IMAGEADOR CMOS UTILIZANDO TECNOLOGIA DE 0.18 um PARA CAPTURA E COMPRESSÃO DE IMAGENS NO PLANO FOCAL

Fernanda Duarte Vilela Reis de Oliveira

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
Dezembro de 2013

IMAGEADOR CMOS UTILIZANDO TECNOLOGIA DE 0.18 um PARA CAPTURA E COMPRESSÃO DE IMAGENS NO PLANO FOCAL

Fernanda Duarte Vilela Reis de Oliveira

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:

José Gabriel Rodríguez Carneiro Gomes, Ph.D.

Antonio Petraglia, Ph.D.

Davies William de Lima Monteiro, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2013

Oliveira, Fernanda Duarte Vilela Reis de

Imageador CMOS utilizando Tecnologia de 0.18 um para Captura e Compressão de Imagens no Plano Focal/Fernanda Duarte Vilela Reis de Oliveira. – Rio de Janeiro: UFRJ/COPPE, 2013.

XV, 77 p.: il.; 29, 7cm.

Orientador: José Gabriel Rodríguez Carneiro Gomes

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

Referências Bibliográficas: p. 74 – 76.

1. Imageadores CMOS. 2. Compressão de Imagens. 3. VQ. 4. DPCM. I. Gomes, José Gabriel Rodríguez Carneiro. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha família.

Agradecimentos

Ao meu orientador, José Gabriel Rodríguez Carneiro Gomes, por todos os conhecimentos que me foram passados, pelo suporte, incentivo, paciência e exemplo. Agradeço também pela disponibilidade para tirar minhas dúvidas, pelas idéias, conselhos e o auxílio durante todo o mestrado.

Aos membros da banca examinadora, Antonio Petraglia e Davies William de Lima Monteiro, que avaliaram e sugeriram importantes melhorias ao trabalho.

Aos professores do Laboratório de Processamento Analógico e Digital de Sinais, em especial ao Professor Fernando Antônio Pinto Barúqui, por me auxiliar em diversas etapas do projeto e por possibilitar a fabricação do circuito integrado projetado nessa dissertação, estabelecendo o convênio com a IBM.

Aos professores do Programa de Engenharia Elétrica, como o Antônio Mesquita e o Antônio Carlos Moreirão pelo aprendizado nas disciplinas do mestrado.

Ao Hugo Haas, cuja dissertação de mestrado e projeto de graduação me ajudaram muito, e pelas inúmeras vezes em que prontamente respondeu às minhas dúvidas.

Aos amigos do Laboratório de Processamento Analógico e Digital de Sinais, principalmente ao Allan Bides, Fabián Olivera, Fabio Lacerda, Fellipe Falleiro, Genildo Nonato, Gustavo Campos, João Ferreira, Jorge De la Cruz, Manoel Perez, Pedro Riascos e Thiago Brito, pela ajuda e companheirismo durante todo o período do meu mestrado.

Aos amigos que me apoiaram, em especial àqueles que fizeram questão de assistir a minha apresentação: Gabriel Melgaço (obrigado também pelo incentivo, carinho e paciência nesses dois anos de mestrado), Alexandre Leizor, Barbara Bomfim, Diego Haddad, Diego Wanderley, Eduardo Anjos, Francinei Gomes, Felipe Clark, Gabriel Araújo, João Cialdino, José Roberto Motta, Professor Joarez Bastos, Leandro D'oliveira, Pedro Grojsgold, Ricardo Flach, Ricardo França, Thaís Maria, Tiago Bitarelli e Vitor Rosa.

À minha família, pelo incentivo, apoio e força, principalmente à minha mãe, Maria Cristina Duarte Vilela, e ao meu pai, *in memoriam*, Fernando Márcio Reis de Oliveira.

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 UTILIZANDO TECNOLOGIA DE 0.18 um PARA CAPTURA E COMPRESSÃO DE IMAGENS NO PLANO FOCAL

Fernanda Duarte Vilela Reis de Oliveira

Dezembro/2013

Orientador: José Gabriel Rodríguez Carneiro Gomes

Programa: Engenharia Elétrica

Utilizando os sensores de imagem CMOS, é possível acrescentar *hardware* para processamento de sinais no mesmo chip da matriz de pixels. Nas câmeras digitais convencionais, a compressão de imagens é feita utilizando um processador digital externo. A nossa abordagem propõe que essa compressão seja feita de forma analógica, dentro da matriz, eliminando a necessidade do *hardware* externo e realizando o processamento de forma paralela. Em 2010, foi fabricado um chip com tecnologia de 0.35 μm que realiza a captura e compressão da imagem utilizando VQ e DPCM. Diversos testes mostraram que algumas melhorias poderiam ser acrescentadas ao projeto do chip. O novo projeto, apresentado nessa dissertação, foi feito com tecnologia de 0.18 μm, o que possibilitou um aumento do *fill-factor*, e foram acrescentadas modificações para aumentar a qualidade da imagem comprimida. Dentre essas modificações, nós podemos destacar que: a faixa dinâmica do sensor foi ajustada para que a corrente máxima gerada pelo fotodiodo seja 20 pA, correspondendo a um pixel branco; a complexidade do VQ foi aumentada, pois aumentamos do número de dimensões para cinco e o número de bits do VQ para nove; espelhos de corrente simples foram substituídos por espelhos de corrente *cascode* em algumas partes do circuito; foi acrescentado um modelo não-linear ao decodificador. Essa dissertação apresenta os resultados de simulação do novo chip e o compara com o projeto anterior.

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

A 0.18 um CMOS IMAGER WITH FOCAL PLANE IMAGE COMPRESSION

Fernanda Duarte Vilela Reis de Oliveira

December/2013

Advisor: José Gabriel Rodríguez Carneiro Gomes

Department: Electrical Engineering

In CMOS image sensor designs, it is possible to include signal processing hardware into the same silicon area where the pixel matrix is located. In conventional digital cameras, image compression is usually carried out by an external digital signal processor. Our approach proposes analog-domain image compression, inside the pixel matrix, thus eliminating the need for external hardware and simultaneously allowing parallel processing. In 2010, we fabricated a CMOS imager based on 0.35 μm technology, which captures and compresses images using DPCM and VQ. Several tests suggested that significant improvements could be done on a newly designed imaging chip. The new design, which is presented in this dissertation, was done with 0.18 μm technology, which allows for a fill-factor improvement, and other changes were made in order to increase the compressed image quality. Among these changes, we point out that: the dynamic range was adjusted so that the maximum photocurrent was set to 20 pA, corresponding to an entirely white pixel; the VQ complexity was increased, both by increasing the number of input dimensions to five and the number of VQ bits to nine; single current mirrors were replaced by cascode current mirrors at specific circuit positions; and pixel non-linearity models are made available to the decoder. This dissertation presents simulation results obtained from the new chip and compares them with those obtained from the previous design.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

No final da década de 1990, a melhoria da qualidade da imagem gerada pelos sensores CMOS, que foi devida a algumas técnicas de projeto acrescentadas a esses sensores, como o fotodiodo grampeado, a introdução de microlentes e o CDS (*correlated double sampling*), permitiu que esses sensores se tornassem uma alternativa aos sensores CCD (*charge-coupled device*)[1]. A utilização dessa nova tecnologia trouxe diversas vantagens para a fabricação das câmeras digitais: seu custo de fabricação é mais baixo que o do CCD; o CMOS possui a disponibilidade de foundries para manufatura de protótipos; a tensão de alimentação é mais baixa, o que o torna melhor para aplicações que requerem baixo consumo; possui flexibilidade na leitura do pixel, isto é, qualquer pixel da matriz pode ser escolhido para leitura, sem que haja necessidade de ler nenhum outro pixel; e, principalmente, sua tecnologia de fabricação permite a integração de circuitos no mesmo chip que possui a matriz de pixels, desde circuitos simples de controle e leitura, até circuitos complexos que processam o sinal da matriz [1]-[3].

Uma câmera fotográfica que, além de tirar fotos, interpreta o sinal recebido e realiza alguma ação dependendo desse sinal, isto é, que processa esse sinal de alguma forma, é chamada de *smart camera* [4]. O advento dos sensores CMOS (*complementary metal-oxide semiconductor*) tornou esse tipo de câmera muito comum em estudos acadêmicos, devido à sua característica de permitir que outros circuitos sejam fabricados no mesmo *wafer* que a matriz de pixels. Ultimamente, muitos artigos acadêmicos que exploram essa característica dos imageadores CMOS foram publicados: em 2003, um imageador CMOS inspirado no funcionamento da retina humana foi apresentado [5]; a referência [6], de 2007, mostra um circuito integrado capaz de realizar a compressão da imagem capturada trabalhando em modo de tensão; a referência [7], de 2012, mostra um sensor de imagem com um conversor $\Sigma\Delta$ a nível do pixel; um artigo publicado em 2013, apresenta um sensor CMOS inspirado no sistema visual de insetos que é capaz de detectar movimentos [8]; outro artigo, que também foi publicado em 2013, apresenta um sensor de imagem que faz operações

ao nível do pixel para detectar eventos em uma cena [9]. Além desses, muitos outros artigos foram publicados nessa área, pois é muito interessante ter um sensor de imagem cuja saída já seja a informação desejada. Desta forma, não é necessário nenhum *hardware* de processamento externo, criando um sistema completo em um único chip. Se utilizarmos *hardware* de processamento analógico, temos outra vantagem: é possível acelerar o processamento do sinal, realizando as operações necessárias em paralelo. Realizando o processamento em modo de corrente, podemos garantir que a excursão de sinal não será limitada, mesmo para tecnologias CMOS que necessitam de baixa tensão de alimentação.

Um chip com tecnologia de 0.35 μm que realiza captura e compressão de imagens foi projetado, fabricado e testado recentemente. Os resultados dos testes, assim como uma explicação detalhada do projeto, podem ser vistos em [10]. Esse chip realiza a compressão da imagem de forma analógica, utilizando uma transformada linear, quantização vetorial (VQ) e uma modulação por código de pulsos diferenciais, mais conhecida através da expressão em inglês *differential pulse code modulation* (DPCM). Os principais conceitos teóricos dessas três técnicas serão mostrados nas Seções 2.1, 2.2 e 2.3. Esse chip foi fabricado com tecnologia AMS 0.35 Opto (*austriamicrosystems*, www.ams.com), possui uma matriz de 32 $\times$ 32 pixels, um *fill factor*, razão entre a área ativa do pixel e a área total do pixel, de 7%, sua tensão de alimentação é igual a 3.3 V e possui uma saída serial e comprimida. Para os testes desse chip, foi construído um aparato óptico com uma lente apropriada para o tamanho do chip e para a distância desejada ao alvo. Nesse chip, é necessária uma grande quantidade de luz para que obtenhamos uma imagem clara. Temos interesse em melhorar as seguintes características desse chip:

- Qualidade da imagem, utilizando a PSNR (*peak signal to noise ratio*, a ser definida na Seção 2) como medida de qualidade;
- *Fill factor*;
- Sensibilidade do sensor;
- Resolução da matriz de pixels.

1.1 Objetivos

Essa dissertação tem como objetivo apresentar o projeto de um novo circuito integrado que, assim como o projeto da referência [10], realiza a captura e compressão de uma imagem utilizando VQ e DPCM. As seguintes modificações serão acrescentadas:

- Utilizar cinco componentes da transformada linear, ao invés de quatro, e, assim, melhorar a qualidade da imagem;

- Realizar o projeto com tecnologia de 0.18 μm e melhorar o *fill factor*, aproveitando o fato de que essa tecnologia, que é mais moderna, permite a utilização de transistores menores e de distâncias menores entre os componentes. O aumento do *fill factor* também deve melhorar a qualidade da imagem;

- Utilizar espelhos de corrente *cascode* para a implementação da transformada linear. Esses espelhos de corrente são mais precisos e distorcem menos o sinal, o que também deve resultar em um aumento na qualidade da imagem;

- Aumentar a sensibilidade do sensor, considerando durante o projeto do circuito de leitura do fotodiodo que a sua capacitância vale 5 fF e a fotocorrente gerada por ele varia de 0 a 20 pA;

- Incluir uma saída sem compressão ao chip, para facilitar a avaliação do efeito que a compressão causa na imagem;

- Projetar um chip com resolução de 64 $\times$ 64 pixels.

De uma forma geral, as modificações têm como objetivo melhorar a qualidade da imagem. É importante lembrar, que a tecnologia de 0.18 μm tem a desvantagem de possuir uma menor eficiência quântica quando comparada a tecnologia de 0.35 μm, pois as junções da tecnologia mais nova são mais rasas. Uma eficiência mais baixa significa que uma mesma quantidade de luz irá gerar uma corrente menor no fotodiodo. Ao longo dos capítulos da dissertação essas modificações serão explicadas e avaliadas. Foi incluído ao chip um conversor analógico-digital de aproximações sucessivas que converte a tensão de saída do circuito de leitura em uma palavra binária de dez bits. Verificamos, nos testes do circuito anterior, que é muito importante uma saída sem compressão para que possamos avaliar melhor os resultados. O projeto desse conversor, no entanto, não será explicado nessa dissertação, pois ele foi desenvolvido e testado por outro aluno do Laboratório de Processamento Analógico e Digital de Sinais.

1.2 Estrutura do Texto

O Capítulo 2 explica os conceitos básicos sobre compressão de imagens utilizados para o projeto do circuito integrado. O capítulo está dividido nas três técnicas que são utilizadas para realizar a compressão (transformada linear, VQ e DPCM).

Os circuitos analógicos utilizados para realizar a compressão são descritos no Capítulo 3. Também são apresentadas, nesse capítulo, as simulações nominal e de Monte Carlo de cada circuito separado. Nesse mesmo capítulo, são mostrados os layouts dos circuitos e explicadas as técnicas de casamento que foram usadas.

O decodificador utilizado para reconstruir as imagens a partir dos bits gerados pelo chip é apresentado no Capítulo 4.

No Capítulo 5, são mostrados os resultados de simulação do circuito todo.

O Capítulo 6 apresenta as conclusões do projeto. Nesse capítulo, também é feita uma comparação entre o projeto atual e o anterior.

Capítulo 2

Compressão de Imagens

A compressão de dados surge da necessidade de se armazenar ou transmitir uma grande quantidade de informação de forma eficiente. Além das limitações que podem ser impostas pelo canal de transmissão dos dados, sem a compressão, diversos sistemas muito utilizados ultimamente não seriam possíveis, pois a quantidade de dados a serem enviados ou armazenados é muito grande. Um exemplo disso é que, atualmente, existem câmeras fotográficas digitais não profissionais, com matrizes de 16 megapixels. Para que não haja perdas, cada pixel deve ser representado com oito bits. Sem a compressão, seriam necessários 16 MB para armazenar uma única foto. No entanto, utilizando a compressão do padrão JPEG (*Joint Picture Experts Group*), é possível reduzir a quantidade de bytes para 2 MB, ou até menos, caso sejam acrescentadas mais perdas no codificador. Devido à grande importância da compressão, surgiram diversas técnicas ao longo das últimas décadas que variam de acordo com o compromisso entre complexidade, qualidade e taxa de compressão.

O JPEG surgiu como um padrão de compressão de imagens por volta de 1985 [11] e, devido à sua baixa complexidade e alta eficiência, é muito popular até hoje. A Figura 2.1 mostra um esquema simples do funcionamento de um codificador JPEG com perdas. Como pode ser visto na figura, a primeira etapa da compressão consiste em dividir a imagem em blocos. A divisão é feita para reduzir a complexidade do algoritmo durante o cálculo da transformada linear, no caso, a Transformada Cosseno Discreto (DCT). São utilizados blocos de 8×8 pixels. Após dividir a imagem em blocos, cada bloco irá passar por uma DCT. Como a DCT é baseada em cossenos, os resultados requerem representação em ponto flutuante. As primeiras perdas atribuídas a compressão JPEG são dadas pelo arredondamento dos coeficientes da transformada, uma vez que não é possível armazenar todas as casas decimais. Além disso, esse padrão de compressão tem como objetivo eliminar características da imagem que possuam pouca percepção pelo ser humano. Quanto menos perceptível ao olho humano for um determinado coeficiente da DCT, mais erro de quantização ele pode assumir.

Figura 2.1: Esquema de compressão do padrão JPEG.

Ao realizar o produto interno entre o bloco de 8×8 pixels e a matriz de transformada linear (DCT), realizamos uma decomposição no domínio da frequência. Os coeficientes de baixa frequência são agrupados no canto superior esquerdo do bloco resultante, sendo que a frequência dos coeficientes é aumentada gradativamente até o canto inferior direito. As frequências mais altas são aquelas em que o olho humano possui menos percepção. No caso de imagens, alta frequência quer dizer uma transição abrupta entre os valores dos pixels, em dois conjuntos de pixels.

Assumindo que cada coeficiente no domínio da frequência corresponde a uma variável aleatória, esta variável aleatória tem distribuição Laplaciana com um desvio-padrão específico. O coeficiente participa da construção do bloco original em uma proporção igual ao quadrado da razão entre o desvio-padrão do coeficiente e a soma de todos os desvios-padrão. Como a raiz quadrada da soma das variâncias no domínio da transformada equivale à energia do sinal também do domínio original, então o desvio-padrão do coeficiente representa a sua energia individual. De forma simplificada, dizemos que cada coeficiente encontrado representa uma fração de energia da imagem correspondente a uma determinada frequência. Sabe-se que em imagens naturais, coeficientes de alta frequência possuem menos energia do que os de baixa frequência. Sendo assim, para diminuir o número de bits por pixel, é mais interessante eliminar os coeficientes relativos às altas frequências. Após a DCT, os coeficientes são quantizados para serem representados por bits. Para o primeiro coeficiente, que equivale à média do bloco, no entanto, vamos considerar que existe correlação entre os blocos na imagem. A média do bloco anterior será utilizada como estimativa para a média do bloco atual, e a diferença entre as duas, chamada de erro, será quantizada.

O esquema utilizado para realizar a compressão analógica da imagem capturada dentro do chip é muito parecido com o esquema do JPEG. As diferenças ocorrem devido às dificuldades de implementação do algoritmo utilizando circuitos analógicos. A primeira diferença é que a compressão considera somente imagens em escala de cinza. Imagens coloridas exigiriam um esquema com filtros de cor que tornaria a implementação mais complicada. Assim como no caso do padrão JPEG, a imagem capturada pelo circuito integrado será dividida em blocos. No entanto, optamos por trabalhar com blocos de 4×4 pixels, para simplificar a implementação utilizando transistores. Para a transformada linear, vamos utilizar uma aproximação da DCT, que será explicada na Seção 2.1. A transformada completa para um bloco de 4×4 pixels irá gerar 16 coeficientes. No entanto, para diminuir a quantidade de *hardware*

necessário para a implementação do algoritmo, somente as cinco componentes de maior energia do sinal serão calculadas. O resultado a ser quantizado é um vetor com cinco elementos. Ao invés de quantizamos cada elemento separadamente, é mais interessante quantizar o vetor como um todo, pois isso permite que o vetor seja melhor representado. A quantização vetorial utilizada é mostrada na Seção 2.2. A Seção 2.3 explica a codificação da componente média, feita por DPCM, sendo que a diferença entre a média de um bloco e um valor estimado para essa média será o valor quantizado e transmitido.

Apesar do esquema utilizado para compressão ser parecido com o JPEG, não é possível comparar o resultado de compressão do chip com uma compressão JPEG devido às modificações que foram feitas para permitir a implementação analógica. Essas modificações, assim como os erros causados pelo hardware, serão responsáveis por uma queda na qualidade da imagem que não ocorre no JPEG. O esquema do JPEG foi apresentado somente com o intuito de mostrar que o método de compressão proposto utiliza um esquema clássico baseado em dividir a imagem em blocos e fazer uma quantização após uma transformação linear. Apesar da queda de qualidade gerada pela implementação analógica, esse tipo de processamento tem as vantagens de ser mais rápido, pois o cálculo dos bits que representam cada bloco é feito em paralelo, e de criar um sistema completo dentro de um único chip, sem a necessidade de memória ou processador externo.

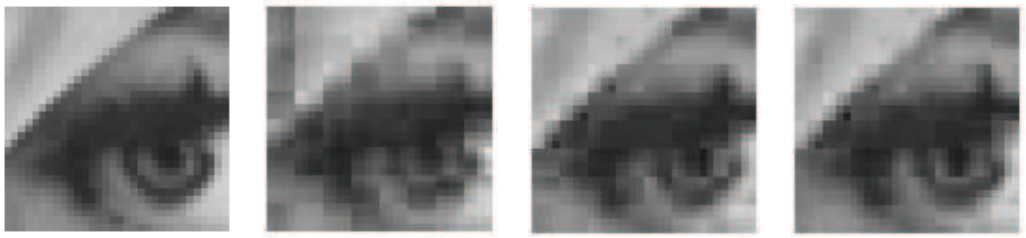
Do ponto de vista do codificador, o chip novo, apresentado nessa dissertação, utiliza as mesmas técnicas de compressão do chip de 0.35 μm, mas será projetado com uma tecnologia mais moderna, de 0.18 μm, e, com o objetivo de melhorar a qualidade da imagem comprimida, foram feitas algumas modificações ao algoritmo do projeto anterior: agora serão consideradas cinco componentes, ao invés de quatro; e o valor estimado para a média do primeiro bloco do DPCM, de cada linha de blocos, é próximo do valor médio que a média pode atingir, ao invés de zero, que era o caso do projeto anterior. A nível de software, a última modificação é muito simples. Basta trocar o valor de uma variável. A nível de *hardware*, no entanto, será necessária uma nova corrente de referência, o que requer o projeto de um circuito preciso, responsável por gerar essa corrente, e uma maior área externa à matriz de pixels, para posicionar esse circuito. O projeto será explicado na Seção 3.1.1. A primeira modificação, no entanto, irá afetar tanto *hardware* quanto software. Com cinco componentes, é necessário projetar um novo VQ e assim teremos um novo dicionário. Mesmo que nenhum bit seja acrescentado ao projeto do quantizador, no mínimo um bit será incluído, o bit de sinal, o que resulta em um aumento da taxa.

A PSNR, definida pela Equação (2.1) [12], foi a medida utilizada para avaliar a qualidade de uma imagem. O MSE (*mean square error*), é mostrado na Equação (2.2), onde $\mathbf{x}$ é uma imagem original com M $\times$ N pixels, e $\hat{\mathbf{x}}$ é a imagem comprimida.

$$\text{PSNR} = 10 \log_{10} 1/\text{MSE}. \tag{2.1}$$

$$\text{MSE} = \frac{1}{MN} \sum_{i}^{N} \sum_{j}^{M} ||x(i,j) - \hat{x}(i,j)||^2. \tag{2.2}$$

Se considerássemos que o VQ e o DPCM possuíssem um número de bits infinito, isto é, se não existisse erro de quantização, com o descarte de onze componentes AC que era feito no projeto anterior, a PSNR da compressão da Figura 2.2(a) ficaria limitada a 29.9 dB. Se forem descartadas dez componentes, ao invés de onze, esse limite sobe para 35.5 dB. Para avaliar se as modificações feitas no projeto valeriam o custo em complexidade e taxa, a imagem da Figura 2.2(a) foi comprimida utilizando três diferentes métodos. O cálculo da PSNR foi feito para cada um dos métodos: projeto anterior, com quatro componentes e VQ com sete bits; projeto novo, com cinco componentes e VQ com sete bits; e projeto novo, com cinco componentes e VQ com nove bits. Os resultados podem ser vistos nas Figuras 2.2(b), (c) e (d), respectivamente. No caso das Figuras 2.2(c) e (d), a média estimada para o primeiro bloco de cada linha vale aproximadamente 0.5. Na Figura 2.2(b), a média estimada vale zero.



(a) (b) (c) (d)

Figura 2.2: Simulação realizada no MATLAB utilizando um pedaço de 32 × 32 pixels da imagem Lena: (a) original, (b) compressão utilizando quatro componentes e VQ com sete bits, (c) compressão utilizando quatro componentes e VQ com sete bits e (d) compressão utilizando cinco componentes e VQ com nove bits.

A PSNR da Figura 2.2(b) é igual a 23.2 dB e a taxa é igual a 0.94 bpp (bits por pixel). A Figura 2.2(c) possui uma PSNR de 28.8 dB e taxa de 1.00 bpp. Para a Figura 2.2(d), a PSNR é igual a 29.9 dB e a taxa é igual a 1.13 bpp. Devido ao aumento significativo da PSNR, e a melhora subjetiva na qualidade da imagem, optamos por utilizar o terceiro método como base para o projeto do circuito integrado. Sendo assim, serão considerados cinco componentes da transformada linear e o VQ irá utilizar nove bits.

2.1 Análise de Componentes Principais

A Análise de Componentes Principais (PCA) é uma técnica muito usada em compressão de imagens. Essa técnica realiza uma transformada linear e tem como objetivo descorrelacionar os dados. A transformada linear utilizada no projeto é descrita na referência [13]. Essa transformada foi escolhida porque ela é baseada na DCT e possui somente multiplicadores inteiros. Para a implementação utilizando *hardware* analógico, é muito importante uma matriz de transformação com multiplicadores inteiros, pois ela pode ser projetada utilizando espelhos de corrente cujos transistores possuem largura e comprimento fixos e proporcionais ou inversamente proporcionais aos multiplicadores da transformada. O projeto desse circuito será explicado na Capítulo 3. No caso do nosso projeto, a transformada é aplicada a blocos de 4×4 pixels e somente as cinco componentes de maior energia do sinal serão utilizadas. Os valores do bloco de pixels serão transformados em um vetor coluna $\mathbf{y}(n)$ com 16 posições. Esse vetor será multiplicado pela matriz apresentada na Equação (2.3). O resultado é o vetor coluna $\mathbf{p}(n)$ que contém as 5 componentes mais energéticas do sinal: componentes 2, 5, 3, 9 e 6, guardadas nas posições um a cinco de $\mathbf{p}(n)$.

$$\mathbf{H} = \begin{bmatrix} 2 & 1 & -1 & -2 & 2 & 1 & -1 & -2 & 2 & 1 & -1 & -2 & 2 & 1 & -1 & -2 \\ 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -2 & -2 & -2 & -2 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 4 & 2 & -2 & -4 & 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 & -4 & -2 & 2 & 4 \end{bmatrix} \tag{2.3}$$

Ao realizar uma aproximação da DCT de forma a obter a matriz da Equação (2.3), obtemos uma matriz cujas linhas não possuem norma unitária. Além disso, as componentes 2 e 5 são componentes mais energéticas que as demais. Por esses dois motivos, é necessário aplicar fatores de escala às componentes do vetor $\mathbf{p}(n)$. Caso esses fatores não sejam aplicados, os desvios-padrão de cada componente, considerando um conjunto de 21 imagens de projeto, são muito diferentes entre si, o que prejudica o projeto do VQ. Para encontrar os fatores necessários que tornam os desvios-padrão aproximadamente iguais, aplicamos a transformada tal como mostrada na Equação (2.3), isto é, sem normalização, às 21 imagens e, em seguida, calculamos o desvio padrão para cada componente. O vetor de desvios-padrão encontrado é igual a [0.5997 0.6731 0.1595 0.1887 0.4343], onde a primeira posição é o desvio-padrão da segunda componente, a segunda da quinta componente, a terceira da terceira, a quarta da nona e a quinta da sexta. Para que haja uma boa distribuição, escolhemos, arbitrariamente, que os fatores de escala das componentes três e nove sejam iguais a 2. Os demais são calculados a partir desses: o primeiro fator é dado por $2 \cdot 0.5997/0.1595 \approx 8$, o segundo por $2 \cdot 0.6731/0.1595 \approx 8$, e assim por diante. Os fatores de escala resultantes são [8 8 2 2 5], para as componentes dois, cinco, três, nove e seis, respectivamente. O VQ será aplicada ao vetor $\mathbf{p}(n)$ após a

multiplicação por esses fatores de escala.

Ao final da decodificação, é importante lembrar de retirar esses fatores de escala, dividindo cada valor de $\hat{\mathbf{p}}(n)$ por seu respectivo fator. Além disso, como as linhas de $\mathbf{H}$ não possuem norma unitária, não basta multiplicar a transposta de $\mathbf{H}$ por $\hat{\mathbf{p}}(n)$ para encontrar $\hat{\mathbf{y}}(n)$. É necessário multiplicar $\mathbf{H}^T$ pela inversa de $\mathbf{H} \cdot \mathbf{H}^T$ e só então multiplicar o resultado por $\hat{\mathbf{p}}(n)$, obtendo assim a reconstrução $\hat{\mathbf{y}}(n)$.

Os valores das componentes do vetor $\mathbf{p}(n)$ possuem igual probabilidade de serem positivos ou negativos. Por esse motivo, cinco bits são utilizados para representar os sinais das cinco componentes. O vetor $\mathbf{x}(n)$ tem suas componentes iguais aos valores absolutos das componentes de $\mathbf{p}(n)$. Ele será enviado para o VQ, onde será representado através de nove bits.

2.2 Quantização Vetorial

Um quantizador é um sistema responsável por mapear um sinal de entrada, normalmente analógico, em um conjunto finito com valores conhecidos. Sendo Q o quantizador, ele será definido como o mapeamento $Q : \Re \rightarrow C$, onde $\Re$ é a reta real e C é o dicionário, que divide a reta real em células: $C \equiv \{y_1, y_2, ..., y_N\} \subset \Re$ [14]. O tamanho do dicionário, N, que é igual ao número de células, define a mínima taxa de bits necessária para representar todos os valores presentes em C. Os valores escalares y_n são os possíveis valores de saída do quantizador. Para um determinado sinal de entrada x_n, será escolhido o y_n mais próximo, que melhor representa x_n. Esses valores de saída, ou valores de reconstrução, são definidos segundo um conjunto de dados de projeto, e esse conjunto deve ser escolhido como um bom representante das possíveis entradas do quantizador. Dado um tamanho fixo, é possível definir um dicionário que gere a menor distorção possível considerando as estatísticas do conjunto de projeto. A distorção é definida usualmente como:

$$D = \frac{1}{N} \sum_{n=1}^{N} \|x_n - y_n\|^2 . \tag{2.4}$$

Caso o número de células não seja definido, é possível encontrar C otimizando J na equação:

$$J = D + \lambda H, \tag{2.5}$$

onde D é a distorção e H é a entropia, definida a partir da probabilidade de cada célula do dicionário:

$$H = \sum_{k=1}^{K} p_k \log_2 \left(\frac{1}{p_k} \right). \tag{2.6}$$

A entropia é um limite inferior para a taxa (de bits) associada à representação binária de y_n. A Equação (2.5) define um compromisso entre taxa e distorção. Quando desejamos transformar um sinal de analógico para digital, é necessário definir um quantizador. O VQ é uma generalização do quantizador escalar. O nosso interesse em utilizar um VQ se dá devido ao fato de que é mais interessante, no sentido da minimização conjunta de D e H, realizar o mapeamento de um vetor em um dicionário de vetores, do que quantizar cada dimensão separadamente. No caso, desejamos mapear vetores de cinco dimensões $\mathbf{x}(n)$.

Para reduzir a complexidade do VQ de modo a possibilitar implementação com *hardware* analógico, foi feito um projeto utilizando partições lineares. Inicialmente, encontramos o vetor $\mathbf{f}(n)$ dado por: $\mathbf{f}(n) = \mathbf{W}\mathbf{x}(n)$, onde $\mathbf{W}$ é a matriz de transformada linear mostrada na Equação (2.7). A matriz $\mathbf{W}$ é um arredondamento da transformada linear ótima, chamada de Transformada de Karhunen-Loève (KLT). Essa transformada foi calculada considerando todos os vetores $\mathbf{x}(n)$ do conjunto de 21 imagens utilizadas para projeto. A transformada ótima é aquela que consegue mínima correlação e máxima concentração de energia entre as componentes resultantes. O arredondamento foi feito para simplificar o projeto do circuito analógico.

$$\mathbf{W} = \begin{bmatrix} 0.25 & 0.5 & 0.25 & 0.5 & 0.5 \\ 0.5 & -0.25 & 0.5 & -0.5 & 0.25 \\ -0.5 & -0.5 & 0 & 0 & 0.75 \\ 0.25 & 0.5 & -0.5 & -0.5 & 0.5 \\ 0.75 & -0.5 & -0.5 & 0.25 & 0 \end{bmatrix} \tag{2.7}$$

Cada linha do vetor-coluna $\mathbf{f}(n)$ será quantizada por um quantizador escalar diferente. Como são utilizados nove bits para quantização, são criadas até 512 células possíveis, logo temos até 512 centroides. No entanto, 115 posições não são utilizadas. O dicionário do VQ possui cinco linhas e 512 colunas, mas 115 colunas possuem valor zero. Os nove bits gerados pela compressão representam uma coluna desse dicionário. Ao escolher uma coluna, estamos reconstruindo o vetor $\hat{\mathbf{x}}(n)$ para cada bloco de 4×4 pixels. Apesar do vetor $\mathbf{f}(n)$ ser utilizado na codificação, ele não é usado na decodificação, pois o dicionário do VQ contém diretamente os possíveis valores dos módulos das componentes. Os detalhes da decodificação serão explicados no Capítulo 4, mas é importante enfatizar que a operação inversa do VQ é dada por uma consulta em um dicionário, e não pelo inverso do quantizador escalar seguido da multiplicação pela inversa de $\mathbf{W}$.

O algoritmo utilizado para realizar o projeto é o mesmo utilizado para o projeto de um ECVQ (*entropy-constrained vector quantizer*), onde é feita a otimização da função de custo Lagrangeana mostrada na Equação (2.5). No nosso caso, a função foi otimizada para λ igual a 3.8×10^{-4}. A única diferença entre o projeto de um ECVQ comum e o projeto do nosso ECVQ com partições lineares aparece na condição da partição, onde, ao invés de calcularmos partições utilizando diagramas de Voronoi, modificamos a partição movendo retas ao longo de eixos definidos pela transformada linear $\mathbf{W}$.

Para cada dimensão de $\mathbf{f}(n)$ é definido um quantizador escalar diferente. Para a primeira dimensão, são definidos oito intervalos, para a segunda, quatro intervalos, para a terceira, quatro intervalos, para a quarta, dois intervalos e para a quinta, dois intervalos. Os limiares de cada quantizador serão implementados em modo de corrente e serão utilizados em comparadores. Para que isso seja possível, esses limiares foram arredondados a partir dos valores ideais obtidos por otimização. O arredondamento desses limiares possui pouca influência no resultado final: se não houvesse o arredondamento, a compressão da imagem da Figura 2.2 teria PSNR igual a 30.0 dB, ao invés de 29.8 dB. Os valores das correntes que representam os limiares, em microampères são:

$$\mathbf{T}_{\mathrm{VQ1}} = \begin{bmatrix} 0.5 & 1.0 & 1.75 & 3.0 & 4.5 & 6.0 & 9.0 \end{bmatrix} \tag{2.8}$$

$$\mathbf{T}_{\mathrm{VQ2}} = \begin{bmatrix} -0.25 & 0.75 & 2.5 \end{bmatrix} \tag{2.9}$$

$$\mathbf{T}_{\mathrm{VQ3}} = \begin{bmatrix} -1.5 & -0.25 & 0.75 \end{bmatrix} \tag{2.10}$$

$$\mathbf{T}_{\mathrm{VQ4}} = \begin{bmatrix} 0.25 \end{bmatrix} \tag{2.11}$$

$$\mathbf{T}_{\mathrm{VQ5}} = \begin{bmatrix} 0 \end{bmatrix} \tag{2.12}$$

Após as comparações com os 15 limiares, temos 15 bits resultantes. Esses códigos de 15 bits serão mapeados em códigos Gray com 3, 2, 2, 1 e 1 bits, utilizando circuitos lógicos baseados em portas XNOR. O resultado são os nove bits do VQ.

2.3 Codificação da Componente Média

Assim como na compressão JPEG, no caso da componente média do bloco, vamos nos aproveitar do fato de que, em uma imagem natural, existe grande correlação entre os pixels e a existência de transições abruptas de luminância é improvável.

Isto é, em geral, o valor de um pixel é muito parecido com os valores dos pixels adjacentes a ele. Existe uma redundância espacial na imagem. O mesmo conceito pode ser estendido para os blocos de 4×4 pixels. Sendo assim, se fizermos a diferença entre a média de um bloco e o bloco seguinte, existe grande probabilidade do resultado ser próximo de zero.

O DPCM é uma técnica de processamento de sinais muito utilizada quando sabemos que existe grande correlação entre o sinal atual e uma amostra adjacente [15]. Assim, o valor da amostra adjacente pode ser usado como previsão para o sinal atual. Nessa técnica, o valor quantizado será a diferença entre o sinal e o valor previsto do sinal. No nosso caso, o valor previsto da média de um bloco é igual à média do bloco anterior. Essa técnica será utilizada para cada linha de blocos. O DPCM é definido pelas seguintes equações:

$$e[n] = m[n] - \hat{m}[n] \tag{2.13}$$

$$\hat{e}[n] = \mathrm{SQ}\{e[n]\} \tag{2.14}$$

$$\hat{m}[n] = \hat{m}[n-1] + \hat{e}[n-1], \tag{2.15}$$

onde, $m[n]$ é a média do bloco n e $\hat{m}[n]$ é o valor previsto para essa média. A média do bloco pode variar de 0 a 1. Como foi explicado, o valor previsto é encontrado a partir da média do bloco anterior. No entanto, se não considerarmos que existe um erro devido à quantização, esse erro será propagado ao longo de toda a linha de blocos. Sendo assim, o valor previsto do bloco atual é calculado pela soma do erro quantizado do bloco anterior com o valor previsto desse mesmo bloco. No caso da média do primeiro bloco, o valor estimado é definido como o centroide do quantizador escalar mais próximo do valor médio das médias dos blocos, igual a 0.4688. No projeto anterior, o valor de referência é igual a zero. Nesse, se a média do primeiro bloco for próxima do valor máximo, o erro de quantização entre e e $\hat{e}$ será próximo de 0.5, pois o maior valor que $\hat{e}$ pode atingir é 0.4688. Dessa forma, o DPCM demora a se estabilizar e o erro gerado pela quantização na média do primeiro bloco irá reduzir a PSNR. Utilizando o valor 0.4688 como referência para o primeiro bloco evitamos que isso aconteça. Com isso, o erro calculado para o primeiro bloco de cada linha é dado por:

$$e[1] = m[1] - 0.4688. \tag{2.16}$$

Uma vez calculado o primeiro erro, ele deve ser quantizado. O valor previsto do segundo bloco ($\hat{m}[2]$) será dado por esse erro quantizado somado a 0.4688. Os erros

dos blocos seguintes serão calculados de acordo com as Equações (2.13), (2.14) e (2.15).

Ao quantizarmos a diferença entre a média de um bloco e do bloco anterior, é interessante utilizar um bit para o sinal, pois é equiprovável que a diferença seja positiva ou negativa. O módulo da diferença pode ser quantizado utilizando um quantizador escalar não-linear, com mais limiares próximos do zero, uma vez que existe uma maior probabilidade da diferença ser próxima de zero. No caso do chip projetado, são utilizados sete limiares. Três bits são necessários para indicar em qual intervalo do quantizador escalar o módulo da diferença entre as médias está posicionado.

O quantizador escalar do DPCM é calculado utilizando um conjunto de 21 imagens de treino e o algoritmo de Lloyd. Nesse algoritmo, os centroides iniciais são escolhidos de forma aleatória. Para esses centroides, é definida a partição: a melhor forma de dividir as células é aquela que minimiza o erro médio quadrático. Isso é feito escolhendo cada limiar como sendo o valor médio entre dois centroides. Para cada partição, um novo conjunto de centroides é calculado: cada centroide é o valor que melhor representa o seu intervalo, isto é, o centro de massa do intervalo. As condições do centroide e da partição são repetidas até que haja uma relativa estabilização nas posições dos centroides, terminando com o cálculo da condição do centroide. O quantizador escalar projetado para o DPCM do chip possui os centroides mostrados na Equação (2.17).

$$\mathbf{C}_{\mathrm{DPCM}} = \begin{bmatrix} 0.0063 & 0.0250 & 0.0563 & 0.1000 & 0.1500 & 0.2250 & 0.3250 & 0.4688 \end{bmatrix} \quad (2.17)$$

Como foi explicado, esses centroides são usados para representar, de forma aproximada, o erro $e[n]$. Como existe uma alta probabilidade de $e[n]$ ser próximo de zero, os centroides ficam mais concentrados perto do zero. O dicionário mostrado na Equação (2.17) foi encontrado considerando que o valor da média do bloco varia de forma adimensional de 0 a 1. No entanto, no circuito integrado, a saída de cada pixel varia de 0 μA a 10 μA. Para simplificar a implementação e garantir que o sinal não ficará muito pequeno e, assim, mais vulnerável a ruídos, não realizamos a divisão por 16 para calcular a média. Dividimos somente por 4. Como não estamos mais realizando a média, chamamos esse sinal de $s(n)$, ao invés de $m(n)$. Para a implementação prática, C_{DPCM} deve ser corrigido de forma que a faixa dinâmica se ajuste à do sinal. Isso é feito multiplicando os valores por 10 (devido à saída do circuito de leitura) e por 4 (pois não dividimos por 16, e sim por 4). Como serão necessárias correntes que implementem a diferença entre os centroides para o circuito de reconstrução de $\hat{e}(n)$, o que será explicado na Seção 3.5, é necessário

arredondar os valores dos centroides (multiplicados por 40) para até duas casas decimais, possibilitando a implementação prática. O resultado, em microampères, é mostrado na Equação (2.18). A corrente utilizada como referência para o primeiro bloco do DPCM ($\hat{s}(1)$) é aquela que equivale a 0.4688 na nova faixa dinâmica. Essa corrente é igual a 18.75 μA.

$$\mathbf{C}_{\text{DPCMpratica}} = \left[\begin{array}{cccccccc} 0.25 & 1.00 & 2.25 & 4.00 & 6.00 & 9.00 & 13.00 & 18.75 \end{array} \right] \tag{2.18}$$

Os limiares utilizados são apresentados na Equação (2.19). Os limiares originais também foram multiplicados por 40 e arredondados para encontrar os limiares mostrados na equação. Os valores dessa equação também serão implementados em modo de corrente, e estão apresentados em microampères. Como as mudanças nos centroides seriam pequenas, o cálculo dos centroides não foi refeito após o arredondamento dos limiares.

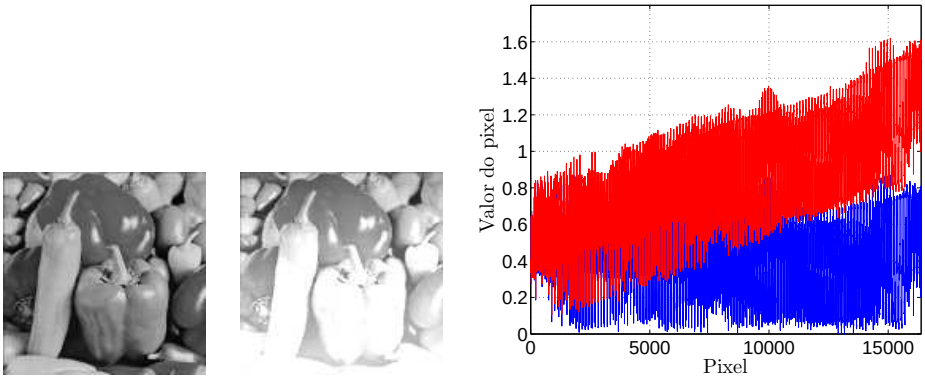
$$\mathbf{T} = \left[\begin{array}{ccccccc} 0.5 & 1.5 & 3.0 & 5.0 & 7.5 & 11.0 & 16.0 \end{array} \right] \tag{2.19}$$

Capítulo 3

Projeto do Circuito Integrado

Para realizar a compressão da imagem dentro da matriz de pixels, são utilizados circuitos em modo de corrente responsáveis por implementar as operações necessárias para que a imagem capturada seja comprimida de forma analógica. As seções a seguir mostram os projetos dos circuitos utilizados no chip, desde a captura da imagem até a conversão analógico para digital.

Por conta dos erros de fabricação, nós sabemos que as células dos quantizadores não serão implementadadas como foram projetadas. A sensibilidade do algoritmo utilizado foi estudada no artigo [16]. É esperado que, mesmo com os erros de projeto, a queda final da PSNR seja baixa. Para cada circuito que será explicado, foi feita uma simulação de Monte Carlo, de forma que, se incluirmos as variações encontradas através das simulações no codificador ideal programado no MATLAB, podemos verificar, separadamente, a influência do erro de cada circuito na queda de PSNR. No caso do DPCM, em que um bloco depende do bloco anterior, os erros de circuito podem ser críticos. A Figura 3.1 mostra um exemplo de como um erro no centroide do DPCM pode afetar o resultado. O erro acrescentado foi gerado com uma função normal com média zero e desvio-padrão 0.0004. Como o erro é aleatório, a curva vermelha também poderia ter desviado da curva azul de forma negativa. No caso da Figura 3.1, o DPCM foi executado para todos os pixels, utilizando o último pixel de uma linha para prever o primeiro pixel da linha seguinte. Com essa abordagem, o erro é acumulado de uma linha para a seguinte. Isso não ocorre no caso do projeto, que separa o DPCM por linhas. No nosso caso, esse tipo de erro poderia resultar em linhas mais escuras ou mais claras que outras. Para controlar esse possível acúmulo de erro por linhas, são acrescentados três circuitos contendo blocos cascateados de DPCM com entrada zero no final de cada linha de blocos da matriz de pixels. Como a entrada desses três circuitos é conhecida, nós podemos prever o desvio dos valores encontrados a partir dos bits desses três blocos, e acrescentar uma correção no algoritmo de decodificação. Nos resultados apresentados no Capítulo 5, esses bits adicionais não foram considerados, pois os erros do DPCM não foram significativos.



(a) (b) (c)

Figura 3.1: (a) Imagem Pepper reconstruída sem erro de DPCM, (b) imagem Pepper reconstruída com erro de DPCM e (c) gráfico com os valores dos pixels da imagem reconstruída sem erro, em azul e com erro, em vermelho.

Outra etapa importante do projeto de um circuito integrado é o *layout*, onde são representadas as camadas físicas do processo de fabricação dos dispositivos. A Seção 3.7 mostra o *layout* do imageador e explica a importância dessa etapa para o projeto.

3.1 Espelhos de Corrente

A Figura 3.2 mostra os dois tipos de espelho de corrente utilizados no circuito: espelho simples e *cascode*. Em geral, o espelho de corrente é utilizado com o objetivo de gerar uma corrente de referência para um circuito [17]. No nosso caso, além dele ser utilizado para gerar as correntes de referência que definem os limiares das quantizações, esse circuito também será necessário para implementar as matrizes de transformada linear **H** e **W**, e para fazer o somatório dos valores dos 16 pixels de um bloco para o DPCM. Além disso, como todo o processamento é feito em modo de corrente, o espelho de corrente também é utilizado dentro de vários circuitos, para fazer cópias de sinais, ou simplesmente para inverter o sentido da corrente, como no caso do circuito de leitura, mostrado na Seção 3.2, e do circuito de valor absoluto, que será explicado na Seção 3.3.

O espelho de corrente irá gerar uma corrente de saída que, idealmente, é diretamente proporcional à corrente de entrada. Isso acontece porque mantemos os transistores M_1 e M_2 na saturação e com V_{GS} iguais, tanto no espelho simples como no *cascode*. Para o espelho *cascode*, os transistores M_0 e M_3 também devem ser mantidos na saturação. Considerando o modelo de primeira ordem do transistor MOS, temos:

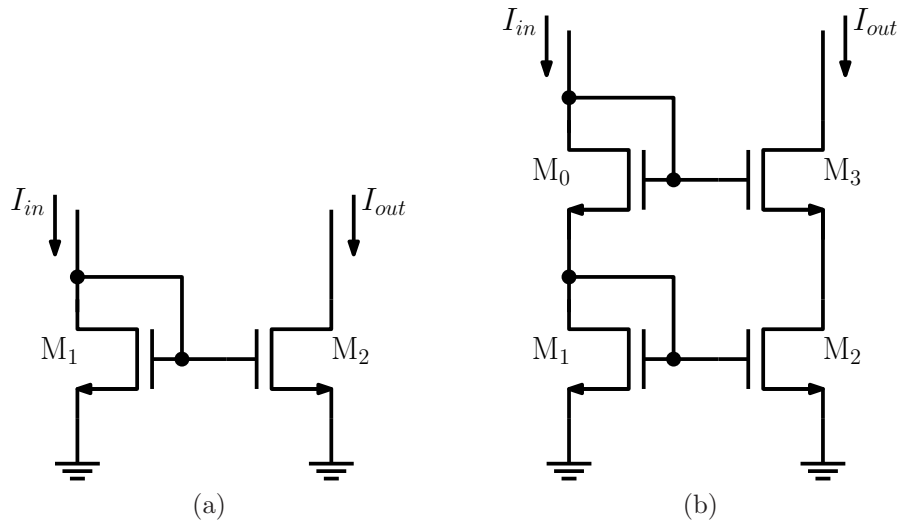


Figura 3.2: (a) Espelho de corrente simples e (b) espelho de corrente *cascode*.

$$I_{in} = K \cdot \frac{W_1}{L_1} \cdot (V_{GS} - V_{t1})^2 \Rightarrow (V_{GS} - V_{t1})^2 = \frac{I_{in}}{K \cdot W_1/L_1} \tag{3.1}$$

$$I_{out} = K \cdot \frac{W_2}{L_2} \cdot (V_{GS} - V_{t2})^2 \Rightarrow (V_{GS} - V_{t2})^2 = \frac{I_{out}}{K \cdot W_2/L_2} \tag{3.2}$$

Se consideramos que a tensão de *threshold* dos transistores é igual:

$$\frac{I_{out}}{K \cdot W_2/L_2} = \frac{I_{in}}{K \cdot W_1/L_1} \tag{3.3}$$

$$I_{out} = I_{in} \cdot \frac{W_2/L_2}{W_1/L_1} \tag{3.4}$$

Como podemos ver pela Equação (3.4), a corrente de saída será igual à corrente de entrada multiplicada pela razão entre os tamanhos dos transistores. No entanto, sabemos que o V_{DS} do transistor também influencia a corrente que passa pelo *drain*, por isso é importante tentar manter as tensões V_{DS} de M_1 e M_2 fixas e iguais, de forma a garantir a saturação. No caso do espelho de corrente simples, não será possível controlar o V_{DS}. Sendo assim, a transferência entre a corrente de saída e a corrente de entrada não será constante. O erro gerado pelo V_{DS} ocorre devido ao efeito da modulação de comprimento do canal, que é mais significativo quando os transistores possuem um comprimento de canal próximo do mínimo [18]. No nosso caso, como todo o circuito ficará dentro da matriz de pixels, desejamos que os transistores sejam pequenos de forma a minimizar a área ocupada pelo circuito. Assim, o erro na cópia da corrente, gerado devido à modulação de comprimento do canal, deixa o circuito impreciso.

Para tentar manter a tensão de V_{DS} dos transistores M_1 e M_2 fixas, e assim melhorar a precisão do circuito, podemos utilizar o espelho de corrente *cascode*. A desvantagem desse espelho é que a sua excursão de sinal é menor que a excursão do espelho simples, o que torna a sua polarização mais difícil [19]. Por esse motivo, não foi possível utilizar o espelho *cascode* em todo o circuito. Esse espelho foi utilizado para gerar as correntes de referência, a matriz **H** e o somatório dos valores dos pixels. Os espelhos da matriz **W** e os espelhos dos demais circuitos foram mantidos simples. No circuito integrado anterior, fabricado com tecnologia de 0.35 μm, o espelho *cascode* só foi utilizado para gerar as correntes de referência.

Analisando o esquemático do espelho *cascode*, encontramos a tensão do *drain* do transistor M_2 definida pela Equação (3.5), onde a tensão de *gate* do transistor M_3 pode ser substituída pela Equação (3.6). O resultado é que V_{D2} é função da tensão de *drain* do transistor M_1 e das tensões V_{GS} dos transistores M_0 e M_3, como pode ser visto na Equação (3.7).

$$V_{D2} = V_{G3} - V_{GS3}, \tag{3.5}$$

$$V_{G3} = V_{GS0} + V_{D1}, \tag{3.6}$$

$$V_{D2} = V_{GS0} + V_{D1} - V_{GS3}. \tag{3.7}$$

Se, através da escolha das dimensões dos transistores, garantirmos que $V_{GS0} = V_{GS3}$, temos:

$$V_{D2} = V_{D1}. \tag{3.8}$$

Incluindo o efeito da modulação de canal nos cálculos da corrente de saída do espelho de corrente, temos o resultado mostrado na Equação (3.9). Através dessa equação, percebemos que, garantindo que a Equação (3.8) seja cumprida, o efeito será cancelado.

$$I_{out} = I_{in} \cdot \frac{W_2/L_2}{W_1/L_1} \cdot \frac{1 + \lambda V_{DS2}}{1 + \lambda V_{DS1}} \tag{3.9}$$

3.1.1 Correntes de Referência

Para definir os níveis de quantização, tanto do DPCM, quanto do VQ, são necessárias diversas correntes de referência. Decidimos que só teríamos um pino de referência de 1 μA. As demais correntes são geradas através de cópias dessa, mudando somente a multiplicidade dos transistores para realizar multiplicações e divisões. Parte desse

circuito é mostrado na Figura 3.3. Como pode ser visto na figura, a corrente de referência deve ser injetada no circuito. Os dois transistores ligados à entrada, M_1 e M_2, possuem largura igual a 0.5 μm, comprimento igual a 5 μm e multiplicidade 4. Todos os transistores tipo N ligados aos *gates* de M_1 e M_2 possuem essa mesma largura e comprimento, assim como os transistores P que também formam espelhos *cascode*. Com todos os transistores do mesmo tamanho, o casamento entre eles se torna mais simples. O comprimento do canal foi escolhido de forma a diminuir as variações de descasamento, pois utilizando o modelo de Pelgrom [20], podemos concluir que é importante um comprimento grande para diminuir o efeito do descasamento.

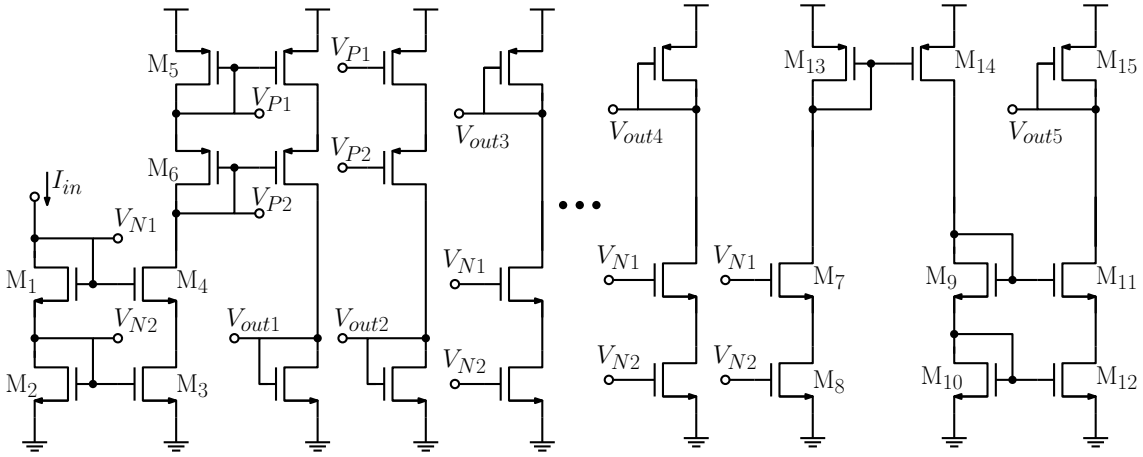


Figura 3.3: Circuito que gera as correntes de referência.

Podemos ver pela Figura 3.3 que, apesar da corrente ser gerada por espelhos *cascode*, por simplicidade, a ligação com o restante do circuito será feita através de espelhos simples. A partir da corrente de entrada de 1 μA, precisamos gerar correntes de 0.25 μA, 0.5 μA, 0.75 μA, 1 μA, 1.25 μA, 1.5 μA, 1.75 μA, 2.0 μA, 2.25 μA, 3.0 μA, 4.0 μA, 4.5 μA, 5.0 μA, 5.75 μA, 6.0 μA, 7.5 μA, 9.0 μA, 11.0 μA, 16.0 μA ligadas, na saída, a transistores P e correntes de 0.25 μA, 1.5 μA e 18.75 μA ligadas, na saída, a transistores N. Os transistores com *gates* ligados às saídas, exceto o ligado à saída de 18.75 μA, possuem largura igual a 1.5 μm, comprimento igual a 1 μm e multiplicidade 1.

No caso das correntes ligadas aos transistores de saída tipo N, inicialmente, invertemos o sentido da corrente de entrada com espelhos *cascode* tipo P ligados aos transistores M_3 e M_4, que possuem multiplicidade 4. Quando fazemos esse tipo de ligação, a polarização dos transistores N M_3 e M_4 fica bastante limitada. Por esse motivo, os transistores P ficaram com multiplicidades muito altas. Quanto maior a multiplicidade, menor a queda de tensão sobre M_5 e M_6, o que garante que M_3 seja mantido na saturação. Os transistores M_5 e M_6 ficaram com multiplicidade igual a 20. Para gerar uma corrente de 1.5 μA, precisamos que os trasistores ligados

ao *gate* de M_5 e M_6 tenham multiplicidade 30. Para gerar 0.25 μA, precisamos de multiplicidade 5. No entanto, para gerar a corrente de 18.75 μA, optamos por usar dois transistores tipo N em *cascode* com multiplicidade 15, ligados da mesma forma que M_3 e M_4 aos transistores M_1 e M_2. Dessa forma, realizamos uma multiplicação por 3.75. A saída desse espelho está ligada espelhos P simples, ao invés de *cascode*, para diminuir a área necessária para implementar o circuito. Os dois transistores desse espelho simples possuem largura 1.5 μm e comprimento 2 μm. O transistor da entrada possui multiplicidade 2 e o da saída multiplicidade 10, realizando outra multiplicação, agora por cinco. No total, multiplicamos 1 μA por 18.75, como desejado. Para conectar ao restante do circuito, a saída desse espelho P simples é conectada ao *gate* e *drain* de um transistor N com largura 1 μm, comprimento 2 μm e multiplicidade 2. O *source* desse transistor está ligado a terra.

Para as correntes ligadas aos transistores de saída tipo P, são feitas diversas cópias a partir das tensões geradas por M_1 e M_2 alterando somente a multiplicidade dos transistores ligados a M_1 e M_2. A multiplicidade varia de 1, para gerar 0.25 μA, até 44 para gerar 11.0 μA. Para gerar 16.0 μA, seria necessária uma multiplicidade igual a 64, por isso optamos por dividir a cópia em duas partes, utilizando espelhos simples. O esquema utilizado é mostrado na Figura 3.3 (transistores M_7 a M_{15}), onde V_{out5} seria a saída de referência para gerar 16 μA. A saída V_{out5} está ligada a um espelho N *cascode*, onde os transistores M_{11} e M_{12} possuem multiplicidade 16 e os transistores M_9 e M_{10} multiplicidade 4. Dessa forma, multiplicamos por 4. A estrada desse espelho está ligada a um espelho P simples (M_{13} e M_{14}), com comprimento 1.5 μm, largura 2 μm, e multiplicidade 20 tanto para o transistor da entrada (M_{13}) quanto da saída(M_{13}). Finalmente, a entrada desse espelho P simples está ligada a transistores tipo N com *gates* ligados a M_1 e M_2. Esses transitores (M_7 e M_8) possuem multiplicidade 16, de forma que temos outra multiplicação por 4. No total, multiplicamos a corrente de entrada, de 1 μA, por 16, gerando o resultado desejado.

A simulação transiente nominal desse circuito ao longo de 1μs levou aos resultados mostrados na Tabela 3.1. As correntes que aparecem negativas foram medidas no *drain* dos transistores P de saída, e as correntes que aparecem positivas foram medidas no *drain* dos transistores N de saída. Como podemos ver na tabela, os maiores erros são aqueles em que utilizamos os espelhos simples e realizamos duas etapas de espelhos de corrente para gerar as correntes, como é o caso da corrente de -16.0 μA, com erro de 0.23%, e o caso da corrente de 18.75 μA, com erro de 0.68%. Portanto, estes erros mais altos estão associados à necessidade do uso de espelhos de corrente em cascata. Ainda assim, são erros aceitáveis, inferiores a 1%.

A Figura 3.4 mostra os gráficos gerados pela simulação nominal. Para melhor visualização, as correntes estão separadas de acordo com a parte do circuito em que

Tabela 3.1: Correntes de referência.

Corrente desejada (μA)	Resultado de simulação (μA)	Erro relativo (%)
-16.0	-15.9638	0.2264
-11.0	-10.9994	0.0057
-9.0	-8.9997	0.0036
-7.5	-7.4999	0.0020
-6.0	-6.0000	0.0002
-5.75	-5.7500	0.0001
-5.0	-5.0001	0.0011
-4.5	-4.5001	0.0018
-4.0	-4.0001	0.0025
-3.0	-3.0001	0.0041
-2.5	-2.5001	0.0050
-2.0	-2.0001	0.0061
-1.75	-1.7501	0.0066
-1.5	-1.5001	0.0072
-1.25	-1.2501	0.0079
-1.0	-1.0001	0.0087
-0.75	-0.7501	0.0097
-0.5	-0.5001	0.0111
-0.25	-0.2500	0.0135
+0.25	+0.2500	0.0140
+1.5	+1.4998	0.0159
+18.75	+18.8781	0.6831

são utilizadas. Somente a corrente de 18.75 μA, que é utilizada como referência para o primeiro bloco de cada linha, foi multiplicada por -1 e colocada junto com as correntes do quantizador escalar do DPCM, também com o objetivo de melhorar a visualização.

A simulação de Monte Carlo do circuito extraído é mostrada na Figura 3.5. A Tabela 3.2 mostra os valores máximo e mínimo de cada corrente tanto para a simulação de Monte Carlo do circuito esquemático quanto para a do circuito extraído. Comparando os dois resultados, percebemos que o *layout*, que será bem explicado na Seção 3.7, consegue um bom casamento entre os transistores, pois os resultados são próximos. A tabela também mostra a variação total em relação ao valor ideal de cada corrente de referência devido ao processo de fabricação e ao descasamento entre os transistores. A variação máxima encontrada é de 12.4%, para a corrente de 0.25 μA, com saída em um transistor tipo P. No nosso caso, uma variação máxima de $\pm$6.2% é aceitável, pois acrescentando essa variação aos limiares do codificador do MATLAB, e rodando 1000 simulações, obtivemos uma PSNR média de 29.0 dB para a imagem da Figura 2.2(d), resultando em uma perda

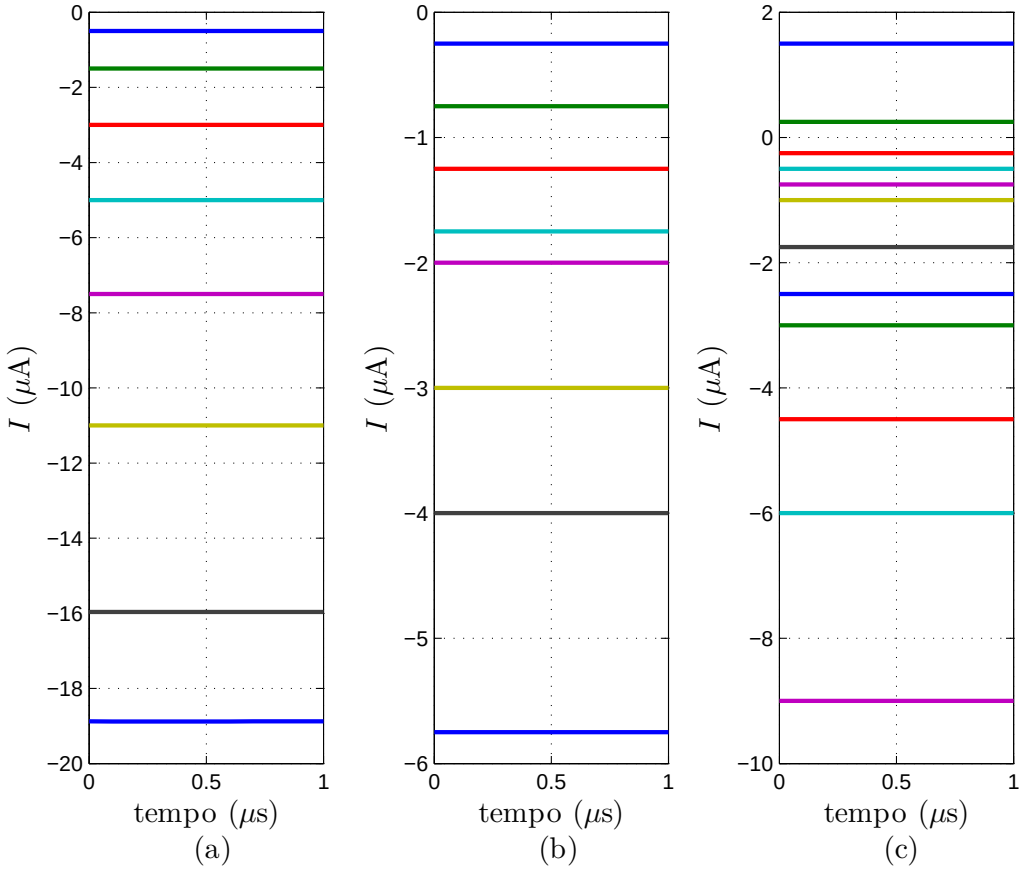


Figura 3.4: Simulação nominal das correntes de referência: (a) correntes utilizadas no quantizador escalar do DPCM e corrente de 18.75 μA, utilizada como referência para o primeiro bloco de uma linha de blocos, (b) correntes utilizadas no circuito de reconstrução do DPCM e (c) correntes utilizadas no VQ.

de 0.9 dB em relação ao resultado teórico.

O circuito que gera essas correntes de referência é grande, o *layout* desse circuito possui 86.5 μm $\times$ 428.7 μm, e é único para toda a matriz. Por esses motivos, é mais interessante que esse circuito fique fora da matriz, onde podemos inclusive garantir um melhor casamento entre os transistores.

3.1.2 Produto Interno

No chip anterior, fabricado com tecnologia de 0.35 μm, o circuito utilizado para implementar a matriz **H** foi feito com espelhos de corrente simples. Para o novo chip, foram feitos testes com o espelho simples e o *cascode* para avaliar as vantagens e desvantagens de utilizar o *cascode*. Como o *cascode* apresentou uma melhor linearidade, foi decidido que a matriz **H** seria implementada utilizando espelhos *cascode*. A implementação da matriz **H** é simples, pois os valores dessa matriz são fixos. Sendo assim, a multiplicação entre o vetor de valores dos pixels e a matriz **H** é um produto

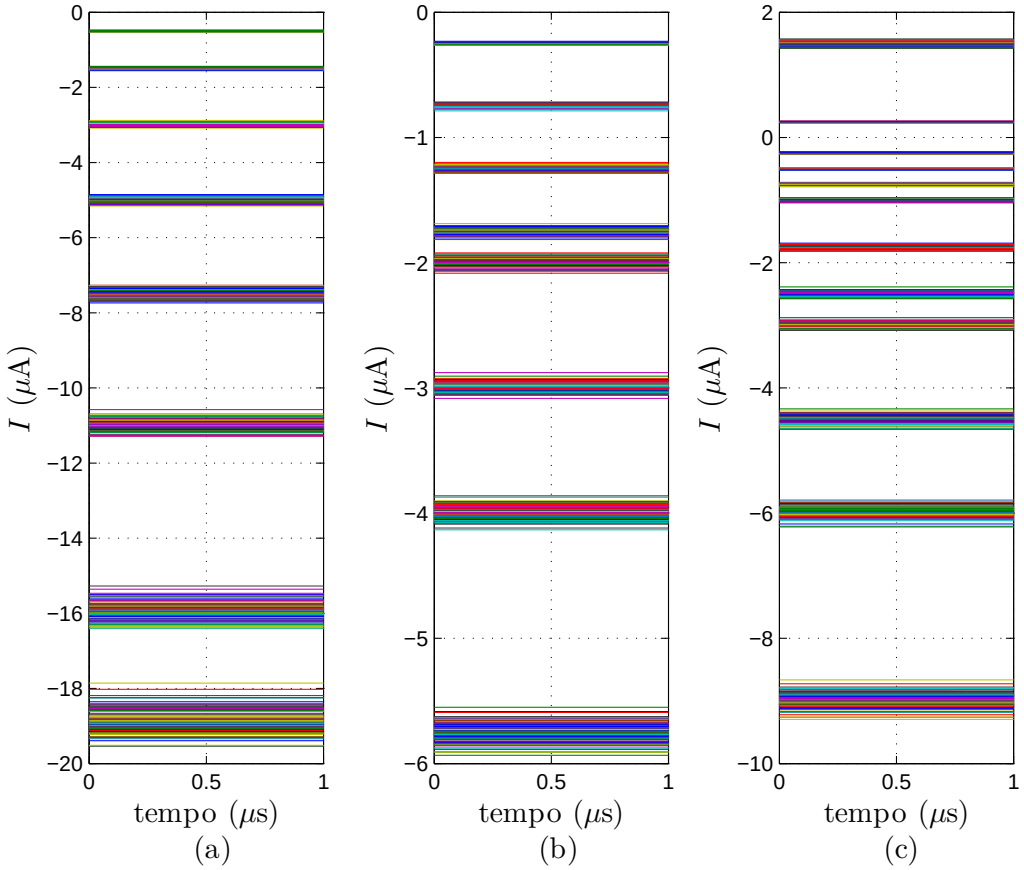


Figura 3.5: Simulação de Monte Carlo das correntes de referência: (a) correntes utilizadas no quantizador escalar do DPCM e corrente de 18.75 μA, utilizada como referência para o primeiro bloco de uma linha de blocos, (b) correntes utilizadas no circuito de reconstrução do DPCM e (c) correntes utilizadas no VQ.

interno com coeficientes fixos. A multiplicação será feita copiando as correntes dos pixels para espelhos com diferentes multiplicidade e, para realizar o somatório, basta conectar a saída de todos os espelhos em um mesmo nó.

Um exemplo de como é feita a implementação com *cascode* pode ser visto na Figura 3.6. Nessa figura, estamos fazendo duas multiplicações, uma positiva, com espelho N, e uma negativa, com espelho P. A saída do circuito de leitura do pixel, explicado na Seção 3.2, são duas tensões que devem ser ligadas a dois transistores tipo P, conectados em *cascode*, como M_1 e M_2, e M_3 e M_4. Os transistores M_1 e M_2 são utilizados para inverter o sentido da corrente para realizar multiplicações positivas com espelhos, como por exemplo o espelho formado pelos transistores M_5 a M_8. Como I_1 e I_2 são geradas a partir das mesmas entradas, isto é, do mesmo pixel, essa implementação seria para uma mesma coluna de $\mathbf{H}$. Trocando os espelhos *cascode* por espelhos simples, temos a implementação anterior. Nesse caso, a saída do circuito de leitura, que é ligada a esse circuito, deve ser uma única tensão, também gerada por um espelho simples.

Tabela 3.2: Simulação de Monte Carlo das correntes de referência. Valores máximo e mínimo de cada corrente obtida através dos circuitos esquemático e extraído, e variação total de cada corrente em relação ao valor ideal.

	Circuito Esquemático			Circuito Extraído		
Ideal (μA)	Máx. (μA)	Mín. (μA)	Var. (%)	Máx. (μA)	Mín. (μA)	Var. (%)
-16.0	-15.4521	-16.6942	7.7633	-15.2743	-16.4079	7.0847
-11.0	-10.7691	-11.4163	5.8838	-10.5768	-11.2869	6.4557
-9.0	-8.8119	-9.2637	5.0194	-8.6635	-9.2932	6.9957
-7.5	-7.3312	-7.8241	6.5714	-7.2689	-7.7364	6.2340
-6.0	-5.8514	-6.1951	5.7296	-5.7887	-6.2248	7.2677
-5.75	-5.6283	-5.9509	5.6098	-5.5507	-5.9338	6.6608
-5.0	-4.9031	-5.1854	5.6463	-4.8545	-5.1555	6.0204
-4.5	-4.3969	-4.6685	6.0364	-4.3355	-4.6602	7.2159
-4.0	-3.8935	-4.1361	6.0660	-3.8616	-4.1332	6.7888
-3.0	-2.9222	-3.1174	6.5077	-2.8782	-3.0837	6.8497
-2.5	-2.4315	-2.5829	6.0536	-2.3870	-2.5744	7.4994
-2.0	-1.9438	-2.0785	6.7370	-1.9222	-2.0837	8.0718
-1.75	-1.7004	-1.8106	6.2987	-1.6871	-1.8153	7.3228
-1.5	-1.4582	-1.5410	5.5185	-1.4433	-1.5616	7.8866
-1.25	-1.2042	-1.3038	7.9699	-1.1990	-1.2867	7.0181
-1.0	-0.9569	-1.0429	8.5958	-0.9602	-1.0424	8.2228
-0.75	-0.7211	-0.7856	8.5948	-0.7177	-0.7877	9.3280
-0.5	-0.4785	-0.5244	9.1896	-0.4805	-0.5218	8.2642
-0.25	-0.2369	-0.2675	12.2218	-0.2322	-0.2633	12.4498
+0.25	+0.2628	+0.2418	8.3807	+0.2604	+0.2333	10.8377
+1.5	+1.5677	+1.4496	7.8756	+1.5720	+1.4247	9.8167
+18.75	+19.7358	+18.1611	8.3987	+19.5377	+17.8597	8.9493

Os transistores da matriz **H** possuem largura igual a 1 μm e um comprimento igual a 0.5 μm, tendo sua multiplicidade variada de acordo com a multiplicação. Os transistores M_1 e M_2 possuem multiplicidade 2, o que resulta em uma divisão por 4 da corrente do pixel. Essa divisão sempre é feita para garantir que o sinal de saída não seja alto demais. Ela será considerada ao ajustar os fatores de escala que foram explicados na Seção 2.1. O ajuste desses fatores é feito nos circuitos para cálculo do valor absoluto de cada componente do vetor $\mathbf{p}(n)$. Também devemos considerar essa divisão ao escolher a multiplicidade de M_3 e M_4. Por exemplo, se desejamos implementar uma multiplicação por -1, então M_3 e M_4 possuem multiplicidade 2. Se desejamos implementar uma multiplicação por -2, M_3 e M_4 possuem multiplicidade 4. No caso das multiplicações positivas, a mesma regra é seguida, pois estabelecemos que M_5 e M_6 sempre possuem multiplicidade 2.

A corrente máxima de saída de cada pixel é 10 μA, como será explicado na Seção 3.2. Cada uma dessas correntes de saída será dividida por 4, resultando em um valor máximo de 2.5 μA na entrada de cada espelho que implementa a matriz **H**. Se o vetor **y**, que contém os valores dos pixels, for tal que todos os coeficientes positivos da primeira linha são multiplicados por 2.5 e todos os coeficientes negativos da primeira

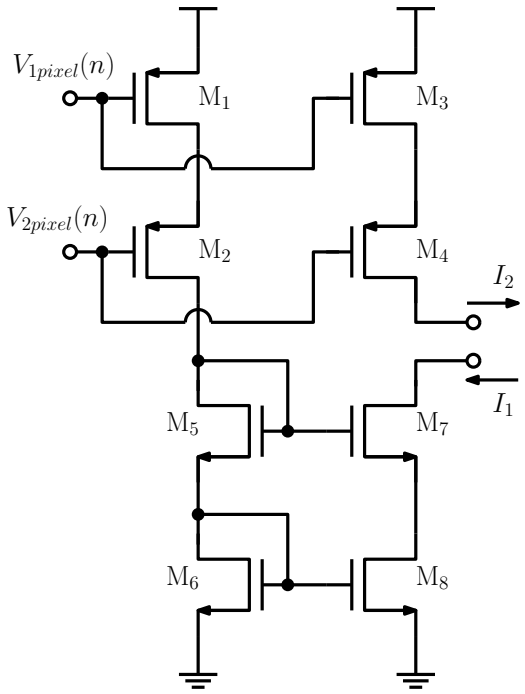


Figura 3.6: Diagrama esquemático com exemplo de implementação da matriz **H**.

linha são multiplicados por zero, conseguimos encontrar a corrente máxima de saída da primeira linha da matriz **H**. Repetindo o procedimento para as demais linhas, e para os coeficientes negativos, encontramos que a corrente máxima de saída da primeira e da segunda linha é igual a 30 μA e a corrente mínima para essas duas linhas é igual a -30 μA; para as linhas três e quatro, temos 20 μA no máximo e no mínimo -20 μA; para quinta linha, 45 μA e -45 μA. Esses valores são críticos, pois a polarização dos transistores se torna mais difícil para correntes mais altas. É importante que os circuitos que são ligados às saídas a matriz **H**, os circuitos de valor absoluto, estejam devidamente projetados para receber essas correntes. Esses circuitos serão explicado na Seção 3.3. A parte que é ligada aos espelhos foi utilizada para os testes da matriz **H** e, por ora, é importante saber que dependendo do sentido da corrente de entrada ela irá fluir pelo transistor M_7 ou M_9 do circuito de valor absoluto, mostrado na Figura 3.15. Cada corrente de saída da matriz será ligada a um diferente circuito de valor absoluto através pino de entrada (I_{in}) da Figura 3.15.

Como à linha dois da matriz é igual a linha um, exceto pela ordem dos coeficientes, só precisamos simular uma das duas linhas. O mesmo ocorre para as linhas três e quatro. Para as linhas um, três e cinco, foram feitas simulações DC com correntes de entrada variando de 0 a 10 μA, inicialmente só para as multiplicações positivas, mantendo os coeficientes negativos com entrada zero e, em seguida, só para as multiplicações negativas, mantendo os coefientes positivos com entrada zero.

As correntes de saída foram medidas no *drain* dos transistores M_7 e M_9 do circuito

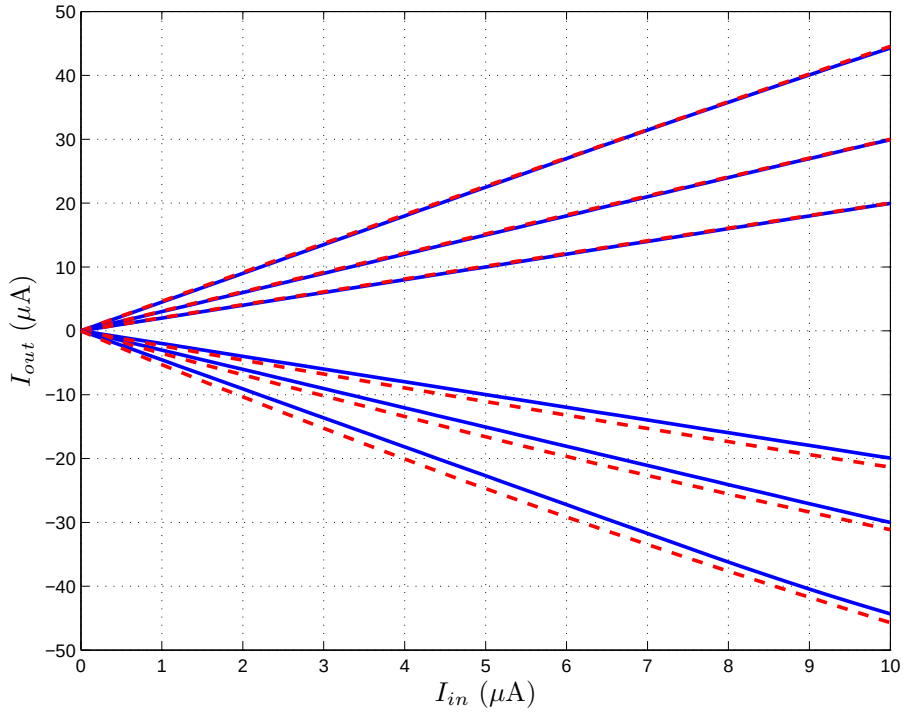


Figura 3.7: Correntes de saída da matriz **H**, em tracejado para o espelho simples e em linha cheia para o espelho *cascode*.

de valor absoluto. Por esse motivo, no gráfico, as correntes que são drenadas dos espelhos P da matriz, e que são lidas no transistor M_9, aparecem positivas, e as correntes que são injetadas nos espelhos N da matriz, e que são lidas no transistor M_7, aparecem negativas. Os resultados das três linhas são apresentados na Figura 3.7. As linhas tracejadas foram encontradas para uma matriz **H** formada com espelhos simples e as linhas cheias foram encontradas para uma matriz **H** formada com espelhos *cascode*.

Pelo gráfico da Figura 3.7, a diferença entre as curvas geradas pelos espelhos simples ou *cascode* parece muito pequena. No entanto, é importante calcular a derivada de cada curva para analisar a sua linearidade e verificar se um dos circuitos distorce mais a saída. A Figura 3.8 mostra a derivada de cada curva da Figura 3.7. Idealmente, a derivada deve ser constante, pois a relação entre a corrente de entrada e saída deve ser linear. Como podemos ver pelos gráficos, o espelho simples apresenta uma variação de derivada maior que a do espelho *cascode*. Assim, podemos concluir que é mais interessante utilizar o espelho *cascode*, cuja derivada é praticamente constante para valores baixos de corrente. Para valores mais altos, percebemos que mesmo para o espelho *cascode* o sinal é um pouco distorcido. Ainda assim, o espelho *cascode* é mais apropriado que o espelho simples.

Também foram feitas simulações de Monte Carlo para espelhos simples e *cascode*. É esperado que o circuito com espelhos *cascode* possua uma variação maior que o

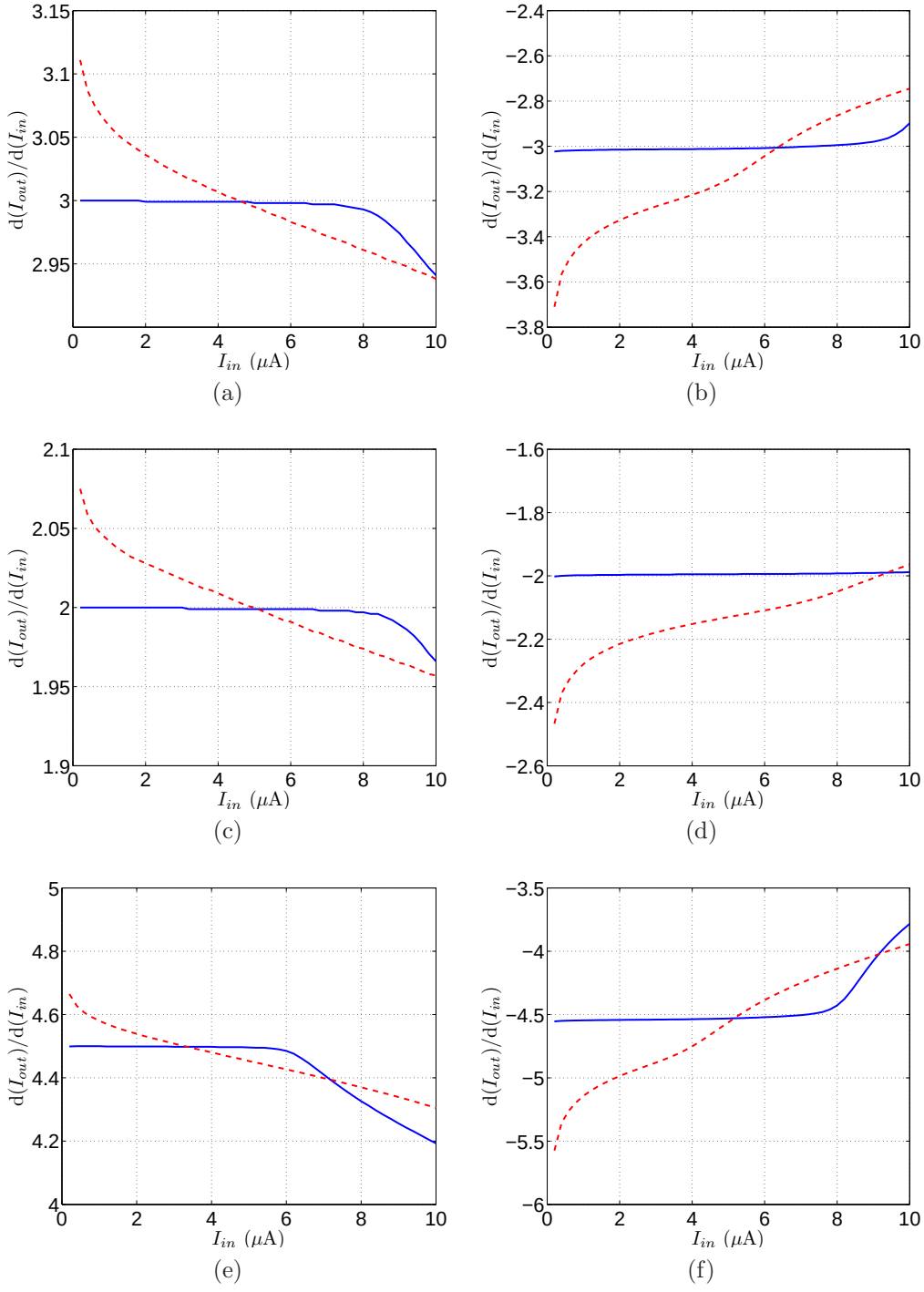


Figura 3.8: Derivada das correntes de saída do circuito que implementa o produto interno entre a matriz **H** e o vetor de pixels: (a) corrente positiva e (b) negativa relativa à linha 1, (c) corrente positiva e (d) negativa relativa à linha 3, (e) corrente positiva e (f) negativa relativa à linha 5. Em linha cheia, para implementação com espelho cascode, e em linha tracejada, para implementação com espelho simples.

circuito com espelhos simples, pois os espelhos *cascode* utilizam transistores menores, que são mais sensíveis às variações de descasamento. Os resultados são mostrados na Figura 3.9.

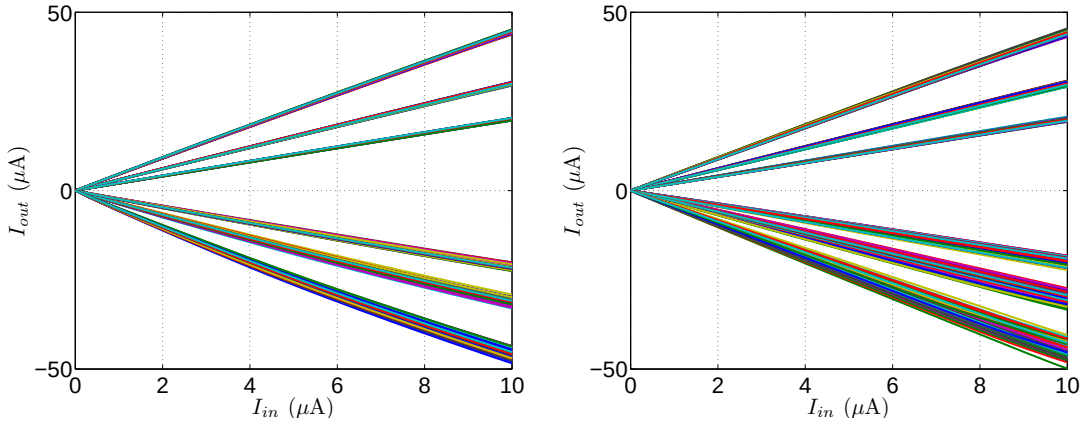


Figura 3.9: (a) Simulação de Monte Carlo para o circuito de produto interno da matriz H com espelhos simples e (b) com espelhos cascode.

Comparando os resultados mostrados nas Figuras 3.9(a) e (b) numericamente, descobrimos que a variação do circuito com *cascode* é aproximadamente duas vezes maior que a variação do circuito com espelhos simples. A máxima variação encontrada no circuito com *cascode* é de 4.7 μA. Essa variação ocorre para a componente P_5, quando a corrente de entrada vale 10 μA para todos os multiplicadores negativos e 0 μA para todos os multiplicadores positivos. Para esse mesmo caso, a variação de corrente para o circuito com espelhos simples é de 2.4 μA. Percentualmente, essa variação do circuito com espelho simples é igual a $\pm$5.3%, e do circuito com espelhos *cascode* é igual a $\pm$10.4%. Como a variação está em torno de 10%, que é considerada aceitável, e a distorção do espelho *cascode* é menor que a do espelho simples, o circuito foi projetado utilizando a configuração *cascode*. Aplicando a variação ao codificador do MATLAB, a PSNR média da Figura 2.2(d) após 1000 rodadas vale 27.16 dB.

As outras etapas do circuito que são implementadas somente com espelhos de corrente são o somatório das correntes dos pixels de um bloco, para o DPCM, e a matriz **W**. Além das cópias das correntes de cada pixel para a transformada **H**, cujos resultados depois alimentarão o VQ, fazemos mais uma cópia de cada pixel do bloco, também com espelhos *cascode*, e somamos todas em um nó. Nesse caso, também realizamos a divisão por quatro que foi mencionada acima. A Capítulo 2 explica o porquê dessa divisão. Os transistores do somatório possuem largura igual a 1 μm, comprimento igual a 0.5 μm e multiplicidade 2.

No caso da matriz **W**, fizemos a implementação utilizando espelhos simples, para facilitar a polarização. As multiplicações positivas são feitas com espelhos N e as negativas com espelhos P. Assim como nos demais casos, utilizamos transistores com dimensões fixos, modificando somente a multiplicidade para possibilitar uma imple-

mentação mais precisa das divisões da matriz **W**. Todos os transistores utilizados nessa matriz possuem largura igual a 0.5 μm e comprimento igual a 2 μm.

3.2 Circuito de Leitura do Fotodiodo

A Figura 3.10 mostra o circuito de leitura do fotodiodo. Nessa figura, os transistores M_1, M_6, M_7 possuem largura e comprimento iguais a 0.5 μm, M_2 possui largura 2.5 μm e comprimento 1 μm, M_3, M_4 e M_5 possuem largura igual a 1 μm e comprimento igual a 2 μm, M_8 e M_9 têm largura 1 μm, comprimento 0.5 μm e multiplicidade igual a 8. Como a compressão é feita em modo de corrente, esse circuito irá transformar a tensão gerada pelo fotodiodo em corrente. Isso é feito através do transistor M_2, que serve de transcondutor. Utilizando as chaves P_1 e P_2 é possível capturar o sinal em dois momentos, um no começo, e outro no final do tempo de integração do fotodiodo. Com isso, realizamos o CDS. A diferença entre as duas capturas é feita no nó de saída do circuito (V_{out2}). A tecnologia utilizada não possui o modelo de um fotodiodo. Por isso, o fotodiodo representado na Figura 3.10, para efeitos de simulação, foi substituído por um capacitor de 5 fF com uma fonte de corrente em paralelo. As chaves P_1 e P_2 são complementares e são utilizados transistores *dummies*, com metade do tamanho dos transistores que funcionam como chave, na saída. A utilização de *dummies* na entrada foi testada e não apresentou melhorias. Os transistores P e N utilizados na chave possuem dimensões iguais e próximas da mínima.

Para iniciar a captura de uma imagem, o sinal de *Reset* (por convenção eu vou utilizar *Reset* em letra maiúscula quando me referir ao sinal e *reset* em letra minúscula quando me referir a técnica) é colocado em nível lógico alto, fazendo com que o transistor M_1 opere como uma chave fechada. Dessa forma, o fotodiodo, que pode ser modelado como um capacitor em paralelo com uma fonte de corrente, será carregado para um determinado valor de tensão um pouco abaixo de 1.8 V. Para que o transistor M_2 esteja operando fora da região de corte durante toda a operação do fotodiodo, devemos garantir que tensão no seu *gate* seja sempre menor que a tensão de *source* menos o módulo da tensão de limiar. No caso, o *source* está ligado à tensão de alimentação, e tem potencial igual a 1.8 V, e a tensão de limiar do transistor tipo P definida pela tecnologia é igual a -0.378 V. Logo, a tensão no *gate* desse transistor deve ser sempre menor que 1.422 V. Para garantir que isso aconteça, o transistor de *reset*, M_1, será ativado com 1.5 V, ao invés de 1.8 V. Quando o *reset* do fotodiodo é feito com uma tensão abaixo da tensão de alimentação, é chamado de *soft reset*.

Uma vez que o fotodiodo foi carregado, o sinal de *Reset* irá voltar a zero, colocando o transistor M_1 em corte. O fotodiodo será então descarregado por foto-

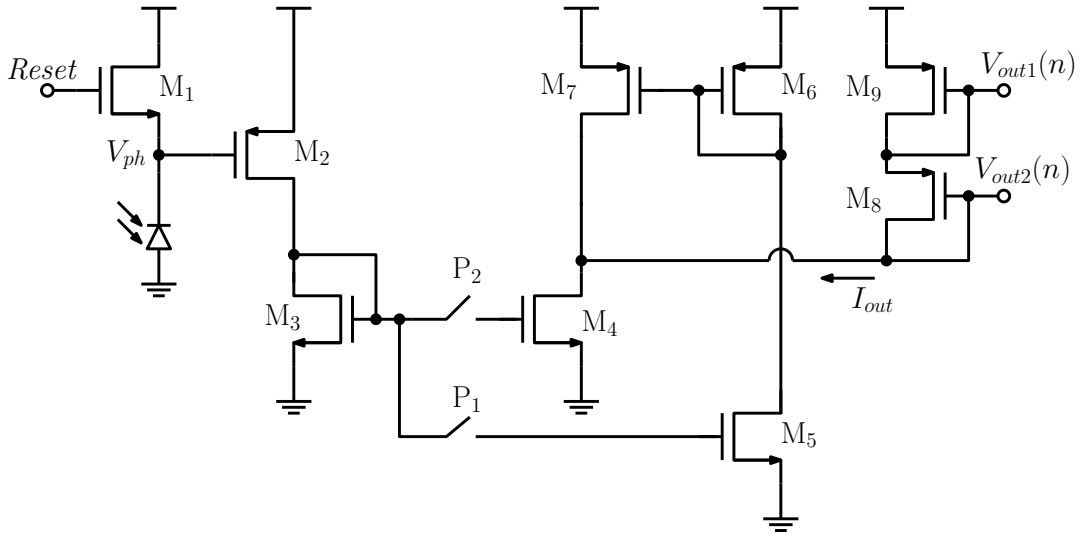


Figura 3.10: Circuito de leitura do fotodiodo.

corrente proporcional à luz que incide sobre ele. A luz incidente é representada no modelo do fotodiodo como a fonte de corrente que é colocada em paralelo com o capacitor de 5 fF. Quanto mais luz, maior é o valor da corrente. No caso desse projeto, consideramos uma corrente de 20 pA no fotodiodo é suficiente para gerar uma imagem branca. Esse valor foi estabelecido através do que foi observado nas medidas do chip fabricado com tecnologia de 0.35 μm. No entanto, ele não é preciso, pois as tecnologias são diferentes e é esperado que a eficiência quântica seja menor. Sendo assim, pode ser necessário alterar o tempo de integração durante as medidas práticas para garantir uma imagem clara.

O transistor M_2, que está trabalhando como um transcondutor, opera na região de saturação. Portanto, haverá uma não-linearidade na transformação da tensão no *gate* para a corrente no *drain*. Essa não-linearidade será analisada mais adiante nessa seção, assim como as suas consequências.

No início da operação do circuito de leitura, tanto a chave P_1 quanto a chave P_2 estão fechadas. Logo após o *Reset*, a chave P_1 é aberta, fazendo com que o nível de tensão do *gate* do transistor M_5 seja mantido, devido à capacitância parasita C_{GS} desse transistor. Com isso, a corrente de *drain* que passa por M_5 será constante e indicativa da tensão no fotodiodo no início da integração. A chave P_2 será aberta após 300 μs, que é o tempo de integração escolhido. Assim como no caso do transistor M_5, a tensão no *gate* do transistor M_4 também será mantida e a corrente que passa pelo *drain* desse transistor após a abertura da chave P_2 será constante. Os transistores M_6 e M_7 são responsáveis por inverter o sentido da corrente capturada no início da leitura, de forma que é feita uma subtração no nó de *drain* dos transistores M_4 e M_7, completando a operação de CDS. Os transistores M_8 e M_9 compõe a metade de um espelho de corrente *cascode*. A corrente que passa por esses transistores pode

ser copiada se outros dois transistores tipo P forem conectados a M_8 e M_9 de forma a completar o espelho. Os espelhos de corrente foram explicados em detalhes na Seção 3.1.

Os gráficos apresentados na Figura 3.11 mostram os sinais de controle utilizados para as simulações, a tensão no fotodiodo e as correntes nos transistores M_4, M_5 e M_8 durante o tempo de integração, quando a corrente no fotodiodo é máxima (igual a 20 pA). O sinal de *Reset* é mostrado na Figura 3.11(a). Esse sinal é mantido em 1.5 V por 55 μs, tempo suficiente para que a capacitância do nó do fotodiodo seja carregada. Podemos observar a consequência do *Reset* na Figura 3.11(d), que mostra a tensão no fotodiodo. Como o *Reset* é acionado com 1.5 V, a tensão no fotodiodo irá se estabilizar em 1.2 V, aproximadamente. Dessa forma, o transistor M_2 estará ativo desde o início da integração. Quando o *Reset* volta para 0 V, a tensão no fotodiodo começa a diminuir em função da quantidade de luz. Para diminuir o efeito da não-linearidade, fazemos com que a variação da tensão do fotodiodo seja pequena, em torno de 0.3 V. O ganho do transistor M_2 foi escolhido por meio de simulações de forma que, ao final do tempo de integração de 300 μs, a corrente máxima gerada seja igual a 11 μA, quando a tensão no seu *gate* for 0.9 V, e a corrente mínima seja igual a 1 μA, quando a tensão no seu *gate* for 1.2 V.

Os sinais de controle das chaves que realizam a amostragem do sinal, P_1 e P_2, mostrados nas Figuras 3.11(b) e (c), começam em nível lógico alto (chave fechada) somente para facilitar a convergência do simulador. Na prática, o que importa é o momento em que as chaves abrem para fazer a amostragem. Portanto, podemos analisar o resultado após os 5 μs iniciais de simulação, quando P_1 e P_2 ficam iguais a 0 V. A chave P_1, que realiza a primeira amostragem, se abre instantes após o *Reset*. Dessa forma, a corrente correspondente ao valor inicial do fotodiodo é guardada. Esse efeito pode ser observado na Figura 3.11(e), pois a corrente no transistor M_5 é mantida constante, em aproximadamente 1 μA, a partir do momento em que o sinal P_1 assume nível lógico 0, em 70 μs. Como o objetivo é fazer uma amostragem logo no início da integração, o ciclo do sinal P_1 será igual ao ciclo do *Reset*. A chave P_2 será aberta no final do tempo de integração, 300 μs após a primeira amostragem. O resultado também é apresentado na Figura 3.11(e), uma vez que a corrente no transistor M_4 se mantém constante no instante que P_2 assume nível lógico 0. Podemos observar uma pequena injeção de carga proveniente do fechamento de P_2 nesse gráfico. A diferença entre as duas correntes mostradas na Figura 3.11(e) é apresentada na Figura 3.11(f). Como pode ser visto nessa figura, o projeto foi feito de forma que a corrente de saída do circuito de leitura seja igual a 10 μA quando corrente no fotodiodo é máxima.

A relação de transferência entre a corrente do fotodiodo e a corrente de saída do circuito (I_{out}), que é igual à corrente que passa pelo transistor M_8, pode ser vista no

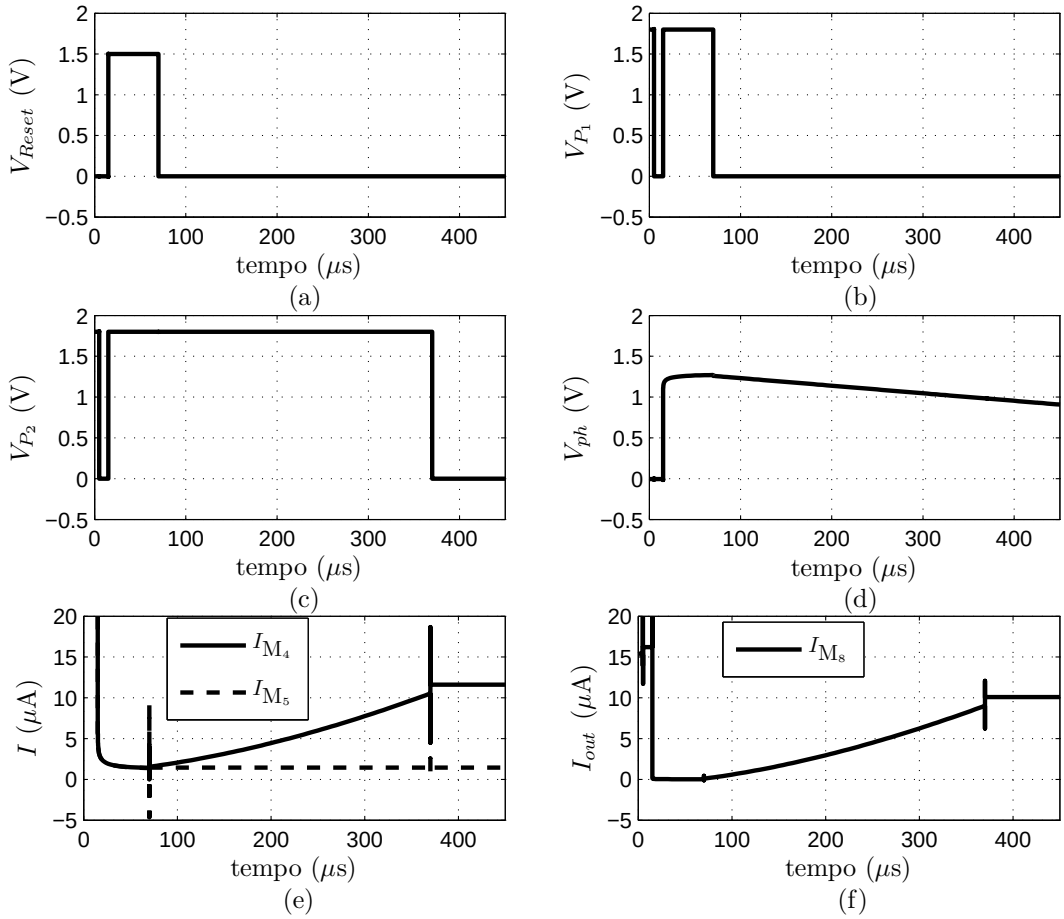


Figura 3.11: (a) *Reset*, (b) P_1 , (c) P_2, (d) tensão no fotodiodo quando a luminosidade é máxima, (e) corrente nos transistores M_4 e M_5 quando a luminosidade é máxima e (f) corrente de saída quando a luminosidade é máxima (subtração entre as duas correntes mostradas no gráfico (e)).

gráfico em linha cheia da Figura 3.12. Essa curva foi encontrada através de várias simulações (com diferentes valores de corrente no fotodiodo) do circuito de leitura no Cadence. Como esperado, uma vez que o transistor M_2 está na saturação, a relação entre entrada e saída é quadrática. Essa não-linearidade possui duas consequências principais: a imagem capturada ficará mais escura do que a imagem original e o CDS será menos efetivo que o desejado. O primeiro efeito pode ser percebido ao compararmos a transferência ideal, curva tracejada mostrada na Figura 3.12, com a estimada por simulação. Como a curva ideal possui valores de saída mais altos que a curva encontrada através da simulação do circuito, o circuito de compressão entende que imagem é mais escura do que deveria. Por exemplo, pelos pontos marcados na Figura 3.12 pelas linhas formadas com pontos e traços, nós podemos perceber que se nós tivermos uma entrada de 6.7 pA, a saída do circuito de leitura será 2 μA, quando deveria ser 3.35 μA. Assim, o circuito entende que está recebendo uma corrente referente a uma entrada de 4 pA, o que corresponde a um sinal mais

escuro. Essa não-linearidade causa uma diminuição significativa na PSNR, apesar do efeito subjetivo não ser muito ruim. Na Seção 5.4, será proposto um método para corrigir essa não-linearidade no decodificador.

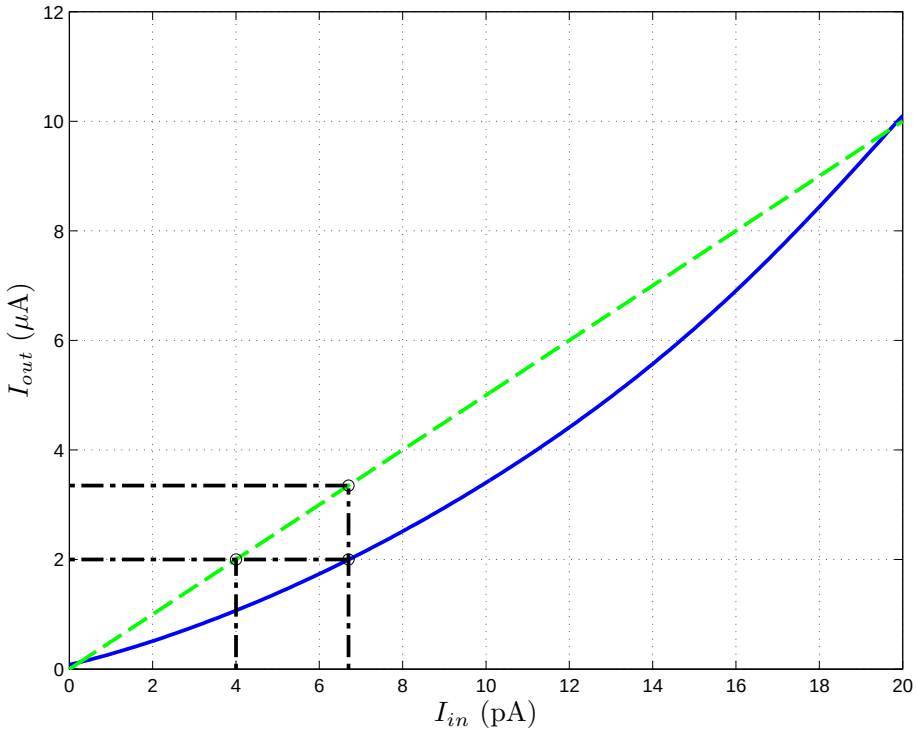


Figura 3.12: Em linha cheia, relação entre a corrente no fotodiodo (I_{in}) e a corrente de saída do circuito de leitura (Iout) e em linha tracejada, relação ideal entre I_{in} e I_{out}. As linhas formadas com pontos e traços marca os pontos 4 e 6.7 pA de I_{in}, e 2 e 3.35 μA de I_{out} que serão mencionados no texto.

O segundo efeito indesejado, causado pela não-linearidade da resposta, afeta a resposta do CDS. Desejamos que o CDS reduza os ruídos que estão presentes tanto no início quanto no final da integração do fotodiodo. A diferença de V_t entre os transistores de *reset* presentes na matriz de pixels é um exemplo de um desses ruídos. Como a corrente de saída do transistor M_2 depende do V_t, se essa tensão de limiar for diferente de um transistor da matriz para outro, uma mesma quantidade de luz pode gerar saídas diferentes em dois pixels da matriz. Com o CDS, conseguimos diminuir esse erro. Utilizando o modelo de primeira ordem do transistor MOS na saturação, conseguimos ver a parcela restante de V_t na saída. As Equações (3.10), (3.11) e (3.12) mostram a corrente proporcional à amostragem no início do tempo de integração, no final do tempo de integração e a subtração das duas amostras de corrente, onde V_{sg1} é a tensão entre *source* e *gate* do transistor M_2 logo após o *Reset*, e V_{sg2} é a tensão entre *source* e *gate* do transistor M_2 após 300 μs de descarga do fotodiodo. Com essas equações, percebemos que o CDS consegue acabar com a parcela de erro proporcional a V_t^2, mas não com a proporcional a V_t.

$$I_{out(inicial)} \propto (V_{sg1} - |V_t|)^2 = V_{sg1}^2 - 2 \cdot V_{sg1} \cdot |V_t| + |V_t|^2 \quad (3.10)$$

$$I_{out(final)} \propto (V_{sg2} - |V_t|)^2 = V_{sg2}^2 - 2 \cdot V_{sg2} \cdot |V_t| + |V_t|^2 \quad (3.11)$$

$$I_{out(final)} - I_{out(inicial)} \propto (V_{sg2}^2 - V_{sg1}^2) - 2 \cdot (V_{sg2} - V_{sg1}) \cdot |V_t| \quad (3.12)$$

Para avaliar o efeito do processo de fabricação do chip no circuito de leitura foram feitas simulações de Monte Carlo no simulador da Cadence. O simulador permite que serjam considerados erros devidos às falhas no processo de fabricação ("processo") e erros devidos aos decasamentos entre os dispositivos ("*mismatch*"). Ambos os erros são causados pela fabricação, mas a simulação de "processo" considera as variações que afetam todos os componentes da mesma forma e o "*mismatch*" considera variações individuais de cada componente, de forma que podemos verificar o efeito dessas flutuações locais (descasamento). Para o circuito de leitura foram feitas simulações considerando somente processo, somente *mismatch* e processo e *mismatch* juntos. O resultado da simulação de Monte Carlo da tensão sobre o fotodiodo pode ser visto na Figura 3.13. Comparando os resultados dos três tipos de simulação de Monte Carlo feitas, concluímos que os erros devido ao processo de fabricação são os maiores responsáveis pelas variações de tensão.

As Figura 3.14(a) e (b) mostram o efeito da variação de processo e *mismatch* nas correntes que passam pelos transistores M_5 e M_4, respectivamente. Devido ao CDS, a variação da corrente de saída, mostrada na Figura 3.14(c), é menor que a variação das correntes antes da subtração. Comparando a simulação de Monte Carlo da corrente de saída com a simulação nominal, mostrada na Figura 3.14(d), concluímos que a saída possui uma variação de mais ou menos 27% em relação ao valor nominal. Idealmente, essa variação deveria ser a menor possível. No entanto, como a maior parte dela é consequência do processo de fabricação, não é possível melhorá-la significativamente se mantivermos o mesmo circuito. Essa variação causa um efeito de ruído na imagem e pode diminuir bastante a PSNR. No entanto, o ruído causado é correspondente a um padrão fixo e pode ser identificado para cada circuito e subtraído no decodificador.

3.3 Circuito de Valor Absoluto

A Figura 3.15 mostra o circuito utilizado para calcular o módulo de uma corrente de entrada [21]. Independente do sentido da corrente de entrada, teremos na saída correntes proporcionais à corrente de entrada. Ela pode ser copiada por um transistor tipo P ou por um transistor tipo N.

Nesse circuito, o nó de entrada é capacitivo. A corrente I_{in} irá carregar ou

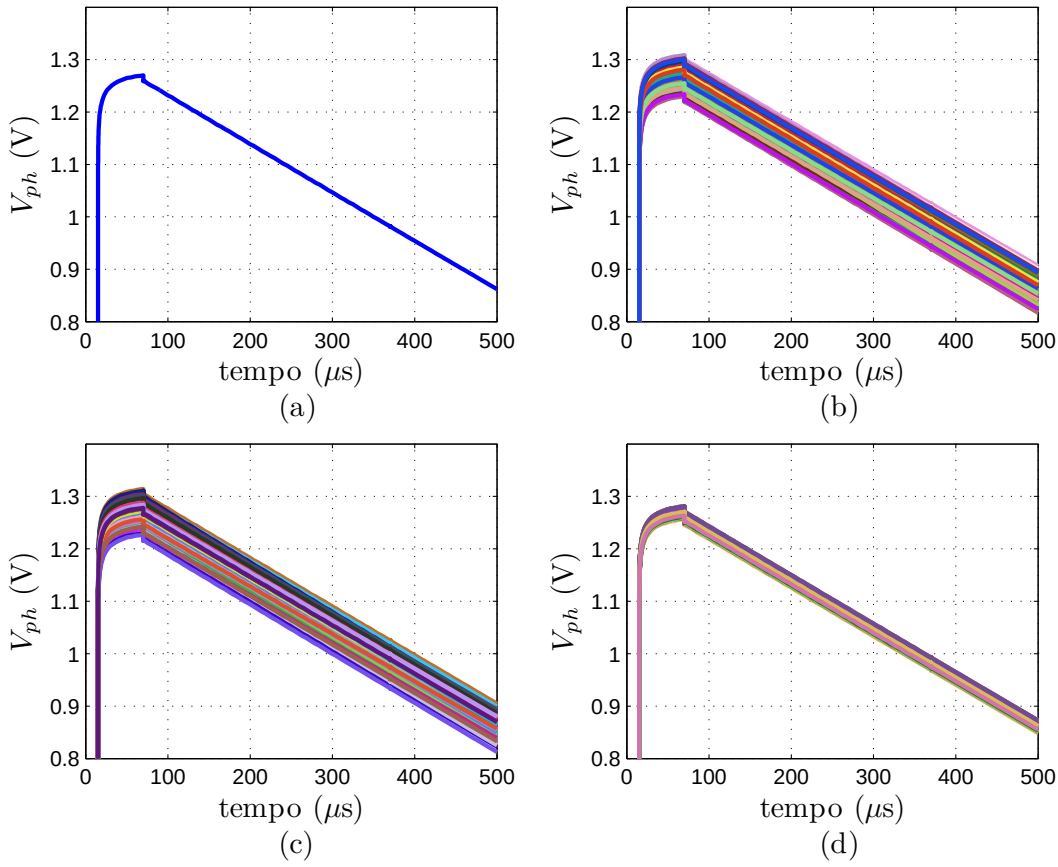


Figura 3.13: (a) Tensão nominal no fotodiodo, (b) simulação de Monte Carlo completa, (c) simulação de Monte Carlo somente com erros devidos a falhas no processo de fabricação ("processo"), (d) simulação de Monte Carlo somente com erros devidos a descasamento entre os dispositivos fabricados ("*mismatch*").

descarregar esse capacitor, caso a corrente seja injetada ou drenada do circuito, respectivamente. Assim, o nó de entrada atingirá uma tensão que, no mínimo, irá deixar um dos transistores, M_3 ou M_4, em triodo e o outro em saturação. Idealmente, um dos transistores deve ficar em triodo e o outro em corte, mas quando a corrente de entrada é muito pequena, a tensão no nó de entrada não atinge o valor necessário para que um dos transistores fique em corte.

Quando a corrente é injetada, a tensão no nó de entrada será tal que o transistor M_4 entrará em triodo, com nível lógico positivo. Logo, a tensão no *drain* desse transistor será próxima de zero. Com isso, o transistor M_1 ficará em corte e o transistor M_2 em triodo, fazendo a corrente passar pelo espelho de corrente formado por M_9 e M_{10}. No caso em que a corrente é drenada, ocorre o processo inverso: a tensão no nó de entrada será suficiente para que M_3 fique em triodo, com nível lógico zero. Dessa forma, o *drain* desse transistor ficará com uma tensão próxima de 1.8 V, com nível lógico 1. Consequentemente, o transistor M_2 ficará em corte e o transistor M_1 em triodo, fazendo a corrente passar pelo espelho formado por M_7

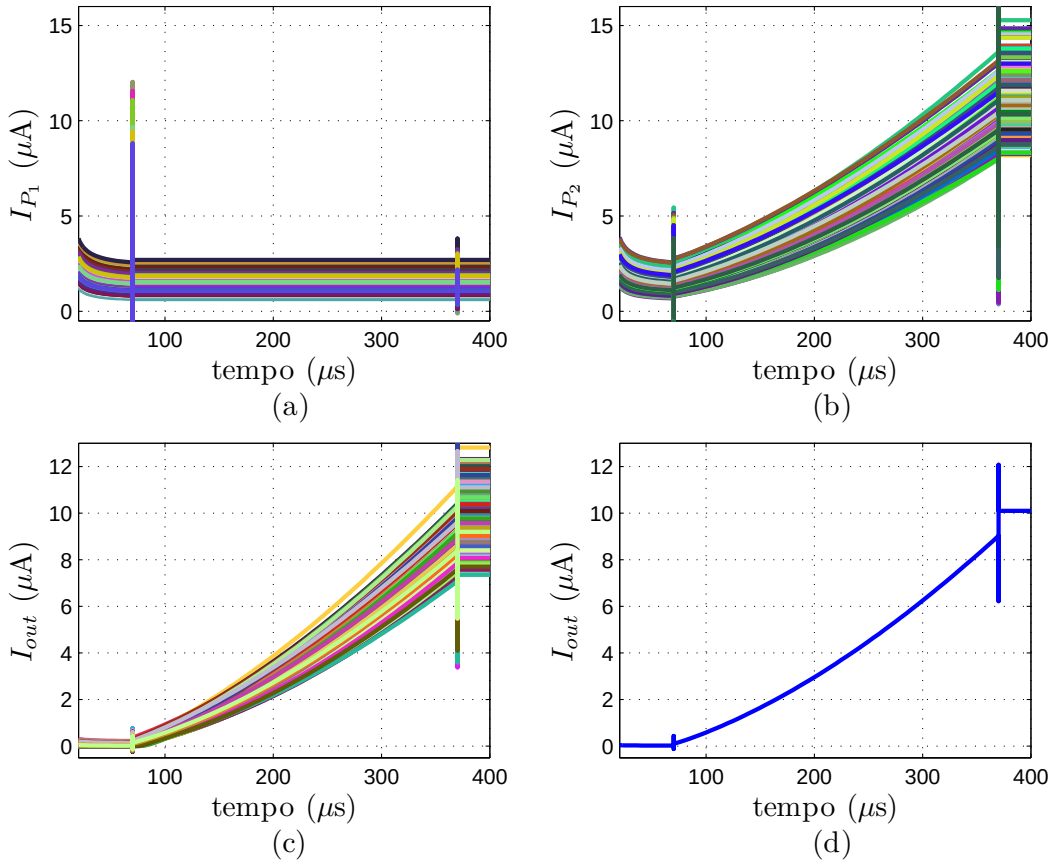


Figura 3.14: (a) Simulação de Monte Carlo da corrente que passa no transistor M_5, (b) simulação de Monte Carlo da corrente que passa no transistor M_4, (c) simulação de Monte Carlo da corrente que passa no transistor M_8 e (d) valor nominal da corrente que passa no trasistor M_8.

e M_8 para só depois passar por M_9 e M_{10}.

O transistor M_{10} é ligado a M_{11}, de forma que a corrente possa ser copiada em qualquer sentido: ligando o *gate* de um transistor tipo N ao *gate* de M_9 ou ligando o *gate* de um transistor tipo P ao *gate* do transistor M_{11}. O sinal, que indica o sentido da corrente de entrada, será representado pelo bit $s_m(n)$. Os transistores M_5 e M_6 formam um inversor.Como o *gate* desses transistores está ligado ao *drain* dos transistores M_3 e M_4, quando a corrente for injetada, e a tensão no *drain* de M_3 e M_4 for próxima de zero, $s_m(n)$ possuirá nível lógido 1. No caso em que a corrente for drenada e a tensão no *drain* de M_3 e M_4 for próxima de 1.8 V, $s_m(n)$ possuirá nível lógico zero.

São necessários seis circuitos de valor absoluto por bloco de 4×4 pixels: um para o DPCM e um para cada uma das cinco componentes de entrada do VQ. No entanto, somente quatro desses circuitos são distintos: os circuitos de valor absoluto das componentes p_1 e p_2 são iguais, bem como os circuitos de valor absoluto das componentes p_3 e p_4. Os circuitos foram projetados de forma diferente por dois

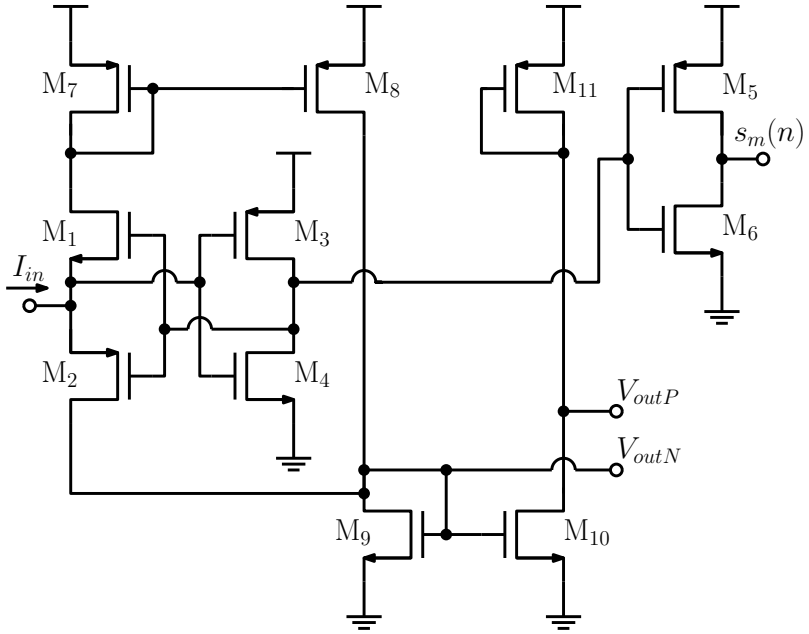


Figura 3.15: Diagrama esquemático do circuito de valor absoluto.

motivos: as correntes de entrada de cada circuito são diferentes e os fatores de escala das componentes são diferentes. No circuito de valor absoluto do DPCM, a corrente de entrada pode variar de -40 μA a 40 μA, pois, como foi explicado na Seção 3.2, a saída do circuito de leitura, que pode chegar a 10 μA, é dividida por quatro. Sendo assim, teremos no máximo 2.5 μA multiplicados por 16, considerando todos os pixels do bloco. Para os circuitos das componentes p_1 e p_2, devemos considerar as saídas do circuito que realiza o produto interno com a matriz **H**. Assim, descobrimos que a corrente de entrada do circuito de valor absoluto de p_1 e p_2 pode variar de -30 μA a 30 μA. Fazendo o mesmo para p_3 e p_4, a corrente pode variar de -20 μA a 20 μA, e para p_5, a corrente pode variar de -45 μA a 45 μA.

Os fatores de escala das cinco componentes foram explicados na Seção 2.1. Esses fatores serão implementados na saída de cada circuito de valor absoluto das componentes. Isso é feito modificando as multiplicidades dos transistor M_9 e M_{10} e considerando essa modificação ao implementar o produto interno com a matriz **W**.

Os circuitos de valor absoluto do DPCM e da componente p_5 não precisam da saída negativa, portanto nenhum dos dois possui os transistores M_{10} e M_{11}. Para todos os circuitos, os transistores M_1 a M_6 possuem largura igual a 0.5 μm e comprimento igual a 0.18 μm. Os transistores restantes do DPCM, M_7 a M_9, possuem largura igual a 1.5 μm e comprimento igual a 1 μm, todos com multiplicidade 1. Os circuitos para as componentes p_1 (que é igual a p_2) e p_3 (que é igual a p_4) possuem M_7 e M_8 com W igual a 2 μm e L igual a 1 μm. Os transistores M_9 a M_{11} possuem W igual a 0.5 μm e L igual a 2 μm, no entanto a multiplicidade de M_9 para as componentes 1 e 2 é igual a 8 e para as componentes 3 e 4 é igual a 2. Para todas as

quatro componentes, a multiplicidade de M_{10} é igual a 4 e a multiplicidade de M_{11} é igual a 8. No caso da componente p_5, M_7 e M_8 possuem lagura igual a 2.5 μm e comprimento igual a 1 μm. O transistor M_9 dessa componente possui largura igual a 0.5 μm e comprimento igual a 2 μm, com multiplicidade igual a 5.

A simulação nominal dos quatro circuitos diferentes de valor absoluto pode ser vista na Figura 3.16. Para melhor visualização, o sinal negativo das componentes p_1 e p_3 foi multiplicado por 2 e por 0.5, respectivamente. Dessa forma, compensamos a multiplicação feita pelo espelho formado por M_9 e M_{10} para que o sinal negativo fique na mesma faixa que o sinal positivo. A simulação de Monte Carlo pode ser vista na Figura 3.17. A maior variação desses circuitos é mostrada na Figura 3.17(b), onde a parte negativa possui variação de 12%. O efeito dessa variação na PSNR da imagem é insignificante.

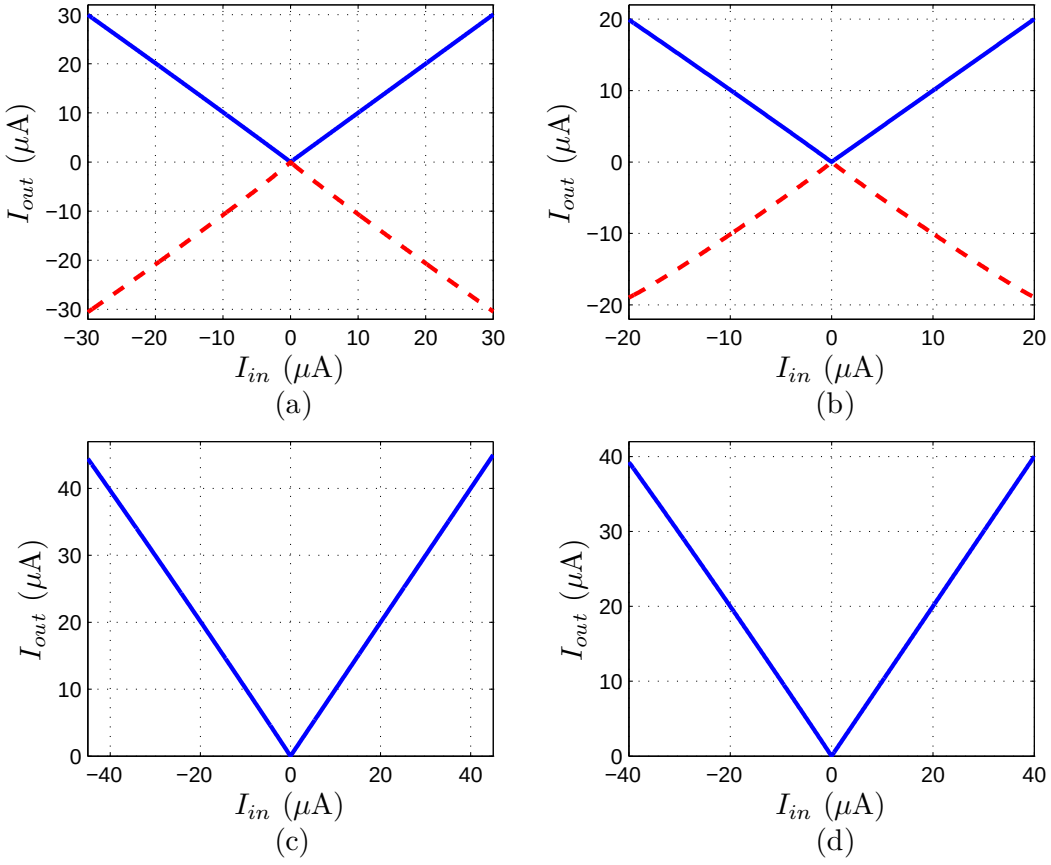


Figura 3.16: Simulação nominal do circuito de valor absoluto. (a) Simulação do circuito para as componentes p_1 e p_2: saída positiva em linha cheia e saída negativa linha tracejada; (b) componentes p_3 e p_4: saída positiva em linha cheia e saída negativa em linha tracejada; (c) componente p_5 e (d) circuito do DPCM.

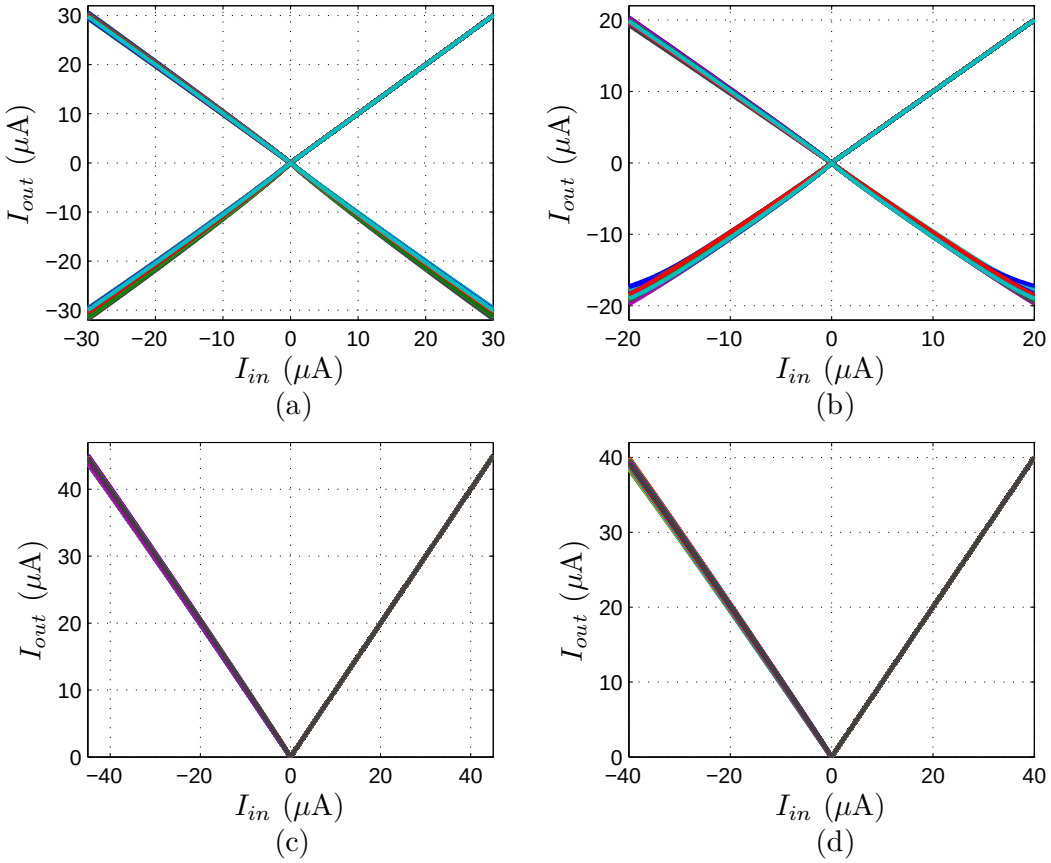


Figura 3.17: (a) Simulação de Monte Carlo do circuito de valor absoluto para as componentes p_1 e p_2, saída positiva e negativa; (b) componentes p_3 e p_4; saída positiva e saída negativa, (c) componente p_5 e (d) circuito do DPCM.

3.4 Comparadores de Corrente

Para verificar a célula do quantizador do DPCM em que se encontra uma determinada corrente, utilizamos os comparadores de corrente mostrados na Figura 3.18. Os transistores M_1 e M_2 são duas metades de espelhos de corrente tipo P e N, respectivamente. A tensão na saída irá indicar se a corrente I_p é maior ou menor que a corrente I_n. O circuito que realiza a comparação é o inversor formado por M_3 e M_4. No VQ, o *gate* de M_3 e M_4 será ligado à saída de corrente da matriz $\mathbf{W}$, que deve ser comparada com um dos limiares apresentados nas Equações (2.8) a (2.12).

Se a corrente copiada pelo transistor M_1 for maior que a corrente copiada pelo transistor M_2, a tensão de saída V_{out} é igual a 0 V, nível lógico 0. Isso acontece, pois o nó V_x é capacitivo e será carregado por um breve período de tempo. Como I_p é maior que I_n, a corrente resultante para o capacitor irá carregá-lo. Se ocorrer o contrário, corrente de M_2 maior que de M_1, V_{out} será igual a 1.8 V, nível lógico 1. Nesse caso, com I_n maior que I_p, a corrente resultante irá descarregar o capacitor.

Para avaliar se o circuito comparador escolhido atende às especificações do ima-

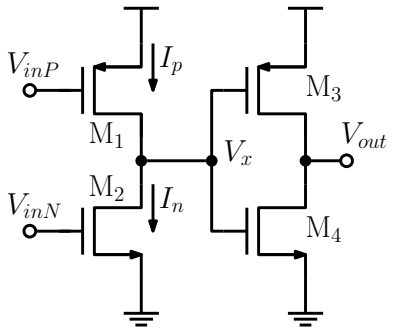


Figura 3.18: Diagrama esquemático do circuito comparador de corrente.

geador, foi feita uma simulação DC utilizando o circuito mostrado na Figura 3.18, com V_{inP} ligado ao *gate* de um transistor tipo P conectado como entrada de um espelho de corrente e com V_{inN} ligado ao *gate* de um transistor tipo N também conectado como entrada de um espelho de corrente. Há uma simulação variando I_n e escolhendo I_p igual a cada valor de limiar positivo (0.25 μA, 0.5 μA, 0.75 μA, 1.0 μA, 1.5 μA, 1.75 μA, 2.5 μA, 3.0 μA, 4.5 μA, 5.0 μA, 6.0 μA, 7.5 μA, 9.0 μA, 11.0 μA, 16.0 μA), e outra simulação variando a corrente I_p e escolhendo I_n igual a cada valor de limiar negativo (-1.5 μA, -0.25 μA). O resultado das duas simulações pode ser visto na Figura 3.19.

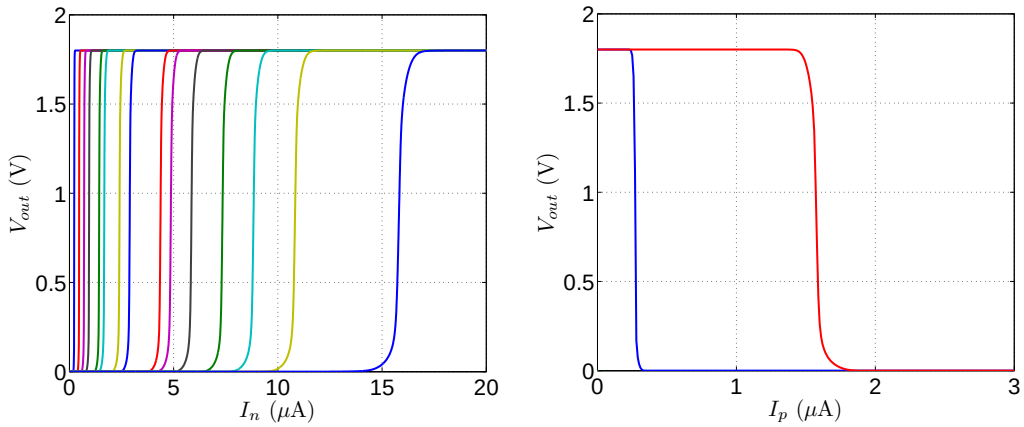


Figura 3.19: (a) Simulação DC dos comparadores do DPCM e dos comparadores positivos do VQ, e (b) simulação DC dos comparadores negativos do VQ.

No caso da Figura 3.19(a), a corrente I_p é fixa e igual aos limiares, e I_n é a corrente que desejamos comparar com os limiares. Na Figura 3.19(b) ocorre o contrário, pois agora os limiares são negativos e representados por I_n. Nesse caso, I_p é a corrente a ser comparada com os limiares. Em ambos os casos, quando a corrente a ser comparada atinge o limiar, a tensão na saída muda, como desejado.

Tanto para o DPCM quanto para o VQ, os transistores M$_3$ e M$_4$ possuem largura

igual a 0.5 μm e comprimento igual a 0.18 μm. Para o DPCM, M_1 e M_2 possuem largura 1.5 μm e comprimento 1 μm.

3.5 Circuito de Reconstrução do DPCM

Para gerar o sinal do DPCM do bloco atual que será referência para o bloco seguinte ($\hat{s}(n+1)$), é necessário reconstruir, em modo de corrente, o erro que foi codificado. O erro codificado é a diferença entre o somatório das correntes do bloco dividido por quatro e uma previsão desse sinal. É essa previsão que é dada pelo circuito de reconstrução do bloco anterior. Como foi explicado na Seção 2.3, no caso do primeiro bloco, consideramos uma corrente de referência igual 18.75 μA, por ser o centroide mais próximo da metade do valor máximo que $s(n)$ pode atingir. O valor de $s(n)$ pode ser no máximo igual a $10 \cdot 16/4 = 40$ μA, pois o circuito de leitura foi projetado para ter uma corrente de saída que pode variar de 0 a 10 μA, são 16 pixels no bloco, e esse somatório é dividido por quatro no circuito que realiza a soma das correntes. A subtração entre $s(n)$ e $\hat{s}(n)$ pode ser representada por um bit de sinal ($d_0(n)$) e sete bits de módulo ($o_{01}(n)$ até $o_{07}(n)$). Esses oito bits são utilizados para realizar a reconstrução. A Figura 3.20 mostra o diagrama esquemático do circuito de reconstrução.

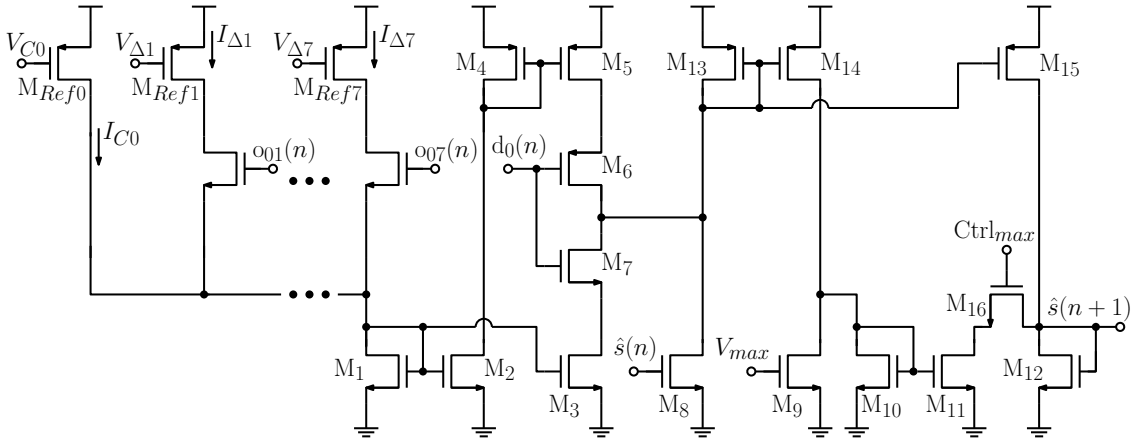


Figura 3.20: Diagrama esquemático do circuito de reconstrução do DPCM.

Os sete bits de módulo são usados para controlar sete chaves, cada uma implementada com um transistor tipo N, ligadas a correntes de referência. Essas correntes são geradas aplicando ao *gate* dos transistores M_{Ref0} ao M_{Ref7} tensões proporcionais às correntes desejadas. O circuito explicado na Seção 3.1.1 será responsável por gerar essas tensões. Quando somamos as correntes à corrente I_{C0}, essa soma equivale a um centroide do quantizador escalar do DPCM. Os bits que representam o módulo são gerados a partir de um codificador que funciona como um termômetro: se a corrente passar de um determinado limiar, todos os bits abaixo desse limiar serão iguais

a 1 e todos os bits acima desse limiar serão iguais a 0. Sendo assim, se a corrente for representada pelo primeiro centroide, ela é menor que todos os limiares e todos os bits serão zero. Se a corrente for representada pelo último centroide (o oitavo), ela é maior que todos os limiares e todos os bits serão iguais a 1. Se a corrente for representada pelo quarto centroide, ela é maior que os três primeiros limiares, menor que os três últimos limiares e a sequência de bits será igual a 1110000.

Sabendo como os bits funcionam, podemos deduzir que, para gerar a corrente referente ao primeiro centroide, basta deixar todas as chaves abertas. Somente I_{C0} deve ser considerado para o restante do circuito. Para gerar a corrente referente ao segundo centroide, só a primeira chave deve estar fechada, assim I_{C0} será somado a $I_{\Delta 1}$. O terceiro centroide é encontrado pelo somatório de I_{C0}, $I_{\Delta 1}$ e $I_{\Delta 2}$ e assim por diante, até que todas as chaves estejam fechadas, para que o oitavo centroide possa ser encontrado.

Uma vez que o módulo da corrente foi decodificado, pois encontramos o cetroide que o representa, devemos utilizar o bit de sinal para considerar o sentido correto da corrente. Com o sentido da corrente, o sinal está completo e devemos somá-lo com a corrente $\hat{s}(n)$ do bloco anterior. Dessa forma, encontramos a estimativa do bloco seguinte.

No final desse circuito acrescentamos uma etapa para limitar a corrente de saída, que pode ser ligada fechando a chave M_{16}. Caso essa chave esteja aberta, a corrente de saída pode ultrapassar o valor máximo de 40 μA (16 pixels com 2.5 μA na saída). No entanto, se a chave estiver fechada, são feitas duas cópias da corrente reconstruída. Uma delas está ligada ao mesmo nó que uma corrente de referência que pode variar dependendo do valor da corrente à qual queremos limitar o circuito. Os resultados apresentados com o circuito que limita a corrente consideram uma referência de 40 μA e outra de 44 μA. Se a corrente reconstruída for maior que a corrente de referência, a diferença será copiada pelo espelho de corrente formado por M_{10} e M_{11}. Essa diferença será então retirada da corrente reconstruída, pois M_{11} está ligado ao mesmo nó que M_{15}, que copia a corrente reconstruída. Dessa forma, a corrente que será copiada para o bloco seguinte será no máximo igual à corrente de referência. Se a corrente reconstruída for menor que a corrente de referência, o espelho de corrente formado por M_{10} e M_{11} não irá funcionar e a corrente que será copiada para o bloco seguinte será igual à corrente que passa por M_{13}.

Dois testes do circuito de reconstrução foram feitos conectando cinco blocos de DPCM seguidos. No primeiro teste, a corrente de entrada dos três primeiros blocos é constante e máxima, igual a 40 μA. A corrente do quarto bloco varia de 0 a 50 μA, com o objetivo de analisar a resposta do circuito caso a corrente exceda o valor máximo. A corrente do quinto bloco é constante e igual a 20 μA. No segundo teste, a corrente do terceiro bloco é igual a 10 μA e as demais correntes permanecem iguais

às do teste anterior. Nos dois casos a corrente de saída não é limitada.

Os resultados são mostrados na Figura 3.21. Como podemos ver na figura, o DPCM funciona como esperado. A corrente reconstruída é capaz de seguir a corrente de entrada, se a diferença entre a corrente de entrada e a corrente prevista for menor que 18.75 μA, que é o maior centroide do quantizador escalar do DPCM. No caso mostrado na Figura 3.21(a), a corrente prevista é aproximadamente igual a 39.5 μA. Então, até que a corrente de entrada seja maior que 20.75 μA, a diferença entre a corrente de entrada e a corrente prevista vai ser considerada igual a -18.75 μA. Assim, a corrente reconstruída será igual a 39.5-18.75 μA, que é igual a 20.75 μA, como pode ser visto na figura. O mesmo cálculo pode ser feito para o caso mostrado na Figura 3.21(b).

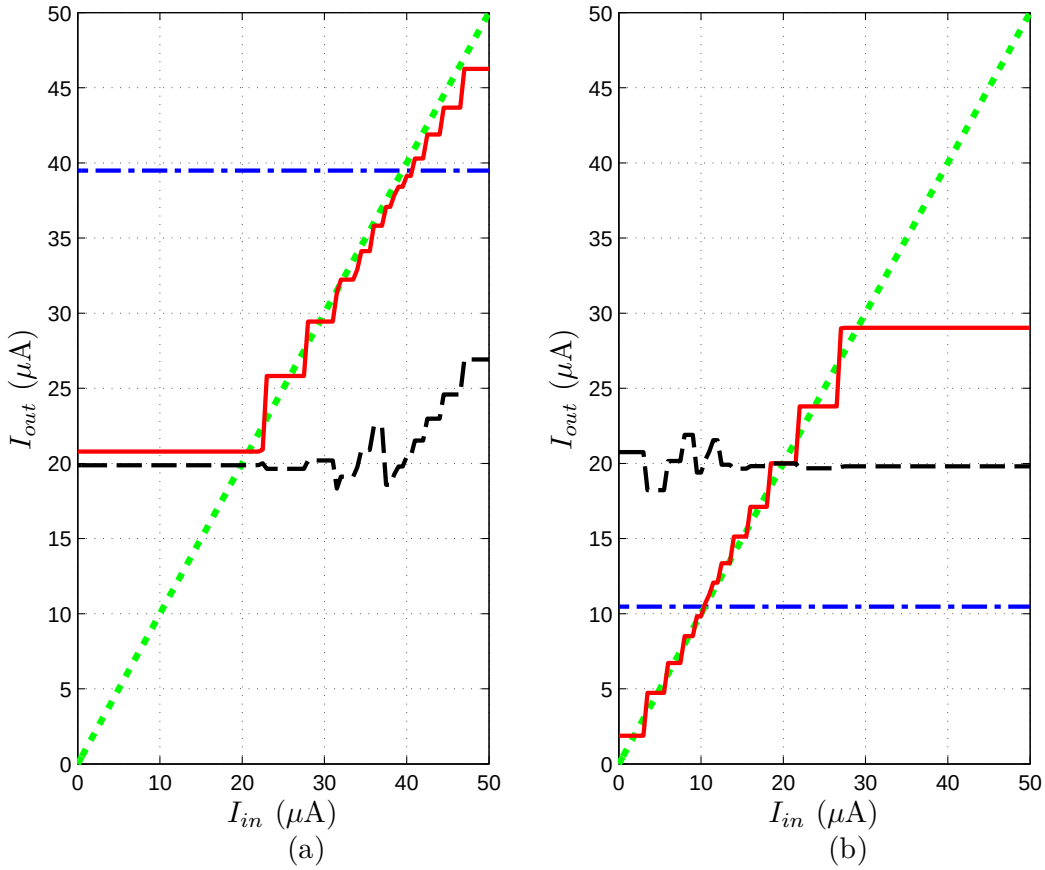


Figura 3.21: Simulação DC feita para um conjunto de blocos de DPCM conectados em série, onde as correntes são medidas no bloco n e no bloco $n+1$. O somatório de $y(n-1)$ é igual a (a) 40 μA e (b) 10 μA. A linha pontilhada é o somatório de $y(n)$. Na linha com traços e pontos, temos a corrente prevista $\hat{s}(n)$. Em linha cheia, temos a corrente reconstruída $\hat{s}(n+1)$ e em linha tracejada, temos a corrente reconstruída do bloco seguinte, $\hat{s}(n+2)$, quando o somatório de $y(n+1)$ é igual a 20 μA.

Pela Figura 3.21(a) podemos ver que, quando a corrente ultrapassa o seu valor máximo, o circuito do DPCM continua funcionando e, consequentemente, o valor

do bloco seguinte se afasta de 20 μA. Foi para evitar que isso aconteça que nós acrescentamos o circuito que limita a corrente e fizemos testes para limites de 40 μA e 44 μA. A Figura 3.22 mostra o resultado dessa simulação.

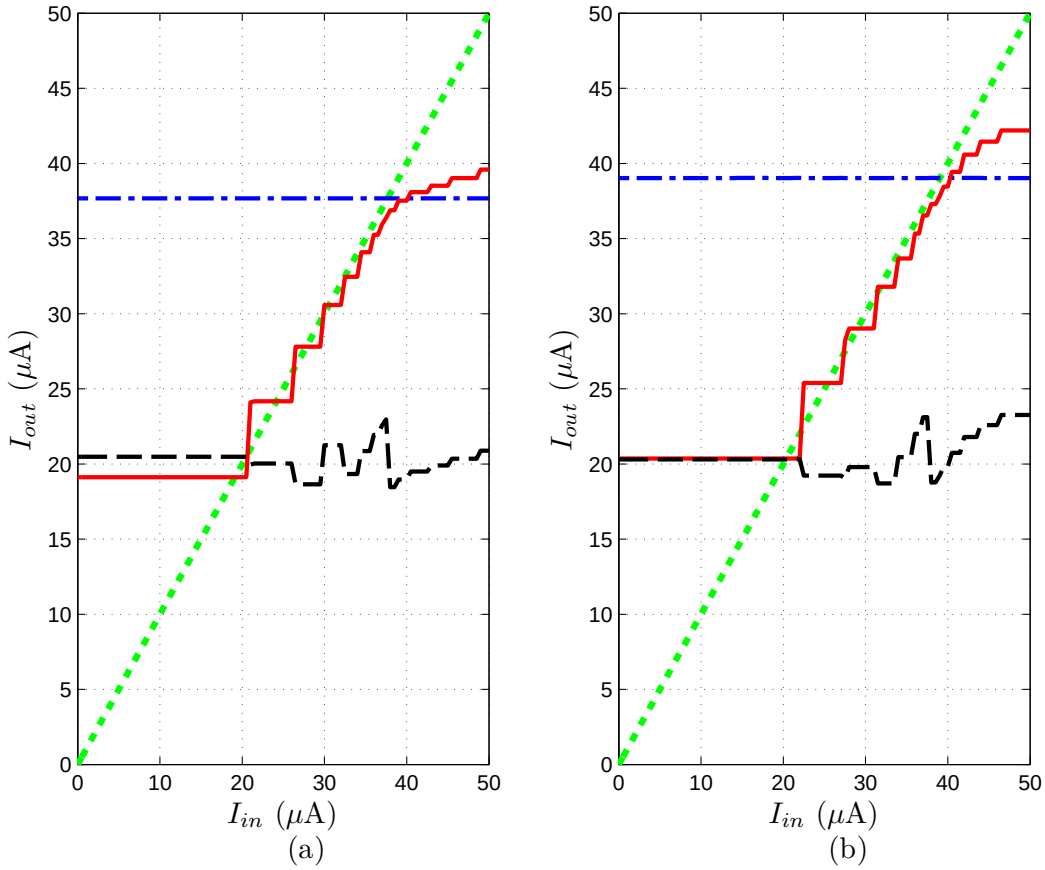


Figura 3.22: Simulação DC feita para um conjunto de blocos de DPCM conectados em série com o circuito para limitar a corrente ativado (a) em 40 μA e (b) em 44 μA. As correntes são medidas no bloco n e no bloco $n+1$. A linha pontilhada é o somatório de $y(n)$. Na linha com traços e pontos, temos a corrente prevista $\hat{s}(n)$. Em linha cheia, temos a corrente reconstruída $\hat{s}(n+1)$ e em linha tracejada, temos a corrente reconstruída do bloco seguinte, $\hat{s}(n+2)$, quando o somatório de $y(n+1)$ é igual a 20 μA.

Pelas duas figuras, percebemos que ligar o circuito que limita a corrente pode distorcer o sinal, o que é indesejado. No caso da Figura 3.22(a), a corrente prevista, $\hat{s}(n)$, sofre uma influência considerável desse circuito, mesmo não passando de 40 μA. Pela Figura 3.21(a) ela deveria ser igual a 39.5 μA, mas seu valor é 37.9 μA quando o circuito limitador de corrente é ligado para uma corrente de referência de 40 μA. Se aumentarmos um pouco esse limite, para 44 μA, percebemos que a distorção se torna menor, mas a corrente não fica limitada a 40 μA. Mesmo com esses resultados, optamos por manter esse circuito, uma vez que são necessários poucos transistores, para analisar o seu efeito em testes experimentais. A comparação entre as três correntes reconstruídas (circuito que limita a corrente desativado, com ele

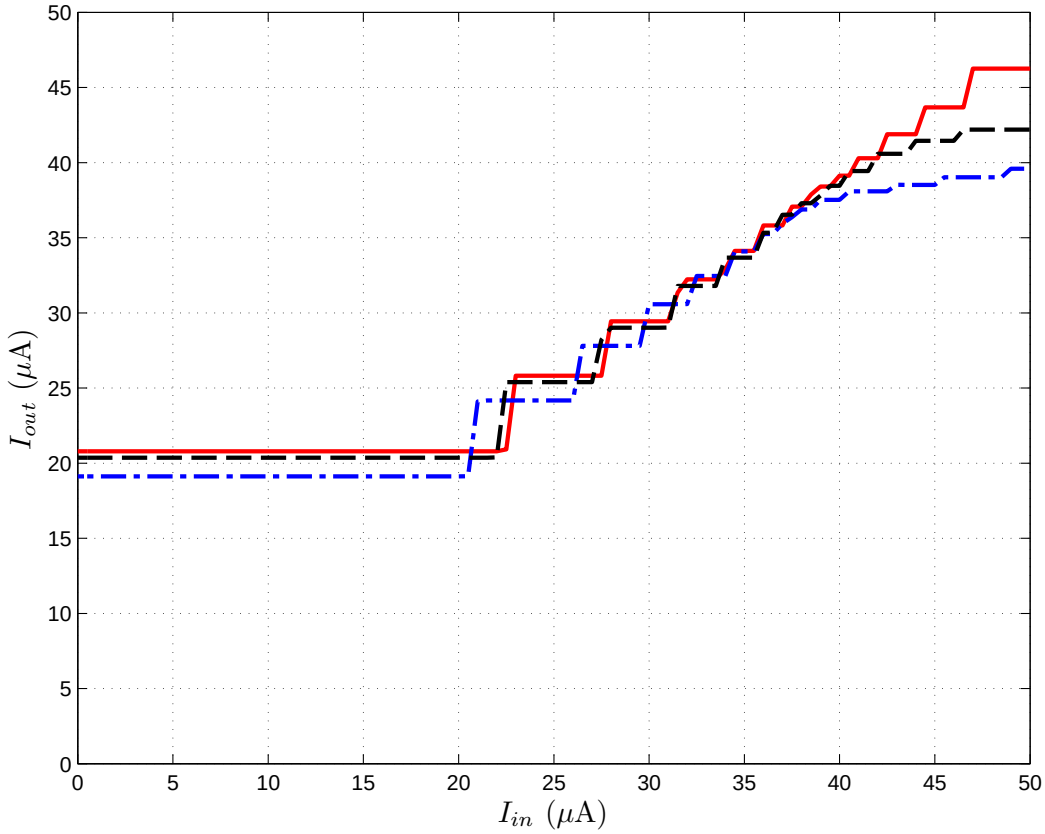


Figura 3.23: Comparação entre as correntes reconstruídas. Em linha cheia, temos $\hat{s}(n+1)$ gerado com o circuito que limita a corrente desligado. Em linha tracejada, temos $\hat{s}(n+1)$ gerado com o circuito que limita a corrente ligado para 44 μA e na linha com traços e pontos, temos $\hat{s}(n+1)$ gerado com o circuito que limita a corrente ligado para 40 μA.

ativado para 40 μA e com ele ativado para 44 μA) é apresentada no gráfico da Figura 3.23.

A simulação de Monte Carlo foi feita para os mesmos casos mostrados na Figura 3.21, onde ambos possuem o circuito para limitar a corrente desativado. O resultado é mostrado na Figura 3.24. A variação máxima de corrente é igual a 6.83 μA, mostrada na Figura 3.24(a), quando a corrente de entrada é igual a 50 μA. Para esse mesmo caso, quando a corrente de entrada é igual a 0 μA, a corrente de entrada varia 4.44 μA. No caso da Figura 3.24(b), quando a corrente de entrada é igual a 50 μA, a variação é de 1.90 μA e quando a corrente de entrada é igual a 0 μA, ela varia 2.78 μA. Como já foi explicado, um erro desse circuito pode causar uma propagação de erros no DPCM. No entanto, é difícil avaliar se a propagação de erros é perceptível com base nas simulações de Monte Carlo mostradas, pois nelas, vários blocos de DPCM estão ligados juntos e o circuito de reconstrução não é simulado de forma independente. Para identificar se a propagação de erros provoca erros perceptíveis, foram feitas algumas simulações com o circuito todo, mostradas na

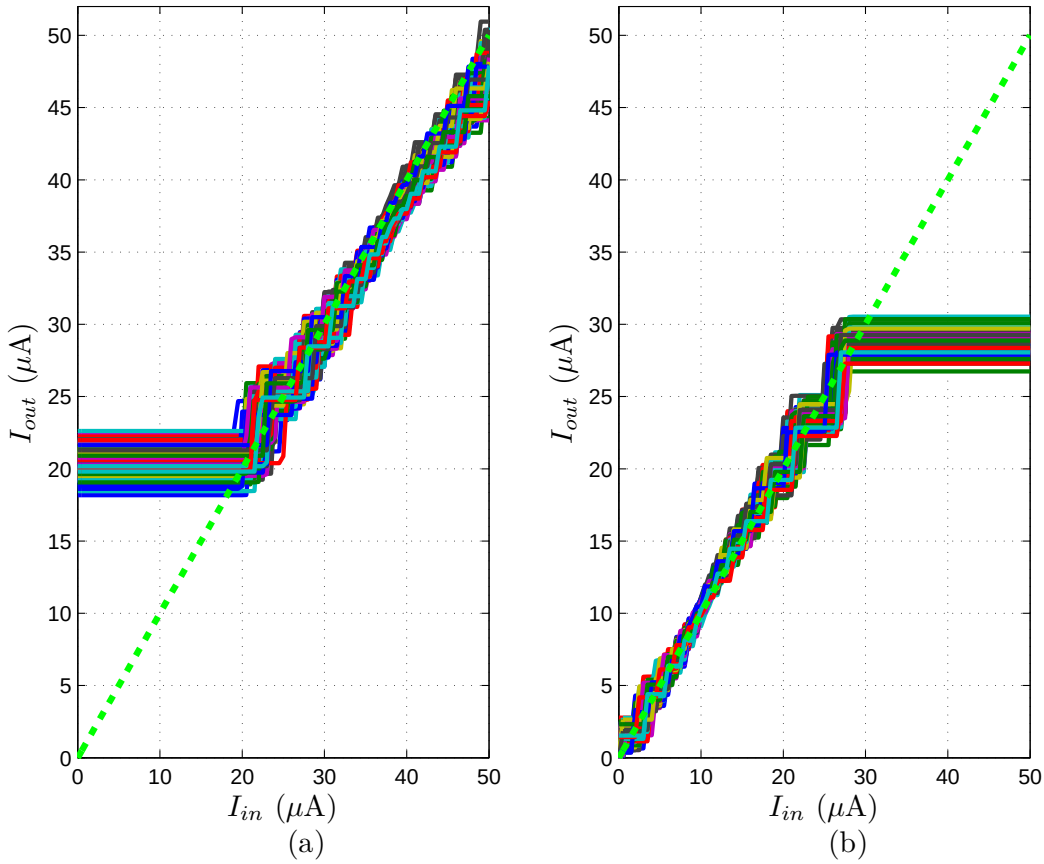


Figura 3.24: Repetições de Monte Carlo da análise DC variando a corrente de entrada (somatório de $y(n)$), em linha pontilhada, de 0 μA a 50 μA. Em linha cheia, Monte Carlo da corrente $\hat{s}(n+1)$, (a) quando a corrente $\hat{s}(n)$ é igual a 39.5 μA, e (b) quando a corrente $\hat{s}(n)$ é igual a 10.5 μA.

Seção 5.2. Além disso, por precaução, foram incluídos os três circuitos de DPCM com entrada zero ao final de cada linha de blocos, como foi explicado no início desse capítulo.

3.6 Bloco de 4 por 4 Pixels

Cada bloco de 4 × 4 pixels é formado pelos circuitos explicados nas seções anteriores. O diagrama de blocos da Figura 3.25 mostra a forma como esses circuitos são interconectados para formar o circuito que comprime cada bloco de 4 × 4 pixels. Os circuitos citados nessa figura serão implementados entre os fotodiodos da matriz de pixels.

O bloco funciona da seguinte forma: os 16 circuitos de leitura irão gerar tensões de referência associadas a correntes que devem ser copiadas para o circuito que implementa o produto interno da matriz **H** e para o circuito que implementa o somatório das correntes. O circuito da matriz **H** irá gerar cinco correntes de saída,

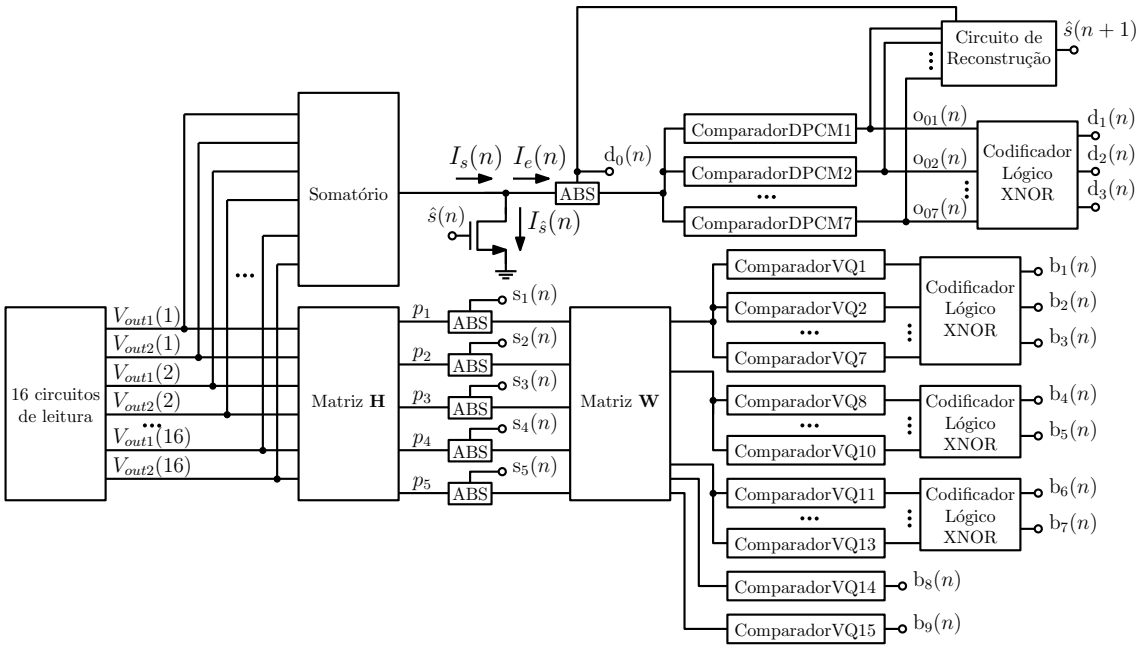


Figura 3.25: Diagrama de blocos do bloco de 4 × 4 pixels.

que representam as cinco componentes de maior energia do bloco. O circuito que realiza o somatório irá gerar uma corrente de saída. Essa corrente será subtraída da corrente referente a $\hat{s}(n)$ gerada pelo bloco anterior. A subtração é feita conectando o *drain* de um transistor que copia a corrente do bloco anterior ao mesmo nó que a corrente de saída do circuito que realiza o somatório. Essas seis correntes serão conectadas a seis diferentes circuitos de valor absoluto. Cada circuito de valor absoluto irá gerar um bit de sinal.

O vetor com os valores absolutos das cinco componentes deve ser multiplicado por **W**. Na saída de todos os circuitos de valor absoluto é feita a correção dos fatores de escala, onde as componentes um e dois devem ser divididas por dois, as componentes três e quatro devem ser multiplicadas por dois e a componente cinco deve ser multiplicada por quatro quintos. Apesar de não estar representado na Figura 3.25, cada circuito de valor absoluto conectado às componentes 1 a 4 irá gerar duas tensões de saída, uma equivalente ao módulo da componente e a segunda equivalente ao módulo multiplicado por -1. O circuito de valor absoluto conectado à componente 5 irá gerar somente uma tensão de saída. Essa tensão é equivalente ao módulo da componente. Nesse caso só é necessária essa tensão, pois a última coluna de **W** só possui valores positivos. A matriz **W** é implementada por linha através de espelhos de corrente simples. A primeira linha está conectada às saídas positivas dos cinco circuitos de valor absoluto das componentes. Essa linha será repetida sete vezes e cada saída será ligada a um comparador de corrente, pois o resultado do produto interno dessa linha deve ser comparado com sete limiares em modo

de corrente. Esses sete bits serão codificados por portas lógicas XNOR, resultado nos três primeiros bits do VQ. A segunda linha da matriz **W** está conectada às saídas positivas dos circuitos de valor absoluto das componentes 1, 3 e 5 e às saídas negativas dos circuitos de valor absoluto das componentes 2 e 4. Essa linha será repetida três vezes e ligada a 3 comparadores. Os bits resultantes são codificados resultando nos dois bits seguintes do VQ. A mesma ideia é repetidas para as linhas três, quatro e cinco da matriz **W**.

No caso do circuito de valor absoluto do DPCM, teremos somente uma tensão de saída, proporcional ao módulo da corrente de entrada. Essa corrente será copiada sete vezes e comparada com sete limiares. As sete tensões resultantes da comparação (sete bits) serão utilizadas para gerar, os outros três bits do DPCM (o bit de sinal já havia sido encontrado anteriormente), através de uma codificação com XNOR, e para o circuito de reconstrução. No circuito de reconstrução, os sete bits vão definir quais correntes de referência devem ser usadas para representar a corrente que foi codificada. A corrente resultante será somada à corrente gerada pelo bloco anterior através da conexão do *drain* de um transistor que copia a corrente do bloco anterior ao mesmo nó dessa corrente. O resultado do somatório é uma corrente que representa uma estimativa do somatório de todos os pixels do bloco atual. Essa estimativa deve ser transferida para o bloco seguinte. Para tal, essa corrente passa por um transistor tipo N com *drain* e *gate* conectados e *source* ligado a terra, isto é, meio espelho de corrente. A tensão resultante deve ser transferida para o bloco seguinte, onde, para realizar a cópia, basta conectarmos esta tensão ao *gate* de outro transistor tipo N que também possui *source* conectado a terra.

Dessa forma, encontramos os nove bits do VQ, os quatro bits do DPCM e os cinco bits de sinal que representam os pixels do bloco de uma imagem. As saídas do bloco serão bits que devem ser conectados a um registrador de deslocamento que estará ligado à saída do chip.

3.7 *Layout*

No processo de fabricação de um circuito integrado CMOS o *layout* é utilizado para gerar máscaras que representam as camadas do chip. Essas máscaras estarão presentes em todas as etapas do processo de fabricação [22]. Elas são utilizadas no processo de fotolitografia, que é uma técnica essencial para a fabricação do circuito integrado. A fotolitografia funciona da seguinte forma: o *wafer* de silício, tipo P, é coberto com uma substância sensível à luz ultravioleta, chamada de *photoresist*. O *wafer* é então aquecido, de forma que o *photoresist* se torna rígido. Uma máscara é colocada sobre o *wafer*, por onde são lançados feixes de luz ultravioleta. Somente a parte exposta pela máscara será atingida pela luz e o *photoresist* dessa região

que ficou exposta é retirado utilizando um agente revelador. O *photoresist* restante protege a área coberta. Dependo da máscara usada, a região onde não há *photoresist* pode ser usada para implantação iônica, sendo criados poços N ou P, crescimento de óxido de *gate*, depósito de silício policristalino ou de metal, dentre outros.

No processo, existe uma ordem definida para a utilização das máscaras. A primeira máscara utilizada, por exemplo, é aquela que define as regiões onde existem transistores tipo P. Como o *wafer* é tipo P, precisamos criar poços N na região onde os transistores P estão localizados, de forma que o substrato desses transistores seja tipo N. Utilizando implantação de átomos pentavalentes (como átomos do fósforo, o arsênio e o atimônio), que, ao atingirem o silício, liberam um elétron e viram íons positivos, criamos os poços necessários. A máscara seguinte é responsável pelas regiões do circuito que contêm óxido de *gate*, e assim por diante, até que todas as máscaras necessárias sejam utilizadas para a fabricação do chip. Com o desenvolvimento da técnica de fotolitografia houve um grande avanço na fabricação dos circuitos integrados e foi possível fabricar chips cada vez menores. Ainda assim, a fotolitografia possui restrições e o processo de fabricação acrescenta erros aos dispositivos. Por esse motivo, é importante realizar simulações de Monte Carlo, para garantir que o circuito estará dentro das especificações mesmo com os erros de fabricação. Durante o projeto do *layout*, devemos seguir determinadas regras definidas pela fábrica, como, por exemplo, a distância entre metais, distância entre componentes, largura mínima dos metais. Essas regras surgem devido às restrições das técnicas usadas durante a fabricação de um chip.

O *layout* foi projetado em uma ferramenta do *software* da Cadence, onde cada camada é representada por uma diferente cor e textura. Com essa ferramenta, é possível verificar se as regras de projeto estão sendo cumpridas, comparar o *layout* com o diagrama esquemático, para garantir que todas as conexões foram feitas, e extrair um circuito a partir do *layout*, o que permite uma simulação mais precisa do circuito.

3.7.1 Técnicas de Casamento de Componentes

Devido aos erros do processo de fabricação, os componentes do circuito sofrem variações em seus parâmetros e em suas dimensões. Não é possível garantir que um componente vai possuir as dimensões indicadas no diagrama esquemático depois de fabricado, mas podemos garantir que dois ou mais componentes irão variar da forma mais parecida possível. Ao fazermos isso, estamos "casando" (*matching*) esses componentes no *layout*. Para tal, são utilizadas regras que garantem que exista esse casamento. Nessa seção, iremos explicar as técnicas de casamento que foram utilizadas no projeto. Essas e algumas outras técnicas podem ser vistas em [23].

A primeira regra consiste simplesmente em manter os componentes que devem estar casados próximos. Podemos compreender essa regra se lembrarmos que fatores externos, como a temperatura, afetam o funcionamento dos dispositivos. Se dois componentes estiverem próximos, a diferença de temperatura entre eles será pequena. Logo, a variação dos parâmetros de forma geral será pequena. Não é possível fazer isso para todos os componentes da matriz, pois inevitavelmente, um componente do primeiro bloco estará distante de um componente do último bloco, uma vez que a matriz completa possui 64×64 pixels, isto é, 16×16 blocos. Nesse caso, tentamos compensar o descasamento com técnicas como o CDS. No entanto, dentro de um bloco de 4×4 pixels, mantivemos os transistores que realizam uma mesma operação em um determinado pixel próximos. Por exemplo, todos os transistores que realizam a leitura do fotodiodo e a coluna da matriz **H** que contém os multiplicadores aplicados a esse fotodiodo estão dispostos ao redor do componente fotossensível de 10×10 μm.

Outra regra simples e importante é a regra de manter a mesma orientação para todos os componentes que devem ser casados. O processo de fabricação possui um erro diferente para as direções vertical e horizontal. Portanto, no caso do transistor, a largura sofrerá uma variação diferente do comprimento, mas desejamos que todas as larguras dos componentes casados sofram a mesma variação. No circuito projetado, todos os transistores que não fazem parte de uma etapa digital, ou que são utilizados como chaves, estão dispostos com a largura na horizontal e o comprimento na vertical.

No caso dos espelhos de corrente utilizados no circuito, é importante escolher um elemento básico. Todos os transistores dos espelhos serão compostos de associações em paralelo desse componente. Essa técnica foi utilizada para a implementação das matrizes **H** e **W**, e do circuito que gera as correntes de referência para a matriz. Nos três casos, nós precisamos multiplicar a corrente de entrada por diferentes valores. Isso é feito escolhendo um elemento básico e mudando a multiplicidade desse elemento dependo da multiplicação que deve ser feita. Essa técnica é importante para o casamento, pois certos parâmetros do transistor, como o comprimento efetivo do canal, possuirão uma variação diferente se esse transistor for composto de diversos transistores múltiplos, ou se for composto somente por um transistor com canal mais largo. O casamento entre dois elementos com tamanhos iguais é melhor que o casamento entre dois elementos com tamanhos distintos.

As técnicas de interdigitação e centroide comum foram utilizadas juntas, no circuito que gera as correntes de referência. Um exemplo de interdigitação pode ser visto na Figura 3.26. No caso mostrado na figura, temos dois componentes que devem ser casados compostos por dois elementos básicos cada. A interdigitação consiste em alternar os pedaços desses elementos no *layout* de forma a compensar

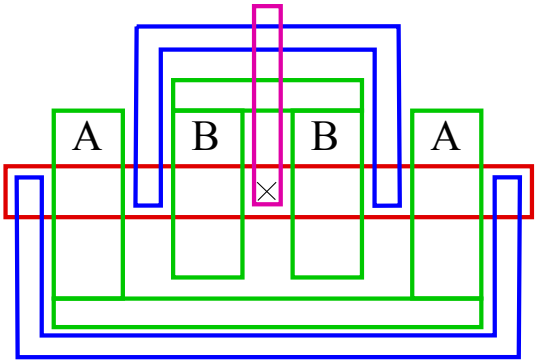


Figura 3.26: Exemplo de interdigitação. A cor verde indica silício policristalino (polisilício), o vermelho indica áreas de difusão n+ e o azul indica linhas de metal de nível 1 e o rosa linhas de metal de nível 2.

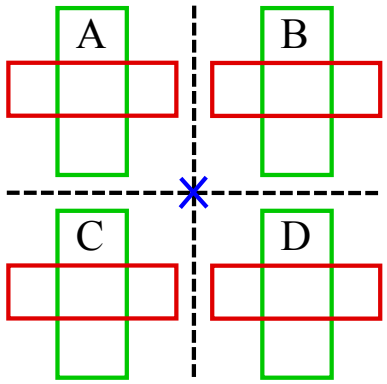


Figura 3.27: Exemplo da técnica de centroide comum.

os efeitos de gradiente do processo. No caso do exemplo da figura, os transistores também estão dispostos em uma organização com centroide comum. O layout em centroide comum tem como objetivo compensar os efeitos de gradiente linear térmico ou de processo. Essa técnica pode ser vista na Figura 3.27. Nessa técnica, os dispositivos são posicionados ao redor de um mesmo ponto central, marcado com um X na figura, de forma que os efeitos de gradiente fiquem distribuídos de forma aproximadamente igual. Para dispositivos compostos de diversos elementos básicos, também é importante seguir a regra do centroide comum. Para utilizar essas técnicas, normalmente chamamos cada componente que deve ser casado com uma letra diferente. No caso de componentes com multiplicidade, todas as partes do mesmo componente são denominadas com a mesma letra. Em seguida, organizamos uma matriz com todos os componentes, de forma que o centro da matriz seja o centroide comum, de preferência interdigitando os componentes. Na fabricação do circuito integrado, os transistores das bordas da matriz irão sofrer maior variação que os transistores do centro, devido a uma etapa chamada de *etching* [23]. Por esse motivo, é comum utilizar dedos de polisilício conectados a terra (*dummy fingers*) nas bordas da matrizes

de transistores. Essa técnica não foi utilizada no projeto, mas é uma técnica que temos interesse de utilizar em projetos futuros.

3.7.2 *Layout* de um Bloco

O *layout* da matriz de pixels foi feito para um bloco de 4 × 4 pixels. No *layout*, os circuitos que realizam a compressão da imagem ficam posicionados entre os elementos fotossensíveis, como pode ser visto na Figura 3.28(b). A distância entre os fotodiodos foi escolhida considerando o maior número de transistores de um mesmo multiplicador da matriz **W**, que é posicionada no centro do bloco. Os transistores dessa matriz foram posicionados de forma que ficassem próximos, na mesma orientação, e que o roteamento fosse facilitado utilizando o segundo nível de metal ("metal 2").

Os níveis de metal 1 e 2 foram utilizados internamente ao bloco, para a conexão entre os transistores do circuito. Podemos ver as ligações feitas por esses metais nas Figuras 3.28(c) e (d). Os níveis de metal 3 e 4 ligam o circuito aos sinais externos à matriz.

As correntes de referência e sinais de controle são inseridos na matriz utilizando as linhas de roteamento de metal 4, como pode ser visto na Figura 3.28(f). Os bits de saída do circuito, são extraídos do bloco utilizando metal 3, como mostrado na Figura 3.28(e).

A Figura 3.28(a) mostra o *layout* completo do bloco, exceto pelos metais AM e MT, que são os dois níveis de metal mais altos. Como sabemos que o funcionamento dos transistores pode ser afetado pela luz, esses metais foram utilizados para cobrir o circuito e assim proteger os transistores, quando o circuito for exposto à luz. Dessa forma, somente os fotodiodos serão sensíveis à luminosidade do ambiente. O bloco mede 108.8 μm × 108.8 μm. Dividindo por 4, cada pixel possui 27.2 μm × 27.2 μm. Como o fotodiodo utilizado mede 10 μm × 10 μm, o *fill factor* é igual a 7.1%.

3.7.3 *Layout* do Circuito que Gera as Correntes de Referência

Para gerar as correntes de referência necessárias para o circuito, é utilizada uma referência externa de 1 μA que será multiplicada e dividida por espelhos de corrente para gerar as 22 referências. Considerando somente os espelhos de corrente tipo N, são necessários um total de 616 transistores. Como são utilizados espelhos de corrente *cascode*, podemos considerar que o casamento deve ser feito em metade dos transistores, pois os transistores estarão ligados dois a dois. Sendo assim, podemos considerar que precisamos casar 308 transistores, onde, desses, temos 22 tipos diferentes (devido às multiplicidades).

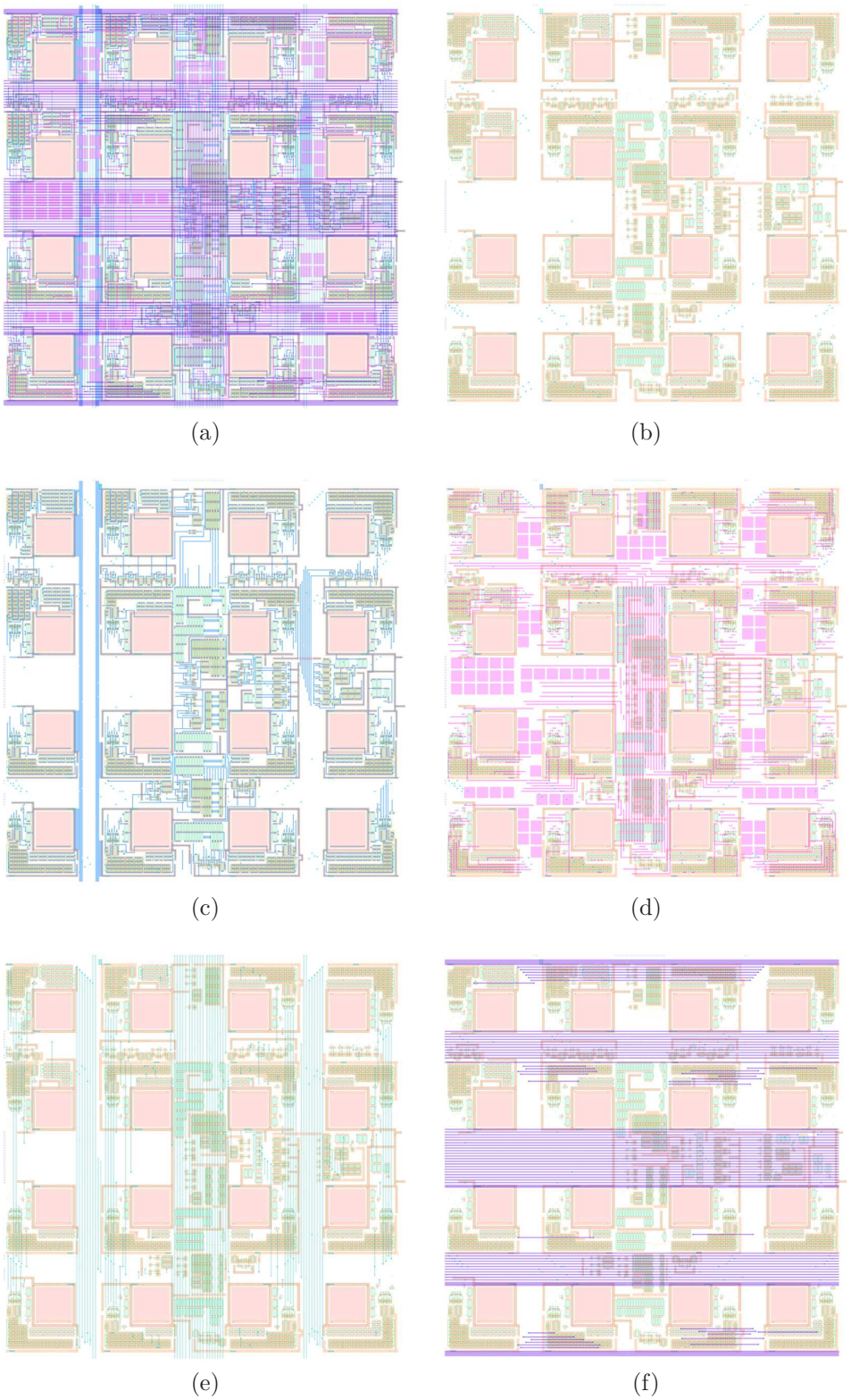


Figura 3.28: *Layout* de um bloco de 4 × 4 pixels. (a) *Layout* completo, com todos os níveis de metal utilizados para o roteamento do circuito, (b) sem nenhum metal, (c) somente com o nível de metal 1, (d) somente com o nível de metal 2, (e) somente com o nível de metal 3 e (f) somente com o nível de metal 4.

Os transistores foram colocados em uma matriz com 14 linhas e 22 colunas, e posicionados de forma que o centroide comum esteja aproximadamente no centro da matriz. A posição desses transistores é mostrada na Figura 3.29. Nessa matriz, cada letra representa um transitor diferente e o total de cópias dessa letra indica o valor da multiplicidade daquele transistor. O transistor responsável pela corrente de referência de 11 μA, por exemplo, possui multiplicidade 44 e é representado na matriz pela letra V.

$$\begin{bmatrix}
V & X & V & X & V & X & V & X & V & X & V & X & V & X & V & X & V & X & V & X & V & X \\
X & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S \\
R & V & R & V & R & V & R & V & R & V & R & V & R & V & R & V & R & V & R & V & R & V \\
P & P & T & K & T & K & T & H & P & I & P & I & P & I & H & T & G & T & G & T & P & P \\
L & P & K & L & K & L & K & J & B & J & L & J & J & H & L & G & L & G & L & G & P & S \\
X & O & N & O & N & O & N & O & N & D & E & F & D & N & O & N & O & N & O & N & O & X \\
X & X & T & T & T & G & T & G & T & F & C & A & E & T & M & T & M & T & G & T & X & X \\
X & X & T & M & T & M & T & M & T & E & C & C & F & T & G & T & G & T & T & T & X & X \\
X & O & N & O & N & O & N & O & N & D & F & E & D & N & O & N & O & N & O & N & O & X \\
S & P & G & L & G & L & G & L & H & J & J & I & J & B & J & K & L & K & L & K & P & L \\
P & P & T & G & T & G & T & H & I & P & I & P & I & P & H & T & K & T & K & T & P & P \\
V & R & V & R & V & R & V & R & V & R & V & R & V & R & V & R & V & R & V & R & V & R \\
S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & Q & S & X \\
X & V & X & V & X & V & X & V & X & V & X & V & X & V & X & V & X & V & X & V & X & V
\end{bmatrix}$$

Figura 3.29: Posicionamento de elementos básicos no *layout* do circuito que gera 22 correntes de referência para a matriz de pixels.

As demais matrizes desse circuito foram projetadas da mesma forma, considerando um centroide comum para os transistores que precisam ser casados. O *layout* resultante é mostrado na Figura 3.30. Esse circuito possui 86.5 μm $\times$ 428.7 μm.

3.7.4 *Layout* Completo

O *layout* completo do circuito é mostrado na Figura 3.31. A matriz de pixels é construída copiando o *layout* do bloco de 4 $\times$ 4 pixels para formar uma matriz de 16 $\times$ 16 blocos. Ao final de cada linha de blocos, acrescentamos três blocos de DPCM com entrada zero, para gerar os bits de correção do DPCM, conforme foi explicado no Capítulo 3. Três circuitos de teste (pixel com saída em modo de corrente, pixel com saída em modo de tensão e pixel 3T) foram acrescentados entre esses circuitos de correção.

Imediatamente acima da matriz, são incluídos os registradores de deslocamento necessários para transformar as saídas paralelas da matriz em uma saída serial. Imediatamente abaixo da matriz, estão posicionados os conversores A/D utilizados para a leitura dos pixels sem compressão. Nas laterais, incluímos os decodificadores e circuitos que geram as correntes de referência.

Ao redor da matriz, são colocados os *pads*, onde serão ligados os fios de ouro que fazem a conexão com o encapsulamento. Ligados aos *pads*, estão os circuitos para proteção eletrostática. A partir desse *layout*, foi gerado o arquivo de GDS enviado para a fabricação. O *layout* do circuito completo possui 2.8 mm $\times$ 2.8 mm.

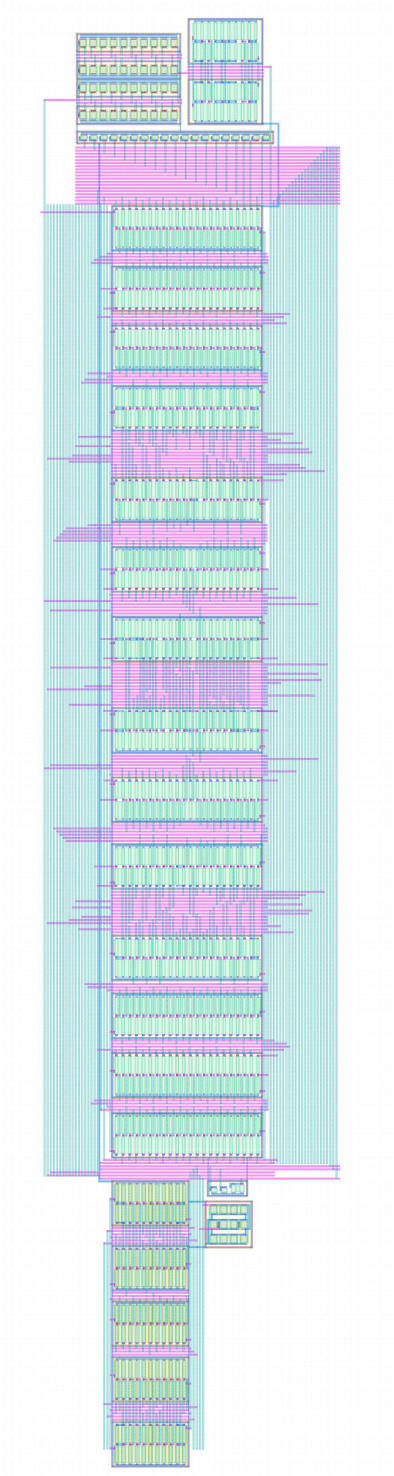


Figura 3.30: *Layout* do circuito que gera as correntes de referência.

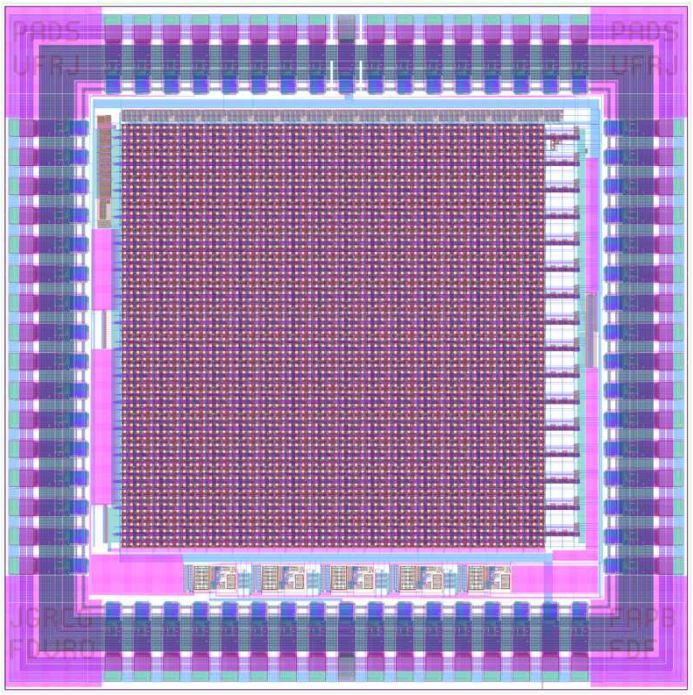


Figura 3.31: *Layout* completo do circuito integrado.

Capítulo 4

Decodificador

Para transformar os bits recebidos pelo chip em uma imagem, utilizamos um decodificador programado no MATLAB. Tanto na etapa de decodificação do DPCM quanto do VQ, fazemos uso de dicionários que foram calculados durante o projeto dos quantizadores, utilizando um banco de imagens de projeto. Utilizando os bits recebidos, devemos encontrar índices que serão utilizados para consultar esses dicionários.

Inicialmente, os bits recebidos do chip devem ser organizados e agrupados em três vetores: bits do DPCM (**D**), bits do VQ (**B**) e bits de sinal (**S**). Para gerar o vetor **S**, os bits de sinal recebidos pelo chip devem ser invertidos pois, no projeto do circuito do VQ, uma corrente drenada do circuito de valor absoluto é positiva e gera bit de sinal igual a 0. No decodificador, esses bits devem ser invertidos para que o sinal seja tratado como positivo. O inverso ocorre para as correntes injetadas. O decodificador é então dividido em duas etapas principais: gerar uma imagem que só contém as texturas, a partir dos vetores (**B**) e (**S**), e gerar uma imagem que só contém as médias dos blocos de 4×4 pixels, a partir do vetor (**D**).

O dicionário do VQ ($\mathbf{C}_{VQ}$) possui cinco linhas e 512 colunas. As cinco linhas são as cinco dimensões do VQ, referentes às componentes AC de maior energia da transformada. As colunas correspondem aos possíveis valores que as componentes podem assumir. Ao escolhermos uma coluna desse dicionário, estamos reconstruindo o vetor **x**. A coluna que deve ser utilizada é escolhida através de um índice encontrado com os nove bits do VQ.

No entanto, os bits do VQ são codificados utilizando operações lógicas XNOR, como mencionado no Capítulo 2. Sendo assim, realizamos a operação inversa mapeando os bits do vetor (**B**) em valores decimais de acordo com a célula em que o vetor de texturas se encontra. Os três primeiros bits são mapeados em um decimal que varia de zero a sete e indica quais limiares da primeira dimensão do VQ foram cruzados. Os dois bits seguintes são mapeados em um decimal referente à segunda dimensão do VQ. Os bits seis e sete indicam os limiares da terceira dimensão e o

oitavo e o nono bits possuem um mapeamento direto e indicam os limiares cruzados na quarta e quinta dimensões, respectivamente. Utilizando os valores encontrados, realizamos uma concatenação dos decimais, escritos em binário, para encontrar o valor da coluna do dicionário que deve ser utilizada.

Uma vez encontrada a coluna desejada, utilizamos o vetor $\mathbf{S}$ para definir o sinal de cada componente do vetor $\hat{\mathbf{p}}$ (reconstrução do vetor $\mathbf{p}$). Devemos então ajustar os fatores de escala de cada componente. As linhas um e dois do vetor $\hat{\mathbf{p}}$ serão multiplicadas por oito, as linhas três e quatro por dois e a linha cinco por cinco.

Como a multiplicação da matriz $\mathbf{H}$ pela sua transposta não é igual à matriz identidade, para encontrar o vetor de texturas reconstruído devemos multiplicar $\hat{\mathbf{p}}$ pela matriz $\mathbf{H}$ transposta e por uma matriz diagonal dada por: $(\mathbf{H}\mathbf{H}^T)^{-1}$. Dessa forma, encontramos o bloco (1,1) de uma matriz com 16 $\times$ 16 blocos, pois o chip possui 64 $\times$ 64 pixels. Realizando esse procedimento para os bits de VQ dos demais blocos, encontramos a matriz de texturas $\mathbf{Y}$.

No caso do DPCM, o primeiro bit do vetor $\mathbf{D}$ é utilizado como sinal, e os três últimos bits desse vetor são utilizados para encontrar o índice do dicionário $\mathbf{C}_{DPCM}$, um vetor com uma linha e oito colunas. No DPCM, o valor codificado é a diferença, isto é, o erro entre o somatório dos valores dos pixels e um valor estimado para somatório. Sendo assim, o dicionário do DPCM contém oito possíveis valores para o erro reconstruído. Como, exceto pelo bit de sinal, os bits do DPCM também foram codificados utilizando portas lógicas XNOR, para encontrar o índice do dicionário, realizamos a operação inversa, transformando o código *gray* utilizado em índices que indicam qual posição do dicionário do DPCM deve ser utilizada. Dessa forma, encontramos todos os índices que serão utilizados na decodificação do DPCM.

A reconstrução do DPCM é dada segundo a seguinte equação:

$$\hat{m}(n) = \hat{m}(n-1) + \hat{e}(n). \tag{4.1}$$

Apesar de codificarmos a diferença entre os somatórios dos blocos, e não a diferença entre as médias, ajustamos os dicionários e os limiares para encontrar as médias diretamente durante a reconstrução. Pela Equação (4.1), precisamos do valor médio do bloco anterior para encontrar o valor médio do bloco atual. No caso do primeiro bloco, foi definido no codificador que valor médio utilizado como referência é igual ao centroide do dicionário que mais se aproxima de 0.5, considerando que o valor de um pixel varia de zero a um. No caso, o centóide escolhido vale 0.4688. Em todo início de linha de blocos, somamos 0.4688 ao erro estimado multiplicado pelo seu sinal (-1 para bit 0 e 1 para bit 1). O erro estimado é dado pelo valor do dicionário $\mathbf{C}_{DPCM}$, que reconstrói a diferença entre a média do primeiro bloco e

0.4688. Dessa forma, encontramos a média reconstruída do primeiro bloco, que deve ser guardada na posição (1,1) da matriz **Im**. Esse valor também será utilizado para encontrar a média do bloco seguinte. Até o final da linha (posição (1,16) da matriz **Im**), utilizamos o valor encontrado na coluna anterior para encontrar o valor da coluna atual. Ao passarmos para outra linha repetimos o procedimento, utilizando o centroide 0.4688 como estimativa da média do primeiro bloco. Ao final da execução do decodificador, a matriz **Im** possuirá as médias de todos os blocos da imagem.

Os resultados gerados pelo DPCM e pelo VQ são duas imagens separadas, que podem utilizadas para comparar separadamente cada etapa da compressão com um resultado ideal gerado pelo MATLAB, como será visto na Capítulo 5. Para encontrar a imagem final, basta somar a matriz **Im** à matriz **Y**. O resultado do VQ, no entanto, possui valores positivos ou negativos, por isso é mais interessante apresentá-lo somando a matriz **Y** a uma matriz do mesmo tamanho com todos os elementos com valor 0.5, possibilitando uma melhor visualização das texturas.

Após o cálculo de **Im**+**Y**, podemos fazer alguns ajustes, como correção do erro quadrático gerado pelo circuito de leitura, ajuste de faixa dinâmica, aplicação de uma tangente hiperbólica para aumentar o contraste, dentre outros. No caso das imagens mostradas no Capítulo 5, nenhum dos ajustes foi necessário. No entanto, devido ao processo de fabricação, esses ajustes podem ser necessários durante os testes experimentais. Para avaliar numericamente a qualidade da imagem, realizamos o cálculo da PSNR entre a imagem resultante da compressão (**Im**+**Y**) e a imagem original.

Capítulo 5

Resultados de Simulação

Para avaliar o funcionamento de todos os circuitos descritos no Capítulo 3 interligados, geramos estímulos de entrada para os circuitos de leitura dos fotodiodos proporcionais aos de valores de pixels em um pedaço da imagem Lena. Ao decodificar os bits de saída de um bloco, temos uma imagem que foi comprimida pelo circuito. Podemos comparar a compressão feita pelo MATLAB com a compressão feita pelo circuito e avaliar as perdas referentes ao *hardware*.

A simulação do circuito integrado completo (matriz de 64 × 64 pixels) é muito custosa, pois o circuito possui uma grande quantidade de componentes e de nós. Por esse motivo, as simulações são feitas para uma linha de blocos por vez e os circuitos digitais periféricos, como decodificador e registrador de deslocamento, não foram acrescentados. Pelo mesmo motivo, a maioria das simulações, como por exemplo simulações de Monte Carlo e do circuito extraído, foi feita para imagens de 32 × 32 pixels. Somente duas simulações nominais, apresentadas no fim do capítulo, foram feitas para imagens com 64 × 64 pixels.

Para realizar a simulação, o *netlist* de uma linha de blocos é gerado pelo Cadence através de um diagrama esquemático em que todas as fontes de corrente de entrada (fontes que simulam a descarga do fotodiodo por corrente fotogerada) possuem valor máximo (20 pA) e estão ligadas a nós com nome "IinM_N". Nesse nome, M é o número do bloco e N o número do pixel no bloco. Um código do MATLAB irá transformar cada pixel de uma imagem de 32 × 32 pixels ou de 64 × 64 pixels, que varia de 0 a 1, em um valor de corrente que varia de 0 pA a 20 pA de forma diretamente proporcional ao valor do pixel. Em seguida esse mesmo código cria uma parte do *netlist* somente com as fontes de corrente ligadas aos nós de entrada. Substituindo as fontes de corrente do *netlist* original pelas geradas pelo MATLAB, podemos realizar simulações através de linha de comando cujos resultados serão as imagens comprimidas pelo circuito esquemático.

5.1 Simulação Nominal com 32 por 32 Pixels

A Figura 5.1 mostra o resultado da compressão feita no MATLAB comparada com o Cadence. Como era esperado, devido a não-linearidade do circuito de leitura, o resultado do Cadence possui menor intensidade que o resultado do MATLAB. A PSNR da imagem mostrada na Figura 5.1(a) é igual a 29.9 dB (Conforme a Figura 2.2(d)) e da Figura 5.1(d) é igual a 15.6 dB. Mesmo com essa queda significativa da PSNR, percebemos visualmente, de forma subjetiva, que o resultado do circuito se assemelha ao resultado ideal. Como a imagem simulada é mais escura que a imagem codificada pelo MATLAB, é esperado que a PSNR seja menor.

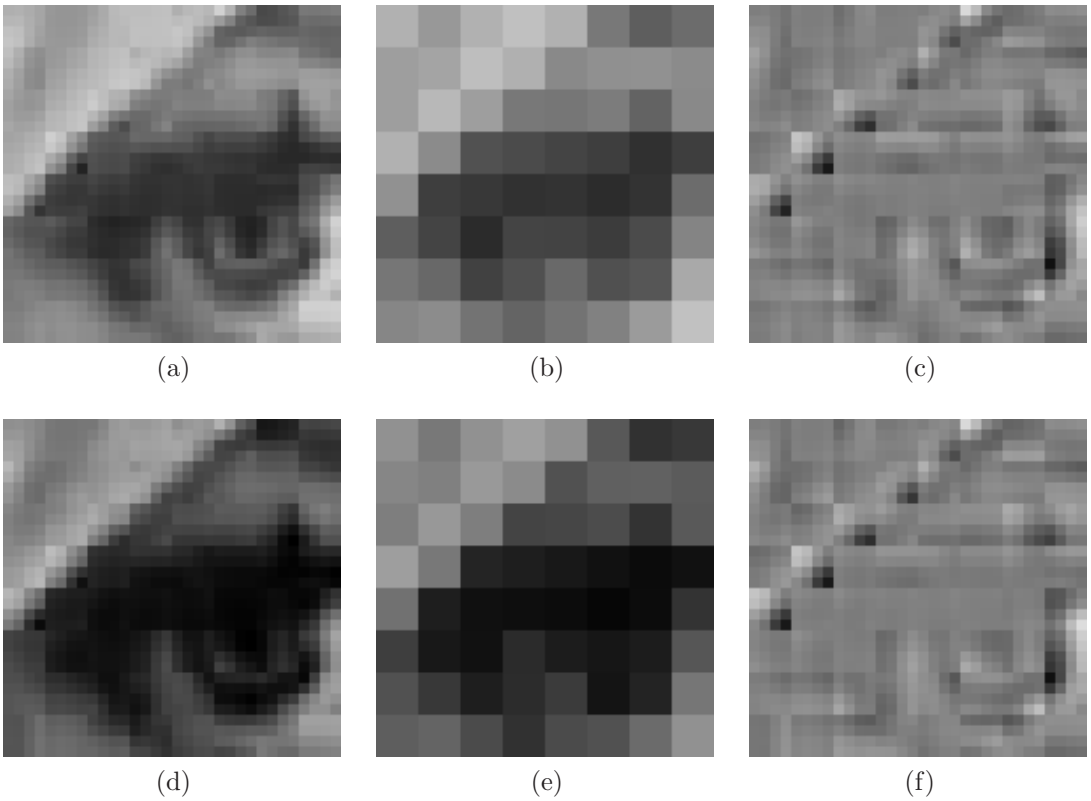


Figura 5.1: (a) Olho da Lena comprimido pelo MATLAB, (b) resultado do DPCM do MATLAB, (c) resultado do VQ do MATLAB, (d) olho da Lena comprimido pelo Cadence através de uma simulação nominal, (e) resultado do DPCM da simulação nominal do Cadence, e (f) resultado do VQ da simulação nominal do Cadence.

Para avaliar a queda de PSNR causada pela não-linearidade do circuito de leitura, foi acrescentada uma distorção quadrática à imagem do olho da Lena e essa nova imagem foi comprimida utilizando o MATLAB. O resultado é mostrado na Figura 5.2. Como esperado, o resultado fica mais próximo do resultado do circuito. A PSNR da Figura 5.2(a) é igual a 17.0 dB. Concluímos que grande parcela da queda da PSNR do resultado do MATLAB quando comparado ao resultado do Cadence é devido à distorção quadrática do circuito de leitura. Na Seção 5.4, vamos mostrar

um resultado em que a PSNR aumenta consideravelmente, chegando a 25 dB, ao aplicarmos uma equação quadrática inversa. De onde também podemos concluir que as baixas PSNRs são devido à distorção do circuito de leitura.

(a) (b) (c)

Figura 5.2: Resultado da compressão no MATLAB com distorção quadrática: (a) VQ e DPCM, (b) DPCM e (c) VQ.

5.2 Simulação de Monte Carlo

Devido ao tempo de simulação, foram feitas somente oito rodadas de Monte Carlo, considerando variações de processo e *mismatch*, para cada linha de blocos. A Figura 5.3 mostra os resultados dessa simulação: imagem completa, somente o DPCM e somente o VQ. Apesar das imagens aparecerem mais ruidosas que no caso da simulação nominal, o resultado é aceitável, pois as imagens estão muito próximas, o que demonstra que o circuito é robusto, e o olho da Lena é bem representado. A primeira, a terceira e a quinta linha da figura mostram os resultados das rodadas 1 a 4, da esquerda para a direita. Já as linhas dois, quatro e seis montram as rodadas 5 a 8, também da esquerda para a direita. Assim como nos casos das simulações anteriores, o resultado do VQ é apresentado somando 0.5 a cada pixels. Sendo assim, para encontrar a segunda imagem da primeira linha, basta somar a segunda imagem da terceira linha com a segunda imagem da quinta linha subtraída de 0.5.

As PSNRs encontradas, da primeira até a oitava rodada, são as seguintes: 13.53 dB, 13.88 dB, 13.69 dB , 15.72 dB, 13.31 dB, 14.23 dB, 13.14 dB e 19.05 dB. Como era esperado, devido à distorção causada pelo circuito de leitura e às incertezas adicionadas pelo método de Monte Carlo, os valores são baixos. A última rodada apresenta uma PSNR maior, pois a imagem é um pouco mais clara que as demais. Apesar do resultado numérico não ser muito bom, o que era esperado, devido à distorção quadrática, subjetivamente, as imagens estão satisfatórias.

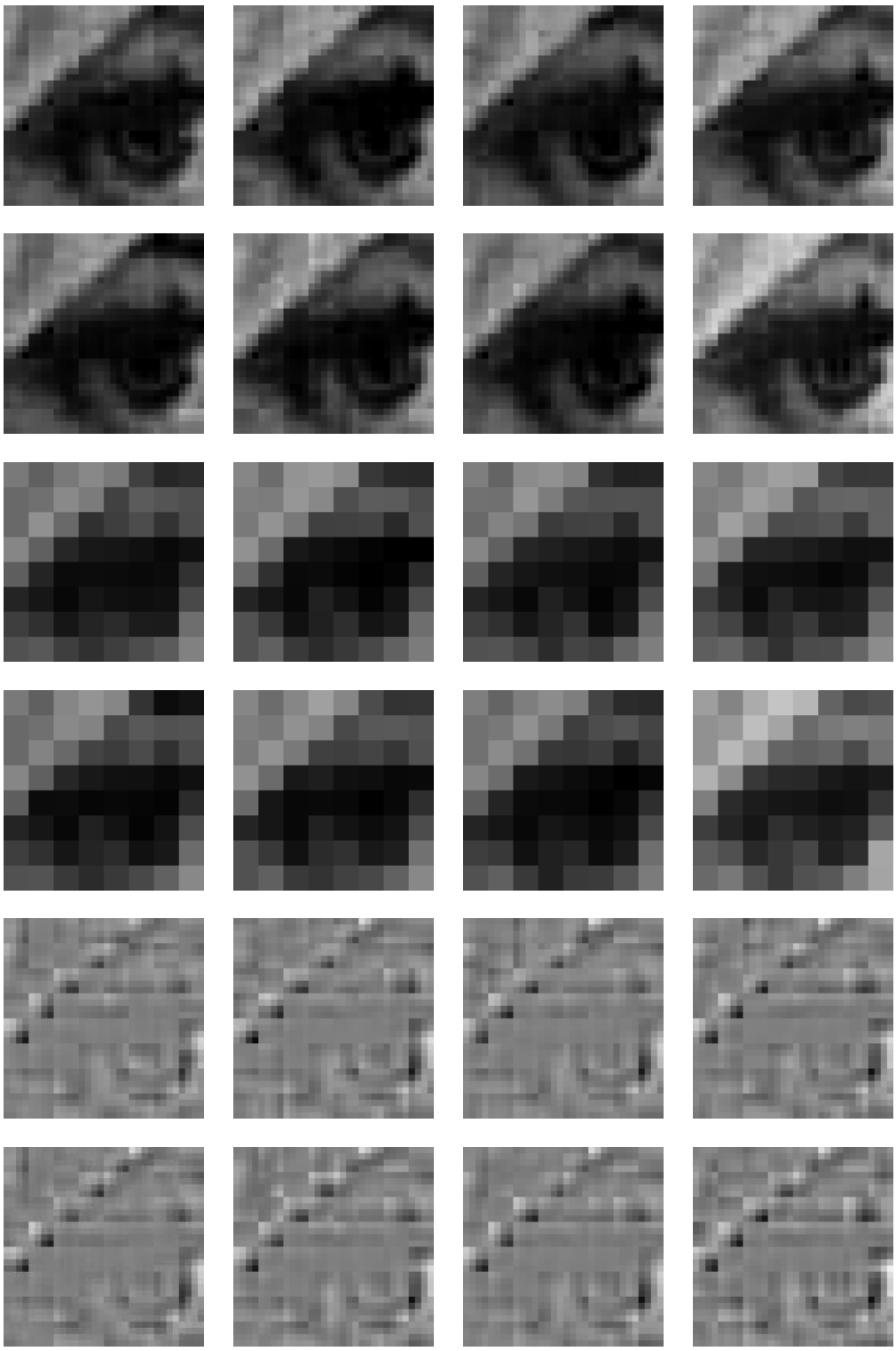
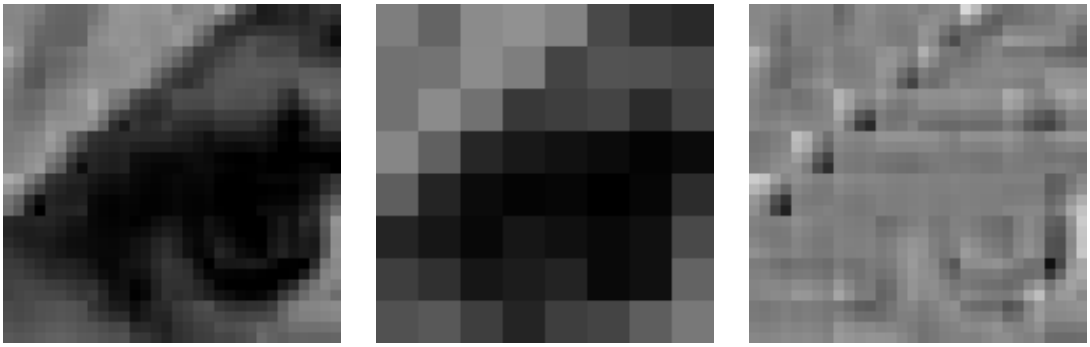


Figura 5.3: Oito rodadas de simulação de Monte Carlo do circuito esquemático para uma imagem de 32 × 32 pixels. As primeiras duas linhas mostram as imagens com DPCM e VQ, a terceira e a quarta linha mostram o resultado do DPCM, e as duas últimas linhas os resultados do VQ.

5.3 Simulação do Circuito Extraído

Como a simulação do circuito completo extraído exigiria muito tempo, os resultados da Figura 5.4 foram encontrados com o circuito extraído do bloco de 4 × 4 pixels. Cada bloco do diagrama esquemático foi substituído pelo seu equivalente extraído e a simulação foi feita da mesma forma que a simulação nominal do circuito esquamático: conectamos à entrada dos blocos capacitores de 5 fF em paralelo com fontes de corrente proporcionais aos valores dos pixels nas imagens, e simulamos por linha de blocos.



(a) (b) (c)

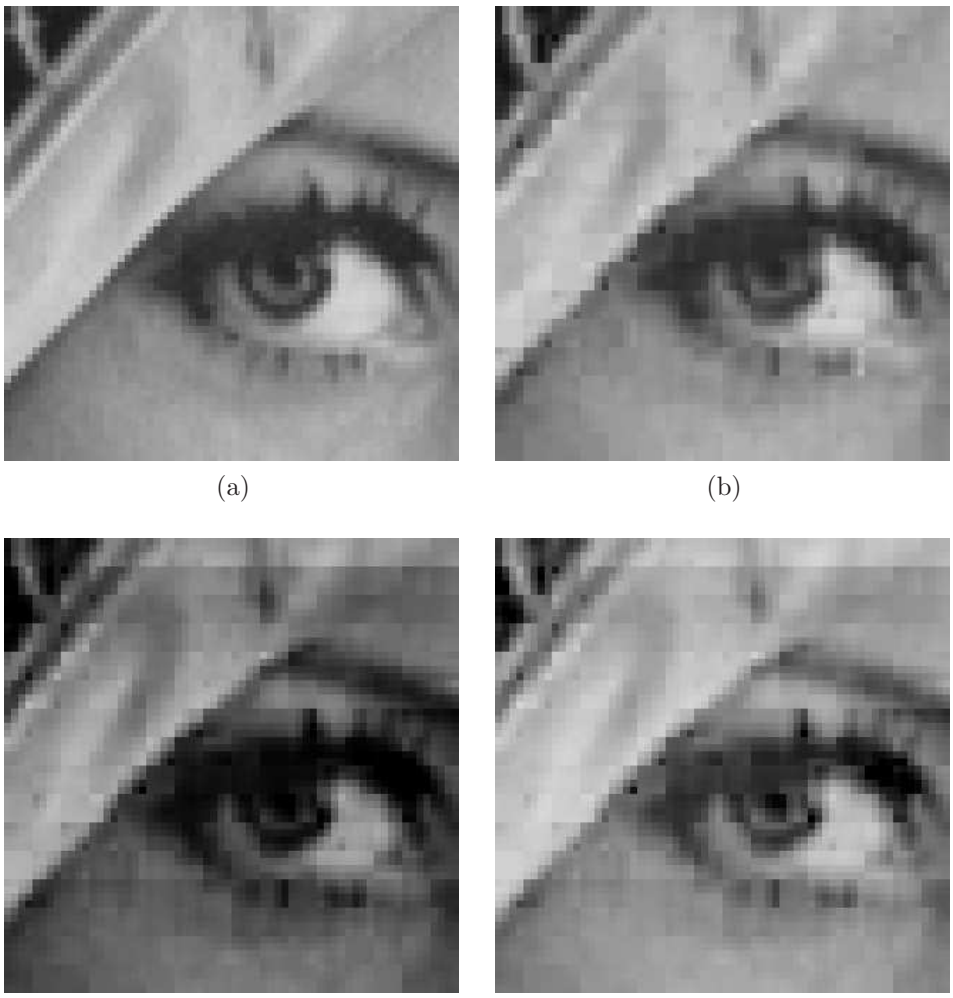
Figura 5.4: Simulação do circuito extraído: (a) VQ e DPCM, (b) DPCM e (c) VQ.

Subjetivamente, o resultado encontrado é muito bom, e se aproxima consideravelmente do resultado da simulação nominal do esquemático. Isso significa que os componentes parasitas gerados pelo *layout* têm pouca influência no resultado final. No entanto a imagem é mais escura que a imagem mostrada na Figura 5.1, o que deixa a PSNR mais baixa, igual a 13.37 dB.

5.4 Simulação Nominal com 64 por 64 Pixels

A imagem utilizada para a simulação com 64 × 64 pixels é apresentada na Figura 5.5(a). O resultado da compressão no MATLAB é apresentado na Figura 5.5(b), com PSNR igual a 30.1 dB. A simulação utilizando o circuito no Cadence obteve como resultado a Figura 5.5(c), que possui PSNR igual a 14.3 dB.

Nas Figuras 5.5(c) e (d), percebemos que a primeira linha de blocos aparece mais clara do que deveria ser. Acreditamos que isso acontece devido a um erro numérico de simulação, que tem como consequência uma saída com bits que oscilam. Isso ocorre pois a média de um bloco dessa linha é muito próxima de um limiar. Sendo assim, uma pequena variação, provavelmente causada por erro numérico, é suficiente para que um dos bits de saída do DPCM oscile. Como os blocos do DPCM são conectados em cascata, esse erro será propagado para todos os blocos seguintes. Se



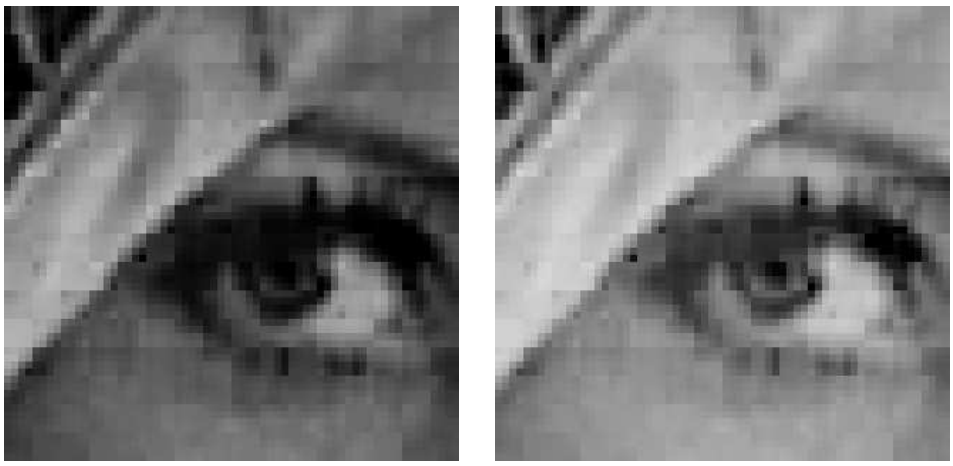
(a) (b)

(c) (d)

Figura 5.5: Resultados da simulação utilizando um pedaço com 64 × 64 pixels da imagem Lena: (a) original, (b) simulação numérica, no MATLAB, (c) simulação a nível de circuitos, no Cadence, e (d) mesma simulação a nível de circuitos com correção da distorção quadrática.

realizarmos a amostragem dos bits da primeira linha de blocos em um instante de tempo diferente, temos o resultado mostrado na Figura 5.6(a), em que a primeira linha de blocos parece semelhante às linhas seguintes.

A Figura 5.7 mostra o valor médio de cada bloco da primeira linha de blocos das Figuras 5.5(a), (c) e 5.6(a). Como esperado, devido ao circuito de leitura, a média dos blocos gerada pelo circuito é mais baixa que a média gerada pelo MATLAB. Comparando os três casos, percebemos que o primeiro erro que pode ter causado a linha mais clara ocorre no terceiro bloco, quando o valor indicado por um ponto vazado sobe, ao invés de descer. Outro erro dessa natureza ocorre no oitavo bloco. Nos dois casos, percebemos que o erro se propaga, pois os pontos vazados permanecem com valores mais altos que os 'x', que são os valores das médias na segunda amostragem dos bits da primeira linha de blocos. Esses pontos também



(a) (b)

Figura 5.6: Resultado da simulação com 64×64 pixels da imagem Lena amostrando a primeira linha de blocos em um instante diferente: (a) sem a correção da distroção quadrática e (b) com correção.

possuem erros, como no caso do quinto bloco, em que o passo dado é maior do que deveria ser. Ainda assim, esse erro não parece tão crítico, visualmente, quanto o erro da primeira amostragem.

Apesar do erro provocado pela instabilidade numérica na simulação e da baixa PSNR, a qualidade subjetiva da imagem é boa. Para avaliação de qualidade subjetiva, costuma-se utilizar também a medida SSIM (*structural similarity*) [24]. As avaliações de SSIM que foram feitas neste trabalho estão todas de acordo com as avaliações de PSNR e estão documentadas no Apêndice A. No caso das Figuras 5.5(c) ou 5.6(a), a baixa PSNR pode ainda ser associada a uma boa qualidade visual porque, apesar da SSIM menor que as demais medidas de SSIM, ainda se trata de uma SSIM razoável em termos absolutos (acima de 0.7). Como nós sabemos que existe uma distorção quadrática causada pelo circuito de leitura, e que ela é em grande parte responsável pela baixa PSNR, com o objetivo de aumentar essa PSNR, foi aplicada à imagem da Figura 5.5(c) uma função quadrática inversa. Para encontrar essa função utilizamos a regra dos mínimos quadrados para encontrar a melhor curva que se ajusta à nuvem de pontos mostrada no gráfico mostrado na Figura 5.8. Nessa figura mostramos todos os valores dos pixels da Figura 5.5(c) em função dos valores dos pixels da Figura 5.5(b). Esse gráfico mostra no eixo das abscissas o valor que nós deveríamos ter encontrado na simulação, e no eixo das ordenadas o valor que efetivamente foi encontrado pelo circuito. A curva ajustada é mostrada em linha cheia.

A Equação (5.1) mostra a função encontrada utilizando mínimos quadrados. Para encontrarmos a função que deve ser aplicada aos pixels da Figura 5.5(c) de forma a corrigí-los, devemos inverter a Equação (5.1). O resultado é mostrado na

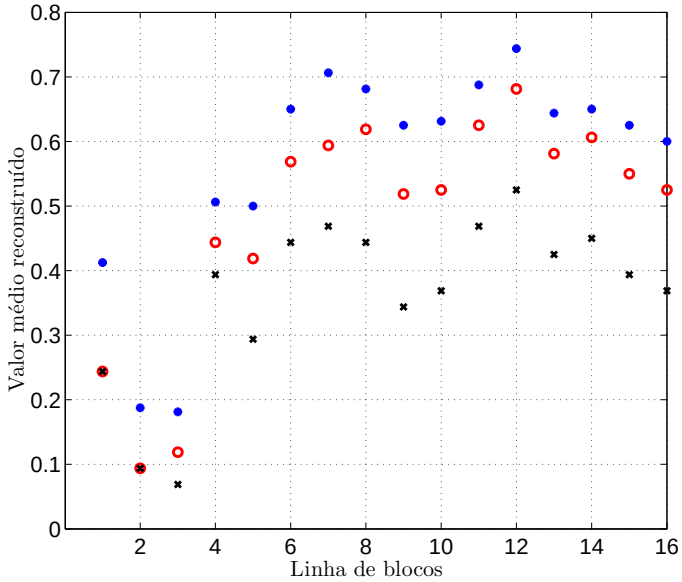


Figura 5.7: Valor médio reconstruído de cada bloco da primeira linha de blocos nas seguintes situações: simulação do MATLAB, mostrada com o ponto cheio; simulação do circuito esquemático em que a primeira linha de blocos fica mais clara, mostrada com o ponto vazado; e simulação do circuito esquemático em que a primeira linha de blocos fica um pouco mais escura, indicada em 'x'.

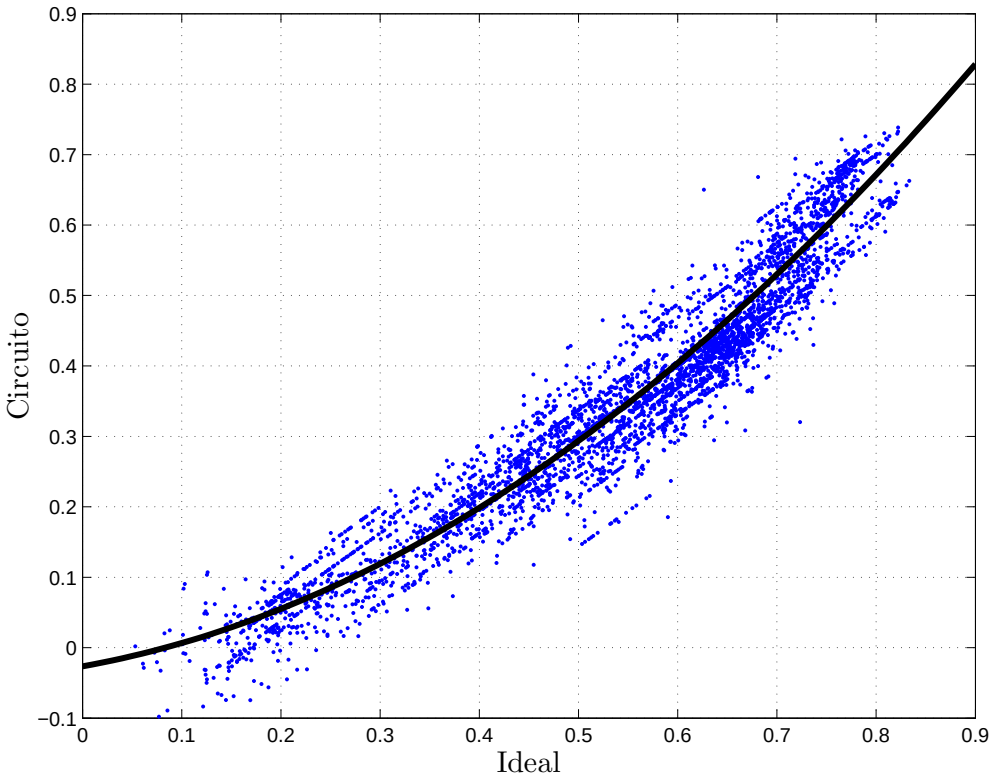


Figura 5.8: Curva de ajuste quadrático entre pixels reconstruídos a partir do circuito e pixels reconstruídos no MATLAB.

Equação (5.2).

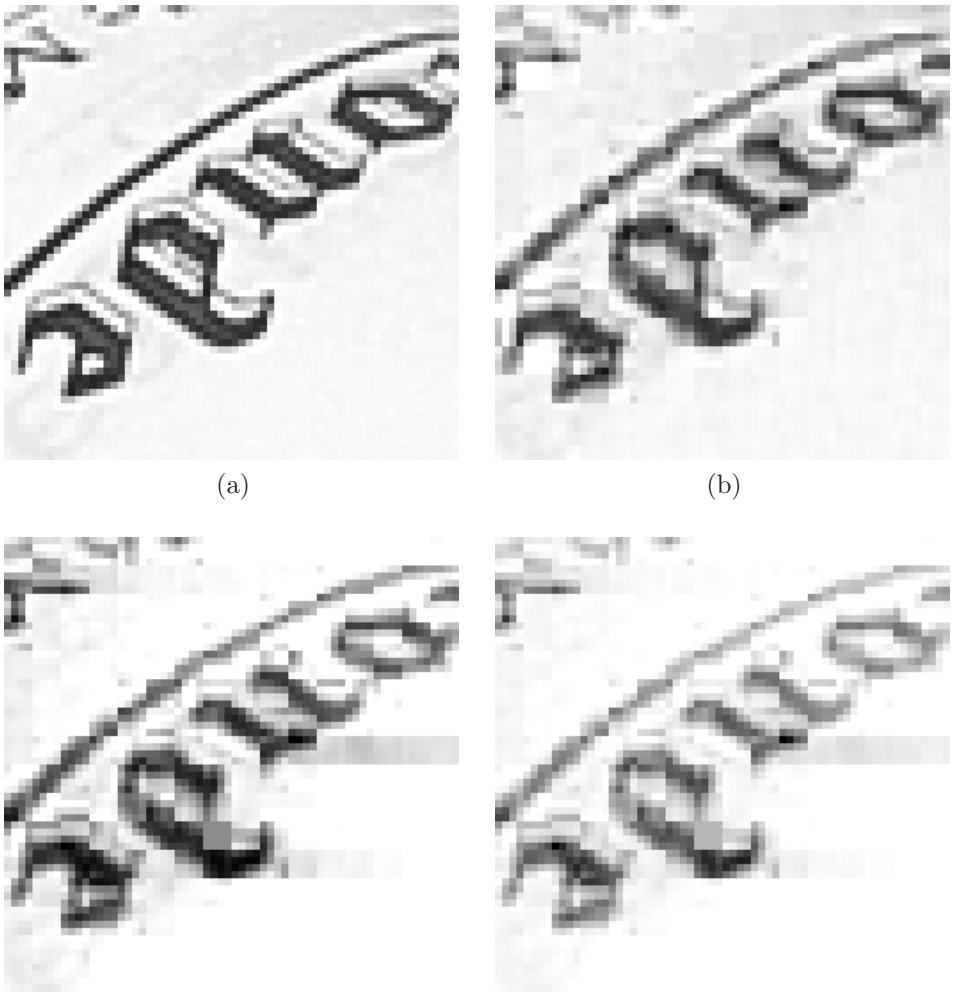
$$Y_{dist} = 0.7743Y_{ideal}^2 + 0.2531Y_{ideal} - 0.0265 \tag{5.1}$$

$$Y_{corrigida} = \frac{\sqrt{3.095Y_{circuito} + 0.146} - 0.253}{1.548} \tag{5.2}$$

A Figura 5.5(d) mostra o resultado da imagem da Figura 5.5(c) corrigida utilizando a Equação (5.2). Como pode ser visto no gráfico da Figura 5.8, para os pontos mais escuros, a simulação do Cadence encontrou valores negativos para os pixels. Alguns desses valores ao serem aplicados à Equação (5.2) resultam em números complexos. Para os casos em que isso acontece, esses números complexos foram substituídos pelo pixel de menor valor real da Figura 5.5(d). A Figura 5.5(d) possui PSNR igual a 25.1 dB. Apesar do resultado não chegar aos 30.1 dB da simulação do MATLAB, percebemos que o ajuste melhora consideravelmente a PSNR da imagem. Os outros 5 dB provavelmente são causados pelas imperfeições do circuito. A mesma correção foi aplicada à Figura 5.6(a). O resultado é a Figura 5.6(b), que possui PSNR igual a 25.4 dB.

Esta PSNR em torno de 25 dB, alcançada pelo novo sistema de compressão de dados (9 bits) em simulações já levando em conta imperfeições do circuito, assemelha-se à PSNR antes alcançada a nível teórico pelo sistema de 7 bits (23 dB conforme a Figura 2.2(b)). Em simulações no Cadence, o sistema anterior alcançou PSNR em torno de 20 dB (mostrada na Tabela 6.1) e, em testes experimentais, ele alcançou PSNR igual 18.4 dB para uma versão 96 x 96 da imagem do olho da Lena.

A Figura 5.9 mostra outra simulação nominal de 64 $\times$ 64 pixels. A imagem original pode ser vista na Figura 5.9(a). Nessa figura podemos ver parte de um jornal. É esperado que a PSNR da compressão dessa imagem seja baixa, pois as letras impressas no jornal possem alta frequência, uma vez que temos transições abruptas de preto para branco, e as altas frequências são descartadas no processo de compressão. O resultado da simulação numérica é apresentado na Figura 5.9(b). A PSNR dessa imagem é igual a 20.2 dB. A simulação no Cadence sem a correção da distorção quadrática é mostrada na Figura 5.9(c) e possui PSNR igual a 15.7 dB. Como a maioria dos pixels dessa imagem possui intensidade alta, o efeito da distorção quadrática não é tão explícito como na imagem da Lena. No entanto, ainda podemos observar o efeito nas letras, que aparecem com uma intensidade mais baixa que o original. Aplicando a Equação (5.2) à Figura 5.9(c), obtemos a Figura 5.9(d), que possui PSNR igual a 16.1 dB.



(a) (b)

(c) (d)

Figura 5.9: (a) Imagem de 64×64 utilizada para simulação, (b) simulação numérica, no MATLAB, (c) simulação a nível de circuitos, no Cadence, e (d) mesma simulação a nível de circuitos com correção da distorção quadrática.

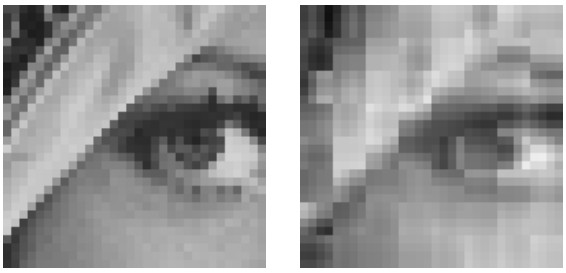
Capítulo 6

Conclusão

Nessa dissertação foi apresentado o projeto de um imageador CMOS, feito utilizando tecnologia de 0.18 μm. A tecnologia utilizada possui substrato P e poços N e P. Foram utilizados neste trabalho seis níveis de metal, um nível de silício policristalino, óxido de gate com espessura de 3.5 nm para os transistores com alimentação de 1.8 V. As simulações apresentadas mostram que o circuito atinge um resultado satisfatório, realizando uma compressão com perdas da imagem capturada com resultado próximo ao obtido pela simulação numérica do algoritmo. O chip foi enviado para fabricação no mês de agosto, para a rodada do dia 12 de agosto de 2013 e já foi fabricado. A Tabela 6.1 resume as características do projeto e faz uma comparação entre o projeto atual e o projeto antigo, feito com tecnologia de 0.35 μm. As PSNRs das simulações numéricas são as mesmas apresentadas no Capítulo 2. A PSNR da simulação do projeto anterior no Cadence apresentada na tabela foi retirada de uma dissertação de mestrado de Hugo L. Haas [25]. Para o cálculo dessa PSNR foi utilizada a imagem mostrada na Figura 6.1(a). O resultado da simulação no Cadence sem considerar o circuito de leitura é mostrado na Figura 6.1(b). Como a imagem da Figura 6.1(a) é semelhante à mostrada na Figura 5.5, o resultado de simulação foi utilizado para comparação. Para o projeto novo, foi considerada a melhor PSNR da simulação conforme explicado na Seção 5.4.

Como pode ser visto pela tabela, o novo projeto apresenta uma melhora significativa de PSNR, com um pequeno aumento na taxa de bits e na complexidade do circuito. Na simulação numérica, a melhora da PSNR é de 6.6 dB. Na simulação do Cadence, foi observada uma melhora de 5.5 dB. A diferença entre o resultado da simulação no MATLAB e a simulação no Cadence pode ser explicada pelos erros gerados devido à implementação analógica. Apesar do aumento do número de transistores por bloco, a utilização de um processo de fabricação mais moderno, cujas distâncias, comprimentos e larguras mínimas são menores que o processo anterior, permitiu o aumento do *fill factor* (razão entre a área ativa e a área total do pixel).

Pelos gráficos mostrados na Seção 3.1.2 não temos como avaliar o efeito direto



(a) (b)

Figura 6.1: (a) Imagem de 32 × 32 pixels utilizada para teste do projeto anterior e (b) resultado da simulação no Cadence sem considerar o circuito de leitura [25].

Tabela 6.1: Comparação entre o projeto apresentado na dissertação e o projeto anterior, feito com 0.35 μm.

	Projeto anterior	Projeto novo
PSNR (numérica)	23.2 dB	29.8 dB
PSNR (Cadence)	19.9 dB	25.4 dB
Taxa de bits	0.94 bpp	1.13 bpp
Coeficientes da transformada	4	5
Bits de sinal	4	5
Bits do VQ	7	9
Processo de fabricação	AMS 0.35 μm Opto	IBM 0.18 μm
Número de transistores por bloco	607	833
Área total do chip	2.4 mm × 2.1 mm	2.8 mm × 2.8 mm
Área do pixel	37.5 μm × 37.5 μm	27.2 μm × 27.2 μm
Área do fotodiodo	10 μm × 10 μm	10 μm × 10 μm
Fill factor	7.1%	13.5%
Resolução	32 × 32	64 × 64
DPCM $\hat{m}(1)$	0.0	0.4688
Tensão de alimentação	3.3 V	1.8 V
Tipo de saída	Serial, com compressão	Serial, com e sem compressão

do *cascode* na PSNR. O estudo desse efeito está em andamento. Para tal, serão feitas novas simulações trocando a matriz **H** por uma matriz com espelhos simples, de forma que poderemos medir a PSNR para a imagem comprimida com o circuito com espelhos simples e comparar os dois resultados.

Para os testes do chip, uma placa está sendo projetada utilizando um PIC 18LF4550 que será responsável por fazer a comunicação entre o chip e um computador via USB, por gerar os sinais de controle necessários para a captura de uma foto e por realizar a leitura dos bits gerados pelo circuito integrado. O PIC escolhido foi o mesmo utilizado para o projeto de 0.35 μm. Esse PIC possui uma interface USB e, durante os testes do projeto antigo, conseguimos perceber que ele atende aos requisitos desejados, em termos de velocidade (estabelecer intervalos de integração de 100 μs a 80 ms, gerar sinais de controle que permitem a execução do algoritmo

de compressão em 50 μs, fazer a leitura das imagens comprimidas em taxas de até 10 quadros/seg aproximadamente) e simplicidade (estrutura de programação relativamente simples). Devido ao número limitado de pinos do PIC escolhido, e pelo fato de desejarmos gerar sinais extremamente precisos para controlar o conversor A/D, será necessário utilizar um segundo PIC para gerar os sinais de controle do conversor A/D. A lente utilizada no projeto anterior também deve ser aproveitada para esse projeto, sendo apenas alrerada a distância entre o chip e a lente, e entre a lente e o alvo, para adaptar a área do novo chip à distância focal da lente.

Alguns testes importantes, que serão realizados após a fabricação da placa de teste e da adaptação do aparato óptico, para caracterização do imageador, são os seguintes: *modulation transfer function* (MTF), onde é medida a resposta em frequência espacial do chip; medida de *fixed pattern noise* (FPN); medida de potência consumida; extração da curva de resposta do circuito de leitura, através das estruturas de teste; comparação da faixa dinâmica de resposta do pixel 3T com o pixel em modo de corrente, utilizado no circuito, também através das estruturas de teste. Para os testes de MTF, também é interessante realizar simulações utilizando alvos padrão de teste como entradas do circuito esquemático, de forma que podemos comparar as simulações com os resultados experimentais. Utilizando o conversor A/D incluído no circuito, poderemos também avaliar experimentalmente o efeito da compressão nas imagens de forma separada, levando em conta os valores reais dos pixels amostrados.

Referências Bibliográficas

[1] NAKAMURA, J. *Image Sensors and Signal Processing for Digital Still Cameras*. 1 ed. EUA, CRC Press, Talyor & Francis Group, 2006.

[2] OTHA, J. *Smart CMOS Image Sensors and Applications*. 1 ed. EUA, CRC Press, Talyor & Francis Group, 2008.

[3] YADID-PECHT, O., ETIENNE-CUMMINGS, R. *CMOS Imagers: From Phototransduction to Image Processing*. 1 ed. EUA, Kluwer Academic Publishers, 2004.

[4] BELBACHIR, A. N. *Smart Cameras*. 1 ed. Nova Iorque, EUA, Springer, 2010.

[5] CULURCIELLO, E., ETIENNE-CUMMINGS, R., BOAHEN, K. "A biomorphic digital image sensor", *Solid-State Circuits, IEEE Journal of*, v. 38, n. 2, pp. 281–294, 2003.

[6] LEON-SALAS, W. D., BALKIR, S., SAYOOD, K., et al. "A CMOS Imager With Focal Plane Compression Using Predictive Coding", *IEEE Journal of Solid-State Circuits*, v. 42, n. 11, pp. 2555–2572, novembro de 2007.

[7] IGNJATOVIC, Z., MARICIC, D., BOCKO, M. "Low Power, High Dynamic Range CMOS Image Sensor Employing Pixel-Level Oversampling $\Sigma\Delta$ Analog-to-Digital Conversion", *Sensors Journal, IEEE*, v. 12, n. 4, pp. 737–746, abril de 2012.

[8] SARKAR, M., BELLO, D., VAN HOOF, C., et al. "Biologically Inspired CMOS Image Sensor for Fast Motion and Polarization Detection", *Sensors Journal, IEEE*, v. 13, n. 3, pp. 1065–1073, março de 2013.

[9] COTTINI, N., GOTTARDI, M., MASSARI, N., et al. "A 33 μW 64 $\times$ 64 Pixel Vision Sensor Embedding Robust Dynamic Background Subtraction for Event Detection and Scene Interpretation", *Solid-State Circuits, IEEE Journal of*, v. 48, n. 3, pp. 850–863, março de 2013.

[10] OLIVEIRA, F., HAAS, H., GOMES, J., et al. "CMOS Imager With Focal-Plane Analog Image Compression Combining DPCM and VQ", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 60, n. 5, pp. 1331–1344, maio de Sensitivity analysis of multilayer perceptrons applied to focal-plane image compression2013.

[11] PENNEBAKER, W. B., MITCHELL, J. L. *JPEG: Still Image Data Compression Standard*. Kluwer Academic Publishers, 1993.

[12] WANG, Z., BOVIK, A. C. "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures", *Signal Processing Magazine, IEEE*, v. 26, n. 1, pp. 98–117, janeiro de 2009.

[13] MALVAR, H. S., HALLAPURO, A., KARCZEWICZ, M., et al. "Low-Complexity Transform and Quantization in H.264/AVC", *IEEE Trans. Circuits and Systems for Video Technology*, v. 13, n. 7, pp. 598–603, julho de 2003.

[14] GERSHO, A., GRAY, R. M. *Vector Quantization and Signal Compression*. Massachusetts, EUA, Kluwer Academic Publishers, 1992.

[15] HAYKIN, S. *Communication Systems*. Quarta Edição. EUA, John Wiley & Sons, Inc., 2001.

[16] GOMES, J. G. R. C., PETRAGLIA, A., MITRA, S. K. "Sensitivity analysis of multilayer perceptrons applied to focal-plane image compression", *Circuits, Devices Systems, IET*, v. 1, n. 1, pp. 79–86, fevereiro de 2007.

[17] RAZAVI, B. *Fundamentals of Microelectronics*. Preview ed. , Wiley E-Text, 2006.

[18] RAZAVI, B. *Design of Analog CMOS Integrated Circuits*. Cingapura, McGraw-Hill, 2001.

[19] SEDRA, A. S., SMITH, K. C. *Microelectronic Circuits*. 6 ed. Oxford, Nova Yorque, Oxford University Press, 2011.

[20] PELGROM, M., TUINHOUT, H., VERTREGT, M. "Transistor matching in analog CMOS applications". In: *Electron Devices Meeting, 1998. IEDM '98. Technical Digest., International*, pp. 915–918, Califórnia, EUA, dezembro de 1998.

[21] MEHTA, S., ETIENNE-CUMMINGS, R. "A Simplified Normal Optical Flow Measurement CMOS Camera", *IEEE Trans. Circuits and Systems*, v. 53, n. 6, pp. 1223–1234, junho de 2006.

[22] SAINT, C., SAINT, J. *IC Layout Basics: A Practical Guide*. EUA, McGraw-Hill, 2001.

[23] SAINT, C., SAINT, J. *IC Mask Design: Essential Layout Techniques*. EUA, McGraw-Hill, 2002.

[24] WANG, Z., BOVIK, A., SHEIKH, H., et al. "Image quality assessment: from error visibility to structural similarity", *Image Processing, IEEE Transactions on*, v. 13, n. 4, pp. 600–612, abril de 2004.

[25] HAAS, H. L. "Projeto de Circuitos para Compressão de Imagens no Plano Focal de Câmeras CMOS", *Dissertação de Mestrado, COPPE/UFRJ*, 2012. http://objdig.ufrj.br/60/teses/coppe_m/HugoDeLemosHaas.pdf, acessado em: 03 de dezembro de 2013.

Apêndice A

Avaliações de SSIM

A Tabela A.1 mostra a SSIM dos resultados mostrados no Capítulo 5. Os resultados da SSIM têm boa correlação (0.82) com a PSNR, reforçando o fato que a medida de PSNR é confiável.

Tabela A.1: Comparação entre a PSNR e a SSIM da imagens comprimidas.

	PSNR	SSIM
Figura 5.1(a)	29.9 dB	0.93
Figura 5.1(d)	15.6 dB	0.69
Figura 5.2(a)	17.0 dB	0.79
Figura 5.3 (rodada 1)	13.6 dB	0.58
Figura 5.3 (rodada 2)	13.9 dB	0.54
Figura 5.3 (rodada 3)	13.7 dB	0.61
Figura 5.3 (rodada 4)	15.6 dB	0.68
Figura 5.3 (rodada 5)	13.3 dB	0.56
Figura 5.3 (rodada 6)	14.2 dB	0.57
Figura 5.3 (rodada 7)	13.1 dB	0.51
Figura 5.3 (rodada 8)	19.1 dB	0.79
Figura 5.4(a)	13.4 dB	0.54
Figura 5.5(b)	30.1 dB	0.91
Figura 5.5(c)	14.3 dB	0.75
Figura 5.5(d)	25.1 dB	0.87
Figura 5.6(a)	14.0 dB	0.75
Figura 5.6(b)	25.4 dB	0.88
Figura 5.9(b)	20.2 dB	0.85
Figura 5.9(c)	15.7 dB	0.69
Figura 5.9(d)	16.1 dB	0.72