FERRAMENTA DE PROJETO ASSISTIDO POR COMPUTADOR PARA REFERÊNCIAS DE TENSÃO CMOS DE BAIXO CONSUMO

Ana Claudia Italiano Mesquita

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.

Orientadores: Luis Fabián Olivera Mederos
Mariane Rembold Petraglia

Rio de Janeiro
Setembro de 2024

FERRAMENTA DE PROJETO ASSISTIDO POR COMPUTADOR PARA REFERÊNCIAS DE TENSÃO CMOS DE BAIXO CONSUMO

Ana Claudia Italiano Mesquita

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

Orientadores: Luis Fabián Olivera Mederos
Mariane Rembold Petraglia

Aprovada por: Profa. Mariane Rembold Petraglia
Profa. Fernanda D. V. R. de Oliveira
Prof. Alessandro Goncalves Girardi

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2024

Italiano Mesquita, Ana Claudia

Ferramenta de projeto assistido por computador para referências de tensão CMOS de baixo consumo/Ana Claudia Italiano Mesquita. – Rio de Janeiro: UFRJ/COPPE, 2024.

XII, 60 p.: il.; 29, 7cm.

Orientadores: Luis Fabián Olivera Mederos
Mariane Rembold Petraglia

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2024.

Referências Bibliográficas: p. 50 – 54.

1. Referência de tensão. 2. Ferramenta Assistida. 3. Algorítimo Evolucionário. I. Olivera Mederos, Luis Fabián *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Em memória de Luciana
Italiano e abuela Cacha.

Agradecimentos

Agradeço aos professores Antonio Petraglia, Mariane Petraglia e Carlos Teodósio, que contribuíram para minha formação ao longo dessa trajetória.

À minha mãe, Ana Lice, que sempre me apoiou e deu tanto suporte emocional, além de me ajudar financeiramente no início do mestrado.

À minha família uruguaia, que mesmo longe, se manteve positiva e motivada com a conquista deste título: Gabriela e Julio.

Ao meu avô Mesquita e ao meu pai, Gutemberg, que são meus fãs de carteirinha, e à família Italiano.

E, por fim, ao meu marido, Fabián, te amo, mi amor.

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

FERRAMENTA DE PROJETO ASSISTIDO POR COMPUTADOR PARA REFERÊNCIAS DE TENSÃO CMOS DE BAIXO CONSUMO

Ana Claudia Italiano Mesquita

Setembro/2024

Orientadores: Luis Fabián Olivera Mederos
Mariane Rembold Petraglia

Programa: Engenharia Elétrica

As referências de tensão (VRs) são componentes essenciais encontrados em qualquer sistema em circuitos integrados, e minimizar seu consumo de energia estática pode aumentar a eficiência de energia. O projeto de circuitos analógicos em nós nanométricos pode ser desafiador devido à necessidade de múltiplos parâmetros na modelagem de transistores para prever com precisão seu comportamento, e simulações extensivas são geralmente realizadas para garantir a operação adequada do circuito. Diversas ferramentas de projeto assistido por computador (CAD) têm sido relatadas nos últimos anos para reduzir o tempo típico necessário para projetos analógicos. Nesta dissertação de mestrado apresenta-se uma ferramenta CAD para o projeto robusto de VRs em tecnologia de semicondutor de óxido metálico complementar (CMOS) para ultra-baixa potência (ULP), que otimiza a VR usando um algoritmo evolutivo que considera a variabilidade do processo de fabricação. A ferramenta desenvolvida é implementada utilizando a linguagem de programação Python e é compatível com os simuladores Ngspice, Hspice e Spectre, possibilitando um fluxo de projeto de microeletrônica de código aberto. A fim de demonstrar o funcionamento adequado da ferramenta desenvolvida, são apresentados projetos, simulações e implementação de layout, em tecnologia SkyWater de 130 nm, de alguns protótipos de VRs, incluindo o envio para fabricação de um deles através do Programa de Universalização de Projeto de CIs (UNIC-CASS). Adicionalmente, a primeira versão da ferramenta, chamada EAVREF, é fornecida em um repositório no SourceForge.

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

COMPUTER-AIDED TOOL FOR LOW-POWER CMOS VOLTAGE REFERENCE DESIGNS

Ana Claudia Italiano Mesquita

September/2024

Advisors: Luis Fabián Olivera Mederos
Mariane Rembold Petraglia

Department: Electrical Engineering

Voltage references (VRs) are essential components found in any integrated circuit system, and minimizing their static power consumption can enhance energy efficiency. Designing analog circuits in nanometer nodes can be challenging due to the need for multiple parameters in transistor modeling to accurately predict their behavior, and extensive circuit simulations are usually performed to ensure proper circuit operation. Various computer-aided design (CAD) tools have been reported in recent years to reduce the typical time required for analog designs. This master's thesis presents a CAD tool for the robust design of VRs in complementary metal-oxide-semiconductor (CMOS) technology for ultra-low power (ULP), which optimizes the VR target using a variability-aware evolutionary algorithm. The developed tool is implemented using the Python programming language and is compatible with Ngspice, Hspice, and Spectre simulators, enabling an open-source microelectronics design flow. To demonstrate the proper functioning of the developed tool, the designs, simulations, and layout implementation of some VR prototypes in 130 nm SkyWater technology are presented, including the sending for fabrication of one of them through the Universalization of IC Design from CASS Program (UNICCASS). Additionally, the first version of the tool, named EAVREF, is provided in a repository on SourceForge.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo está organizado da seguinte forma. Na Seção 1.1 é apresentada uma revisão teórica e do estado da arte de referências de tensão (VRs, do inglês *Voltage References*) CMOS, com ênfase em ultra baixo consumo de energia. Na Seção 1.2 apresenta-se um contexto teórico de algoritmos evolucionários (EAs), destacando a importância destes em aplicações de circuitos integrados na literatura. A Seção 1.3 descreve a motivação e objetivo desta dissertação, no que se refere ao desenvolvimento de uma ferramenta assistida por computador baseada em EAs para projeto de VRs CMOS de ultra baixo consumo de energia. A Seção 1.4 destaca as principais contribuições da dissertação.

1.1 Referências de Tensão CMOS: Bloco Essencial em Circuitos Integrados

As referências de tensão CMOS são um bloco fundamental em circuitos integrados, já que, uma grande variedade de circuitos utilizam sua saída para o correto funcionamento. As referências de tensão CMOS exploram as propriedades dos transistores CMOS para gerar uma tensão de referência constante e precisa. Alguns exemplos de circuitos que utilizam VRs são: conversores DC-DC; conversores analógicos-digitais (A/D); conversores digitais-analógicos (D/A); osciladores de relaxação; osciladores controlados por tensão; entre outros. Um dos objetivos principais das referências de tensão é gerar uma tensão precisa com variações de temperatura (T) e tensão de alimentação, e desta forma, utilizar sua saída confiável para polarizar circuitos, ou ter um valor de referência preciso nas conversões.

Existem duas técnicas tradicionais para implementação de referências de tensão. A primeira, proposta por Widlar em 1971 [4], é baseada na combinação de dois comportamentos opostos com temperatura, sendo um comportamento complementar com a temperatura absoluta (CTAT) somado com outro proporcional com

temperatura absoluta (PTAT), e conhecida como CTAT+PTAT (ver Figura 1.1(a)). A segunda técnica se baseia na combinação subtrativa de dois comportamentos similares com temperatura, também conhecida como CTAT−CTAT (Figura 1.1(b)).

A técnica CTAT+PTAT representa uma abordagem engenhosa e eficaz na geração de VRs em circuitos integrados CMOS. Devido à dependência linear com a temperatura das implementações PTAT, em contraste com o comportamento de segunda ordem do CTAT, tais topologias não proporcionam um coeficiente de temperatura (TC, do inglês Temperature Coefficient) ideal. Assim, é comum a necessidade de aplicar uma compensação de curvatura para alcançar um TC reduzido, o que, por sua vez, aumenta a complexidade no projeto do circuito [5].

Uma melhoria significativa no TC pode ser alcançada utilizando as configurações CTAT−CTAT. Estas são criadas pela diferença entre dois comportamentos com temperatura similares, extraídos das tensões limiares dos transistores. Estas tensões limiares são comumente referidas na literatura como referências de tensão multi-limiares. Geralmente, são implementadas utilizando processos CMOS não convencionais, onde são obtidas duas tensões limiares diferentes nos transistores através de técnicas como implantes no canal, utilização de diferentes materiais de porta, dopagens de porta distintas, entre outras [6, 7]. No entanto, isso tende a aumentar os custos de fabricação. Alternativamente, as configurações CTAT−CTAT também podem ser implementadas utilizando o processo CMOS padrão, como sugerido em [8], aproveitando as diferenças convencionais entre as tensões limiares dos transistores NMOS e PMOS, respectivamente.

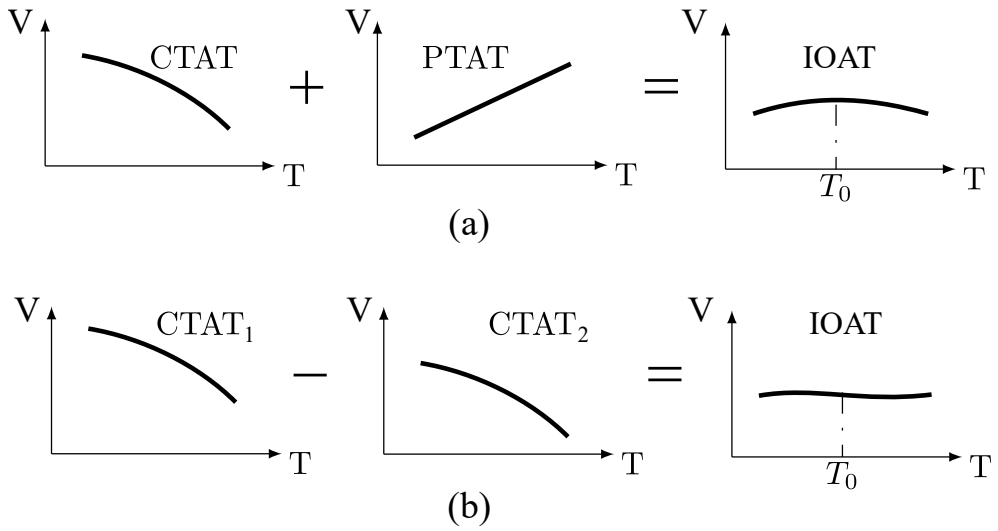


Figura 1.1: Técnicas convencionais de implementação de referências de tensão: (a) PTAT+CTAT; (b) CTAT−CTAT.

1.1.1 Métricas de Desempenho

Existem duas métricas fundamentais para avaliar referências de tensão, tais como, a regulação de linha (LR), e coeficiente de temperatura (TC), as quais avaliam o comportamento da tensão de referência em relação à alimentação e temperatura, respectivamente. A forma tradicional de medir estas métricas é mostrada nas Figuras 1.2(a) e 1.2(b) para LR e TC, respectivamente. Outras métricas de desempenho de referências de tensão, e que podem ser consideradas no projeto, são: relação de rejeição à fonte de alimentação (PSRR) (ver Figura 1.2(c)); consumo de potência (geralmente medido na tensão mínima de funcionamento); área ocupada no silício; faixa de temperatura; e tensão mínima de alimentação.

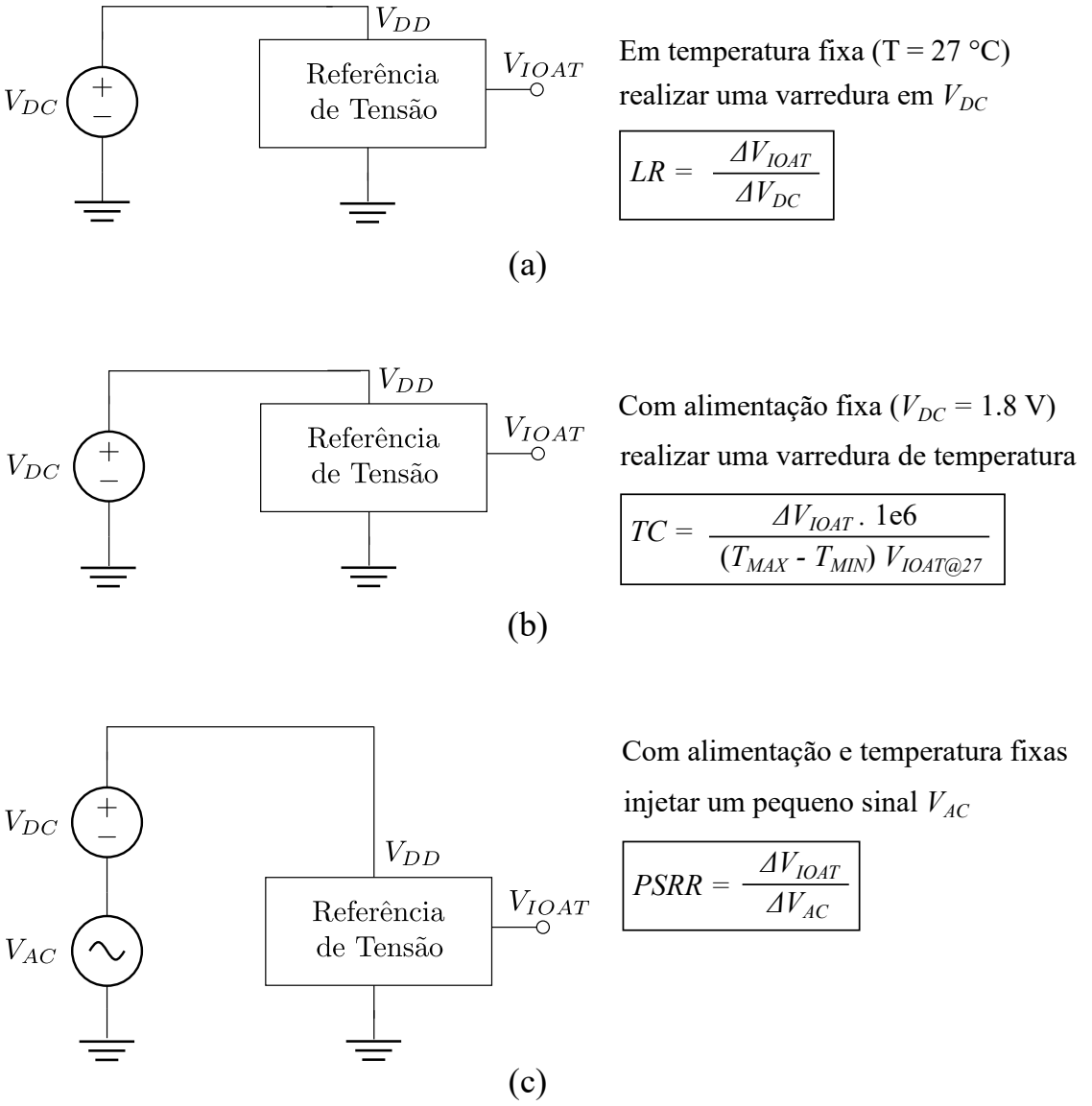


Figura 1.2: Testes de desempenho tradicionais em referências de tensão CMOS: (a) regulação de linha (LR); (b) coeficiente de temperatura (TC); relação de rejeição à fonte de alimentação (PSRR).

1.1.2 Modelo Teórico do Transistor MOS em Inversão Fraca

Ao longo desta dissertação são utilizados alguns parâmetros do transistor, especificamente no funcionamento de inversão fraca, ou conhecido também como região sublimiar. Uma equação adequada para a corrente subliminar entre dreno e fonte, que incorpora os principais efeitos nanométricos, como o *drain induced barrier lowering* (DIBL) e polarização de corpo, é dada por [9, 10]

$$I_{DS} = I_{So}\frac{W}{L}\exp\left(\frac{V_{GS}-V_{T,\text{eff}}}{nU_T}\right)\left(1-\exp\left(\frac{-V_{DS}}{U_T}\right)\right) \ , \tag{1.1}$$

onde U_T é a tensão térmica, $V_{T,\text{eff}}$, n, W/L e I_{So} são, respectivamente, a tensão limiar efetiva do transistor MOS, o *slope factor*, a razão de aspecto e a corrente específica dos transistores MOS. Adicionalmente, V_{GS} e V_{DS} denotam as tensões de porta-fonte e dreno-fonte, respectivamente. É possível aproximar a tensão limiar efetiva $V_{T,\text{eff}}$ em (1.1) por uma aproximação de primeira ordem, como

$$V_{T,\text{eff}} = V_{To} - \lambda V_{DS} - \eta V_{BS} \ , \tag{1.2}$$

onde λ e η são parâmetros para prever os efeitos de DIBL e polarização do corpo, respectivamente, e V_{BS} é a tensão de substrato para fonte. Também, V_{To} é a tensão de limiar extrapolada em V_{BS}=V_{DS}=0 V.

1.1.3 Referências de Tensão 2T para Ultra Baixo Consumo

Os reguladores de tensão auto-polarizados (SB) têm sido amplamente relatados na literatura nos últimos anos como uma excelente solução de compromisso entre consumo de energia e área [9, 11–13], já que as topologias SB empregam a corrente de vazamento sub-limiar para polarização do circuito. Em [11], Albano et al. propuseram a ideia original de um regulador de tensão ULP de 2 transistores (2T) baseado na fonte de corrente auto-polarizada (SB) com o gate conectado para terra (GG) (Figura 1.3(a)). Uma topologia alternativa de regulador de tensão ULP de 2T foi publicada em [12] por Wang et al., na qual os autores propuseram a fonte de corrente SB com gate conectado para source (GS) (Fig.1.3(b)). Deve-se notar que as topologias relatadas em [11, 12] geram as tensões de referência injetando a corrente SB em um MOS conectado como diodo, o que na prática pode ser implementado tanto por dispositivos NMOS (Fig.1.3(c)) quanto PMOS (Fig.1.3(d)).

O comportamento da corrente em relação ao MOS conectado como diodo é um aspecto crucial das alternativas GG-SB e GS-SB, pois isso controla a tensão de referência. Em contraste, a estrutura GS-SB pode operar efetivamente em temperaturas menores nas mesmas dimensões que a GG-SB, ao custo de um maior consumo de

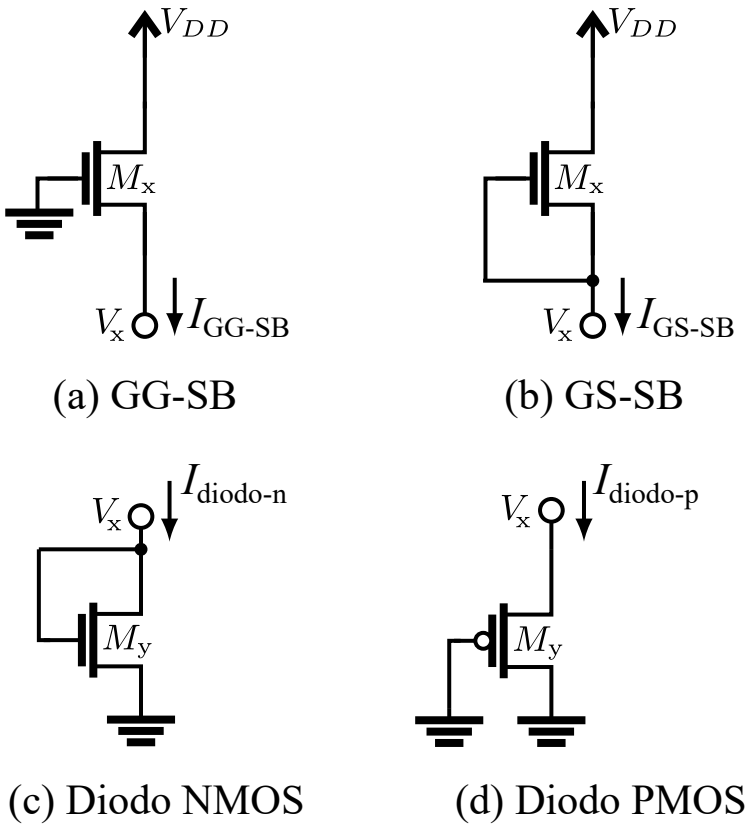


Figura 1.3: Componentes das referências de tensão 2T: (a) GG-SB; (b) GS-SB; (c) NMOS conectado como diodo; (d) PMOS conectado como diodo.

energia.

A sensibilidade da tensão de referência em relação à fonte de alimentação é um aspecto adicional que deve ser considerado nos projetos de referências de tensão. Uma estimativa analítica pode ser obtida derivando a equação de corrente em inversão fraca em cada topologia. A Tabela 1.1 apresenta os resultados e inclui a sensibilidade da variação relativa de corrente em relação a V_x (ver Figura 1.3). Como pode

Tabela 1.1: Comportamento da sensibilidade das implementações GG-SB and GS-SB.

Topologia	$\Delta V_x/\Delta V_{DD}$ [V/V]	$(\Delta I_{DS}/I_{DS})/\Delta V_x$ [1/V]
GG-SB	$\lambda/(\lambda+\eta+1)$	$-(\lambda+\eta+1)/(nU_T)$
GS-SB	$\lambda/(\lambda+\eta)$	$-(\lambda+\eta)/(nU_T)$

ser observado, as sensibilidades dependem dos efeitos da tensões de dreno e corpo, representados por λ e η, respectivamente. Além disso, do ponto de vista das correntes de vazamento, a estrutura GS-SB tem uma influência de fuga menor em baixas temperaturas do que a estrutura GG-SB. Por outro lado, deve-se notar a partir da

Tabela 1.1 que a estrutura GS-SB possui uma alta sensibilidade a variações na fonte de alimentação, o que pode limitar a especificação de regulação de linha (LR) dependendo da tecnologia CMOS. Portanto, como evidenciado na Tabela 1.1, valores altos de λ degradam a regulação das referências 2T, e assim, várias estruturas de 3 e 4 transistores, comumente chamadas de 3T e 4T, respectivamente, foram relatadas na literatura para aumentar a razão de rejeição da fonte de alimentação (PSRR) em relação às estruturas de 2T.

1.1.4 Referências de Tensão 3T, 4T e 5T para Ultra Baixo Consumo

Aproveitando as vantagens das referências 2T, várias estruturas 3T, 4T e 5T foram publicadas com dispositivos adicionais para melhorar o desempenho. As Figuras 1.4(d) [13] e 1.4(e) [2] mostram as versões de 3T das estruturas GG e GS, respectivamente, nas quais a alternativa de diodo MOS com conexão cascode foi usada em lugar do diodo MOS tradicional. A estrutura *self cascode* (SC) melhora a regulação de alimentação em relação às versões de 2T. Além disso, a estrutura de 4T com os dois gates conectados para terra (DGG) [13] (ver a Fig.1.4(g)) foi proposta para melhorar ainda mais a regulação de alimentação. As VRs tradicionais de 3T e 4T comumente precisam de pelo menos 3 (ou 4) transistores operando na região de saturação. Como a alimentação mínima é outra característica importante, uma referência 3T, com um transistor funcionando em região triodo (3T-TR), foi proposta em [9] (ver a Fig. 1.4(f)). A estrutura 3T-TR requer apenas 2 transistores operando em saturação, alcançando assim uma alimentação mínima similar às estruturas 2T.

Como mostrado na Fig.1.4(c), a estrutura 2T-GS pode ser implementada usando apenas dispositivos PMOS [3] (2T-GSP), a qual pode servir como uma alternativa dependendo da disponibilidade de dispositivos no processo CMOS utilizado. Devido à fraca regulação de linha desta estrutura de 2T, o trabalho [3] também sugere uma topologia de 5 transistores totalmente PMOS (5T-PMOS) (veja a Figura. 1.4(h)), que melhora consideravelmente a sensibilidade de alimentação em relação ao 2T-GSP.

Adicionalmente, para alcançar uma tensão de referência estável na presença de alto vazamento de gate, causado pelos efeitos de tunelamento direto em processos de nanômetros profundos (< 65 nm), em [14] foi proposta a estrutura 4T-GGLC (ver Figura. 1.4(i)).

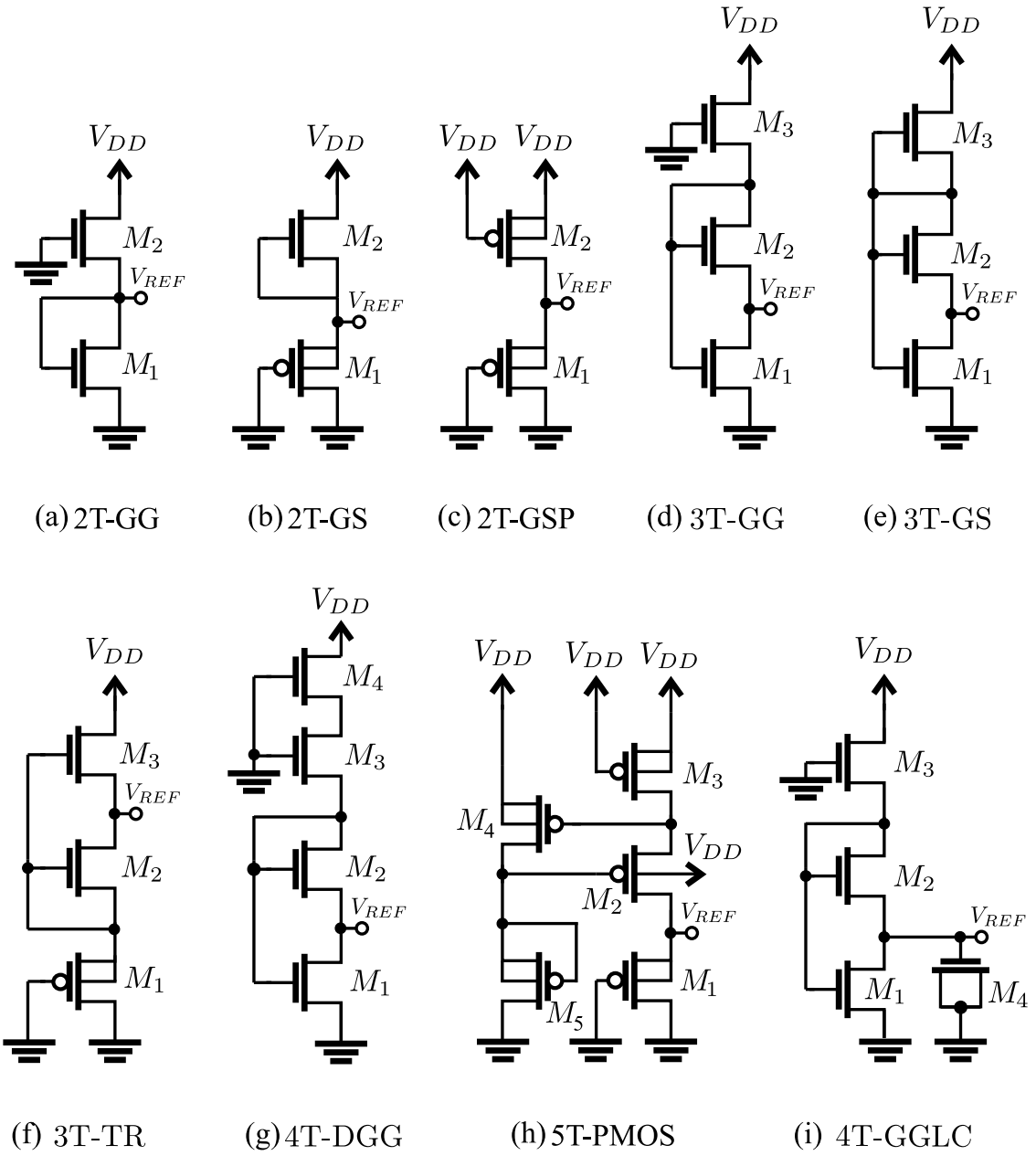


Figura 1.4: Topologias de referências de tensão CMOS de 2T, 3T, 4T e 5T, apropriadas para ultra baixo consumo e baixa tensão.

1.2 Algoritmos Evolucionários

Em um mundo cada vez mais complexo e interconectado, a busca por soluções ótimas para problemas desafiadores tem se tornado uma necessidade em diversas áreas do conhecimento. Os algoritmos evolucionários (EAs) emergem como uma poderosa ferramenta inspirada na natureza para enfrentar esses desafios [15, 16]. Baseados nos princípios da evolução biológica, como seleção natural, reprodução e mutação, os EAs oferecem uma abordagem robusta e adaptável para otimização e busca em espaços complexos.

1.2.1 Fundamentos da Computação Evolucionária

A computação evolucionária engloba uma família de algoritmos que simulam o processo de evolução natural para encontrar soluções dos problemas. Essa abordagem se destaca por sua capacidade de lidar com problemas de difícil modelagem, espaços de busca extensos e funções de custo complexas, onde métodos tradicionais de otimização podem se mostrar ineficazes.

Os EAs operam sobre uma população de soluções candidatas, representadas como indivíduos. Cada indivíduo é codificado como um conjunto de parâmetros, frequentemente chamado de cromossomo. Essa codificação pode ser binária, inteira, real ou até mesmo uma combinação de diferentes tipos de dados, dependendo da natureza do problema.

O processo evolutivo se inicia com a geração de uma população inicial, geralmente de forma aleatória. Cada indivíduo é então avaliado por uma função de aptidão (*fitness*), que quantifica a qualidade da solução em relação ao problema. Indivíduos com maior aptidão têm maior probabilidade de serem selecionados para reprodução, transmitindo suas características para a próxima geração. A reprodução em EAs envolve a aplicação de operadores genéticos, como cruzamento (*crossover*) e mutação. O cruzamento combina informações genéticas de dois pais para criar filhos potencialmente melhores, enquanto a mutação introduz variações aleatórias para explorar novas regiões do espaço de busca.

Este ciclo de avaliação, seleção, reprodução e mutação é repetido iterativamente por um número determinado de gerações ou até que um critério de parada seja atingido. Ao longo das gerações, a população evolui, com indivíduos mais aptos se tornando mais prevalentes e a qualidade média das soluções aumentando gradualmente.

1.2.2 Algoritmos Genéticos

Os algoritmos genéticos (GAs) são um dos pilares da computação evolucionária e um dos EAs mais amplamente utilizados. Sua popularidade se deve à sua simplicidade conceitual, facilidade de implementação e capacidade de resolver uma variedade de problemas complexos. Em um GA, a representação dos indivíduos é crucial para o sucesso do algoritmo. A escolha da representação depende da natureza do problema e pode influenciar na eficácia dos operadores genéticos. A representação binária é a mais tradicional, mas representações com números reais, inteiros ou permutações também são comuns. Os operadores de cruzamento e mutação em GAs são projetados para manipular a representação genética dos indivíduos. O cruzamento tipicamente envolve a troca de segmentos de cromossomos entre dois pais, enquanto a mutação altera aleatoriamente um ou mais genes em um indivíduo. A taxa de cruzamento e a taxa de mutação são parâmetros importantes que controlam a intensidade desses operadores e influenciam a convergência do algoritmo.

Um GA é composto por várias etapas interconectadas que emulam o processo de evolução natural. Como pode ser visto no diagrama simplificado da Figura 1.5, a estrutura tradicional de um GA inclui:

- Inicialização da população: uma população inicial de indivíduos é gerada aleatoriamente ou a partir de algum conhecimento prévio sobre o problema;
- Avaliação da aptidão: cada indivíduo da população é avaliado de acordo com uma função de aptidão (*fitness*), que mede a qualidade da solução representada pelo indivíduo;
- Seleção: indivíduos com maior aptidão são selecionados para reprodução, com base em diferentes mecanismos de seleção, como roleta, torneio ou ranking;
- Cruzamento (*crossover*): os indivíduos selecionados são emparelhados e submetidos ao cruzamento, onde informações genéticas são trocadas para criar novos indivíduos (filhos);
- Mutação: os filhos gerados pelo cruzamento são submetidos à mutação, onde pequenas alterações aleatórias são introduzidas em seus genes;
- Nova geração: os filhos resultantes do cruzamento e mutação, juntamente com alguns indivíduos da geração anterior (elitismo), formam a nova população;
- Critério de parada: o processo evolutivo continua até que um critério de parada seja atingido, como um número máximo de gerações, um tempo limite ou a convergência para uma solução satisfatória.

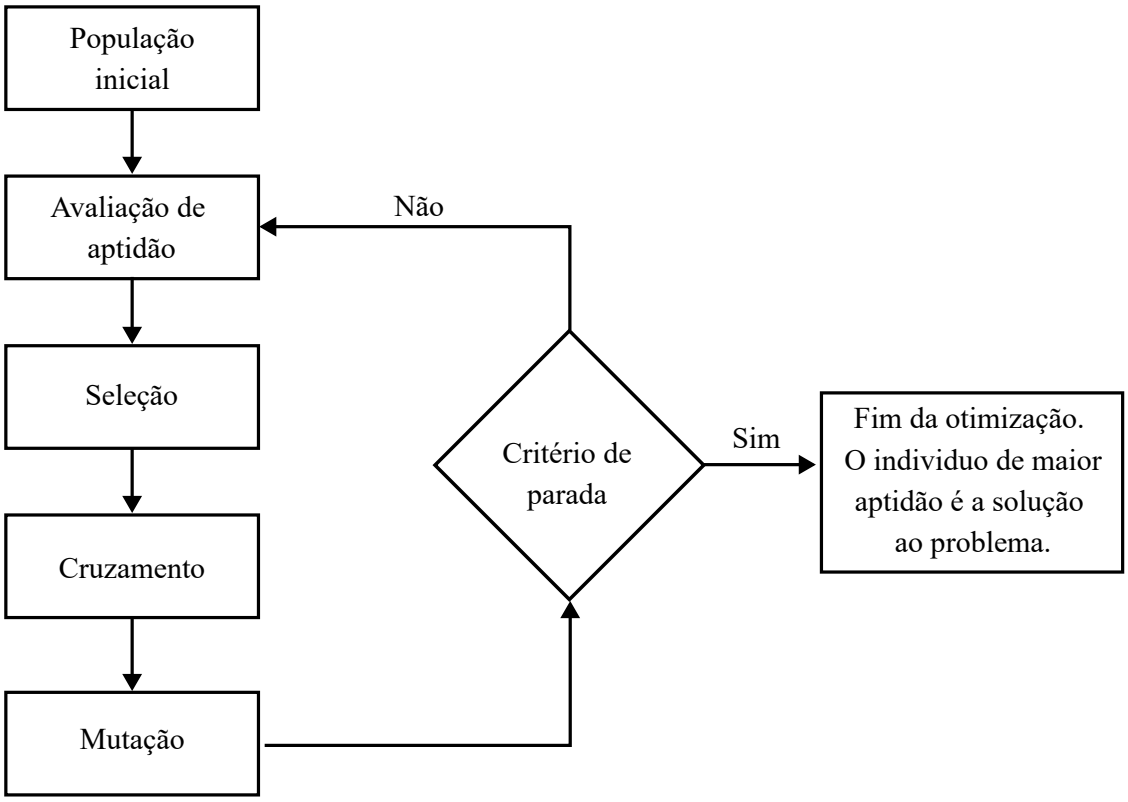


Figura 1.5: Diagrama ilustrativo da estrutura de um algoritmo genético básico.

1.2.3 Aplicações de EAs em Otimização de Circuitos

Os EAs têm encontrado aplicações em uma ampla gama de áreas, demonstrando seu potencial para resolver problemas complexos do mundo real. Um exemplo é a otimização de circuitos eletrônicos, onde os EAs são utilizados em conjunto com ferramentas de automação de projeto eletrônico (EDA) para encontrar soluções que maximizem o desempenho e minimizem o custo. A otimização de circuitos microeletrônicos CMOS é um desafio cada vez mais demandado pela industria, envolvendo uma grande variedade de parâmetros e restrições. No projeto destes circuitos, os EAs podem ser aplicados para explorar soluções que atendam a critérios específicos, tais como, desempenho, eficiência energética, área ocupada e custo de fabricação.

Ferramentas EDA desempenham um papel crucial nesse processo, permitindo simular e avaliar o desempenho de diferentes projetos de circuitos gerados pelos EAs. Essa avaliação fornece *feedback* para o processo de otimização, guiando os algoritmos na busca por soluções cada vez melhores. Alguns exemplos de aplicações de EAs em otimização de circuitos microeletrônicos são:

- Otimização de topologias de circuitos [17–21]: EAs podem ser utilizados para encontrar o melhor dimensionamento de componentes e interconexões em um circuito, buscando minimizar a área, reduzir o consumo de energia e melhorar o desempenho.

- Síntese lógica [22, 23]: EAs podem ser aplicados na geração de circuitos lógicos a partir de descrições de alto nível, buscando minimizar o número de portas lógicas, reduzir o atraso de propagação e otimizar o consumo de energia.
- Layout de circuitos [24, 25]: EAs podem auxiliar na organização dos componentes em um circuito integrado, buscando minimizar a área ocupada, reduzir o comprimento das interconexões e melhorar a eficiência térmica.

1.3 Motivação desta Dissertação

Nos últimos anos, houve um aumento expressivo no âmbito dos projetos de circuitos de ultra baixa potência (ULP) e ultra baixa tensão (ULV). Os sistemas autoalimentados baseados em colheita de energia surgiram como uma estratégia para aprimorar a eficiência de energia de sistemas em um chip (SoC) [26, 27]. Para o uso de coletores baseados em geradores termométricos e células fotovoltaicas como fontes de energia, é necessário garantir a operação dos circuitos com uma tensão de alimentação de aproximadamente 100–200 mV. Uma parte vital dos sistemas de gerenciamento de energia são as referências de tensão [13, 28], as quais são necessárias para a operação de osciladores de relaxação e osciladores em anel, que compõem o bloco de inicialização destes sistemas.

O projeto de circuitos analógicos em nós nanométricos pode ser desafiador devido à necessidade de múltiplos parâmetros na modelagem de transistores para prever com precisão seu comportamento. Desta forma, extensas simulações de circuitos eletrônicos com alto custo computacional são geralmente necessárias para garantir a operação adequada. Várias ferramentas de projeto assistido por computador (CAD) foram publicadas para reduzir o tempo típico necessário em projetos de circuitos analógicos [17–21]. Essas ferramentas são particularmente vantajosas quando múltiplos parâmetros precisam ser otimizados simultaneamente. As ferramentas de automação de projeto eletrônico têm ganho uma grande relevância no campo da microeletrônica, o qual tem registrado um mercado internacional de 10 bilhões de dólares em 2019, e projetado para aproximadamente 26 bilhões em 2028 [1, 29, 30] (ver Figura 1.6).

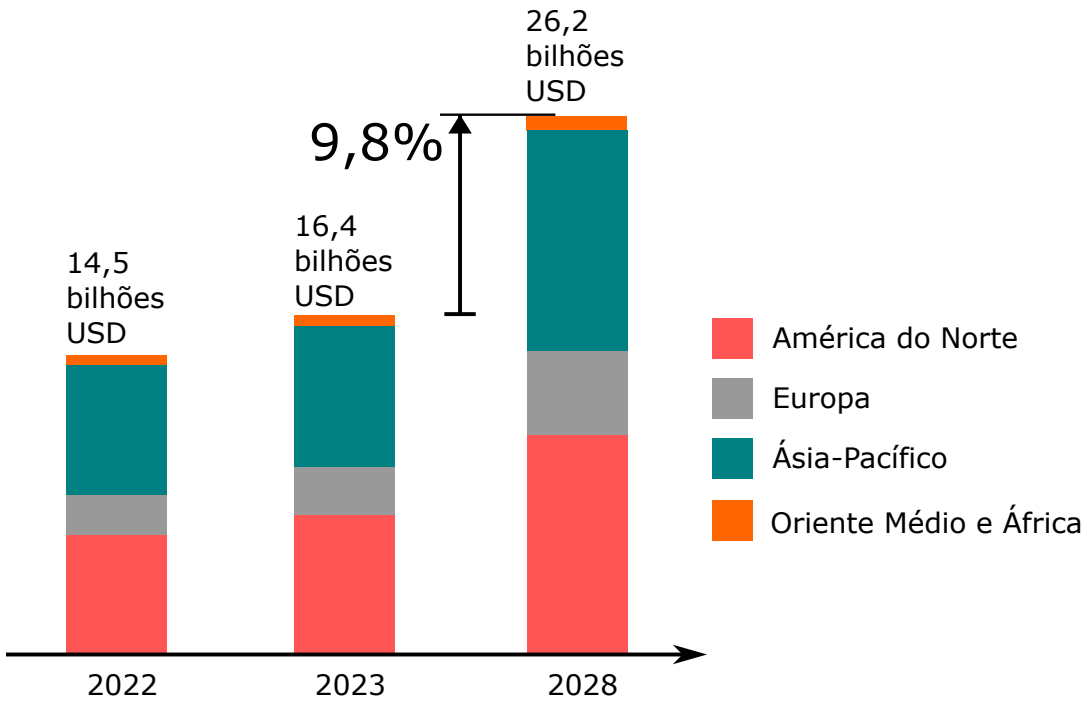


Figura 1.6: Previsão global do mercado de automação de projeto eletrônico (EAD) [1].

1.4 Contribuições desta Dissertação

Nesta dissertação, é apresentada uma ferramenta CAD para o projeto de referências de tensão CMOS de ultra baixo consumo, a qual leva em consideração a variabilidade do processo de fabricação dos dispositivos, o que torna o projeto do circuito mais robusto. A abordagem proposta utiliza um algoritmo evolucionário (EA) que incorpora simulações de circuito para avaliar sua função de aptidão, a qual é baseada na FoM tradicional de referências de tensão. A FoM é uma métrica convencional usada para avaliar o desempenho de circuitos analógicos em geral, sendo que as referências de tensão CMOS têm uma FoM bem definida [13, 28], que é frequentemente empregada para comparações de projetos na literatura. Esta leva em consideração aspectos cruciais como faixa de temperatura, consumo de potência, coeficiente de temperatura e área ocupada no silício. A ferramenta proposta é compatível com simuladores de microeletrônica populares, como Ngspice, Hspice e Spectre, melhorando a robustez e confiabilidade dos projetos produzidos, pois é possível utilizar modelos de dispositivos fornecidos pelas fábricas de circuitos integrados. Para mostrar a eficácia da ferramenta desenvolvida nesta dissertação, dois exemplos de projetos são realizados usando a tecnologia SkyWater 130 nm, e seus resultados de simulação pós-layout são apresentados. Além disso, a primeira versão estável da ferramenta, denominada EAVREF, juntamente com um breve tutorial de instalação e uso, é disponibilizado

em um repositório do SourceForge [31]. Desta forma, as contribuições principais desta dissertação podem ser separadas nos seguintes tópicos:

- **Artigo publicado/apresentado no SBCCI 2024**: Um artigo científico, intitulado *"EAVREF: An Evolutionary Algorithm Based Tool for Low-Power CMOS Voltage Reference Designs"* [32], foi publicado e apresentado na conferência internacional *37th Symposium on Integrated Circuits And Systems Design* (SBCCI) com objetivo de divulgação da ferramenta na literatura. Neste artigo foram apresentados os detalhes do funcionamento da ferramenta desenvolvida e o resultado de simulações pós-layout dos dois circuitos projetados.

- **Ferramenta CAD funcional**: a ferramenta desenvolvida, denominada EAVREF, foi disponibilizada para a comunidade de microeletrônica na sua versão 1.0 de forma livre [31], possuindo uma interface gráfica de usuário, permitindo uma utilização mais intuitiva. A versão 1.0 é compatível com Windows e Linux, tornando a ferramenta de fácil acesso para qualquer usuário. A compatibilidade com Ngpsice, permite que seja utilizada em fluxo de microeletrônica aberto, conjuntamente com processo CMOS 130 nm da Skywater.

- **Envio para fabricação de um Protótipo de CI**: Um protótipo de referência de tensão projetado nesta dissertação, dimensionado com EAVREF, foi enviado para fabricação através do programa *Universalization of IC Design from Circuits and Systems Society* (UNIC-CASS). [33]. O teste experimental deste CI é trabalho futuro desta dissertação.

1.5 Estrutura do Documento

O restante do documento está organizado da seguinte forma. O Capítulo 2 descreve o fluxograma de funcionamento da ferramenta, chamada de EAVREF, a estrutura do algoritmo evolucionário proposto, além da interface gráfica e outras funcionalidades da ferramenta. O Capítulo 3 desenvolve dois projetos de referências de tensão de ultra baixo consumo mediante a utilização de EAVREF, além da implementação do layout dos circuitos resultantes e sua respectiva simulação pós-layout. No Capítulo 4 são apresentadas as conclusões e trabalhos futuros desta dissertação.

Capítulo 2

EAVREF: Ferramenta para Projeto de Referências de Tensão

Neste capítulo, começamos com uma descrição das métricas usadas para avaliar o desempenho de referências de tensão, seguida por uma descrição geral da ferramenta proposta, descrevendo o do algoritmo evolucionário e a interface gráfica desenvolvida, entre outros detalhes das funcionalidades da ferramenta.

2.1 Métricas de Desempenho para Avaliar Referências de Tensão

Uma métrica tradicional para comparar o desempenho de projetos de referências de tensão CMOS é a FoM definida como [13, 28]

$$\mathrm{FoM} = \frac{(T_{\max} - T_{\min})^2 \cdot 10^{-21}}{\mathrm{TC} \cdot I_o \cdot V_{DD,\min} \cdot A_S} \quad \left[\frac{^{\circ}\mathrm{C}^3}{\mathrm{W} \cdot \mathrm{mm}^2}\right] \ , \tag{2.1}$$

onde o TC é o coeficiente de temperatura, I_o é o consumo de corrente em temperatura ambiente, $V_{DD,\min}$ é a alimentação mínima, A_S é a área ocupada no silício, e $T_{\min}$ e $T_{\max}$ são as temperaturas mínima e máxima de operação, respectivamente. O cálculo do parâmetro TC usando a abordagem de caixa é ilustrado na Figura 2.1(a).

A implementação do algoritmo evolucionário na ferramenta proposta requer a incorporação de uma função de aptidão. Nesta pesquisa, empregamos a seguinte função de aptidão específica para os projetos de ULP e ULV VR, baseada na métrica apresentada na Equação (2.1) e em algumas intuições heurísticas da convergência do algoritmo:

$$\Psi_{\mathrm{fit}} \ = \ \frac{(T_{\max} - T_{\min})^2 \cdot 10^{-12}}{(\mathrm{TC}_{\mathrm{avg}} + 3\mathrm{TC}_{\mathrm{std}})^2 \cdot I_{o,\mathrm{avg}}^{0.5} \cdot V_{DD,\min} \cdot (5A_G)^{0.7}} \ . \tag{2.2}$$

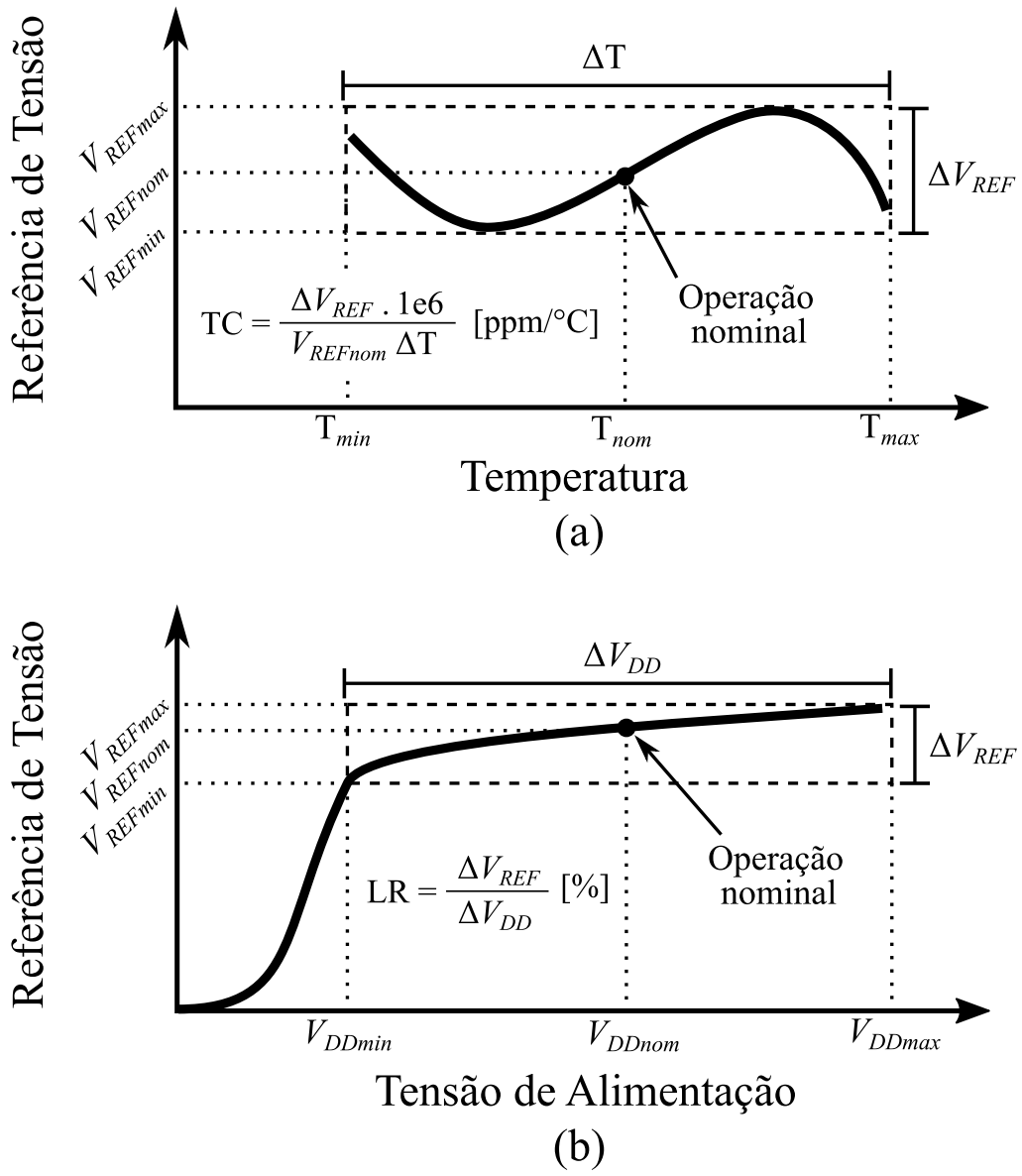


Figura 2.1: Definição de métricas usando o método de caixa: (a) coeficiente de temperatura (TC); (b) regulação de linha (LR).

Na Equação (2.2), é proposta a utilização dos resultados de simulações de Monte Carlo para determinar os valores médio e de desvio padrão do coeficiente de temperatura, TCavg e TCstd, respectivamente. Deve-se notar que o procedimento para incluir simulações de Monte Carlo na função de aptidão do algoritmo mostrou excelente capacidade de gerar circuitos robustos mantendo baixo custo computacional, uma vez que apenas 20 execuções de Monte Carlo em cada geração foram suficientes.

Também na Equação (2.2), o consumo médio de corrente em temperatura ambiente é denotado como $I_{o,\text{avg}}$, a tensão mínima de alimentação é representada como $V_{DD,\text{min}}$, e a área total ocupada por portas dos transistores do circuito é denotada como A_G (somatório dos comprimentos vezes larguras). Para obter uma estimativa preliminar da área ocupada pelo silício através de A_G, um fator de preenchimento

de 5 é empregado na Equação (2.2). A regulação de linha (LR) também é calculada usando a abordagem de caixa, como mostrado na Figura 2.1(b). A partir da Figura 2.1(b), pode ser observado que a alimentação mínima é determinada implicitamente pela especificação de LR, pois LR define a faixa de tensão de alimentação entre mínimo e máximo.

A implementação para estimar TC em Python é mais simples que a do LR, pois as temperaturas miníma e máxima são fixas desde o inicio da análise. No entanto, a implementação do LR é mais complexa, pois a tensão miníma de funcionamento não é conhecida e deve ser extraída. De forma a facilitar o entendimento, a Figura 2.2 mostra o código Python que foi utilizado para calcular LR e $V_{DD,\text{min}}$. Resumidamente, o código implementa o procedimento ilustrado na Figura 2.1(b), calculando LR pelo método de caixa para diferentes tensões mínimas, fazendo uma varredura na tensão de maior a menor. O último valor de tensão que cumpre LR especificado pelo usuário (no código, vdd_tol), é considerado $V_{DD,\text{min}}$.

```
LR = np.ones(p.size)*100  # starts in 100%
vddMin = np.ones(p.size)
for i in range(p.size):
    Vref_aux = cp.copy(vref_vdd[i])
    j = 0
    while ( j < len(vdd)-1 ) & ( LR[i] > vddTol ):
        MaxVref = max(Vref_aux[j:])
        MinVref = min(Vref_aux[j:])
        if Vref_aux[j] >= config.vrefmin:
            LR[i] = (MaxVref-MinVref)/(vdd[-1]-vdd[j])/(MaxVref+
                MinVref)*2*100
            vddMin[i] = cp.copy(vdd[j])
        j = j + 1
    if j == len(vdd)-1:
        vddMin[i] = vdd[-1]*10
        LR[i] = 100
```

Figura 2.2: Código Python para cálculo de LR e $V_{DD,\text{min}}$.

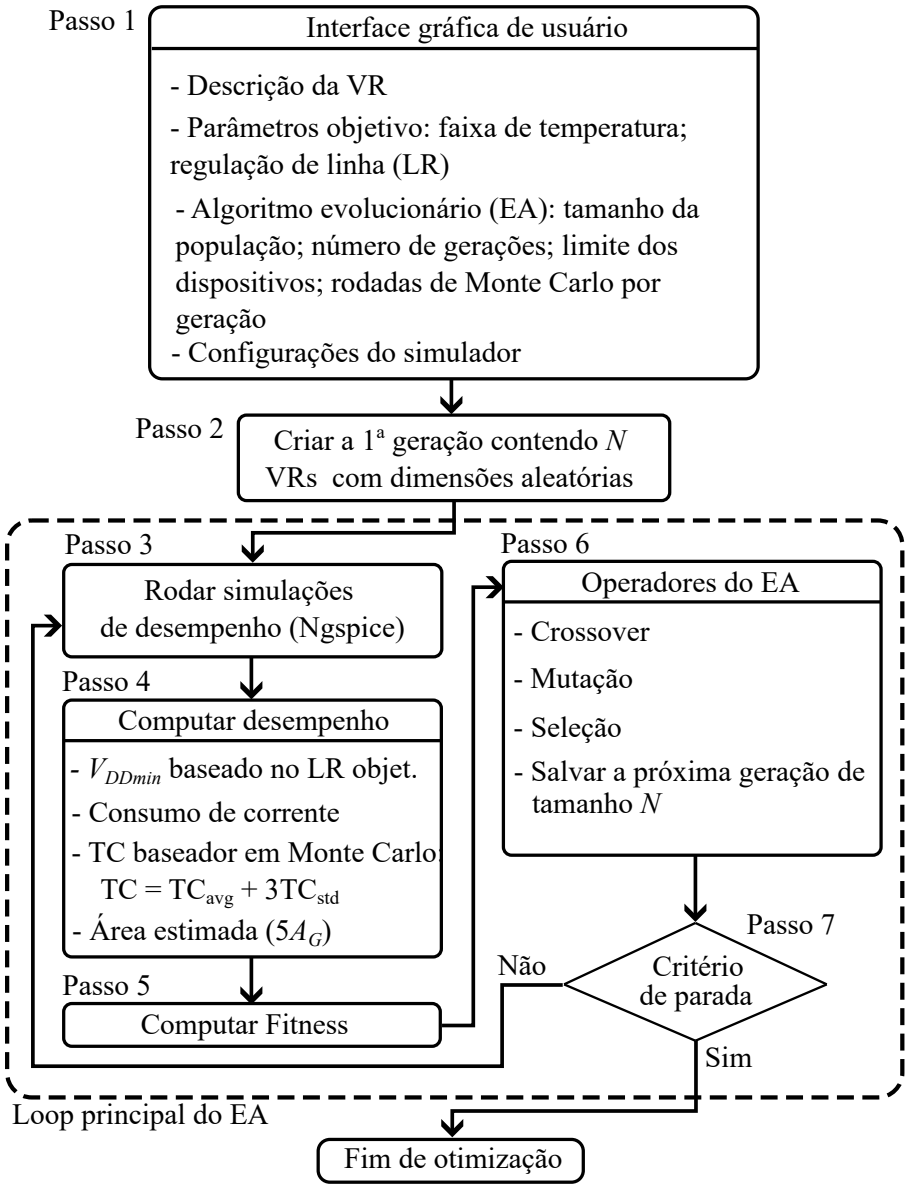


Figura 2.3: Fluxograma simplificado do funcionamento de EAVREF.

2.2 Fluxograma de Funcionamento EAVREF

O loop principal do código e sua respectiva interface gráfica foram implementados usando a linguagem de programação Python. O fluxograma apresentado na Fig. 2.3 ilustra simplificadamente o funcionamento da ferramenta quando é executada para otimizar VRs, usando algoritmo evolucionário. O usuário começa especificando a topologia do VR e definindo alguns parâmetros cruciais fixos, como faixa de temperatura, regulação de linha (LR) e outras configurações do algoritmo evolucionário (EA), como pode ser visto na Figura 2.3, Passo 1. Esses parâmetros do EA são descritos em mais detalhes na Seção 2.3. Subsequentemente, o usuário inicia o loop de otimização. O loop é controlado automaticamente com Python e só termina quando os critérios de parada são satisfeitos. Após a inicialização do loop, N circuitos de

VRs, com dimensões aleatórias, são criados para a geração inicial do EA (Figura 2.3, Passo 2). O desempenho desses circuitos de VRs são avaliados pela execução de simulações usando Ngspice (Figura 2.3, Passo 3). Os resultados dessas simulações são então armazenados como variáveis Python para poder calcular os parâmetros de desempenho (Figura 2.3, Passo 4). Após calcular os parâmetros desempenho, a função de aptidão descrita na Equação (2.2) pode ser avaliada para todas as VRs contidas na população do EA (Figura 2.3, Passo 5). O resultado da aptidão é empregado na evolução do EA para a seleção da população subsequente (Figura 2.3, Passo 6). Até que a condição de parada seja alcançada (Figura 2.3, Passo 7), o loop retorna para executar simulações de desempenho da geração seguinte (Figura 2.3, Passo 3).

2.3 Estrutura do Algoritmo Evolucionário

O problema de otimização apresentado busca maximizar a função de aptidão da Equação (2.2), que é determinada principalmente pelas dimensões dos dispositivos MOS. Usando comprimentos de junção fixos, um dispositivo MOS tem sua largura (W) e comprimento (L) como variáveis, restringidas por valores mínimos e máximos no processo de fabricação. Portanto, Ψ_{fit}, definida em (2.2), é uma função das dimensões dos dispositivos MOS, ou seja,

$$\Psi_{\text{fit}}(\{W_i, L_i\}) \ , \ \text{ para } \ i = 1, 2, .., M \ , \tag{2.3}$$

onde W_i e L_i são a largura e o comprimento, respectivamente, do i-ésimo dispositivo MOS pertencente à referência de tensão descrita pelo usuário, e M é o número total de dispositivos MOS. A seguir, a representação do domínio e os operadores do algoritmo evolucionário, como cruzamento (do inglês, *crossover*), mutação e seleção (Figura 2.3, Passo 6), são descritos.

2.3.1 Representação e Codificação

Como ilustrado na Figura 2.4, a população do EA é formada por N indivíduos, cada um com $2M$ parâmetros. Os parâmetros de cada indivíduo, conhecidos como genes, são as dimensões dos dispositivos MOS, e neste trabalho, são codificados como números reais restritos pelos limites fornecidos pelo fabricante do circuito integrado.

2.3.2 População Inicial

Os algoritmos evolucionários costumam iniciar sua primeira população de forma aleatória para garantir a diversidade da população. Em nosso problema, gerar uma

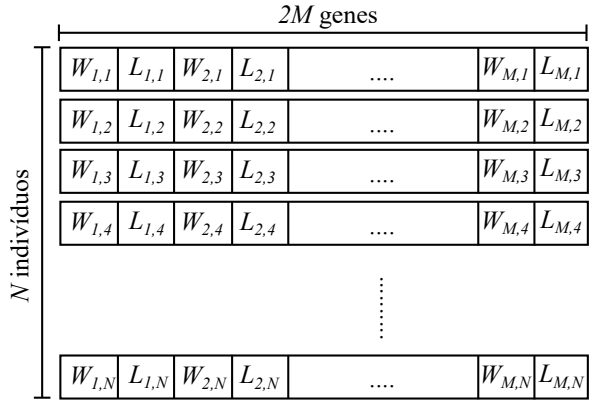


Figura 2.4: Diagrama ilustrativo da representação da população de N indivíduos.

população inicial consiste em escolher tamanhos iniciais para os dispositivos MOS. Isso pode ser implementado de uma forma muito simples, utilizando uma função geradora de números aleatórios com probabilidade uniforme na faixa desejada. Por exemplo, para a largura do transistor, os limites da faixa seriam as larguras mínima e máxima fornecidas pelo fabricante. No entanto, como pode ser visto na ilustração da Figura 2.5, a escala linear torna as dimensões pequenas de menos relevância, e o sorteio inicial produz dimensões exageradas, o que não é uma boa solução inicial para nossa função aptidão pelo aumento de área. Portanto, baseado nas observações da convergência do algoritmo, neste trabalho foi proposto realizar a geração aleatória inicial das dimensões em escala logarítmica, mostrando bons resultados no tempo de convergência e eventuais máximos locais.

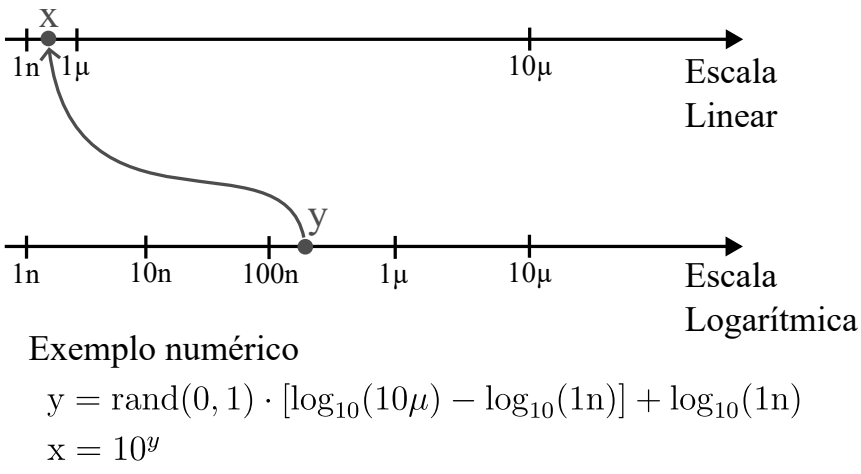


Figura 2.5: Diagrama ilustrativo da transformação de escala logarítmica linear utilizada para gerar a população inicial.

2.3.3 Cruzamento

Seguindo a abordagem convencional de Algoritmos Evolutivos, a geração de novos indivíduos (descendentes) ocorre por meio do cruzamento de pais selecionados através de um torneio aleatório entre toda a população. A Figura 2.6 ilustra um exemplo de torneio em uma população de 8 indivíduos, no qual pares de indivíduos são sorteados aleatoriamente para competir entre si. Neste trabalho, adotamos a implementação simplificada em que os vencedores de cada par são aqueles com maior aptidão.

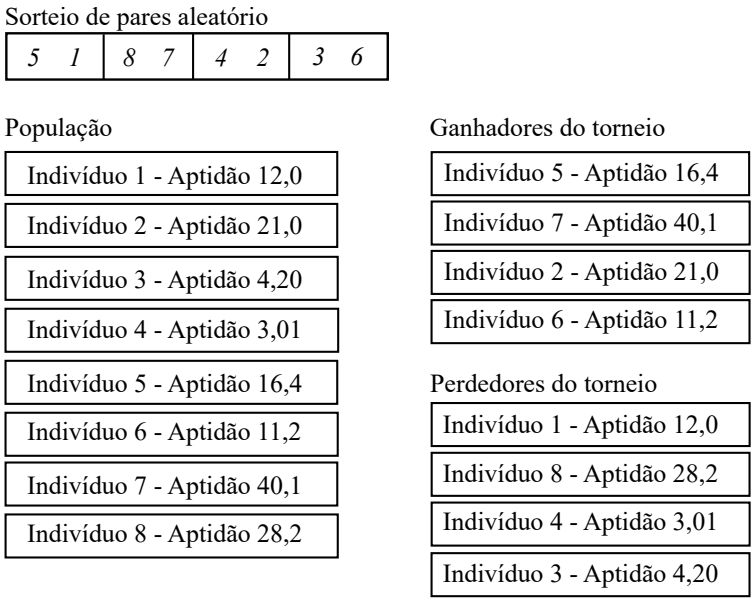


Figura 2.6: Exemplo ilustrativo de torneio aleatório em uma população de 8 indivíduos.

É importante ressaltar que a natureza aleatória da seleção por torneio contribui para a manutenção da diversidade da população. Isso pode ser observado no exemplo da Figura 2.6, onde o indivíduo 8, apesar de apresentar excelente aptidão, enfrenta um competidor ainda melhor. Similarmente, o indivíduo 6, com aptidão abaixo da média, consegue vencer o torneio. A operação de cruzamento é calculada tomando a média aritmética de ambos os pais em cada posição dos genes como pode ser visto na Figura 2.7.

2.3.4 Mutação

A mutação dos genes, que são codificados como números reais, é implementada seguindo o operador proposto no trabalho [34], e é dada por

$$\gamma_i^{\text{mut}} = \gamma_i + \alpha_i \cdot r_i \cdot 2^{-4\mu_i} \ , \tag{2.4}$$

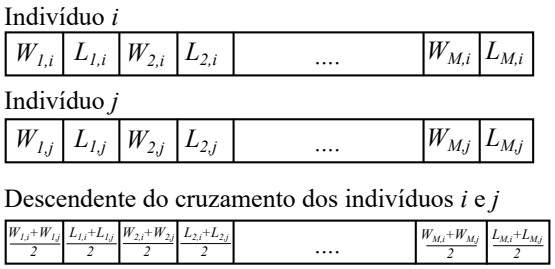


Figura 2.7: Diagrama ilustrativo da operação de cruzamento.

onde γ_i^{mut} é valor da mutação resultante, γ_i é valor do gene a ser mutado, α_i é uma variável aleatória uniforme variando de -1 a 1, r_i é o fator de alcance da mutação normalizado para 20% dos limites de γ_i, e μ_i é uma variável aleatória uniforme variando de 0 a 1. Como também foi sugerido em [34], a taxa de probabilidade de mutação é dada por $1/\sqrt{2M}$.

2.3.5 Seleção

Esta etapa envolve a criação da população da próxima geração. Como pode ser observado no exemplo ilustrativo da Figura 2.8, neste trabalho foi proposta a seguinte estrutura para a nova geração: ganhadores do torneio (50%); descentes de pares de ganhadores do torneio (25%); o restante da população conformada pelos perdedores do torneio com maior aptidão (25%).

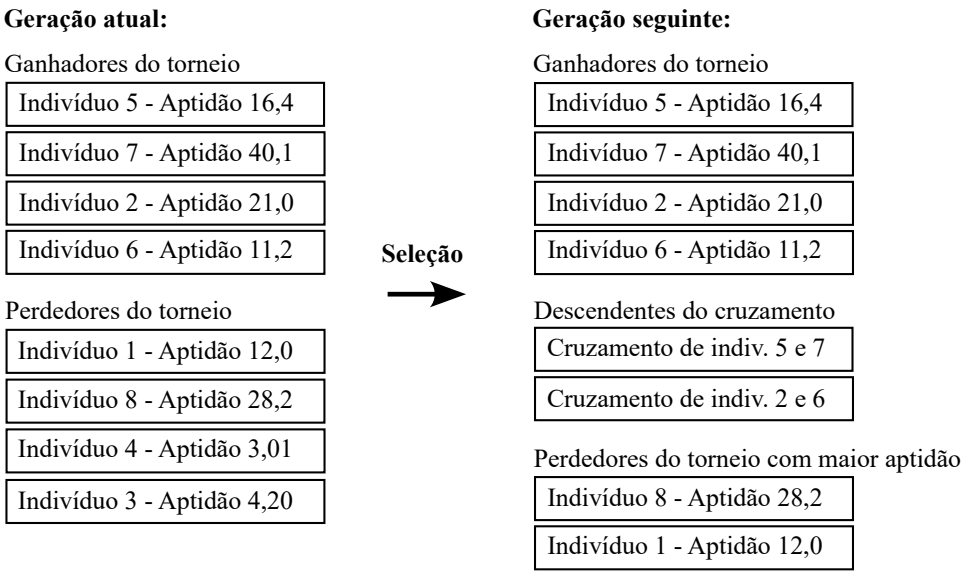


Figura 2.8: Exemplo ilustrativo da etapa de seleção do algoritmo evolucionário.

2.3.6 Biblioteca Python para Implementação do EA

No Apêndice B é apresentado o código Python completo desenvolvido para implementar o algoritmo evolucionário, o qual é um dos módulos principais da ferramenta EAVREF. Como pode ver visto na Figura 2.9, a biblioteca chamada *ea.py* está dividida em duas classes Python principais: i) *circuit*; e ii) *population*. A classe *circuit* é capaz de definir as dimensões de uma referência de tensão, ou em outras palavras, guardar as caraterísticas de um individuo do algoritmo evolucionário. Por outro lado, a classe *population* manipula e define todas as operações da população de circuitos, tais como torneio, cruzamento, seleção, e mutação. Além disso, armazena as N instâncias da classe *circuit*, e é responsável por salvar todas as estatísticas de aptidão de toda a população.

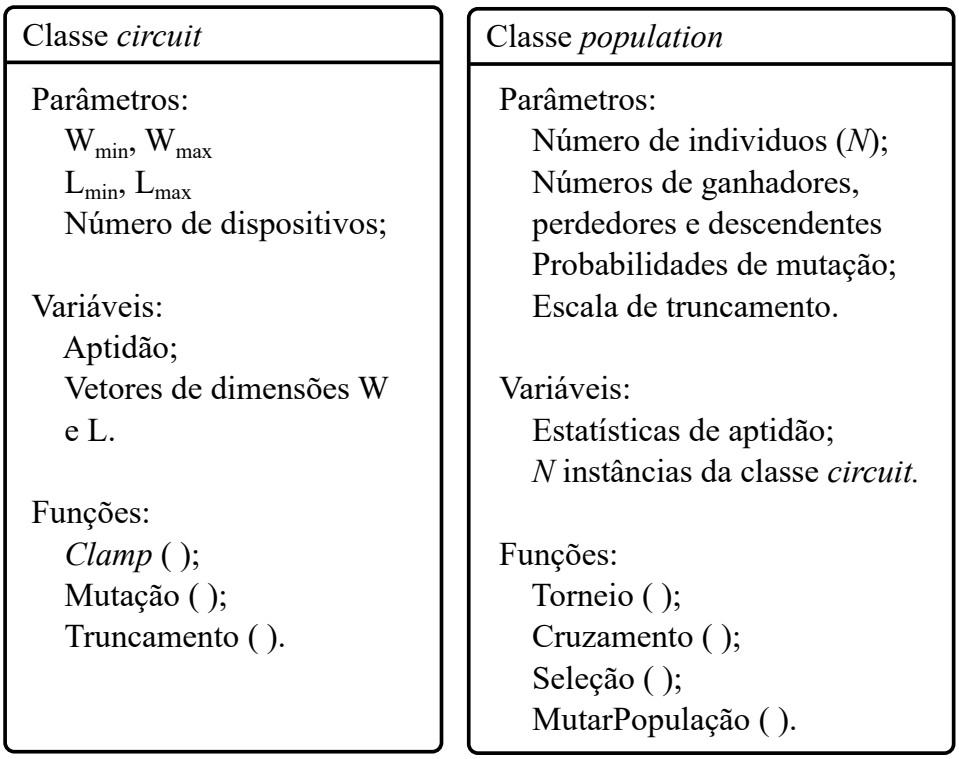


Figura 2.9: Estrutura simplificada da biblioteca Python para implementação do EA.

Como pode ser visto nas classes *circuit* e *population*, foi utilizada uma função de trucamento. Essa função é responsável por limitar a busca de números reais até um certo limiar. Por exemplo, se temos um processo CMOS cujo layout tem precisão de 1 nm, usualmente conhecido como *lambda* mínimo da tecnologia, seria ineficiente o EA procurar números reais menores que $1e-9$. Portanto, a função de trucamento, da classe *circuit*, limita essa busca, e sua escala é configurada diretamente desde a classe *population* para todos os indivíduos ao mesmo tempo.

2.4 Interface Gráfica da Ferramenta

A interface gráfica do usuário (GUI, do inglês, *graphical user interface*) da ferramenta, mostrada na Fig. 2.10, foi programada usando a biblioteca Qt5 do Python. Como pode ser visto no canto superior esquerdo, o usuário pode definir um circuito CMOS VR usando uma descrição padrão de netlist. No canto superior direito, o usuário pode ajustar vários parâmetros de otimização do algoritmo evolucionário, como tamanho da população, número máximo de gerações, número de simulações de Monte Carlo por geração e dimensões máximas e mínimas dos dispositivos. No canto inferior esquerdo, pode ser definida a configuração das simulações, como selecionar o simulador (Hspice, Spectre ou Ngspice), processo CMOS, faixa de temperatura, e temperatura ambiente, entre outros. Uma vez que a descrição do circuito e as configurações sejam as desejadas, o processo de otimização pode ser iniciado pressionando o botão *Start*. As atualizações do progresso do algoritmo são exibidas no quadro de texto de saída no canto inferior direito.

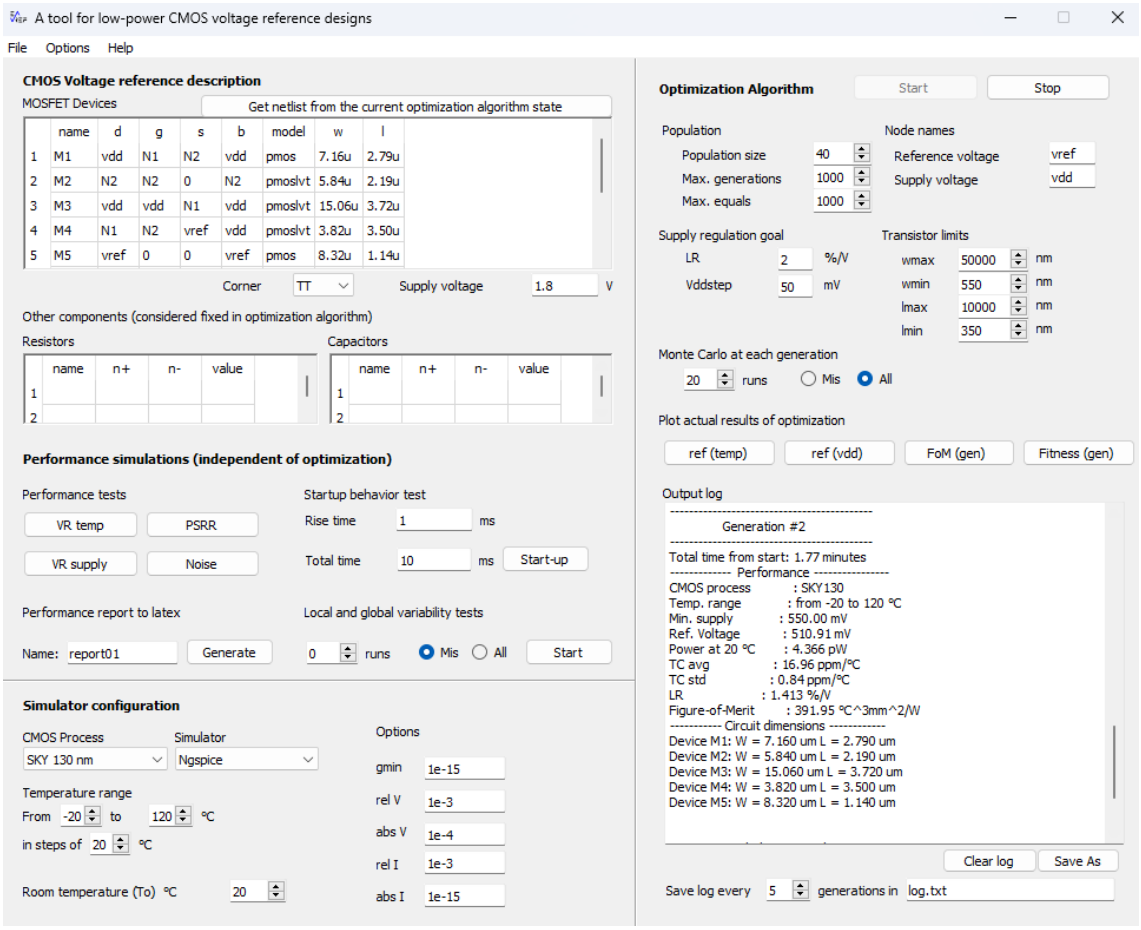


Figura 2.10: Interface gráfica principal de EAVREF.

Outras funcionalidades para projetar manualmente as referências de tensão são fornecidas na interface, tais como as seguintes simulações tradicionais: comporta-

mento da tensão de referência em função da temperatura (*VR temp*) e do fornecimento (*VR supply*); razão de rejeição de fonte de alimentação (PSRR); teste transiente de inicialização da referência (*Start-up*); e comportamento de variabilidade de fabricação local e global. Além disso, um relatório resultante dessas simulações pode ser gerado em LaTeX.

2.5 Outras Funcionalidades de EAVREF

2.5.1 Simulações Manuais para Projeto VRs

Nas seções anteriores foi apresentado o funcionamento da ferramenta para otimizar topologias de VR usando o algoritmo evolucionário proposto. No entanto, muitas vezes é desejado testar circuitos ideais manualmente pelo projetista, ou até mesmo testar manualmente o circuito otimizado em condições especificas. Para isso, como pode ser observado na Figura 2.11, foi programada uma seção da ferramenta, baixo o nome de *Performance simulations*, na qual podem ser executadas as simulações tradicionais de VRs de forma simples. Simulações nominais tais como, o comportamento da tensão de referência em relação à fonte de alimentação (Seta 1) e temperatura (Seta 2), PSRR e ruído intrínseco (Caixa 3). Os resultados detalhados do PSRR e do ruido são mostrados pela janela de *Output log* no canto inferior direito. Uma outra simulação importante em referências de tensão, é a capacidade da referência em iniciar o funcionamento após um degrau de tensão de alimentação, conhecido como teste *Startup*, e indicado com a Seta 4. O comportamento do circuito em relação às incertezas do processo de fabricação pode ser testado como indicado pelas Setas 5 e 6, onde pode ser considerado unicamente descasamento (*Mis*), ou também, descasamento e processo simultaneamente (*All*) como costuma ser configurado em outros simuladores tradicionais. Também pode ser ajustado o número de rodadas Monte Carlo, M = 100 no caso do teste exemplo da Figura 2.11.

2.5.2 Geração Automática de Relatório LaTeX

Todas as figuras das simulações de performance, junto com a descrição do circuito em formato *Netlist*, podem ser documentadas rapidamente em um relatório LaTeX, gerado automaticamente por EAVREF. Para isso o usuário deve ter colocado previamente a descrição do circuito no canto superior esquerdo, e apertar o botão *Generate*, como indicado na Caixa 7 da Figura 2.11.

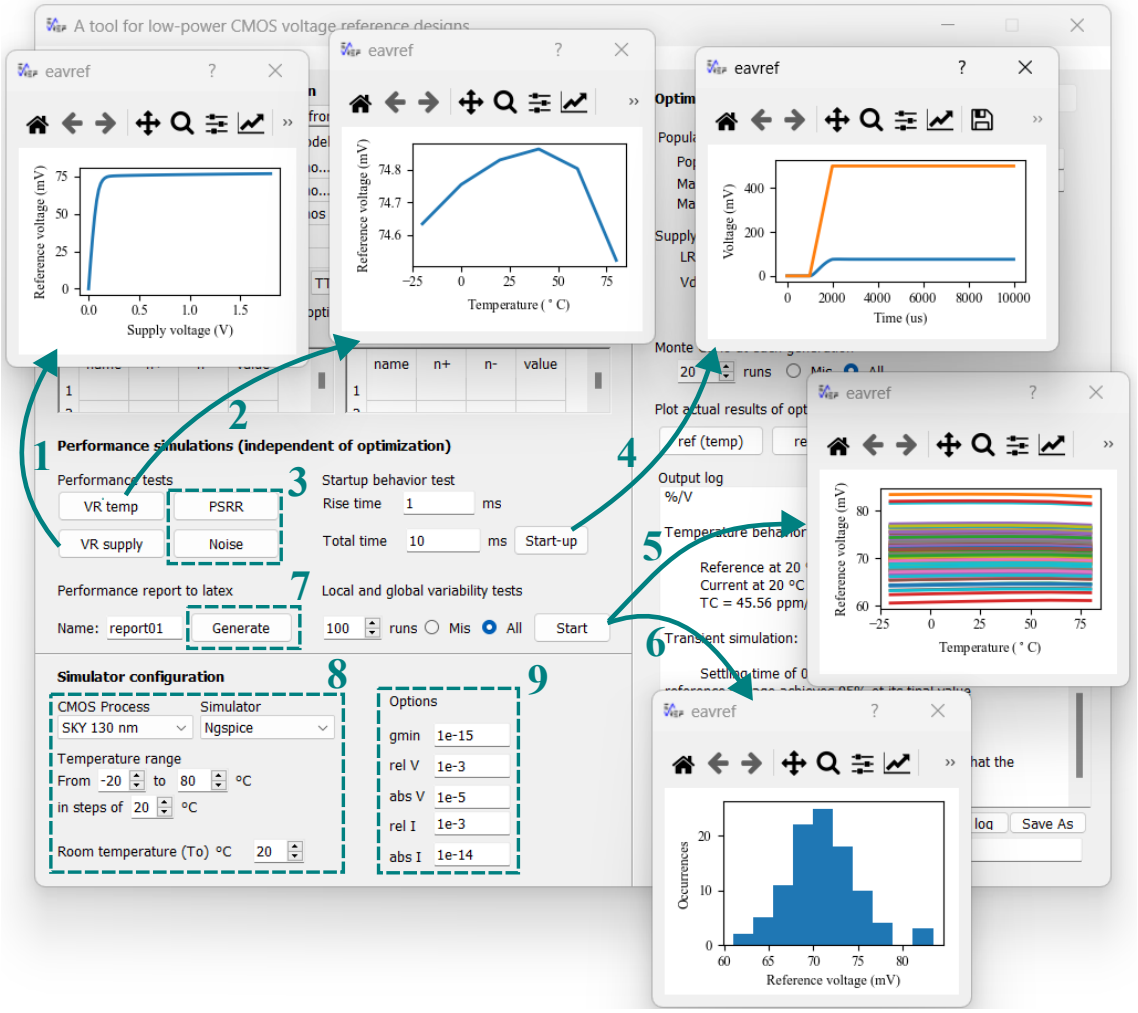


Figura 2.11: Outras funcionalidades de EAVREF para projeto manual de referências de tensão CMOS.

2.5.3 Configurações Avançadas do Simulador

Como pode ser visto na Caixa 8 da Figura 2.11, a interface apresenta a possibilidade de alterar o processo CMOS que deseja ser utilizado, o simulador, faixa de temperatura, passo de temperatura[1], e temperatura ambiente. Na Caixa 9 podem ser vistas algumas configurações mais avançadas do simulador, tais como, a condutância mínima (*gmin*) que é usada para garantir a convergência numérica e a estabilidade da simulação, a tolerância numérica relativa (*rel V*) e absoluta de tensão (*abs V*), e a tolerância numérica relativa (*rel I*) e absoluta de corrente (*abs I*).

2.5.4 Visualização de *Netlist*

Em muitos casos pode ser de grande ajuda dispor da informação do baixo nível da simulação (*netlist*), seja para entender algum conflito na simulação, ver os nomes do

[1] O passo de temperatura, dependendo dos recursos do computador, interfere no tempo de otimização, já que mais pontos requerem maior poder computacional.

modelos que realmente estão sendo utilizados pelo simulador, entre outros. Como pode ser observado na Figura 2.12, no menu de opções, pode ser aberto um editor de texto próprio da ferramenta para visualizar o *netlist* da ultima simulação executada.

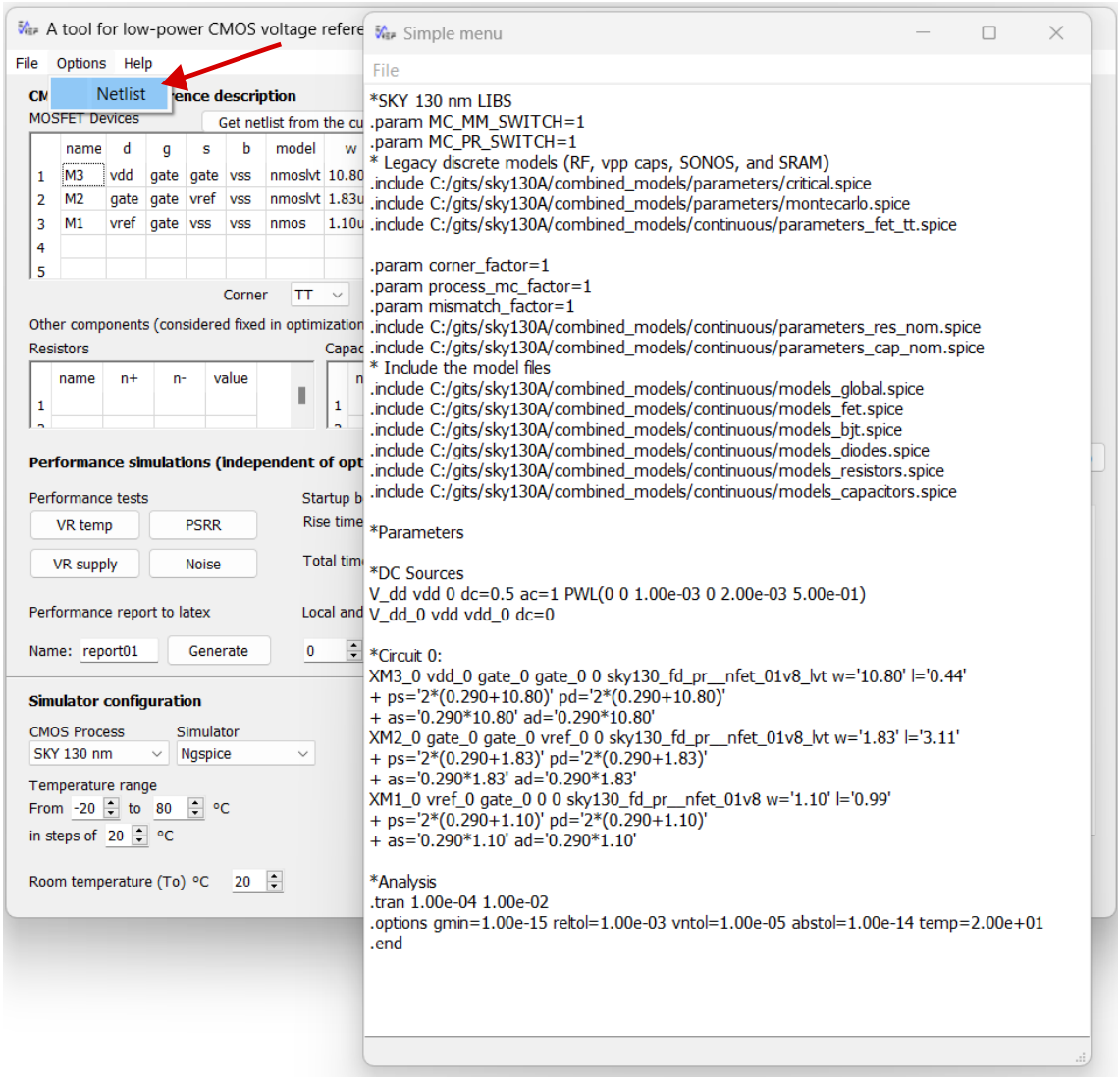


Figura 2.12: Visualização de *Netlist* em editor de texto.

2.6 Detalhes Adicionais da Implementação Python

Como mencionado anteriormente, o código principal da ferramenta foi implementado usando a linguagem de programação Python, a qual, devido à grande variedade de bibliotecas disponíveis, permite integrar simuladores eletrônicos tradicionais, como Ngspice, Hspice e Spectre. Dessa forma, os resultados das simulações podem ser facilmente convertidos em vetores numéricos, permitindo o processamento dos dados para, por exemplo, calcular a função custo do algoritmo, gerar gráficos, fornecer

uma interface gráfica para usuários, entre outras funcionalidades. A seguir, são detalhadas as bibliotecas mais importantes utilizadas neste trabalho.

2.6.1 Biblioteca DeCiDa

DeCiDa (do inglês, *Device and Circuit Data Analysis*) [35] é uma biblioteca Python para simulação de circuitos e análise de dados. Ela oferece funções específicas para os simuladores HSpice, Spectre, NGspice, entre outros. A biblioteca DeCiDa foi escrita em Python e utiliza pacotes como *numpy* e *matplotlib*. A biblioteca facilita a leitura e a análise dos resultados de simulações em diversos formatos e inclui ferramentas para visualização e manipulação de dados.

Uma vez concluída uma simulação, gera-se um arquivo de resultados, cujo formato é específico para cada simulador. Isso é indiferente para a biblioteca DeCiDa, pois precisamos apenas configurá-la para o simulador utilizado. Assim, a biblioteca fornecerá todos os resultados da simulação em vetores *numpy*.

2.6.2 Biblioteca PyQt5

A biblioteca PyQt5 [36] é uma poderosa ferramenta para o desenvolvimento de interfaces gráficas em Python. Baseada no *framework* Qt, ela permite a criação de aplicações com interfaces sofisticadas. Suas funcionalidades incluem a manipulação de *widgets*[2], gestão de layouts e integração com bases de dados. A PyQt5 suporta o desenvolvimento de aplicações multi-plataforma, funcionando em sistemas operacionais como Windows, macOS e Linux. A instalação da biblioteca PyQ5 permite a utilização do editor *Qt Designer*, do qual pode ser visto um exemplo do nosso projeto na Figura 2.13.

Para integrar a programação gráfica Qt com Python, o editor *Qt Designer* gera um arquivo com extensão *.ui* em linguagem XML (do inglês, *Extensible Markup Language*), o qual deve ser transformado para *.py* com a aplicação *pyuic5*, também fornecida pela biblioteca PyQt5.

Uma outra característica fundamental da interface gráfica implementada é o suporte ao processamento multithread, ou, em outras palavras, ao processamento paralelo. A biblioteca *PyQt5.QtCore* oferece este suporte, o que melhora a eficiência e a capacidade de resposta da aplicação gráfica. Por exemplo, enquanto a otimização do algoritmo evolucionário está sendo executada, a ferramenta não fica bloqueada, permitindo a utilização das demais funcionalidades em paralelo, tais como simulações nominais, geração de relatórios, etc.

[2]Os *widgets* podem incluir botões, caixas de texto, barras de rolagem, menus, caixas de seleção, entre outros. Eles são elementos fundamentais nas interfaces gráficas, permitindo que os usuários insiram dados, façam seleções e naveguem pela aplicação de forma intuitiva e visualmente amigável.

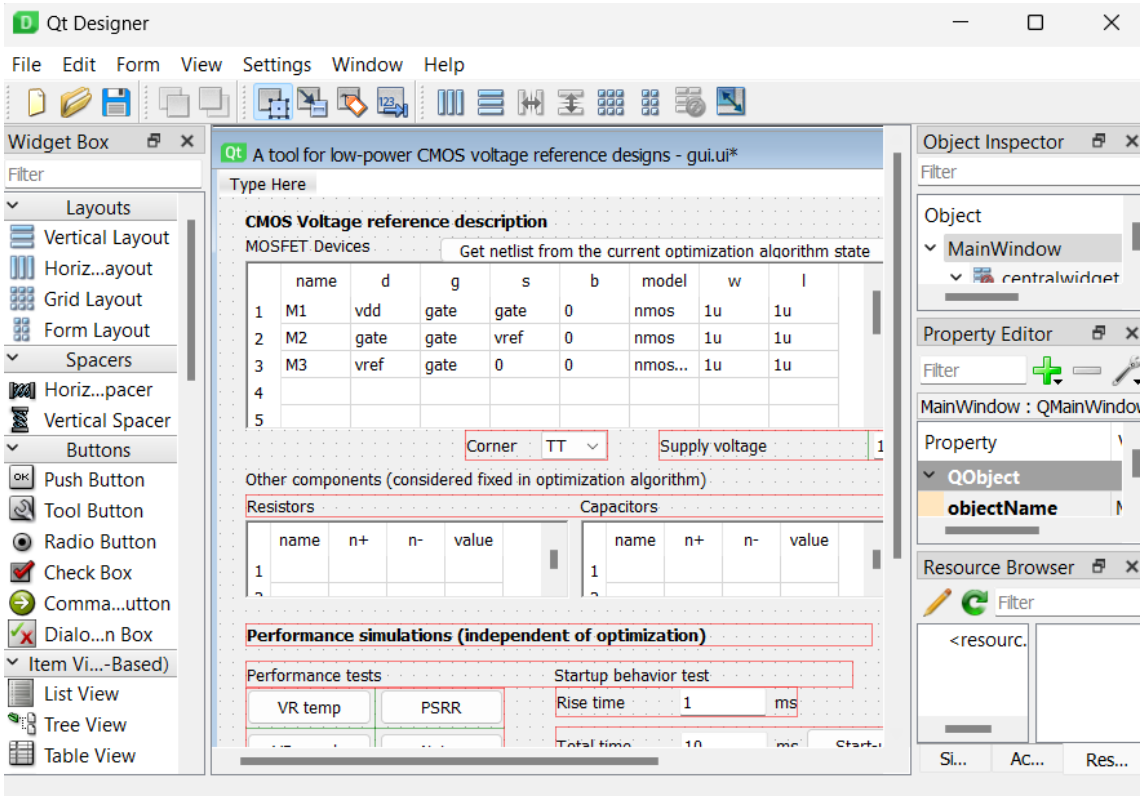


Figura 2.13: Ferramenta *Qt Designer* para programação de interfaces gráficas multi-plataforma.

2.6.3 Biblioteca *os*

A biblioteca *os* (*miscellaneous operating system interfaces*) é um módulo que oferece uma interface para interação com o sistema operacional. Ela está disponível no pacote padrão do Python, sem a necessidade de instalação adicional. A utilização da biblioteca *os* permite a execução de diversas tarefas relacionadas ao sistema operacional, incluindo a manipulação de arquivos e diretórios, a execução de comandos do sistema, por exemplo simuladores eletrônicos, e a obtenção de informações sobre o ambiente do sistema. Na Figura 2.14 pode ser visto o método *run* implementado para executar simulações, e ao mesmo tempo colocar os resultados na pasta *./results*.

```
def run(netlistname="netlist",dir=''):
    if glob.glob(dir+"/hspice*") != []:
        simulator ='hspice'
    elif glob.glob(dir+"/ngspice*") != []:
        simulator ='ngspice'
    #remove all result files with the name "netlistname"
    if config.os_name == 'Windows':
        remove = 'DEL /F /S /Q /A "{0}\\results\\{1}*"'.format(
            config.path,netlistname)
        input = "{0}\\netlists\\{1}.sp".format(config.path,
            netlistname)
        output = "{0}\\results\\{1}".format(config.path,netlistname
            )
        os.system(remove+' > nul')
    elif config.os_name == 'Linux':
        os.system('rm -rf {0}/results/{1}*'.format(config.path,
            netlistname))
        input = "{0}/netlists/{1}.sp".format(config.path,
            netlistname)
        output = "{0}/results/{1}".format(config.path,netlistname)
    #execute the simulation command
    if simulator == 'hspice':
        command = "{0} -i {1} -o {2}".format(dir+'/hspice',input,
            output)
    if simulator == 'ngspice':
        command = "{0} -b -r {1}.raw -o {1}.txt {2}".format(dir+'/
            ngspice',output,input)
    try:
        if config.os_name == 'Windows':
            os.system(command+' > nul')
        elif config.os_name == 'Linux':
            os.system(command+' > /dev/null')
    except:
        print('Simulation command error.')
        return
    return simulator
```

Figura 2.14: Código Python implementado para execução de simulações Ngspice ou Hspice com uso da biblioteca *os*.

Capítulo 3

Projetos de Referências de Tensão CMOS

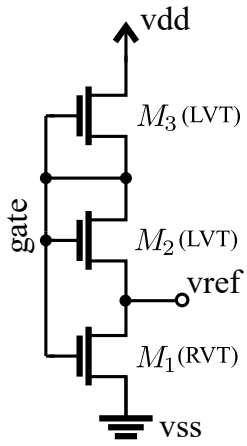
A fim de mostrar a efetividade da ferramenta proposta, neste capítulo são otimizadas, e verificadas com extensas simulações, duas topologias de referências de tensão para baixo consumo, as quais, são apropriadas para a tecnologia SkyWater 130 nm, tendo em vista seus dispositivos MOS disponíveis. A Figura 3.1 mostra estas duas topologias e suas respectivas descrições em linguagem de circuito (*netlist*[1]) na janela da ferramenta EAVREF. Como pode ser apreciado na Figura 3.1(a), a estrutura 3T-GS utiliza unicamente três transistores, e é implementada usando todos os dispositivos NMOS, M_3 e M_2 de tensão limiar baixa (LVT), e M_1 de tensão limiar regular (RVT). Por outro lado, na Figura 3.1(c), é mostrada a estrutura 5T-PMOS, a qual é implementada unicamente usando dispositivos p-MOS. Da mesma forma que a estrutura anterior, esta utiliza combinação dispositivos com tensões limiares regulares (RVT) e baixas (LVT). A estrutura 5T-PMOS apresenta interessantes caraterísticas, já que o fato de poder isolar os substratos dos transistores PMOS independentemente pode melhorar as imperfeições causadas pelo efeito de corpo, conectando os substratos nos terminais de *source* dos transistores.

3.1 Projeto das Topologias 3T-GS e 5T-PMOS usando EAVREF

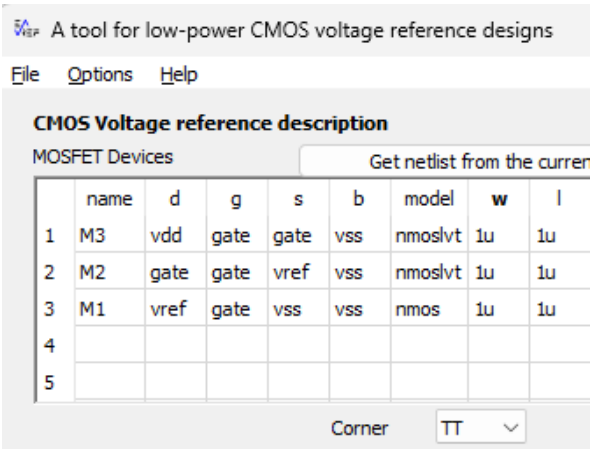
3.1.1 Configuração de Parâmetros para Otimização

Para começar a otimização dos circuitos, primeiramente precisamos configurar os parâmetros do algoritmo, os quais são utilizados para calcular as métricas e aptidão

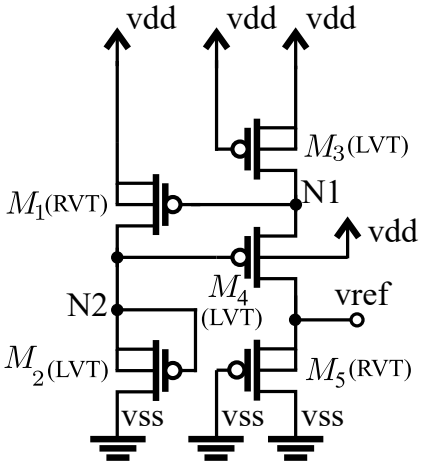
[1] *Netlists* surgiram na década de 1960 como uma forma de descrever circuitos eletrônicos para simulação em computador, substituindo diagramas esquemáticos desenhados à mão.



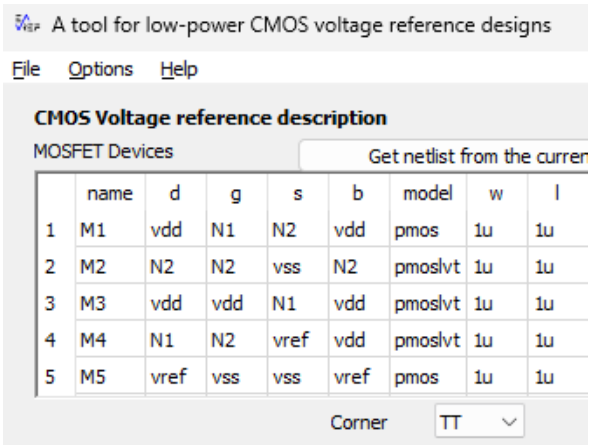
(a) Esquemático 3T-GS



(b) Descrição netlist 3T-GS



(c) Esquemático 5T-PMOS



(d) Descrição netlist 5T-PMOS

Figura 3.1: Duas topologias de VRs ULP apropriadas para a tecnologia SkyWater 130 nm: (a) Esquemático 3T-GS [2]; (b) Descrição de 3T-GS em EAVREF; (c) Esquemático 5T-PMOS [3]; (b) Descrição de 5T-PMOS em EAVREF

correspondentes, simular os circuitos, limitar o domínio de busca, entre outros. As configurações para as duas topologias podem ser vistas na Tabela 3.1. Um bom desempenho no tempo de otimização, observado a partir de vários testes do algoritmo, foi obtido com o uso de 40 indivíduos na população, rodando por 1000 gerações. Não podemos esquecer que o algoritmo proposto utiliza simulações Monte Carlo em todas as gerações para avaliar o coeficiente de temperatura, mas 20 rodadas por geração foi um número aceitável no desempenho da otimização. O passo de V_{DD}, configurado em 50 mV nos dois casos, é utilizado para fazer a varredura de regulação de linha mostrada previamente na Figura 2.1(b). A regulação de linha foi limitada em 2 %/V como pior caso, a qual basicamente determina a tensão de alimentação mínima. As simulações Monte Carlo incluem variações de processo de fabricação e

também de descasamento dos transistores, comumente chamadas, globais e locais, respetivamente.

Tabela 3.1: Configuração de Parâmetros para Otimização

Parâmetro	3T-GS	5T-PMOS
Tamanho da população	40	40
Número de gerações	1000	1000
Regulação de linha (LR)	2 %/V	2 %/V
Passo de V_{DD}	50 mV	50 mV
V_{DD} máximo	1.8 V	1.8 V
Monte Carlo por Geração	20 rodadas	20 rodadas
Tipo de Monte Carlo	Global e local	Global e local
W máximo	50 μm	50 μm
W mínimo	550 nm	550 nm
L máximo	10 μm	10 μm
L mínimo	350 nm	350 nm
Faixa de temperatura	De -20 $^{\circ}C$ até 80 $^{\circ}C$	De -20 $^{\circ}C$ até 120 $^{\circ}C$
Passo de temperatura	20 $^{\circ}C$	20 $^{\circ}C$

3.1.2 Rodadas de Otimização

As Figuras 3.2 e 3.3 mostram cinco execuções independentes do algoritmo evolucionário, realizadas para as topologias 3T-GS e 5T-PMOS, respectivamente, mostrando a aptidão (*fitness*) do melhor individuo da população em cada geração. Como pode ser observado, foi escolhida a melhor rodada de execução em cada caso, ou seja, de maior aptidão, a qual foi computada pelo algoritmo com a Equação (2.2), resultando em 126.4 e 1789, respectivamente, para as topologias 3T-GS e 5T-PMOS. A apreciável diferença na aptidão é resultado das diferentes faixas de temperatura consideradas na otimização, que contribui quadraticamente na Equação (2.2). A 3T-GS foi otimizada com faixa de temperatura de -20°C até 80°C, enquanto a 5T-PMOS na faixa de -20°C até 120°C (ver Tabela 3.1).

As dimensões resultantes das referências otimizadas podem ser vistas na Tabela 3.2.

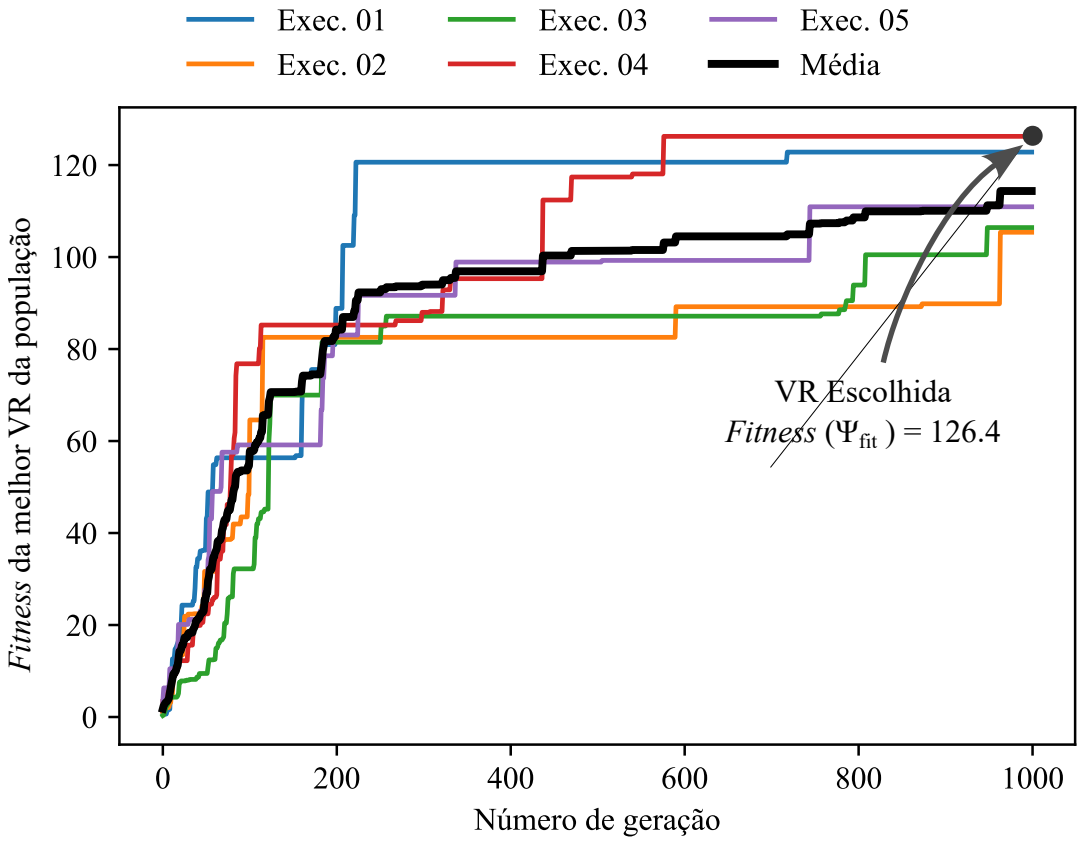


Figura 3.2: Resultado da aptidão do melhor individuo de cinco rodadas independentes de EAVREF para a topologia 3T-GS.

Em relação ao custo computacional, o tempo médio de cada otimização foi de 8 e 22.4 segundos por geração para as topologias 3T-GS e 5T-PMOS, respectivamente, utilizando uma estação de trabalho Intel(R) Core(TM) i7-12700H 2.30 GHz com 16 GB de RAM. Essa diferença no custo computacional se deve ao fato de a topologia 5T-PMOS possuir dois dispositivos a mais que a 3T-GS, o que reflete em 80 dispositivos a mais nas simulações Ngspice quando utilizada uma população de 40 indivíduos no EA.

3.2 Implementação dos Layouts

3.2.1 Layout da Topologia 3T-GS

A Figura 3.4 mostra a implementação do layout da referência de tensão 3T-GS otimizada, a qual ocupa 78.4 μm^2 (8.24 μm x 9.51 μm). Um parâmetro interessante para analisar o desempenho do layout é o fator de preenchimento, o qual é definido como a área total divida pela área de *gate* dos transistores. Neste caso o valor obtido foi 6.81, um pouco maior que 5, considerado em forma de aproximação na função

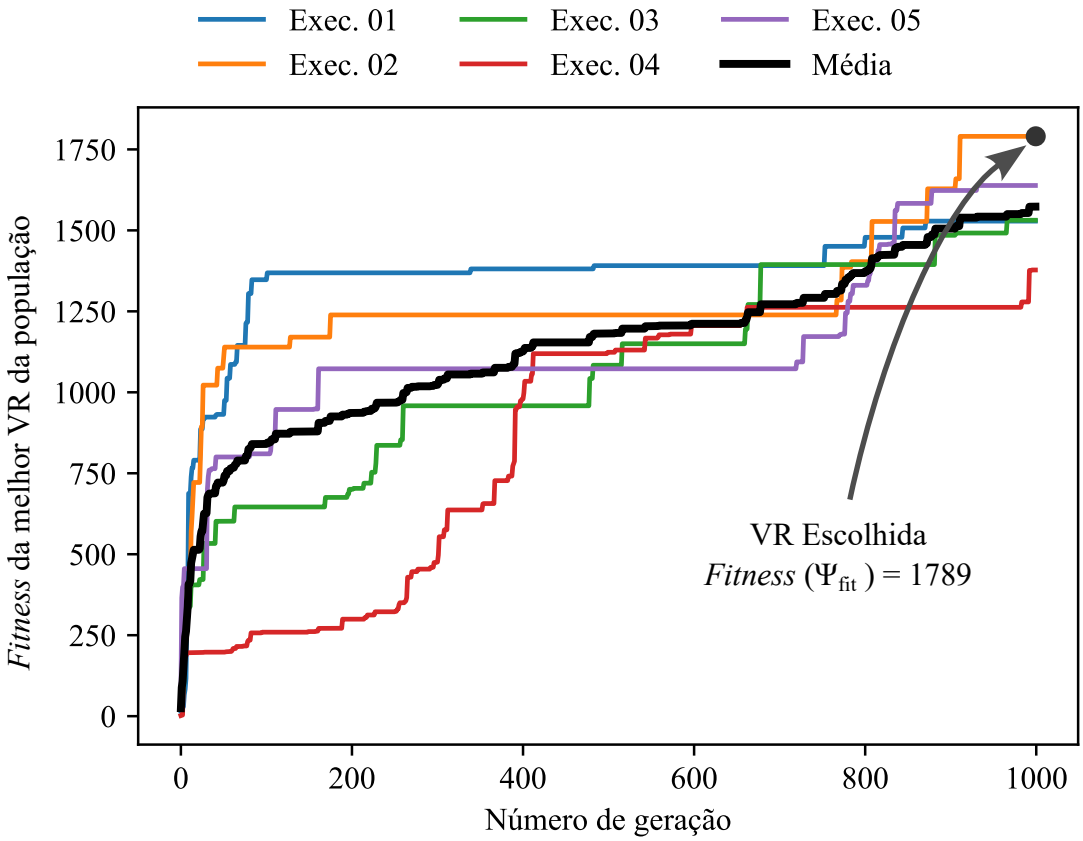


Figura 3.3: Resultado da aptidão do melhor individuo de cinco rodadas independentes de EAVREF para a topologia 5T-PMOS.

de (2.2). No entanto, é importante observar que o layout implementado também ocupa área com anéis de guarda, o qual requer mais espaçamento dos transistores. Em vista da sua largura, a implementação do transistor M_3 foi feita utilizando dois transistores em paralelo de dimensões $W = 6.50$ μm e $L = 420$ nm.

3.2.2 Layout da Topologia 5T-PMOS

O layout da topologia 5T-PMOS pode ser visto na Figura 3.5, e ocupa 78.4 μm^2 (14.38 μm x 22.33 μm). Observe que a distância entre os poços n resulta em espaçamento considerável entre eles, o qual é uma regra de distância mínima exigida pela tecnologia. Além disso, o fator de preenchimento em relação às áreas do gate, incluindo os anéis de proteção, é de 7.65, aproximadamente 12.3% maior em relação à topologia 3T-GS.

Tabela 3.2: Dimensões resultantes das VRs após otimização usando EAVREF

Topologia	W_1/L_1 (μm/μm)	W_2/L_2 (μm/μm)	W_3/L_3 (μm/μm)	W_4/L_4 (μm/μm)	W_5/L_5 (μm/μm)
3T-GS	1.01/1.01 (R)	1.70/2.96 (L)	13.0/0.42 (L)	n/a	n/a
5T-PMOS	3.16/1.32 (R)	1.27/6.64 (L)	4.71/2.34 (L)	5.80/2.20 (L)	4.22/1.32 (R)

(R): dispositivos RVT; (L): dispositivos LVT.

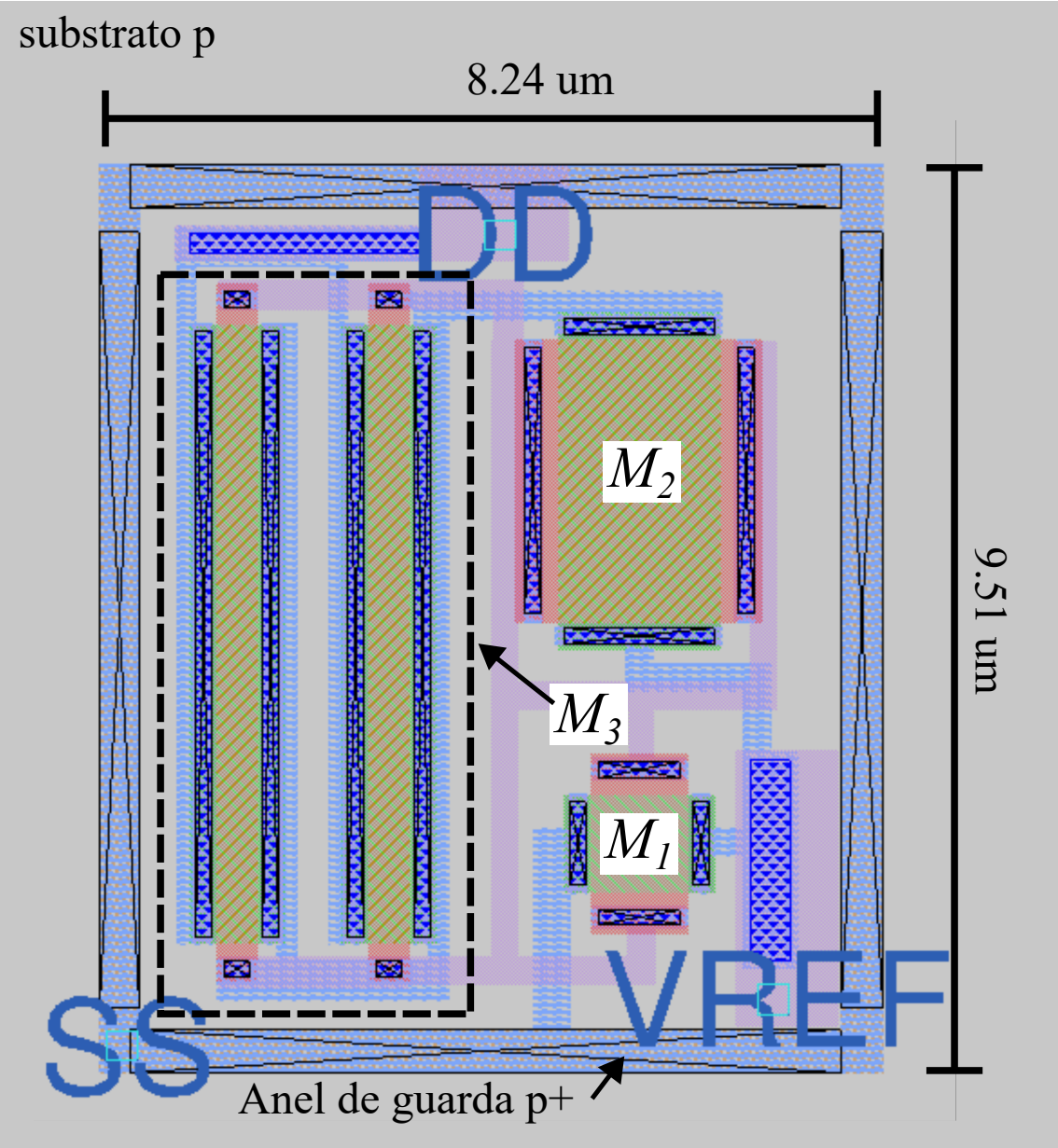


Figura 3.4: Implementação do layout da referência de tensão 3T-GS.

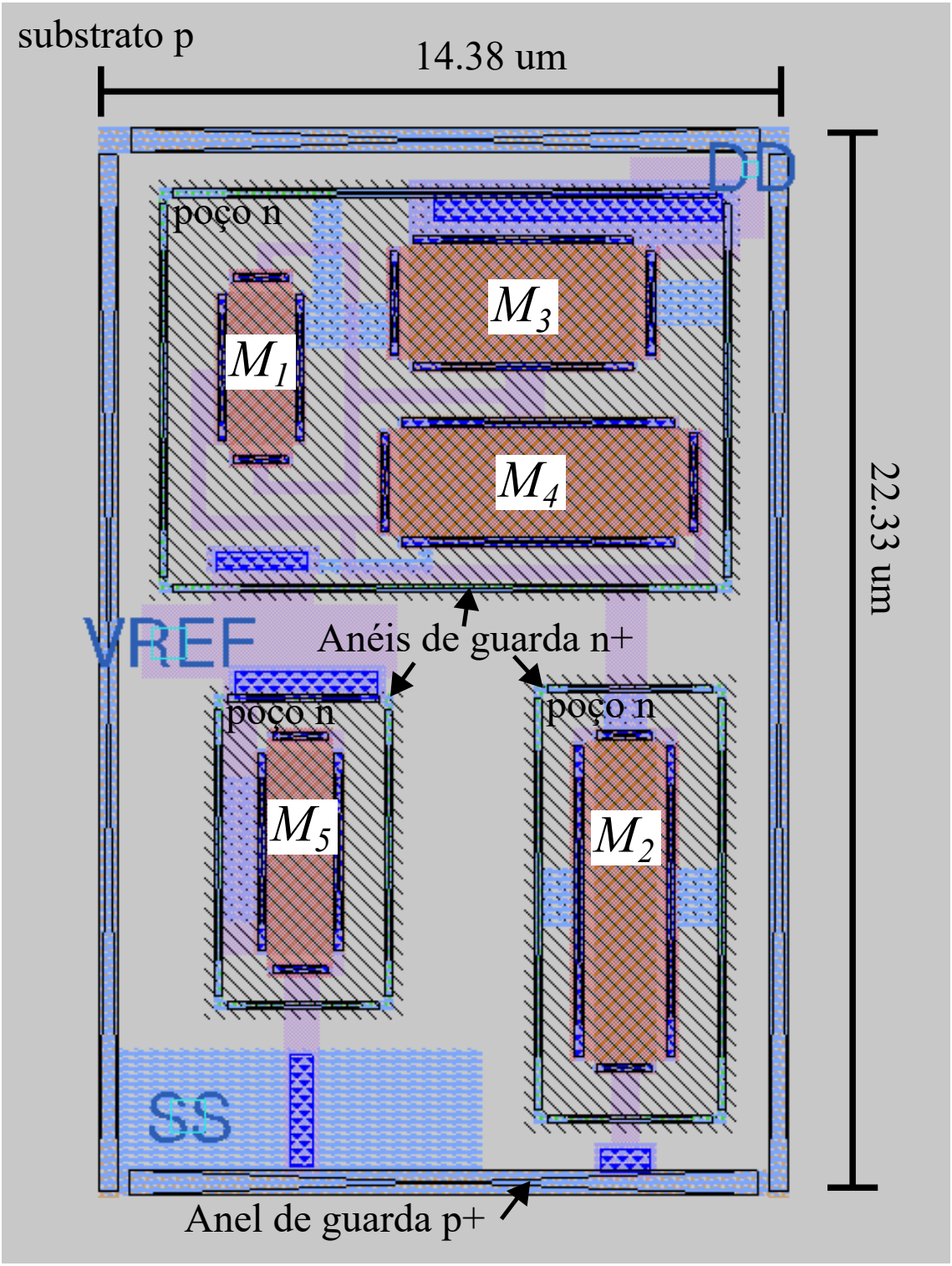


Figura 3.5: Implementação do layout da referência de tensão 5T-PMOS.

3.2.3 Envio para Fabricação de um Protótipo 3T-GS

O programa UNIC-CASS [33], Universalização do Projeto de Circuitos Integrados a partir da CASS, é uma iniciativa estruturada de aprendizado experimental em projeto de circuitos integrados (CI), abrangendo todas as etapas, desde o projeto até o teste. Seu objetivo é aprimorar o conhecimento e a acessibilidade às tecnologias para projetistas em países com renda baixa a média e/ou com poucas oportunidades. Na primeira edição do UNIC-CASS, o grupo de estudantes do CEFET/RJ e UFRJ apresentaram uma proposta, intitulada, *Power Management Circuits for Low-Voltage Harvesting Applications*, sob coordenação do Prof. Luis Fabián Olivera Mederos, a qual foi selecionada para participar. Como pode ser visto no circuito esquemático da Figura 3.6, no projeto foram enviados para fabricação 4 circuitos: um oscilador em anel (Nicole Corradini, IC CEFET/RJ), uma referência de corrente (Italo Bruni, Doutorando UFRJ), um regulador linear LDO (*low-dropout*) (Igor Meirelles, Mestrado CEFET/RJ), e uma referência de tensão CMOS de ultra baixo consumo (Ana Italiano, Mestranda UFRJ), sendo este último, resultado desta dissertação.

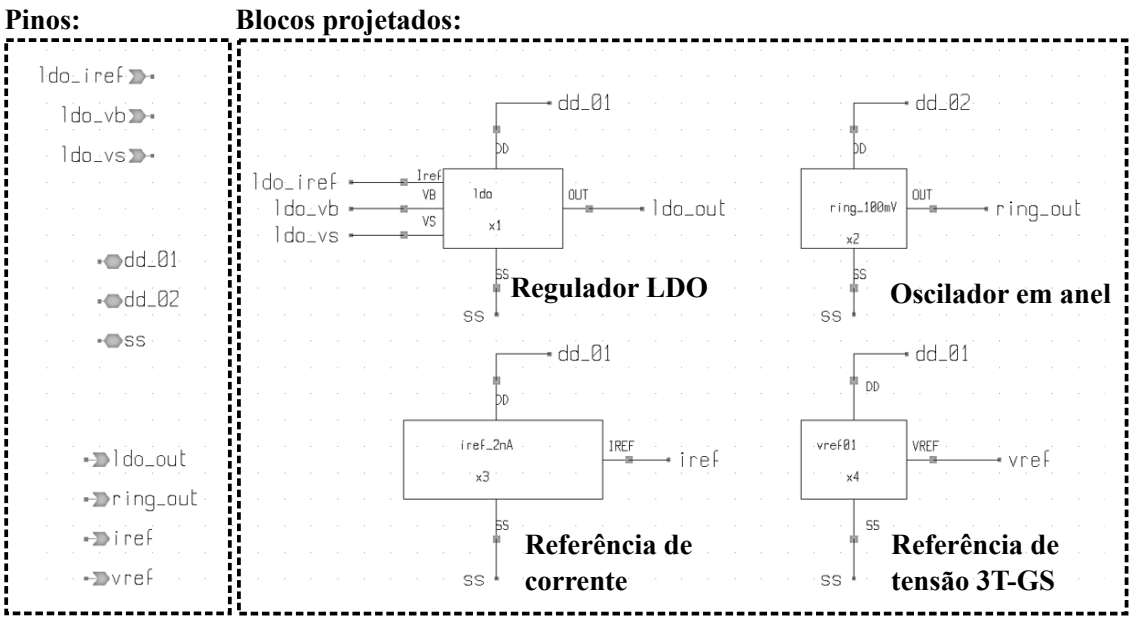


Figura 3.6: Vista superior esquemática do CI enviado para fabricação através do Programa de UNIC-CASS.

A Figura 3.7 mostra a vista final do circuito integrado preparado por três equipes participantes, onde pode ser observado no canto inferior esquerdo, o circuito desenvolvido pelo grupo de estudantes do CEFET e UFRJ.

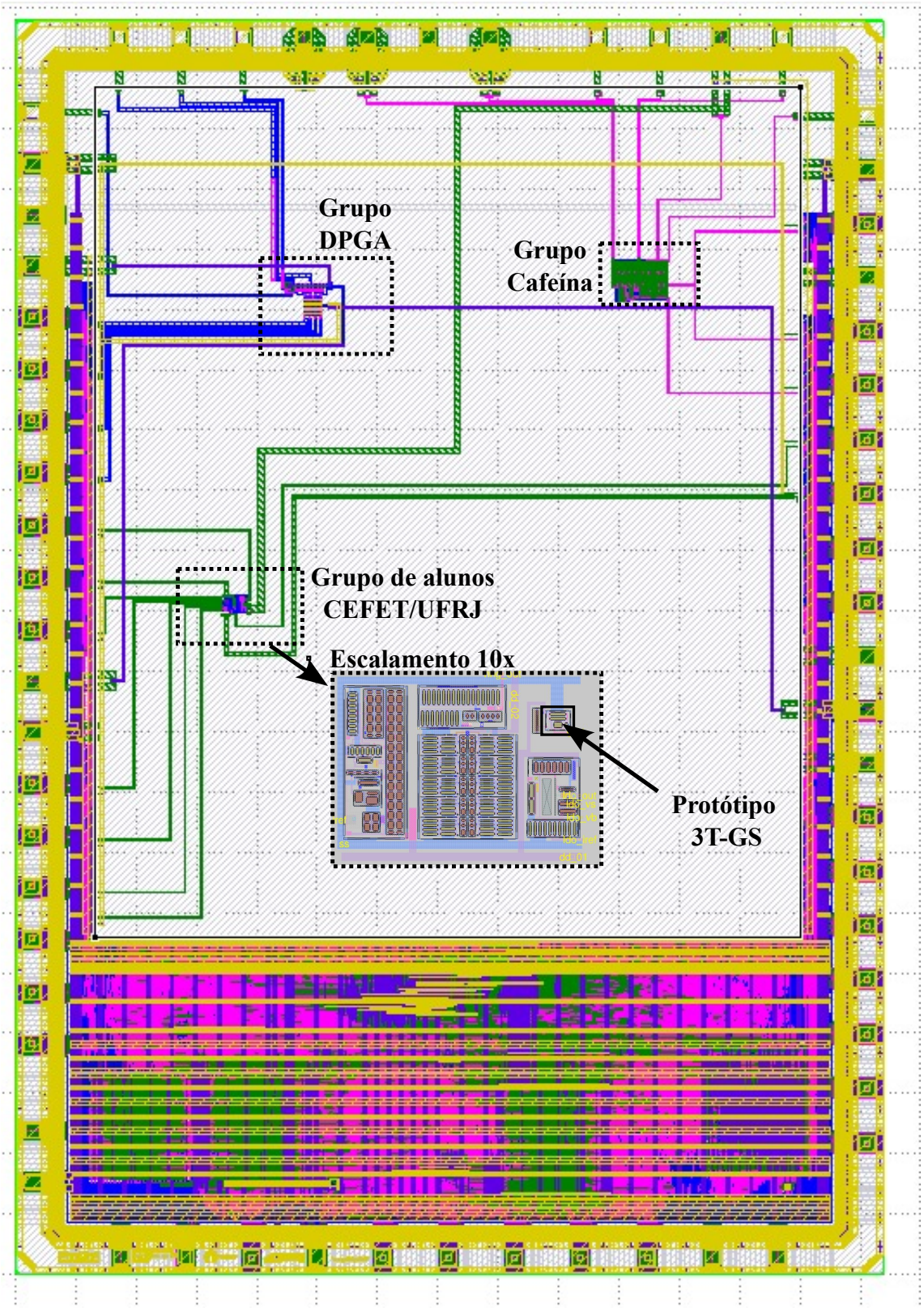


Figura 3.7: Layout final do circuito integrado enviado para fabricação através do programa de UNIC-CASS em tecnologia SkyWater 130 nm.

3.3 Simulações Pós-Layout

Para validar os projetos produzidos, foram realizadas extensas simulações pós-layout, em condições nominais, e também, com variações Monte Carlo.

3.3.1 Tensões de Referência Nominais

A Figura 3.8(a) mostra o comportamento nominal (*corner* TT[2]) da tensão de referência como uma função da temperatura na faixa de −20 °C até 80 °C. Como

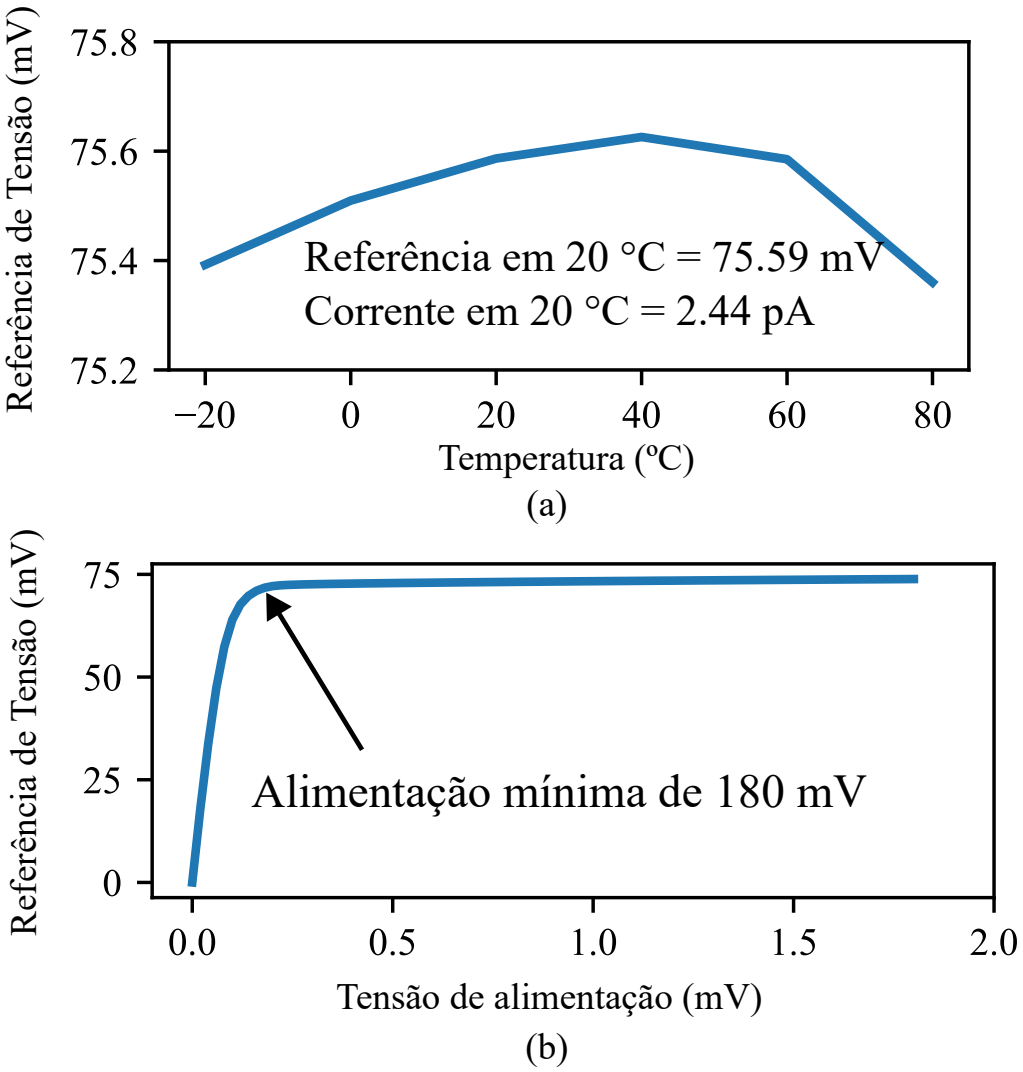


Figura 3.8: Simulações pós-layout do valor de referência da estrutura 3T-GS em condições nominais: (a) em função da temperatura; (b) em função da tensão de alimentação.

pode ser observado, o consumo de corrente em 20 °C é 2.44 pA. Na Figura 3.8(b)

[2] *Corners*: existem 5 pontos bem conhecido para simulações em tecnologias CMOS, considerando cantos extremos da variação global processo: n-típico/p-típico (TT); n-rápido/p-rápido (FF); n-rápido/p-lento (FS); n-lento/p-rápido (SF); n-lento/p-lento (SS).

mostra-se o resultado da tensão de referência quando é variada a tensão de alimentação de 0 V até 1.8 V. Neste resultado podemos observar que a tensão mínima de alimentação para o correto funcionamento é 180 mV, o qual é um bom resultado quando é desejada a caraterística de ultra baixa tensão. O consumo de potência, considerando o consumo de corrente de 2.44 pA e tensão mínima de alimentação 180 mV, resulta em 449 fW.

As mesmas simulações nominais foram realizadas para a estrutura 5T-PMOS, e podem ser vistas na Figura 3.9. A tensão de referência resultou em 514.29 mV, consumindo 3.58 pA em temperatura ambiente (20 °C). Como pode ser visto, a tensão mínima de alimentação é 550 mV. Desta forma, o consumo de potência resultante foi 1.96 pW, consumindo aproximadamente quatro vezes mais do que a estrutura 3T-GS.

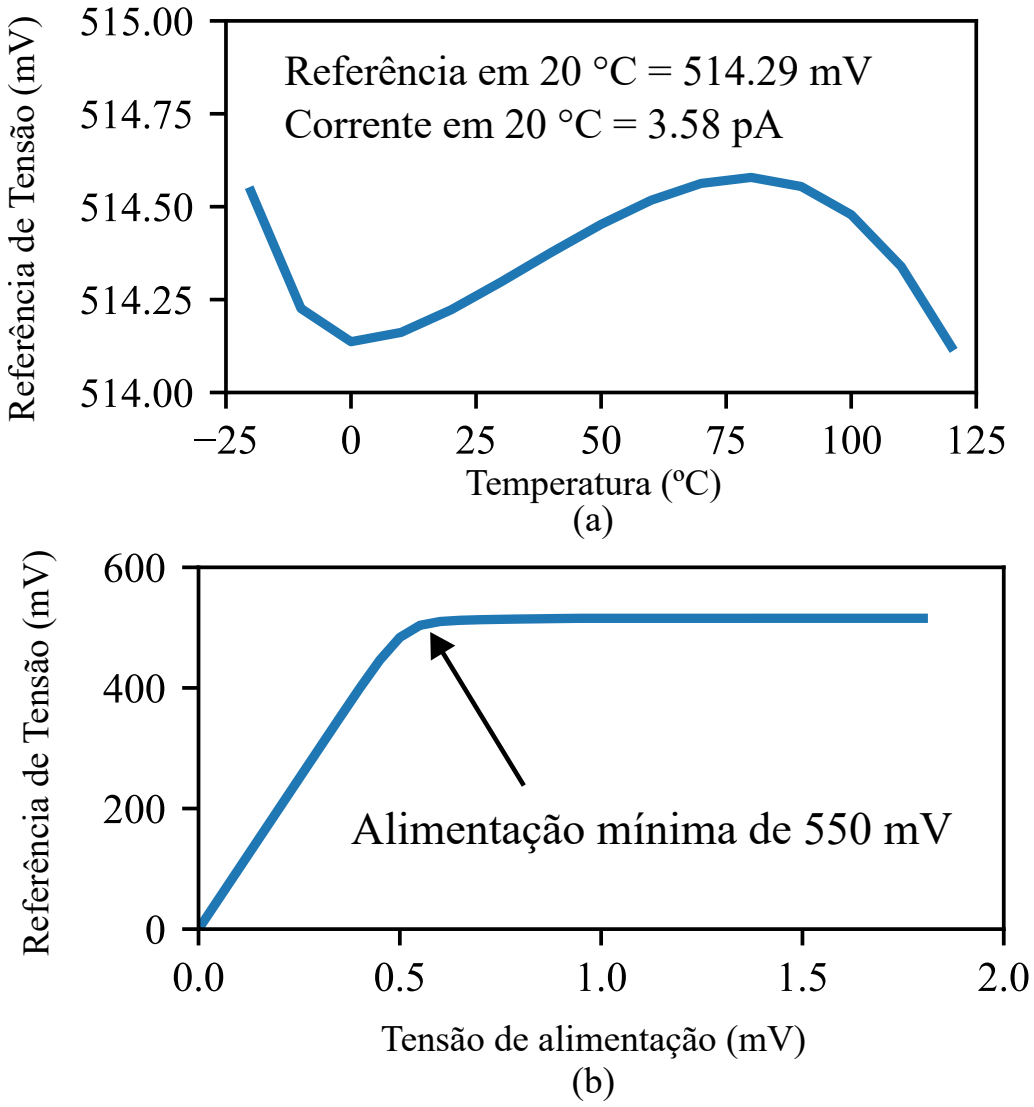


Figura 3.9: Simulações pós-layout do valor de referência da estrutura 5T-PMOS em condições nominais: (a) em função da temperatura; (b) em função da tensão de alimentação.

3.3.2 Comportamento Nominal com Temperatura em Função da Tensão de Alimentação

A Figura 3.10 mostra o resultado de simulação nominal da tensão de referência, da estrutura 3T-GS, em função da temperatura para várias tensões de alimentação. Como pode ser observado, a curvatura se mantém similar, e começa a mudar quando chega na tensão mínima de alimentação de 180 mV.

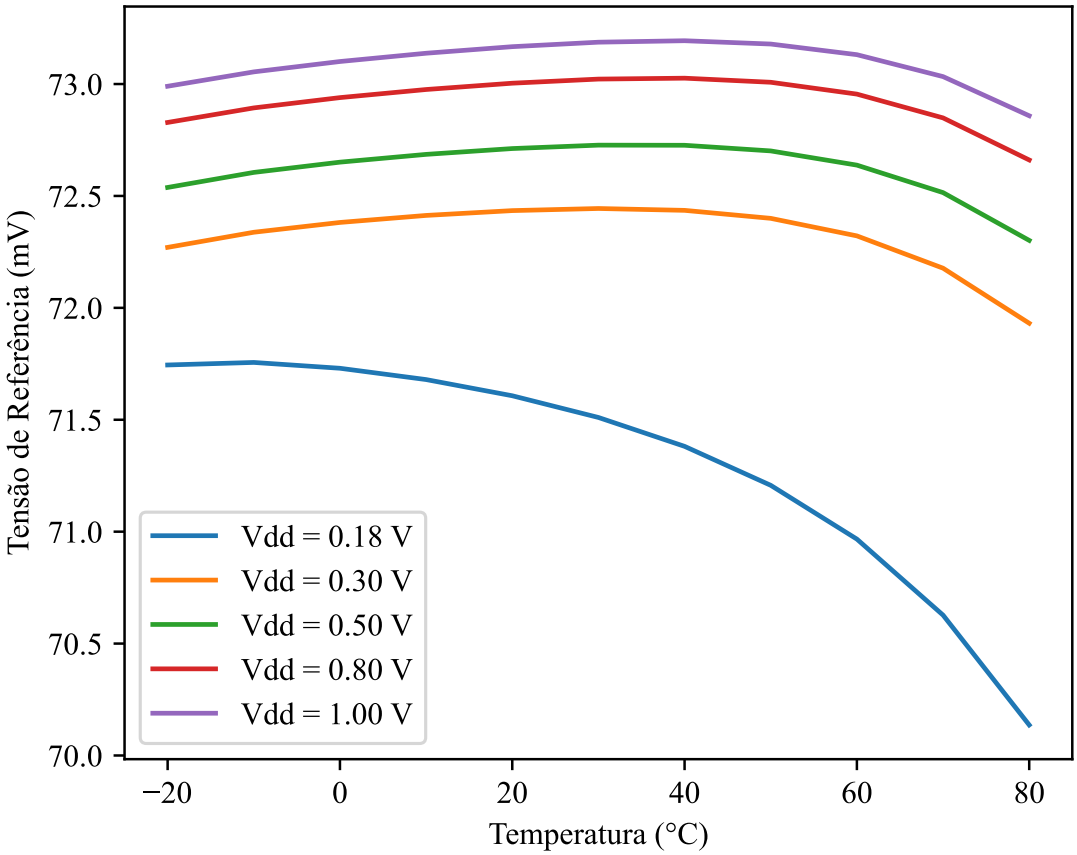


Figura 3.10: Comportamento da tensão de referência com a temperatura da estrutura 3T-GS em função de várias tensões de alimentação.

Por outro lado, a mesma simulação para estrutura 5T-PMOS é mostrada na Figura 3.11. Da mesma forma, na tensão mínima de alimentação, a referência começa a alterar a curvatura, mas ainda se mantendo operacional em 550 mV. Como esperado, em vista da sua excelente regulação de linha mostrada na Fig. 3.9, tensões de alimentação maiores de 1V causam uma diferença pequena na tensão de referência.

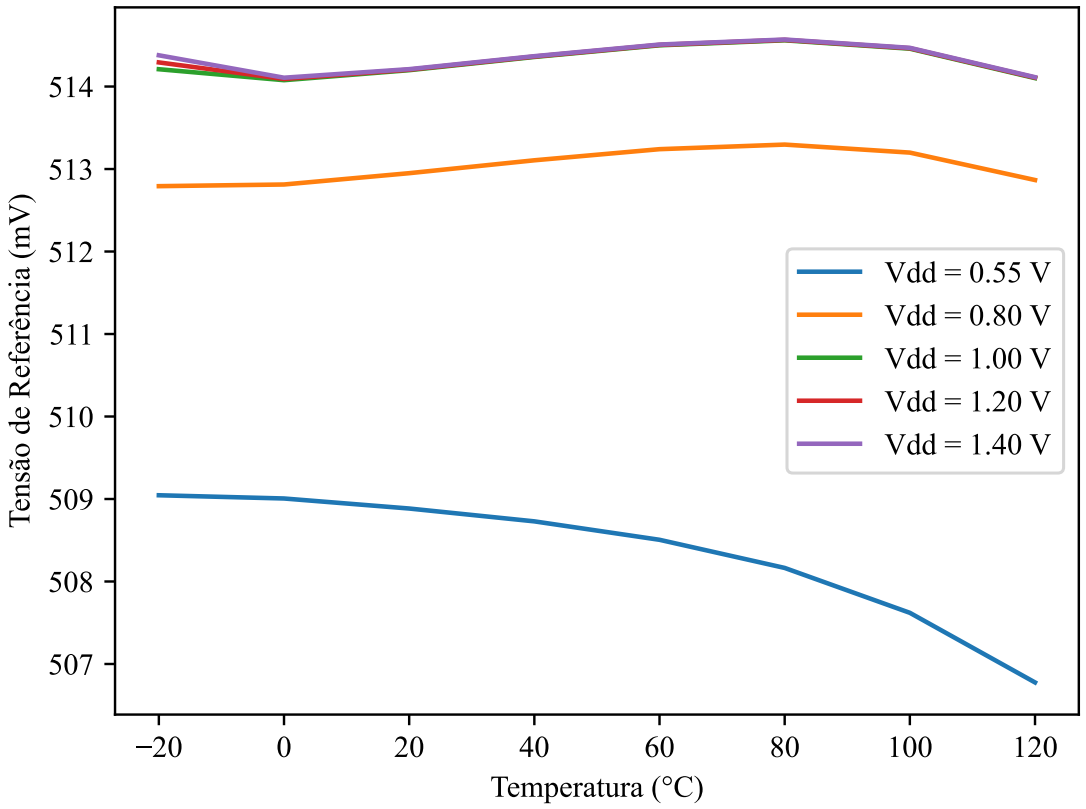


Figura 3.11: Comportamento da tensão de referência com a temperatura da estrutura 5T-PMOS em função de várias tensões de alimentação.

3.3.3 Teste de Inicialização do Circuito

O teste de inicialização é fundamental para garantir o funcionamento correto do circuito a partir do momento em que o sistema é ligado. Para realizá-lo, aplica-se um degrau de tensão na alimentação do circuito e observa-se a resposta transiente das tensões de referência. Um aspecto crucial a ser observado nesse teste é o tempo que a referência leva para se estabilizar, pois esse tempo determinará quando os demais circuitos que dependem da referência estarão prontos para operar corretamente. A Figura 3.12 mostra o resultados de simulações deste teste, nas quais foi aplicado um degrau na tensão de alimentação de com tempo de subida de 0.20 ms, e tensão desde 0V até 300 mV e 700 mV, nos casos das estruturas 3T-GS e 5T-PMOS, respectivamente. Como pode ser observado, os circuitos 3T-GS e 5T-GS demoram 1.25 ms e 3.67 ms, respetivamente, em estabilizar a sua tensão de saída.

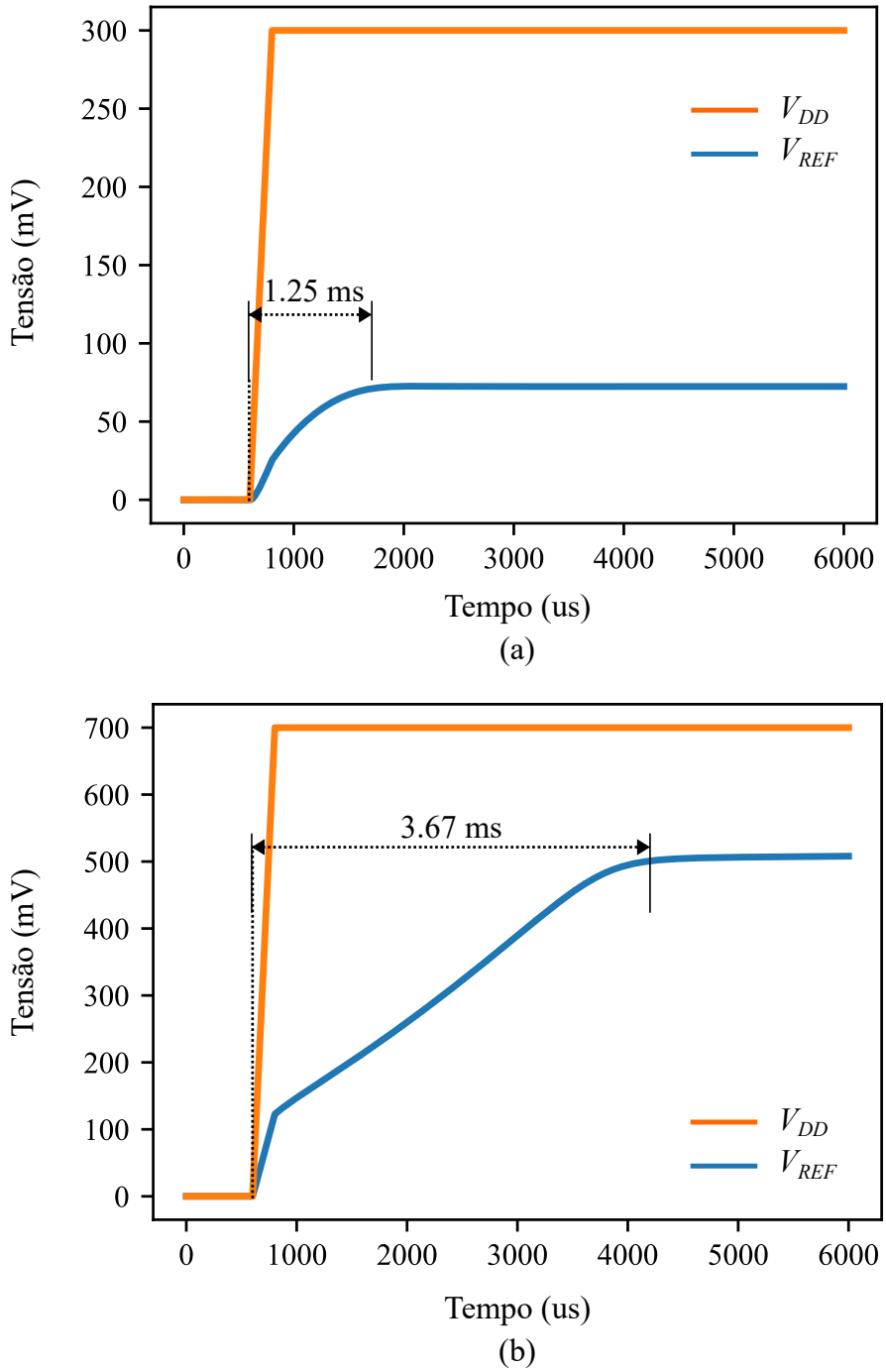


Figura 3.12: Resultados dos testes transientes de inicialização: (a) 3T-GS; (b) 5T-PMOS.

3.3.4 Variações com Processo de Fabricação

A Figuras 3.13(a) e 3.14(a) mostram os histogramas resultantes das simulações Monte Carlo da tensão de referência em temperatura nominal (20°C), os quais resultaram em 73,74 mV (desvio padrão de 3.79 mV) para a estrutura 3T-GS, e 514.29 mV (desvio padrão de 3.72 mV) para a estrutura 5T-PMOS. Esses valores correspondem a variações relativas de 5.1% e 0.7% pra 3T-GS e 5T-PMOS, respetivamente, o qual é um excelente resultado no caso da 5T-PMOS quando comparado com valores da literatura.

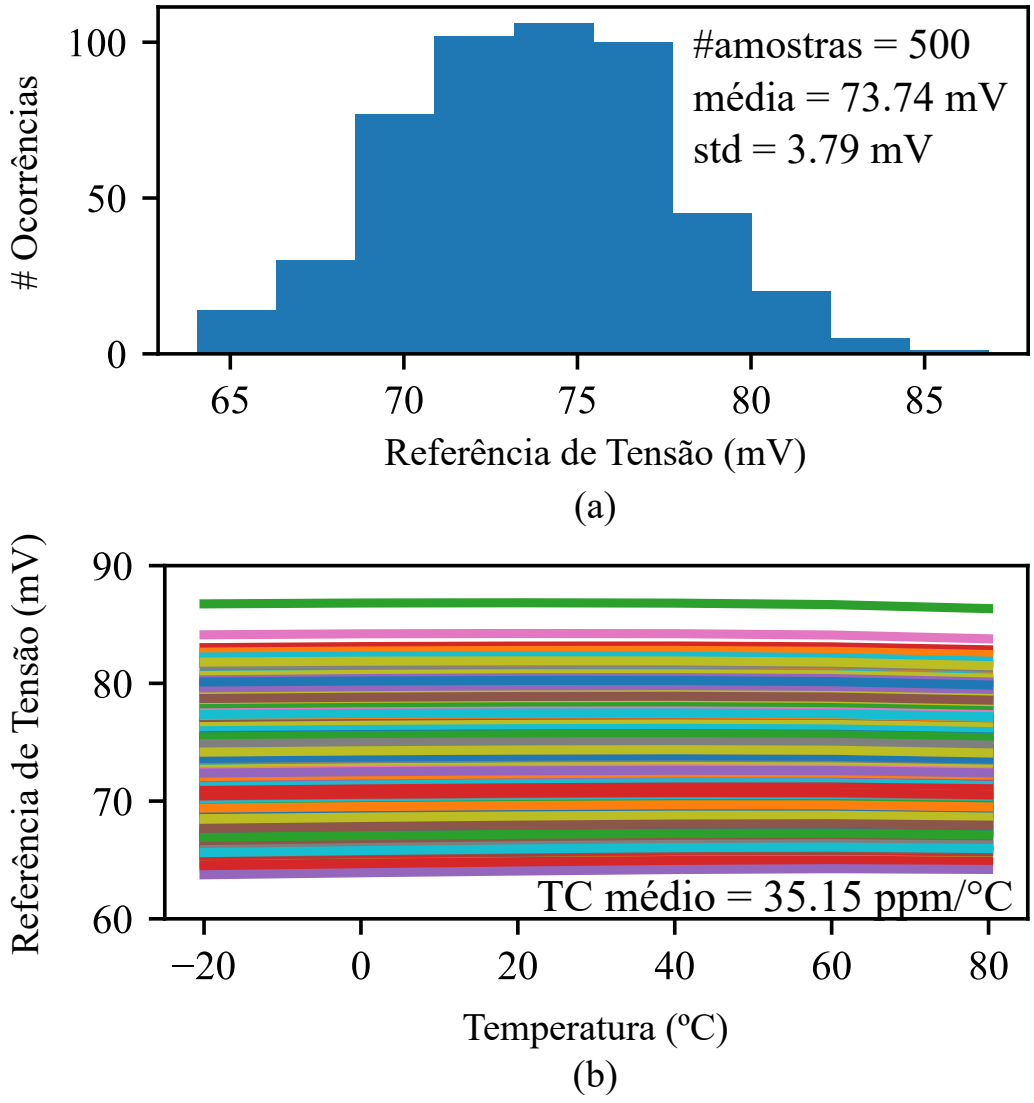


Figura 3.13: Simulações Monte Carlo pós-layout da estrutura 3T-GS: (a) número de ocorrências em função da tensão nominal em 20 °C; (b) tensão de referência em função da temperatura.

Adicionalmente, nas Figuras 3.13(b) (3T-GS) e 3.14(b) (5T-PMOS) são mostrados os resultados de 500 rodadas de simulações Monte Carlo dos comportamentos

da tensões de referência variando a temperatura. Deve-se observar que os TCs, nos quais foram consideradas variações de processo e descasamento, mostraram excelentes valores médios de 35,15 ppm/°C (3T-GS) e 6,99 ppm/°C (5T-PMOS), mantendo seus desvios padrão em 7,06 ppm/°C (3T-GS) e 1,10 ppm/°C (5T-PMOS). Tais resultados de TC refletem duas características importantes do fluxo de projeto

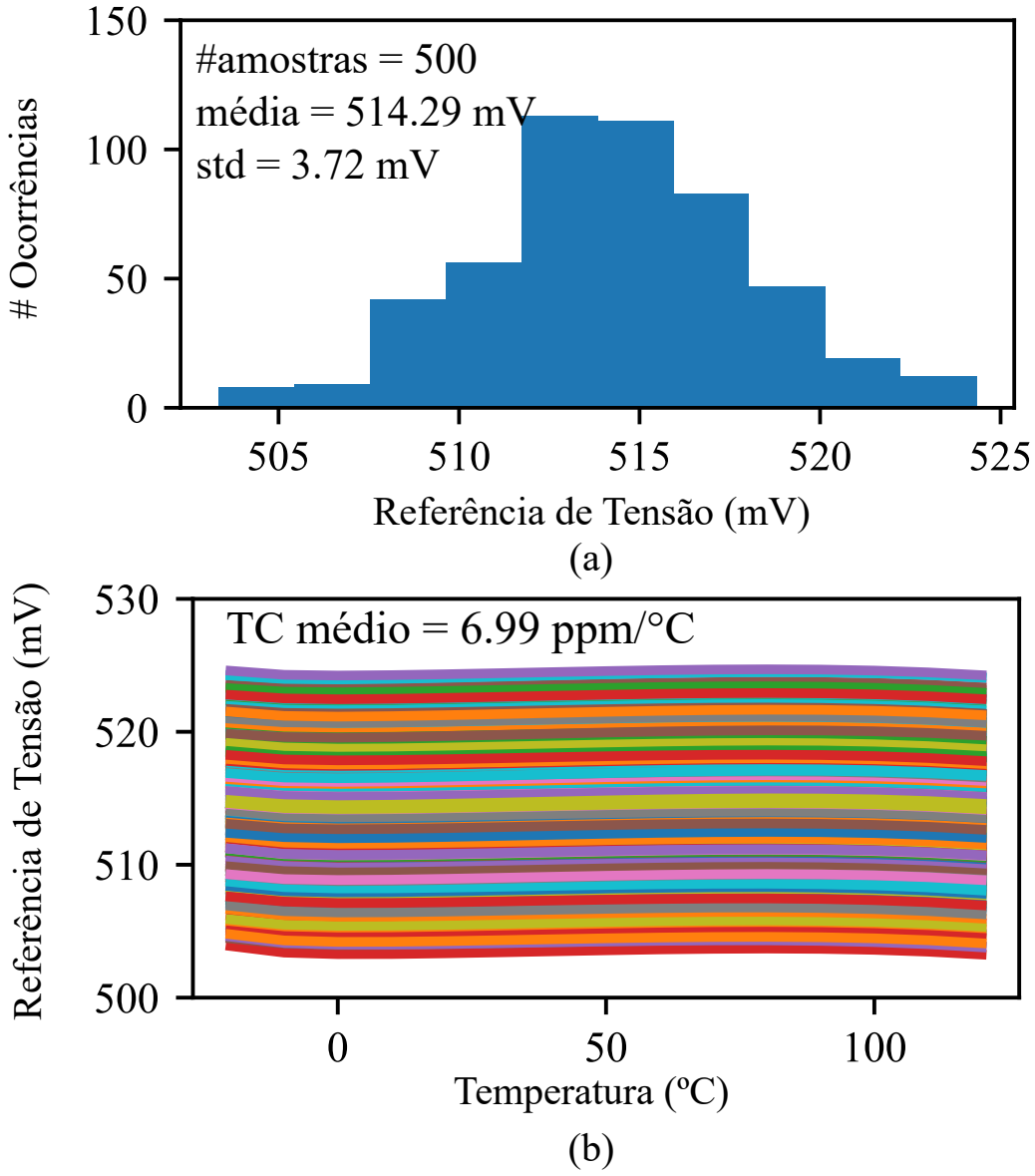


Figura 3.14: Simulações Monte Carlo pós-layout da estrutura 5T-PMOS: (a) tensão nominal em 20 °C; (b) tensão de referência em função da temperatura.

proposto. Primeiro, o uso de simulações de Monte Carlo na evolução do EA pode levar a projetos robustos, que podem lidar com a variabilidade do processo e do descasamento. Segundo, as topologias 3T-GS e 5T-PMOS são boas escolhas porque geram tensões de referência a partir da diferença entre dispositivos do mesmo tipo, NMOS (3T-GS) e PMOS (5T-PMOS), aproveitando os dispositivos LVT e RVT.

3.3.5 Comparação do Desempenho com o Estado da Arte

A Tabela 3.3 mostra o resumo do desempenho das simulações pós-layout em comparação com os projetos de baixo consumo de potência publicados no estado da arte da literatura. Como apresentado inicialmente em [3] e confirmado neste trabalho, a topologia 5T-PMOS mostra excelente comportamento na regulação de linha (0.0025 %/V). No entanto, a otimização EAVREF teve a capacidade de reduzir o consumo de corrente e a mínima tensão de alimentação em comparação com o que foi relatado em [3].

Tabela 3.3: Desempenho comparado com projetos no estado da arte

Topologia	Processo	$V_{DD,\text{min}}$ (mV)	LR (%/V)	PSRR[a] (dB)	$V_{REF,\text{avg}}/V_{REF,\text{std}}$ (mV)	Potência (pW)	$TC_{\text{avg}}/TC_{\text{std}}$ $\left(\frac{\text{ppm}}{°C}\right)$	Faixa de temp. (°C)	Area (μm^2)
Projetos de baixo consumo produzidos por EAVREF									
3T-GS	130 nm	180	1.7	−55.40	73.74 / 3.79	0.45	35.15 / 7.06	−20 to 80	72.8
5T-PMOS	130 nm	550	0.0025	−38	514.29 / 3.72	1.97	6.99 / 1.10	−20 to 120	321
Projetos de baixo consumo publicados na literatura									
3T-GS [37][b]	65 nm	400	0.47	n/a	342.8 / 11.01	0.420	252 / 85.7	−40 to 60	104
5T-PMOS [3]	180 nm	1000	0.0024	−62.0	437.9 / 33.1	101	35.3 / 15.1	0 to 120	7520
4T-DGG [13][b]	130 nm	120	1.62	−38.4	8.42 / n/a	0.328	1537 / n/a	0 to 125	900
[38]	180 nm	600	0.00025	−77.0	470.67 / 34.37	61.2	48.87 / 25.3	0 to 120	3780
2T-GG [11][b]	180 nm	150	2.03	−64.0	17.69 / 0.29	26.08	1462.4 / 324	0 to 120	1200

[a] PSRR calculado em 100 Hz. [b] Resultados obtidos com medições experimentais.

A Figura 3.15 apresenta o resultado da FoM em função da potência consumida dos projetos desenvolvidos em comparação com a literatura. Como pode ser observado, os projetos desta dissertação apresentaram consumos de potência da mesma ordem que os trabalhos [13, 37], mas mostraram apreciáveis melhorias nas figuras de mérito, as quais resultaram em 8684.2 e 4434.1 °C/W/mm^2, para os projetos 3T-GS e 5T-PMOS, respectivamente.

Outro resultado importante para comparação com a literatura é mostrado na Figura 3.16, a qual mostra a FoM em função da área que cada projeto ocupa no silício. Como pode ser verificado novamente, os projetos desenvolvidos pela ferramenta EAVREF, 3T-GS e 5T-PMOS, mostram um excelente resultado quando comparado com a literatura em relação à área.

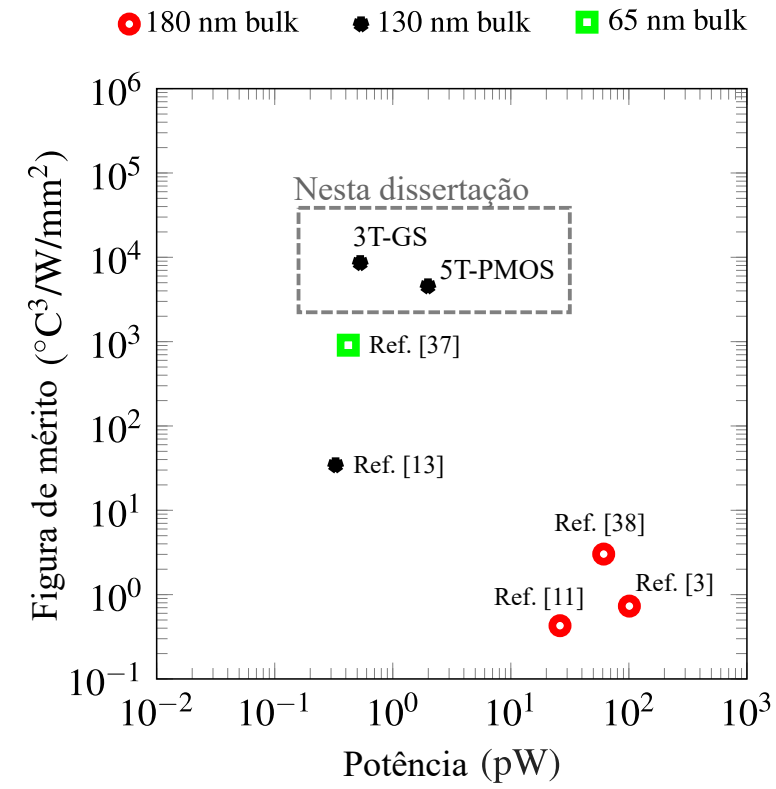


Figura 3.15: FoM em função da potência consumida.

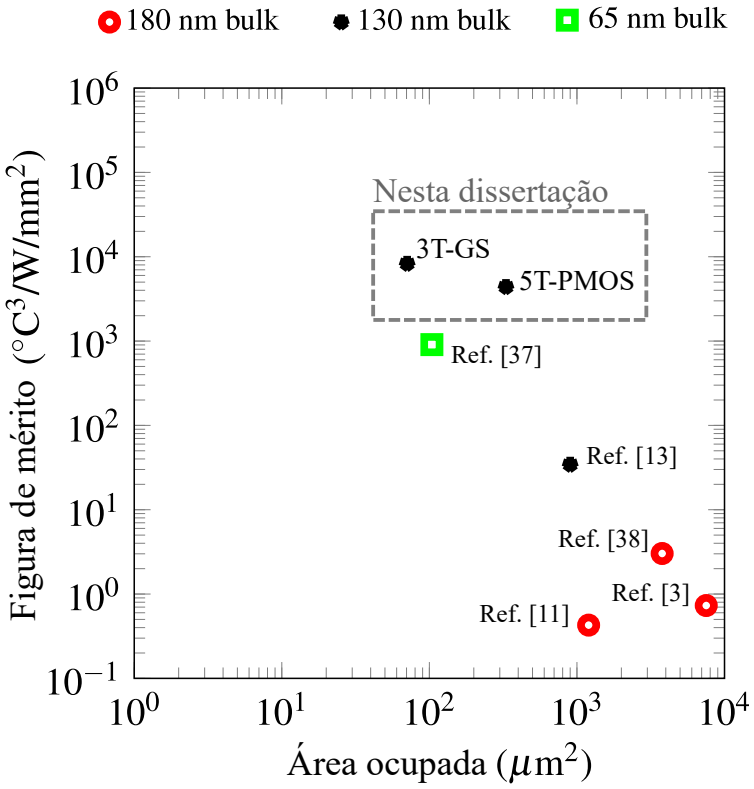


Figura 3.16: FoM em função da área ocupada.

Capítulo 4

Conclusões

Este trabalho de dissertação de mestrado desenvolveu uma ferramenta baseada em um algoritmo evolucionário (EA) para projetos de referência de tensão CMOS de baixo consumo de potência e baixa tensão, que, de acordo com o que foi encontrado na literatura, reduz uma lacuna nas técnicas de projeto para este tipo de circuitos. A combinação de uma aptidão baseada em FoM no algoritmo, juntamente com simulações de Monte Carlo usando variabilidade de processo e descasamento simultaneamente, demostraram capacidade de produzir circuitos robustos e com alto desempenho em relação a suas métricas.

Para validar a ferramenta apresentada, foi realizado o projeto e verificação pós-layout de dois circuitos de referência de tensão CMOS em tecnologia SkyWater 130 nm [39]. Por um lado, a estrutura de 3 transistores (3T) de fonte de corrente auto-polarizada com gate conectado no source (GS), definida como 3T-GS, mostrou os seguintes resultados nas simulações pós-layout: tensão de referência de 73,74 mV (desvio padrão de 3,79 mV), tensão mínima de alimentação de 180 mV, coeficiente de temperatura (TC) de 35,2 ppm/°C (desvio padrão de 7,06 ppm/°C), LR de 1,7 %/V, área ocupada no silício de 72,8 μm^2, faixa de temperatura desde -20 °C até 80 °C e consumo de potência mínimo de 450 fW. Por outro lado, os resultados pós-layout da estrutura de 5 transistores (5T), que utiliza unicamente transistores PMOS (5T-PMOS), mostrou um valor de tensão de referência de 514,29 mV, tensão mínima de alimentação de 550 mV, TC de 6,99 ppm/°C (desvio padrão de 1,1 ppm/°C), LR de 0,0025 %/V, área ocupada no silício de 321 μm^2, faixa de temperatura desde -20 °C até 120 °C e consumo de potência mínimo de 1,97 pW.

A ferramenta desenvolvida, denominada EAVREF, foi disponibilizada para a comunidade de microeletrônica na sua versão 1.0 de forma livre [31], possuindo uma interface gráfica de usuário. A versão 1.0 é compatível com Windows e Linux, tornando a ferramenta de fácil acesso para qualquer usuário. A compatibilidade com Ngpsice, permite que seja utilizada em fluxo de microeletrônica aberto, conjuntamente com processo CMOS 130 nm da SkyWater. Com objetivo de divulgação da

ferramenta para a comunidade de microeletrônica, e também divulgar os resultados desta dissertação na literatura, um artigo científico foi publicado e apresentado no *37th Symposium on Integrated Circuits And Systems Design* (SBCCI) [32].

Um protótipo 3T-GS projetado nesta dissertação, dimensionado com EAVREF, foi enviado para fabricação em circuito integrado através da primeira edição do programa *Universalization of IC Design from Circuits and Systems Society* (UNICCASS) [33], o qual permitirá obter resultados experimentais, e assim, completar o fluxo de projeto em microeletrônica.

4.1 Trabalhos Futuros

Com intuito de continuar esta pesquisa, as seguintes tarefas são propostas como trabalhos futuros desta dissertação:

- O protótipo 3T-GS enviado para fabricação, em tecnologia SkyWater 130 nm, deve ser caraterizado em testes e medições para completar o fluxo de projeto.
- Algumas atualizações já foram pensadas na ferramenta em eventuais próximas versões, tais como aumentar sua compatibilidade com mais processos CMOS, e criar um método de ingresso manual de processos CMOS para permitir o amplo uso da ferramenta pela comunidade de microeletrônica.

Referências Bibliográficas

[1] "Electronic Design Automation Market Size, Share, Growth Drivers, Trends, Opportunities - 2030". 2024. Disponível em: <https://www.marketsandmarkets.com/Market-Reports/electronic-design-automation-market-55791440.html?gad_source=1&gclid=Cj0KCQjw97SzBhDaARIsAFHXUWCo20U1Ibm24m6U_a62wNQ_JgmCsQO1R5iGDqOaaiLjY2ddixPhxXgaAgoZEALw_wcB>.

[2] DE OLIVEIRA, A. C., CAICEDO, J. G., KLIMACH, H. D., et al. "0.3 V supply, 17 ppm/°C 3-transistor picowatt voltage reference". In: *IEEE 7th Latin American Symposium on Circ and Syst*, pp. 263–266, 2016.

[3] XU, H., HU, J., CHEN, Y., et al. "Pico-Ampere Voltage References for IoT Systems". In: *IEEE International Symposium on Circ and Syst*, pp. 1–5, 2019.

[4] WIDLAR, R. "New developments in IC voltage regulators", *IEEE Journal of Solid-State Circuits*, v. 6, n. 1, pp. 2–7, 1971. doi: 10.1109/JSSC.1971.1050151.

[5] LIU, X., LIANG, S., LIU, W., et al. "A 2.5 ppm/°C Voltage Reference Combining Traditional BGR and ZTC MOSFET High-Order Curvature Compensation", *IEEE Trans Circuits Syst II Express Briefs*, v. 68, n. 4, pp. 1093–1097, 2021. doi: 10.1109/TCSII.2020.3027768.

[6] TOBEY, M., GIALIANI, D., ASKIN, P. "Flat-band voltage reference", *U.S.Patent 3975648*, 1976.

[7] XIA, X., XIE, L., SUN, W., et al. "Temperature-stable voltage reference based on different threshold voltages of NMOS transistors", *IET Circ Device Syst*, v. 3, n. 5, pp. 233–238, 2009. ISSN: 1751-858X. doi: 10.1049/iet-cds.2008.0292.

[8] OLIVERA, F., PETRAGLIA, A. "Adjustable Output CMOS Voltage Reference Design", *IEEE Trans Circuits Syst II Express Briefs*, v. 67, n. 10, pp. 1690–1694, 2020. doi: 10.1109/TCSII.2019.2943303.

[9] OLIVERA, F., DA SILVA, L. S., PETRAGLIA, A. "A 120 mV Supply, Triode-Regulated Femto-Watt CMOS Voltage Reference Design", *IEEE Trans Circuits Syst II Express Briefs*, v. 68, n. 2, pp. 587–591, 2021.

[10] ALIOTO, M. "Understanding DC Behavior of Subthreshold CMOS Logic Through Closed-Form Analysis", *IEEE Trans Circuits Syst I Regul Pap*, v. 57, n. 7, pp. 1597–1607, 2010.

[11] ALBANO, D., CRUPI, F., CUCCHI, F., et al. "A Sub- $\boldsymbol{kT/q}$ Voltage Reference Operating at 150 mV", *IEEE Trans Very Large Scale Integr Syst*, v. 23, n. 8, pp. 1547–1551, 2015.

[12] WANG, H., MERCIER, P. P. "A 14.5 pW, 31 ppm/°C resistor-less 5 pA current reference employing a self-regulated push-pull voltage reference generator". In: *IEEE International Symposium on Circ and Syst*, pp. 1290–1293, 2016.

[13] DE OLIVEIRA, A. C., CORDOVA, D., KLIMACH, H., et al. "A 0.12–0.4 V, Versatile 3-Transistor CMOS Voltage Reference for Ultra-Low Power Systems", *IEEE Trans Circuits Syst I Regul Pap*, v. 65, n. 11, pp. 3790–3799, 2018.

[14] OLIVERA, F., PETRAGLIA, A. "Gate leakage compensation technique for self-cascode based voltage references", *Electronics Letters*, v. 56, n. 22, pp. 1174–1176, 2020.

[15] EIBEN, A., SMITH, J. *Introduction to Evolutionary Computing*. 2 ed. Berlin, Heidelberg, Springer, 2015.

[16] BÄCK, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. United Kingdom, Oxford University Press, 02 1996.

[17] DE LIMA MORETO, R. A., THOMAZ, C. E., GIMENEZ, S. P. "Gaussian Fitness Functions for Optimizing Analog CMOS Integrated Circuits", *IEEE Trans Comput Aided Design Integr Circuits Syst*, v. 36, n. 10, pp. 1620–1632, 2017.

[18] LI, Y., WANG, Y., LI, Y., et al. "An Artificial Neural Network Assisted Optimization System for Analog Design Space Exploration", *IEEE Trans Comput Aided Design Integr Circuits Syst*, v. 39, n. 10, pp. 2640–2653, 2020.

[19] ELGABRY, O., HUSSIEN, F., MOHIELDIN, A. N. "Integrated circuit technology characterization and evaluation using automated CAD tool", *AEU Int J Electron Commun*, v. 97, pp. 68–78, 2018. ISSN: 1434-8411.

[20] LBERNI, A., SALLEM, A., MARKTANI, M. A., et al. "Influence of the operating regimes of MOS transistors on the sizing and optimization of CMOS analog integrated Circuits", *AEU Int J Electron Commun*, v. 143, pp. 154023, 2022. ISSN: 1434-8411.

[21] BUDAK, A. F., GANDARA, M., SHI, W., et al. "An Efficient Analog Circuit Sizing Method Based on Machine Learning Assisted Global Optimization", *IEEE Trans Comput Aided Design Integr Circuits Syst*, v. 41, n. 5, pp. 1209–1221, 2022.

[22] FINDER, A., DRECHSLER, R. "An Evolutionary Algorithm for Optimization of Pseudo Kronecker Expressions". In: *2010 40th IEEE International Symposium on Multiple-Valued Logic*, pp. 150–155, 2010. doi: 10.1109/ISMVL.2010.36.

[23] OHMORI, K., KASAI, T. "Logic synthesis using a genetic algorithm". In: *1997 IEEE International Conference on Intelligent Processing Systems (Cat. No.97TH8335)*, v. 1, pp. 137–142 vol.1, 1997. doi: 10.1109/ICIPS.1997.672753.

[24] LI, Y., XIE, G., HAN, Q., et al. "Field-Coupled Nanocomputing Placement and Routing With Genetic and A* Algorithms", *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 69, n. 11, pp. 4619–4631, 2022. doi: 10.1109/TCSI.2022.3197450.

[25] SOARES, C. F. T., DE FILHO, A. C. M., PETRAGLIA, A. "Optimizing Capacitance Ratio Assignment for Low-Sensitivity SC Filter Implementation", *IEEE Transactions on Evolutionary Computation*, v. 14, n. 3, pp. 375–380, 2010. doi: 10.1109/TEVC.2009.2025033.

[26] HESHAM, R., SOLTAN, A., MADIAN, A. "Energy Harvesting Schemes for Wearable Devices", *AEU Int J Electron Commun*, v. 138, pp. 153888, 2021. ISSN: 1434-8411.

[27] ZHANG, J., DAS, R., ZHAO, J., et al. "Battery-Free and Wireless Technologies for Cardiovascular Implantable Medical Devices", *Adv Mater Technol*, v. 7, n. 6, pp. 2101086, 2022.

[28] OLIVERA, F., PETRAGLIA, A. "Ultra-low-power CMOS voltage references: Analysis and optimization regarding technology node", *AEU Int J Electron Commun*, v. 164, pp. 154644, 5 2023. ISSN: 14348411.

[29] MOSIN, S., KISLYAKOV, M. "An In-Pandemic View on the Global Trends in Microelectronic Design and Market". In: *2021 IEEE East-West Design & Test Symposium (EWDTS)*, pp. 1–4, 2021. doi: 10.1109/EWDTS52692.2021.9581014.

[30] REIS, R. "Trends on EDA for low power". In: *2015 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*, pp. 1–4, 2015. doi: 10.1109/NEMO.2015.7415104.

[31] ITALIANO, A., PETRAGLIA, M., OLIVERA, F. "EAVREF: A tool for low-power CMOS voltage reference designs". 2024. Disponível em: <`https://eavref.sourceforge.io`>.

[32] ITALIANO, A., ALMEIDA, L., BRITO, T., et al. "EAVREF: An Evolutionary Algorithm Based Tool for Low-Power CMOS Voltage Reference Designs". In: *2024 Symposium on Integrated Circuits and Systems Design*, pp. 1–4, 2024.

[33] "Universalization of IC Design from CASS (UNIC-CASS) | IEEE CASS". 2024. Disponível em: <`https://ieee-cas.org/universalization-ic-design-cass-unic-cass`>.

[34] MÜHLENBEIN, H., SCHLIERKAMP-VOOSEN, D. "Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization", *Evolutionary Computation*, v. 1, n. 1, pp. 25–49, 1993. doi: 10.1162/evco.1993.1.1.25.

[35] BOOTH, R. "DeCiDa: Device and Circuit Data Analysis". 2020. Disponível em: <`https://pypi.org/project/DeCiDa/`>. Versão 1.1.5.

[36] RIVERBANK COMPUTING LIMITED. "PyQt5: Python Bindings for Qt". 2019. Disponível em: <`https://pypi.org/project/PyQt5/`>. Versão 5.15.10.

[37] WANG, H., MERCIER, P. P. "A 420 fW self-regulated 3T voltage reference generator achieving 0.47%/V line regulation from 0.4-to-1.2 V". In: *IEEE European Solid State Circ Conf*, pp. 15–18, 2017.

[38] XU, H., LU, C., HU, J., et al. "A 28 ppm/°C, 2.54 ppm/v, −77 dB@100 Hz pico-ampere voltage reference for high-end IoT systems", *AEU Int J Electron Commun*, v. 100, pp. 16–24, 2019. ISSN: 1434-8411.

[39] GOOGLE, SKYWATER. "SkyWater SKY130 PDK's documentation". 2024. Disponível em: <`https://skywater-pdk.readthedocs.io/en/main/index.html`>.

[40] VOGT, H., ATKINSON, G., NENZI, P. "Ngspice 42: Open Source Spice Simulator". 2024. Disponível em: <`https://ngspice.sourceforge.io/`>.

Apêndice A

Configurações de EAVREF

Neste apêndice são descritas as configurações rápidas e necessárias para funcionamento EAVREF. Com intuito de compartilhar a ferramenta com a comunidade científica de forma libre, EAVREF se encontra disponível para baixar do repositório da SourceForge [31]. A versão libre da ferramenta utiliza Ngspice como simulador principal [40] e o processo CMOS aberto SkyWater de 130 nm [39]. É importante mencionar que a ferramenta foi programada para ser compatível com os simuladores Spectre e Hspice, podendo ser adaptada para qualquer outro processo CMOS que tenha os modelos dos transistores nestes simuladores. Para o correto funcionamento da ferramenta com Ngspice e SkyWater, o arquivo *config.ini* deve ser configurado da seguinte forma:

```
[DEFAULT]
[simulators]
# -------- HSpice configuration ------------
#for now, hspice is only set to execute in windows os, so, the
#LM_LICENSE and lmgrd.exe server must be running
hspice_bin_path = ''
# -------- NGSpice configuration -----------
ngspice_bin_path = C:/ngspice-42_64/Spice64/bin
# -------- Spectre configuration -----------
#Set "no" for no RHEL linux distributions to execute in
    compatibilty mode
RHEL_linux = yes
# license and executable paths
spectre_LM_LICENSE_FILE = license.dat
spectre_bin_path = ''
[technologies]
# -------- PDK configuration ----------------
sky130_path = C:/gits/sky130A
```

Apêndice A

Biblioteca Python Desenvolvida para o Algoritmo Evolucionário

Neste apêndice é apresentado o código Python desenvolvido para implementar o algoritmo evolucionário proposto. A biblioteca chamada *ea.py* está dividida em duas classes Python principais: i) *circuit*; ii) *population*. A classe *circuit* é capaz de definir as dimensões de uma referência de tensão, ou em outras palavras, guardar as caraterísticas de um individuo do algoritmo evolucionário. Por outro lado, a classe população manipula e define todas as operações da população de circuitos, tais como torneio, cruzamento, seleção, e mutação. O código Python desta biblioteca pode ser visto a seguir:

```
import numpy as np
import numpy.random as rd
import copy as cp
import config

class circuit:
    # lim = [wmin,wmax,lmin,lmax]
    # n_mos = number of mos transistors
    def __init__(self,n_mos=4,lim=[0.5e-6,10e-6,0.5e-6,10e-6],n_res
        =0,lim_res=[1e3,10e3],gridscale=10e-9):
         self.gridscale = gridscale
         self.n_mos = n_mos
         self.n_res = n_res
         self.lim = lim
         self.wmin = lim[0]
         self.wmax = lim[1]
         self.lmin = lim[2]
         self.lmax = lim[3]
```

```
18          self.rmin = lim_res[0]
19          self.rmax = lim_res[1]
20          if n_mos>0:
21              # Logarithm-Uniformy distribution of initial values (
                    best performance)
22              self.Ls = self.grid(np.power(10,np.matlib.rand(n_mos)*(
                    np.log10(lim[3])-np.log10(lim[2]))+np.log10(lim[2]))
                    )
23              self.Ws = self.grid(np.power(10,np.matlib.rand(n_mos)*(
                    np.log10(lim[1])-np.log10(lim[0]))+np.log10(lim[0]))
                    )
24              for i in range(self.n_mos):
25                  self.Ls[0,i] = self.clamp(self.Ls[0,i],self.lmin,
                        self.lmax)
26                  self.Ws[0,i] = self.clamp(self.Ws[0,i],self.wmin,
                        self.wmax)
27          if n_res>0:
28              self.Rs = np.matlib.rand(n_res)*(lim_res[1]-lim_res[0])
                    +lim_res[0]
29          else:
30              self.Rs = 0
31          self.fom = 0
32
33      # Clamping values inside a desired range
34      def clamp(self,value,minv,maxv):
35          return max(min(maxv,value),minv)
36
37      def grid(self,value):
38          return np.round(value/self.gridscale)*self.gridscale
39
40      def mutation(self,prob,percent,mutation_type):
41          k_precision = 4
42          if (mutation_type == 'log'):
43          # Circuit logaritmic-mutation with probability and relative
                percentual force
44              for i in range(self.n_mos):
45                  range_Ls = (np.log10(self.lim[3])-np.log10(self.lim
                        [2]))*percent
46                  self.Ls[0,i] = self.grid(np.power(10,np.log10(self.
                        Ls[0,i]) + (np.matlib.rand(1) < prob)*(np.random
                        .uniform(-1,1,1)*range_Ls*2**(-np.random.uniform
                        (0,1,1)*k_precision))))
47                  self.Ls[0,i] = self.clamp(self.Ls[0,i],self.lmin,
                        self.lmax)
48              for i in range(self.n_mos):
```

```
49                  range_Ws = (np.log10(self.lim[1])-np.log10(self.lim
                        [0]))*percent
50                  self.Ws[0,i] = self.grid(np.power(10,np.log10(self.
                        Ws[0,i]) + (np.matlib.rand(1) < prob)*(np.random
                        .uniform(-1,1,1)*range_Ws*2**(-np.random.uniform
                        (0,1,1)*k_precision))))
51                  self.Ws[0,i] = self.clamp(self.Ws[0,i],self.wmin,
                        self.wmax)
52              for i in range(self.n_res):
53                  self.Rs[0,i] = self.grid(self.Rs[0,i] + (np.matlib.
                        rand(1) < prob)*self.Rs[0,i]*percent*np.matlib.
                        randn(1)/3)
54                  self.Rs[0,i] = self.clamp(self.Rs[0,i],self.rmin,
                        self.rmax)
55          elif (mutation_type == 'lin'):
56          # Circuit mutation with probability and relative percentual
                force
57              for i in range(self.n_mos):
58                  self.Ls[0,i] = self.grid(self.Ls[0,i] + (np.matlib.
                        rand(1) < prob)*self.Ls[0,i]*percent*np.matlib.
                        randn(1)/3)
59                  self.Ls[0,i] = self.clamp(self.Ls[0,i],self.lmin,
                        self.lmax)
60              for i in range(self.n_mos):
61                  self.Ws[0,i] = self.grid(self.Ws[0,i] + (np.matlib.
                        rand(1) < prob)*self.Ws[0,i]*percent*np.matlib.
                        randn(1)/3)
62                  self.Ws[0,i] = self.clamp(self.Ws[0,i],self.wmin,
                        self.wmax)
63              for i in range(self.n_res):
64                  self.Rs[0,i] = self.grid(self.Rs[0,i] + (np.matlib.
                        rand(1) < prob)*self.Rs[0,i]*percent*np.matlib.
                        randn(1)/3)
65                  self.Rs[0,i] = self.clamp(self.Rs[0,i],self.rmin,
                        self.rmax)
66
67  class population:
68      # lim = [wmin,wmax,lmin,lmax]
69      # n_mos = number of mos transistors
70      # size = populations size
71      def __init__(self,size=10,n_mos=4,lim=[0.5e-6,10e-6,0.5e-6,10e
            -6],n_res=0,lim_res=[1e3,10e3]):
72          self.size = size
73          self.n_mos = n_mos
74          self.n_res = n_res
```

```
75          self.ckts = []
76          self.newckts = []
77          for i in range(size):
78              self.ckts.append(circuit(n_mos,lim,n_res,lim_res))
79              self.newckts.append(circuit(n_mos,lim,n_res,lim_res))
80          self.n_winner = int(size/2 - np.mod(size/2,2)) # must be
               even value
81          self.n_clones = int(0) # must be lower than n_winner
82          # Probabilities
83          self.p_mut_pop = 1/(n_mos*2+n_res) #mutation with size "1/N
               ", where N is the number of variables of indiv.
84          self.perc_mut_pop = 0.2
85          self.fitness_prev = 0
86          self.fitness_actual = 0
87          self.roundscale = 1e9
88
89      def tourneament(self):
90          # Search the best fitness circuit
91          best = 0
92          for i in range(self.size):
93              if (self.ckts[i].fom > self.fitness_actual):
94                  self.fitness_actual = self.ckts[i].fom
95                  best = i
96          # Tourneament
97          ind = rd.permutation(self.n_winner*2)
98          win = []
99          los = []
100         for i in range(self.n_winner):
101             if (self.ckts[ind[2*i]].fom >= self.ckts[ind[2*i+1]].
                   fom):
102                 win.append(ind[2*i])
103                 los.append(ind[2*i+1])
104             else:
105                 los.append(ind[2*i])
106                 win.append(ind[2*i+1])
107         return win, los, best
108
109     def crossover(self,win,best):
110         # Save the maximum historical fom individual at first
               position
111         self.newckts[0].Ws = cp.copy(self.ckts[best].Ws)
112         self.newckts[0].Ls = cp.copy(self.ckts[best].Ls)
113         self.newckts[0].Rs = cp.copy(self.ckts[best].Rs)
114         self.newckts[0].fom = cp.copy(self.ckts[best].fom)
115         # Winners on new pop
```

```
116         for i in range(self.n_winner-1):
117             self.newckts[i+1].Ws = cp.copy(self.ckts[win[i]].Ws)
118             self.newckts[i+1].Ls = cp.copy(self.ckts[win[i]].Ls)
119             self.newckts[i+1].Rs = cp.copy(self.ckts[win[i]].Rs)
120             self.newckts[i+1].fom = cp.copy(self.ckts[win[i]].fom)
121         # Crossovered offprings below
122         for i in range(int(self.n_winner/2)):
123             self.newckts[self.n_winner + i].Ws = self.ckts[win[2*i
                ]].grid((self.ckts[win[2*i]].Ws + self.ckts[win[2*i
                +1]].Ws)/2)
124             self.newckts[self.n_winner + i].Ls = self.ckts[win[2*i
                ]].grid((self.ckts[win[2*i]].Ls + self.ckts[win[2*i
                +1]].Ls)/2)
125             self.newckts[self.n_winner + i].Rs = self.ckts[win[2*i
                ]].grid((self.ckts[win[2*i]].Rs + self.ckts[win[2*i
                +1]].Rs)/2)
126
127     def selection(self,win,los):
128         n_offprings = self.n_winner/2
129         start_survivors =  int(self.n_winner + n_offprings + self.
               n_clones)
130         n_survivors =  int(self.size - self.n_winner - n_offprings
               - self.n_clones)
131         fit_surv = []
132         for i in range(n_survivors):
133             fit_surv.append(self.ckts[los[i]].fom)
134         ind = sorted(range(len(fit_surv)), key=lambda i: fit_surv[i
               ], reverse=True)
135         # Survive the best fitness losers
136         for i in range(n_survivors):
137             self.newckts[start_survivors+i].Ws = cp.copy(self.ckts[
                    los[ind[i]]].Ws)
138             self.newckts[start_survivors+i].Ls = cp.copy(self.ckts[
                    los[ind[i]]].Ls)
139             self.newckts[start_survivors+i].Rs = cp.copy(self.ckts[
                    los[ind[i]]].Rs)
140             self.newckts[start_survivors+i].fom = cp.copy(self.ckts
                    [los[ind[i]]].fom)
141
142     def mutate_pop(self):
143         # Mutate all with exception of the best
144         for i in range(self.size-1):
145             self.newckts[i+1].mutation(self.p_mut_pop,self.
                    perc_mut_pop,config.mutation_type)
```