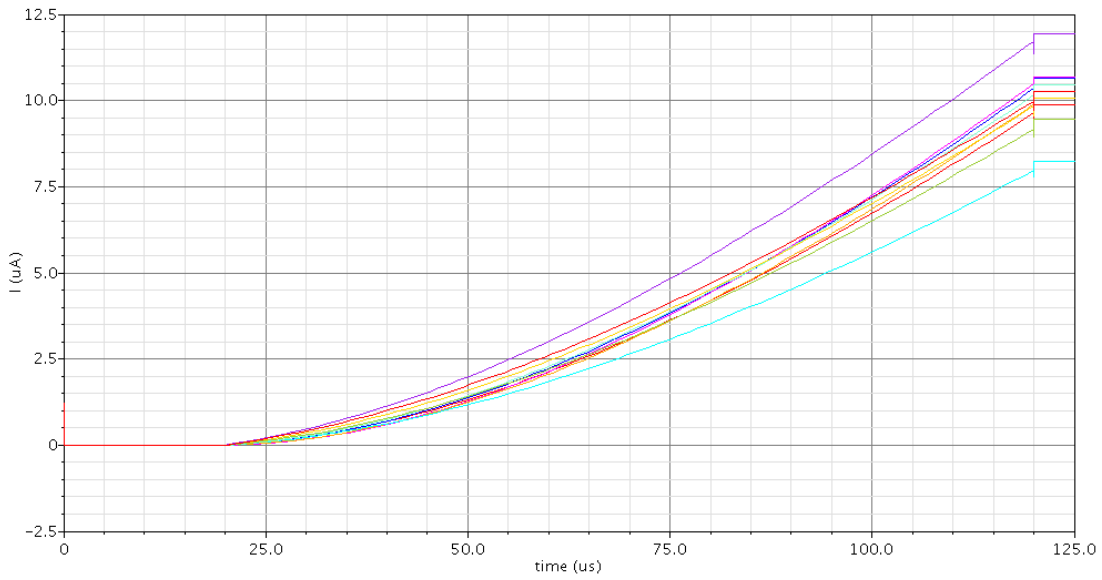

Figura 4.11: I_{out} obtida através de dez rodadas de Monte Carlo pelo conjunto APS e CDS.

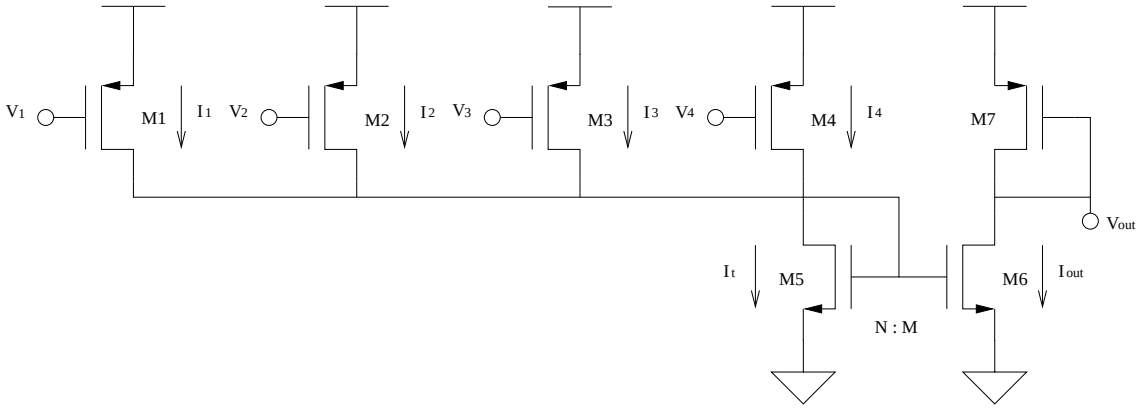

Figura 4.12: Diagrama esquemático do circuito operador de baixa frequência.

versões espelhadas idealmente na razão de um para um das correntes as quais se deseja efetuar a soma. O transistor M5 absorve uma corrente $I_t = I_1 + I_2 + I_3 + I_4$ e espelha este valor para o transistor M6 na razão de $\frac{M}{N}$. Seja I_{out} a corrente de dreno tanto de M6 quanto de M7, seu valor é fornecido pela Equação (4.4) quando M6 opera na saturação.

$$I_{out} = \frac{M}{N}\left(I_1 + I_2 + I_3 + I_4\right) \tag{4.4}$$

Os valores $M = 1$ e $N = 4$ são usados para implementar os coeficientes da maneira descrita no Capítulo 2, além de garantir que I_{out} nunca terá um valor maior do que a corrente máxima do sensor APS. A Tabela 4.9 mostra as dimensões dos transistores utilizadas.

A Figura 4.13 mostra os resultados da simulação elétrica transiente obtida para

Tabela 4.9: Dimensões dos transistores para o operador de baixa frequência.

Transistor	W(μm)	L(μm)	Quantidade	Transistor	W(μm)	L(μm)	Quantidade
M1	1	2	1	M5	1	2	4
M2	1	2	1	M6	1	2	1
M3	1	2	1	M7	1	2	1
M4	1	2	1				

o circuito a parâmetros típicos, mantendo-se $I_1 = I_2 = I_3 = I_4$ e provocando-se um aumento linear destes valores com o tempo. A Figura 4.14 mostra a variação dos resultados para I_{out} obtida por dez rodadas de Monte Carlo, com os valores de I_1 a I_4 variando da mesma forma apresentada na simulação da Figura 4.13.

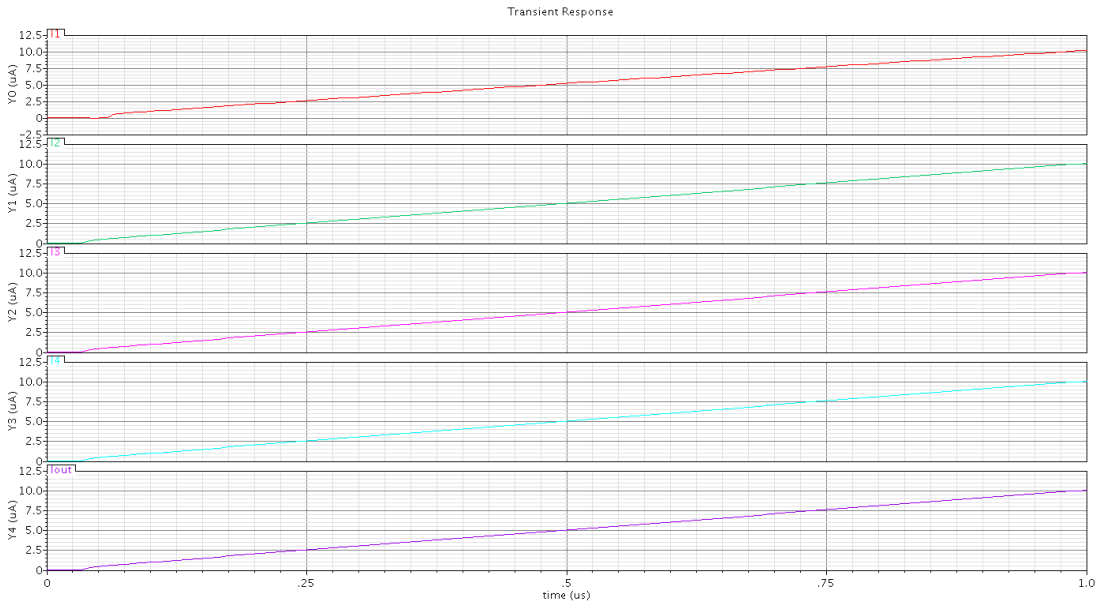

Figura 4.13: Simulação elétrica do operador de baixa frequência a parâmetros típicos.

4.6 Operador de Alta Frequência e Circuito de Módulo e Sinal

Este circuito é composto de duas partes: uma responsável pelo cálculo de um coeficiente de uma sub-banda de alta frequência e outro responsável por obter o módulo e o sinal deste resultado. O circuito de módulo e sinal apresentado neste trabalho foi originalmente apresentado em [16]. Ambos são mostrados na Figura 4.15

O circuito operador de alta frequência é formado pelos transistores M1 a M4. As correntes a serem somadas, I_1 e I_2, são espelhadas pelos transistores M1 e M2 enquanto as correntes a serem subtraídas, I_3 e I_4, são espelhadas pelos transistores M3 e M4. As tensõe V_1 do nó de soma, V_2 do nó após o primeiro inversor e V_sig do

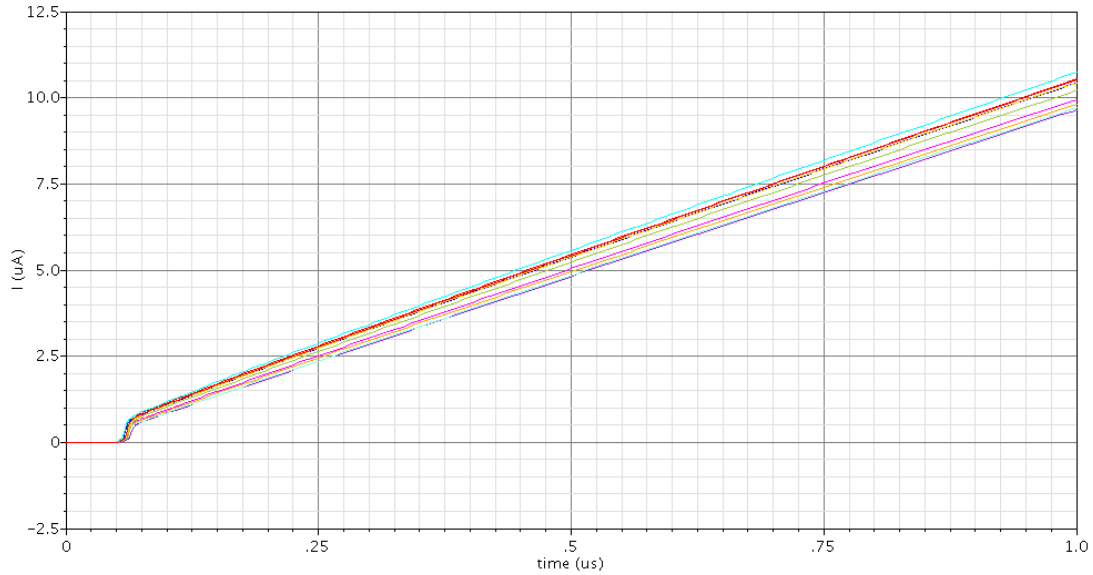

Figura 4.14: I_{out} obtida através de dez rodadas de Monte Carlo para o operador de baixa frequência.

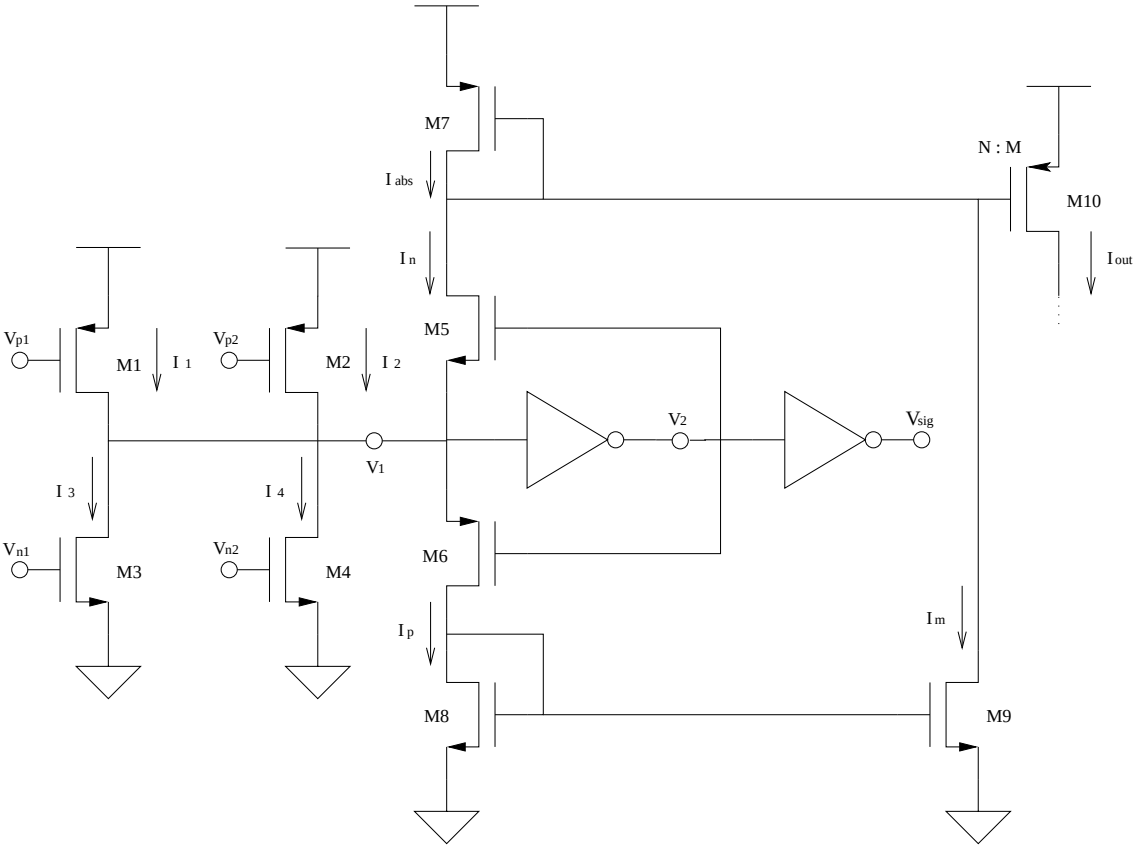

Figura 4.15: Esquemático dos circuitos operador de alta frequência e de módulo e sinal.

nó após o segundo inversor dependem do resultado da operação $I_1 + I_2 - I_3 - I_4$, como indica a Tabela 4.10.

O circuito de módulo, formado pelos transistores M5 a M8, age de maneira a

Tabela 4.10: Tensões nos nós do operador HF e do circuito de módulo e sinal.

$I_1 + I_2 - I_3 - I_4$	V_1(V)	V_2(V)	V_{sig}(V)
>0	≈3,3	0	3,3
<0	≈0	3,3	0

compensar a soma de correntes no nó de V1. Quando $V2 = 0$V, o transistor M6, que funciona como uma chave, é ligado, permitindo M8 absorver a diferença de corrente injetada no nó de V1, gerando-se a relação $I_p = I_1 + I_2 - I_3 - I_4$. O valor de I_p é copiado para I_m através do espelho formado pelos transistores M8 e M9, e como M5 permanece cortado, $I_n = 0$ A.

Já quando $V2 = 3,3$ V, a chave ligada é o transistor M5, e M7 absorve a diferença de corrente drenada do nó de V1 tal que $I_n = I_3 + I_4 - I_1 - I_2$. M6 permanece cortado e, portanto, $I_p = 0$ A. O valor da corrente em módulo é, portanto, dado por $I_{abs} = I_p + I_n$, relação válida em ambos os casos. A Tabela 4.11 resume o funcionamento do circuito.

Tabela 4.11: Valor de I_{abs} de acordo com V_1.

V_1(V)	I_p(A)	I_n(A)	I_{abs}(A)
≈3,3	$I_1 + I_2 - I_3 - I_4$	0	$I_1 + I_2 - I_3 - I_4$
≈0	0	$I_3 + I_4 - I_1 - I_2$	$I_3 + I_4 - I_1 - I_2$

Por fim, I_{abs} é copiada em I_{out} pelo espelho de corrente formado entre M7 e M10, sendo dividida por um fator de escala igual a dois ($N = 2$, $M = 1$) para se obter um valor idêntico ao coeficiente da transformada desejado, além de garantir que I_{out} nunca terá um valor maior do que a corrente máxima do sensor APS. A Equação (4.5) fornece o resultado de I_{out}.

$$I_{out} = \frac{M}{N} \left| \left(I_1 + I_2\right) - \left(I_3 + I_4\right) \right| \tag{4.5}$$

A Tabela 4.12 mostra as dimensões implementadas para os transistores.

Tabela 4.12: Dimensões dos transistores para o operador de alta frequência e o circuito de módulo e sinal.

Transistor	W(μm)	L(μm)	Quantidade	Transistor	W(μm)	L(μm)	Quantidade
M1	1	2	1	M6	0,4	0,35	1
M2	1	2	1	M7	1	2	2
M3	1	6	1	M8	1	2	1
M4	1	6	1	M9	1	2	1
M5	0,4	0,35	1	M10	1	2	1

A Figura 4.16 mostra os resultados da simulação elétrica transiente obtida para o circuito a parâmetros típicos, variando-se as correntes I_1 e I_3 conforme mostrado. A

Figura 4.17 mostra a variação dos resultados obtida para V_{sig} através de dez rodadas de Monte Carlo, e a Figura 4.18 mostra a variação dos resultados obtida para I_{out} da mesma maneira. Em ambos os casos, os valores de I_1 e I_3 foram variados da mesma forma ocorrida na simulação da Figura 4.16.

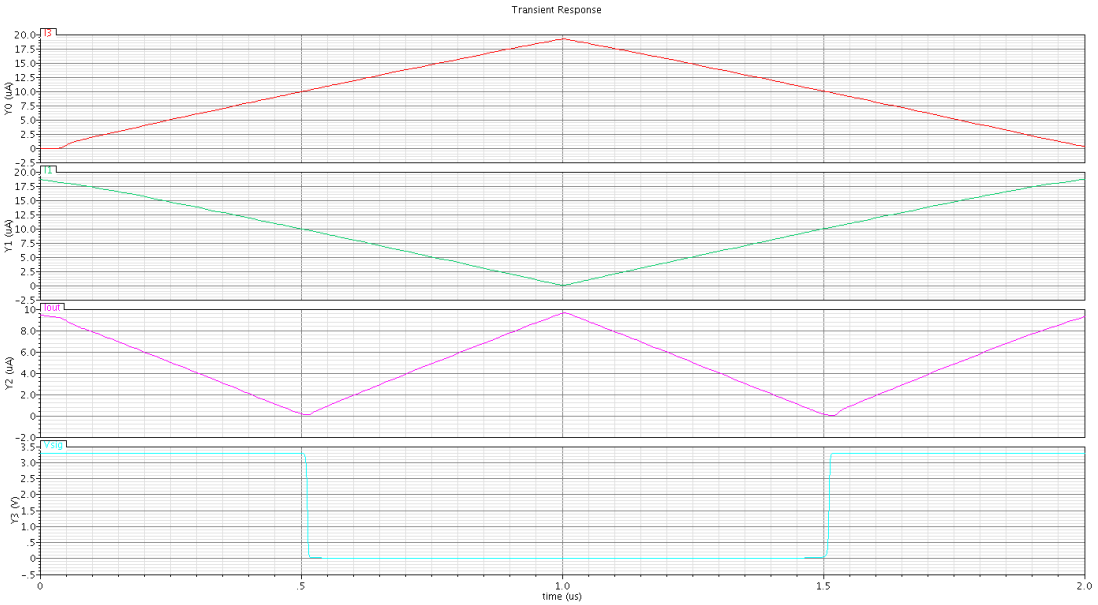

Figura 4.16: Simulação elétrica dos circuitos operador de alta frequência e de módulo a parâmetros típicos.

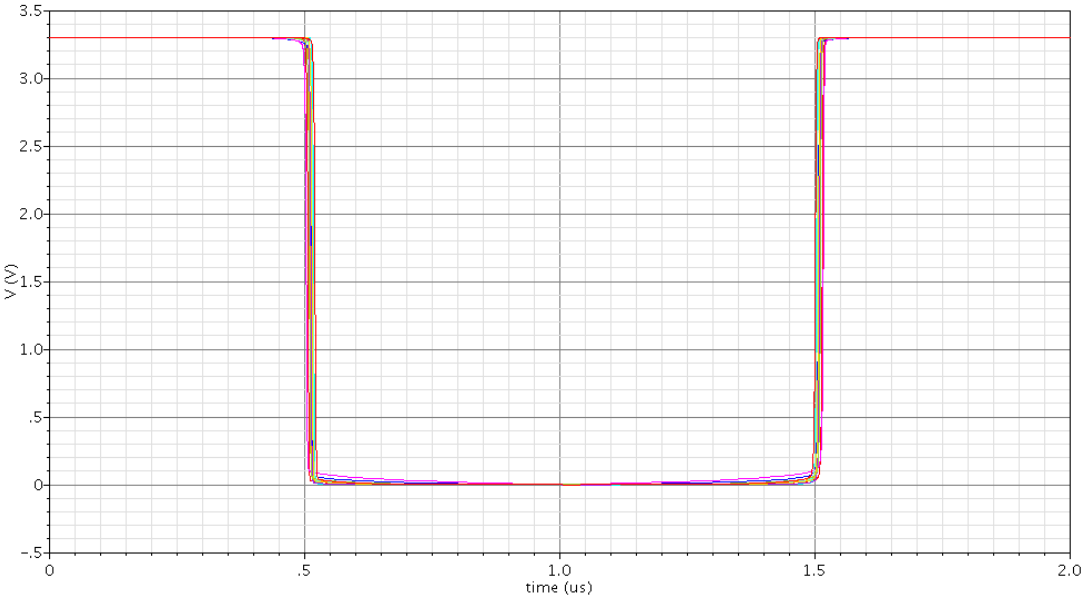

Figura 4.17: V_{sig} obtido através de dez rodadas de Monte Carlo para o circuito de sinal.

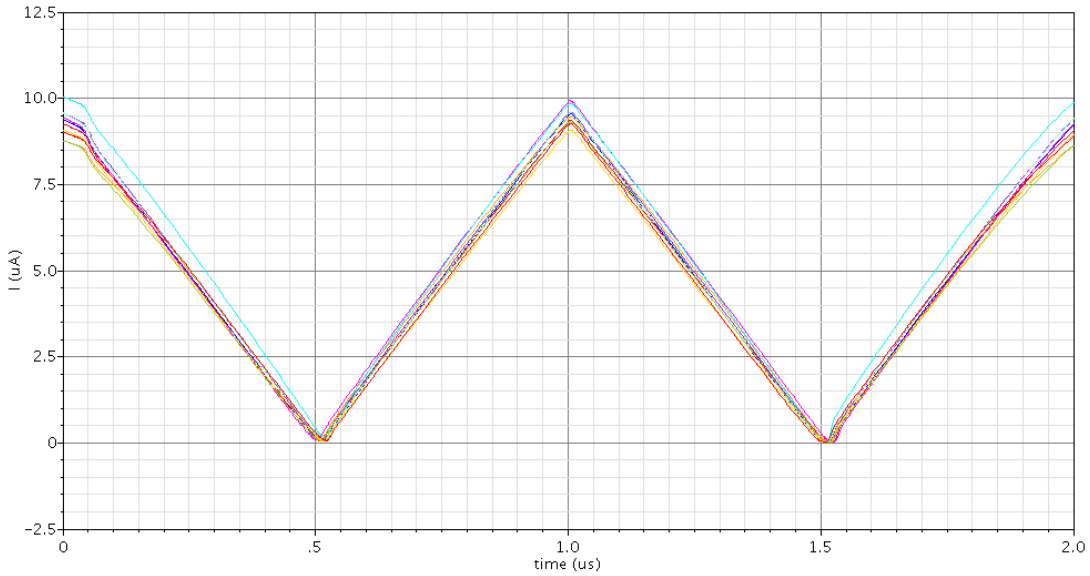

Figura 4.18: I_{out} obtida através de dez rodadas de Monte Carlo para o circuito de módulo.

4.7 Comparador de Corrente

O comparador de corrente simples apresentado é utilizado no circuito de comparação do valor de um coeficiente com o limiar T. A Figura 4.19 mostra o diagrama esquemático do comparador.

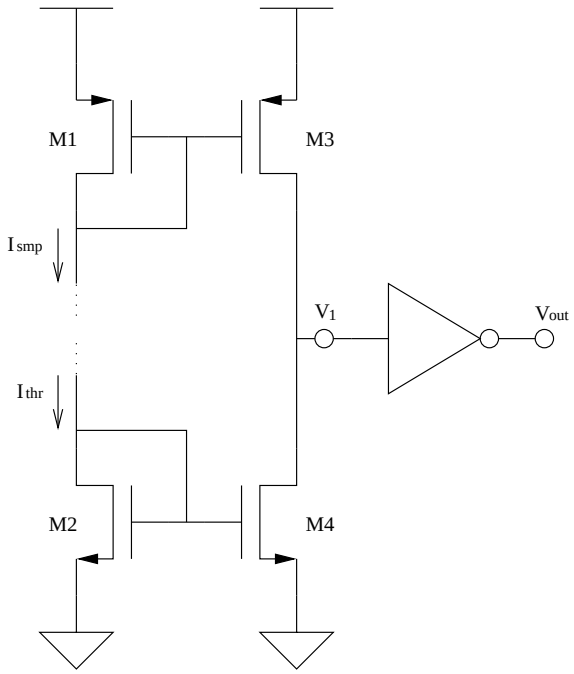

Figura 4.19: Diagrama esquemático do circuito comparador de corrente.

Uma corrente I_{smp} cujo valor representa o módulo do coeficiente é espelhada de M1 para M3, enquanto o mesmo ocorre entre M2 e M4 para uma corrente I_{thr}, cujo valor representa o limiar T. I_{thr} deve ser uma fração da corrente máxima do sensor APS para que a comparação ocorra entre coeficientes de mesma escala, o que é garantido pelos circuitos dos operadores LF e HF.

Como não há mais nenhum outro ramo ligado ao nó do terminal de dreno de M3 e M4, a tensão V_1 sobre o mesmo irá variar de forma a ajustar a corrente compartilhada por estes transistores. A tensão de saída V_{out}, portanto, será dada de acordo com a Tabela 4.13.

Tabela 4.13: Sinais do circuito comparador de corrente.

$I_{smp} - I_{thr}$	V_1(V)	V_{out}(V)
>0	≈3,3	0
<0	≈0	3,3

Portanto, na implementação deste circuito, uma saída de valor lógico 0 equivale a um coeficiente maior do que o limiar T, enquanto uma saída de valor lógico 1 equivale a este ser menor. A Tabela 4.14 mostra as dimensões implementadas para os transistores do circuito.

Tabela 4.14: Dimensões dos transistores do comparador de corrente.

Transistor	W(μm)	L(μm)	Transistor	W(μm)	L(μm)
M1	1	2	M3	1	2
M2	1	2	M4	1	2

A Figura 4.20 mostra os resultados da simulação elétrica transiente obtidos para o circuito a parâmetros típicos, variando-se I_{smp} e I_{thr} da maneira indicada. A Figura 4.21 mostra a variação dos resultados obtida para V_{out} através de dez rodadas de Monte Carlo para a mesma variação de I_{smp} e I_{thr} ocorrida na simulação da Figura 4.20.

4.8 Codificador de Árvore de Zeros

O circuito codificador de árvore de zeros tem como função determinar se o coeficiente calculado na posição do seu *pixel* é uma raiz de árvore de zeros.

Como cada *pixel* precisa dos sinais de quatro outros *pixels* além do seu próprio para determinar esta condição, a utilização de portas lógicas criaria a necessidade de se rotear quatro trilhas de metal no *layout* para cada ligação referente à árvore hierárquica das sub-bandas entre um pai e seus filhos. Para reduzir este número de trilhas, este circuito foi projetado de forma a substituir a utilização de portas lógicas por uma lógica de *pull-down*, utilizando transistores atuando como chaves e fontes

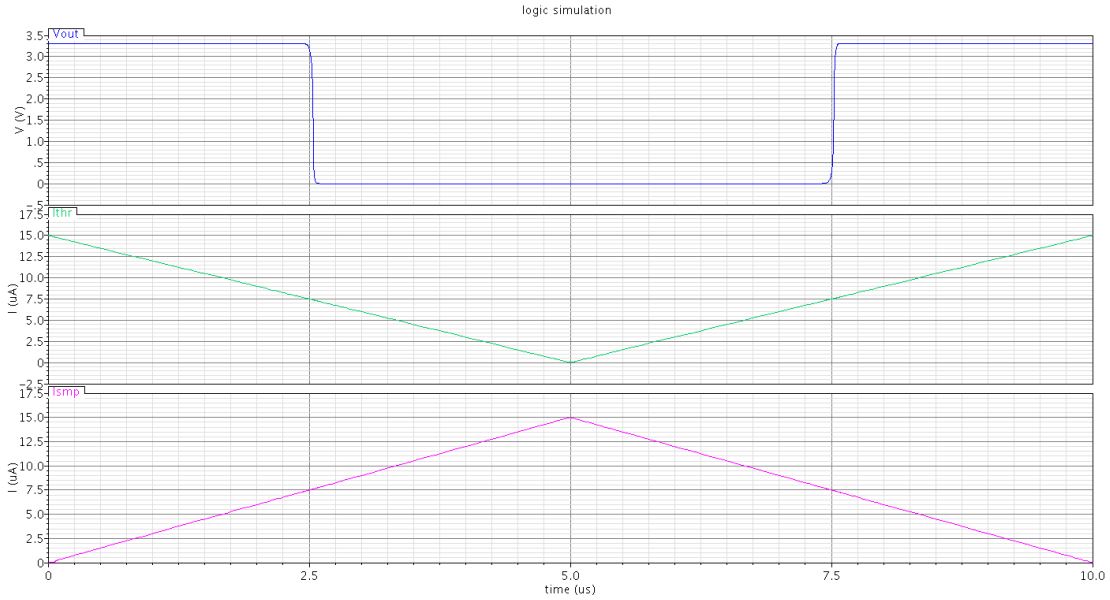

Figura 4.20: Simulação elétrica do circuito comparador de corrente a parâmetros típicos.

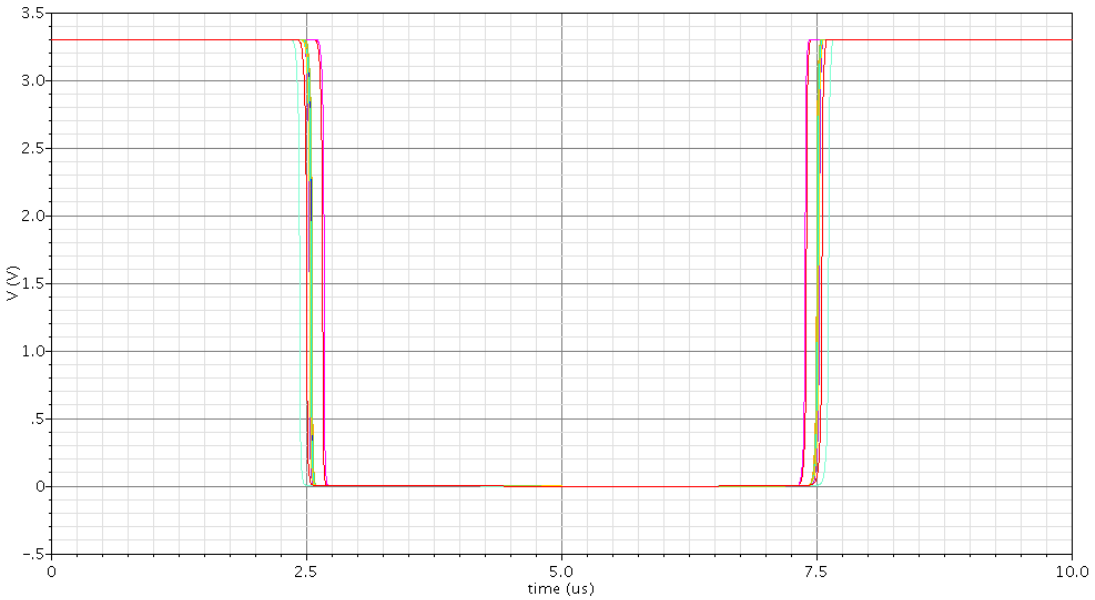

Figura 4.21: V_{out} obtido através de dez rodadas de Monte Carlo para o comparador de corrente.

de corrente. Desta maneira, apenas uma trilha de metal é necessária para realizar a ligação entre um pai e seus quatro filhos.

O diagrama esquemático do circuito é dividido em duas partes: uma referente aos *pixels* cujos coeficientes são pais na árvore, mostrada na Figura 4.22(a), e outra para os coeficientes que são filhos, exibida na Figura 4.22(b). Isto quer dizer que os *pixels* que calculam os coeficientes de $HL5$, de $LH5$ e de $HH5$ (raízes) possuem somente o circuito em (a), enquanto os *pixels* que calculam os coeficientes de $HL1$,

de $LH1$ e de $HH1$ (folhas) possuem somente o circuito em (b). *Pixels* responsáveis pelo cálculo de coeficientes de frequência intermediária a estes (galhos) possuem ambos os circuitos.

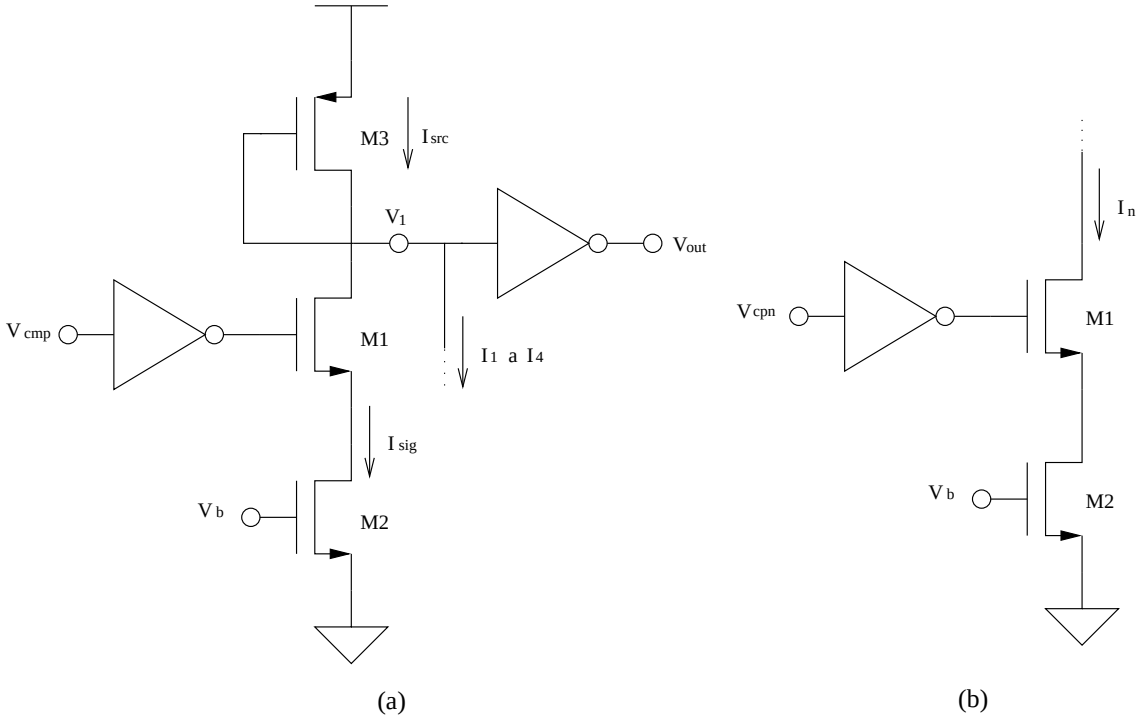

Figura 4.22: Diagrama esquemático do circuito codificador de árvore de zeros.

O nó do terminal de dreno dos transistores M1 é comum a um dado *pixel* cujo coeficiente seja um pai e aos quatro *pixels* cujos coeficientes sejam os filhos deste. Seja V_{cmp} o sinal neste pai referente à significância do seu coeficiente, e sejam V_{cp1}, V_{cp2}, V_{cp3} e V_{cp4} os respectivos sinais nestes filhos referentes à significância dos seus próprios coeficientes. Em todos estes, M1 estará ligado quando o coeficiente deste *pixel* for significativo, isto é, $V_{cmp} = 0$V para o pai e $V_{cp1} = 0$, $V_{cp2} = 0$, $V_{cp3} = 0$ e $V_{cp4} = 0$ para os filhos.

Na Figura 4.22(a), M3 é configurado como um diodo que deixa passar uma corrente I_{src}. M3 deve ser projetado de maneira que sua largura seja mínima e seu comprimento tenha um valor alto para diminuir sua capacidade máxima de fornecimento de corrente. Já os transistores M2 são feitos de maneira a espelhar uma corrente maior do que a que possa ser fornecida por M3 na Figura 4.22(a). O valor da tensão V_b deve ser tal que force os transistores M2 a individualmente drenar uma corrente maior do que M3 possa injetar no nó do seu dreno.

Em todas as situações, tem-se que $I_{src} = I_{sig} + I_1 + I_2 + I_3 + I_4$. Quando todos os transistores M1 estiverem cortados, tem-se a situação onde $I_{src} \approx 0$. Nesta situação, a tensão V_1 do nó de dreno de M3 é elevada até $\approx 3,3$ V, reduzindo drasticamente a corrente da fonte. Como resultado, tem-se que $V_{out} = 0$ V. Do contrário, basta

que um entre os cinco transistores M1 esteja ligado para que $I_{src} > 0$ e a tensão V_1 precise cair a um nível próximo de zero, gerando um sinal $V_{out} = 3,3$ V.

A Tabela 4.15 mostra a lógica do circuito, que implementa a função lógica OR.

Tabela 4.15: Lógica do circuito codificador de árvore de zeros.

V_{cmp}(V)	V_{cp1}(V)	V_{cp2}(V)	V_{cp3}(V)	V_{cp4}(V)	V_1(V)	V_{out}(V)	I_{src}
1	1	1	1	1	1	0	0
0	X	X	X	X	0	1	>0
X	0	X	X	X	0	1	>0
X	X	0	X	X	0	1	>0
X	X	X	0	X	0	1	>0
X	X	X	X	0	0	1	>0

A codificação que classifica um coeficiente como uma raiz de árvore de zeros é dada por V_{out}. É importante enfatizar que este circuito classifica erroneamente coeficientes que não são raízes de uma árvore de zeros como sendo, já que não há comunicação entre dois coeficientes separados na árvore hierárquica por mais de um nível de diferença (por exemplo, $LH4$ e $LH2$) para considerar nesta saída. Isto, porém, não modifica o resultado final já que o decodificador lê os coeficientes de maior nível na árvore primeiro e supostamente tem a capacidade de interpretar os resultados, evitando ler os coeficientes pertencentes às arvores de zeros.

Os valores das dimensões implementadas para os transistores são mostrados na Tabela 4.16. Os valores de M1 e M2 são válidos tanto para os circuitos na Figura 4.22(a) quanto para os circuitos na Figura 4.22(b) do diagrama esquemático.

Tabela 4.16: Dimensões dos transistores do codificador de árvore de zeros.

Transistor	W(μm)	L(μm)
M1 (pai e filhos)	1	2
M2 (pai e filhos)	0,4	0,35
M3 (somente pai)	0,4	2

A Figura 4.23 mostra os resultados da simulação elétrica transiente obtida para o circuito a parâmetros típicos, utilizando-se somente um filho do pai referido. O V_b foi obtido pela tentativa de espelhamento de uma corrente de ≈10 μA (correntes menores podem ser utilizadas com um aumento de L do transistor M3, de forma a reduzir o consumo do circuito). A Figura 4.24 mostra a variação dos resultados obtida para V_{out} através de dez rodadas de Monte Carlo com as mesmas entradas usadas na simulação da Figura 4.23.

4.9 Conversor Digital-Analógico

O conversor digital para analógico apresentado neste trabalho [17] possui cinco *bits* e opera em modo de corrente. Seu diagrama esquemático é apresentado na Figura 4.25.

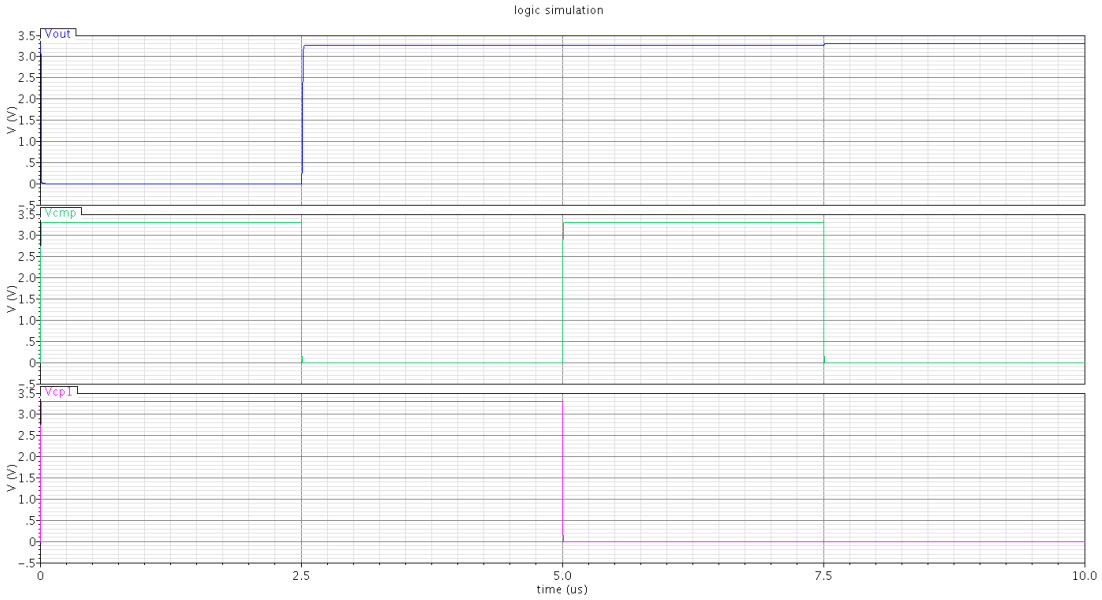

Figura 4.23: Simulação elétrica do circuito codificador de àrvore de zeros a parâmetros típicos.

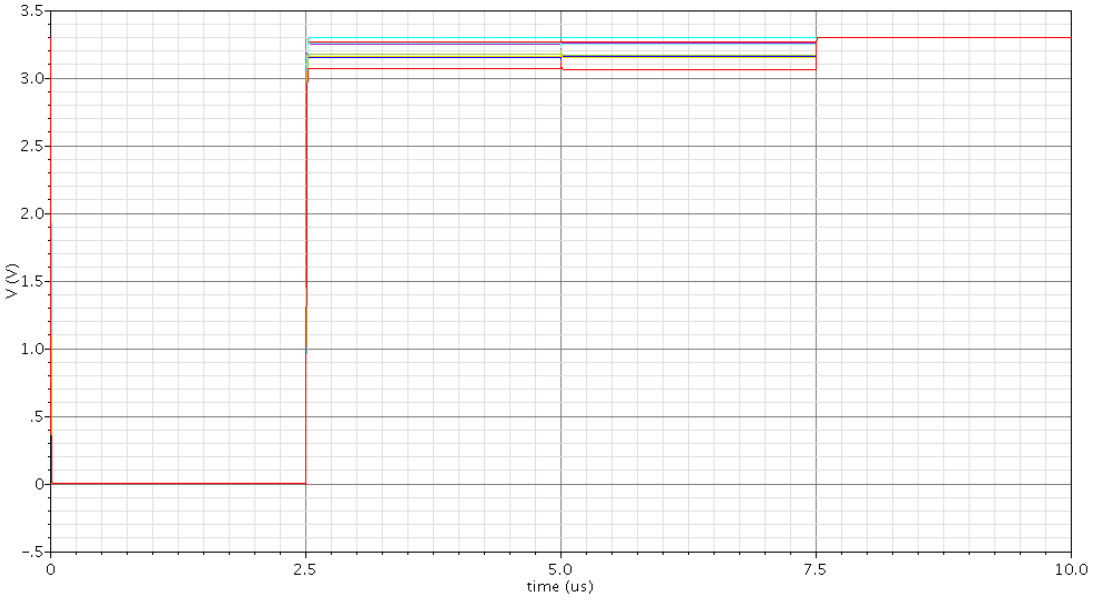

Figura 4.24: V_{out} obtido através de dez rodadas de Monte Carlo para o circuito codificador de àrvore de zeros.

Uma corrente I_{ref} igual à corrente máxima de saída do sensor APS é injetada pelo terminal de dreno do transistor M1. Esta corrente é copiada na razão $\frac{B}{A}$ para o dreno do transistor M3 através do espelho de corrente formado pelos transistores M1 e M2.

A corrente de M3 é também copiada para o terminal de dreno do transistor M4 na razão unitária através do espelho formado por estes dois transistores, gerando uma corrente I_N. Os transistores M5 e M7 são chaves alternativas, fazendo com

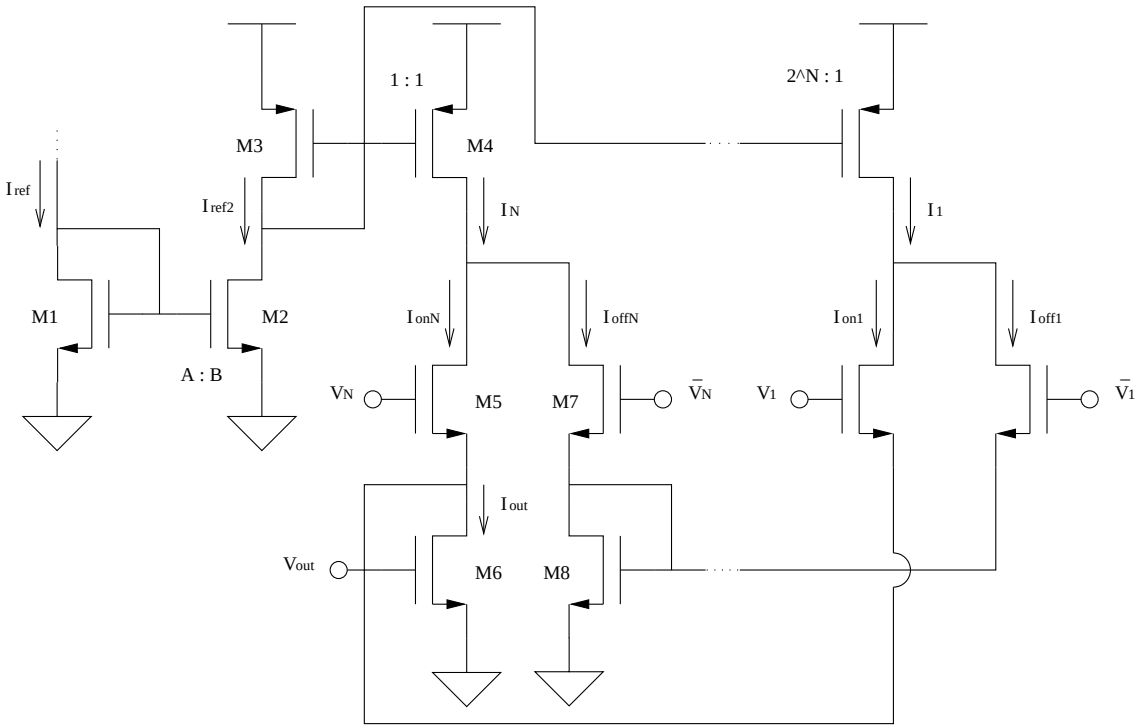

Figura 4.25: Diagrama esquemático do conversor D/A utilizado.

que a corrente I_N passe ou por M5 ou M7 dependendo do valor de V_N. Quando $V_N = 3,3$ V, a corrente passa por M5 e também por M6. Quando $V_N = 0$ V, a corrente passa por M7 e também por M8.

Na verdade, ocorre que existe um número $N-1$ de circuitos iguais ao formado pelos transistores M4, M5 e M7 ligados em paralelo a este, onde N é o número de *bits* desejado na entrada do conversor D/A. Nesta implementação, para um conversor D/A de cinco *bits* de entrada, foi feito $N = 5$. Os N transistores espelhando M3 geram as correntes I_N a I_1, cujos valores são progressivamente reduzidos à metade conforme sua ordem nesta sequência, devido à razão entre o número de transistores do espelho, que vai sendo dobrada.

Portanto, no caso genérico, o valor de uma corrente I_n no terminal de dreno de um transistor espelhando M3 é dado pela Equação (4.6), onde n é o n-ésimo *bit* da entrada do conversor D/A em ordem de significância ($n = 1$ é o *bit* menos significativo).

$$I_n = \frac{B}{A}\frac{1}{2^{(N-n)}}I_{ref} \tag{4.6}$$

Estas correntes I_n vão sendo somadas no nó de dreno de M6 ou no nó de dreno de M8, dependendo da configuração das respectivas tensões de chaveamento V_N a V_1. Assim, a corrente I_{out} no terminal de dreno de M6 torna-se uma soma de frações de uma corrente de referência I_{ref}, podendo ser copiada através do espelhamento de

M6. O circuito implementa, portanto, um conversor D/A de pesos binários (*binary-weighted diginal to analog converter*, [17]) necessariamente com seu valor máximo igual a $\frac{2^{N+1}-1}{2^N}$ do valor máximo do *pixel*.

Neste trabalho o conversor D/A será utilizado para gerar um valor para o limiar T usado na comparação com os coeficientes das sub-bandas de alta frequência obtidos pelas transformadas. Estes coeficientes possuem em sua maioria valores baixos, estando concentrados numa região bem próxima de zero e, por isto, a geração de valores muito altos para o limiar T não é interessante.

Para que não haja "desperdício" da resolução do conversor D/A, pode-se ajustar o valor do fator multiplicativo $\frac{B}{A}$. Para este trabalho, os valores para este ajuste foram calculados utilizando-se o método da quantização linear aplicado a um conjunto de coeficientes obtidos pela aplicação da transformada *wavelet* cinco vezes sobre as imagens de um banco de imagens. Esta quantização considerou o peso binário dos intervalos de resolução do conversor A/D, respeitando esta característica. O valor encontrado para o fator multiplicativo foi próximo de $\frac{1}{4}$ e arredondado para tal.

Este conversor possui duas características implementadas fim de se reduzir os erros de conversão. Primeiro, a existência do transistor M8, que é necessária para não levar ao corte um transistor qualquer que espelhe M3 quando ocorrer de sua tensão V_n ser igual a zero, já que isto mudaria as capacitâncias entre os seus terminais, gerando injeção de carga no circuito. Segundo, os espelhos de M3 são decrescentes em tamanho ao invés de crescentes, o que incorre em um aumento da área utilizada mas também numa melhoria do casamento entre os transistores dos espelhos dos *bits* de maior significância com M3.

A Tabela 4.17 mostra os resultados de I_{out} possíveis para a implementação realizada, com $N = 5$, $A = 4$ e $B = 1$.

Os valores das dimensões dos transistores implementados são dados pela Tabela 4.18. São mostrados apenas os tamanhos dos transistores nomeados na Figura 4.25, já que os omitidos possuem tamanho igual aos seus equivalentes funcionais, variando somente sua quantidade (multiplicidade) conforme descrito.

A Figura 4.26 mostra os resultados da simulação elétrica transiente obtida para o circuito a parâmetros típicos, variando-se as tensões V_n e com uma corrente $I_{ref} = 12$ μA. A Figura 4.27 mostra a variação dos resultados obtida para I_{out} através de dez rodadas de Monte Carlo com os mesmos sinais usados na simulação da Figura 4.26.

4.10 Conversor Analógico-Digital

O conversor analógico-digital utilizado é um conversor em modo de corrente com saída em código de Gray originalmente descrito em [18] e adaptado para utilizar o circuito de módulo descrito neste trabalho. Este conversor é utilizado na codificação

Tabela 4.17: Valores de corrente de saída para o conversor D/A.

V_5	V_4	V_3	V_2	V_1	I_{out}	V_5	V_4	V_3	V_2	V_1	I_{out}
0	0	0	0	0	0	1	0	0	0	0	$\frac{16}{64}I_{ref}$
0	0	0	0	1	$\frac{1}{64}I_{ref}$	1	0	0	0	1	$\frac{17}{64}I_{ref}$
0	0	0	1	0	$\frac{2}{64}I_{ref}$	1	0	0	1	0	$\frac{18}{64}I_{ref}$
0	0	0	1	1	$\frac{3}{64}I_{ref}$	1	0	0	1	1	$\frac{19}{64}I_{ref}$
0	0	1	0	0	$\frac{4}{64}I_{ref}$	1	0	1	0	0	$\frac{20}{64}I_{ref}$
0	0	1	0	1	$\frac{5}{64}I_{ref}$	1	0	1	0	1	$\frac{21}{64}I_{ref}$
0	0	1	1	0	$\frac{6}{64}I_{ref}$	1	0	1	1	0	$\frac{22}{64}I_{ref}$
0	0	1	1	1	$\frac{7}{64}I_{ref}$	1	0	1	1	1	$\frac{23}{64}I_{ref}$
0	1	0	0	0	$\frac{8}{64}I_{ref}$	1	1	0	0	0	$\frac{24}{64}I_{ref}$
0	1	0	0	1	$\frac{9}{64}I_{ref}$	1	1	0	0	1	$\frac{25}{64}I_{ref}$
0	1	0	1	0	$\frac{10}{64}I_{ref}$	1	1	0	1	0	$\frac{26}{64}I_{ref}$
0	1	0	1	1	$\frac{11}{64}I_{ref}$	1	1	0	1	1	$\frac{27}{64}I_{ref}$
0	1	1	0	0	$\frac{12}{64}I_{ref}$	1	1	1	0	0	$\frac{28}{64}I_{ref}$
0	1	1	0	1	$\frac{13}{64}I_{ref}$	1	1	1	0	1	$\frac{29}{64}I_{ref}$
0	1	1	1	0	$\frac{14}{64}I_{ref}$	1	1	1	1	0	$\frac{30}{64}I_{ref}$
0	1	1	1	1	$\frac{15}{64}I_{ref}$	1	1	1	1	1	$\frac{31}{64}I_{ref}$

Tabela 4.18: Dimensões dos transistores do conversor D/A.

Transistor	W(μm)	L(μm)	Quantidade	Transistor	W(μm)	L(μm)	Quantidade
M1	1	2	4	M5	0,4	0,35	1
M2	1	2	1	M6	0,4	0,35	1
M3	1	2	16	M7	1	2	1
M4	1	2	16	M8	1	2	1

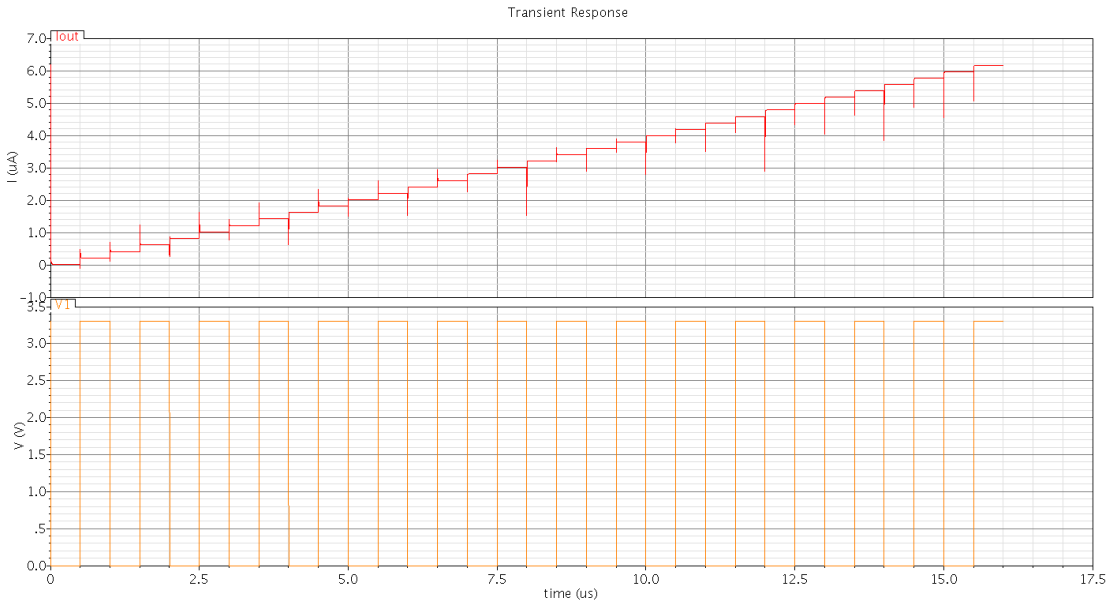

Figura 4.26: Simulação elétrica do conversor D/A a parâmetros típicos.

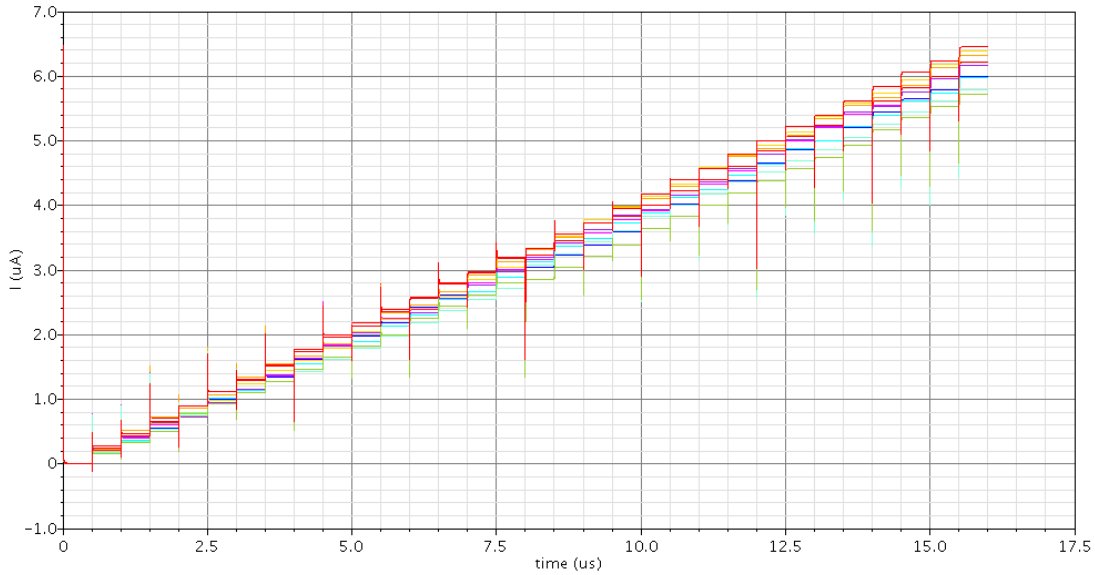

Figura 4.27: I_{out} obtido através de dez rodadas de Monte Carlo para o conversor D/A.

do módulo dos coeficientes da transformada, que necessariamente tem seu valor menor ou igual à corrente máxima do sensor APS.

A idéia geral por trás deste conversor analógico-digital é baseada numa operação de valor absoluto. Uma corrente fixa I_{ref} é subtraída de uma corrente de entrada a cada estágio e a corrente resultante é retificada e então dobrada em módulo. Pela repetição deste processo, mostrado na Figura 4.28, um código de Gray pode ser gerado.

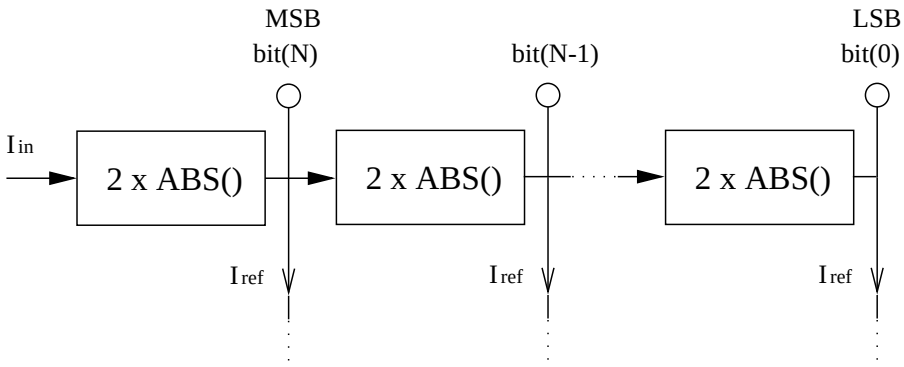

Figura 4.28: Idéia geral por trás de um conversor A/D com saída em código de Gray.

O diagrama esquemático do circuito que realiza a conversão de um *bit* é mostrado na Figura 4.29. Os transistores M1 a M5, juntamente do inversor INV1, formam um circuito de módulo idêntico ao já apresentado, fazendo valer a relação $I_{abs} = |I_{in}|$. Os transistores M6 e M8 são configurados como espelhos de M1, numa razão de dois para um, fazendo $I_{mir} = 2I_{abs}$. Os transistores M6 e M7, juntamente do inversor INV2, constituem um circuito comparador de corrente também igual ao já apresentado. Os

transistores M8 e M9 funcionam como um circuito de subtração similar ao operador HF, sendo ligados ao circuito de módulo do bloco do *bit* seguinte fazendo-se $I_{out}[bit$ N] $= I_{in}[bit$ N-1]. No bloco que gera o *bit* LSB, os transistores M8 e M9 são desnecessários e portanto foram removidos na implementação.

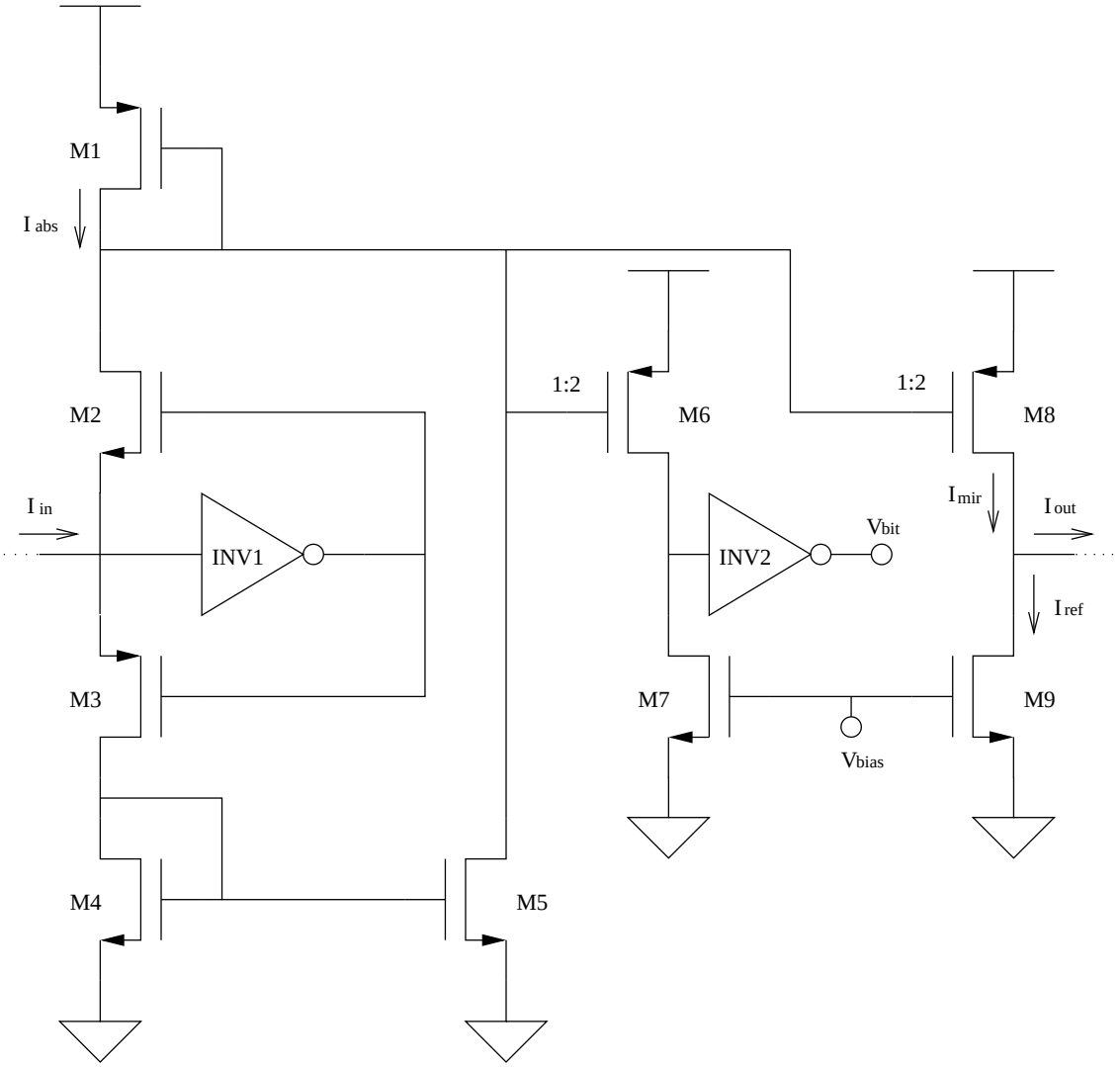

Figura 4.29: Diagrama esquemático de um bloco equivalente a um *bit* do conversor A/D.

Quando uma tensão V_{bias} for aplicada ao terminal de porta dos transistores M7 e M9 de todos os módulos de um *bit* ligados em série, no intuito de gerar por espelhamento uma corrente I_{ref} cujo valor é igual à corrente máxima do sensor APS, o comparador de corrente do circuito de cada módulo de um *bit* irá gerar as saídas descritas pela Tabela 4.19.

A implementação realizada neste trabalho utilizou cinco destes módulos para implementar um conversor A/D de cinco *bits* de resolução que com os dois *bits* de prefixo da codificação do algoritmo e o *bit* de sinal totaliza oito *bits*. A Tabela 4.20 mostra a codificação das saídas em relação à faixa de corrente de entrada.

Tabela 4.19: Sinais de saída do bloco de um *bit* do conversor A/D.

$I_{abs}[bit\ \mathrm{N}] = \lvert I_{in}[bit\ \mathrm{N}]\rvert$	$V_{bit}[bit\ \mathrm{N}](\mathrm{V})$	$\lvert I_{out}[bit\ \mathrm{N}]\rvert = I_{abs}[bit\ \text{N-1}]$
$> \frac{I_{ref}}{2}$	0	$I_{abs}[bit\ \mathrm{N}] - I_{ref}$
$< \frac{I_{ref}}{2}$	3,3	$I_{ref} - I_{abs}[bit\ \mathrm{N}]$

Tabela 4.20: Codificação das saídas do conversor A/D pela faixa de corrente de entrada.

1^o bloco I_{in}	MSB *bit* 4	*bit* 3	*bit* 2	*bit* 1	LSB *bit* 0
$I_{in} < \frac{1}{32} I_{ref}$	1	0	0	0	0
$\frac{1}{32} I_{ref} < I_{in} < \frac{2}{32} I_{ref}$	1	0	0	0	1
$\frac{2}{32} I_{ref} < I_{in} < \frac{3}{32} I_{ref}$	1	0	0	1	1
$\frac{3}{32} I_{ref} < I_{in} < \frac{4}{32} I_{ref}$	1	0	0	1	0
$\frac{4}{32} I_{ref} < I_{in} < \frac{5}{32} I_{ref}$	1	0	1	1	0
$\frac{5}{32} I_{ref} < I_{in} < \frac{6}{32} I_{ref}$	1	0	1	1	1
$\frac{6}{32} I_{ref} < I_{in} < \frac{7}{32} I_{ref}$	1	0	1	0	1
$\frac{7}{32} I_{ref} < I_{in} < \frac{8}{32} I_{ref}$	1	0	1	0	0
$\frac{8}{32} I_{ref} < I_{in} < \frac{9}{32} I_{ref}$	1	1	1	0	0
$\frac{9}{32} I_{ref} < I_{in} < \frac{10}{32} I_{ref}$	1	1	1	0	1
$\frac{10}{32} I_{ref} < I_{in} < \frac{11}{32} I_{ref}$	1	1	1	1	1
$\frac{11}{32} I_{ref} < I_{in} < \frac{12}{32} I_{ref}$	1	1	1	1	0
$\frac{12}{32} I_{ref} < I_{in} < \frac{13}{32} I_{ref}$	1	1	0	1	0
$\frac{13}{32} I_{ref} < I_{in} < \frac{14}{32} I_{ref}$	1	1	0	1	1
$\frac{14}{32} I_{ref} < I_{in} < \frac{15}{32} I_{ref}$	1	1	0	0	1
$\frac{15}{32} I_{ref} < I_{in} < \frac{16}{32} I_{ref}$	1	1	0	0	0
$\frac{16}{32} I_{ref} < I_{in} < \frac{17}{32} I_{ref}$	0	1	0	0	0
$\frac{17}{32} I_{ref} < I_{in} < \frac{18}{32} I_{ref}$	0	1	0	0	1
$\frac{18}{32} I_{ref} < I_{in} < \frac{19}{32} I_{ref}$	0	1	0	1	1
$\frac{19}{32} I_{ref} < I_{in} < \frac{20}{32} I_{ref}$	0	1	0	1	0
$\frac{20}{32} I_{ref} < I_{in} < \frac{21}{32} I_{ref}$	0	1	1	1	0
$\frac{21}{32} I_{ref} < I_{in} < \frac{22}{32} I_{ref}$	0	1	1	1	1
$\frac{22}{32} I_{ref} < I_{in} < \frac{23}{32} I_{ref}$	0	1	1	0	1
$\frac{23}{32} I_{ref} < I_{in} < \frac{24}{32} I_{ref}$	0	1	1	0	0
$\frac{24}{32} I_{ref} < I_{in} < \frac{25}{32} I_{ref}$	0	0	1	0	0
$\frac{25}{32} I_{ref} < I_{in} < \frac{26}{32} I_{ref}$	0	0	1	0	1
$\frac{26}{32} I_{ref} < I_{in} < \frac{27}{32} I_{ref}$	0	0	1	1	1
$\frac{27}{32} I_{ref} < I_{in} < \frac{28}{32} I_{ref}$	0	0	1	1	0
$\frac{28}{32} I_{ref} < I_{in} < \frac{29}{32} I_{ref}$	0	0	0	1	0
$\frac{29}{32} I_{ref} < I_{in} < \frac{30}{32} I_{ref}$	0	0	0	1	1
$\frac{30}{32} I_{ref} < I_{in} < \frac{31}{32} I_{ref}$	0	0	0	0	1
$\frac{31}{32} I_{ref} < I_{in} < I_{ref}$	0	0	0	0	0

Os valores das dimensões dos transistores implementados para cada módulo de um *bit* são dados pela Tabela 4.21.

A Figura 4.30 mostra os resultados da simulação elétrica transiente obtida para

Tabela 4.21: Dimensões dos transistores do conversor A/D.

Transistor	W(μm)	L(μm)	Quantidade	Transistor	W(μm)	L(μm)	Quantidade
M1	1	2	2	M6	1	2	4
M2	0,4	0,35	1	M7	1	2	4
M3	0,4	0,35	1	M8	1	2	4
M4	1	2	1	M9	1	2	4
M5	1	2	1				

o circuito a parâmetros típicos, variando-se a corrente I_{in} conforme indicado. A Figura 4.31 mostra a variação dos resultados obtida para V_{bit4} através de dez rodadas de Monte Carlo com a mesma variação de I_{in} da simulação da Figura 4.30.

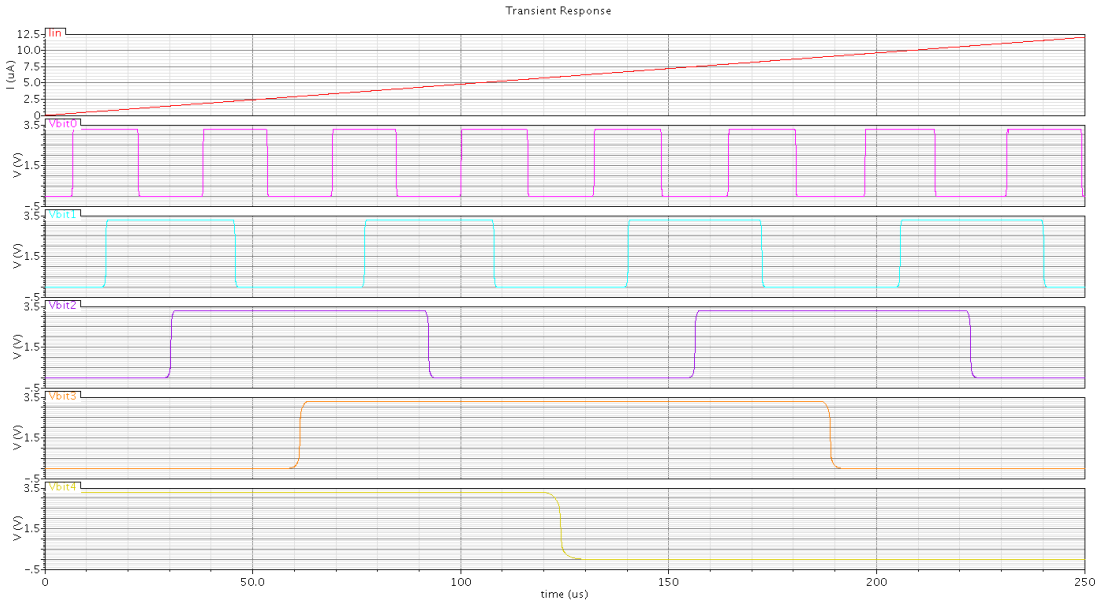

Figura 4.30: Simulação elétrica do conversor A/D a parâmetros típicos.

4.11 Registradores de Deslocamento

Os registradores de deslocamento utilizados foram implementados utilizando-se as células de flip-flop D disponibilizadas pela AMS.

Um registrador de deslocamento de entrada de 10 *bits* para a seleção da linha e da coluna do *pixel* e um outro de 5 *bits* para a configuração do limiar T foram implementados. O diagrama esquemático de um registrador de entrada genérico é dado pela Figura 4.32.

Também foi implementado um registrador de deslocamento de saída de 8 *bits* para a transmissão dos coeficientes. O diagrama esquemático de um registrador de saída genérico é dada pela Figura 4.33.

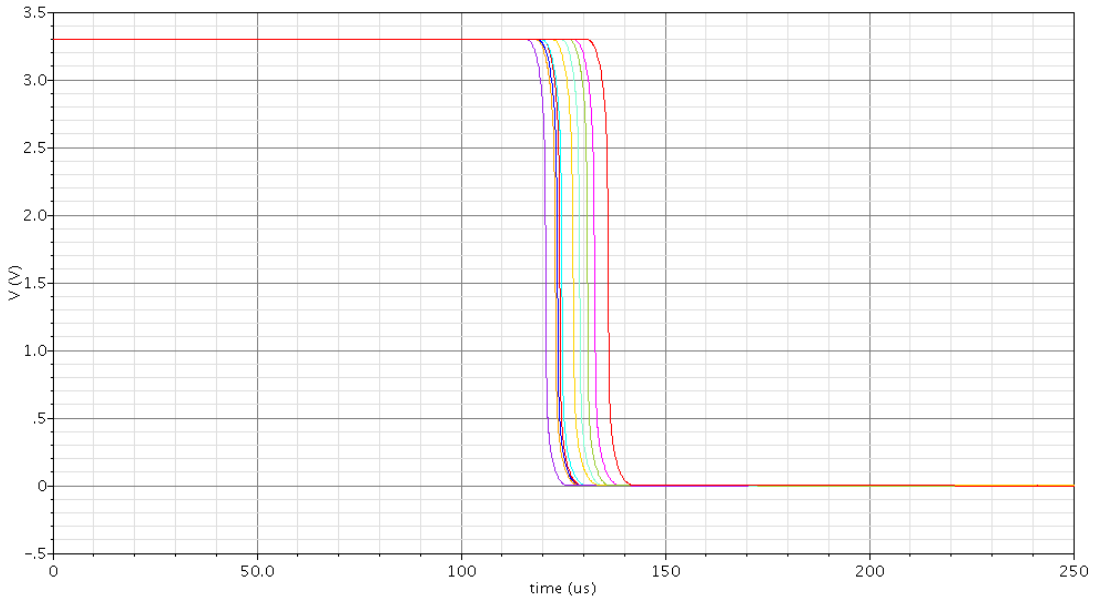

Figura 4.31: V_{bit4} obtido através de dez rodadas de Monte Carlo para o conversor A/D.

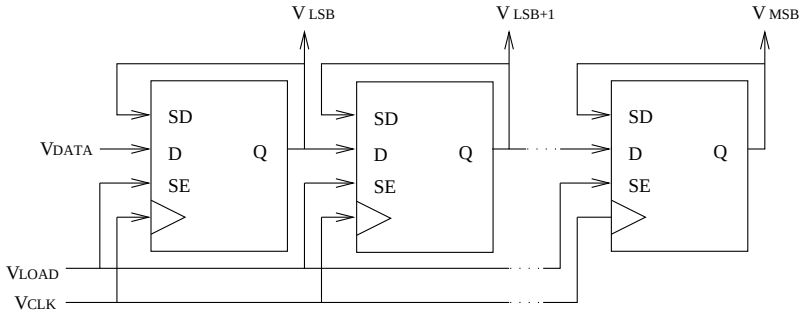

Figura 4.32: Registradores de deslocamento de entrada utilizados.

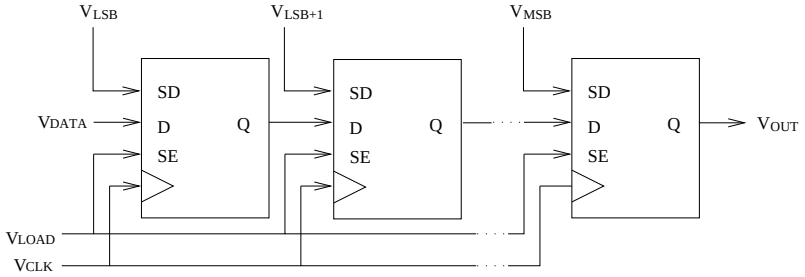

Figura 4.33: Registradores de deslocamento de saída utilizados.

4.12 Seletores de Linha e Coluna

Os seletores de linha e coluna funcionam por meio de portas lógicas. Os cinco primeiros *bits* enviados ao registrador de seleção são referentes à seleção da linha e os cinco últimos são referentes à seleção da coluna.

A seleção da primeira linha ou coluna do imageador é dada pela sequência 00000, e a da última pela sequência 11111. A Figura 4.34 ilustra o circuito utilizado para a realização da seleção de uma primeira linha ou coluna.

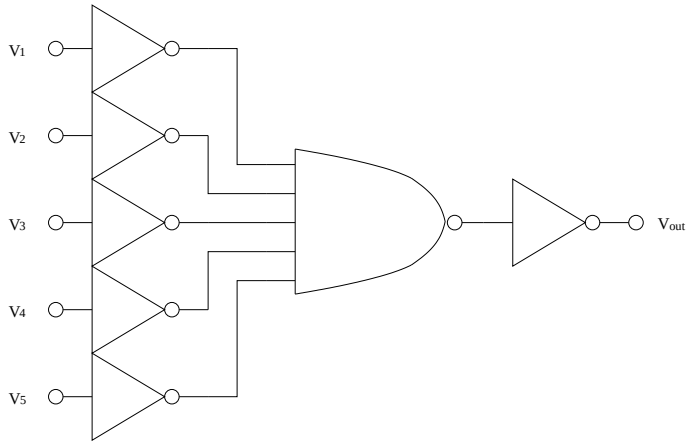

Figura 4.34: Lógica de seleção da primeira linha ou coluna do imageador.

4.13 *Layout*

Um *layout* foi criado para a fabricação do circuito. O imageador projetado ocupou uma área de 1687 μm por 1709 μm, com um *fill factor* um pouco maior do que 3%. A Figura 4.35 mostra o *layout* do chip a ser fabricado contendo o imageador proposto. Um outro circuito de teste, não relacionado com este trabalho, foi adicionado ao projeto e o total destes circuitos, incluindo os *pads*, ocupou uma área de aproximadamente 2596 μm por 2391 μm.

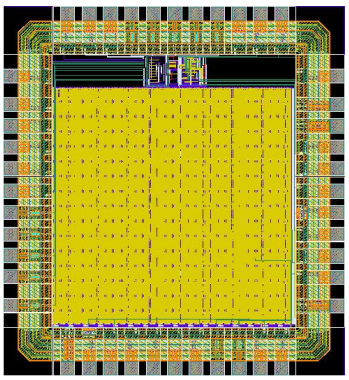

Figura 4.35: Chip contendo o circuito desenvolvido.

O processo utilizado dispõe de quatro camadas de metal, das quais três foram utilizadas: metal 1 (em azul), metal 2 (em branco) e metal 3 (em amarelo, a mais externa). A camada de metal 4 foi poupada e acabou não sendo utilizada. A única camada de silício policristalino é representada em vermelho. Em verde, encontram-se as difusões no substrato, incluindo-se entre estas a que forma o fotodiodo.

A Figura 4.36 mostra, de maneira mais aproximada, o *layout* de um quadrante da matriz contento 4 × 4 *pixels* e suas trilhas de ligações. Verifica-se que, em alguns quadrantes, existe um grande espaço não utilizado, sugerindo que uma melhoria na técnica de *layout* poderia reduzir o espaço ocupado pelo imageador, aumentando assim o *fill factor* do mesmo. Este espaço não utilizado foi preenchido por estruturas *dummy* para permitir a fabricação do cicuito.

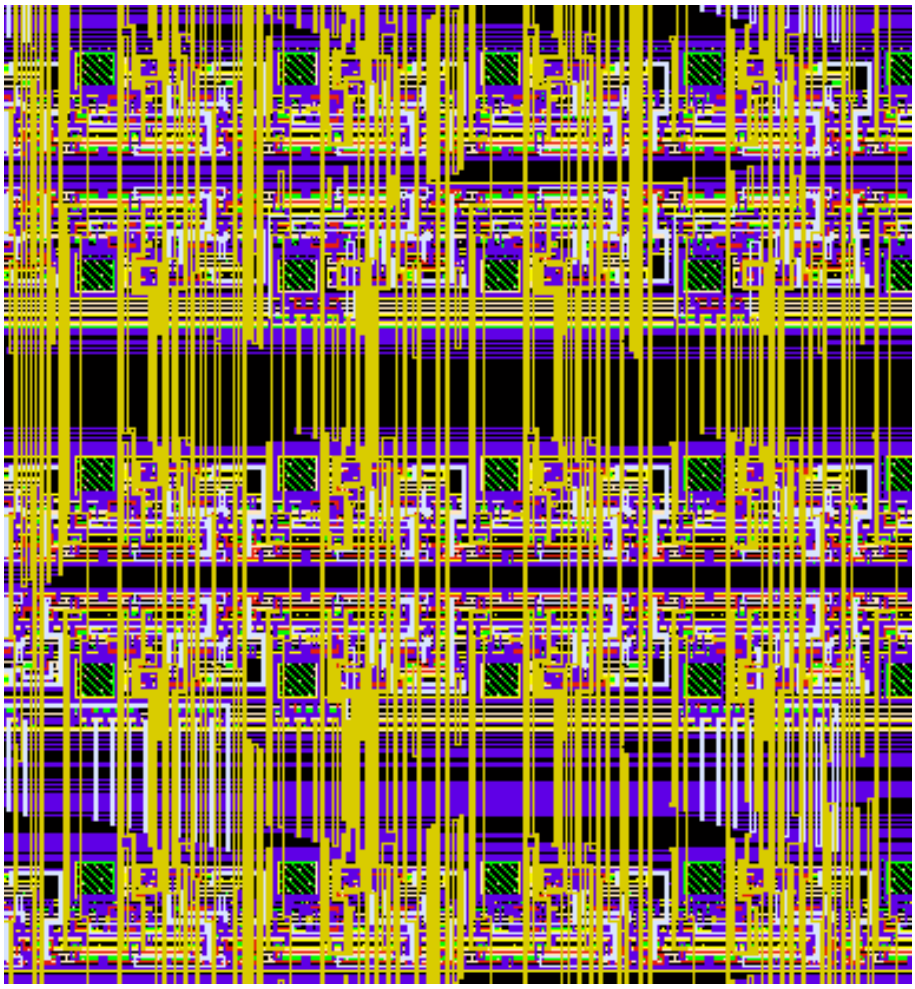

Figura 4.36: *Layout* de um grupo de 4 × 4 *pixels*.

A Figura 4.37 mostra de maneira mais detalhada o *layout* de um dos *pixels* desenvolvidos neste trabalho.

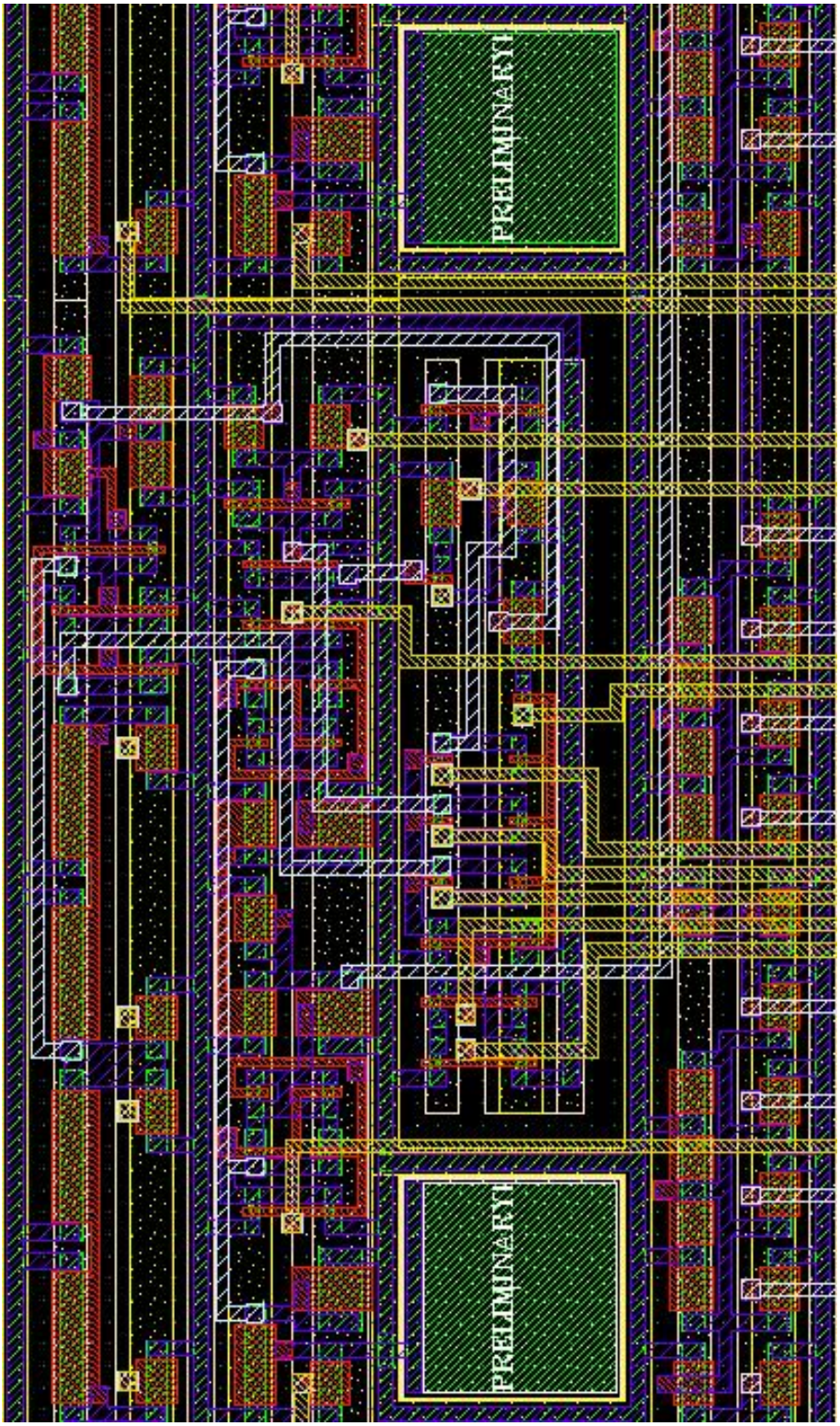

Figura 4.37: *Layout* de um *pixel*, rotacionado em noventa graus.

Capítulo 5

Resultados Obtidos

Este capítulo mostra alguns dos resultados obtidos pelas simulações computacionais do algoritmo EZW e pelas simulações elétricas do circuito. São fornecidas comparações entre os mesmos e a imagem sem compressão utilizada, uma região de 32×32 *pixels* da imagem "Lena"original de tamanho 512×512. Esta imagem foi linearmente quantizada de forma que cada *pixel* seu seja dado numa palavra de 8 *bits*.

Os indicadores escolhidos para serem utilizados nas comparações são a PSNR (Equação (1.1)) da imagem obtida e a taxa de compressão. Ambos os indicadores são sempre referentes à imagem sem compressão utilizada. Na PSNR, considera-se P_{max} com valor igual a 255. A taxa de compressão TC, por sua vez, é dada pela Equação (5.1). O número de *bits* da imagem codificada sem compressão é dado pelo produto entre o número de *bits* usados na codificação de um *pixel* desta imagem pelo número de *pixels* da imagem, resultando num total de 8192 *bits*.

$$TC = \frac{N^o \text{ de } bits \text{ da imagem codificada sem compressão}}{N^o \text{ de } bits \text{ da imagem codificada com compressão}} \tag{5.1}$$

A Tabela 5.1 resume a codificação realizada pelos circuitos implementados. O *bit* mais significativo (MSB), primeiro a ser transmitido, indica se o coeficiente é uma raiz de árvore de zeros. O *bit* seguinte indica se este é significante. O terceiro *bit* é o sinal do coeficiente, restando os últimos cinco *bits* para a codificação do módulo do mesmo.

Tabela 5.1: Decodificação do algoritmo EZW implementado.

*bit*8 (MSB)	*bit*7	*bit*6	Decodificação
0	0	X	Raiz de árvore de zeros, módulo zero
1	0	X	Zero isolado, módulo zero
1	1	0	Coeficiente negativo, módulo dado pelos *bits* 1 a 5
1	1	1	Coeficiente positivo, módulo dado pelos *bits* 1 a 5

Os módulos dos coeficientes devem ser decodificados utilizando-se a Tabela 4.20. As simulações computacionais no MATLAB, diferentemente das simulações elétricas, foram realizadas considerando-se um *bit* de sinal e oito de módulo, além dos outros dois *bits* do código de prefixo referente ao algoritmo EZW. É esperado, portanto, uma diferença tanto na PSNR quanto na taxa de compressão calculadas para os resultados obtidos por estas simulações.

A Figura 5.1(a) mostra a imagem "Olho da Lena"retirada da imagem original sem compressão. A Figura 5.1(b) mostra a imagem obtida pela decodificação dos resultados obtidos da simulação elétrica do circuito pelo Spectre para parâmetros típicos, utilizando-se $T = 0$ como valor do limiar.

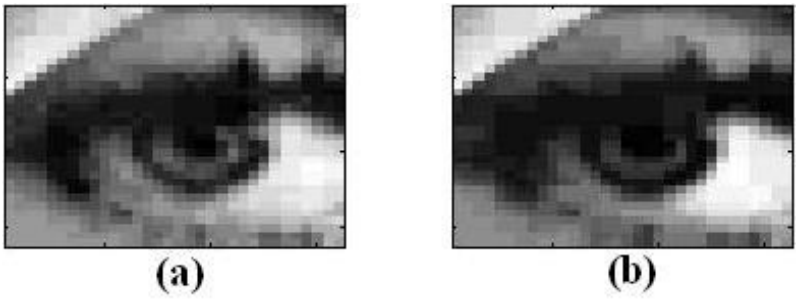

Figura 5.1: Imagens "Olho da Lena"original e a obtida pela simulação elétrica do circuito a um limiar zero.

O valor da PSNR obtida na Figura 5.1(b), considerando-se neste incluído o erro de quantização, foi de 27,32 dB. Não houve compressão já que $T = 0$.

A Figura 5.2 mostra os resultados obtidos por dez rodadas de simulação de Monte Carlo realizadas no simulador Spectre para a mesma imagem "Olho da Lena", considerando os efeitos das variações de processo e descasamento. As simulações foram realizadas para um limiar $T = 0$. A comparação entre resultados obtidos a um limiar nulo permite a verificação isoladamente da sensibilidade da parte do circuito referente ao cálculo dos coeficientes da transformada *wavelet*.

A Tabela 5.2 mostra o valor da PSNR para cada imagem obtida.

Tabela 5.2: Valores da PSNR para as imagens da Figura 5.2.

Linha	Coluna	PSNR (dB)	Linha	Coluna	PSNR (dB)
1	1	20,56	1	2	23,07
2	1	22,27	2	2	23,22
3	1	22,98	3	2	23,88
4	1	22,37	4	2	21,51
5	1	21,68	5	2	23,15

Uma análise dos resultados obtidos sugere que as variações de processo e descasamento entre transistores, mesmo com os cuidados tomados, ainda interferem de forma significativa na qualidade do resultado final.

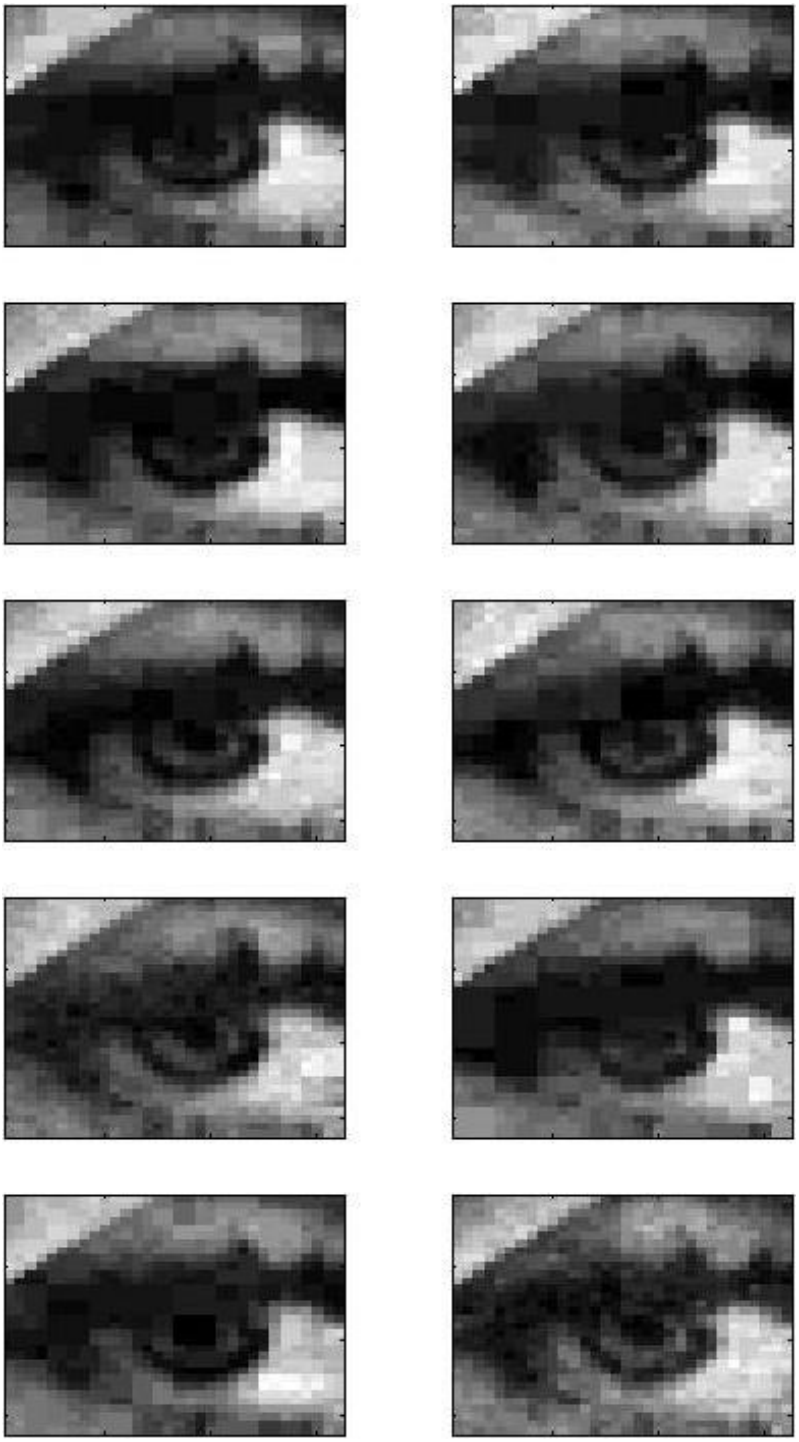

Figura 5.2: Simulações elétricas de Monte Carlo do circuito para "Olho da Lena"com um limiar $T = 0$.

Para verificar a funcionalidade dos circuitos referentes à codificação de árvore de zeros, realizou-se novamente as simulações anteriores, desta vez a um limiar T igual a $\frac{3}{64}$ do valor máximo da corrente do sensor APS. A Figura 5.3(a) mostra os resultados obtidos pela simulação computacional e a Figura 5.3(b) os obtidos pela simulação elétrica a parâmetros típicos.

Para a Figura 5.3(a), a PSNR obtida foi de 27,6 dB e foram lidos um total

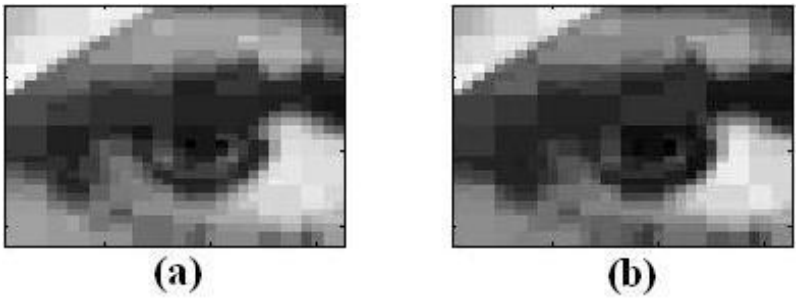

Figura 5.3: Simulações de "Olho da Lena"a um limiar T igual a $\frac{3}{64}$ do valor máximo da corrente do sensor APS.

de 3695 *bits*, gerando uma taxa de compressão de aproximadamente 3,05. Para a Figura 5.3(b), a PSNR obtida foi de 24,94 dB e foram lidos um total de 3596 *bits*, o que resulta numa taxa de compressão de 3,13 de acordo com os parâmetros estabelecidos para cada tipo de simulação. A diferença na taxa de compressão pode ser justificada pela diferente quantidade de *bits* utilizada na codificação dos coeficientes nos dois casos e pela possível variação no número de árvores de zeros causada pelos erros no cálculo dos coeficientes, como visto para o caso onde $T = 0$.

A Figura 5.4 mostra os resultados obtidos por quatro rodadas de simulação de Monte Carlo realizadas no simulador Spectre, considerando os efeitos das variações de processo e descasamento. Todas as simulações foram realizadas para o mesmo limiar anterior de $\frac{3}{64}$ do valor máximo da corrente do sensor APS.

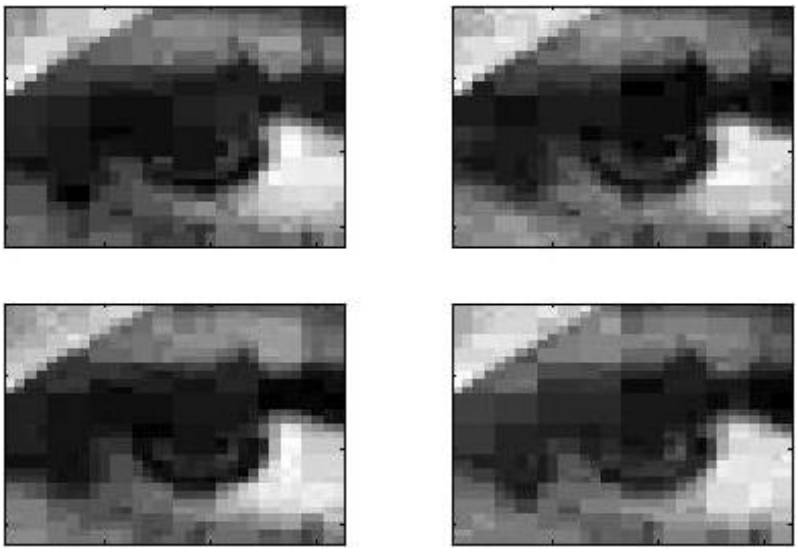

Figura 5.4: Simulações elétricas de Monte Carlo do circuito para "Olho da Lena"com um limiar T igual a $\frac{3}{64}$ do valor máximo da corrente do sensor APS.

A Tabela 5.3 mostra o valor da PSNR e da taxa de compressão TC obtidas para cada imagem.

Nota-se pelos resultados obtidos que a taxa de compressão calculada sofre pouca variação, podendo inclusive se aproximar mais do valor esperado de 3,05 (obtido pela

Tabela 5.3: Valores da PSNR para as imagens da Figura 5.4.

Linha	Coluna	PSNR (dB)	Número de *bits*	TC
1	1	21,51	3680	≈3,06
2	1	20,82	3716	≈3,03
1	2	22,84	4004	≈2,81
2	2	23,63	3650	≈3,09

simulação matemática do algoritmo) do que a própria taxa obtida pela simulação elétrica sem as variações de processo. Porém, uma análise mais detalhada dos valores obtidos para os coeficientes significativos mostra que aqueles obtidos pelas simulações elétricas com variações de processo tendem a se distanciar mais do valor matematicamente previsto, indicando que possa haver um aumento do número de árvores de zeros geradas erroneamente em alguns casos.

Por fim, realiza-se mais um par de simulações para constatar o ganho de compressão e perda de qualidade da imagem pelo incremento de mais um nível na escala do limiar T. A Figura 5.5 mostra os resultados obtidos para "Olho da Lena"comprimida pelo algoritmo EZW a um limiar igual a $\frac{4}{64}$ do valor máximo da corrente do sensor APS pela simulação computacional (Figura 5.5(a)) e pela simulação elétrica a parâmetros típicos (Figura 5.5(b)).

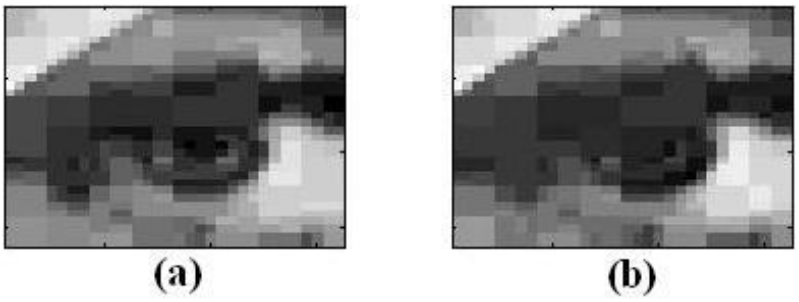

Figura 5.5: Simulações de "Olho da Lena"a um limiar T igual a $\frac{4}{64}$ do valor máximo da corrente do sensor APS.

A PSNR obtida para a Figura 5.5(a) foi de 25,72 dB com 2933 *bits* lidos, gerando uma taxa de compressão de aproximadamente 3,84. Já para a Figura 5.5(b), a PSNR foi de 24 dB e 3176 *bits* foram lidos. A taxa de compressão resultante foi cerca de 3,55.

Capítulo 6

Conclusões

O circuito apresentado realiza a compressão de imagens obtidas através de um imageador pela aplicação do algoritmo EZW no próprio plano focal. O circuito foi simulado para a confirmação do seu funcionamento, e os resultados foram comparados à simulações feitas por *software* pelo mesmo método de compressão.

Apesar dos ruídos causados pelas variações de processo, o circuito demonstrou que é capaz de obter imagens satisfatórias e com uma taxa de compressão razoável.

O fator de preenchimento do circuito, que é um pouco maior do que 3%, ficou abaixo do valor médio obtido para trabalhos do mesmo tipo (com ênfase em processamento de imagens no plano focal). Este valor é justificado pela implementação de cinco transformadas *wavelet* num mesmo imageador. O fator de preenchimento poderia ser aumentado pelo emprego de melhores técnicas de *layout*, sobretudo no roteamento dos sinais entre os *pixels*.

O circuito foi fabricado, mas até a conclusão deste trabalho não pôde ser testado.

6.1 Trabalhos Futuros

O imageador apresentado apresenta muitos pontos que podem ser reprojetados de maneira a tentar melhorar sua resposta final. Algumas das possíveis melhorias são sugeridas nesta seção.

Primeiramente, existe a questão da precisão dos circuitos utilizados. É possível determinar quais partes do circuito afetam os valores dos coeficientes de forma mais expressiva e que, portanto, necessitam serem reprojetadas para alcançarem maior precisão e acurácia. Como os circuitos são basicamente formados por espelhos de corrente, pode-se utilizar transistores com dimensões maiores, principalmente no comprimento, para a redução do descasamento entre estes. Uma outra alternativa seria a implementação de espelhos de corrente de Wilson, o que no entanto aumentaria o número de transistores.

Estas modificações, no entanto, serão viáveis apenas se houver um reprojeto do *layout* para se tentar um aumento do *fill factor*. Isto pode ser conseguido pela utilização da camada de metal 4, que não foi utilizada, para a simplificação de alguns roteamentos de sinais.

Para o sensor APS, é necessária uma melhoria da linearidade da corrente do transistor atuando como transcondutor. O problema é possivelmente causado devido às diferenças existentes entre as tensões de *threshold* do transistor de *reset* (NMOS) e do transistor atuando como transcondutor (PMOS) na célula 3-T, o que faz com que, imediatamente após o início do tempo de amostragem, o transistor que deveria atuar como transcondutor ainda esteja em corte. Em outras palavras, a linearidade é prejudicada devido à transição entre as regiões de corte e triodo deste transistor. Uma solução para este problema seria utilizar uma cascata de dois transistores NMOS para o *reset*, o que no entanto causaria uma perda na faixa dinâmica do circuito, ou mesmo poderia tirar o transistor atuando como transcondutor da região ôhmica. Outra solução seria realizar o controle da tensão do terminal de *gate* do transistor de *reset*, de forma a controlar a tensão sobre o nó de integração de cargas.

Sobre o acúmulo de erros no cálculo dos coeficientes da transformada, causado principalmente pelos seguidos blocos operadores LF, pode-se tentar uma redução do número dos mesmos. Um único bloco LF, cujos transistores possuam dimensões maiores, pode ser implementado em substituição a todos os contidos em um único *pixel*, de forma a reduzir o impacto das variações de processo no resultado final. Os quatro sinais de entrada deste bloco podem ser selecionados através de um chaveamento. Esta implementação foi a abordagem inicial deste projeto, que acabou sendo descartada devido à injeção de carga causada pelas chaves utilizadas. Mesmo assim, esta idéia é merecedora de um pouco mais de atenção pelo seu potencial de melhoria na resposta do circuito.

Uma outra questão diz respeito ao consumo de potência. O próprio modo de corrente já contribui para um aumento da potência utilizada pelo circuito. Em adição a este fator, alguns circuitos utilizados, como por exemplo o codificador de árvore de zeros, notoriamente consomem muita energia. Portanto, pode ser melhor substituir os mesmos pelo seu equivalente em modo de tensão, se for o caso em que o aumento da área utilizada (devido ao maior número de trilhas necessárias para o roteamento dos sinais, por exemplo) não diminua ainda mais o fator de preenchimento. Ou então, pode-se tentar realizar um controle de potência pelo chaveamento da fonte em alguns pontos, cortando o consumo de corrente destes durante a fase de leitura do circuito pelo microcontrolador externo.

Já os circuitos de seleção podem ser implementados por uma lógica de transistores ao invés da lógica de portas utilizada. Desta maneira há uma maior economia de espaço e uma maior velocidade de resposta do circuito, apesar de ser necessário

um maior tempo de projeto já que estes circuitos teriam que ser individualmente projetados para cada linha ou coluna.

Por fim, poderia ser interessante a tentativa de se implementar o circuito em modo de tensão ou com uma menor quantidade de transformadas realizadas, visando um aumento do fator de preenchimento, e então se realizar a comparação entre os resultados obtidos.

Referências Bibliográficas

[1] NAKAMURA, J. *Image Sensors and Signal Processing for Digital Still Cameras*. 1ª Edição, Flórida, USA, Taylor & Francis/CRC Press, 2005. ISBN: 0849335450.

[2] Tópicos de referência e informações de ajuda da Mathworks. Disponível em: <http://www.mathworks.com/help/vision/ref/psnr.html>, acessado em 23 de Março de 2013.

[3] LEÓN-SALAS, W. D., BALKIR, S., SAYOOD, K., et al. "A CMOS Imager With Focal Plane Compression Using Predictive Coding", *IEEE J. Solid-State Circuits*, v. 42, n. 11, pp. 2555-2572, Novembro de 2007.

[4] ARTYOMOV, E., YADID-PECHT, O. "Adaptive Multiple-resolution CMOS Active Pixel Sensor", *IEEE Trans. Circuits and Systems I: Regular Papers*, v. 53, n. 10, pp. 2178-2186, Outubro de 2006.

[5] CHI, Y. M., ABBAS, A., CHAKRABARTTY, S., et al. "An Active Pixel CMOS Separable Transform Image Sensor". In: *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 1281-1284, Taipei, Taiwan, Maio de 2009.

[6] OLIVEIRA, F. D. V. R., HAAS, H. L., GOMES, J. G. R. C., PETRAGLIA, A. "CMOS Imager With Focal-plane Analog Image Compression Combining DPCM And VQ". In: *IEEE Trans. Circuits and Systems I: Regular Papers*, artigo aceito para publicação e em produção final, DOI: 11.1109/TCSI.2012.2226505, 2013.

[7] SHAPIRO, J. M. "Embedded Image Coding Using Zerotrees Of Wavelets Coefficients", *IEEE Trans. Signal Processing*, v. 41, pp. 3445-3462, Dezembro de 1993.

[8] HAAR, A. "Zur Theorie der Orthogonalen Funktionen-Systeme", *Math. Ann.*, v. 69, pp. 331-371, Junho de 1910.

[9] BURRUS, C. S., GOPINATH, R.A., GUO, H. *Introduction to Wavelets and Wavelet Transforms: A Primer*. 1ª Edição, Texas, USA, Prentice-Hall, 1998. ISBN: 0134896009.

[10] FLEET, P. J. V. *Discrete Wavelet Transformations: An Elementary Approach with Applications*. 1ª Edição, New Jersey, USA, John Wiley & Sons, 2008. ISBN: 0470183113.

[11] RAZAVI, B. *Fundamentals of Microelectronics*. 1ª Edição, Los Angeles, USA, Wiley, 2008. ISBN: 0471478466.

[12] HAAS, H. L. *Análise da Sensibilidade de Compressão de Imagens no Plano Focal de Câmeras CMOS*. Universidade Federal do Rio de Janeiro, Projeto Final de Curso, Dezembro de 2008.

[13] PHILIPP, R. M., ORR, D., GRUEV, V., et al. "Linear Current-Mode Active Pixel Sensor", *IEEE J. Solid-State Circuits*, v. 42, n. 11, pp. 2482-2491, Novembro de 2007.

[14] FERRI, G., GUERRINI, N. C. *Low Voltage, Low Power CMOS Current Conveyors*. 1ª Edição, Massachusetts, USA, Kluwer Academic Publishers, 2003. ISBN: 1402074867.

[15] OLIVEIRA, F. D. V. R. *Circuito Integrado para Compressão de Imagens no Plano Focal utilizando Quantização Vetorial e DPCM*. Universidade Federal do Rio de Janeiro, Projeto Final de Curso, Janeiro de 2012.

[16] MEHTA, S., ETIENNE-CUMMINGS, R. "A Simplified Normal Optical Flow Measurement CMOS Camera", *IEEE Trans. Circuits and Systems I: Regular Papers*, v. 53, n. 6, pp. 1223-1234, Junho de 2006.

[17] KESTER, W. *The Data Conversion Handbook*. 1ª Edição, Massachusetts, USA, Newnes, 2005. ISBN: 0750678410.

[18] WILAMOWSKI, B.M., SINANGIL, M.E., DUNGAR, G. "A Gray-Code Current Mode ADC Structure". In: *IEEE Mediterranean Electrotechnical Conference*, pp. 35-38, Maio de 2006.

Apêndice A

Códigos Desenvolvidos no MATLAB

Este apêndice fornece alguns códigos escritos em linguagem MATLAB para auxiliar as simulações.

O primeiro código é referente a um *script* que realiza a transformada em *wavelets* de uma imagem gravada no arquivo "figura.mat"utilizando os filtros apresentados neste trabalho. O *script* então aplica a lógica das árvores de zeros do algoritmo EZW de acordo com um limiar de valor definido na variável T, reconstruindo a imagem de volta logo em seguida. Imagens da figura são exibidas durante o processo de computação. Ao final da restauração é calculada a PSNR da imagem obtida em relação à original.

EZW.m

```
% Bruno Bastos Cardoso - brunobc@pads.ufrj.br
% Script para compressão e restauração de uma imagem pelo
algoritmo EZW.

load figura.mat;
F=rgb2gray(F);
figure; imagesc(abs(F)); colormap('gray'); truesize;
F=double(F);

% Constantes

T=4; % Limiar da árvore de zeros, varia de 0 a 255
nmax=5; % Número de níveis de wavelets
```

```
FiltroLL=[1 1;1 1]/4;
FiltroHL=[1 -1;1 -1]/2;
FiltroLH=[1 1;-1 -1]/2;
FiltroHH=[1 -1;-1 1]/2;

% Structs

MLL=struct('F',{},'R',{});
MLH=struct('F',{},'R',{});
MHL=struct('F',{},'R',{});
MHH=struct('F',{},'R',{});

W=struct('F',{},'R',{});

% Decomposição em Wavelets

for n=1:1:nmax
  if n==1
    MLL(n).F=conv2(F,FiltroLL);
    MHL(n).F=conv2(F,FiltroHL);
    MLH(n).F=conv2(F,FiltroLH);
    MHH(n).F=conv2(F,FiltroHH);
  else
    MLL(n).F=conv2(MLL(n-1).F,FiltroLL);
    MHL(n).F=conv2(MLL(n-1).F,FiltroHL);
    MLH(n).F=conv2(MLL(n-1).F,FiltroLH);
    MHH(n).F=conv2(MLL(n-1).F,FiltroHH);
  end
  MLL(n).F=MLL(n).F(2:2:end,2:2:end);
  MHL(n).F=MHL(n).F(2:2:end,2:2:end);
  MLH(n).F=MLH(n).F(2:2:end,2:2:end);
  MHH(n).F=MHH(n).F(2:2:end,2:2:end);
  for m=n:-1:1
    if m==n
      W(n).F=[MLL(m).F MHL(m).F ; MLH(m).F MHH(m).F];
    else
      W(n).F=[W(n).F MHL(m).F ; MLH(m).F MHH(m).F];
    end
```

```
  end
  figure; imagesc(abs(W(n).F)); colormap('gray'); truesize;
end

% EZW

for n=nmax:-1:1
  for lin=1:1:size(MLL(n).F,1)
    for col=1:1:size(MLL(n).F,2)
      if abs(MHL(n).F(lin,col))<=T
        MHL(n).F(lin,col)=0;
        if n>1 & abs(MHL(n-1).F(2*lin-1:2*lin,2*col-1:2*col))<=T
          for k=n-1:-1:1
            MHL(k).F(2^(n-k)*lin-(2^(n-k)-1):2^(n-k)*lin,
2^(n-k)*col-(2^(n-k)-1):2^(n-k)*col)=0;
          end
        end
      end
      if abs(MLH(n).F(lin,col))<=T
        MLH(n).F(lin,col)=0;
        if n>1 & abs(MLH(n-1).F(2*lin-1:2*lin,2*col-1:2*col))<=T
          for k=n-1:-1:1
            MLH(k).F(2^(n-k)*lin-(2^(n-k)-1):2^(n-k)*lin,
2^(n-k)*col-(2^(n-k)-1):2^(n-k)*col)=0;
          end
        end
      end
      if abs(MHH(n).F(lin,col))<=T
        MHH(n).F(lin,col)=0;
        if n>1 & abs(MHH(n-1).F(2*lin-1:2*lin,2*col-1:2*col))<=T
          for k=n-1:-1:1
            MHH(k).F(2^(n-k)*lin-(2^(n-k)-1):2^(n-k)*lin,
2^(n-k)*col-(2^(n-k)-1):2^(n-k)*col)=0;
          end
        end
      end
    end
  end
```

```
end

% Restaurando o nível n

for n=nmax:-1:1
  MLL(n).R=zeros(2*size(MLL(n).F));
  MHL(n).R=zeros(2*size(MHL(n).F));
  MLH(n).R=zeros(2*size(MLH(n).F));
  MHH(n).R=zeros(2*size(MHH(n).F));
  if n==nmax
    MLL(n).R(1:2:end,1:2:end)=MLL(n).F;
  else
    MLL(n).R(1:2:end,1:2:end)=MLL(n+1).R + MHL(n+1).R + MLH(n+1).R
+ MHH(n+1).R;
  end
MLL(n).R=conv2(MLL(n).R,FiltroLL*4);
MLL(n).R=MLL(n).R(1:end-1,1:end-1);

MHL(n).R(1:2:end,1:2:end)=MHL(n).F;
MHL(n).R=conv2(MHL(n).R,fliplr(FiltroHL));
MHL(n).R=MHL(n).R(1:end-1,1:end-1);

MLH(n).R(1:2:end,1:2:end)=MLH(n).F;
MLH(n).R=conv2(MLH(n).R,flipud(FiltroLH));
MLH(n).R=MLH(n).R(1:end-1,1:end-1);

MHH(n).R(1:2:end,1:2:end)=MHH(n).F;
MHH(n).R=conv2(MHH(n).R,FiltroHH);
MHH(n).R=MHH(n).R(1:end-1,1:end-1);
end

% Criando imagens e calculando PSNR

for n=nmax:-1:1
  if n==nmax
    W(n).R=[MLL(n).F MHL(n).F ; MLH(n).F MHH(n).F];
  else
    W(n).R=[[MLL(n+1).R+MHL(n+1).R+MLH(n+1).R+MHH(n+1).R] MHL(n).F
```

```
; MLH(n).F MHH(n).F];
  end
  if n>1
    for m=n-1:-1:1]
      W(n).R=[W(n).R MHL(m).F ; MLH(m).F MHH(m).F];
    end
  end
  figure; imagesc(abs(W(n).R)); colormap('gray'); truesize;
end

R=MLL(1).R+MHL(1).R+MLH(1).R+MHH(1).R;
figure; imagesc(abs(R)); colormap('gray'); truesize;

PSNR=10*log10((255.^2)/(mean(mean((R-F).^2))))
```

Nos esquemáticos implementados, para cada *pixel*, existe uma fonte de corrente ideal posicionada no respectivo nó de integração do seu fotodiodo. Os valores destas fontes são passados na forma de variáveis para a instância mais externa do circuito no netlist, onde estes valores são então definidos.

Este segundo *script* gera um arquivo "sources.txt"em formato de *netlist* para o simulador Spectre contendo valores para as variáveis das fontes citadas. Estes valores gerados fazem com que a drenagem de cargas elétricas causada pelas fontes de corrente seja equivalente à incidência luminosa nos *pixels* como quando expostos à da imagem salva no arquivo "figura.mat", permitindo uma simulação visual do resultado do circuito completo. As linhas de texto geradas devem substituir as linhas referentes ao valor destas variáveis dentro do arquivo *netlist* original, que precisam ser identificadas e copiadas manualmente.

A lógica que dá o nome das variáveis destas fontes de corrente foi feita de forma a seguir a identificação da árvore hierárquica ao qual pertence o coeficiente fornecido pelo seu *pixel*. Para sub-bandas com mais de um coeficiente, devido a restrições quanto ao número e ao tipo de caracteres no nome das variáveis, utiliza-se uma notação ABCD em referência às árvores mostradas nas Figuras 3.5 a 3.8. Nesta notação, A é sempre o *pixel* filho mais ao topo e à esquerda, B o mais ao topo e à direita, C o mais em baixo e à esquerda e D o mais em baixo e à direita.

Por exemplo, para se saber os índice de linha e coluna do *pixel* cuja variável da fonte de corrente recebe o nome $LH2CDB$, encontra-se primeiramente na árvore LH da Figura 3.5 o seu filho pertencente ao no quadrante C, isto é, o *pixel* localizado nas coordenadas (25,1) do imageador. Em seguida, busca-se na Figura 3.6 a árvore

em que o *pixel* (25,1) seja o pai, e nela localiza-se seu filho no quadrante D - no caso o *pixel* (29,9). Por fim, na Figura 3.7, encontra-se a árvore onde (29,9) é o pai e em seguida seu filho no quadrante B, que é o *pixel* (27,13). Logo, a fonte de corrente de valor $LH2CDB$ é a fonte que está ligada ao fotodiodo do *pixel* de coordenadas (27,13) do imageador.

Fig2Net.m

```
% Bruno Bastos Cardoso - brunobc@pads.ufrj.br
% Script para geração de fontes em formato netlist.

% Constantes

nmax=5; % Imagem de dimensões EXATAMENTE iguais a 2^nmax
ImagePath='Images\'; % Caminho parcial para o diretório de imagens

% Carregando Imagens

PIC(1).name='figura.mat'; % Arquivo com a matriz de valores
inteiros representando os pixels da figura
load([ImagePath PIC(1).name]);
F=rgb2gray(F);

% Convertendo valores de luminância sobre os fotodiodos como
corrente entre 1e-12A e 400e-12A

FIG=(double(F)/255).*(399e-12) + 1e-12; % Valores para fontes

% Configurando valores e nomes das fontes

n=1;
cont(n).nam='LL5';
cont(n).val=FIG(1,1);
n=n+1;

for sca=nmax:-1:1

  % Orientação HL
```

```
for lin=1:2^sca:(1+2^nmax-2^sca)
  for col=(1+2^(sca-1)):2^sca:(1+2^nmax-2^(sca-1))
    cont(n).val = FIG(lin,col);
    nameBuffer='HL';
    nameBuffer=[nameBuffer num2str(sca)];
    if sca<nmax
      for ord=(nmax-1):-1:sca
        quadLin=(mod(floor((lin-1)./2^ord),2)==0);
        quadCol=(mod(floor((col-1)./2^ord),2)==0);
        if quadLin
          if quadCol
            nameBuffer=[nameBuffer 'A'];
          else
            nameBuffer=[nameBuffer 'B'];
          end
        else
          if quadCol
            nameBuffer=[nameBuffer 'C'];
          else
            nameBuffer=[nameBuffer 'D'];
          end
        end
      end
    end
    cont(n).nam = nameBuffer;
    n=n+1;
  end
end

% Orientação LH

for lin=(1+2^(sca-1)):2^sca:(1+2^nmax-2^(sca-1))
  for col=1:2^sca:(1+2^nmax-2^sca)
    cont(n).val = FIG(lin,col);
    nameBuffer='LH';
    nameBuffer=[nameBuffer num2str(sca)];
    if sca<nmax
      for ord=(nmax-1):-1:sca
```

```
        quadLin=(mod(floor((lin-1)./2^ord),2)==0);
        quadCol=(mod(floor((col-1)./2^ord),2)==0);
        if quadLin
          if quadCol
            nameBuffer=[nameBuffer 'A'];
          else
            nameBuffer=[nameBuffer 'B'];
          end
        else
          if quadCol
            nameBuffer=[nameBuffer 'C'];
          else
            nameBuffer=[nameBuffer 'D'];
          end
        end
      end
    end
    cont(n).nam = nameBuffer;
    n=n+1;
  end
end

% Orientação HH

for lin=(1+2^(sca-1)):2^sca:(1+2^nmax-2^(sca-1))
  for col=(1+2^(sca-1)):2^sca:(1+2^nmax-2^(sca-1))
    cont(n).val = FIG(lin,col);
    nameBuffer='HH';
    nameBuffer=[nameBuffer num2str(sca)];
    if sca<nmax
      for ord=(nmax-1):-1:sca
        quadLin=(mod(floor((lin-1)./2^ord),2)==0);
        quadCol=(mod(floor((col-1)./2^ord),2)==0);
        if quadLin
          if quadCol
            nameBuffer=[nameBuffer 'A'];
          else
            nameBuffer=[nameBuffer 'B'];
```

```
            end
          else
            if quadCol
              nameBuffer=[nameBuffer 'C'];
            else
              nameBuffer=[nameBuffer 'D'];
            end
          end
        end
      end
      cont(n).nam = nameBuffer;
      n=n+1;
    end
  end
end

% Gerando um arquivo txt no formato netlist apenas com as fontes

fname = 'sources.txt';
fid = fopen(fname, 'w');
for n=1:1:2^(2*nmax)
  if mod(n,4)==1
    if n==1
      fprintf(fid, 'parameters ');
    else
      fprintf(fid, ' ');
    end
  end
  fprintf(fid, [cont(n).nam '=%5.3e '], cont(n).val);
  if mod(n,4)==0
    fprintf(fid, '\\\r\n');
  end
end
fclose(fid);
```

Este terceiro *script* lê um arquivo "results.csv"com os resultados das saídas do circuito após a sua simulação elétrica (transiente, no caso) obtida pelo simulador CADENCE Spectre. Os dados no arquivo em questão estão separados uns dos

outros por vírgulas em colunas (csv significa *comma-separated values*), e o mesmo é exportado pelo *software* CADENCE Custom IC Design Tools. O script então grava os dados em um arquivo "csvData.mat", no formato de dados usado pelo MATLAB.

Csv2Matlab.m

```
% Bruno Bastos Cardoso - brunobc@pads.ufrj.br
% Script para leitura dos resultados em um arquivo CSV.

csvFile = dlmread('results.csv', ',', 1, 0);

time=csvFile(:,1);
colA=csvFile(:,2);
colB=csvFile(:,3);
colC=csvFile(:,4);
colD=csvFile(:,5);
colE=csvFile(:,6);
linA=csvFile(:,7);
linB=csvFile(:,8);
linC=csvFile(:,9);
linD=csvFile(:,10);
linE=csvFile(:,11);
out0=csvFile(:,12);
out1=csvFile(:,13);
out2=csvFile(:,14);
out3=csvFile(:,15);
out4=csvFile(:,16);
out5=csvFile(:,17);
out6=csvFile(:,18);
out7=csvFile(:,19);
P1=csvFile(:,20);
P2=csvFile(:,21);
RST=csvFile(:,22);

save Data/csvData.mat time colA colB colC colD colE linA linB linC
linD linE out0 out1 out2 out3 out4 out5 out6 out7 P1 P2 RST;
```

Por último, este quarto *script* carrega o arquivo "csvData.mat", converte as saídas lidas para dados digitais, decodifica os valores referentes aos módulos dos

coeficientes da *wavelet* lidos, remonta os coeficientes da transformada em *wavelet* pela lógica do algoritmo EZW e finalmente restaura a imagem, exibindo-a na tela. Também carrega a imagem "figura.mat"e calcula a PSNR.

Matlab2Picture.m

```
% Bruno Bastos Cardoso - brunobc@pads.ufrj.br
% Exibição visual de uma simulação elétrica.

load Data/csvData.mat time colA colB colC colD colE linA linB linC
linD linE out0 out1 out2 out3 out4 out5 out6 out7 P1 P2 RST;

nmax=5;
limLog=1.15 % Limiar entre os estados lógicos

FiltroLL=[1 1;1 1]/4;
FiltroHL=[1 -1;1 -1]/2;
FiltroLH=[1 1;-1 -1]/2;
FiltroHH=[1 -1;-1 1]/2;

MLL=struct('F',{},'R',{},'ZT',{});
MLH=struct('F',{},'R',{},'ZT',{});
MHL=struct('F',{},'R',{},'ZT',{});
MHH=struct('F',{},'R',{},'ZT',{});

sampleP=1e-6; % Tempo de amostagem das saídas digitais
startT=104.5e-6; % Tempo de ínicio da amostragem

n=1;
out=struct('bin',{},'int',{});
for decTime=(startT-sampleP/5):(sampleP/2):time(size(time,1))
  iDec=max(find(time<=decTime));
  out(n).bin=[out7(iDec) out6(iDec) out5(iDec) out4(iDec)
out3(iDec) out2(iDec) out1(iDec) out0(iDec)];
  out(n).bin=out(n).bin>1.15;
  out(n).int=out(n).bin(4)*2^4 + out(n).bin(5)*2^3 +
out(n).bin(6)*2^2 + out(n).bin(7)*2^1 + out(n).bin(8)*2^0;
```

```
  n=n+1;
end


n=1;
for lin=1:1:2^nmax
  for col=1:1:2^nmax
    if out(n).bin(1)==1
      image(lin,col)=0;
    else
      switch out(n).int % Revertendo os bits do ADC em código de
Gray
        case 31 % 11111
          image(lin,col)=((-1)^out(n).bin(3))*10;
        case 30 % 11110
          image(lin,col)=((-1)^out(n).bin(3))*11;
        case 29 % 11101
          image(lin,col)=((-1)^out(n).bin(3))*9;
        case 28 % 11100
          image(lin,col)=((-1)^out(n).bin(3))*8;
        case 27 % 11011
          image(lin,col)=((-1)^out(n).bin(3))*13;
        case 26 % 11010
          image(lin,col)=((-1)^out(n).bin(3))*12;
        case 25 % 11001
          image(lin,col)=((-1)^out(n).bin(3))*14;
        case 24 % 11000
          image(lin,col)=((-1)^out(n).bin(3))*15;
        case 23 % 10111
          image(lin,col)=((-1)^out(n).bin(3))*5;
        case 22 % 10110
          image(lin,col)=((-1)^out(n).bin(3))*4;
        case 21 % 10101
          image(lin,col)=((-1)^out(n).bin(3))*6;
        case 20 % 10100
          image(lin,col)=((-1)^out(n).bin(3))*7;
        case 19 % 10011
          image(lin,col)=((-1)^out(n).bin(3))*2;
        case 18 % 10010
```

```
        image(lin,col)=((-1)^out(n).bin(3))*3;
      case 17 % 10001
        image(lin,col)=((-1)^out(n).bin(3))*1;
      case 16 % 10000
        image(lin,col)=((-1)^out(n).bin(3))*0;
      case 15 % 01111
        image(lin,col)=((-1)^out(n).bin(3))*21;
      case 14 % 01110
        image(lin,col)=((-1)^out(n).bin(3))*20;
      case 13 % 01101
        image(lin,col)=((-1)^out(n).bin(3))*22;
      case 12 % 01100
        image(lin,col)=((-1)^out(n).bin(3))*23;
      case 11 % 01011
        image(lin,col)=((-1)^out(n).bin(3))*18;
      case 10 % 01010
        image(lin,col)=((-1)^out(n).bin(3))*19;
      case 9 % 01001
        image(lin,col)=((-1)^out(n).bin(3))*17;
      case 8 % 01000
        image(lin,col)=((-1)^out(n).bin(3))*16;
      case 7 % 00111
        image(lin,col)=((-1)^out(n).bin(3))*26;
      case 6 % 00110
        image(lin,col)=((-1)^out(n).bin(3))*27;
      case 5 % 00101
        image(lin,col)=((-1)^out(n).bin(3))*25;
      case 4 % 00100
        image(lin,col)=((-1)^out(n).bin(3))*24;
      case 3 % 00011
        image(lin,col)=((-1)^out(n).bin(3))*29;
      case 2 % 00010
        image(lin,col)=((-1)^out(n).bin(3))*28;
      case 1 % 00001
        image(lin,col)=((-1)^out(n).bin(3))*30;
      case 0 % 00000
        image(lin,col)=((-1)^out(n).bin(3))*31;
    end
```

```
    end
    zerotree(lin,col)=out(n).bin(2);
    n=n+1;
  end
end

for n=nmax:-1:1
  MLL(n).F=zeros(2^(nmax-n));
  MHL(n).F=image(1:2^n:end,1+2^(n-1):2^n:end);
  MHL(n).ZT=zerotree(1:2^n:end,1+2^(n-1):2^n:end);
  MLH(n).F=image(1+2^(n-1):2^n:end,1:2^n:end);
  MLH(n).ZT=zerotree(1+2^(n-1):2^n:end,1:2^n:end);
  MHH(n).F=image(1+2^(n-1):2^n:end,1+2^(n-1):2^n:end);
  MHH(n).ZT=zerotree(1+2^(n-1):2^n:end,1+2^(n-1):2^n:end);
end
MLL(nmax).F=image(1,1);
MLL(n).ZT=zerotree(1,1);

for n=nmax:-1:2   for lin=1:1:2^(nmax-n)
    for col=1:1:2^(nmax-n)
      if MHL(n).ZT(lin,col)==1
        MHL(n-1).F(2*lin-1:2*lin,2*col-1:2*col)=0;
        MHL(n-1).ZT(2*lin-1:2*lin,2*col-1:2*col)=1;
      end
      if MLH(n).ZT(lin,col)==1
        MLH(n-1).F(2*lin-1:2*lin,2*col-1:2*col)=0;
        MLH(n-1).ZT(2*lin-1:2*lin,2*col-1:2*col)=1;
      end
      if MHH(n).ZT(lin,col)==1
        MHH(n-1).F(2*lin-1:2*lin,2*col-1:2*col)=0;
        MHH(n-1).ZT(2*lin-1:2*lin,2*col-1:2*col)=1;
      end
    end
  end
end

for n=nmax:-1:1
  if n~=1
```

```
    MLL(n).R=zeros(size(MLL(n-1).F));
    MLH(n).R=zeros(size(MLL(n-1).F));
    MHL(n).R=zeros(size(MLL(n-1).F));
    MHH(n).R=zeros(size(MLL(n-1).F));
  else
    MLL(n).R=zeros(2^nmax);
    MLH(n).R=zeros(2^nmax);
    MHL(n).R=zeros(2^nmax);
    MHH(n).R=zeros(2^nmax);
  end
  if n==nmax
    MLL(n).R(1:2:end,1:2:end)=MLL(n).F;
  else
    MLL(n).F=MLL(n+1).R-MHL(n+1).R-MLH(n+1).R+MHH(n+1).R;
    MLL(n).R(1:2:end,1:2:end)=MLL(n+1).R + MHL(n+1).R + MLH(n+1).R
+ MHH(n+1).R;
  end
  MLL(n).R=conv2(MLL(n).R,FiltroLL*4);
  MLL(n).R=MLL(n).R(1:end-1,1:end-1);

  MHL(n).R(1:2:end,1:2:end)=MHL(n).F;
  MHL(n).R=conv2(MHL(n).R,fliplr(FiltroHL));
  MHL(n).R=MHL(n).R(1:end-1,1:end-1);

  MLH(n).R(1:2:end,1:2:end)=MLH(n).F;
  MLH(n).R=conv2(MLH(n).R,flipud(FiltroLH));
  MLH(n).R=MLH(n).R(1:end-1,1:end-1);

  MHH(n).R(1:2:end,1:2:end)=MHH(n).F;
  MHH(n).R=conv2(MHH(n).R,FiltroHH);
  MHH(n).R=MHH(n).R(1:end-1,1:end-1);
end

load Images\figura.mat;
F=rgb2gray(F);
F=double(F);
MinF=min(min(F)); % Menor valor de F
MaxF=max(max(F)); % Maior valor de F
```

```
figure; imagesc(uint8(abs(F))); colormap('gray'); truesize;

RImg=MLL(1).R+MHL(1).R+MLH(1).R+MHH(1).R;
RImg=255-RImg;
MinR=min(min(RImg)); % Menor valor de RImg
MaxR=max(max(RImg)); % Maior valor de RImg
RImg=MinF+(RImg-MinR)*(MaxF-MinF)/(MaxR-MinR); % RImg na mesma
escala de F
figure; imagesc(uint8(abs(reconImg))); colormap('gray'); truesize;

PSNR=10*log10((255.^2)/(mean(mean((RImg-F).^2))))
```