OTÁVIO HENRIQUE BARBOSA TORRES

OTIMIZAÇÃO DE AMPLIFICADORES OPERACIONAIS CMOS POR METAHEURÍSTICAS

Trabalho de Conclusão de Curso
apresentado à Escola de Engenharia de São
Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase
em Eletrônica

ORIENTADOR: Prof.João Navarro Soares Jr.

São Carlos

2014

Torres, Otávio Henrique Barbosa
T693o Otimização de Amplificadores Operacionais CMOS por Metaheurísticas / Otávio Henrique Barbosa Torres; orientador João Navarro Soares Jr.. São Carlos, 2014.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Amplificadores operacionais. 2. circuitos integrados MOS. 3. metaheurísticas. 4. otimização. I. Título.

FOLHA DE APROVAÇÃO

Nome: Otavio Henrique Barbosa Torres

Título: "Otimização de amplificadores operacionais CMOS por metaheurísticas"

Trabalho de Conclusão de Curso defendido e aprovado em 27/11/2014,

com NOTA 9,9 (nove, nove), **pela Comissão Julgadora:**

Prof. Dr. João Navarro Soares Júnior - (Orientador - SEL/EESC/USP)

Prof. Adjunto Mario Alexandre Gazziro - (Universidade Federal do ABC)

Prof. Assistente Paulo Roberto Veronese - (SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

"Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar.
Mas o mar seria menor se lhe faltasse uma gota."
Madre Teresa de Calcutá, 1910 - 1997

Agradecimentos

Agradeço a meus familiares pelo apoio e carinho que me deram por todos estes anos, em especial a minha mãe e meu pai, os quais, sem eles, nada disso poderia ter sido alcançado.
Agradeço também aos meus verdadeiros amigos que estiveram comigo nessa jornada, que sempre me apoiaram, não importassem minhas decisões, e sempre me auxiliaram nos momentos mais difíceis.
Por fim, faço um agradecimento especial ao professor doutor João Navarro Soares Jr., que o considero mais que um professor, um amigo, por tudo aquilo que me ensinou, não apenas sobre eletrônica, mas sobre a vida também e pela paciência que teve comigo. Levarei esses ensinamentos por toda a minha vida.

Autor: Otávio Henrique Barbosa Torres

Título: Otimização de Amplificadores Operacionais CMOS por Metaheurísticas

Data: 10/11/2014

Orientador: Prof. Dr. João Navarro Soares Jr

Área de Concentração: Microeletrônica

RESUMO

O projeto de circuitos analógicos é uma tarefa considerada complicada, visto que no processo estão envolvidas diversas variáveis, como dimensões de transistores e de elementos passivos, como resistores e capacitores, e especificações. O projeto de um circuito visa atingir valores para parâmetros de desempenho como, por exemplo, ganho, velocidade, e potência. As técnicas utilizadas para a realização dessa tarefa envolvem a obtenção de modelos matemáticos, que descrevem o comportamento físico aproximado do circuito, e a simulação, a fim de verificar e melhorar os resultados. Todo o procedimento, contudo, é custoso, lento e muitas vezes não leva a resultados satisfatórios. Com a finalidade de contornar esse cenário negativo, surge a ideia de desenvolver um programa que auxilie no projeto e otimização de amplificadores operacionais. Neste trabalho, foi desenvolvida uma plataforma de otimização de amplificadores operacionais que aplica algoritmos metaheurísticos, Algoritmo Genético, Enxame de Partículas e Recozimento Simulado, e simuladores elétricos, HSpice e Eldo. Para o projeto são analisados o ganho diferencial, ganho de modo comum, frequência e fase de ganho unitário, *slew rate*, potência DC consumida, *offset* de entrada, ruído de saída, estado de operação de transistores e o efeito das variações nas fontes de alimentação. Testes de projeto foram realizados com quatro topologias de amplificadores, utilizando a tecnologia CMOS 0,35μm AMS (AustriaMicroSystem). Um dos amplificadores projetados atingiu ganho diferencial de 81 dB, CMRR de 66 dB, frequência de ganho unitário de 10 MHz, margem de fase de 60º, *slew rate* de 7,0 V/μs, potência DC de 50 μW, *offset* sistemático de entrada de 64 μV, ruído na entrada de 3,3 μVrms e PSRR de 84 dB para carga capacitiva de 1,0 pF e tensão de alimentação de 3,0 V. O desempenho dos três algoritmos é também comparado, revelando a superioridade do Enxame de Partículas em relação aos demais. Os resultados atingidos durante o trabalho mostram a eficácia de metaheurísticas na otimização de amplificadores operacionais CMOS.

Palavras Chaves: Amplificadores operacionais, circuitos integrados MOS, metaheurísticas, otimização.

Author: Otávio Henrique Barbosa Torres

Title: Optimization of Operational Amplifier by Metaheuristics

Date: 11/10/2014

Advisor: Prof. Dr. João Navarro Soares Jr.

Concentration Area: Microelectronics

ABSTRACT

The design of analog circuits is considered a complicated task, since several variables are involved in the process, such as dimensions of transistors and passive elements, resistors and capacitors, and specifications. The circuit design aims at values of performance parameters such as gain, speed and power. The techniques used to perform this task involve obtaining mathematical models that describes the approximate physical behavior of the circuit, and the simulation in order to verify and improve the results. The whole procedure, however, is costly, slow and often does not lead to satisfactory results. Aiming at circumvent this negative scenario, an idea arises: the development of a program that assists in the design and optimization of operational amplifiers. In this paper, a platform for optimizing operational amplifiers applying metaheuristic algorithms, Genetic Algorithm, Particle Swarm, and Simulated Annealing, and electrical simulators, HSPICE and Eldo, was developed. For the design it was analyzed the differential gain, common mode gain, unity gain phase and frequency, slew rate, DC power, input offset, output noise, the operation state of transistors and the effect of variations in the power supplies. Design tests were performed with four topologies of amplifiers, using AMS CMOS 0,35μm (AustriaMicroSystem) technology. One of the designed amplifiers reached differential gain of 81 dB, CMRR of 66 dB, unity gain frequency of 10 MHz, 60° phase margin, slew rate of 7.0 V/ms, DC power of 50 μW, systematic input offset of 64 μV, input noise of 3.3 μVrms, and PSRR of 84 dB to 1.0 pF capacitive load and supply voltage of 3.0 V. The performance of all three algorithms is also compared, revealing the superiority of Particle Swarm particles in relation to others. The results achieved in the dissertation show the effectiveness of metaheuristics in the optimization of CMOS operational amplifiers.

Key Words: operational amplifier, MOS integrated circuits, metaheuristics, optimization

LISTA DE FIGURAS

LISTA DE TABELAS

Sumário

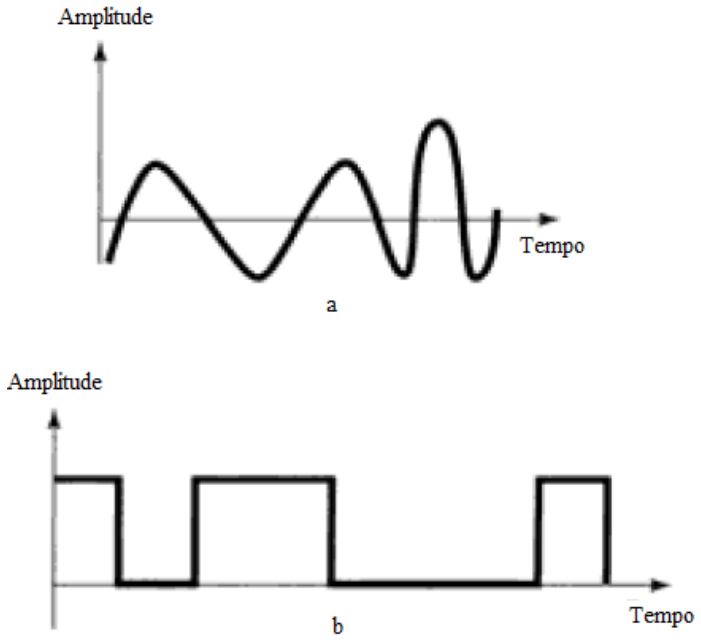
1 INTRODUÇÃO

A tecnologia mais difundida na fabricação de circuitos integrados hoje é a CMOS (*Complementary metal-oxide Semiconductor*) (Allen & Hoberg, 2002), apresentando vantagens consideráveis tanto no consumo de potência como na facilidade para implementação de projetos digitais, onde o CMOS é praticamente a única tecnologia empregada (ITRS, 2005).

Circuitos eletrônicos podem ser divididos em duas grandes categorias: os digitais e os analógicos (Allen & Hoberg, 2002). As diferenças entre as duas categorias estão diretamente relacionadas com o tipo de sinais que os circuitos manipulam nas entradas e saídas, digitais ou analógicos, de onde vêm seus nomes.

Um sinal aplicado em um sistema eletrônico pode ser caracterizado pelo valor de tensão, de corrente, etc. Um sinal analógico é aquele contínuo no tempo, ou seja, ele pode assumir qualquer valor dentro de um espaço contínuo de valores, para um instante qualquer no tempo (Roberts, 2010). Já o digital é aquele que assume valores discretos. Usualmente, o conjunto de valores do sinal digital é binário. Exemplos de sinais contínuos e digitais binários estão apresentados na Figura 1:

Figura 1 – Sinais (a) analógico e (b) digital



Devido ao acentuado grau de regularidade dos circuitos digitais, ferramentas computacionais para auxiliar projetistas no desenvolvimento de projetos digitais têm obtido grande sucesso na automação destes. Todavia, o mesmo não ocorre com os circuitos analógicos devido ao seu maior grau de complexidade. Para o projeto de circuitos analógicos é necessário grande experiência do projetista, para que este consiga obter os modelos matemáticos que descrevem o comportamento físico

aproximado do circuito. A partir destes modelos, que funcionam para um ponto de partida, as dimensões dos dispositivos dos circuitos são determinadas e, em seguida, simulações são realizadas, a fim de verificar e melhorar os resultados. Contudo, todo esse procedimento é custoso, lento e muitas vezes não leva a resultados satisfatórios.

Partindo desse fato, surge o desejo da implementação de uma plataforma computacional que consiga contornar as adversidades oferecidas pelo projeto de circuitos analógicos, reduzindo custos e tempo de projeto.

Na construção de tal plataforma, em que este trabalho contribui, foram utilizadas ferramentas básicas, já bem conhecidas por engenheiros eletricistas: duas delas são os *softwares* de simulação elétrica, HSpice-SYNOPSYS (HSPICE, 2014) e Eldo-MENTOR GRAPHICS (ELDO, 2014), e a outra é o *software* MatLab MATHWORKS, com algoritmos de otimização, versão 2010 (MATLAB, 2014).

Os circuitos estudados nesse texto são amplificadores operacionais, que estão presentes em diversos sistemas eletrônicos na área de controle, instrumentação industrial, instrumentação médica (bioeletrônica e eletromedicina), telecomunicações, entre outras (Pertence Júnior, 2012). A função de um amplificador operacional, grosso modo, é amplificar a diferença de tensão aplicada entre seus terminais de entradas.

Para projetar e otimizar os amplificadores operacionais, três metaheurísticas diferentes serão aplicadas: o Algoritmo Genético (do inglês, *Genetic Algorithm*), o Algoritmo Enxame de Partículas (*Particle Swarm*) e o Algoritmo Recozimento Simulado (S*imulated* A*nnealing*).

Metaheurísticas são famílias de técnicas aproximadas de otimização utilizadas na resolução de problemas cuja solução considerada boa ou ótima não se conhece previamente, não se sabe como alcançar e não se pode determinar através de métodos de força bruta. Adicionalmente, dado um candidato à solução do problema, é possível determinar o grau de qualidade deste (Luke, 2013).

Os problemas de projeto e otimização de circuitos analógicos apresentam as características típicas dos problemas onde se aplicam metaherísticas. Na utilização das metaheurísticas serão aplicadas simulações elétricas para avaliar o grau de qualidade dos circuitos, verificando se eles conseguem garantir certas especificações do projetista.

A tecnologia utilizada neste trabalho é a tecnologia CMOS 0,35μm da AMS (*Austria Micro-Systems*), (AUSTRIAMICROSYSTEMS, 2003) e (AUSTRIAMICROSYSTEMS, 2014).

Esse trabalho é baseado nos estudos iniciais de (Filgueiras, 2010) e de (Torres, 2012).

1.1 Objetivos

Conforme apresentado na seção anterior, o objetivo desse trabalho é desenvolver uma plataforma que auxilie projetistas, mesmo com pouca experiência, no projeto e otimização de

amplificadores operacionais. Também se fará uma avaliação do desempenho dos algoritmos metaheurísticos utilizados nesse trabalho, comparando seus resultados.

1.2 Estrutura da Monografia

O restante da monografia é estruturado conforme descrito a seguir:

- O segundo capítulo contém alguns conceitos básicos para o entendimento deste texto;
- O terceiro capítulo contém os métodos utilizados para a implementação da ferramenta de projeto;
- O quarto capítulo contém os resultados e discussões sobre os mesmos;
- O quinto capítulo contém as conclusões sobre este trabalho;

2 CONCEITOS GERAIS

2.1 Transistores MOS

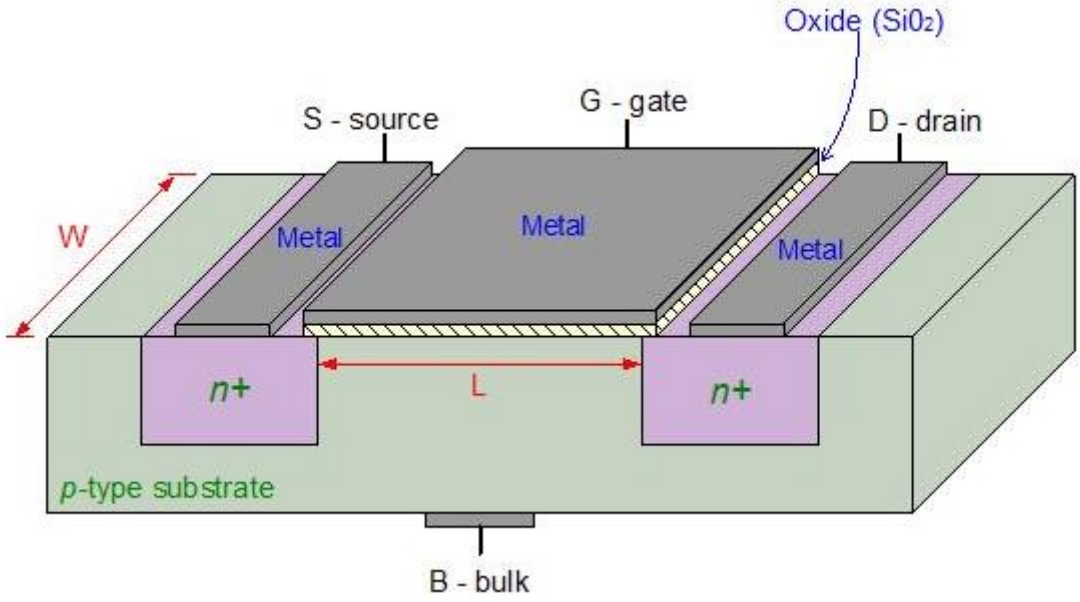
As duas tecnologias mais aplicadas no projeto de circuitos integrados são as dos transistores bipolares e MOS. Por muitos anos, houve a dominância por parte da primeira, já que a segunda possuía um pior desempenho e uma construção mais trabalhosa. No início da tecnologia MOS, o transistor dominante foi o de canal P, também conhecido como PMOS e onde a condução no canal é feita por lacunas, devido às menores dificuldades na sua fabricação.

No inicio da década de 70, com as maiores dificuldades com a fabricação de transistores MOS superadas, viabilizou-se o uso destes para o projeto de microprocessadores e DRAM´s (do inglês, *Dynamic Random-Acess Memory*) e surgiram as famílias lógicas da série 4000. Já no final da década de 70, era claro que a tecnologia MOS seria o "carro chefe" da área de VLSI (*Very-Large-Scale Integration*). A tecnologia com transistores de canal N, também conhecida como NMOS onde a condução no canal é feita por elétrons, tornou-se a mais difundida tanto para projetos de circuitos analógicos quanto digitais (Allen & Hoberg, 2002).

Nos anos 80, devido ao aumento na densidade e velocidade dos circuitos, problemas de potência começam a surgir, abrindo espaço para que a tecnologia mais sofisticada CMOS, que possui transistores com canal tanto P quanto N, ganhasse mais espaço como a tecnologia utilizada para projetos de VLSI. CMOS é, até os dias de hoje, a tecnologia mais empregada (ITRS, 2005).

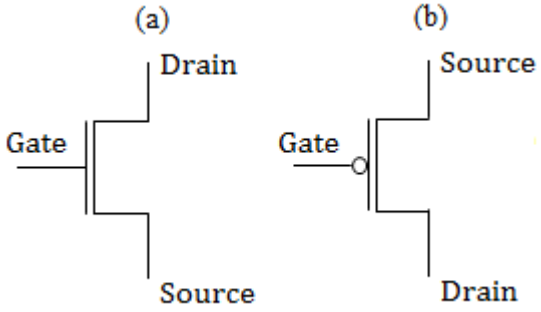
Apesar de a tecnologia CMOS possuir mais etapas para sua fabricação e, por consequência, um maior custo, ela tem características vantajosas, como apresentar menor consumo de potência, permitir maior grau de integração entre vários blocos de um do projeto e ter tamanho reduzido. Na Figura 2 é possível observar a vista lateral de um transistor NMOS. Nela se observa a largura W (do inglês, *width*) e o comprimento L (do inglês, *length*) do canal, os quais são os alvos do projeto de circuitos CMOS. Na Figura 3 encontram-se os símbolos dos transistores NMOS e PMOS.

Figura 2 – Vista lateral de um transistor NMOS



(Pimenta, Moreno, & Zoccal, 2011)

Figura 3 – Símbolos dos transistores (a) NMOS e (b) PMOS



Os transistores MOS, dependendo da tensão entre a porta (*gate*) e a fonte (*source*), quando estão conduzindo podem operar em três distintas regiões:

- Forte inversão: nessa região, a tensão V_{GS} (entre o *gate* e o *source*) é maior que a tensão de limiar (V_{th}). Essa região é utilizada em projetos onde os transistores funcionarão como chaves ou para a amplificação de sinais rápidos;
- Fraca inversão: nessa região, a tensão V_{GS} é muito próxima da tensão V_{th}. Essa região é utilizada quando se deseja que o transistor trabalhe com baixas potências e frequências;
- Inversão moderada: região entre a fraca e forte inversão. A operação de transistores nela não é modelada de maneira exata.

2.2 Amplificadores Operacionais

Os amplificadores operacionais (AmpOp) são um dos mais importantes blocos no projeto de circuitos eletrônicos analógicos. Suas utilizações datam do inicio dos anos 50, quando eram

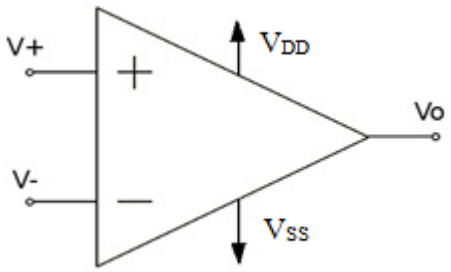
implementados com componentes discretos, como válvulas, diferente das construções atuais que utilizam transistores, capacitores e resistores integrados. Nessa época, o custo de produção de um AmpOp era alto e seu desempenho, comparado com os atuais, baixo.

Amplificadores operacionais são circuitos amplificadores multiestágio, com uma entrada diferencial e cujas características principais são:

- Resistência de entrada grande (idealmente infinita);
- Resistência de saída baixa (idealmente zero);
- Ganho diferencial de tensão grande (idealmente infinito);
- Ganho de modo comum baixo (idealmente zero);
- Resposta em frequência constante (banda idealmente infinita);
- Insensibilidade à temperatura e variações da tensão de alimentação (Pertence Júnior, 2012).

Os AmpOp são dispositivos multiterminais, mas, por simplicidade, comumente são apenas apresentados cinco deles, conforme é exibido na Figura 4:

Figura 4 – Símbolo de um amplificador operacional



O terminal V_+ é chamado de terminal não inversor de entrada e V_-, de terminal inversor de entrada. O terminal de saída do circuito é V_O.

Os AmpOp respondem, idealmente, apenas à diferença de sinais das entradas, (V_+ - V_-), e dessa forma, rejeitam qualquer sinal comum que haja nas entradas. Assim, se houver valores iguais de tensão em V_+ e V_-, a saída será, teoricamente, nula (Sedra & Smith, 2004).

A tensão de saída do AmpOp é, portanto, diretamente proporcional à entrada diferencial e essa relação de proporcionalidade é chamada de ganho diferencial, o qual é idealmente infinito.

Os outros terminais são os de alimentação V_{DD}, tensão de alimentação positiva, e V_{SS}, tensão de alimentação negativa. Nesse texto, V_{SS} será sempre a tensão de referência de todo o circuito.

Os amplificadores operacionais possuem características internas que são de extrema importância, quando se deseja projetá-los. Essas características são introduzidas nas próximas seções.

2.2.1 Ganho diferencial

Conforme já apresentado, o ganho diferencial, ou ganho de malha aberta, é a relação existente entre a tensão de saída do bloco e as tensões de entrada. Essa relação pode ser expressa como:

$$G = \frac{V_O}{V_+ - V_-} \quad \textbf{(1)}$$

onde *G* é esse ganho diferencial.

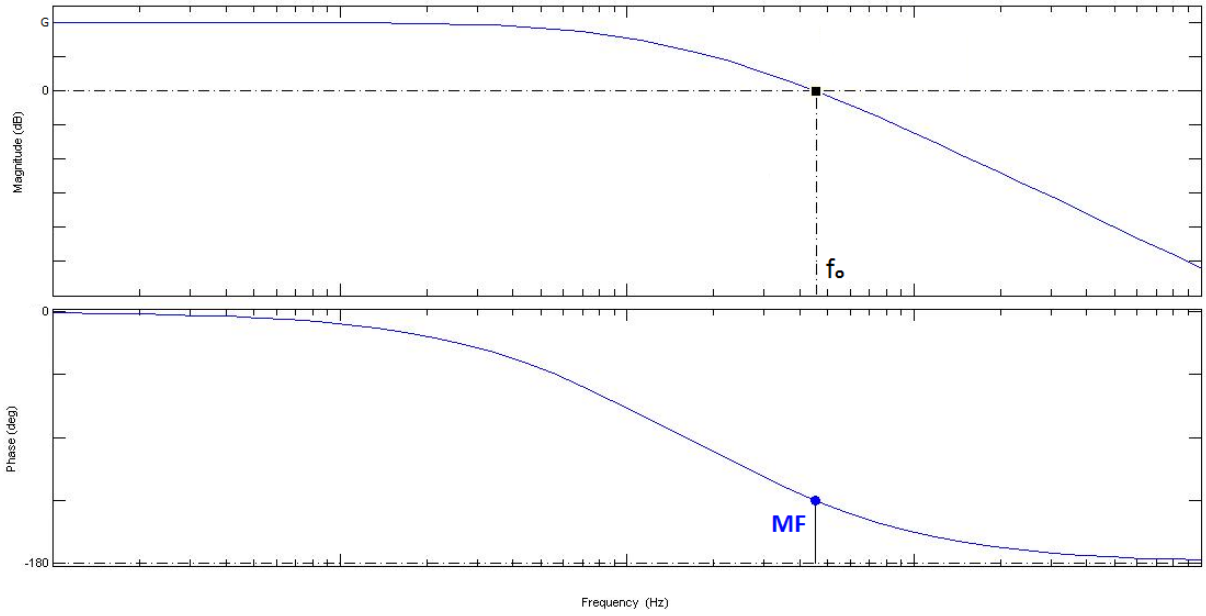
Idealmente, o valor de *G* é infinito para qualquer frequência em que o circuito opere, contudo, na prática, os valores estão na ordem de 40 a 100 dB, ou seja, 10^2 a 10^5 V/V, para faixas limitadas de frequência.

O termo malha aberta, utilizado anteriormente, refere-se ao fato do sistema estar ou não sendo realimentado. Em um sistema em malha aberta, a realimentação não é feita, diferente de um sistema em malha fechada.

2.2.2 Frequência de ganho unitário

Como exposto na seção 2.2.1, o AmpOp apresenta um ganho de malha aberta considerável para faixas restritas de frequência. A partir do momento que a frequência de operação do circuito aumenta, menor será o ganho diferencial que este dispositivo irá proporcionar. Isso ocorre devido a capacitâncias, parasitas ou não, existentes nos componentes internos do circuito. Esse fato pode ser mais bem observado na Figura 5, a qual apresenta o diagrama de Bode (Simon & Barry, 2001) de um amplificador operacional típico em uma configuração de malha aberta:

Figura 5 – Diagrama de Bode, modulo e fase, para um amplificador operacional típico em malha aberta



Nota-se, a partir de Figura 5, que o ganho *G* é praticamente constante até uma determinada frequência. A partir dela, *G*, que era constante, passa a cair gradativamente. Um ponto muito importante desse gráfico localiza-se na frequência em que a curva de ganho atinge 0,0 dB, ou seja, quando se tem uma amplificação de tensão de 1,0 V/V. Nesse ponto encontra-se a frequência de ganho unitário, representada por f_o.

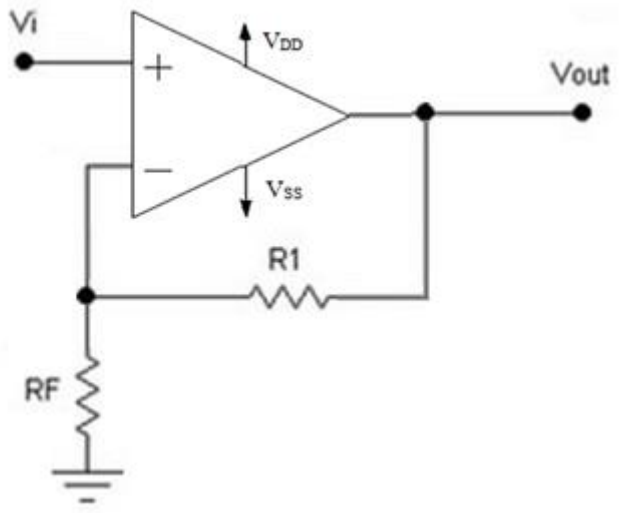
A frequência de ganho unitário é de extrema importância, pois indica para o projetista os limites de frequência em que o AmpOp pode ser empregado. Para amplificadores MOS típicos, f_o se encontra entre 1,0 MHz e 15,0 MHz.

2.2.3 Margem de fase e de ganho

AmpOp são normalmente aplicados em configurações de malha fechada de forma que o ganho, relação entre os sinais de saída e entrada, seja determinado apenas pelos componentes externos que compõe a realimentação da malha fechada.

A realimentação pode ser positiva ou negativa. A diferença entre ambas está apenas em qual terminal (positivo ou negativo) será aplicada a realimentação. Na Figura 6 é observado um exemplo de realimentação negativa.

Figura 6 – Exemplo de realimentação negativa em um amplificador operacional



Circuitos com realimentações positivas são instáveis e podem ser aplicados como osciladores. Para o caso de realimentação negativa, o circuito normalmente deve ser estável, mas nem sempre isso acontece.

A defasagem de fase entre os sinais de entrada e saída depende da frequência (ver Figura 5) e pode atingir -180º em alguma f_{inv}. Nesse caso o sinal de saída encontra-se invertido em comparação com o de entrada. Caso esse sinal seja realimentado negativamente, ele passa a ficar em fase com o sinal de entrada. Se o ganho do sistema, em malha aberta, for maior ou igual a 1,0 V/V na frequência

f_{inv}, o sinal resultante da soma da entrada e da realimentada começa a aumentar indefinidamente (na prática isso não acontece, pois o sistema satura em algum momento).

Define-se margem de fase conforme a equação a seguir:

$$MF = 180° - |\theta_{0dB}| \quad \textbf{(2)}$$

onde θ_{0dB} é o ângulo de fase correspondente à frequência de ganho unitário, considerando apenas valores no intervalo [-180^0, 180^0[.

Já margem de ganho é definida conforme a equação a seguir:

$$MG = -G_{-180°} \quad \textbf{(3)}$$

onde $G_{-180°}$ é o ganho do sistema em malha aberta, em decibéis, na frequência onde o ângulo de fase é -180º.

Uma condição suficiente e necessária para que um sistema com realimentação negativa seja estável é que as margens de fase e de ganho do sistema de malha aberta sejam positivas. Na realidade, se ao menos uma dessas grandezas for positiva, isso implicará que a outra também é, pois elas são dependentes. Para este trabalho, os amplificadores operacionais serão projetados para possuírem margem de fase em torno de 45º a 60º, pois assim se tem uma boa distância do ponto crítico de operação (*MF* = 0º). Adicionalmente, a resposta transitória do amplificador, aplicado em uma configuração com realimentação negativa, para uma entrada em degrau, não apresenta grandes oscilações.

2.2.4 *Slew Rate*

O *slew rate* é taxa máxima de variação da tensão de saída no tempo e pode ser expressa por:

$$SlewRate = \left.\frac{dV_o}{dt}\right|_{máx} \quad \textbf{(4)}$$

onde V_O é a tensão de saída do circuito e t o tempo.

A limitação nessa taxa é um fenômeno que pode causar distorções quando sinais de grandes amplitudes estão presentes na entrada. Para exemplificar isso, considere um circuito configurado como seguidor de tensão, onde a tensão de saída deve acompanhar a entrada, e o sinal de entrada seja do tipo degrau, conforme é observado na Figura 7. Idealmente, para um AmpOp com *slew rate* infinito, a saída possuirá um comportamento igual ao da entrada. Todavia, devido ao s*lew rate* finito, a saída de um circuito real não conseguirá subir instantaneamente para o valor *A*. Ao invés disso, a saída terá a forma de uma rampa linear, como mostra a Figura 8, com inclinação igual ao do *slew rate* (Sedra & Smith, 2004).

Para amplificadores MOS típicos, os valores de *slew rate* encontram-se na faixa de 1,0 V/μs a 20,0 V/μs.

Figura 7 – a) Configuração de um circuito seguidor de tensão e b) uma entrada degrau

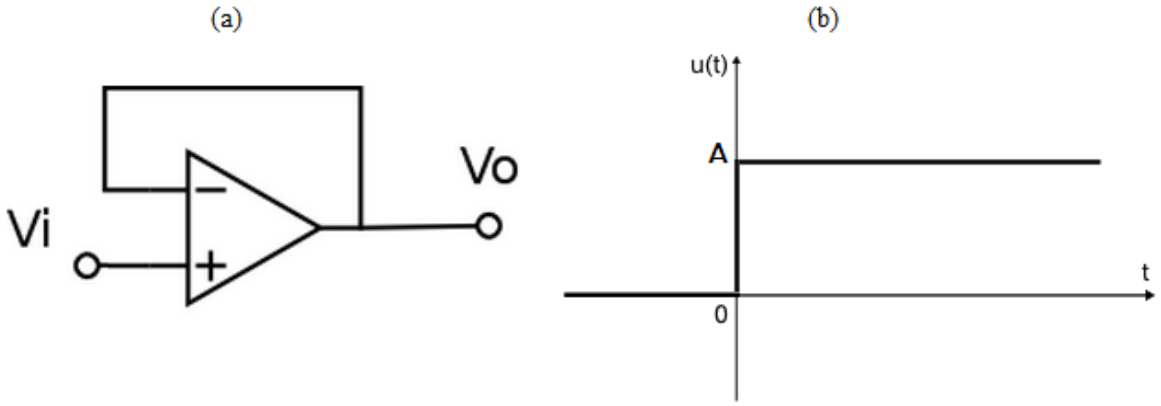
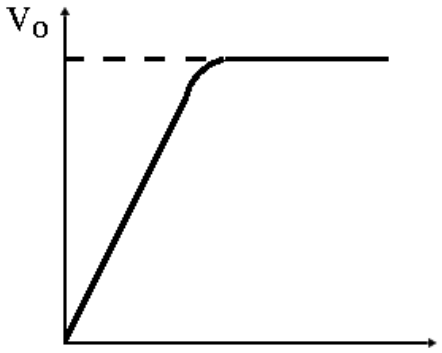


Figura 8 – Resposta ao degrau, devido à taxa de variação de tensão



2.2.5 Tensão de *offset*

Para amplificadores ideais, se tensões iguais fossem aplicadas em seus terminais de entrada, V_O seria nulo de acordo com a equação (1). Porém, isso não acontece na realidade devido ao descasamento de transistores, principalmente no par diferencial da entrada do amplificador, e a não simetria perfeita das topologias. À tensão não nula que aparece na saída quando as entradas forem iguais, dá-se o nome de tensão de *offset* de saída.

O *offset* pode ainda ser descrito com relação à entrada do circuito, definindo a tensão de *offset* de entrada como a tensão aplicada entre os terminais V_+ e V_- do AmpOp tal que a tensão de saída do circuito seja nula.

Para amplificadores MOS típicos, os valores de tensão de *offset* de entrada estão abaixo de 3,0 mV.

2.2.6 Taxa de rejeição de modo comum (CMRR)

Quando um amplificador possui, em suas entradas, as mesmas tensões, diz-se que o AmpOp trabalha em modo comum. A tensão de entrada de modo comum é expressa pela seguinte equação:

$$V_{\text{mod.}comum} = \frac{V_+ + V_-}{2} \tag{5}$$

Conforme dito na seção 2.2.5, idealmente o sinal de saída para entradas iguais será zero. Na prática, haverá um sinal na saída com valor pequeno e chama-se de ganho de modo comum a seguinte relação:

$$A_C = \frac{\Delta V_o}{\Delta V_{\text{mod}.comum}} \tag{6}$$

onde ΔV_O é a variação da tensão na saída do amplificador operacional e $\Delta V_{mod.comum}$ é a variação na tensão de modo comum.

Amplificadores operacionais ideais têm como característica rejeitar a amplificação de sinais de modo comum, amplificando apenas os sinais diferenciais (equação (1)). Todavia, AmpOps reais não conseguem eliminar, totalmente, o ganho de modo comum.

AmpOps devem ser projetados de tal maneira que consigam reduzir ao máximo o ganho de modo comum. A taxa de rejeição de modo comum, ou simplesmente CMRR (do inglês, *Common Mode Rejection Ratio*) expressa o quanto o comportamento do amplificador se aproxima do ideal. Essa taxa pode ser expressa como:

$$CMRR = 20\log_{10}\left(\frac{G}{A_C}\right) \tag{7}$$

Para amplificadores típicos, o valor de CMRR varia entre 60 dB a 100 dB.

2.2.7 Taxa de rejeição de alimentação (PSRR)

Amplificadores operacionais devem ser insensíveis a variações nas fontes de alimentação. Essas variações podem ocorrer na forma de ruído e é indesejável sua amplificação. A medida da capacidade de suprimir os ruídos vindos pelas fontes de alimentação é feita pela taxa de rejeição de alimentação, ou simplesmente, PSRR (do inglês, *Power Supply Rejection Ratio*) que é expressa como:

$$PSRR = 20\log_{10}\left(\frac{G}{A_P}\right) \tag{8}$$

onde A_P é o ganho de amplificação dos sinais que vem pelas fontes de alimentação.

Para mensurar a taxa de rejeição de alimentação, é necessário avaliar o comportamento do sinal de saída para ruídos tanto na fonte de alimentação positiva, resultando em um $PSRR^+$, quanto na fonte negativa, $PSRR^-$.

Idealmente, o valor de PSRR seria infinito, não havendo, portanto, ganho de tensão sobre as fontes de alimentação. Todavia, para amplificadores típicos, o valor de PSRR varia entre 60 dB a 100 dB.

2.2.8 Ruído

Ruídos são sinais de natureza aleatória que interferem na operação dos circuitos. Sua quantização é feita através da densidade espectral de potências, a qual indica quanta potência um sinal possui em uma determinada frequência, e, a partir desta, é possível encontrar a tensão ou corrente média de ruído.

Para transistores MOS, a magnitude dos ruídos é consideravelmente alta, comparada aos bipolares, principalmente em baixas frequências, onde o ruído *1/f*, ou ruído *flicker*, é importante.

Para amplificadores CMOS típicos, os valores de ruídos de entrada encontram-se na faixa de 10 μVrms a 50 μVrms (Tranquilini, 2008).

2.2.9 Potência

Amplificadores ideais são aqueles que não dissipam potência, mas os reais devem ser projetados para dissipar a menor quantia possível.

2.3 Metaheurísticas

Metaheurísticas são métodos computacionais aplicáveis na resolução de problemas de otimização, geralmente usados quando: não se conhece um algoritmo exato para resolver o problema; não se conhece, de antemão, como uma solução ótima se parece; não se conhece uma maneira de se chegar a esta solução; procedimentos utilizando força bruta, tentativa e erro, não são viáveis; mas é possível avaliar a qualidade de uma eventual resposta ao problema.

Várias são as metaheurísticas já propostas, (Luke, 2013), contudo, nesse texto, serão utilizadas apenas três que tem sido muito aplicada em projetos. São elas:

- *Genetic algorithm* (algoritmo genético);
- *Particle swarm* (enxame de partículas);
- *Simulated annealing* (recozimento simulado).

2.3.1 *Genetic algorithm*

Os algoritmos genéticos (do inglês, *Genetic Algorithms*) (Holland, 1992) fazem parte da área de computação denominada Bioinspirada. Na computação Bioinspirada são usados conceitos da biologia para a confecção de novos algoritmos. Os principais ramos dessa área são: Redes Neurais, Sistemas Imunológicos Artificiais e Computação Evolutiva (Torres, 2012).

Os algoritmos genéticos (GA) encontram-se nesse último ramo e são técnicas baseadas no processo evolucionário dos seres vivos. Estes algoritmos dispõem de mecanismos de "seleção natural" de indivíduos e de transmissão "genética" de características entre gerações.

Antes de explicar o funcionamento do GA, devemos esclarecer o que são indivíduos, população e geração. Individuo é uma possível solução do problema. A um conjunto de indivíduos dá-se o nome de população. Uma população também pode ser denominada de geração, contudo este termo relaciona certa população num determinado momento de tempo.

Para o inicio do algoritmo, uma quantidade de indivíduos é criada formando a primeira geração. Os indivíduos são avaliados por uma função, função de aptidão, que determina o quão ótimo eles são, atribuindo notas a esses indivíduos.

Em seguida, os indivíduos mais aptos são selecionados, a partir das notas de avaliação que estes eles recebem: quanto melhor a nota do indivíduo, maior é a chance de que ele permaneça e, assim, continue contribuindo com a busca de um candidato ótimo à solução do problema.

Realizada a seleção dos indivíduos, são gerados novos indivíduos através do "acasalamento" das soluções remanescentes do processo de seleção, que por meio da mistura de características de indivíduos "pais" compõem os indivíduos "filhos"; e da "mutação" dos filhos, que altera aleatoriamente as algumas características de alguns indivíduos. Dessa forma é criada uma nova geração de candidatos à solução ótima do problema.

Os processos de avaliação, de seleção de indivíduos e de formação de novas gerações repetem-se, formando as iterações do GA, e estas iterações são realizadas até que algum critério de parada, como número máximo de gerações, número máximo de indivíduos analisados, tempo de processamento, etc., seja atingido. Ao final uma ou mais soluções ótimas para o problema inicial devem ser encontradas.

Nas próximas seções, os conceitos e termos sobre GA serão mais bem explicados.

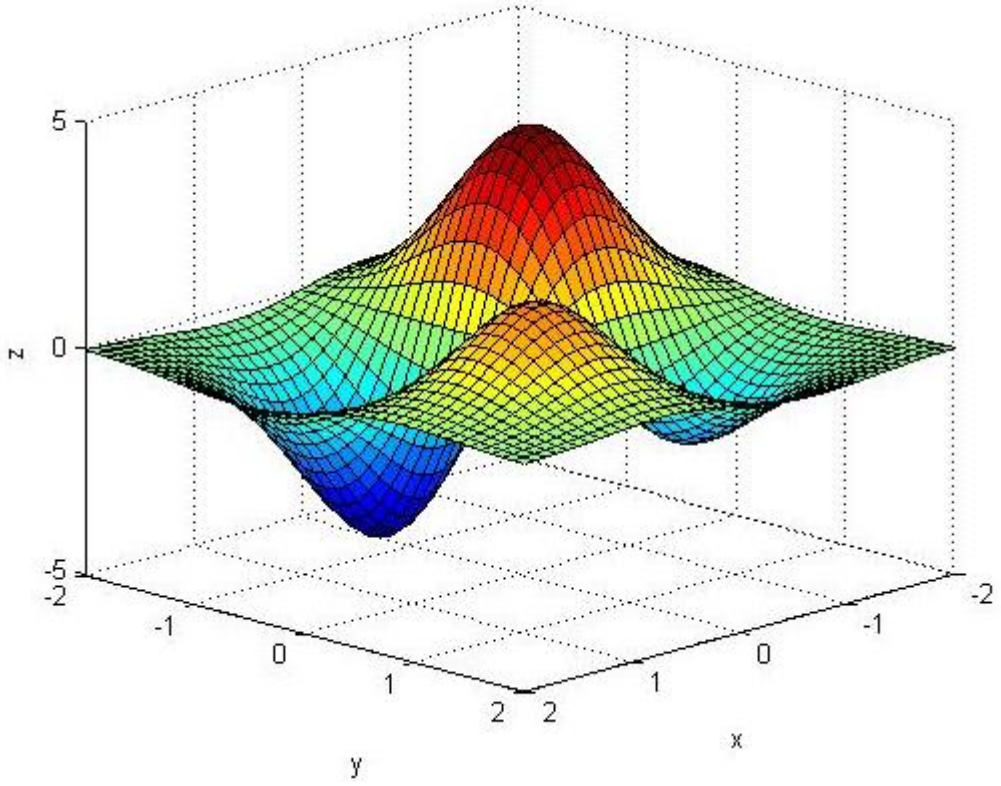
2.3.1.1 Indivíduos e população

No âmbito da biologia, indivíduos podem ser caracterizados por possuírem cromossomos específicos. Esses cromossomos são constituídos de genes, os quais possuem informações referentes às características do indivíduo.

Em linguagem computacional, o cromossomo, que caracterizará um indivíduo, pode ser expresso por um vetor e seus genes pelos elementos contidos nesse vetor.

Para melhor compreender, considere o problema de encontrar o mínimo da função $z = 4x(4y - 1)e^{-x^2-y^2}$, para o intervalo entre -2 ≤ x ≤ 2 e -2 ≤ y ≤ 2, apresentado na Figura 9.

Figura 9 – Função que se deseja encontrar o mínimo valor dentro do intervalo de x e y



Os cromossomos dos indivíduos desse problema seriam representados pelo vetor [x; y] e os genes seriam x e y que podem assumir valores reais entre -2 e 2, inclusive. Em situações mais complexas, o número de elementos do vetor, ou seja, o número de genes de cada cromossomo, bem como o espaço a ser explorado pelo programa, são maiores.

2.3.1.2 Função de aptidão

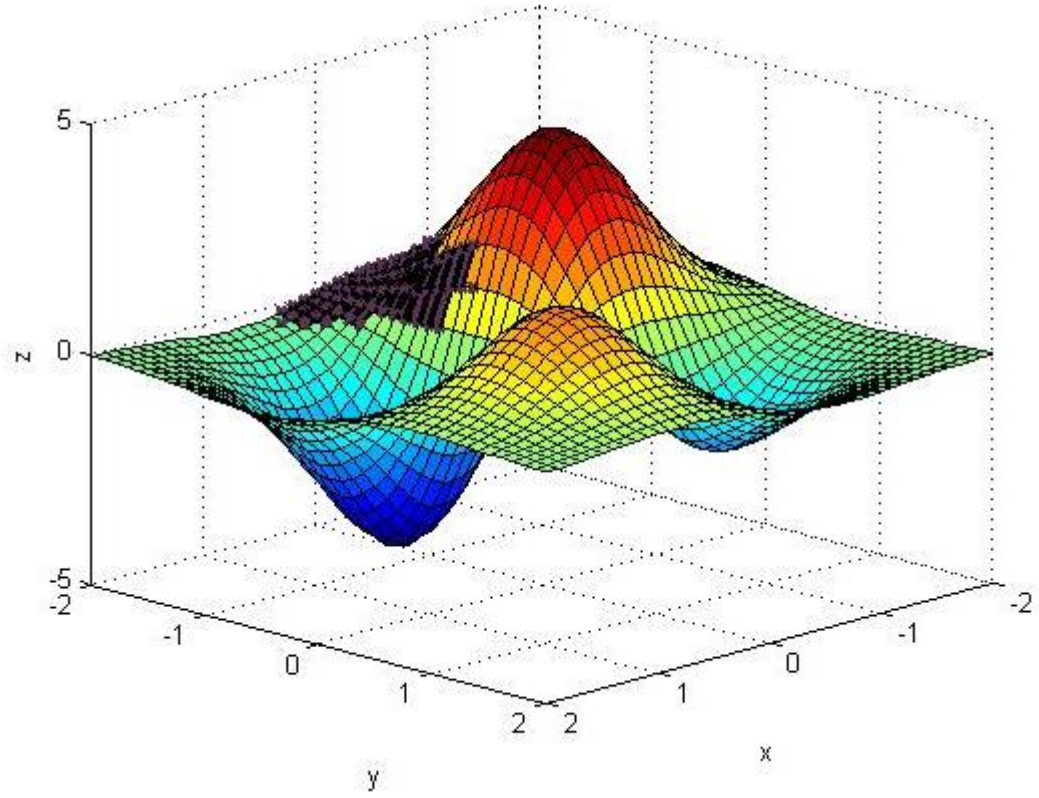
Também chamada de função de *fitness*, a função de aptidão é aquela que avalia o quão ótimo são os indivíduos de uma determinada população. Ela pode ser entendida como geradora da nota que o candidato à solução do problema recebe. Nesse caso, deseja-se a maximização dessa nota. Outra forma de se entender a função de *fitness* é como geradora de uma estimativa da distância que o indivíduo se encontra de uma solução ótima. Agora, deseja-se a minimização dessa distância.

No caso apresentado na seção anterior, a *fitness function* é a própria função $z = 4x(4y - 1)e^{-x^2-y^2}$ e cada indivíduo [x; y] recebe sua respectiva nota, o valor de z.

2.3.1.3 Seleção

Dados os indivíduos de uma determinada geração e estes devidamente avaliados pela *fitness function*, é feita a seleção dos elementos que gerarão os indivíduos da próxima geração, ou pais. Selecionar apenas melhores indivíduos poderia ser um bom método de escolha, visto que estes passariam às populações adiante seus genes bem classificados. Contudo, essa escolha possui a complicação de convergir rapidamente para um conjunto menos diversificado de soluções, restringindo o espaço de busca. Referenciando ao exemplo, seria como se o programa analisasse apenas a região em negrito observado na Figura 10.

Figura 10 – Espaço reduzido de busca por soluções ótimas

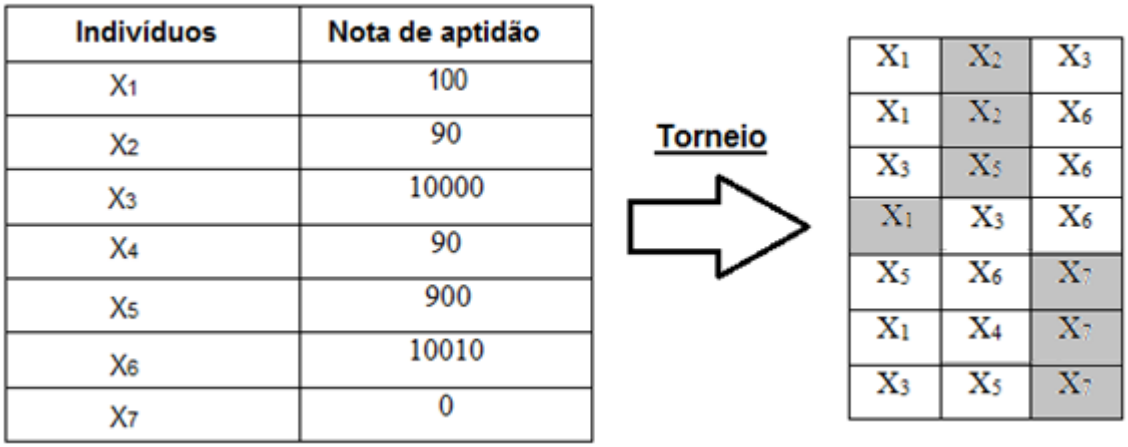


Por outro lado, selecionar indivíduos com notas ruins pode acarretar na permanência de características indesejáveis ao longo de várias gerações. Portanto, deve haver um equilíbrio, entre os melhores e os piores indivíduos no processo de seleção.

Várias são as formas de seleção de indivíduos aplicadas em GA, sendo a mais popular, conforme (Luke, 2013), o método de seleção por torneio, o qual é brevemente explicado a seguir.

Esse método utiliza sucessivas "lutas entre os indivíduos" para selecionar os pais. Essas lutas ocorrem entre *t* indivíduos escolhidos ao acaso. O "vencedor" dessa disputa é sempre aquele que possuir a melhor nota de aptidão. Um exemplo de seleção por torneio é apresentado na Figura 11, onde é feito o torneio entre três indivíduos e a melhor nota é considerada a menor delas.

Figura 11 – Exemplo de método de seleção por torneio



Indivíduos	Nota de aptidão
X_1	100
X_2	90
X_3	10000
X_4	90
X_5	900
X_6	10010
X_7	0

Nesse exemplo, sete indivíduos, tabela a esquerda, constituem uma população e sete disputas são realizadas. As escolhas de quais candidatos formarão uma disputa são aleatórias, permitindo que o mesmo candidato participe de várias delas. Ao final de cada disputa, linhas da tabela a direita, a função de seleção decide quem será o vencedor, elemento hachurado da linha.

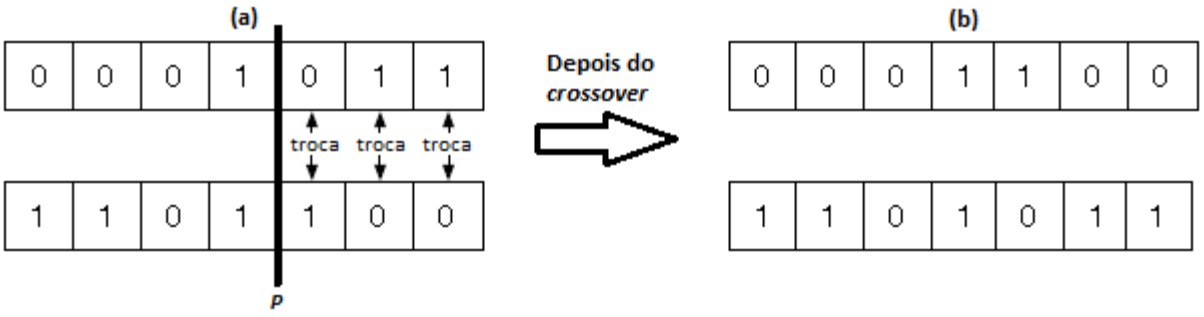
2.3.1.4 Reprodução

A reprodução é o processo no qual a partir de dois ou mais indivíduos remanescentes do processo de seleção, se geram, através da mistura de seus códigos genéticos, um ou mais novos indivíduos, filhos, os quais farão parte de uma nova população. Essa mistura genética ou de características é denominada de *crossover*.

O *crossover* pode ocorrer de várias maneiras, sendo as mais utilizadas (Luke, 2013):

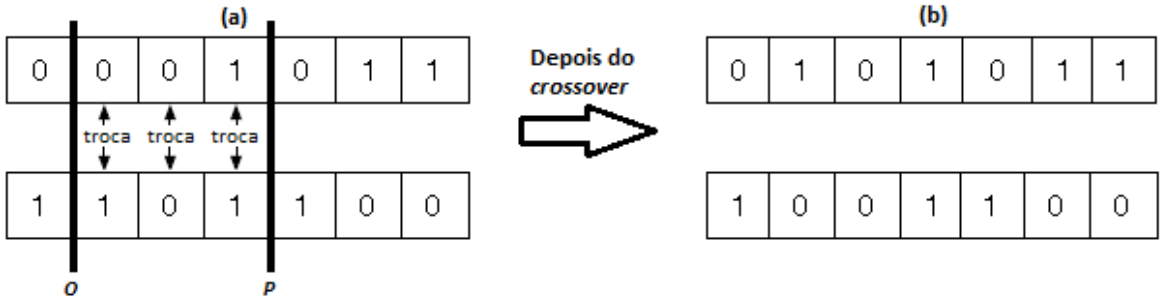
- De um ponto: um ponto de cruzamento no cromossomo é escolhido e a partir dele, os genes dos pais serão permutados. Os genes anteriores a este ponto de um dos pais são ligados aos genes posteriores a este ponto de outro pai, formando o cromossomo do filho. A figura a seguir, Figura 12, mostra como esse processo de um ponto pode ser feito. A partir do ponto P de cruzamento, todos os genes à direita são permutados, gerando novos indivíduos;

Figura 12 – Indivíduos pais (a) gerando, por meio do método de um ponto, os indivíduos filhos (b)



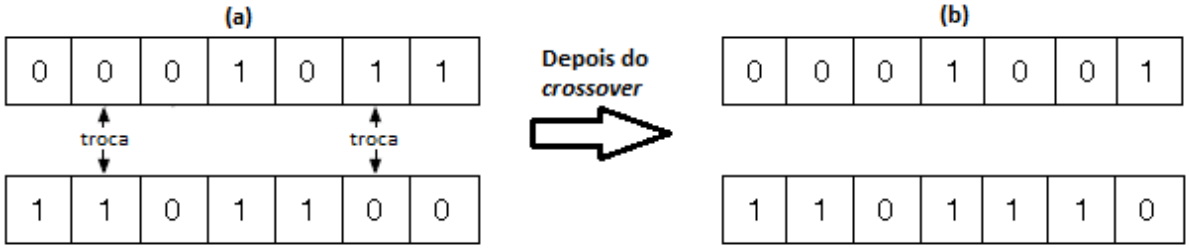
- De multi pontos**:** é o conceito mais amplo da ideia de formação de cromossomos através de pontos. Neste caso diversos pontos de cruzamento são usados. A figura a seguir, Figura 13, mostra como esse processo de multi ponto pode ser feito. Entre os pontos O e P de cruzamento, todos os genes são permutados, gerando novos indivíduos;

Figura 13 – Indivíduos pais (a) gerando, por meio do método de multi ponto, os indivíduos filhos (b)



- Uniforme**:** não utiliza pontos de cruzamento, mas sim, realiza troca de genes dentro de certa probabilidade. No *crossover* uniforme, cada um dos genes possui certa probabilidade de troca e as trocas são independentes entre si, diferente dos outros dois métodos, que obriga genes vizinhos, dentro de um espaço de cruzamento, a realizar uma permutação. A figura a seguir, Figura 14, mostra como esse processo de um ponto pode ser feito.

Figura 14 – Indivíduos pais (a) gerando, por meio do método uniforme, os indivíduos filhos (b)



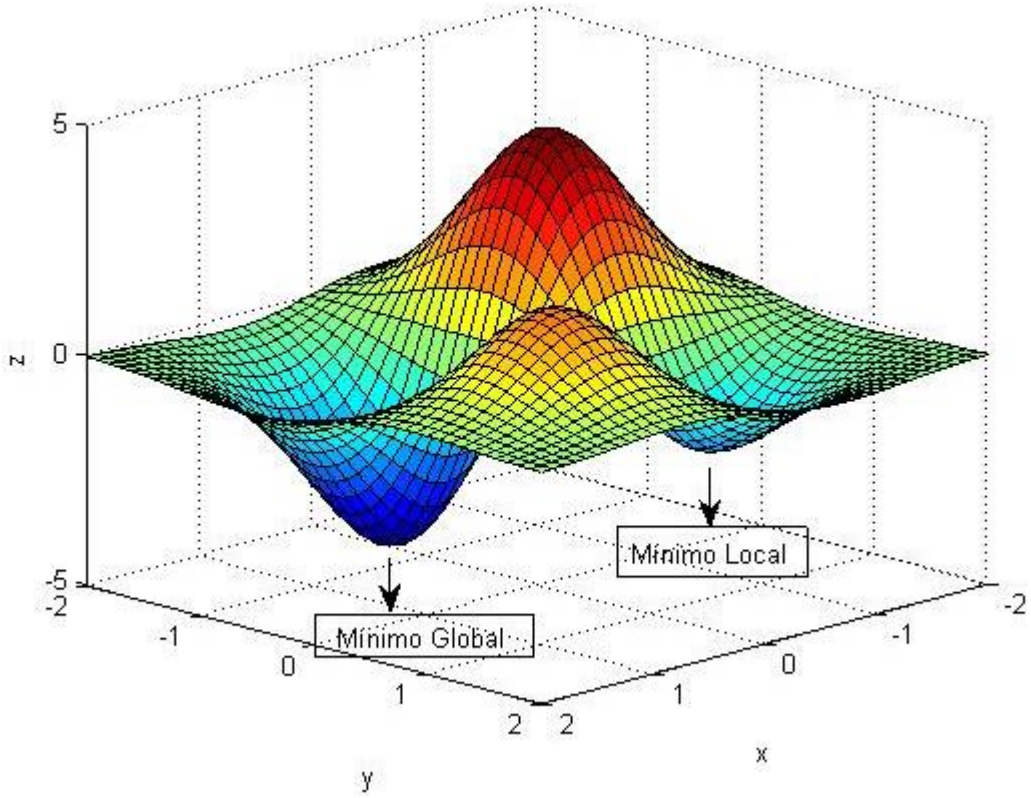
2.3.1.5 Mutação

Essa operação é necessária para a inserção e manutenção da diversidade genética dentro da população, alterando, de maneira aleatória, um ou mais genes de um determinado indivíduo.

Considere, novamente, o exemplo da função da Figura 9. Para o intervalo dado de busca ($-2 \leq x \leq 2$ e $-2 \leq y \leq 2$), nota-se a presença de dois pontos mínimos os quais estão marcados na Figura 15. O mais baixo dos dois pontos é chamado de mínimo global, enquanto o outro é o mínimo local. Um mínimo local é aquele ponto de mínimo para certa vizinhança, mas não para todo o espaço

de busca. Em processos de otimização, mínimos locais são problemas, visto que estes transmitem a falsa impressão que o melhor resultado foi encontrado.

Figura 15 – Mínimos global e local

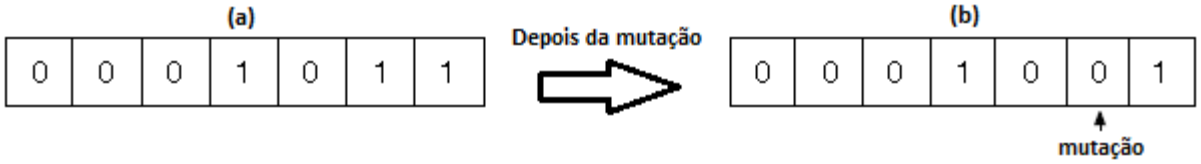


O que a mutação procura garantir é que a probabilidade de se alcançar qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois esse método muda levemente a direção da busca (Carvalho, 2009).

A operação de mutação é aplicada aos genes com uma probabilidade dada pela taxa de mutação; geralmente se utiliza uma taxa de mutação pequena, já que este é um operador genético secundário.

Para exemplificar, considere o caso na Figura 16, onde um indivíduo sofre uma mutação aleatória.

Figura 16 – (a) Indivíduo antes da mutação (b) Indivíduo depois da mutação



O processo de mutação desse exemplo consistiu na mudança de apenas um dos sete genes que constituem um indivíduo desse problema.

2.3.2 *Particle swarm optmization*

Assim como os algoritmos genéticos, a otimização por enxame de partículas (do inglês P*article Swarm Optmization*, PSO) também faz parte da área de Computação Bioinspirada. O PSO foi inspirado no comportamento de grupo de pássaros e cardumes de peixes, enquanto estes realizam sua busca por alimentos em uma determinada região. A posição do alimento é uma alusão a uma solução ótima do problema que se deseja resolver e a região seria a área de busca que o algoritmo deve percorrer. Ao se observar o comportamento do grupo, nota-se que o movimento de cada indivíduo é diretamente influenciado pelas suas próprias experiências e pelas experiências do grupo.

Inicialmente o PSO foi desenvolvido por (Kennedy & Eberheart, 1995) e implementa o comportamento de indivíduos, ou as chamadas partículas, dentro de um grupo (a população).

No início do algoritmo, são geradas partículas, dentro do espaço de busca, que representam possíveis soluções do problema. Cada uma é caracterizada por um vetor posição no espaço (equivalente aos genes do algoritmo genético) e um vetor velocidade. Esta última indica a direção e distância que determinada partícula deve percorrer.

Essas partículas são avaliadas pela função de *fitness*, que fornece uma nota. Depois que todas as partículas recebem suas notas, os vetores velocidade e posição de cada partícula são atualizados: o vetor velocidade é atualizado considerando a melhor solução encontrada pela partícula até então, a melhor solução já encontrada dentre todas as partículas e o valor atual da velocidade; o vetor posição é atualizado em seguida de acordo com a direção e a distância determinadas pelo vetor velocidade.

Os processos de avaliação e de atualização dos vetores velocidade e posição das partículas repetem-se, formando as iterações do PSO, e estas iterações são realizadas até que algum critério de parada, como número máximo de indivíduos analisados, tempo de processamento, etc., seja atingido. Ao final, uma ou mais soluções ótimas para o problema inicial devem ser encontradas.

Diferente do GA, os indivíduos do PSO não são eliminados e nem novos são gerados: os indivíduos que iniciam o processo são aqueles que terminam, com a diferença que estes se movimentam pela área de busca.

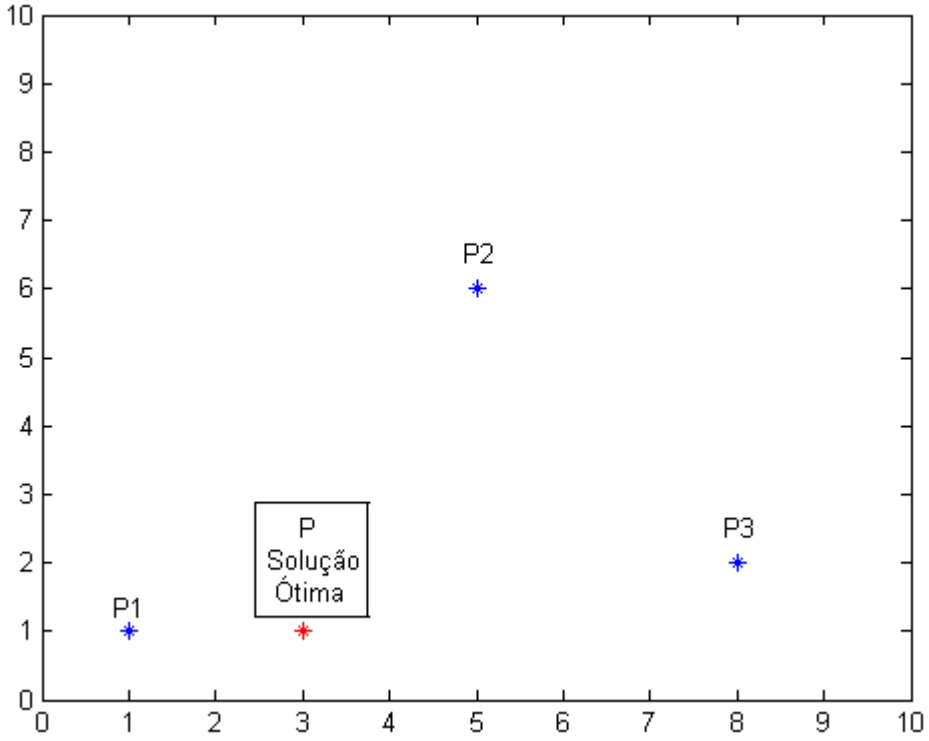
Nas próximas seções, os conceitos e termos sobre PSO serão melhores explicados.

2.3.2.1 Vetor posição

Considere o exemplo apresentado na Figura 17, na qual é mostrado um plano cartesiano (a região de busca); três pontos, as partículas P1, P2 e P3, formam uma população; e outro ponto P representa a solução ótima do problema. Para caracterizar uma partícula *i* no espaço, é atribuído a ela um vetor, denominado de vetor posição $x_i(t)$ da *t*-ésima iteração, que contém as coordenadas cartesianas da partícula. Dessa forma, pode-se observar que os vetores posição das partículas P1, P2 e

P3 e são, respectivamente, (1,1), (5,6), (8,2). Quanto mais complexo for um problema, maior será o número de coordenadas do vetor posição.

Figura 17 – Partículas em um espaço de busca juntamente com a solução ótima



2.3.2.2 Vetor velocidade

Para que as partículas possam encontrar uma solução considerada ótima, é necessário que estas se movimentem pelo espaço de busca. Para isso, cada uma das partículas tem um vetor velocidade associado. Esse vetor indica a direção, o sentido e a quantidade que uma determinada partícula deve se deslocar, sendo ele atualizado a cada iteração, para que um indivíduo não se mova sempre na mesma direção e sentindo.

Considere o exemplo da Figura 18. Nela são exibidos as mesmas partículas da Figura 17 com seus respectivos vetores velocidade indicados. De acordo com (Kennedy & Eberheart, 1995), uma relação para executar a atualização de posição, em função do vetor velocidade, pode ser:

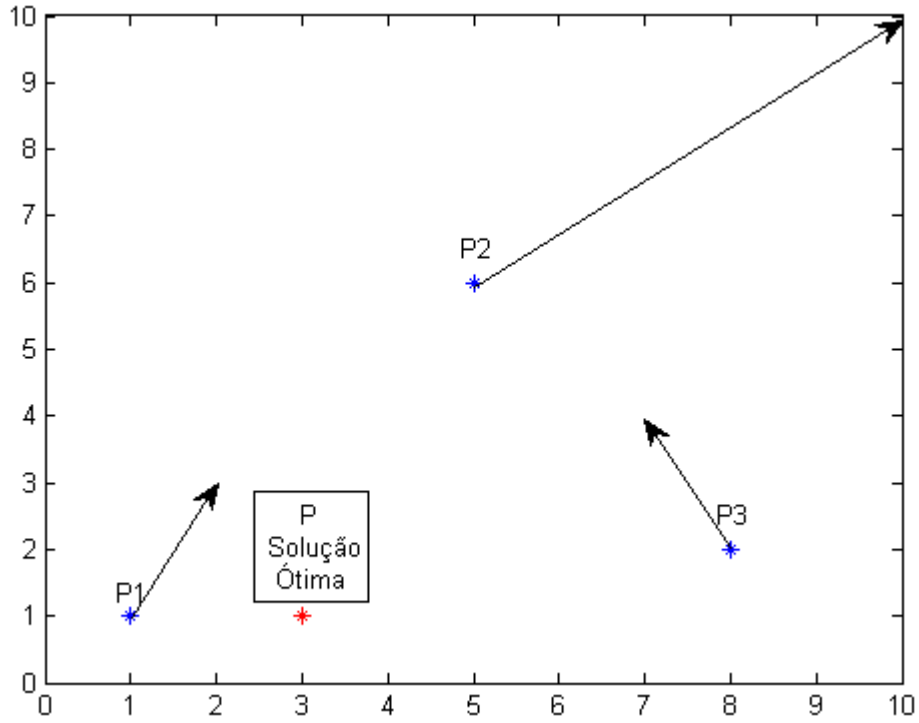
$$x_i(t+1) = x_i(t) + v_i(t+1) \quad \textbf{(9)}$$

onde $v_i(t+1)$ é o vetor velocidade da *i*-ésima partícula no instante $(t+1)$, $x_i(t)$ o vetor posição da *i*-ésima partícula no instante t e $x_i(t+1)$ o vetor posição da *i*-ésima atualizada no instante $t+1$.

Para o exemplo da Figura 18, os vetores velocidade de P1, P2 e P3 são, respectivamente, (1,2), (5,4) e (-1,2). As novas coordenadas de P1, P2 e P3 são exibidas na Figura 19.

Durante a busca, devem-se tomar precauções para que a partícula não ultrapasse os limites do espaço de busca.

Figura 18 – Partículas com seus vetores velocidade

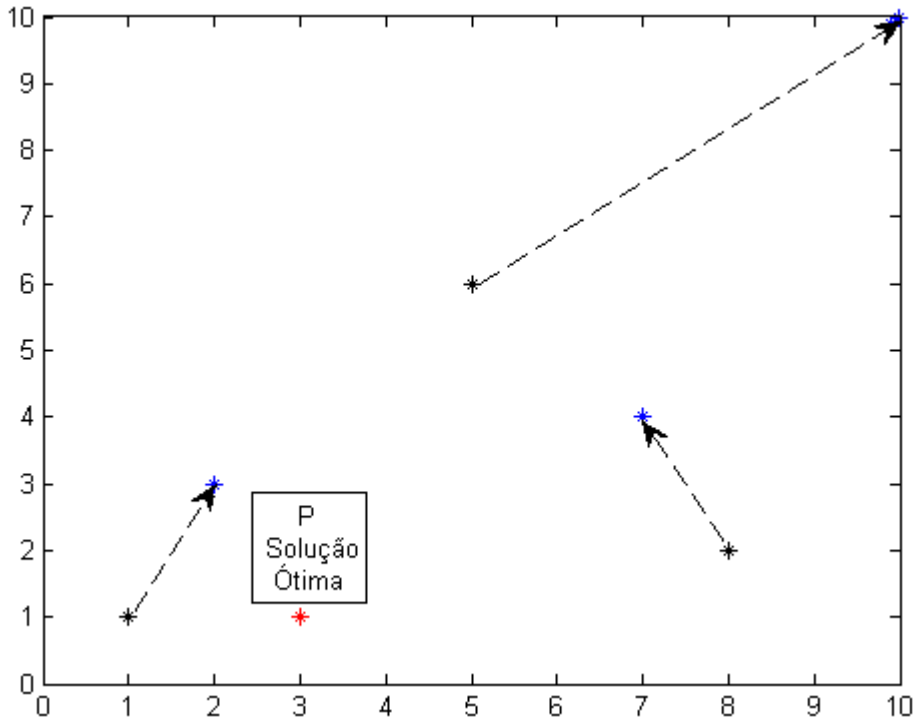


2.3.2.3 Conhecimento individual e global

Dois conceitos ou parâmetros são importantes para atualização dos vetores velocidade das partículas, os de conhecimentos individual e global. O primeiro está associado a cada partícula específica: quando ela se movimenta, suas notas oscilam com o passar das iterações. O vetor posição que proporciona a melhor solução encontrada pela partícula é chamada de conhecimento individual (p_i) e indica qual foi a melhor experiência por ela vivenciada durante todo o ciclo. Esse conhecimento, os indivíduos não o dividem entre si.

O conhecimento global (p_g) é associado ao enxame como um todo: a partícula que vivenciou a melhor experiência, ou seja, a partícula cujo vetor posição mais se aproximou de uma eventual solução ótima, repassa essa informação para seus vizinhos ou para o enxame todo.

Figura 19 – Atualização das coordenadas dos indivíduos



2.3.2.4 Atualização do vetor velocidade

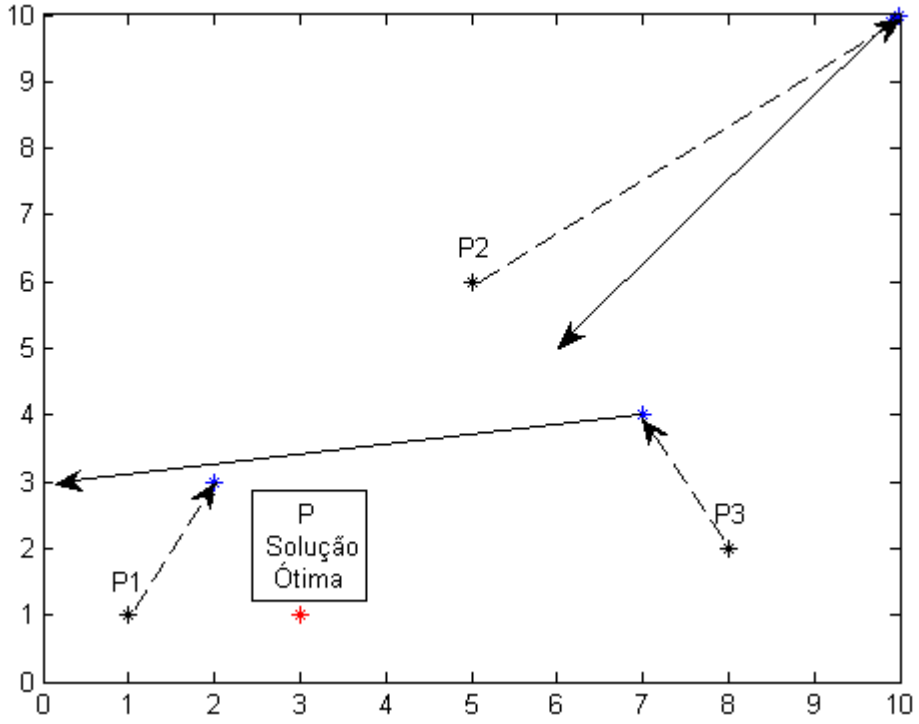
Com os vetores posições e os conhecimentos individuais e global atualizados, devem-se recalcular os vetores velocidades para que os indivíduos continuem a procurar pela solução ótima. Nesse cálculo, cada vetor velocidade é modificado em função das experiências individual e global do enxame. Uma possibilidade de se calcular os novos vetores velocidade, conforme (Kennedy & Eberheart, 1995), é apresentado na seguinte equação:

$$v_i(t+1) = v_i(t) + \alpha\theta_1\big(p_i - x_i(t)\big) + \beta\theta_2(p_g - x_i(t)) \quad \textbf{(10)}$$

onde $v_i(t+1)$ é o vetor velocidade da *i*-ésima partícula atualizada; α, β são constantes de aceleração; θ_1 e θ_2 são variáveis arbitrárias, com distribuição uniforme entre 0 e 1; e p_i e p_g são, respectivamente, os conhecimentos individuais das partículas e o global.

Utilizando-se da equação acima, as partículas do exemplo até então mostrado, poderiam ter seus vetores velocidades atualizados conforme se encontra na Figura 20 (considerando $\alpha = \beta = 1$, $\theta_1 = \theta_2 = 0{,}5$ e $p_i = p_g = 2$).

Figura 20 – Atualização dos vetores velocidade



Com o passar das iterações, as partículas vão tendendo a se aproximar cada vez mais da solução ótima.

2.3.3 *Simulated Annealing*

O algoritmo de recozimento simulado (do inglês, *Simulated Annealing*) (Kirkpatrick, Gellat, & Vecchi, 1983) provém da analogia aos processos termodinâmicos utilizados na metalurgia para obtenção de metais com estados de baixa energia, chamados de recozimento de metais.

Quando metais são esfriados de forma brusca, sua estrutura é deformada, gerando pontos de tensão. Uma estrutura deformada possui energia interna associada maior do que a de estruturas não deformadas, devido à desorganização dos cristais, tornando-se menos estável e, portanto, mais frágil.

Por outro lado, quando metais são resfriados lentamente, sua estrutura cristalina irá se recombinar de maneira a apresentar a menor energia interna, resultando num metal mais estável e menos frágil (Oliveira, 2006). Com a finalidade de se produzir cristais perfeitos, técnicas de recozimento são utilizadas: a temperatura do metal é aumentada até um pouco abaixo da sua temperatura de fusão. Nesse ponto, os átomos estão muito desordenados. A temperatura é então reduzida lentamente e, em seguida, dá-se certo tempo até que o sistema se acostume com a nova temperatura, na tentativa que sua estrutura se organize com o menor potencial energético possível.

O algoritmo de recozimento simulado (SA) tem o comportamento análogo ao recozimento dos metais, apresentado anteriormente, e opera da seguinte maneira: no início do programa, o algoritmo toma um indivíduo como atual a partir de uma solução aleatória ou uma previamente estabelecida.

Esse indivíduo possui certas características que o definem, análogo a uma estrutura cristalina a qual se deseja minimizar a energia interna, e estas podem ser representadas como elementos de um vetor. Por meio de uma função de *fitness*, o algoritmo calcula o grau de energia, nota dos indivíduos, que representa o quão distante da estabilidade global mínima se encontra a estrutura cristalina.

O algoritmo explora uma região em torno do individuo atual e o tamanho desta depende da temperatura da estrutura, que é reduzida passo a passo. Ao vasculhar uma região, novos indivíduos são encontrados, com diferentes características. Como só há um indivíduo atual no SA, o algoritmo deve tomar a decisão de trocar o indivíduo atual pelo novo ou mantê-lo. Para que essa decisão possa ser tomada, é necessário, primeiramente, que seja atribuída uma nota ao novo indivíduo. A decisão é tomada de acordo com a seguinte regra:

- se o grau de energia do novo indivíduo for menor que o atual, então ocorre a troca e o novo indivíduo passa a ser o atual;
- se o grau de energia do novo indivíduo for maior que o atual, a probabilidade de que o novo indivíduo substitua o atual será dado pela seguinte equação:

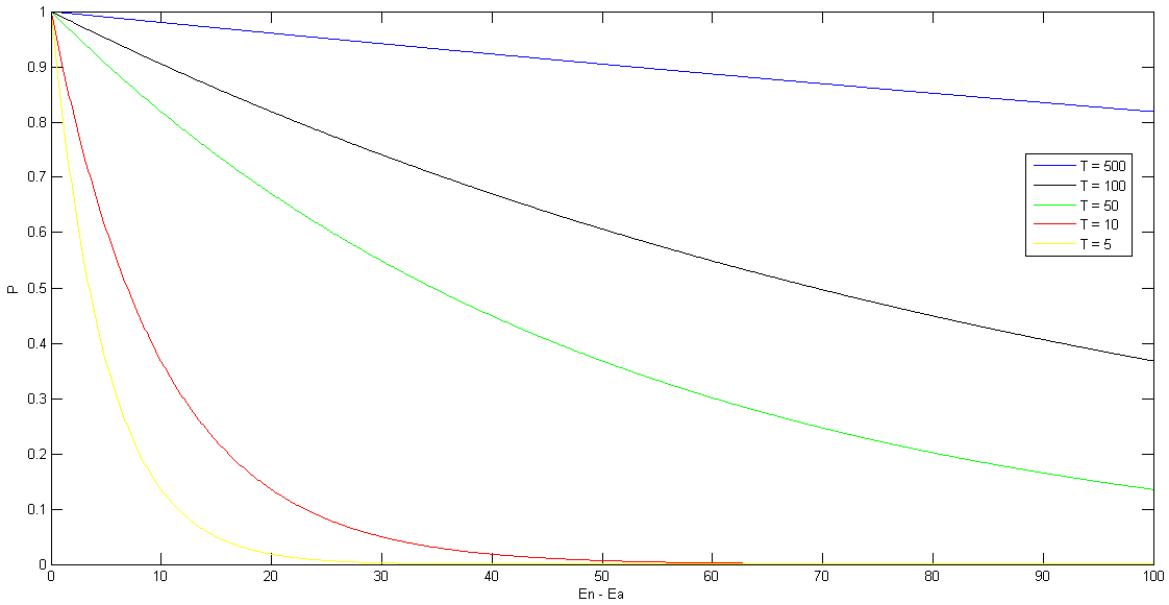
$$P = e^{-\frac{E_N - E_A}{T}} \quad \textbf{(11)}$$

onde P é a probabilidade de haver mudança, E_N é o grau de energia do novo individuo, E_A é o grau de energia do individuo atual e T é a temperatura do sistema.

Caso E_A seja menor do que E_N, quanto maior for a diferença energética, menor será a probabilidade de mudança e quanto maior a for temperatura, maior a chance de haver uma mudança para um estado de maior energia. Com isso se cria a possibilidade do algoritmo fugir de mínimos locais. Nota-se, portanto, que a temperatura do SA possui uma funcionalidade análoga à mutação do GA. Com o passar das transições, e a temperatura do sistema sendo decrescida, a chance de um candidato à solução ótima ser substituído por um pior reduz, diminuindo a chance de fuga de mínimos locais. Para contornar esse problema, o sistema pode apresentar um comportamento não monotônico, permitindo que haja um reaquecimento periódico. Na Figura 21 são exibidas as relações entre a probabilidade P em função da diferença energética para diferentes temperaturas.

Os processos de avaliação de um novo indivíduo, de tomar a decisão de troca do indivíduo atual pelo novo e a alteração da temperatura do sistema repetem-se, formando as iterações do SA, e estas iterações são realizadas até que algum critério de parada, como número máximo de indivíduos analisados, temperatura mínima, tempo de processamento, etc., seja atingido. Ao final, uma ou mais soluções ótimas para o problema inicial devem ser encontradas.

Figura 21 – Gráfico da probabilidade P de que um novo indivíduo substitua o atual em função da diferença energética, $E_N - E_A$, para diferentes valores de temperatura



2.4 Ferramentas Utilizadas

2.4.1 HSpice e Eldo

Através dos simuladores de circuitos elétricos HSpice e Eldo é possível simular o comportamento de vários circuitos em diferentes tipos de análises, como transitória, AC e DC. Essas análises serão utilizadas para medir os valores de ganho, banda de ganho unitário, s*lew rate*, CMRR, *offset*, ruído e PSRR dos AmpOps.

Essas ferramentas utilizam modelos de dispositivos MOS fornecidos pelas *foundries*, que permitem a simulação de forma mais precisa dos circuitos estudados. Para este texto, a tecnologia aplicada será da AMS (*Austria Micro-Systems*) 0,35µm (AUSTRIAMICROSYSTEMS, 2003) (AUSTRIAMICROSYSTEMS, 2014). A AMS fornece modelos BSIM3v3 (*Berkeley Short-channel IGFET Model*) para os transistores MOS.

Com os resultados gerados pelos *softwares* de simulação será mensurado se os amplificadores operacionais testados estão de acordo com as especificações do projetista.

2.4.2 MatLab

MatLab é uma ferramenta de desenvolvimento e interpretação de aplicativos de natureza técnica com linguagem de programação própria de alto nível (MATLAB, 2014). Dentre os motivos pela escolha dessa ferramenta, estão:

- Facilidade de programação;

- Facilidade de implementação de interfaces gráficas, a fim de facilitar a utilização do programa pelo usuário;
- Facilidade em interligar o programa escrito na ferramenta MatLab com o simulador de circuitos elétricos;
- A existência de algoritmos de otimização, como o algoritmo genético, enxame de partículas e recozimento simulado, previamente implementados.

3 METODOLOGIA

3.1 Plataforma do Projeto

Na construção desta plataforma, buscou-se desenvolver um sistema que pudesse projetar e otimizar qualquer tipo de amplificador operacional desejado, independente da aplicação e complexidade que o circuito pudesse apresentar.

Para uma nova topologia de AmpOp, o usuário deve construir um arquivo de descrição do amplificador operacional que deseja otimizar. Esse arquivo tem como função definir os elementos que se encontram nesse circuito, como transistores, elementos passivos e fontes de alimentação, bem como estes elementos estão interligados. O usuário, ainda, deve obedecer tanto à linguagem própria de cada simulador elétrico, linguagem tipo SPICE, como também a algumas restrições impostas, como é apresentado a seguir:

- As variáveis que se desejam otimizar, que estão associadas aos parâmetros de largura e comprimento dos canais dos transistores MOS, de valores de resistores, capacitores e fontes de correntes, devem ser nomeados como 'X' seguido de um número que caracterizará cada variável. Caso se deseje, por exemplo, que o valor de uma fonte de corrente, em microampère, seja uma variável a ser otimizada, sendo esta a terceira variável, pode-se utilizar, na descrição do circuito, a expressão 'X3*1u';
- Qualquer constante do circuito, por exemplo, a capacitância de saída, que se deseja permitir a livre escolha, deve ser nomeada como 'M', seguida de um número que a caracterizará. Dessa forma, caso se deseje que o valor de uma capacitância, em picofarads, seja uma constante, sendo esta a segunda constante, pode-se utilizar, na descrição do circuito, a expressão 'M2*1p';
- Os nós dos elementos que estão ligados às fontes de alimentação V_{DD} e V_{SS} devem ser nomeados, respectivamente, como *vd* e *vs.* A declaração das fontes de tensão não deve ser colocada;
- Os nós de entrada negativa e positiva do amplificador operacional devem ser nomeados, respectivamente, como *in* e *ip*;
- A carga do AmpOp sempre deve estar entre o nó de saída, que deve ser escrito como *out*, e o nó *gnd* (terra);
- O comando

.options statfl=1 NXX Noelck=1 MEASFILE=1 NOTOP RUNLVL=1 dcon=1 ingold=1 o measDGT=8

deve ser incluído para o arquivo de simulação HSpice e o comando

.option NOASCII NOMOD NOOP NOPAGE NOTRC NOTRCLIB NODCINFOTAB Aex=1 numdgt=8 nojwdb

deve ser incluídos para o arquivo de simulação Eldo;

- O comando *.include param* deve ser incluído no fim do arquivo (antes do comando *.end*);

- O arquivo do circuito construído deve-se chamar *circuito.sp* para simulações em HSpice e *circuito.cir* para simulações em Eldo;

Além da descrição do circuito, o usuário deve elaborar a função *AreaCirMea*, que estima o valor da área do circuito. A área de um circuito integrado depende das dimensões de transistores e elementos passivos, como resistores e capacitores, de como estes são colocados no *layout*, das necessidades de casamento entre componentes, da experiência do projetista e outros fatores. Porém, uma estimativa da área pode ser realizada através apenas das dimensões dos dispositivos do circuito. A plataforma irá fazer uso desta função, passando as variáveis otimizáveis do circuito e recuperando a área final do circuito.

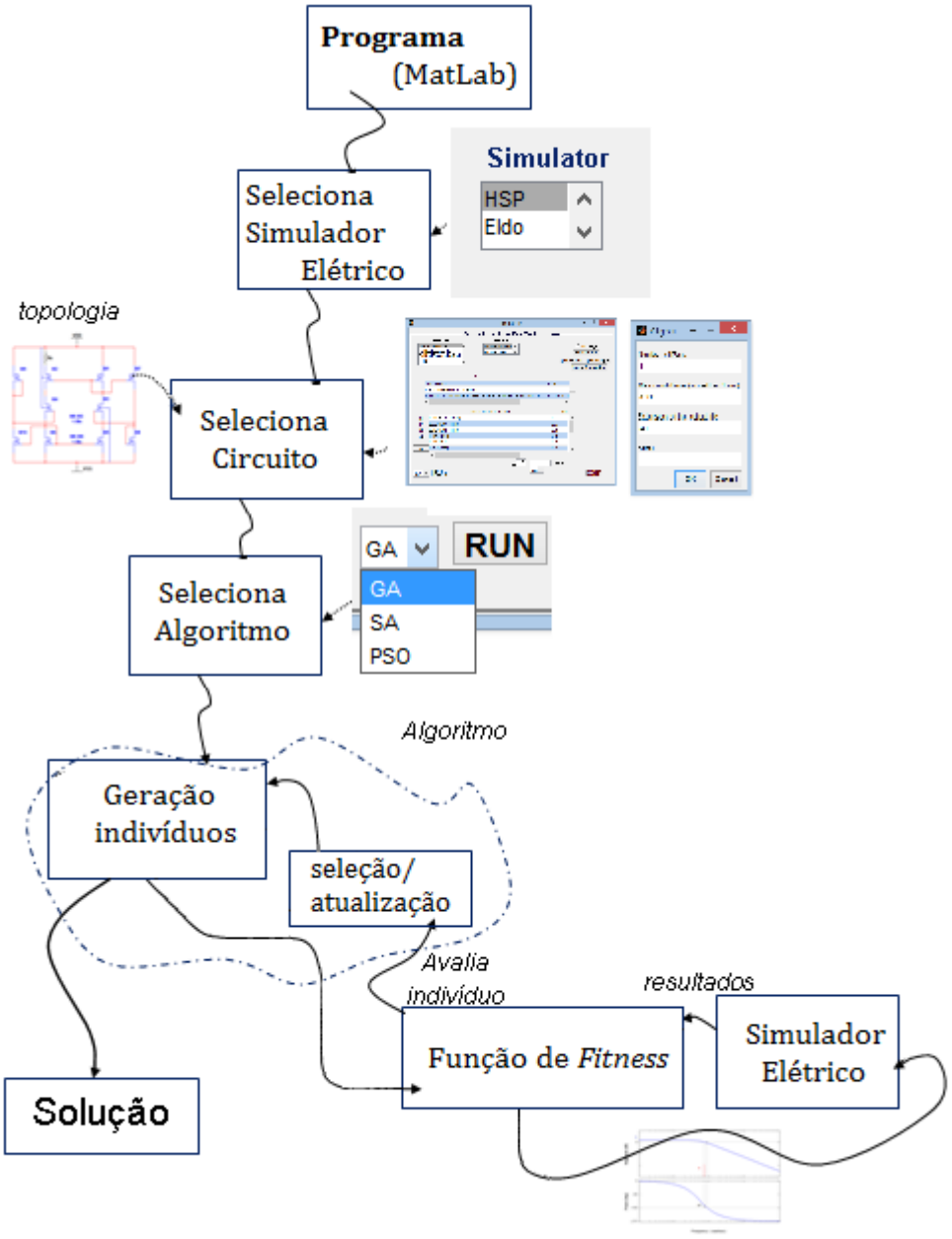
Cada circuito tem uma pasta associada a ele e que, por sua vez, contém:

- os arquivos *circuito.cir* e *circuito.sp* como ditos acima;
- O arquivo *AreaCirMea.m* que descreve, na linguagem do Matlab, a função que estima a área do circuito em questão;
- outra pasta chamada *results*;

Todas as pastas de topologias de AmpOp devem estar dentro de outra pasta do bloco amplificador operacional, chamada *Ampl*.

Na Figura 22, é mostrado o fluxograma dos processos executados pela plataforma durante o projeto e otimização de um circuito.

Figura 22 – Fluxograma da plataforma do projeto



Inicializado o programa, o usuário deve, primeiramente, decidir qual simulador elétrico ele deseja utilizar para, depois, inicializar o sistema de otimização de amplificadores operacionais, cuja interface de dados é apresentada no APÊNDICE A.

Inicializada a otimização, seguindo o fluxograma da Figura 22, o algoritmo entra na região de algoritmos destacada pelo contorno. Essa região da plataforma já está implementada na ferramenta MatLab. Nesse momento, algoritmo gera os primeiros indivíduos. Esses indivíduos são avaliados pela função de *fitness* (*fitnessEldo.m* descrita no APÊNDICE B) que é responsável por:

- Gerar os arquivos de parâmetros (*param*) para o simulador elétrico, que serão explicados adiante;

- Executar o simulador elétrico;
- Adquirir os dados gerados pelo simulador elétrico;
- Realizar os cálculos de avalição. Esses cálculos de avaliação estão relacionados com o desempenho do circuito em questão e explicados adiante.

Caso o algoritmo selecionado tenha sido o GA, depois dos circuitos avaliados, o algoritmo seleciona os melhores indivíduos; realiza o cruzamento entre eles para gerar novos candidatos e formar uma nova população; e realiza eventuais mutações nos indivíduos. Dessa maneira, temos uma geração de candidatos à solução ótima do problema e uma iteração completa-se.

Caso o algoritmo selecionado tenha sido o PSO, depois dos circuitos avaliados, verifica-se, para cada partícula e para o enxame todo, se novos conhecimentos individual e global foram adquiridos. Atualizados os conhecimentos individual e global, atualiza-se o vetor velocidade e o vetor posição de cada partícula. Com isso, os indivíduos adquirem novas características de posição e velocidade e uma iteração completa-se.

Caso o algoritmo selecionado tenha sido SA, depois que o indivíduo atual e o novo são avaliados, decide-se se o indivíduo atual será trocado pelo novo ou se o indivíduo atual será mantido. Decidido quem será o indivíduo atual, procura-se no espaço de busca por um novo indivíduo. Este será avaliado novamente pela função de *fitness* e uma interação completa-se.

O algoritmo selecionado continua executando até ser atingida uma condição de parada, estabelecida pelo usuário ou até ser feita uma interrupção forçada pelo usuário.

Ao fim de todo o processo, o arquivo paramopT terá as variáveis de projeto e o desempenho do melhor circuito encontrado e que pode ser projetado.

Como a convergência da otimização depende de diversos fatores tais como condições iniciais, o número de indivíduos analisados, o número de otimizações, etc., o projeto de um circuito eletrônico por meio de métodos metaheuristicos deve ser realizado através de diversas tentativas e, se possível, com o refinamento dos resultados por meio de uma segunda metaheurística.

3.2 Função de *Fitness*

A função de *fitness* deve gerar arquivos *param*, executar a simulação, ler os resultado e, por fim, calcular o desempenho do circuito. Essa função é única e empregada por todas as metaheuristicas.

Conforme explicado na seção 3.1, o usuário deve criar um arquivo com as descrições do amplificador operacional. Este arquivo não contém os valores das variáveis a serem otimizadas, nem os valores de eventuais constantes, muito menos comandos de simulações, para avaliar o desempenho do circuito. Essas informações todas são colocadas nos arquivos *param*, arquivos de parâmetros, criados pela função de *fitness*. O arquivo *param* tem:

- o valor das variáveis de otimização e das constantes para cada individuo;
- comandos que determinam as simulações que devem ser executadas;

- comandos que determinam as saídas avaliadas;
- comando de inclusão de modelos;

São necessários quatro diferentes arquivos *param* para avaliar um AmpOp, que são preparados por funções parâmetros.

Depois que o arquivo de parâmetros foi gerado, a função de *fitness* executa o simulador elétrico, que gera arquivos de saídas com os resultados. Devido à existência de dois diferentes simuladores elétricos, cada um com comandos e arquivos de saída próprios, foi necessário construir duas funções de *fitness*: *fitnessHSP.m* e a *fitnessEldo.m.* Ambas realizam as mesmas tarefas e utilizam das mesmas funções de parâmetro, mas de formas levemente distintas, devido ao formato dos arquivos de saída de cada simulador elétrico.

Com os resultados das simulações lidas, a função finalmente avalia o circuito, calculando a sua nota conforme as especificações do usuário.

Para o cálculo da nota de um circuito consideraram-se diversos parâmetros de desempenho, associando uma nota a cada um deles e, por fim, faz-se média ponderada de todas as notas (função de *fitness* agregada). Assim o valor retornado (F_T) pela função de *fitness* é calculado por meio dos valores de "n" parâmetros de desempenho, com funções de avaliação (F_i) associadas a eles, e de pesos de ponderação (p_i), fornecidos pelo projetista, com a seguinte equação:

$$F_T = \left(\sum_{i=1}^{n} F_i p_i \right)^2 \quad \textbf{(12)}$$

Um dos grandes desafios na utilização de metaheurísticas na otimização de sistemas é a escolha dos parâmetros de desempenho, a elaboração das funções de avaliação e a escolha dos pesos de ponderação. Os parâmetros de desempenho escolhidos devem ser capazes de avaliar plenamente o funcionamento do circuito em diversas situações.

Para calcular o desempenho global de um amplificador operacional foram realizadas as seguintes medidas:

- *Offset* de entrada;
- Medida da potência;
- Estado de fraca inversão dos transistores;
- Ganho de modo comum;
- Ganho diferencial;
- Frequência e fase de ganho unitário;
- Ruído de saída;
- Ganho das variações das fontes de alimentação;
- *Slew rate*.

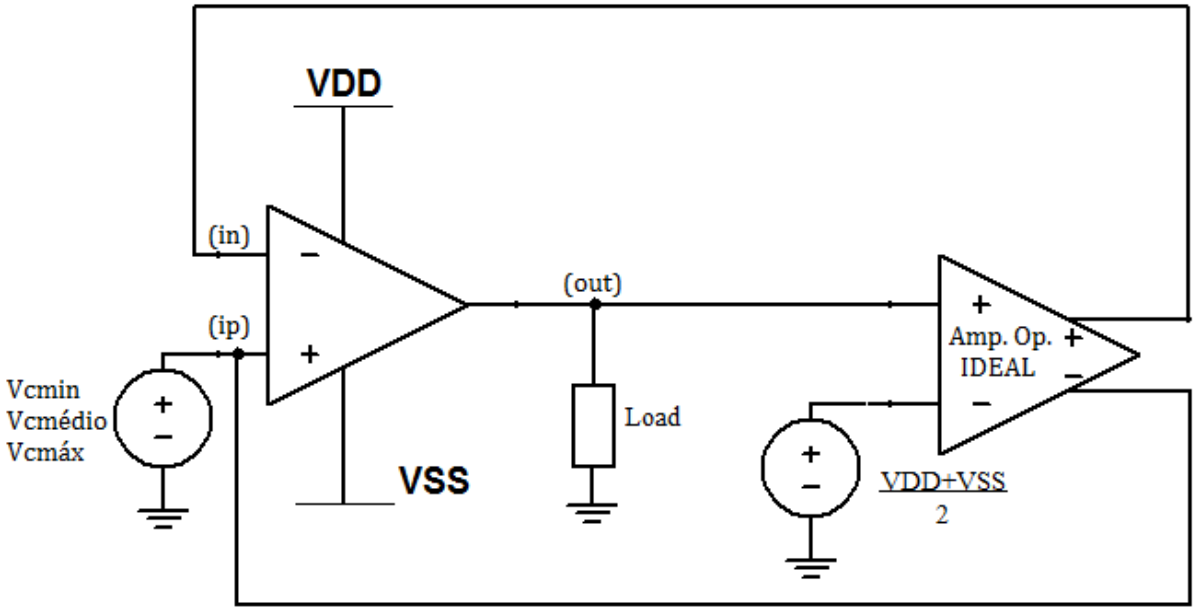
Para encontrar essas medidas, é necessário montar configurações diversas com os amplificadores analisados e executar comandos de simulação diferentes. Essas configurações e comandos são apresentados nas seções a seguir.

3.2.1 Medida do *offset*, potência e fraca inversão

Conforme explicitado na seção 2.2.5, uma tensão de *offset* natural surge na saída dos amplificadores operacionais. Por se tratar de uma medida que posteriormente será utilizada para calcular o desempenho do circuito, deseja-se encontrar a tensão de *offset* de entrada do circuito. Além disso, deseja-se mensurar a potência DC que este circuito consome e se algum dos transistores, especificados pelo projetista, trabalham em fraca inversão.

Para determinar o *offset*, a potência DC do AmpOp e o estado de operação de certos transistores, a função de *fitness* aciona a função *paramEx.m* (descrita no APÊNDICE B). Esta função gera um arquivo *param* que descreve e simula a montagem exibida na Figura 23. Nesta montagem um amplificador operacional diferencial ideal (Amp.Op.IDEAL) é usado.

Figura 23 – Circuito gerado por *paramEx* para calculo de *offset* e potência



A análise DC do simulador elétrico é empregada.

O AmpOp testado possui na sua entrada positiva uma fonte de tensão DC, que dá a tensão de modo comum, e na entrada negativa, a saída do Amp.Op.IDEAL. Na saída do AmpOp estão tanto a carga do circuito quanto a entrada positiva do Amp.Op.IDEAL. Por fim, a entrada negativa do Amp.Op.IDEAL está ligada a outra fonte de tensão DC, que fornece a tensão desejada na saída do AmpOp testado.

Dado o alto ganho do Amp.Op.IDEAL, devemos ter que a tensão de saída do AmpOp atinge o valor de (Vdd+Vss)/2 e que a diferença entre as tensões em *in* e *ip* é o *offset* procurado.

Para a simulação, são usados três valores distintos de tensão na entrada positiva do amplificador testado: Vcmín, Vcmédio e Vcmáx. A primeira e a última são as tensões de modo comum mínima e máxima, respectivamente, fornecidas pelo projetista. A segunda tensão é a média aritmética das outras duas. Deseja-se avaliar a tensão de *offset* dos pontos de operação extremos e do ponto intermediário.

Além do *offset*, é calculada a potência DC que o circuito consome. Para isso se mede a máxima corrente, I_{max}, que passa pela fonte de alimentação Vdd, para as três situações de tensão na entrada positiva, e a potência é dada por:

$$P = (V_{dd} - V_{ss})I_{máx} \tag{13}$$

Por fim, a função *paramEx.m* é utilizada para avaliar, caso o usuário tenha solicitado, o estado de operação de certos transistor. Para isso, o simulador calcula, para os transistores escolhidos, o fator de inversão K_{OP}:

$$K_{OP} = \frac{\frac{|I_D|}{V_t(1+\frac{g_{mb}}{g_m})} - g_m}{gm} \tag{14}$$

onde I_D é a corrente que flui pelo dreno do transistor, V_t é a tensão térmica, g_m é a transcondutância do transistor e g_{mb} é a transcondutância de efeito de corpo do transistor.

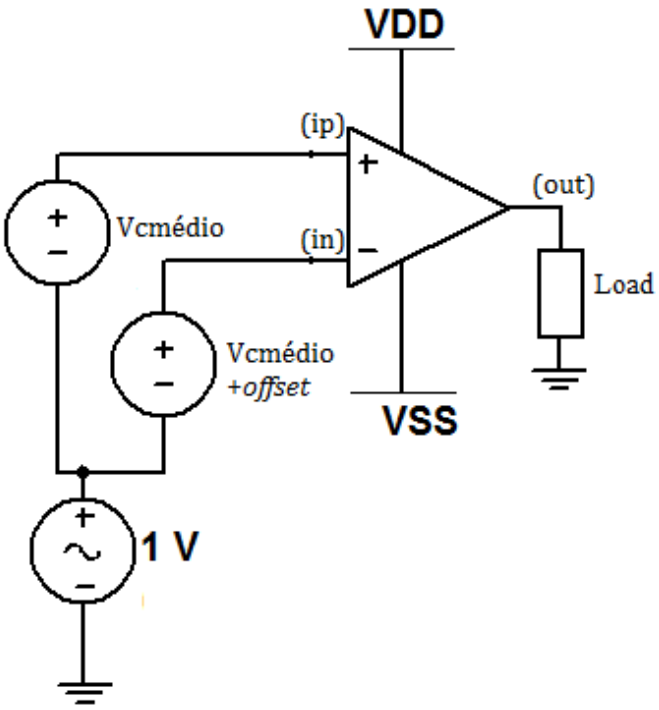
Será considerado que o transistor opera em fraca inversão quando $K_{OP} < 0{,}12$.

3.2.2 Medida do ganho de modo comum

Deseja-se encontrar o valor do ganho de modo comum, A_C, para que ele possa ser usado, posteriormente, no cálculo de desempenho do circuito.

Para determinar A_C, a função de *fitness* aciona a função *paramganhoMC.m* (descrita no APÊNDICE B). Esta função gera um novo arquivo *param* (sobrescrevendo o antigo) que descreve e simula a montagem exibida na Figura 24.

Figura 24 – Circuito gerado por *paramganhoMC.m* para o cálculo do ganho de modo comum



Nesta simulação, a análise AC do simulador elétrico é empregada.

O AmpOp a ser testado tem em sua entrada positiva uma fonte DC com o valor Vcmédio, tensão média calculada na simulação da seção anterior, e na entrada negativa outra fonte DC com o valor (Vcmédio+*offset*), sendo o *offset* a tensão de *offset* determinado anteriormente. Adicionalmente, conecta-se uma fonte AC de 1,0 V de amplitude a cada uma das fontes de tensão DC.

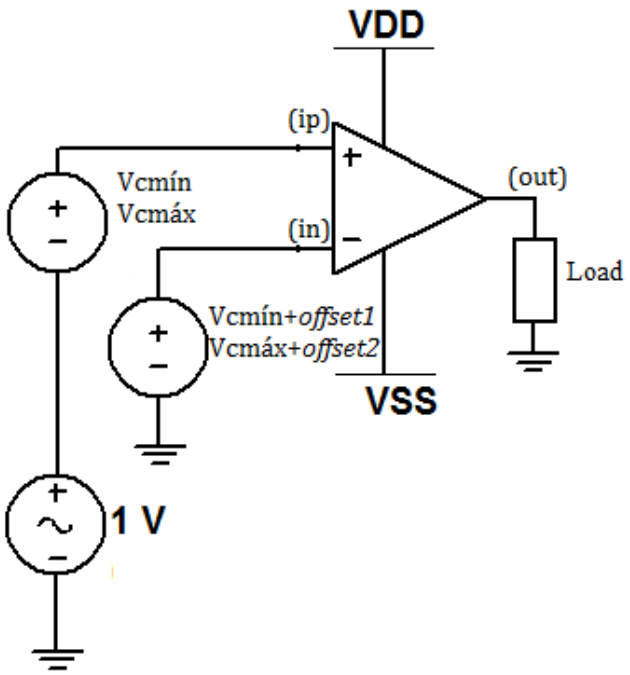
Para a simulação, varia-se a frequência da fonte AC na faixa especificada para o CMRR pelo projetista e mede-se o máximo ganho que esse circuito possui. Este será o ganho de modo comum $A_{Ccircuito}$, em decibéis (dB). Para o cálculo do CMRR é necessário medir o ganho diferencial. Essa medição é feita a seguir.

3.2.3 Medida do ganho diferencial, banda e fase de ganho unitário e ruído

Deseja-se obter o ganho diferencial, *G*, a banda de ganho unitário, f_o, e fase de ganho unitário que esse circuito possui. Adicionalmente, deseja-se medir a tensão média que o ruído produz na saída desse amplificador. Esses valores serão utilizados nos cálculos de desempenho do circuito, posteriormente.

Para determinar o ganho diferencial, a banda e fase de ganho unitário e o ruído, a função de *fitness* aciona a função *paramAC.m* (descrita no APÊNDICE B). Esta função gera um novo arquivo *param* que descreve e simula a montagem exibida na Figura 25.

Figura 25 – Circuito gerado por *paramAC.m* para cálculo de ganho diferencial, banda e fase de ganho unitário e ruído



A análise AC do simulador elétrico é empregada.

O AmpOp testado tem na sua entrada positiva uma fonte DC, que assume os valores Vcmín e Vcmáx, e na sua entrada negativa uma fonte DC, que assume os valores (Vcmín+*offset1*) e (Vcmáx+*offset2*), realizando a compensação de *offset*. Note que, para as diferentes tensões Vcmín e Vcmáx, diferentes tensões de *offset* devem ser utilizadas, *offset1* e *offset2*. Para avaliar qual o ganho diferencial do amplificador, bem como a frequência e fase de ganho unitário e o ruído do circuito, conecta-se uma fonte AC de 1,0 V de amplitude em série com a fonte de tensão DC da entrada positiva.

Uma simulação AC, com frequência variando entre a frequência mínima da faixa especificada para o CMRR e quatro vezes a frequência de ganho unitário especificada pelo usuário, é realizada. Com essa simulação mede-se:

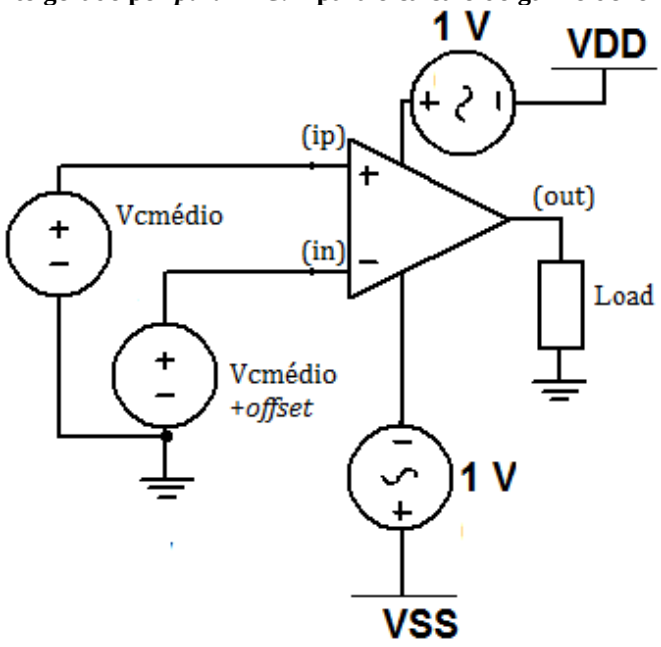
- o máximo valor de tensão obtido na saída. Esse valor medido será o ganho diferencial. Dois valores de ganho são obtidos, um para cada uma das tensões DC das entradas;
- a frequência onde o ganho diferencial é 1,0 V/V e a fase nesse ganho. Esses valores medidos de frequência e fase serão, respectivamente, a frequência, medida em Hz, e fase, medida em graus, de ganho unitário. Dois valores de frequência e fase são obtidos, um para cada uma das tensões DC das entradas;
- o ruído total referido a saída do circuito. Dois valores são obtidos, um para cada uma das tensões DC das entradas;

3.2.4 Medida do ganho devido a variações das fontes de alimentação

Deseja-se encontrar o valor do ganho na saída devido a variações das fontes de alimentação, A_P, para que ele possa ser usado, posteriormente, no cálculo de desempenho do circuito.

Para determinar o ganho devido a variações das fontes de alimentação, A_P, a função *paramAC.m* também descreve e simula a montagem exibida na Figura 26, utilizando o mesmo arquivo *param* da seção anterior (utiliza o comando *.alter*)

Figura 26 – Circuito gerado por *paramAC.m* para o cálculo do ganho de fonte de alimentação



A análise AC do simulador elétrico é empregada.

O AmpOp a ser testado tem em suas entradas uma fonte DC com valor Vcmédio, entrada positiva, e uma fonte DC com valor (Vcmédio+*offset*), entrada negativa, sendo feita a compensação do *offset* do amplificador. Para avaliar o efeito de variações nas tensões de alimentação, aplica-se com a fonte de alimentação DC de valor Vdd, que esta aplicada ao terminal de alimentação positivo, um sinal AC de 1,0 V de amplitude, ou com a fonte de alimentação DC de valor Vss, que esta aplicada ao terminal de alimentação negativa, um sinal AC de 1,0 V de amplitude.

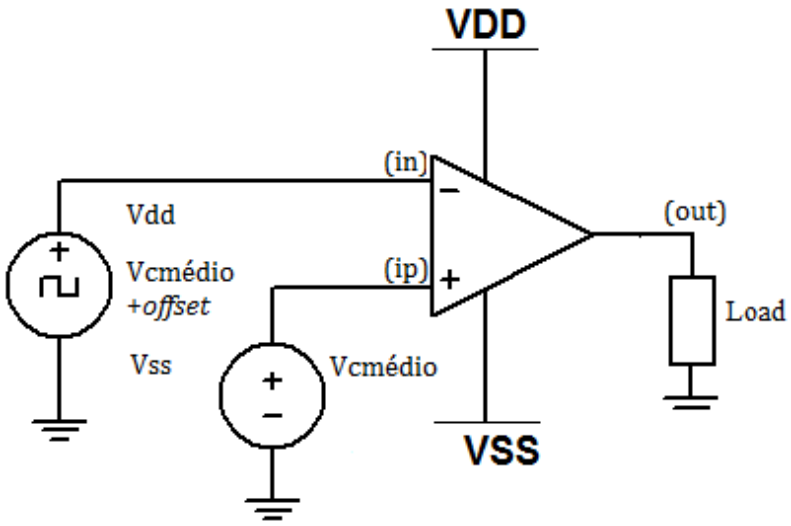
Uma simulação AC, com frequência variando entre a frequência mínima da faixa especificada para o CMRR e quatro vezes a frequência de ganho unitário especificada pelo usuário, é realizada e o máximo valor obtido na saída, é então medido. Esse valor medido será o ganho máximo entre a alimentação positiva e a saída, $A_{Pcircuito}^{+}$, no caso do sinal AC aplicado no terminal positivo, ou entre a alimentação negativa e a saída, $A_{Pcircuito}^{-}$, no caso do sinal AC aplicado no terminal negativo.

3.2.5 Medida do *slew rate*

Deseja-se encontrar o valor do *slew rate* para que ele possa ser usado, posteriormente, no cálculo de desempenho do circuito.

Para determinar o *slew rate*, a função de *fitness* aciona a função *paramSlewRate.m* (descrita no APÊNDICE B). Esta função gera um novo arquivo *param* que descreve e simula a montagem exibida na Figura 27.

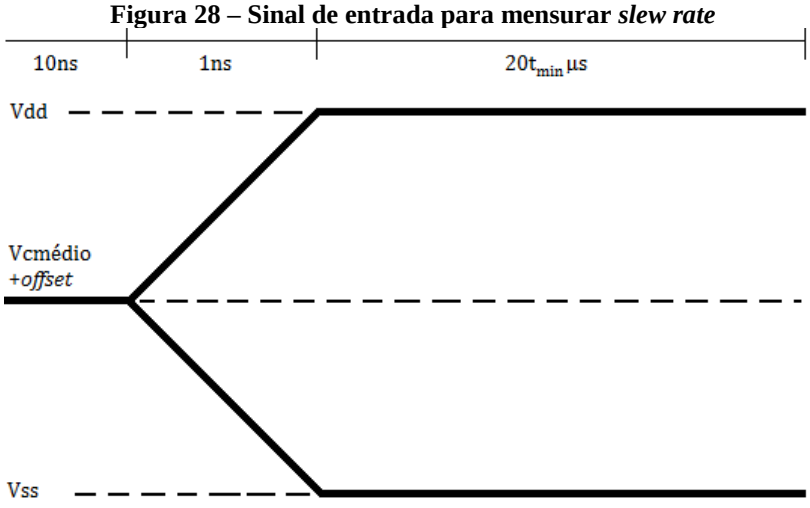
Figura 27 – Circuito gerado por *paramSlewRate.m* para o cálculo do *Slew Rate*



A análise transitória do simulador elétrico é empregada.

O AmpOp a ser testado tem em suas entradas uma fonte DC com valor Vcmédio, entrada positiva, e uma fonte que gera pulsos, entrada negativa. Para avaliar o *slew rate*, aplica-se, primeiramente, um pulso com valor inicial (Vcmédio+*offset*) e valor final Vdd e, posteriormente, um pulso com valor inicial (Vcmédio+*offset*) e valor final Vss, Figura 28.

Figura 28 – Sinal de entrada para mensurar *slew rate*



Para realizar as simulações deve-se calcular o intervalo de tempo que a fonte pulsante ficará em alto, que por questões de simetria, também será o intervalo em baixa. Para isso, utiliza-se do conceito da taxa de variação de tensão, expresso pela equação (4), para estimar o mínimo tempo, t_{min}, necessário para que o circuito, com um *slew rate* mínimo fornecido pelo projetista, consiga elevar seu sinal de saída, a partir do valor médio, até o valor máximo (ou abaixar até o mínimo). Considerando desprezível o *offset* para este cálculo, a variação de tensão entre o ponto inicial e final é expressa pela média aritmética das tensões de alimentação. O cálculo é introduzido na equação (15):

$$t_{mín} = \frac{V_{dd} + V_{ss}}{2(SlewRate_{desejado})} \tag{15}$$

onde $SlewRate_{desejado}$ é o *slew rate* especificado pelo projetista.

Para que circuitos com taxas de variação de tensão bem menores que a esperada possam ser devidamente analisados, utiliza-se o intervalo em alto com valor de $20t_{mín}$.

Com os intervalos em alto e baixo estabelecidos, uma simulação transitória, variando entre zero e $20t_{min}$, é realizada e a máxima taxa de variação de tensão de saída, em V/s, é então medida. Esse valor medido será o *slew rate* de descida, no caso do pulso de subida, ou o *slew rate* de subida, no caso do pulso de descida.

Depois que todas as medidas forem realizadas, as notas dos indivíduos são calculadas de acordo com a equação (12). Contudo, para que essa nota possa ser calculada, é necessário, antes, calcular os valores das funções de avaliação de cada especificação.

3.3 Função de Avaliação para o Ganho (F_1)

Para o Ganho Diferencial, utilizam-se os dados obtidos nas simulações descritas na seção 3.2.3. A função de avaliação referente ao ganho, F_1, é dada por:

$$F_1 = \sum_{i=1}^{2} F_{1,i} \tag{16}$$

A função F_1 é a soma das funções $F_{1,1}$ e $F_{1,2}$ que levam em consideração cada um dos dois ganhos que são mensurados durante a simulação. A relação que exprime $F_{1,i}$ é dada a seguir:

$$F_{1,i} = \begin{cases} 0 & para \quad Gi_{circuito} \geq G \\ \dfrac{G - Gi_{circuito}}{G} & para \quad Gi_{circuito} < G \end{cases} \tag{17}$$

onde $Gi_{circuito}$ é o i-ésimo ganho de malha aberta simulado para diferentes tensões de entrada (Vcmín e Vcmáx), segundo seção 3.2.3, e G é o ganho de malha aberta proposto pelo usuário.

Nas outras funções de avaliação será considerado como o ganho de malha aberta do circuito testado ($G_{circuito}$) o valor mínimo entre $G1_{circuito}$ e $G2_{circuito}$.

3.4 Função de Avaliação para a Frequência de ganho unitário (F_2)

Para a Frequência de Ganho Unitário, utilizam-se os dados obtidos nas simulações descritas na seção 3.2.3. A função de avaliação referente à frequência, F_2, é dada por:

$$F_2 = \sum_{i=1}^{2} F_{2,i} \tag{18}$$

A função F_2 é a soma das funções $F_{2,1}$ e $F_{2,2}$ que levam em consideração cada uma das duas frequências de ganho unitário que são mensuradas durante a simulação. A relação que exprime $F_{2,i}$ é dada a seguir:

$$F_{2,i} = \begin{cases} 0, & para\ fi_{circuito} \geq f_o \\ \dfrac{f_o - fi_{circuito}}{f_o}, & para\ fi_{circuito} < f_o \end{cases} \tag{19}$$

onde $fi_{circuito}$ é a i-ésima frequência de ganho unitário simulada para diferentes tensões de entrada (Vcmín e Vcmáx), segundo seção 3.2.3, e f_o é a frequência de ganho unitário proposta pelo usuário.

3.5 Função de Avaliação para o CMRR (F_3)

Para o CMRR, utilizam-se os dados obtidos nas simulações descritas nas seções 3.2.2 e 3.2.3. A função de avaliação referente ao CMRR, F_3, é dada por:

$$F_3 = \sum_{i=1}^{2} F_{3,i} \tag{20}$$

A função F_3 é a soma das funções $F_{3,1}$ e $F_{3,2}$ que levam em consideração cada um dos dois ganhos de modo comum que são mensurados durante a simulação. A relação que exprime $F_{3,i}$ é dada a seguir:

$$F_{3,i} = \begin{cases} 0 & para\ CMRRi_{circuito} \geq CMRR \\ \dfrac{CMRR - CMRRi_{circuito}}{CMRR} & para\ CMRRi_{circuito} < CMRR \end{cases} \tag{21}$$

onde $CMRRi_{circuito}$ é a *i*-ésima taxa de rejeição de modo comum e CMRR é a taxa de rejeição de modo comum proposta pelo usuário.

O $CMRRi_{circuito}$ é calculado da seguinte forma:

$$CMRRi_{circuito} = Gi_{circuito} - A_{Ccircuito} \qquad (i = 1,2) \tag{22}$$

onde $Gi_{circuito}$ é o *i*-ésimo ganho de malha aberta (para Vcmín e Vcmáx) apresentado na seção 3.3 e $A_{Ccircuito}$ é o ganho de modo comum simulado, segunda a seção 3.2.2.

3.6 Função de Avaliação para o margem de fase (F_4)

Para a Margem de Fase, utilizam-se os dados obtidos nas simulações descritas na seção 3.2.3. A função de avaliação referente à margem de fase, F_4, é dada por:

$$F_4 = \sum_{i=1}^{2} F_{4,i} \tag{23}$$

A função F_4 é a soma das funções $F_{4,1}$ e $F_{4,2}$ que levam em consideração cada uma das duas fases de ganho unitário que são mensuradas durante a simulação. A relação que exprime $F_{4,i}$ é dada a seguir:

$$F_{4,i} = \begin{cases} 0 & para \quad MF_{inferior} \leq MFi_{circuito} \leq MF_{superior} \\ \dfrac{MFi_{circuito} - MF_{superior}}{MFi_{circuito}} & para \quad MFi_{circuito} > MF_{superior} \\ \dfrac{MF_{inferior} - MFi_{circuito}}{MF_{inferior}} & para \quad MFi_{circuito} < MF_{inferior} \end{cases} \tag{24}$$

onde $MFi_{circuito}$ é a *i*-ésima margem de fase e $MF_{inferior}$ e $MF_{superior}$ são as margens de fase inferior e superior, respectivamente, desejadas pelo usuário.

A $MFi_{circuito}$ é calculada da seguinte forma:

$$MFi_{circuito} = 180° - |\theta i_{0dB}| \qquad (i = 1,2) \tag{25}$$

onde θi_{0dB} é a *i*-ésima fase de ganho unitário simulado para diferentes tensões de entrada (Vcmín e Vcmáx), segundo seção 3.2.3.

3.7 Função de Avaliação para o *Slew Rate* (F_5)

Para o *Slew Rate*¸ utilizam-se os dados obtidos nas simulações da seção 3.2.5. A função de avaliação F_5 do *slew rate* é dada por:

$$F_5 = \begin{cases} 0 & para \quad SlewRate \leq SlewRate_{circuito} \\ \dfrac{SlewRate - SlewRate_{circuito}}{SlewRate} & para \quad SlewRate > SlewRate_{circuito} \end{cases} \quad \textbf{(26)}$$

onde $SlewRate_{circuito}$ é a taxa de variação de tensão mínima medida entre os *slew rate* de subida e descida, conforme seção 3.2.5, e *SlewRate* é a taxa de variação de tensão proposta pelo usuário.

3.8 Função de Avaliação para o *PSRR* (F_6)

Para o PSRR, utilizam-se os dados obtidos nas simulações descritas na seção 3.2.4. A função de avaliação referente ao ganho, F_6, é dada por:

$$F_6 = \begin{cases} 0 & para \quad PSRR \leq PSRR_{circuito} \\ \dfrac{PSRR - PSRR_{circuito}}{PSRR} & para \quad PSRR > PSRR_{circuito} \end{cases} \quad \textbf{(27)}$$

onde $PSRR_{circuito}$ é o mínimo entre os valores de $PSRR^+$ e $PSRR^-$ e PSRR é a taxa de rejeição de alimentação proposta pelo usuário.

$PSRR^+$ e $PSRR^-$ são calculados da seguinte forma:

$$PSRR^+ = G_{circuito} - {A_{Pcircuito}}^+ \quad \textbf{(28)}$$

$$PSRR^- = G_{circuito} - {A_{Pcircuito}}^- \quad \textbf{(29)}$$

onde ${A_{Pcircuito}}^+$ é o ganho da fonte de alimentação positiva, segundo seção 3.2.4, e ${A_{Pcircuito}}^-$ é o ganho da fonte de alimentação negativa, segundo 3.2.4.

3.9 Função de Avaliação para a Potência (F_7)

Para a Potência do circuito, utilizam-se os dados obtidos nas simulações descritas na seção 3.2.1. A função de avaliação referente à potência, F_7, é dada por:

$$F_7 = \frac{Pot_{circuito}}{Pot} \quad \textbf{(30)}$$

onde $Pot_{circuito}$ é a potência calculada para o amplificador obtido pela equação (13) e *Pot* é a potência para referência proposta pelo usuário.

3.10 Função de Avaliação para a Área (F_8)

Para a Área do AmpOp, utilizam-se os dados obtidos da função *AreaCirMea.m* descrita na seção 3.1. A função de avaliação referente ao ganho, F_8, é dada por:

$$F_8 = \frac{Area_{circuito}}{Area} \quad \textbf{(31)}$$

onde $Area_{circuito}$ é o valor da área do circuito a ser testado e *Area* é a área para referência proposta pelo usuário.

Foi decidido, tanto para esta função como para a função de avaliação da potência que seus valores nunca zerem. Dessa forma, caso as outras funções de avaliação atinjam zero, todas as outras especificações do usuário alcançadas, a plataforma continuará otimizando a potência e a área do circuito.

3.11 Função de Avaliação para o *offset* (F_9)

Para o *offset*, utilizam-se os dados obtidos nas simulações descritas na seção 3.2.1. A partir dos dados determina-se o pior *offset* (o máximo valor encontrado) entre os três encontrados. A função de avaliação F_9 referente ao *offset* é apresentada a seguir:

$$F_9 = \begin{cases} 0 & para \quad offset_{circuito} \leq offset \\ \dfrac{offset_{circuito} - offset}{offset_{circuito}} & para \quad offset_{circuito} > offset \end{cases} \quad \textbf{(32)}$$

onde $offset_{circuito}$ é a pior tensão de *offset* encontrada nas simulações e *offset* é a tensão de *offset* proposta pelo usuário.

3.12 Função de Avaliação para o ruído (F_{10})

Para o Ruído, utilizam-se dos dados obtidos nas simulações descritas na seção 3.2.3. A função de avaliação referente ao ruído, F_{10}, é dada por:

$$F_{10} = \sum_{i=1}^{2} F_{10,i} \quad \textbf{(33)}$$

A função F_{10} é a soma das funções $F_{10,1}$ e $F_{10,2}$ que levam em consideração cada um dos dois valores de ruídos que são mensurados durante a simulação. A relação que exprime $F_{10,i}$ é dada a seguir:

$$F_{10,i} = \begin{cases} 0 & para \quad ri_{circuito} \leq r \\ \dfrac{ri_{circuito} - r}{ri_{circuito}} & para \quad ri_{circuito} > r \end{cases} \quad \textbf{(34)}$$

onde *r* é o valor rms da tensão do ruído proposta pelo usuário e $ri_{circuito}$ é a *i*-ésima tensão rms do ruído de entrada, que é calculado da seguinte forma:

$$ri_{circuito} = \frac{ri_{saída}}{G_{circuito}} \qquad (i = 1,2) \tag{35}$$

onde $ri_{saída}$ é a *i*-ésima tensão rms do ruído mensurado na saída do circuito (para Vcmín e Vcmáx).

3.13 Função de Avaliação para a fraca inversão (F_{11})

Por fim, para a Fraca Inversão, utiliza-se, caso o usuário tenha desejado que algum transistor trabalhe em fraca inversão, do resultado da equação (14). A função de avaliação referente à fraca inversão, F_{11}, é dada por:

$$F_{11} = \sum_{i=1}^{n} F_{11,i} \tag{36}$$

A função F_{11} é a soma de todas as funções que vão de $F_{11,1}$ até $F_{11,n}$ que levam em consideração cada um dos *n* transistores que se deseja que trabalhem em fraca inversão. A relação que exprime $F_{11,i}$ é dada a seguir:

$$F_{11,i} = \begin{cases} 0 & para \quad K_{OPi} \leq 0,12 \\ 10K_{OPi} & para \quad K_{OPi} > 0,12 \end{cases} \tag{37}$$

onde K_{OPi} é fator de inversão medido do *i*-ésimo transistor.

Se o valor de *n* for igual a zero, o valor de F_{11} também será zero.

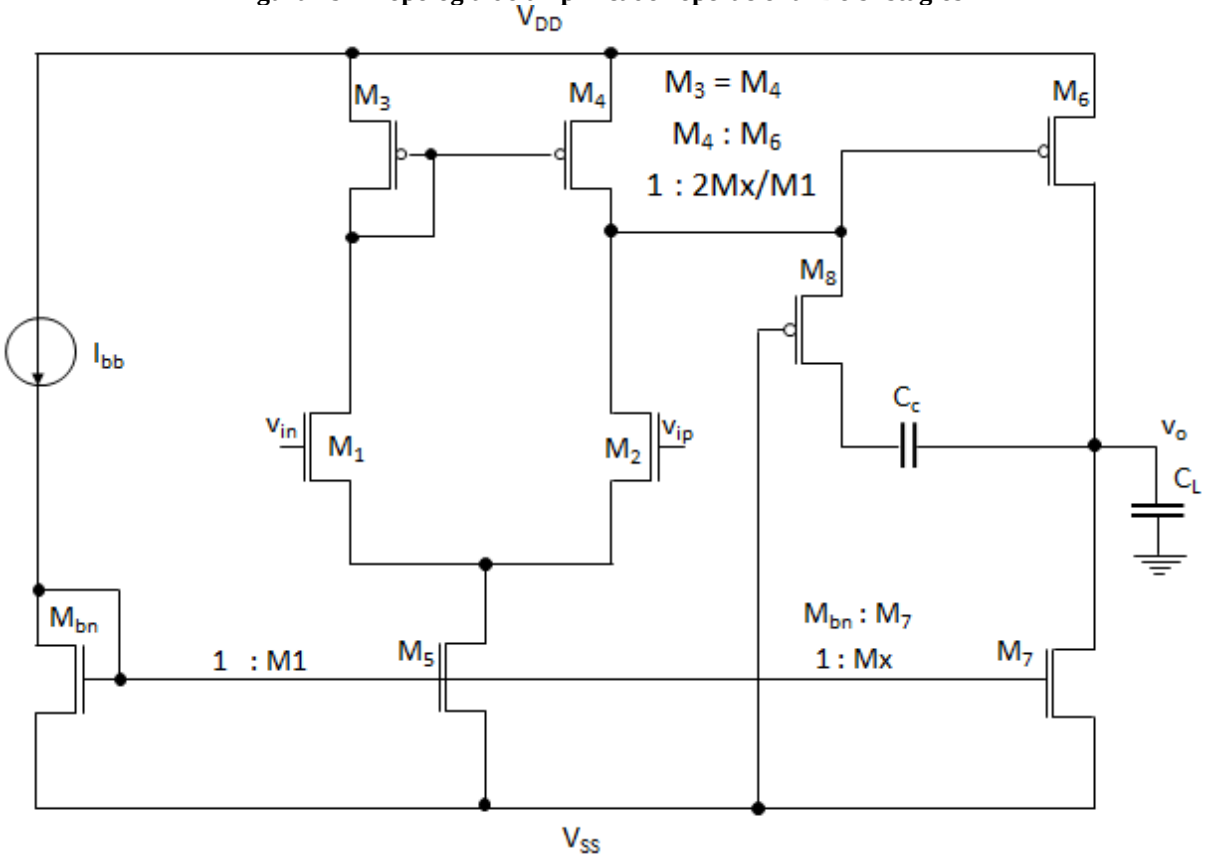
3.14 Amplificadores Operacionais Projetados

Nesse trabalho, foram projetados quatro amplificadores operacionais diferentes. Para uso interno desse documento, eles foram batizados de amplificador de “DoisEstagios”, “ClassAB”, “OTA” e “FoldedCascode”.

3.14.1 Amplificador DoisEstagios

Esta topologia é caracterizada pela presença de cinco transistores NMOS e quatro transistores PMOS. Na Figura 29 é representado o esquema do circuito. V_{in} e V_{ip} são os sinais de entrada e V_O é a saída. Na saída do circuito está a carga C_L.

Figura 29 – Topologia do amplificador operacional DoisEstagios



Esse circuito apresenta uma fonte de corrente I_{bb} que alimenta o espelho de corrente formado pelos transistores M_{bn}, M_5 e M_7 e que polarizam o restante do circuito. Os transistores M_1 e M_2 são iguais e formam o par diferencial da entrada do amplificador operacional. Os transistores M_3 e M_4 também são iguais e formam o espelho de corrente que funciona como carga ativa do par diferencial. O par diferencial amplifica a diferença ($V_{in} - V_{ip}$) de tensão e converte para uma tensão de saída única. Essa tensão é aplicada ao *gate* do transistor M_6, que opera na configuração *source* comum e fornece mais ganho ao sinal. O capacitor C_C junto com o transistor M_8, que opera como um resistor, têm a função de criar e posicionar polos e zeros de tal forma a dar estabilidade ao amplificador.

Alguns transistores são descritos, nos arquivos de simulação, com largura e comprimento de canal obedecendo a certas relações, o que garante a geração de circuitos que podem ser bem construídos. Assim os transistores M_{bn}, M_5 e M_7 são descritos de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem as seguintes relações:

$$\frac{W_5}{W_{bn}} = M1 \quad \text{e} \quad \frac{W_7}{W_{bn}} = Mx \tag{38}$$

onde W_i é a largura de canal transistor *i* e M1 e Mx são constantes inteiras.

Dessa forma se garante, com um *layout* adequado, uma polarização confiável para o amplificador. O valor de M1 é atribuído pelo usuário e normalmente tem um valor alto.

Os transistores M_3, M_4 e M_6 são descritos, em arquivos para simulação, de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem as relações:

$$W_3 = W_4 \quad \text{e} \quad \frac{W_6}{W_4} = \frac{2Mx}{M1} \tag{39}$$

Dessa forma, quando as tensões de entrada são iguais e, em consequência, as tensões de *gate* dos transistores M_3, M_4 e M_6 também são iguais, se garante que as correntes de dreno dos transistores M_6 e M_7 são as mesmas. Isso evita problemas de *offset* sistemático.

Conforme é apresentado na Tabela 1, um indivíduo dessa topologia será representado por onze variáveis de projeto, a serem otimizadas pela plataforma.

Tabela 1 – Variáveis de projeto da topologia DoisEstágios

Variáveis	Parâmetros
X_1	L_1, L_2 (μm)
X_2	L_3, L_4, L_6 (μm)
X_3	L_{bn}, L_5, L_7 (μm)
X_4	L_8 (μm)
X_5	W_1, W_2 (μm)
X_6	W_3, W_4, M1*W_6/(2Mx) (μm)
X_7	W_{bn}, W_5/M1, W_7/Mx (μm)
X_8	Mx
X_9	W_8 (μm)
X_{10}	C_C:C_L
X_{11}	$\log_{10}(I_{bb})$

Além disso, o usuário define duas constantes, exibidas na Tabela 2:

Tabela 2 – Constantes da topologia DoisEstágios

Constantes	Parâmetros
M1	W_5/W_{bn}
M2	C_L (pF)

Algumas observações sobre as variáveis da tabela são:

- X_{10} é a razão entre os valores dos capacitores C_C e C_L;
- X_{11} é o logaritmo na base 10 do valor da fonte de corrente, $\log_{10}(I_{bb})$, sendo a corrente dada em microAmpere. Dessa maneira, ao contrário das outras variáveis onde é realizada uma exploração linear pelo espaço busca, no variável associada à fonte de corrente I_{bb} é realiza

uma exploração exponencial. Com isso, valores menores de corrente são mais bem explorados e circuitos com menor consumo podem ser encontrados;

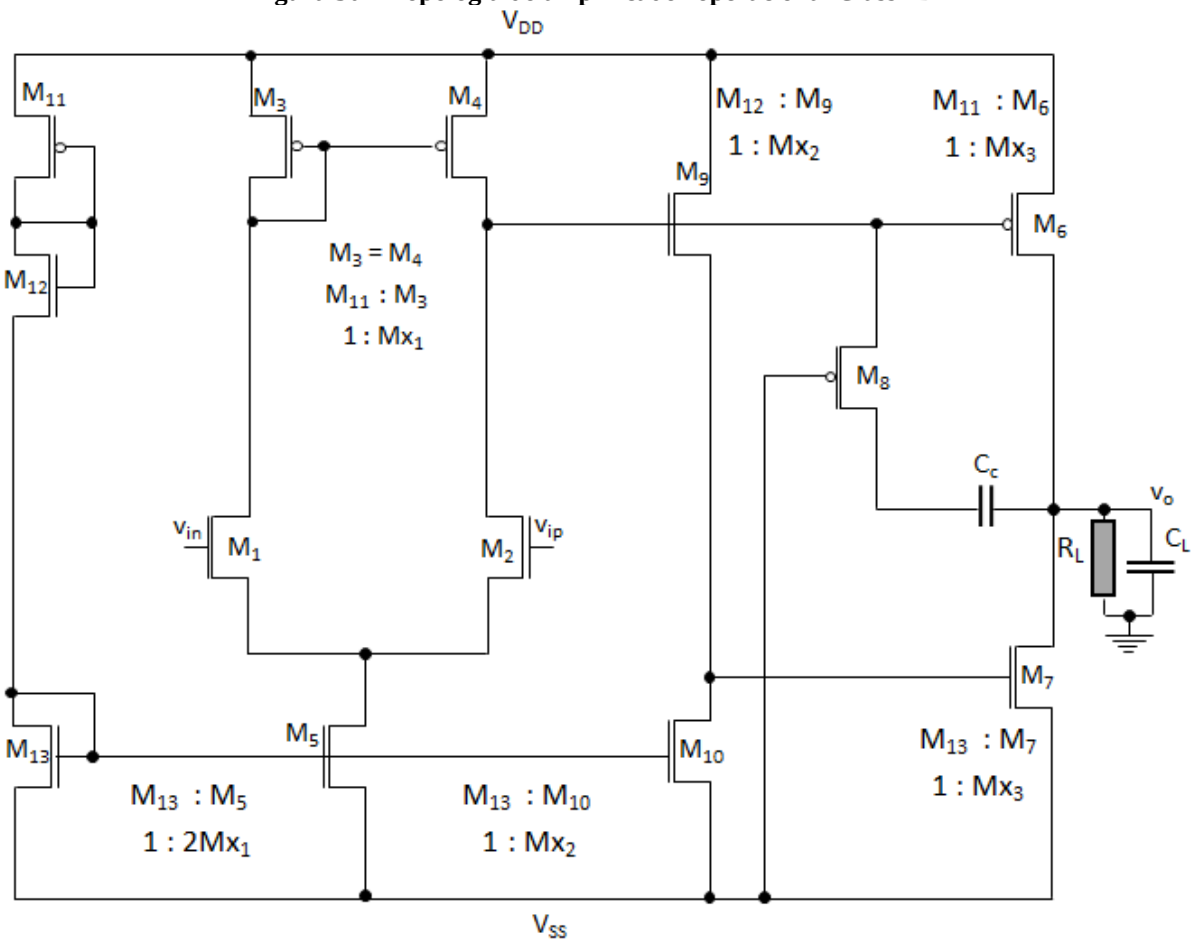
- Várias restrições foram impostas nos parâmetros, por exemplo, fazendo $L_1 = L_2$, o que preveni que indivíduos com características inadequadas sejam gerados e analisados.

A descrição do amplificador operacional da topologia DoisEstagios em linguagem SPICE encontra-se no APÊNDICE C.

3.14.2 Amplificador ClassAB

Esta topologia é um melhoramento do amplificador de DoisEstagios, pois consegue suportar não apenas cargas capacitivas, mas também cargas resistivas. Ela é caracterizada pela presença de oito transistores NMOS e quatro transistores PMOS. Na Figura 30 é representado o esquema do circuito. V_{in} e V_{ip} são os sinais de entrada e V_O é a saída. Na saída do circuito estão as cargas C_L e R_L.

Figura 30 – Topologia do amplificador operacional ClassAB



Esse circuito apresenta uma fonte de corrente, formado pelos transistores M_{11}, M_{12} e M_{13}, que alimenta o espelho de corrente formado pelos transistores M_{13}, M_5 e M_{10} e que polarizam o restante do circuito. Os transistores M_1 e M_2 são iguais e formam o par diferencial da entrada do amplificador

operacional. Os transistores M_3 e M_4 também são iguais e formam o espelho de corrente que funciona como carga ativa do par diferencial. O par diferencial amplifica a diferença ($V_{in} - V_{ip}$) de tensão e converte para uma tensão de saída única. Essa tensão é aplicada ao *gate* do transistor M_9, que opera na configuração *dreno* comum e desloca o sinal. O par de transistores M_6 e M_7 funcionam como um amplificador *push-pull* fornecendo corrente para a carga resistiva se necessário. O capacitor C_C junto com o transistor M_8, que opera como um resistor, têm a função de criar e posicionar polos e zeros de tal forma a dar estabilidade ao amplificador.

Alguns transistores são descritos, nos arquivos de simulação, com largura e comprimento de canal obedecendo a certas relações, o que garante a geração de circuitos que podem de ser bem construídos. Assim os transistores M_{13}, M_5 e M_{10} são descritos de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem as relações:

$$\frac{W_5}{W_{13}} = 2Mx_1 \quad \text{e} \quad \frac{W_{10}}{W_{13}} = Mx_2 \tag{40}$$

onde Mx_1 e Mx_2 são constantes inteiras.

Dessa forma se garante, com um *layout* adequado, uma polarização confiável.

Os transistores M_3, M_4, M_6 e M_{11} são descritos de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem as relações:

$$W_3 = W_4; \frac{W_3}{W_{11}} = Mx_1 \text{ e } \frac{W_6}{W_{11}} = Mx_3 \tag{41}$$

onde Mx_3 é uma constante inteira.

Dessa forma, quando as tensões de entrada são iguais e, em consequência, as tensões de *gate* dos transistores M_3, M_4 e M_6 as mesmas, garante-se que a corrente que passa por M_6 estará bem determinada pela polarização.

Os transistores M_{12} e M_9 são descritos tendo o mesmo comprimento de canal. As larguras de canal desses transistores obedecem a seguinte relação:

$$\frac{W_9}{W_{12}} = Mx_2 \tag{42}$$

Dessa forma se garante que a tensão de *gate* de M_7 é igual à tensão de *gate* de M_{13}. Por fim, os transistores M_7 e M_{13} são descritos tendo o mesmo comprimento de canal e larguras de canal obedecendo a seguinte relação:

$$\frac{W_7}{W_{13}} = Mx_3 \quad \textbf{(43)}$$

Assim, se garante que as correntes de dreno dos transistores M_6 e M_7 são iguais para entradas iguais aplicadas. Isso evita problemas de *offset* sistemático.

Conforme é apresentado na Tabela 3, um indivíduo dessa topologia será representado por quatorze variáveis de projeto, a serem otimizados pela plataforma.

Tabela 3 – Variáveis de projeto da topologia ClassAB

Variáveis	Parâmetros
X_1	L_1, L_2 (μm)
X_2	L_3, L_4, L_6 e L_{11} (μm)
X_3	L_5, L_7, L_{10} e L_{13} (μm)
X_4	L_8 (μm)
X_5	L_9 e L_{12} (μm)
X_6	W_1 e W_2 (μm)
X_7	W_3/Mx_1, W_4/Mx_1, W_6/Mx_3 e W_{11} (μm)
X_8	$W_5/2Mx_1$, W_7/Mx_3, W_{10}/Mx_2 e W_{13} (μm)
X_9	W_8 (μm)
X_{10}	W_9/Mx_2 e W_{12} (μm)
X_{11}	Mx_1
X_{12}	Mx_2
X_{13}	Mx_3
X_{14}	C_C:C_L

Além disso, o usuário define duas constantes, exibidas na Tabela 4.

Tabela 4 – Constantes da topologia ClassAB

Constantes	Parâmetros
M1	C_L(pF)
M2	R_L(kΩ)

Algumas observações sobre as variáveis da tabela são:

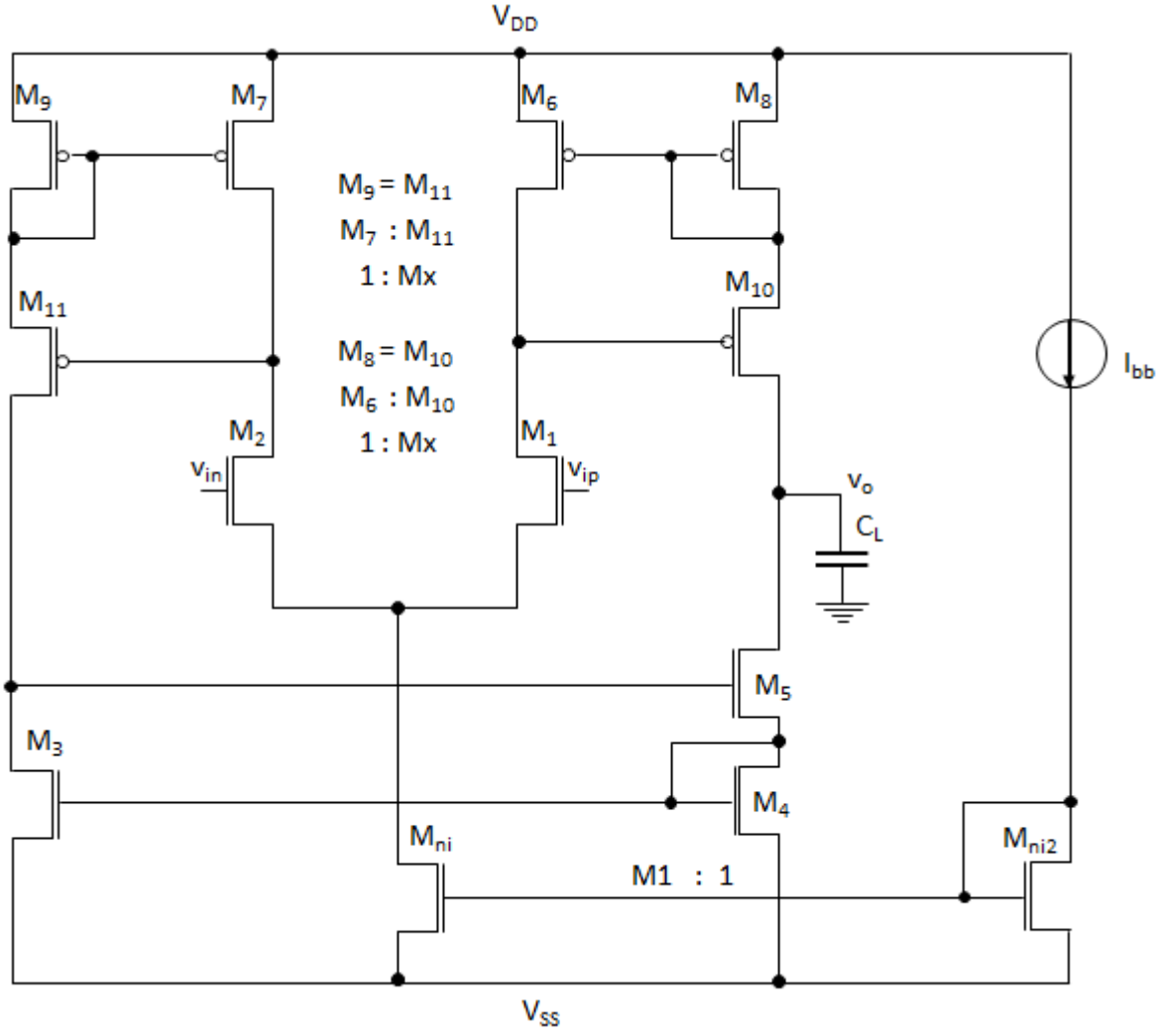
- X_{14} é a razão entre os valores dos capacitores C_C e C_L
- Várias restrições foram impostas nos parâmetros, o que preveni que indivíduos com características inadequadas sejam gerados e analisados.

A descrição do amplificador operacional da topologia ClassAB em linguagem SPICE encontra-se no APÊNDICE C.

3.14.3 Amplificador OTA

Esta é a topologia é caracterizada pela presença de sete transistores NMOS e seis transistores PMOS. Na Figura 31 é representado o esquema do circuito. V_{in} e V_{ip} são os sinais de entrada e V_O é a saída. Na saída do circuito está a carga C_L.

Figura 31 – Topologia do amplificador operacional OTA



Esse circuito apresenta uma fonte de corrente I_{bb} que alimenta o espelho de corrente formado pelos transistores M_{ni} e M_{ni2} e que polarizam o restante do circuito. Os transistores M_1 e M_2 são iguais e formam o par diferencial da entrada do amplificador operacional. O par diferencial converte as variações de tensão em V_{in} e V_{ip} para variações de corrente. A variação de corrente devido a V_{ip} é espelhada para saída pelo espelho de Wilson formado pelos transistores M_6, M_8 e M_{10}, causando uma variação na corrente de dreno de M_8. A variação de corrente devido a V_{in}, por sua vez, é espelhada pelo espelho de Wilson formado pelos transistores M_7, M_9 e M_{11}; a corrente resultante é espelhada

para saída pelo espelho formado pelos transistores M_3, M_4 e M_5, causando uma variação na corrente de dreno de M_4. Na saída, as variações nas correntes de dreno provocam uma grande variação na tensão.

Alguns transistores são descritos, nos arquivos de simulação, com largura e comprimento de canal obedecendo a certas relações, o que garante a geração de circuitos que podem de ser bem construídos. Assim os transistores M_{ni} e M_{ni2} são descritos de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem a seguinte relação:

$$\frac{W_{ni}}{W_{ni2}} = M1 \tag{44}$$

onde M1 é uma constante inteira.

Dessa forma se garante, com um *layout* adequado, uma polarização confiável para o amplificador. O valor de M1 é atribuído pelo usuário e normalmente tem um valor alto.

Os transistores M_6, M_8 e M_{10} e os transistores M_7, M_9 e M_{11} são descritos, em arquivos para simulação, de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem as relações:

$$W_8 = W_{10} = W_9 = W_{11} \quad \text{e} \quad \frac{W_{10}}{W_6} = \frac{W_{11}}{W_7} = Mx \tag{45}$$

onde Mx é uma constantes inteira.

Dessa forma se garante que as correntes espelhadas para M_8 e M_9 são simétricas.

Por fim, os transistores M_3, M_4 e M_5 são descritos tendo o mesmo comprimento e largura de canal. Dessa forma, para entradas iguais de tensão, as correntes de dreno em M_8 e M_4 serão iguais e não haverá *offset* sistemático.

Conforme é apresentado na Tabela 5, um indivíduo dessa topologia será representado por dez variáveis de projeto, a serem otimizadas pela plataforma.

Tabela 5 – Variáveis de projeto da topologia OTA

Variáveis	Parâmetros
X_1	L_1, L_2 (μm)
X_2	L_3, L_4, L_5 (μm)
X_3	L_{ni} e L_{ni2} (μm)
X_4	L_6, L_7, L_8, L_9, L_{10} e L_{11} (μm)
X_5	W_1 e W_2 (μm)
X_6	W_3, W_4, W_5 (μm)
X_7	W_{ni}/M1 e W_{ni2} (μm)
X_8	W_6, W_7, W_8/Mx, W_9/Mx, W_{10}/Mx e W_{11}/Mx (μm)
X_9	$\log_{10}(I_{bb})$
X_{10}	Mx

Além disso, o usuário define duas constantes, exibidas na Tabela 6:

Tabela 6 – Constantes da topologia OTA

Constantes	Parâmetros
M1	W_{ni}/W_{ni2}
M2	C_L (pF)

Algumas observações sobre as variáveis da tabela são:

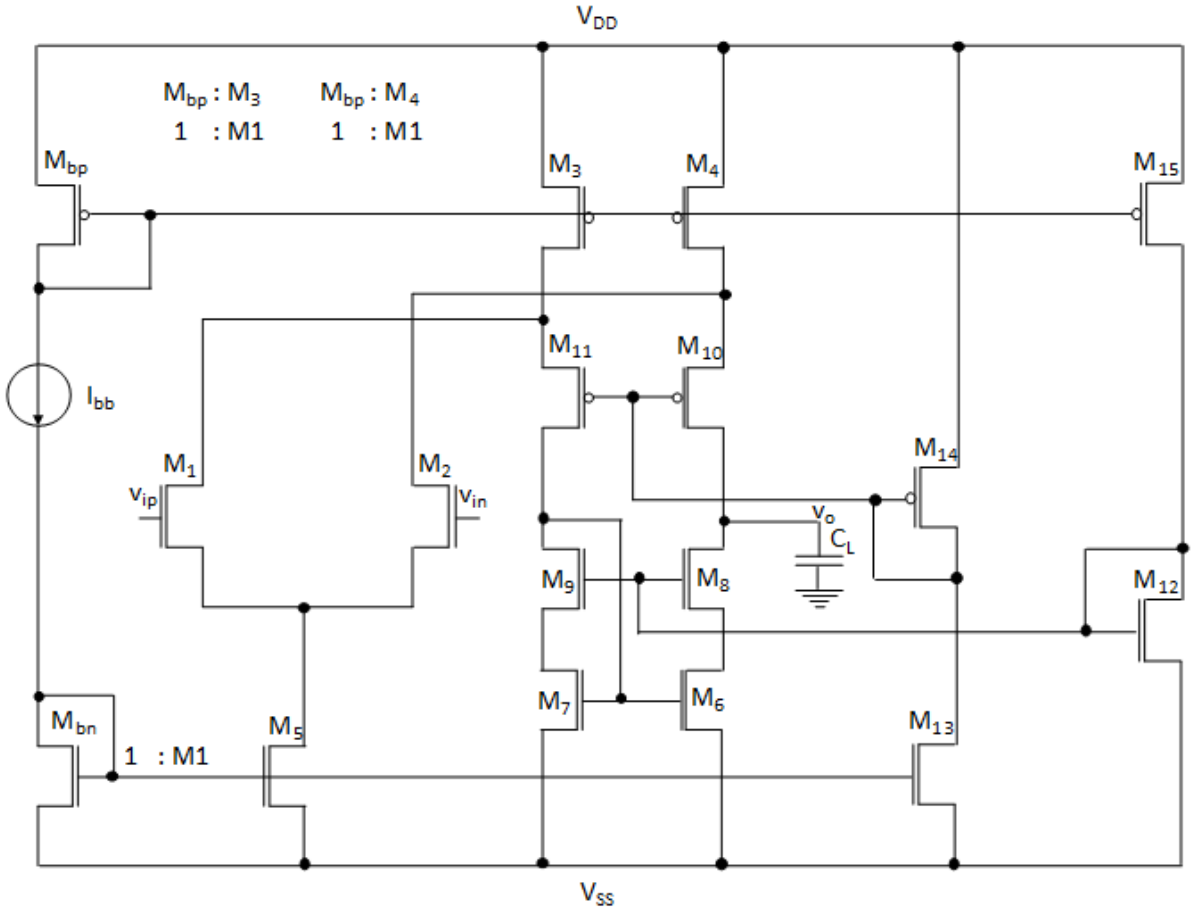
- X_9 é o logaritmo na base 10 do valor da fonte de corrente, $\log_{10}(I_{bb})$, sendo a corrente dada em microampère. Com isso, valores menores de corrente são mais bem explorados e circuitos com menor consumo podem ser encontradas.
- Várias restrições foram impostas nos parâmetros, o que preveni que indivíduos com características inadequadas sejam gerados e analisados.

A descrição do amplificador operacional da topologia OTA em linguagem SPICE encontra-se no APÊNDICE C.

3.14.4 Amplificador FoldedCascode

Esta topologia é caracterizada pela presença de dez transistores NMOS e sete transistores PMOS. Na Figura 32 é representado o esquema do circuito. V_{in} e V_{ip} são os sinais de entrada e V_O é a saída. Na saída do circuito está a carga C_L.

Figura 32 – Topologia do amplificador operacional FoldedCascode



Esse circuito apresenta uma fonte de corrente I_{bb} que alimenta o espelho de corrente formado pelos transistores M_{bn}, M_5 e M_{13} que polarizam parte do circuito. Essa fonte de corrente I_{bb} ainda alimenta o espelho de corrente formado pelos transistores M_{bp}, M_3, M_4 e M_{15} que polarizam o restante do circuito. Os transistores M_1 e M_2 são iguais e formam o par diferencial da entrada do amplificador operacional. Os transistores M_6 e M_7, iguais entre si, junto com os transistores M_8 e M_9, também iguais entre si, formam a carga ativa para o par diferencial. O par diferencial amplifica a diferença ($V_{in} – V_{ip}$) de tensão e converte para uma tensão de saída única V_O. Os transistores M_{10} e M_{11} são iguais e formam um par *cascode* responsáveis por aumentar o ganho do amplificador e isolar a entrada da saída do circuito. Os transistores M_3 e M_4 formam um espelho de corrente que polariza o par *cascode* e a carga ativa do par diferencial.

O transistor M_{13} polariza o transistor M_{14} responsável por fornecer uma tensão de polarização adequada para o par *cascode*; o transistor M_{15} polariza o transistor M_{12} responsável por fornecer uma tensão de polarização adequada para os transistores M_8 e M_9.

Alguns transistores são descritos, nos arquivos de simulação, com largura e comprimento de canal obedecendo a certas relações, o que garante a geração de circuitos que podem de ser bem construídos. Assim os transistores M_{bn}, M_5 e M_{13} são descritos de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem as relações:

$$W_{bn} = W_{13} \quad \text{e} \quad \frac{W_5}{W_{bn}} = M1 \tag{46}$$

onde M1 é uma constante inteira.

Dessa forma se garante, com um *layout* adequado, uma polarização confiável para o amplificador. O valor de M1 é atribuído pelo usuário e normalmente tem um valor alto.

Os transistores M_{bp}, M_3, M_4 e M_{15} são descritos de tal forma a possuírem o mesmo comprimento de canal e larguras de canal que obedecem as relações:

$$W_{bp} = W_{15};\ W_3 = W_4 \text{ e } \frac{W_3}{W_{bp}} = M1 \tag{47}$$

Dessa forma se garante, com um *layout* adequado, uma polarização confiável para o amplificador.

Conforme é apresentado na Tabela 7, um indivíduo dessa topologia será representado por dezessete variáveis de projeto, a serem otimizadas pela plataforma.

Tabela 7 – Parâmetros de otimização da topologia FoldedCascode

Variáveis	Parâmetros
X_1	L_1, L_2 (μm)
X_2	L_{bp}, L_3, L_4, e L_{15} (μm)
X_3	L_{bn}, L_5, L_{13} (μm)
X_4	L_6 e L_7 (μm)
X_5	L_8 e L_9 (μm)
X_6	L_{10} e L_{11} (μm)
X_7	W_1, W_2 (μm)
X_8	W_{bp}, W_3/M1, W_4/M1 e W_{15} (μm)
X_9	W_{bn}, W_5/M1 e W_{13} (μm)
X_{10}	W_6 e W_7 (μm)
X_{11}	W_8 e W_9 (μm)
X_{12}	W_{10} e W_{11} (μm)
X_{13}	L_{12} (μm)
X_{14}	W_{12} (μm)
X_{15}	L_{14} (μm)
X_{16}	W_{14} (μm)
X_{17}	$\log_{10}(I_{bb})$

Além disso, o usuário define duas constantes, exibidas na Tabela 8:

Tabela 8 – Constantes da topologia FoldedCascode

Constantes	Parâmetros
M1	W_5/W_{bn} e W_3/W_{bp}
M2	C_L (pF)

Algumas observações sobre as variáveis da tabela são:

- X_{17} é o logaritmo na base 10 do valor da fonte de corrente, $\log_{10}(I_{bb})$, sendo a corrente dada em microampère. Com isso, valores menores de corrente são mais bem explorados e circuitos com menor consumo podem ser encontradas;
- Várias restrições foram impostas nos parâmetros, o que preveni que indivíduos com características inadequadas sejam gerados e analisados;

A descrição do amplificador operacional da topologia FoldedCascode em linguagem SPICE encontra-se no APÊNDICE C.

4 RESULTADOS

Neste capítulo serão apresentados os resultados das otimizações realizadas com os quatro amplificadores operacionais, utilizando-se das três metaheurísticas apresentadas nesse texto e do simulador elétrico Eldo, bem como a comparação entre os métodos.

Todas as otimizações foram realizadas dez vezes, cada uma delas com, no máximo, 3000 indivíduos estudados e sempre partindo de pontos aleatórios do espaço de busca.

Os pesos das funções de avaliação são constantes para todas as otimizações e são exibidas na Tabela 9.

Tabela 9 – Pesos das funções de avaliação utilizados para todas as topologias

Critério de desempenho	**Ganho**	**Frequência de ganho unitário**	**CMRR**	**PSRR**	**Margem de fase**	***Slew Rate***
Peso	10	10	10	10	100	10
Critério de desempenho	**Potência**	**Área**	***Offset***	**Ruído**	**Fraca inversão**	
Peso	1,0	0,1	10	100	0,0	

Para o algoritmo genético, a estratégia aplicada foi a utilização de duas populações, cada uma com 50 indivíduos. Como, no máximo, são avaliados três mil indivíduos, o número de gerações máximo, para cada uma das dez otimizações, é 29.

Para o enxame de partículas, o número de partículas presentes é igual ao número de variáveis de otimização.

Para o recozimento simulado, a temperatura inicial é igual 2,0 e a temperatura mínima é igual 0,001, com um reaquecimento do sistema ocorrendo quando forem encontrados 50 indivíduos melhores ou quando se atingir a temperatura mínima.

4.1 Otimização do amplificador de DoisEstagios

Para o amplificador com a topologia DoisEstagios, da Figura 29, as especificações passadas para a plataforma foram as seguintes:

- Tensões de alimentação: 3,0 V e 0,0 V;
- Ganho de malha aberta mínima: 80 dB;
- Frequência de ganho unitário mínima: 10 MHz;
- *Slew Rate* mínimo: 7,0 V/μs;
- CMRR mínimo: 60 dB;
- PSRR mínimo: 60 dB;
- Limites de margem de fase: 45º e 60º;

- Faixa de tensões de modo comum: 1,2 V e 1,8 V;
- A banda de frequência para o CMRR: 1,0 Hz a 100,0 Hz;
- O ruído de entrada máximo: 10,0 μV;
- A tensão de *offset* máxima: 200 μV;
- A área do circuito máxima: 50000 μm²;
- A potência máxima: 50,0 μW;
- Duas constantes: $W_5/W_{bn} = 10$ e $C_L = 1{,}0$ pF;
- Nenhum transistor necessita atuar em fraca inversão;

Na Tabela 10 são apresentados os valores dos melhores candidatos obtidos com o auxilio do algoritmo genético, enxame de partículas e recozimento simulado juntamente com os limites mínimo e máximo das variáveis de otimização escolhidos:

Tabela 10 – Valores das variáveis obtidas na otimização com algoritmo genético, enxame de partículas e recozimento simulado da topologia DoisEstagios

Parâmetro		**X_1 (μm)**	**X_2 (μm)**	**X_3 (μm)**	**X_4 (μm)**	**X_5 (μm)**	**X_6 (μm)**
Valor mínimo		1,0	1,0	1,0	1,0	1,0	1,0
Valor obtido	**GA**	6,26	1,24	18,77	2,4	46,19	31,38
	PSO	5,07	1,01	18,43	6,04	11,29	3,37
	SA	8,90	1,67	18,00	1,73	69,03	11,57
Valor máximo		10	10	20	20	200	200

Parâmetro		**X_7 (μm)**	**X_8 (μm)**	**X_9 (μm)**	**X_{10}**	**X_{11}**	
Valor mínimo		1,0	1,0	1,0	0,3	-1,0	
Valor obtido	**GA**	2,04	13,12	5,55	0,94	0,08	
	PSO	1,90	18,28	2,53	0,30	-0,27	
	SA	3,43	15,39	31,62	0,30	-0,14	
Valor máximo		200	20	100	3	0,5	

Na Tabela 11, são mostrados os valores dos parâmetros de desempenho dos melhores circuitos encontrados nas otimizações realizadas pelo algoritmo genético, enxame de partículas e recozimento simulado, bem como os *scores* alcançados por cada um dos algoritmos.

Tabela 11 – Valores dos parâmetros de desempenho e de *score* obtidos da otimização com algoritmo genético, enxame de partículas e recozimento simulado da topologia DoisEstagios

Critério de desempenho		Ganho (dB)	Frequência de ganho unitário (MHz)	CMRR (dB)	PSRR (dB)	Margem de fase	*Score*
Valor obtido	GA	101	13,88	86	100	59,4º	3,0
	PSO	92	17	75	90	56,6º	0,88
	SA	92	15,5	78	91	49º	1,3

Critério de desempenho		*Slew Rate* (V/µs)	Potência (µW)	Área (µm²)	*Offset* na entrada (µV)	Ruído na entrada (µVrms)	
Valor obtido	GA	7,4	86	2200	5,6	1,5	
	PSO	7,0	47	13000	12	4,7	
	SA	7,0	56	7100	9,4	3,7	

Na Figura 33, Figura 34 e Figura 35, são apresentados, respectivamente, o diagrama de Bode da resposta em malha aberta do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 33 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia DoisEstagios, obtido a partir da aplicação do algoritmo genético

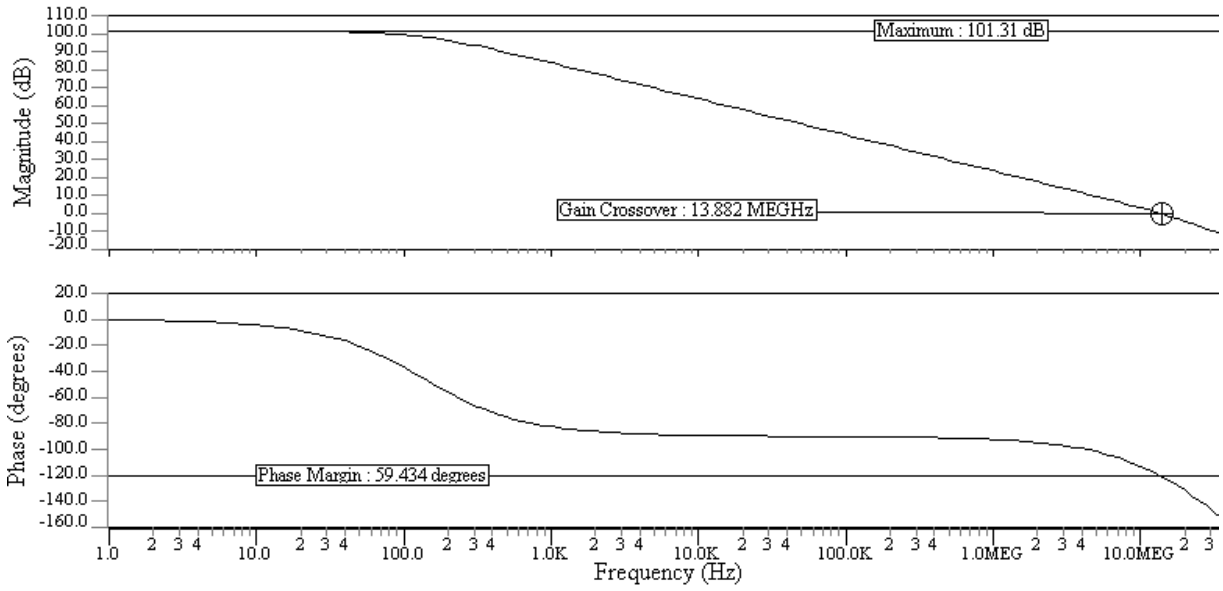


Figura 34 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia DoisEstagios, obtido a partir da aplicação do enxame de partículas

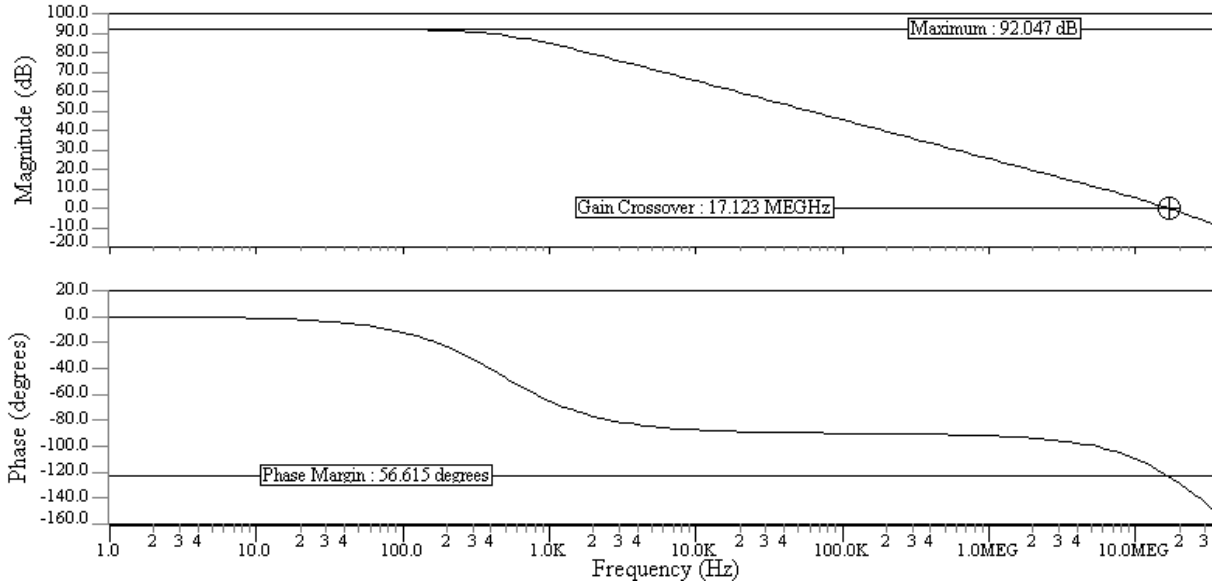
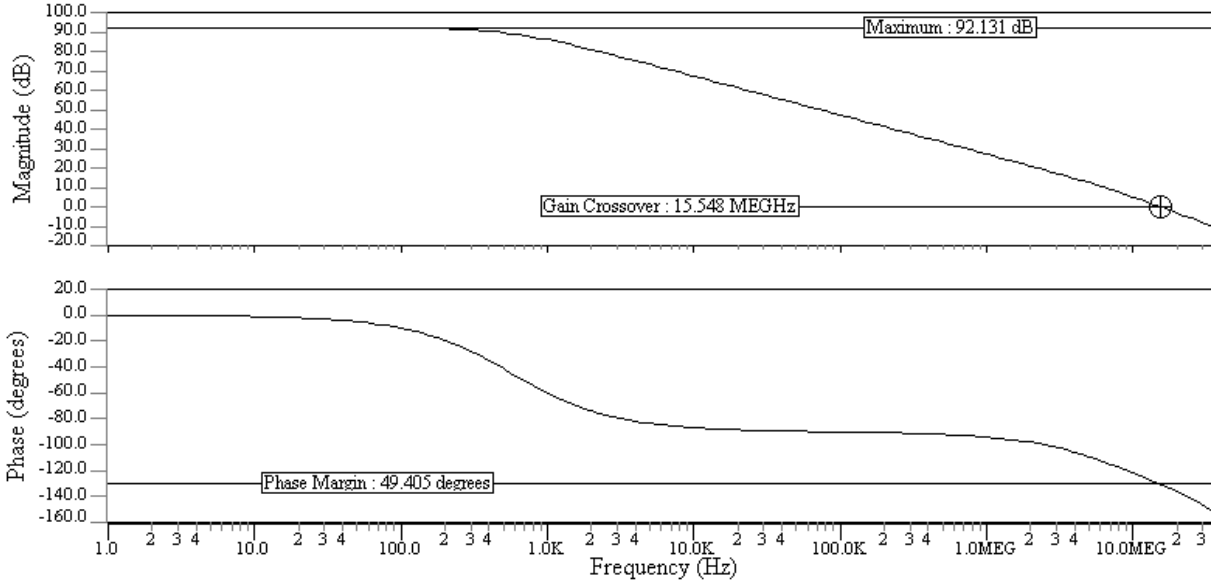


Figura 35 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia DoisEstagios, obtido a partir da aplicação do recozimento simulado



Na Figura 36, Figura 37 e Figura 38 são apresentados, respectivamente, a resposta ao degrau do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 36 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia DoisEstagios, obtido a partir da aplicação do algoritmo genético

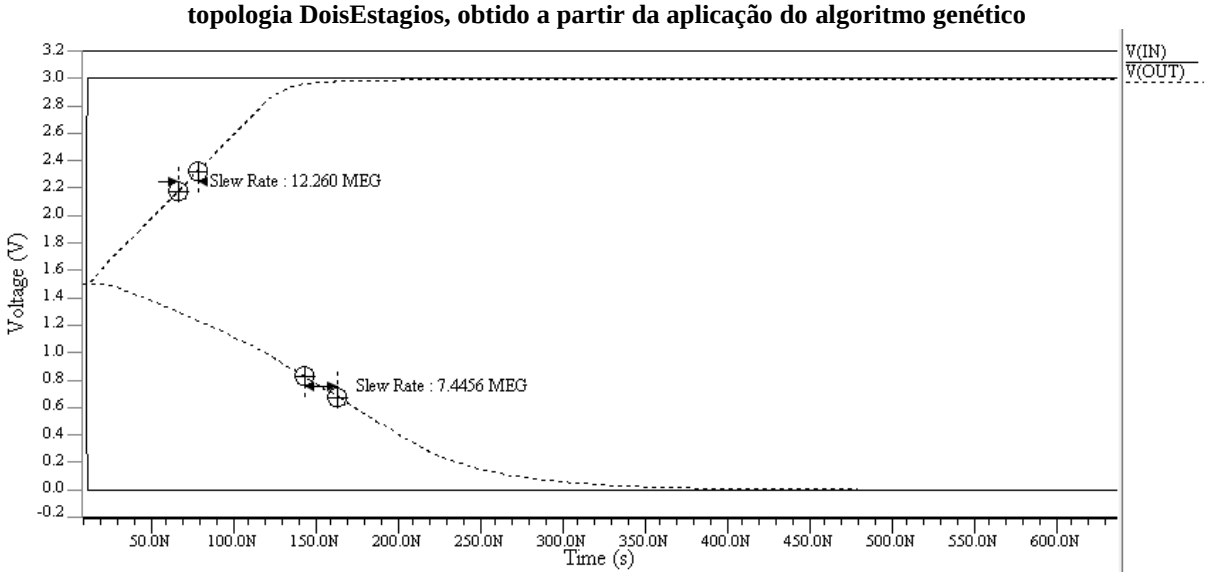


Figura 37 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia DoisEstagios, obtida a partir do enxame de partículas

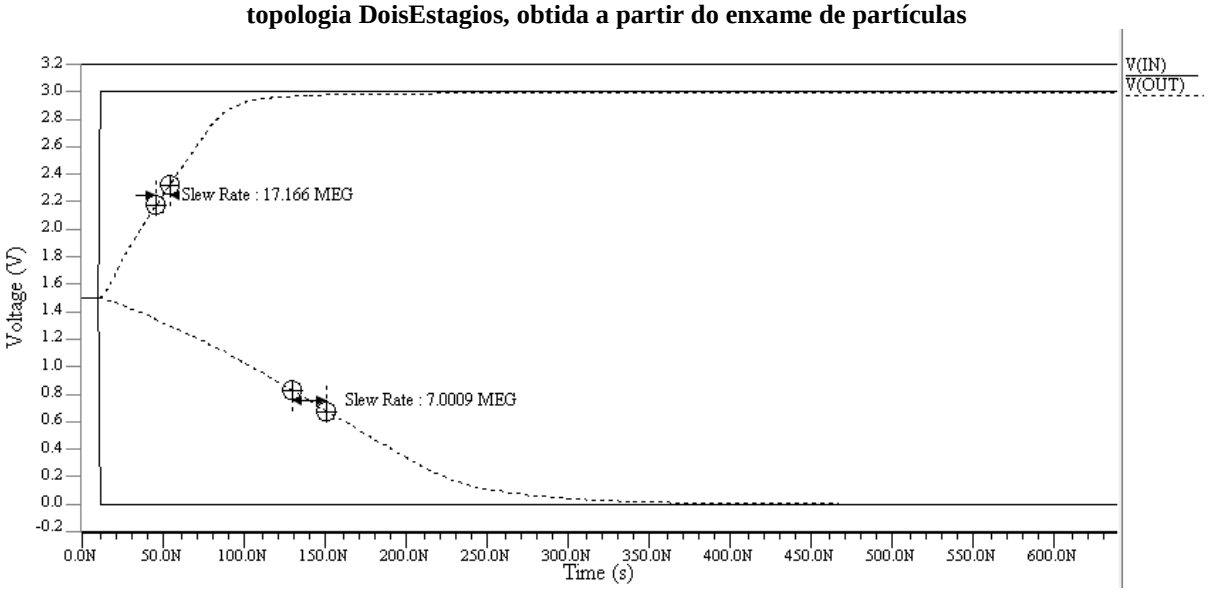
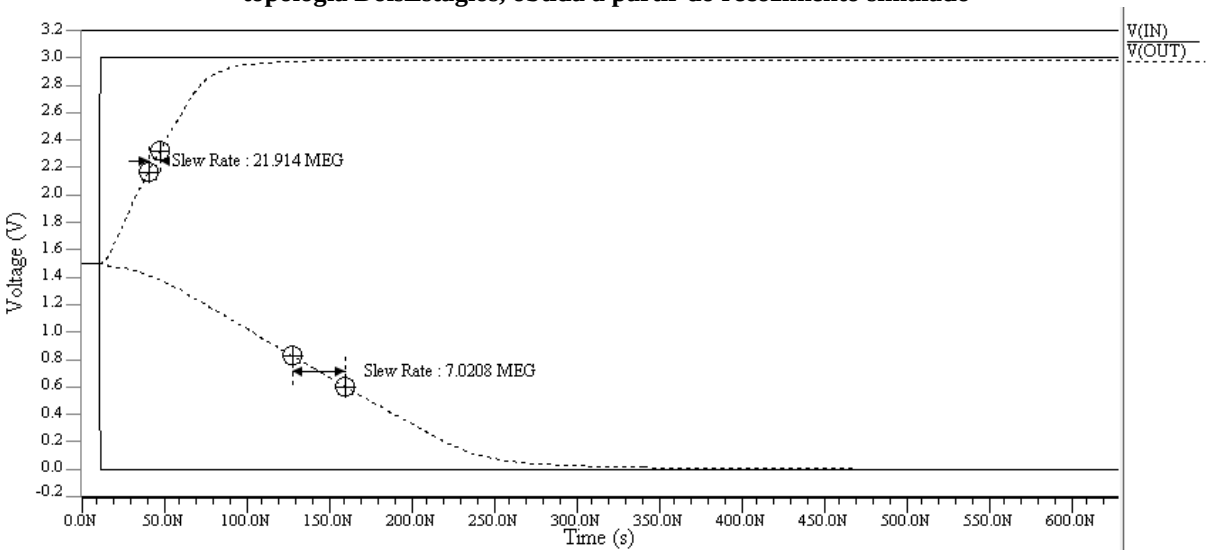
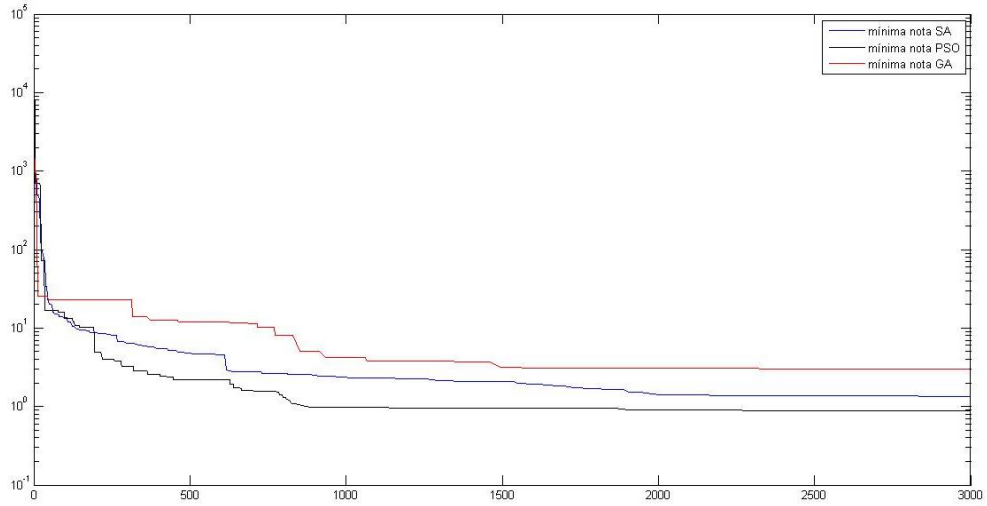


Figura 38 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia DoisEstagios, obtida a partir do recozimento simulado



Na Figura 39 é apresentado o comportamento dos três algoritmos utilizados nessas otimizações, exibindo o gráfico mínimo *score* alcançado, entre as dez otimizações, versus a iteração.

Figura 39 – Evolução dos algoritmos metaheurísticos para a topologia DoisEstagios: gráfico do mínimo *score* alcançado, entre as dez otimizações, versus a iteração para os três algoritmos de otimização



4.2 Otimização do amplificador de ClassAB

Para o amplificador com a topologia ClassAB, da Figura 30, as especificações passadas para a plataforma foram as seguintes:

- Tensões de alimentação: 3,0 V e 0,0 V;
- Ganho de malha aberta mínima: 70 dB;
- Frequência de ganho unitário mínima: 10 MHz;
- *Slew Rate* mínimo: 7,0 V/μs;

- CMRR mínimo: 60 dB;
- PSRR mínimo: 60 dB;
- Limites de margem de fase: 45º e 60º;
- Faixa de tensões de modo comum: 1,2 V e 1,8 V;
- A banda de frequência para o CMRR: 1,0 Hz a 100,0 Hz;
- O ruído de entrada máximo: 10,0 μV;
- A tensão de *offset* máxima: 500 μV;
- A área do circuito máxima: 50000 μm²;
- A potência máxima: 2000 μW;
- Duas constantes: R_L = 1,0 kΩ e C_L = 1,0 pF;
- Nenhum transistor necessita atuar em fraca inversão;

Na Tabela 12 são apresentados os valores dos melhores candidatos obtidos com o auxilio do algoritmo genético, enxame de partículas e recozimento simulado juntamente com os limites mínimo e máximo das variáveis de otimização escolhidos:

Tabela 12 – Valores das variáveis obtidas na otimização com algoritmo genético, enxame de partículas e recozimento simulado da topologia ClassAB

Parâmetro		**X_1 (μm)**	**X_2 (μm)**	**X_3 (μm)**	**X_4 (μm)**	**X_5 (μm)**	**X_6 (μm)**	**X_7 (μm)**
Valor mínimo		0,5	0,5	0,5	0,5	0,5	1,0	1,0
Valor obtido	**GA**	6,24	12,0	0,6	5,13	14,48	253,36	18,36
	PSO	12,98	5,91	0,5	17,77	19,66	476,36	5,07
	SA	12,4	4,59	0,5	2,02	12,62	361,78	4,51
Valor máximo		20	20	10	20	20	500	200
Parâmetro		**X_8 (μm)**	**X_9 (μm)**	**X_{10} (μm)**	**X_{11}**	**X_{12}**	**X_{13}**	**X_{14}**
Valor mínimo		1,0	1,0	1,0	2,0	2,0	2,0	0,3
Valor obtido	**GA**	156,14	91,07	30,07	2,98	2,84	16,1	1,15
	PSO	196,13	19	17,87	2,0	2,0	20	1,88
	SA	105,43	57,21	20,51	2,0	4,0	20	0,91
Valor máximo		200	100	100	10	10	20	3,0

Na Tabela 13, são mostrados os valores dos parâmetros de desempenho dos melhores circuitos encontrados nas otimizações realizadas pelo algoritmo genético, enxame de partículas e recozimento simulado, bem como o *scores* alcançados por cada um dos algoritmos.

Tabela 13 – Valores dos parâmetros de desempenho e de *score* obtidos da otimização com algoritmo genético, enxame de partículas e recozimento simulado da topologia ClassAB

Critério de desempenho		Ganho (dB)	Frequência de ganho unitário (MHz)	CMRR (dB)	PSRR (dB)	Margem de fase	*Score*
Valor obtido	GA	72	13	61	69	54º	0,77
	PSO	71	13	60	74	45º	0,27
	SA	70	19	60	73	51º	0,42

Critério de desempenho		*Slew Rate* (V/µs)	Potência (µW)	Área (µm²)	*Offset* na entrada (µV)	Ruído na entrada (µVrms)	
Valor obtido	GA	7,4	1700	22000	20	1,2	
	PSO	7,1	930	28000	34	1,1	
	SA	11	1200	19000	36	1,7	

Na Figura 40, Figura 41 e Figura 42, são apresentados, respectivamente, o diagrama de Bode da resposta em malha aberta do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 40 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia ClassAB, obtido a partir da aplicação do algoritmo genético

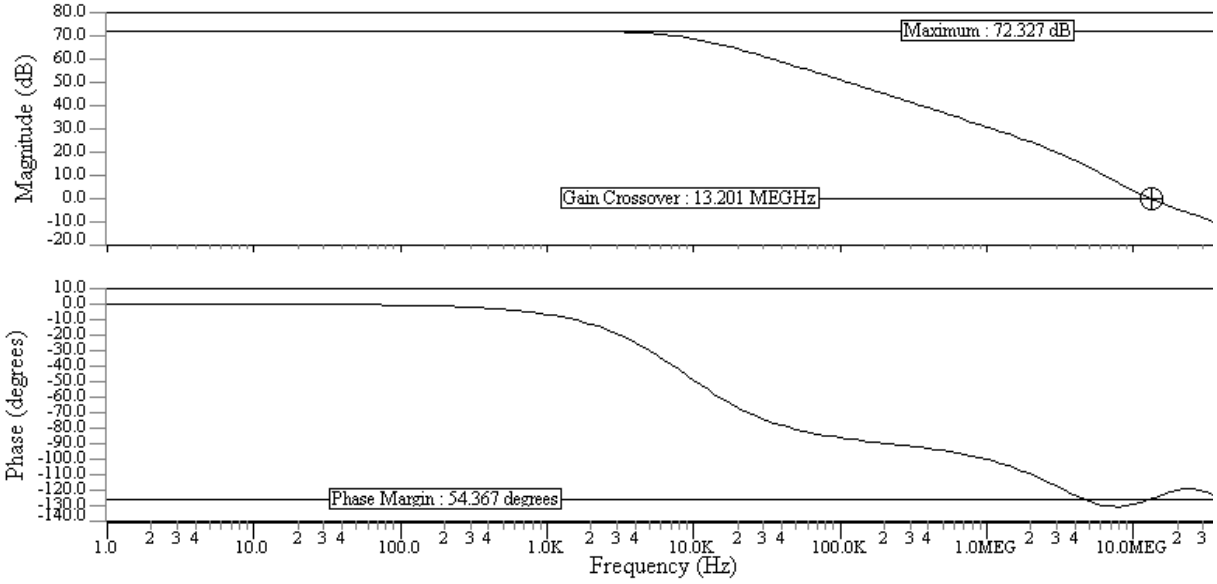


Figura 41 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia ClassAB, obtido a partir da aplicação do enxame de partículas

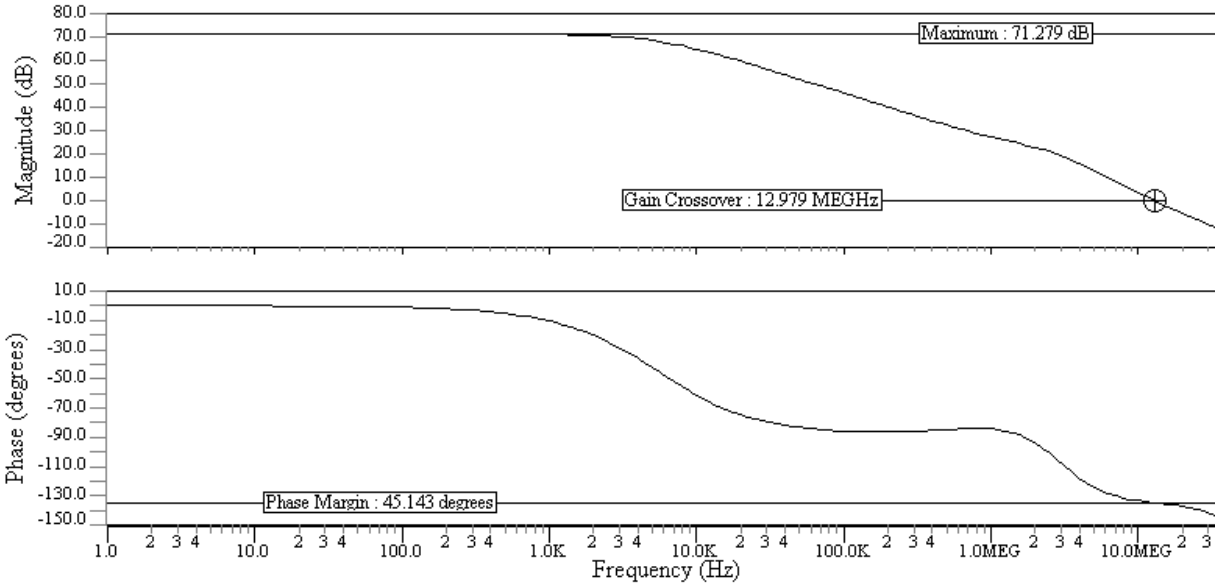
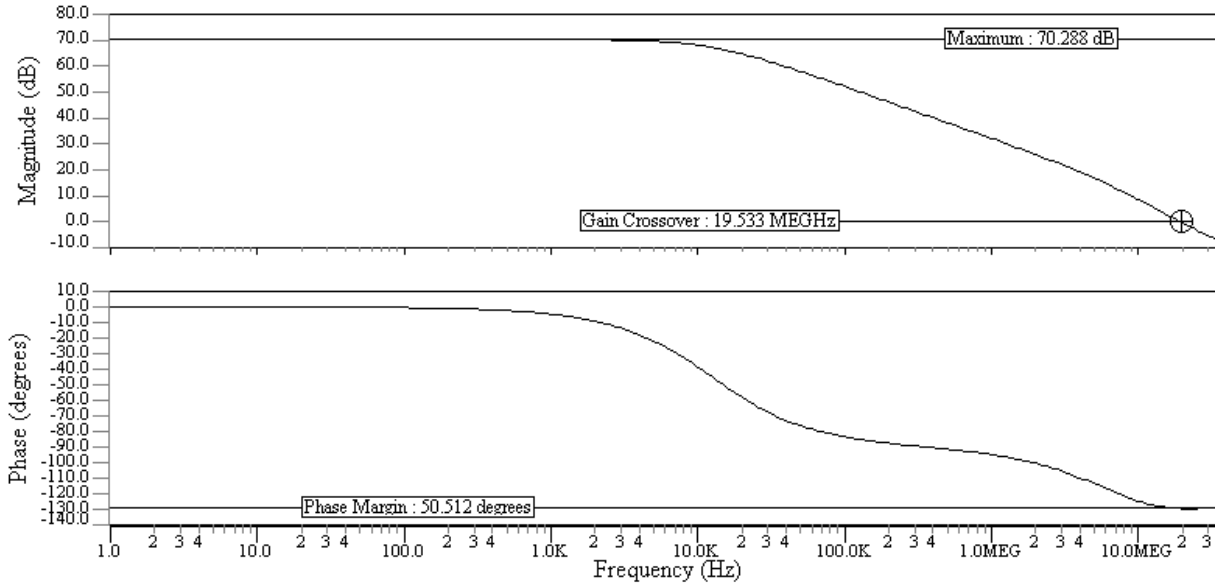


Figura 42 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia ClassAB, obtido a partir da aplicação do recozimento simulado



Na Figura 43, Figura 44 e Figura 45 são apresentadas, respectivamente, a resposta ao degrau do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 43 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia ClassAB, obtido a partir da aplicação do algoritmo genético

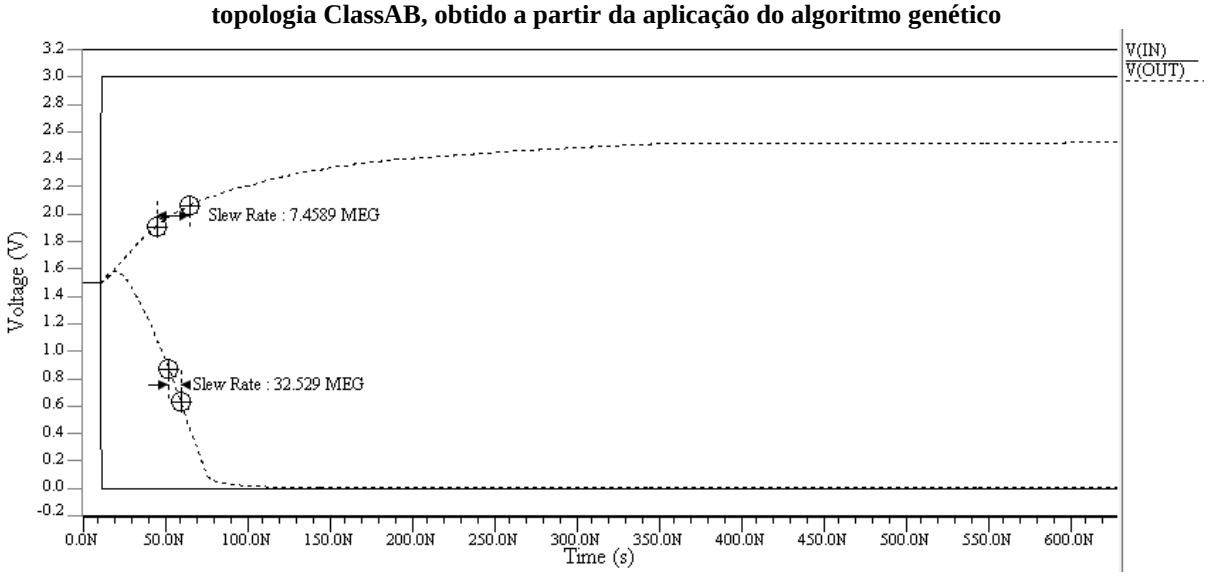


Figura 44 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia ClassAB, obtido a partir da aplicação do enxame de partículas

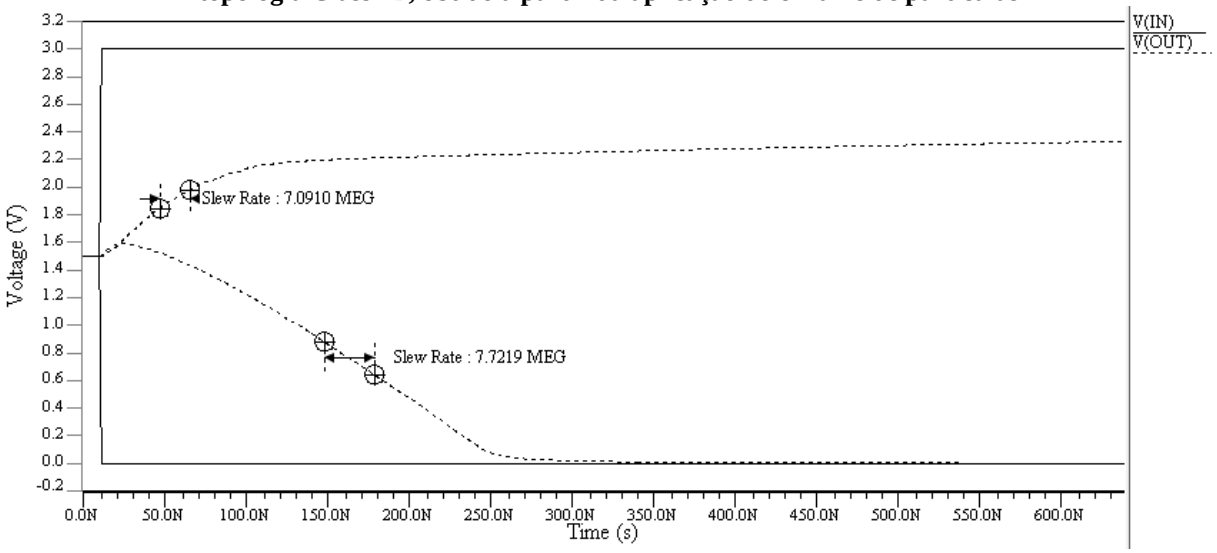
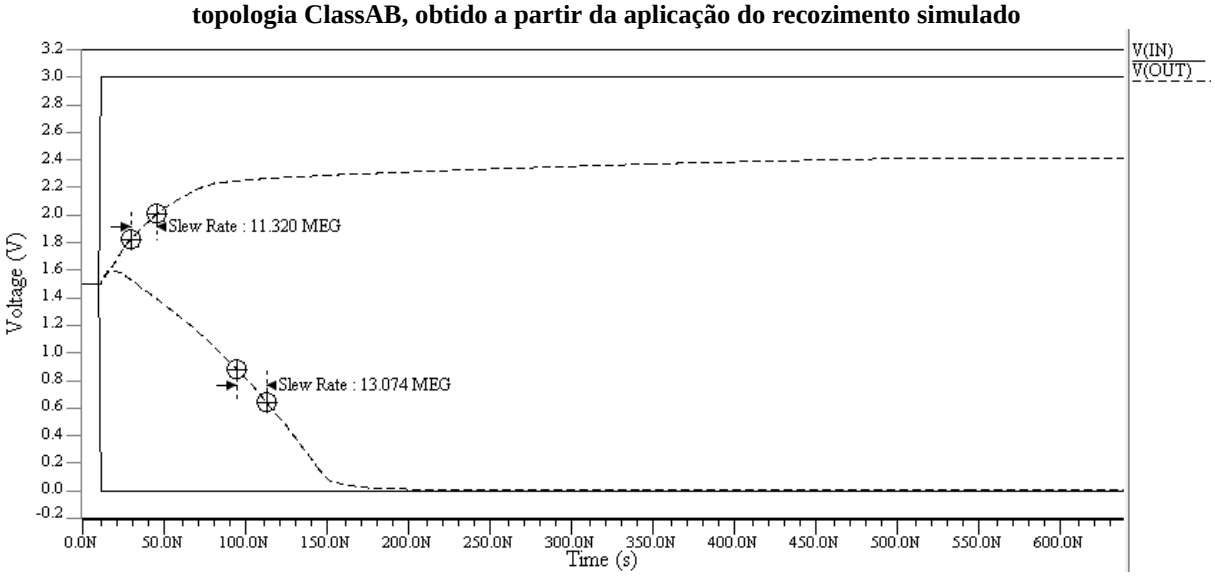
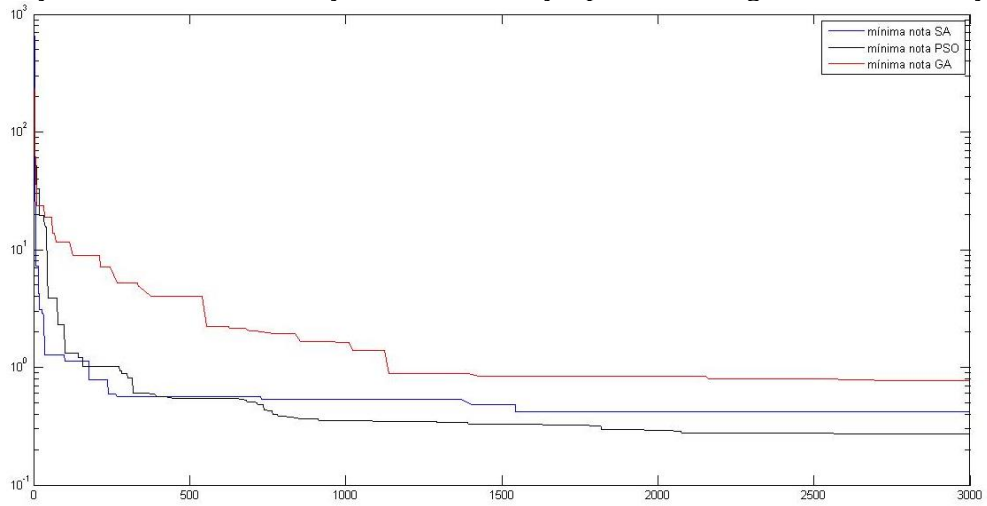


Figura 45 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia ClassAB, obtido a partir da aplicação do recozimento simulado



Na Figura 46 é apresentado o comportamento dos três algoritmos utilizados nessas otimizações, exibindo o gráfico mínimo *score* alcançado, entre as dez otimizações, versus a iteração.

Figura 46 – Evolução dos algoritmos metaheurísticos para a topologia ClassAB: gráfico do mínimo *score* alcançado, entre as dez otimizações, versus a iteração para os três algoritmos de otimização



4.3 Otimização do amplificador de OTA

Para o amplificador com a topologia OTA, da Figura 31, as especificações passadas para a plataforma foram as seguintes:

- Tensões de alimentação: 3,0 V e 0,0 V;
- Ganho de malha aberta mínima: 80 dB;
- Frequência de ganho unitário mínima: 10 MHz;
- *Slew Rate* mínimo: 7,0 V/μs;

- CMRR mínimo: 60 dB;
- PSRR mínimo: 60 dB;
- Limites de margem de fase: 45º e 60º;
- Faixa de tensões de modo comum: 1,2 V e 1,8 V;
- A banda de frequência para o CMRR: 1,0 Hz a 100 Hz;
- O ruído de entrada máximo: 10 μV;
- A tensão de *offset* máxima: 500 μV;
- A área do circuito máxima: 50000 μm²;
- A potência máxima: 1000 μW;
- Duas constantes: $W_{ni}/W_{ni2} = 10$ e $C_L = 1{,}0$ pF;
- Nenhum transistor necessita atuar em fraca inversão;

Na Tabela 14 são apresentados os valores dos melhores candidatos obtidos com o auxilio do algoritmo genético, enxame de partículas e recozimento simulado juntamente com os limites mínimo e máximo das variáveis de otimização escolhidos:

Tabela 14 – Valores das variáveis obtidas na otimização com algoritmo genético, enxame de partículas e recozimento simulado da topologia OTA

Parâmetro		**X_1 (μm)**	**X_2 (μm)**	**X_3 (μm)**	**X_4 (μm)**	**X_5 (μm)**
Valor mínimo		0,5	0,5	0,5	0,5	1,0
Valor obtido	**GA**	1,34	4,99	1,01	0,67	48,99
	PSO	0,51	3,54	5,56	0,91	101,21
	SA	2,37	3,85	3,98	0,9	55,53
Valor máximo		5	5	10	10	400

Parâmetro		**X_6 (μm)**	**X_7 (μm)**	**X_8 (μm)**	**X_9 (μm)**	**X_{10}**
Valor mínimo		1,0	1,0	1,0	-1,0	2,0
Valor obtido	**GA**	9,98	8,67	11,49	-0,49	3,0
	PSO	4,0	1,11	11,32	-0,38	2,0
	SA	3,79	21,57	14,98	-0,43	3,0
Valor máximo		100	100	200	1,0	10

Na Tabela 15, são mostrados os valores dos parâmetros de desempenho dos melhores circuitos encontrados nas otimizações realizadas pelo algoritmo genético, enxame de partículas e recozimento simulado, bem como os *scores* alcançados por cada um dos algoritmos.

Tabela 15 – Valores dos parâmetros de desempenho e de *score* obtidos da otimização com algoritmo genético da topologia OTA

Critério de desempenho		Ganho (dB)	Frequência de ganho unitário (MHz)	CMRR (dB)	PSRR (dB)	Margem de fase	*Score*
Valor obtido	GA	84	14	61	82	52º	0,0015
	PSO	84	13	65	82	51º	0,0015
	SA	93	14	76	89	48º	0,0027

Critério de desempenho		*Slew Rate* (V/µs)	Potência (µW)	Área (µm²)	*Offset* na entrada (µV)	Ruído na entrada (µVrms)	
Valor obtido	GA	7,5	37	1100	490	4,7	
	PSO	7,0	37	900	500	6,9	
	SA	7,5	43	2400	500	4,5	

Na Figura 47, Figura 48 e Figura 49, são apresentados, respectivamente, o diagrama de Bode da resposta em malha aberta do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 47 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia OTA, obtido a partir da aplicação do algoritmo genético

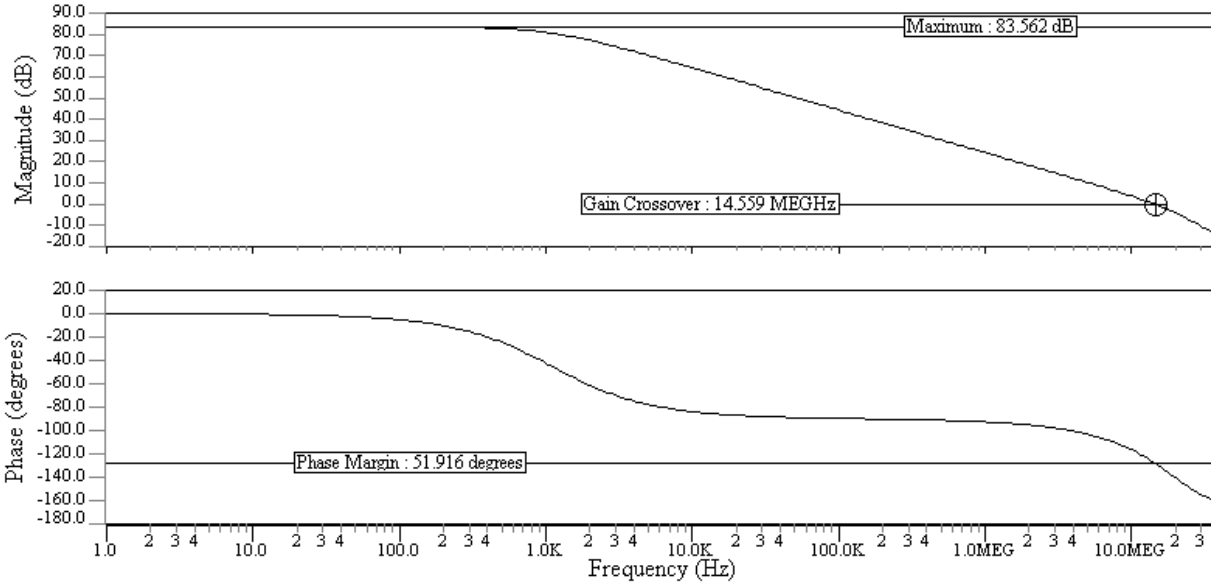


Figura 48 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia OTA, obtido a partir da aplicação do enxame de partículas

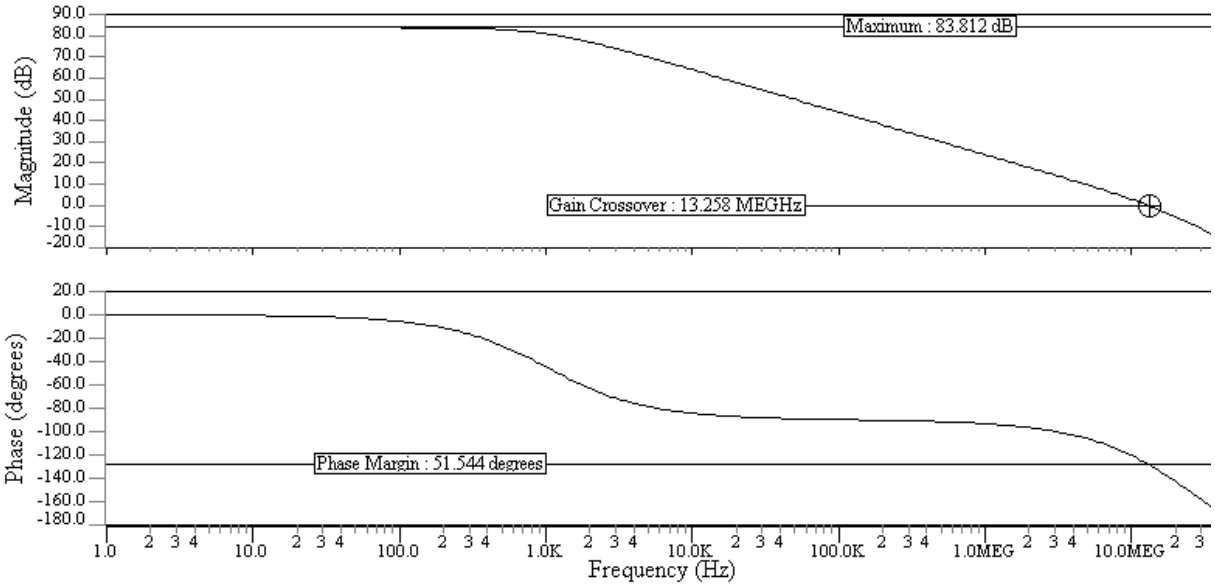
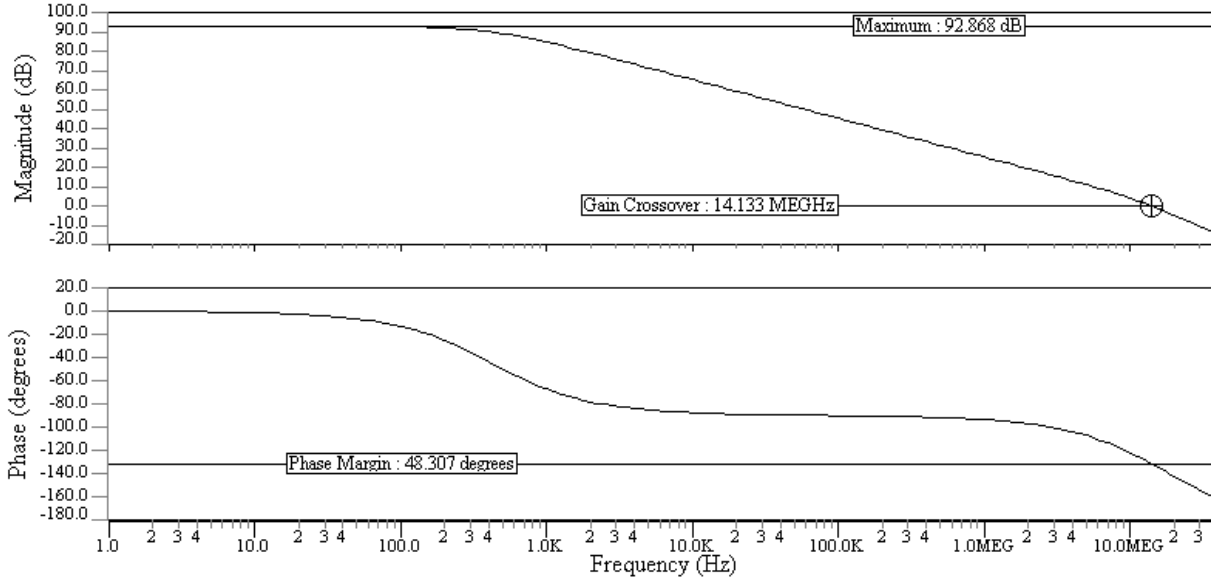


Figura 49 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia OTA, obtido a partir da aplicação do recozimento simulado



Na Figura 50, Figura 51 e Figura 52 são apresentadas, respectivamente, a resposta ao degrau do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 50 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia OTA, obtido a partir da aplicação do algoritmo genético

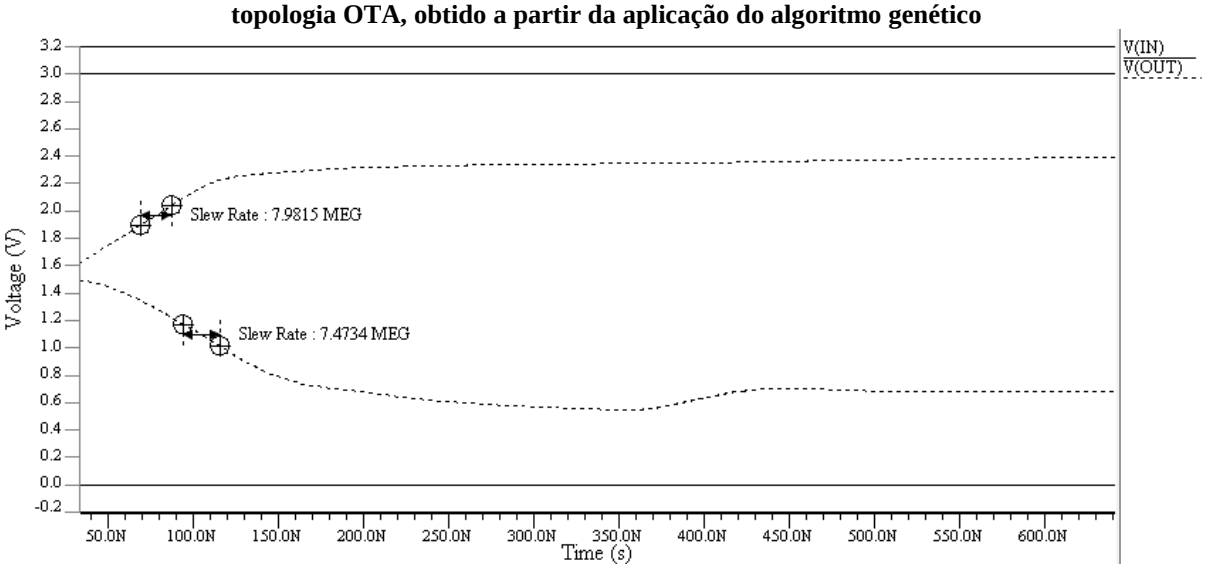


Figura 51 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia OTA, obtido a partir da aplicação do enxame de partículas

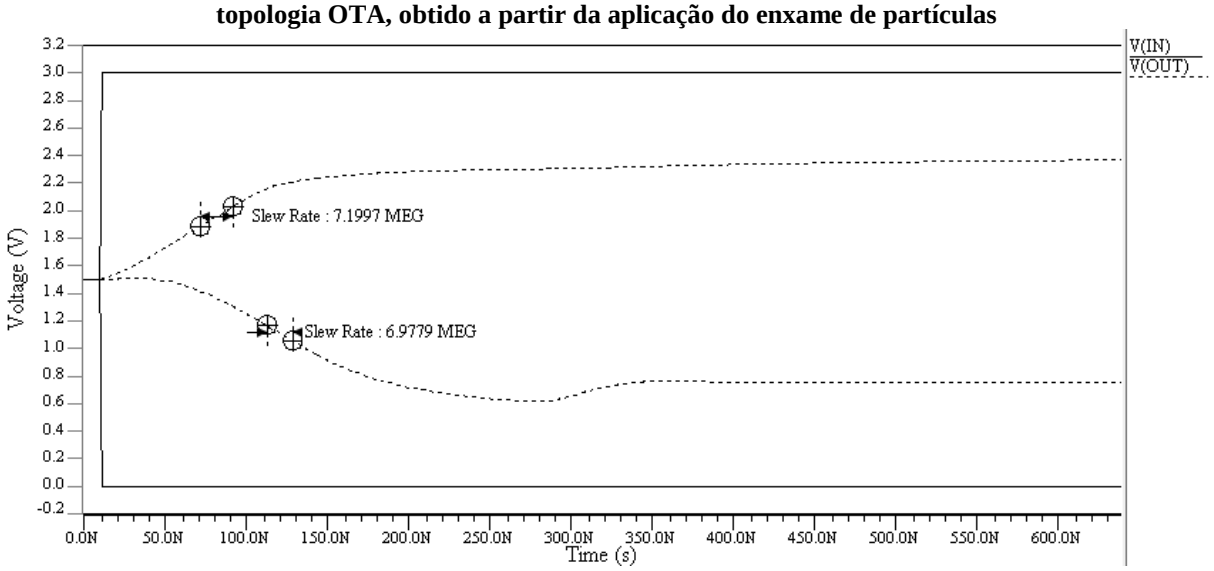
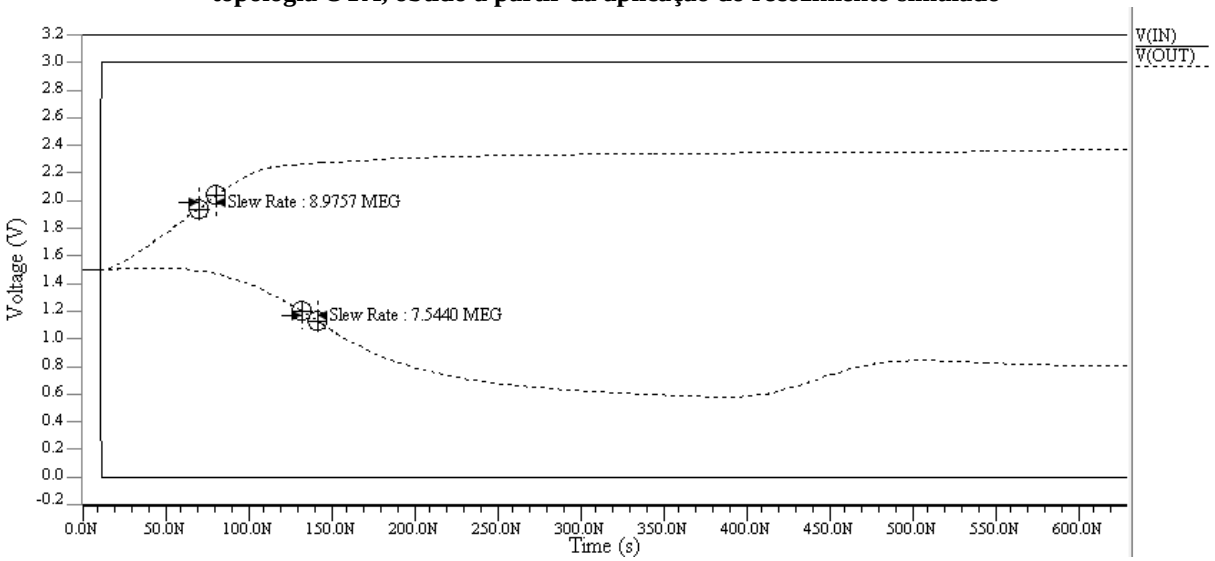
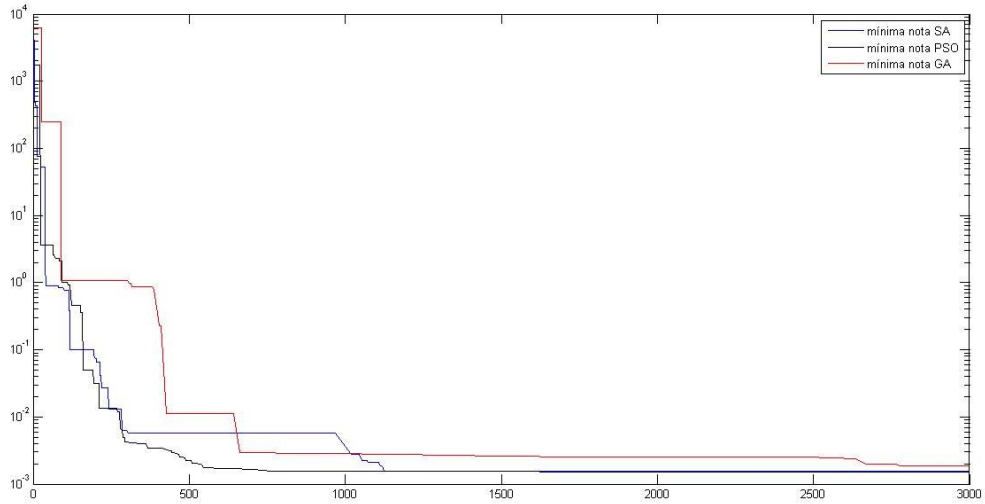


Figura 52 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia OTA, obtido a partir da aplicação do recozimento simulado



Na Figura 53 é apresentado o comportamento dos três algoritmos utilizados nessas otimizações, exibindo o gráfico minimo *score* alcançado, entre as dez otimizações, versus a iteração.

Figura 53 – Evolução dos algoritmos metaheurísticos para a topologia OTA: gráfico do mínimo *score* alcançado, entre as dez otimizações, versus a iteração para os três algoritmos de otimização



4.4 Otimização do amplificador de FoldedCascode

Para o amplificador com a topologia FoldedCascode, da Figura 32, as especificações passadas para a plataforma foram as seguintes:

- Tensões de alimentação: 3,0 V e 0,0 V;
- Ganho de malha aberta mínima: 80 dB;
- Frequência de ganho unitário mínima: 10 MHz;
- *Slew Rate* mínimo: 7,0 V/μs;

- CMRR mínimo: 60 dB;
- PSRR mínimo: 60 dB;
- Limites de margem de fase: 45º e 60º;
- Faixa de tensões de modo comum: 1,2 V e 1,8 V;
- A banda de frequência para o CMRR: 1,0 Hz a 100 Hz;
- O ruído de entrada máximo: 10 μV;
- A tensão de *offset* máxima: 500 μV;
- A área do circuito máxima: 50000 μm²;
- A potência máxima: 1000 μW;
- Duas constantes: $W_5/W_{bn} = W_3/W_{bp} = 10$ e $C_L = 1{,}0$ pF;
- Nenhum transistor necessita atuar em fraca inversão;

Na Tabela 16 são apresentados os valores dos melhores candidatos obtidos com o auxilio do algoritmo genético, enxame de partículas e recozimento simulado juntamente com os limites mínimo e máximo das variáveis de otimização escolhidos.

Tabela 16 – Valores das variáveis obtidas na otimização com algoritmo genético, enxame de partículas e recozimento simulado da topologia FoldedCascode

Parâmetro		**X_1 (μm)**	**X_2 (μm)**	**X_3 (μm)**	**X_4 (μm)**	**X_5 (μm)**	**X_6 (μm)**
Valor mínimo		0,5	0,5	0,5	0,5	0,5	0,5
Valor obtido	**GA**	4,6	0,79	0,57	1,64	3,85	1,68
	PSO	4,76	1,96	0,51	6,62	4,63	0,64
	SA	3,69	2,35	0,58	2,06	1,68	1,25
Valor máximo		5,0	5,0	10	10	10	5,0
Parâmetro		**X_7 (μm)**	**X_8 (μm)**	**X_9 (μm)**	**X_{10} (μm)**	**X_{11} (μm)**	**X_{12} (μm)**
Valor mínimo		1,0	1,0	1,0	1,0	1,0	1,0
Valor obtido	**GA**	33,44	6,79	24,03	32,56	44,21	38,9
	PSO	55,14	1,17	1,09	4,16	12,75	2,0
	SA	53,69	2,81	17,71	50,34	16,24	78,97
Valor máximo		400	100	100	100	100	100
Parâmetro		**X_{13} (μm)**	**X_{14} (μm)**	**X_{15} (μm)**	**X_{16} (μm)**	**X_{17}**	
Valor mínimo		0,5	1,0	0,5	1,0	-1,0	
Valor obtido	**GA**	13,47	1,15	7,15	3,74	-0,06	
	PSO	18,27	1,28	18,64	1,09	-0,14	
	SA	19,56	2,2	14,83	5,93	-0,08	
Valor máximo		20	100	20	100	1,0	

Na Tabela 17, são mostrados os valores dos parâmetros de desempenho dos melhores circuitos encontrados nas otimizações realizadas pelo algoritmo genético, enxame de partículas e recozimento simulado, bem como os *scores* alcançados por cada um dos algoritmos.

Tabela 17 – Valores dos parâmetros de desempenho e de *score* obtidos da otimização com algoritmo genético da topologia FoldedCascode

Critério de desempenho		**Ganho (dB)**	**Frequência de ganho unitário (MHz)**	**CMRR (dB)**	**PSRR (dB)**	**Margem de fase**	***Score***
Valor obtido	**GA**	89	10,7	73	96	59º	0,0038
	PSO	81	10	66	84	60º	0,0027
	SA	87	11	73	94	59º	0,0038

Critério de desempenho		***Slew Rate* (V/µs)**	**Potência (µW)**	**Área (µm²)**	***Offset* na entrada (µV)**	**Ruído na entrada (µVrms)**	
Valor obtido	**GA**	7,4	57	2500	29	3,9	
	PSO	7,0	50	1100	64	3,3	
	SA	7,1	57	2400	38	2,9	

Na Figura 54, Figura 55 e Figura 56 são apresentados, respectivamente, o diagrama de Bode da resposta em malha aberta do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 54 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia FoldedCascode, obtido a partir da aplicação do algoritmo genético

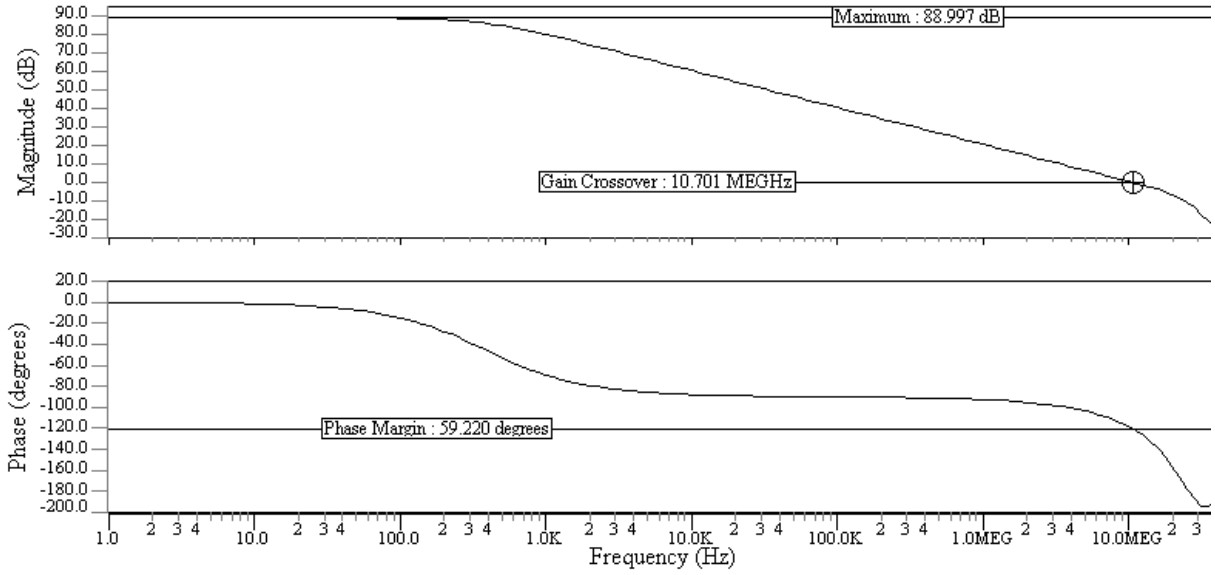


Figura 55 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia FoldedCascode, obtido a partir da aplicação do enxame de partículas

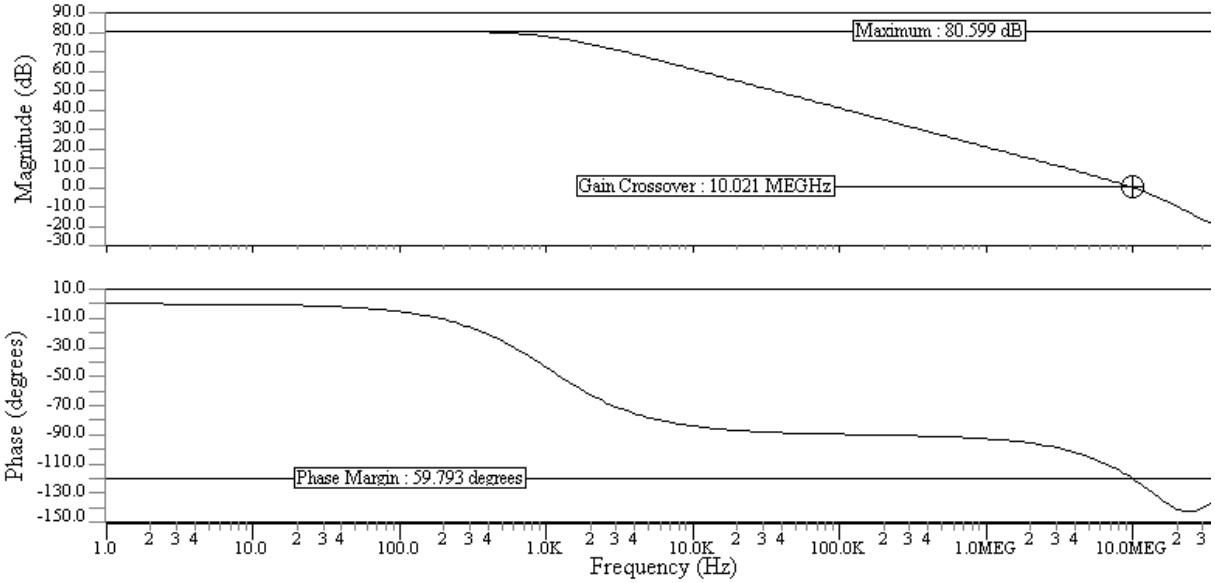
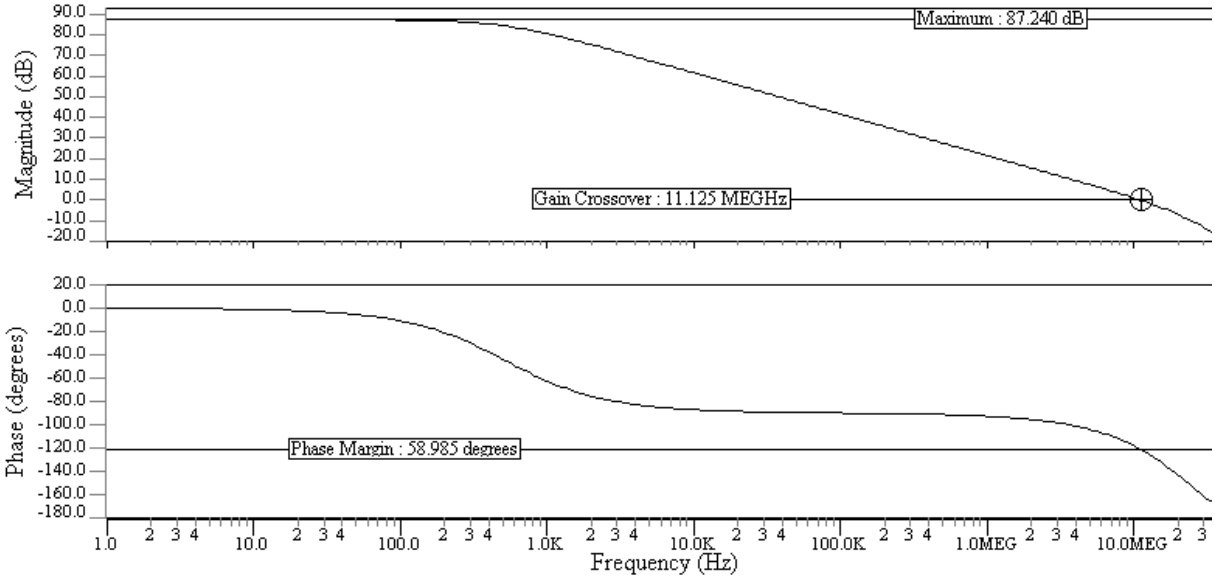


Figura 56 – Diagrama de Bode, modulo e fase, para o melhor circuito, na topologia FoldedCascode, obtido a partir da aplicação do recozimento simulado



Na Figura 57, Figura 58 e Figura 59 são apresentadas, respectivamente, a resposta ao degrau do AmpOp otimizado pelo GA, pelo PSO e pelo SA.

Figura 57 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia FoldedCascode, obtido a partir da aplicação do algoritmo genético

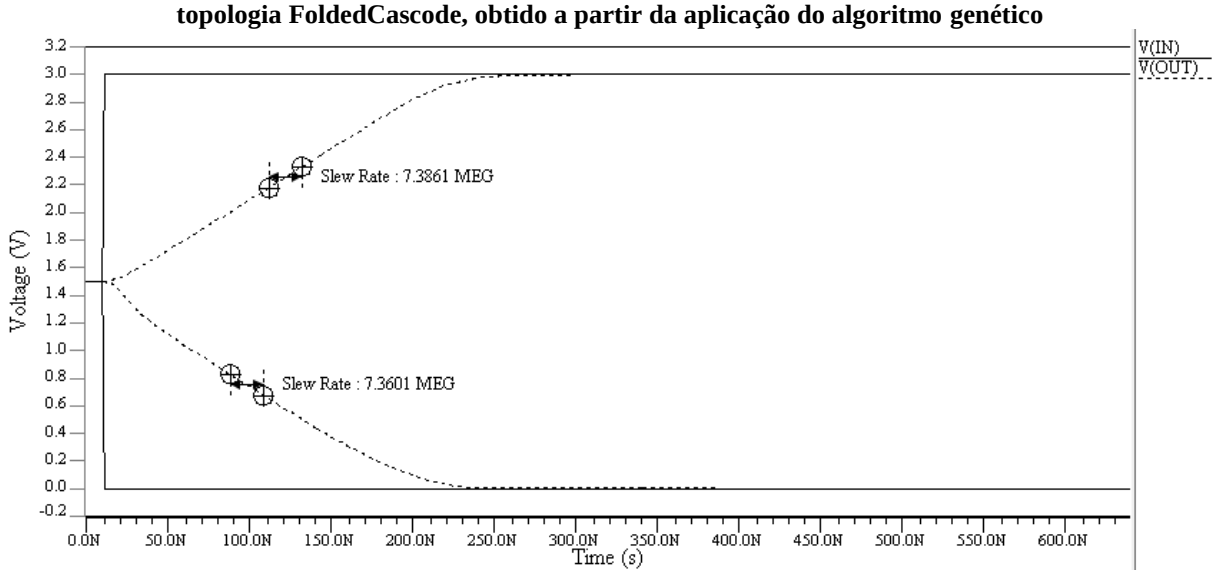


Figura 58 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia FoldedCascode, obtido a partir da aplicação do enxame de partículas

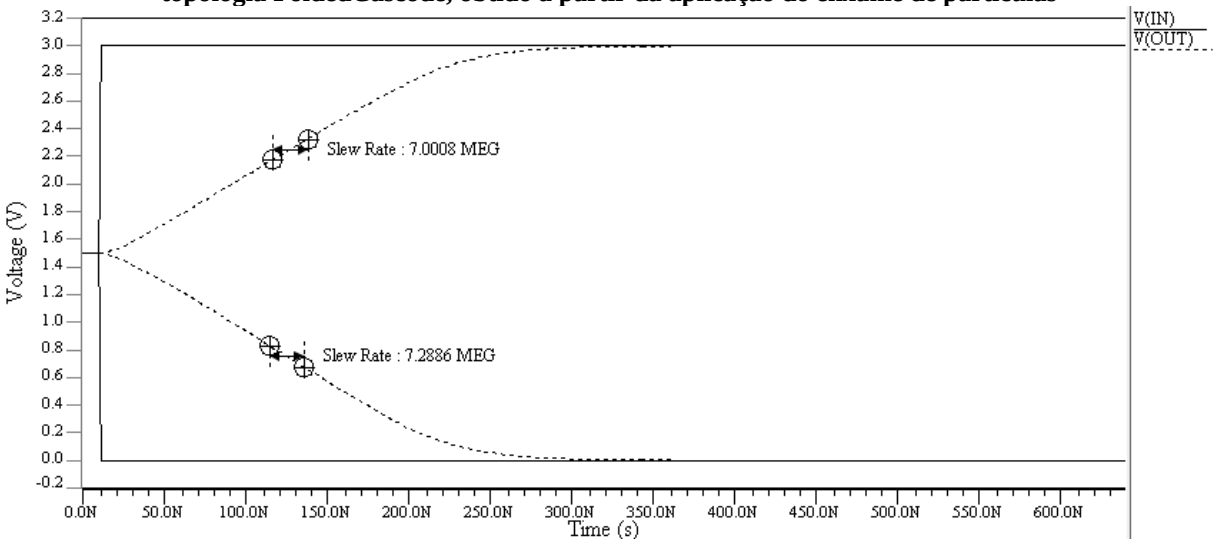
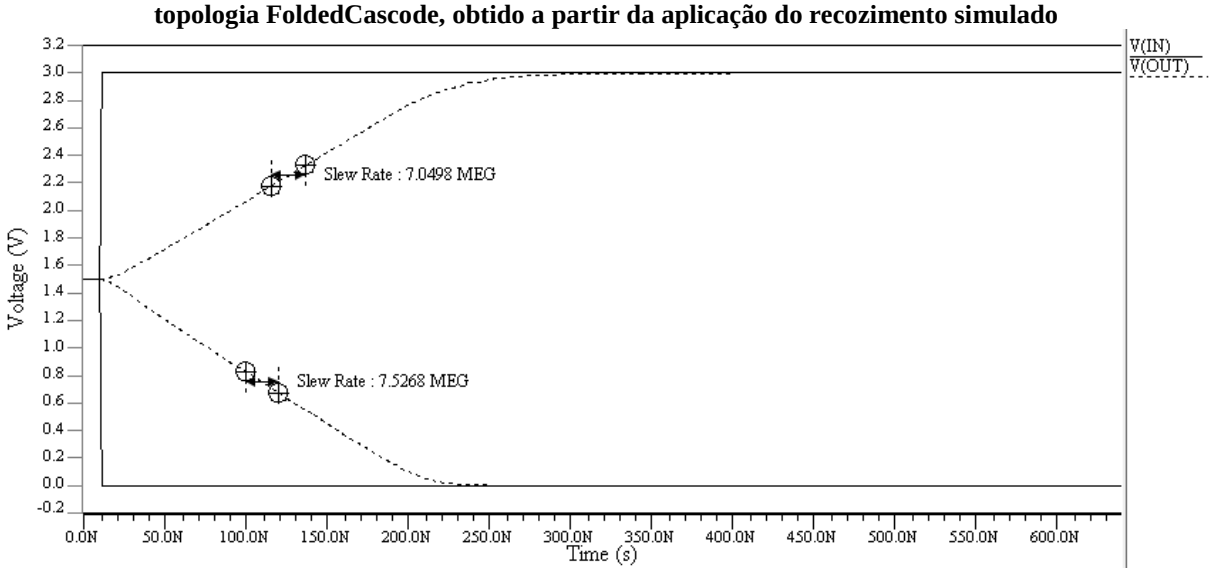
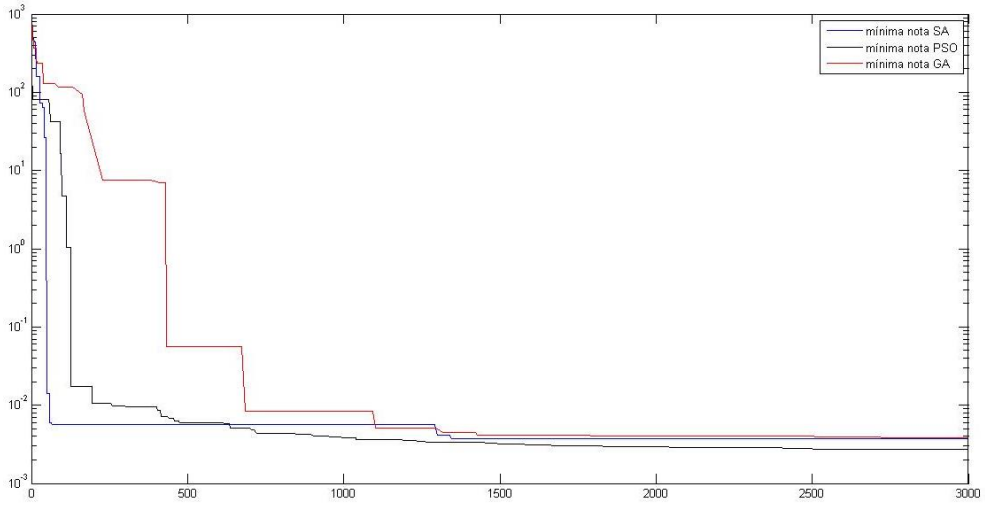


Figura 59 – Resposta no tempo a degraus de subida e descida aplicados a entrada do melhor circuito, na topologia FoldedCascode, obtido a partir da aplicação do recozimento simulado



Na Figura 60 é apresentado o comportamento dos três algoritmos utilizados nessas otimizações, exibindo o gráfico minimo *score* alcançado, entre as dez otimizações, versus a iteração.

Figura 60 – Evolução dos algoritmos metaheurísticos para a topologia FoldedCascode: gráfico do mínimo *score* alcançado, entre as dez otimizações, versus a iteração para os três algoritmos de otimização



4.5 Discussão

A partir dos resultados, nota-se que todos os circuitos conseguiram igualar ou superar as especificações exigidas pelo usuário, independente da metaheurística utilizada.

Tanto a topologia de ClassAB quanto a de OTA apresentaram uma resposta ao degrau, na análise de *slew rate*, bem diferente das topologias de DoisEstagios e FoldedCascode:

- As topologias de DoisEstagios e de FoldedCascode tiveram uma taxa de variação de tensão, de subida e de descida, alta, até o momento que a tensão de saída saturasse em V_{DD} ou V_{SS}, respectivamente;

- As taxas de variação de tensão de subida das topologias ClassAB e OTA foram altas apenas nos instantes iniciais da aplicação do sinal degrau de descida. Em seguida, houve uma diminuição drástica em seus valores, não alcançando a tensão V_{DD};
- Também a taxa de variação de tensão de descida da topologia de OTA foi alta apenas nos instantes iniciais da aplicação do sinal degrau de subida. Em seguida houve a diminuição em seu valor, não alcançando a tensão V_{SS}.

O algoritmo PSO apresentou certa superioridade em relação aos demais, tanto na velocidade com que achou bons circuitos como no *score* final fornecido para todas as topologias, como pode ser observado através dos gráficos que mostram a evolução dos algoritmos metaheurísticos.

Para se ter uma ideia do tempo gasto em cada algoritmo para realizar as otimizações, foi anotado o tempo que um computador, com as especificações apresentadas na Tabela 18, levou para otimizar um circuito com a topologia ClassAB a partir do recozimento simulado. O algoritmo levou cerca de 19 horas e 24 minutos para realizar as dez otimizações.

Tabela 18 – Especificações do computador utilizado para as simulações

Processador	Intel® Core™ i7-3537U 2,0 GHz
Sistema Operacional	Windows 8 de 64 bits
Memória RAM	8,0 GB

5 CONCLUSÃO

Neste trabalho, foi apresentada a construção de uma plataforma de auxilio a projetistas, mesmo de pouca experiência, no projeto e otimização de amplificadores operacionais. Tal plataforma utiliza algoritmos metaheuristicos.

Para tanto foram apresentados inicialmente os conhecimento básicos a cerca de transistores MOS, amplificadores operacionais e metaheuríscas, bem como as ferramentas utilizadas. O funcionamento da plataforma, com sua linguagem padrão, os métodos para mensurar o desempenho de cada circuito e a forma de atribuir notas de avaliação, foi explicado em seguida.

Diferentes topologias de AmpOp foram utilizadas para projeto e em todas se alcançou as especificações mínimas oferecidas pelo usuário. Por fim, uma breve discussão sobre os resultados foi realizada, fazendo uma comparação, também, do desempenho de cada algoritmo metaheurístico aqui utilizado.

Várias foram as dificuldades encontradas durante a confecção da plataforma. A principal delas foi decidir como realizar as medidas das características de cada AmpOp (conforme apresentado na seção 3.2). Várias alterações foram realizadas durante o projeto, a fim de deixar a plataforma mais robusta e genérica.

Das topologias aqui analisadas, apenas com a OTA houve algumas dificuldades de simulação com o simulador elétrico Eldo. Além disso, duas topologias estudadas (ClassAB e OTA) apresentaram comportamento na resposta transitória ao degrau inesperado, com uma taxa de variação da tensão de saída alta no início, mas depois diminuindo drasticamente. No geral, entretanto, todas as topologias alcançaram os valores de especificações desejados, com baixo consumo de potência (com exceção de ClassAB, devido à presença de uma resistência de carga) e ocupando uma pequena área, comprovando o sucesso da plataforma.

Pôde-se observar também a superioridade de convergência do algoritmo de enxame de partículas para todas as topologias aqui estudadas: esta metaheurística foi a que conseguiu encontrar os melhores candidatos, dentre os amplificadores operacionais testados. Além disso, a maioria dos indivíduos que apresentaram o mínimo *score* alcançado dentre as dez otimizações, entre os três algoritmos metaheurísticos, foi encontrado pelo PSO. Contudo, todos os algoritmos metaheuríscos se mostraram eficazes e confiáveis no projeto de amplificadores operacionais CMOS.

Um fator de grande importância para a convergência dos algoritmos e de difícil acerto é o valor dos pesos aplicados as metaheuristicas. A escolha desses valores é crucial e afeta o resultado final, visto que uma atribuição de pesos realizada ao acaso pode levar o algoritmo a regiões do espaço de busca onde só são analisados indivíduos com péssimo desempenho. Ao final, o projetista pode considerar, de forma errônea, que a topologia usada não é boa para alcançar as especificações, quando, na realidade, são os valores dos pesos que estão mal escolhidos.

Em trabalhos futuros, deve-se:

- Programar uma opção para que os transistores trabalhem em forte inversão;
- Deixar a plataforma mais robusta contra falhas do simulador elétrico;
- Permitir que o usuário escolha não somente os valores dos pesos, mas também as funções de avaliação;
- Verificar a faixa dinâmica de operação dos amplificadores operacionais;
- Comparar os resultados obtidos com a plataforma com resultados de projetos feitos de forma manual;
- Fabricar e testar os amplificadores projetados;
- Acrescentar novas topologias;
- Acrescentar novas metaheurísticas e realizar novas comparações entre elas;
- Construir um instalador da plataforma.

REFERÊNCIAS BIBLIOGRÁFICAS

Allen, P. E., & Hoberg, D. R. (2002). *CMOS Analog Circuit Design.* Oxford: Oxford University Press.

AUSTRIAMICROSYSTEMS. (2003). *0.35µm CMOS C35 Design Rules* (ENG 183, rev. 3 ed.). AustrianMicroSystems.

AUSTRIAMICROSYSTEMS. (2014). Acesso em 14 de Agosto de 2014, disponível em AMS FOUNDRY SUPORT: http://asic.austriamicrosystems.com/

Carvalho, A. P. (2009). *Algoritmos Genéticos*. Acesso em 5 de Setembro de 2014, disponível em Site do ICMC: http://www.icmc.usp.br/pessoas/andre/research/genetic/

ELDO. (2014). Acesso em 17 de Setembro de 2014, disponível em Site da Mentor Graphics: http://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/eldo/

Filgueiras, I. F. (2010). *Otimização de circuitos CMOS por Algoritmo Genético.* Universidade de São Paulo, Departamento de Engenharia Elétrica e Computação, São Carlos.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* MIT Press.

HSPICE. (2014). Acesso em 6 de Agosto de 2014, disponível em SYNOPSYS: http://www.hspice.com

ITRS. (2005). Acesso em 30 de Outubro de 2014, disponível em International technology roadmap for semiconductors: http://www.itrs.net/Links/2009ITRS/Home2009.htm

Kennedy, J., & Eberheart, R. (1995). Particle swarm optimization. *IEEE International Conference* (pp. 1942 - 1948). Perth: IEE.

Kirkpatrick, S., Gellat, C. D., & Vecchi, M. P. (Maio de 1983). Optimization by Simulated Annealing. *Science, vol. 220*, pp. 671-680.

Luke, S. (2013). *Essentials of Metaheuristics* (Vol. II). Fairfax: Lulu.com.

MATLAB. (2014). Acesso em 16 de Junho de 2014, disponível em The Language of Techinical Computing: http://www.mathworks.com/products/matlab/

Oliveira, A. C. (2006). *Uso do Algoritmo Genético e Recozimento Simulado para o Problema de Alocação de Salas.* Universidade Federal de Lavras, Departamento de Ciência da Computação, Lavras.

Pertence Júnior, A. (2012). *Eletrônica Analógica: Amplificadores Operacionais e Filtros Analógicos.* Porto Alegre: Teckne.

Pimenta, T. C., Moreno, R. L., & Zoccal, L. B. (20 de Julho de 2011). *RF CMOS Background*. Acesso em 15 de Agosto de 2014, disponível em Site da INTECH: http://www.intechopen.com/books/current-trends-and-challenges-in-rfid/rf-cmos-background

Roberts, M. J. (2010). *Fundamentos de Sinais e Sistemas.* Porto Alegre: McGrawHill.

Sedra, A. S., & Smith, K. C. (2004). *Microeletronic Circuits* (5ª ed.). Oxford University Press.

Simon, H., & Barry, V. V. (2001). *Sinais e Sistemas.* Porto Alegre: Bookman.

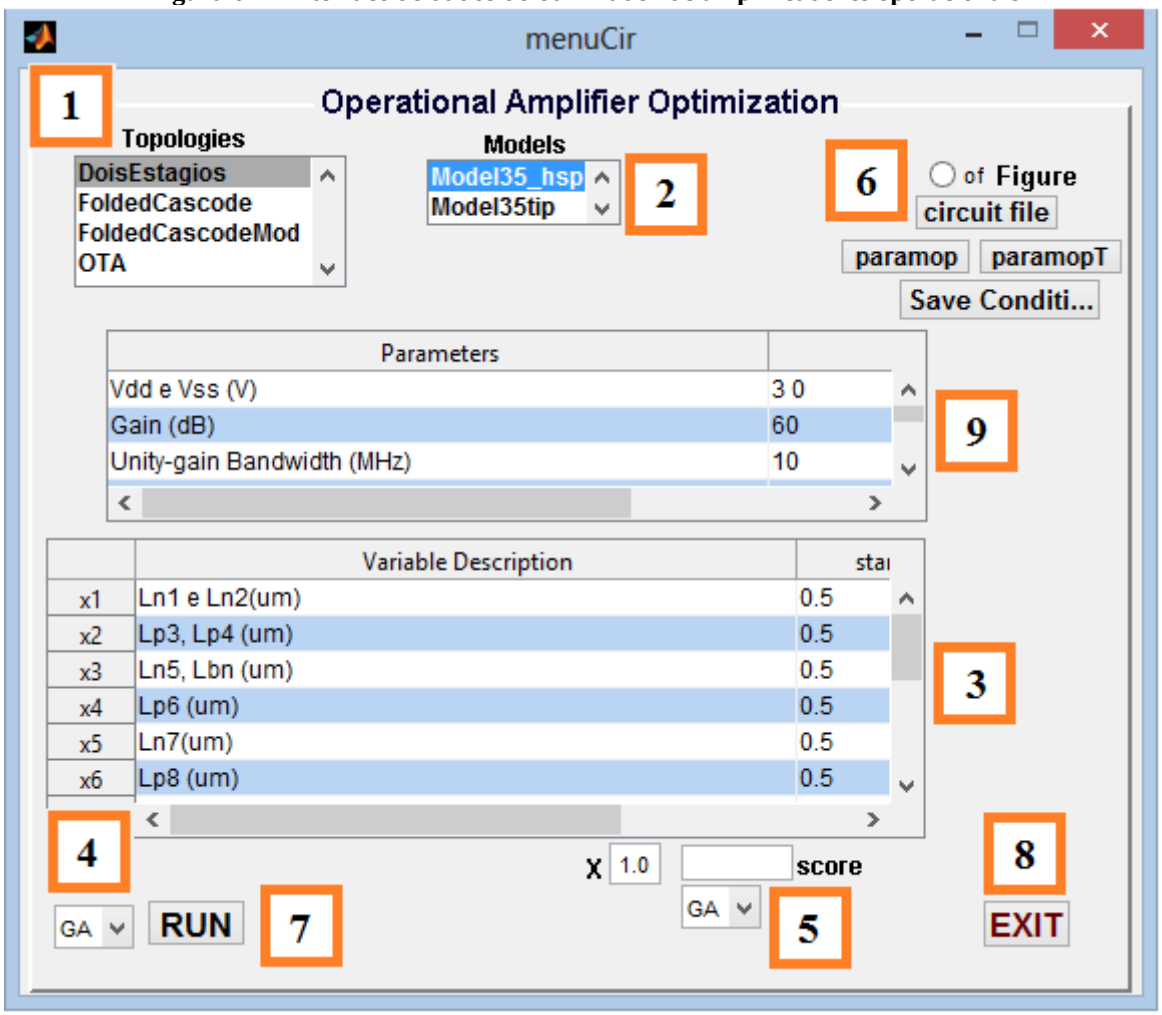
Torres, O. H. (2012). *Otimização de Circuitos CMOS por Algoritmo Genético.* Universidade de São Paulo, Departamento de Engenharia Elétrica e Computação, São Carlos.

Tranquilini, B. C. (2008). *Projeto de Amplificador Operacional em Tecnologia CMOS.* Universidade de São Paulo, Departamento de Engenharia Elétrica e Computação, São Carlos.

APÊNDICE A

Interface de dados e de execução

Figura 61 – Interface de dados do otimizador de amplificadores operacionais



Na interface de dados, o usuário pode:

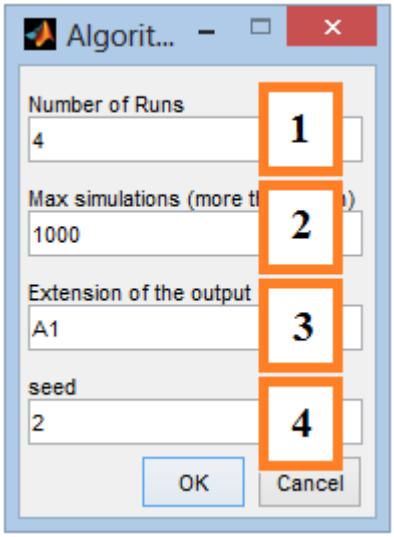
- escolher a topologia desejada **(1)**;
- escolher os modelos dos transistores **(2)** a serem usados;
- escolher os limites mínimos e máximos **(3)** das variáveis que se desejam otimizar, bem como valores de uma solução inicial. A melhor solução encontrada por um determinado algoritmo, até então, está também disponível caso o usuário deseje utilizá-la como inicial **(5);**
- escolher o tipo de algoritmo que se deseja usar **(4)**;
- verificar, enquanto a otimização ocorre, os valores das variáveis e o desempenho do melhor circuito até então encontrado na simulação atual (paramop) ou do melhor

circuito encontrado por todas as otimizações realizadas até o momento (paramopT) **(6)**.

- iniciar o projeto e otimização **(7)**
- escolher os parâmetros de desempenho do circuito **(9)**:
  - Tensão das fontes de alimentação Vdd e Vss, respectivamente, em V (caso não seja informado o valor da segunda fonte, o programa considera 0,0 V);
  - O ganho diferencial mínimo em dB;
  - A banda de ganho unitário mínima, em MHz;
  - O *slew* rate mínimo, em V/µs;
  - O CMRR mínimo, em dB;
  - O PSRR mínimo, em dB;
  - O intervalo para a margem de fase, em graus;
  - As tensões de modo comum mínima e máxima que o circuito irá trabalhar, em V;
  - A faixa de frequência para o CMRR, em Hz;
  - O ruído referido a entrada máximo, em µV;
  - A tensão de *offset* de entrada máxima, em mV;
  - Um valor de área do circuito para referência, em µm²;
  - Um valor de potência para referência, em µW;
  - As constantes que o circuito utiliza (se houver);
  - Os transistores que deverão atuar em fraca inversão (se houver);
  - Os pesos para a avaliação do circuito (será explicado mais adiante);

Ao selecionar o botão (7), o usuário se depara com outra interface, a interface de execução apresentada na Figura 62.

Figura 62 – Interface de execução do otimizador de amplificadores operacionais



Na interface de execução, o usuário deve indicar o número otimizações que serão realizadas **(1)**; o número máximo de indivíduos analisados **(2)** em cada otimização; a extensão dos arquivos de saída **(3)** da plataforma e a semente para a geração das soluções iniciais **(4)** (caso nenhum dado seja fornecido neste campo, a solução inicial é extraída da interface de dados).

APÊNDICE B

Função *fitnessEldo.m*

```
function [sc, sci] = fitnessEldo(x)

global slash;
global simuladorEldo;
global circuito;
global ParDados
global genesLB;
global AreaCir;
global weakTrans;
global const;
global pesos;
global Best
global BestRes;
global modoind;
global modo;
global dif;
global cont;
global namext;
global Ganho ;
global Vss;
global Vcmin;
global Vcmax;
global CMRRparam;
global Vdd;
global SlewRate;
global Fo;
global Pot;

%leitura das especificações do usuário
Vdd = max(str2num(ParDados{1, 2}));
if (length(str2num(ParDados{1, 2})) == 2)
    Vss = min(str2num(ParDados{1, 2}));
else
    Vss = 0;
end
Ganho = str2num(ParDados{2, 2});
Fo = str2num(ParDados{3, 2})*1e6;
SlewRate = str2num(ParDados{4, 2})*1e6;
CMRRparam(1) = str2num(ParDados{5, 2});
PSRR = str2num(ParDados{6, 2});
PhaseMarginMin = min(str2num(ParDados{7, 2}));
PhaseMarginMax = max(str2num(ParDados{7, 2}));
Vcmin = min(str2num(ParDados{8,2}));
Vcmax = max(str2num(ParDados{8,2}));
CMRRparam(2) = min(str2num(ParDados{9, 2}));
CMRRparam(3) = max(str2num(ParDados{9, 2}));
noise = str2num(ParDados{10,2})*1e-6;
offset0 = str2num(ParDados{11,2})*1e-3;
AreaCir = str2num(ParDados{12, 2});
Pot = str2num(ParDados{13, 2})*1e-6;
const = str2num(ParDados{14, 2});
weakTrans = strread(ParDados{15, 2}, '%s');
```

```
pesos = str2num(ParDados{16, 2});

j=1;
Weakin = 0;
simsucess =0;
%desnormaliza os genes
for i=1:length(genesLB)
    if (dif(i) ~= 0)
        xr(i)= (x(j)*dif(i)+genesLB(i));
        j=j+1;
    else xr(i)=genesLB(i);
    end;
end;
%calculo da área do circuito
cond = [circuito slash 'AreaCirMea.m'];
if exist(cond)
    eval(['cd ', circuito]);
    AreaMed = AreaCirMea(xr);
    eval('cd ..');
else AreaMed = 0;
end;


%xr = x.*dif+genesLB;
disp _____________________________
cont=cont +1;
fprintf('simulação = %d\n', cont);


% Encontrando os valores de offset, weakTrans e potência
arq = fopen([circuito slash 'param'],'w');
paramEx(arq,xr);
fclose(arq);
[~, b] = system([simuladorEldo circuito slash 'circuito.cir'], '-echo');

% lê os resultados da simulação
Meas= file2tableF(4, [circuito slash 'circuito.aex']);

% verifica se executou bem a primeira simulação
if(length(Meas) > 3 + length(weakTrans))
    Vcmaxoffset = Meas(1); Vcminoffset = Meas(2); Vcmedioffset = Meas(3);
    Potcircuit = (Vdd-Vss)*abs(Meas(4));
    offset = max([abs(Vcmax - Vcmaxoffset) abs(Vcmin - Vcminoffset)
abs((Vcmax+Vcmin)/2 - Vcmedioffset)]);
    %Weakin
    for i= 1: length(weakTrans);
        j= 4+i;
        Meas(j)= abs(Meas(j));
        if Meas(j)> 0.12
            Weakin=10*Meas(j) +Weakin;
        end;
    end;

    %Encontrando o Ganho de Modo Comum (Ac)
    arq = fopen([circuito slash 'param'],'w');
    paramganhoMC(arq,xr,Vcmedioffset);
    fclose(arq);
    [~, b] = system([simuladorEldo circuito slash 'circuito.cir'], '-
echo');
```

```
    % lê os resultados da simulação
    Meas= file2tableF(4, [circuito slash 'circuito.aex']);

    % verifica se executou bem a segunda simulação
    if(length(Meas) > 0)
        GanhoModoComum = abs(Meas(1));

        %Análise ac para ambas as tensões de modo comum (a máxima e a
mínima), ruído e PSRR
        arq = fopen([circuito slash 'param'],'w');
        paramAC(arq,xr,Vcminoffset,Vcmaxoffset,Vcmedioffset);
        fclose(arq);
        [~, b] = system([simuladorEldo circuito slash 'circuito.cir'], '-
echo');

        % lê os resultados da simulação
        Meas= file2tableF(4, [circuito slash 'circuito.aex']);

        % verifica se executou bem a terceira simulação
        if (length(Meas) > 14 )
            GanhoMax = Meas(1);
            GanhoMax1 = Meas(7);
            CircuitNoise = Meas(2);
            CircuitNoise1 = Meas(8);
            FaseFrequenciaGanhoUnitario = Meas(4);
            FaseFrequenciaGanhoUnitario1 = Meas(10);
            FrequenciaGanhoUnitario = Meas(5);
            FrequenciaGanhoUnitario1 = Meas(11);

            PowerGain = Meas(13);
            GroundGain = Meas(15);

            %Análise do Slew Rate
            arq = fopen([circuito slash 'param'],'w');
            paramSlewRate(arq,xr, Vcmedioffset);
            fclose(arq);
            [~, b] = system([simuladorEldo circuito slash 'circuito.cir'],
'-echo');

            % lê os resultados da simulação
            Meas= file2tableF(4, [circuito slash 'circuito.aex']);

            % verifica se executou bem a quarta simulação
            if(length(Meas) > 6)
                SlewRatecircuit = min([abs(Meas(3)) abs(Meas(7))]);

                if(FaseFrequenciaGanhoUnitario > 0)
                    FaseFrequenciaGanhoUnitario =
FaseFrequenciaGanhoUnitario - 360;
                end
                if(FaseFrequenciaGanhoUnitario1 > 0)
                    FaseFrequenciaGanhoUnitario1 =
FaseFrequenciaGanhoUnitario1 - 360;
                end
                MargemFase= 180 - abs(FaseFrequenciaGanhoUnitario);
                MargemFase1= 180 - abs(FaseFrequenciaGanhoUnitario1);
                CMRRcircuit = (GanhoMax-GanhoModoComum);
                CMRRcircuit1 = (GanhoMax1-GanhoModoComum);
```

```
                CMRR = CMRRparam(1);

                %FGM: (GanhoMax > Ganho) e (GanhoMax1 > Ganho) deve ocorrer
                FGM = max(0, (Ganho - GanhoMax)/Ganho) + max(0, (Ganho -
GanhoMax1)/Ganho);

                GanhoMax = min(GanhoMax,GanhoMax1);

                %FFGU: (FrequenciaGanhoUnitario > Fo) e
(FrequenciaGanhoUnitario1 > Fo) deve ocorrer
                FFGU = max(0, (Fo - FrequenciaGanhoUnitario) / Fo) + max(0,
(Fo - FrequenciaGanhoUnitario1) / Fo);

                FrequenciaGanhoUnitario =
min(FrequenciaGanhoUnitario,FrequenciaGanhoUnitario1);

                %FCMRR: (CMRRcircuit > CMRR ) e (CMRRcircuit1 > CMRR ) deve
ocorrer
                FCMRR = max(0, (CMRR - CMRRcircuit)/CMRR) + max(0,  (CMRR -
CMRRcircuit1)/CMRR) ;

                CMRRcircuit = min(CMRRcircuit,CMRRcircuit1);

                %FMF: (PhaseMarginMax > MargemFase > PhaseMarginMin) e
(PhaseMarginMax > MargemFase1 > PhaseMarginMin) deve ocorrer
                FMF =max([0  (MargemFase - PhaseMarginMax)/PhaseMarginMax
(-MargemFase + PhaseMarginMin)/PhaseMarginMin]);
                FMF = FMF + max([0  (MargemFase1 -
PhaseMarginMax)/PhaseMarginMax  (-MargemFase1 +
PhaseMarginMin)/PhaseMarginMin]);

                %FSL: (SlewRate > SlewRatecircuit) deve ocorrer
                FSL = max(0, (SlewRate - SlewRatecircuit)/SlewRate);

                %FPSRR: (PSRRcircuit > PSRR) deve ocorrer
                PSRRcircuit = min(GanhoMax-PowerGain,GanhoMax-GroundGain);
                FPSRR = max(0, (PSRR - PSRRcircuit)/PSRR);

                %FP
                FP = Potcircuit/Pot;

                % Farea
                Farea = AreaMed/AreaCir;

                %Foffset
                Foffset = max(0, (offset-offset0)/offset0);

                %Fnoise
                CircuitNoise = CircuitNoise/(10^(GanhoMax/20));
                CircuitNoise1 = CircuitNoise1/(10^(GanhoMax/20));

                Fnoise = max(0, (CircuitNoise - noise)/noise) + max(0,
(CircuitNoise1 - noise)/noise);

                CircuitNoise = max(CircuitNoise,CircuitNoise1);

                % nota final
```

```
                sc = ( pesos(1)*FGM + pesos(2)*FFGU  + pesos(3)*FCMRR +
pesos(4)*FPSRR + pesos(5)*FMF + pesos(6)*FSL + pesos(7)*FP + pesos(8)*Farea
+pesos(9)*Foffset +pesos(10)*Fnoise +pesos(11)*Weakin)^2;
                if(~isempty(weakTrans))
                    sci = [FGM   FFGU   FCMRR  FPSRR FMF  FSL FP Farea
Foffset Fnoise Weakin];
                else
                    sci = [FGM   FFGU   FCMRR  FPSRR FMF  FSL FP Farea
Foffset Fnoise];
                end

                % salva a melhor solução atual
                if(Best.score > sc)
                    arq = fopen([circuito slash 'results' slash 'optimos.'
modo namext],'a+');
                    fprintf(arq,'%d  %.3g\n', cont, sc);
                    fclose(arq);
                    beep;
                    Best.score = sc;
                    fprintf('*> ');
                    arq = fopen([circuito slash 'paramop'],'w');
                    paramOpt(arq,xr,Vcminoffset,Vcmaxoffset,Vcmedioffset);
%gera um arquivo de simulação que pode ser usado pelo usuário
                    fprintf(arq,'\r\n*Score=%.2g  Ganho=%.2gdB  (FGM=%.2g)
Fo=%.2gMHz  (FFGU=%.2g) CMRR=%.2gdB (FCMRR=%.2g) PSRR=%.2gdB (FPSRR=%.2g)
MF=%.2g (FMF=%.2g) \n*Slew Rate=%.2gV/us (FSL=%.2g) Pot.=%.2guW (FP=%.2g)
Area=%.2gum2 (Farea=%.2g)\n*Offset=%.2gV (Foffset=%.2g) Noise=%.2guV
(Fnoise=%.2g) Weakin=%.2g \r\n',sc, GanhoMax, FGM,
FrequenciaGanhoUnitario*1e-6, FFGU, CMRRcircuit, FCMRR, PSRRcircuit,
FPSRR,MargemFase, FMF, SlewRatecircuit*1e-6,FSL,Potcircuit*1e6,FP,AreaMed,
Farea,offset,Foffset,CircuitNoise*1e6,Fnoise,Weakin);
                    fclose(arq);
                    % salva a melhor solução de todas
                    if(Best.scoreT > sc)
                        Best.scoreT = sc;
                        Best.parameters = xr;
                        arq = fopen([circuito slash 'paramopT'],'w');

paramOpt(arq,xr,Vcminoffset,Vcmaxoffset,Vcmedioffset); %gera um arquivo de
simulção que pode ser usado pelo usuário
                        fprintf(arq,'*Score=%.2g  Ganho=%.2gdB  (FGM=%.2g)
Fo=%.2gMHz  (FFGU=%.2g) CMRR=%.2gdB (FCMRR=%.2g) PSRR=%.2gdB (FPSRR=%.2g)
MF=%.2g (FMF=%.2g) \n*Slew Rate=%.2gV/us (FSL=%.2g) Pot.=%.2guW (FP=%.2g)
Area=%.2gum2 (Farea=%.2g)\n*Offset=%.2gV (Foffset=%.2g) Noise=%.2guV
(Fnoise=%.2g) Weakin=%.2g \r\n',sc, GanhoMax, FGM,
FrequenciaGanhoUnitario*1e-6, FFGU, CMRRcircuit, FCMRR, PSRRcircuit,
FPSRR,MargemFase, FMF, SlewRatecircuit*1e-6,FSL,Potcircuit*1e6,FP,AreaMed,
Farea,offset,Foffset,CircuitNoise*1e6,Fnoise,Weakin);
                        fclose(arq);
                        BestRes{modoind} = Best;
                    end;
                end

                % Resultados
                fprintf('Score=%.2g  Ganho=%.2gdB  (FGM=%.2g)  Fo=%.2gMHz
(FFGU=%.2g) CMRR=%.2gdB (FCMRR=%.2g) PSRR=%.2gdB (FPSRR=%.2g)\nMF=%.2g
(FMF=%.2g) Slew Rate=%.2gV/us (FSL=%.2g) Pot.=%.2guW (FP=%.2g) Area=%.2gum2
(Farea=%.2g)\nOffset=%.2gV (Foffset=%.2g) Noise=%.2guV (Fnoise=%.2g)
Weakin=%.2g \n',sc, GanhoMax, FGM, FrequenciaGanhoUnitario*1e-6, FFGU,
CMRRcircuit, FCMRR, PSRRcircuit, FPSRR,MargemFase, FMF, SlewRatecircuit*1e-
```

```
6,FSL,Potcircuit*1e6,FP,AreaMed,
Farea,offset,Foffset,CircuitNoise*1e6,Fnoise,Weakin);
                simsucess =1;
            end
        end
    end
end

% problemas na simulação
if (simsucess == 0)
    sc = inf('double');
    sci = [ inf('double')  inf('double')  inf('double')  inf('double')
inf('double')  inf('double') inf('double') inf('double') inf('double')
inf('double') inf('double')];
    fprintf('Score=%.3g  \n\n',sc);
end;

% parâmetros utilizados
for i = 1:length(xr)
    if (mod(i, 10) == 0) fprintf('\n');
    end
    fprintf('X%i= %1.1f   ', i, xr(i));
end;
fprintf('\n');

end
```

Função *paramEx.m*

```
function paramEx(arq,x)

global modelo;
global Vcmin;
global Vcmax;
global Vdd;
global Vss;
global const;
global weakTrans;
global slash;
global modelsD;

fprintf(arq, '.include ..%s..%s%s%s%s\r\n', slash, slash, modelsD, slash,
modelo);

% coloca os parâmetros
fprintf(arq,'.Param   ');
for i = 1:length(x)
    if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
    end
    fprintf(arq,'X%i= %1.3f  ', i, x(i));
end;
fprintf(arq,'\r\n');
fprintf(arq,'* ( ');
for i = 1:length(x)
    fprintf(arq,'%1.3f  ', x(i));
end;
fprintf(arq,')\r\n');
```

```
% coloca as constantes
nconst = length(const);
if (nconst > 0)
    fprintf(arq,'.Param   ');

    for i = 1:nconst
        if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
        end
        fprintf(arq,'M%i= %1.2f  ', i, const(i));
    end;
    fprintf(arq,'\r\n');
end;
fprintf(arq,'\r\n');

fprintf(arq, 'Vdd vd 0 DC %.2fV\r\n', Vdd);
fprintf(arq,'Vss vs 0 DC %.2fV\r\n',Vss);
fprintf(arq,'Vgnd gnd 0 DC %.2fV\r\n',(Vdd+Vss)/2);
fprintf(arq,'Vcm ip 0 DC %.3fV\r\n',Vcmin);
fprintf(arq,'Ex in ip  out aux 1000 \r\n');
fprintf(arq,'Vaux aux 0 DC %.3fV\r\n',(Vdd+Vss)/2);
fprintf(arq,'\r\n');

% comando para simulação
fprintf(arq,'.DC Vcm %gV %gV %gV\r\n',Vcmin,Vcmax+0.01, (Vcmax-Vcmin)/4);
%Tenho que fazer esse Vcmax + passo, porque, senão ele não pega o Vcmax

% comandos para medir tensões de offset e corrente
fprintf(arq,'.Meas DC tensaoINmax find V(in) when V(ip) =
%.2fV\r\n',Vcmax);
fprintf(arq,'.Meas DC tensaoINmin find V(in) when V(ip) =
%.2fV\r\n',Vcmin);
fprintf(arq,'.Meas DC tensaoINmedia find V(in) when V(ip) =
%.2fV\r\n',(Vcmax+Vcmin)/2);
fprintf(arq,'.Meas DC corrente max I(Vdd)\r\n');

% comando para avaliar fraca inversão
for i= 1: length(weakTrans);
    fprintf(arq, '.MEASURE DC %s_weak find
par(''((abs(Lx4(%s))/(26m*(1+Lx9(%s)/Lx7(%s))) - Lx7(%s))/Lx7(%s))'')  when
V(ip) = %1.2f\r\n', weakTrans{i,1 }, weakTrans{i,1 }, weakTrans{i,1 },
weakTrans{i,1}, weakTrans{i,1 }, weakTrans{i,1 }, (Vcmax+Vcmin)/2);
end;

end
```

Função *paramMC.m*

```
function paramganhoMC(arq,x,Vfeed)

global modelo;
global Vcmin;
global Vcmax;
global Vdd;
global Vss;
global const;
global CMRRparam;
global slash;
global modelsD;
```

```
fprintf(arq, '.include ..%s..%s%s%s%s\r\n', slash, slash, modelsD, slash,
modelo);

% coloca os parâmetros
fprintf(arq,'.Param   ');
for i = 1:length(x)
    if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
    end
    fprintf(arq,'X%i= %1.3f  ', i, x(i));
end;
fprintf(arq,'\r\n');
fprintf(arq,'** ( ');
for i = 1:length(x)
    fprintf(arq,'%1.3f  ', x(i));
end;
fprintf(arq,')\r\n');

% coloca as constantes
nconst = length(const);
if (nconst > 0)
    fprintf(arq,'.Param   ');

    for i = 1:nconst
        if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
        end
        fprintf(arq,'M%i= %1.2f  ', i, const(i));
    end;
    fprintf(arq,'\r\n');
end;
fprintf(arq,'\r\n');

fprintf(arq, 'Vdd vd 0 DC %.2fV\r\n', Vdd);
fprintf(arq,'Vss vs 0 DC %.2fV\r\n',Vss);
fprintf(arq,'Vgnd gnd 0 DC %.2fV\r\n',(Vdd+Vss)/2);
fprintf(arq,'V1 in a DC %.8f\r\n',Vfeed);
fprintf(arq,'V2 ip a DC %.2f\r\n',(Vcmax+Vcmin)/2);
fprintf(arq,'Vaux a  0 DC 0V AC 1V\r\n');
fprintf(arq,'\r\n');

% comando
fprintf(arq,'.AC DEC %.2f %.2f
%.2f\r\n',CMRRparam(3)/(10*CMRRparam(2)),CMRRparam(2),CMRRparam(3));

% comandos de medida do ganho de modo comum
fprintf(arq, '.meas AC maximo max Vdb(out)\r\n');

end
```

Função *paramAC.m*

```
function paramAC(arq,x,Vfeed1,Vfeed2,Vfeed3)

global modelo;
global const;
global Vdd;
global Vss;
global Vcmin;
```

```
global Vcmax;
global slash;
global Fo;
global modelsD;
global CMRRparam;

fprintf(arq, '.include ..%s..%s%s%s%s\r\n', slash, slash, modelsD, slash,
modelo);

% coloca os parâmetros
fprintf(arq,'.Param   ');
for i = 1:length(x)
    if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
    end
    fprintf(arq,'X%i= %1.3f  ', i, x(i));
end;
fprintf(arq,'\r\n');
fprintf(arq,'** ( ');
for i = 1:length(x)
    fprintf(arq,'%1.3f  ', x(i));
end;
fprintf(arq,')\r\n');

% coloca as constantes
nconst = length(const);
if (nconst > 0)
    fprintf(arq,'.Param   ');

    for i = 1:nconst
        if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
        end
        fprintf(arq,'M%i= %1.2f  ', i, const(i));
    end;
    fprintf(arq,'\r\n');
end;
fprintf(arq,'\r\n');
fprintf(arq,'.param cond = 1\r\n');
fprintf(arq, 'Vdd vd 0 DC %.2fV\r\n', Vdd);
fprintf(arq,'Vss vs 0 DC %.2fV\r\n',Vss);
fprintf(arq,'Vgnd gnd 0 DC %.2fV\r\n',(Vdd+Vss)/2);
fprintf(arq,'V1in in 0 DC %.8fV \r\n',Vfeed1);
fprintf(arq,'V2in ip 0 DC %.2fV AC 1V \r\n',Vcmin);
fprintf(arq,'\r\n');

% comandos de execução
fprintf(arq,'.AC DEC 40 %.2f %.2fMeg\r\n',CMRRparam(2),4*Fo*1e-6);
fprintf(arq,'.noise V(out) V2in  1\r\n');
fprintf(arq,'\r\n');

% comando de medida
fprintf(arq,'.meas AC maximo max Vdb(out)\r\n'); % faz a medida do ganho de
malha aberta
fprintf(arq,'.if (cond ==1)\r\n');
fprintf(arq,'.meas AC noise_integ RMS onoise\r\n'); %faz a media do ruido
fprintf(arq,'.meas AC ruidoTotal Param = ''noise_integ*sqrt(%.2fMeg-
%.2f)''\r\n',4*Fo*1e-6,CMRRparam(2)); %multiplica pela raiz da faixa de
frequência
fprintf(arq,'.meas AC MF find vp(out) when vdb(out)=0\r\n'); % faz a medida
da fase de ganho unitário
```

```
fprintf(arq,'.meas AC freq when vdb(out)=0\r\n'); % faz a medida da
frequência de ganho unitário
fprintf(arq,'.endif\r\n');
fprintf(arq,'\r\n');

fprintf(arq,'.ALTER\r\n');
fprintf(arq,'V1in in 0 DC %.8fV\r\n',Vfeed2);
fprintf(arq,'V2in ip 0 DC %.2fV AC 1V\r\n',Vcmax);
fprintf(arq,'\r\n');

% Calculo de PSRR para a fonte de alimentação
fprintf(arq,'.ALTER\r\n');
fprintf(arq,'.param cond = 0\r\n');
fprintf(arq,'Vdd vd 0 DC %.2fV AC 1V\r\n', Vdd);
fprintf(arq,'V1in in 0 DC %.8fV\r\n',Vfeed3);
fprintf(arq,'V2in ip 0 DC %.2fV\r\n',(Vcmax+Vcmin)/2);
fprintf(arq,'\r\n');

% Calculo de PSRR para o terra
fprintf(arq,'.ALTER\r\n');
fprintf(arq, '.param cond = 0 \r\n');
fprintf(arq,'Vss vs 0 DC %.2fV AC 1V\r\n',Vss);
fprintf(arq,'V1in in 0 DC %.8fV\r\n',Vfeed3);
fprintf(arq,'V2in ip 0 DC %.2fV\r\n',(Vcmax+Vcmin)/2);

end
```

Função *paramSlewRate.m*

```
function paramSlewRate(arq,x, Vof)

global modelo;
global SlewRate;
global Vdd;
global Vss;
global Vcmin;
global Vcmax;
global const;
global slash;
global modelsD;

fprintf(arq, '.include ..%s..%s%s%s%s\r\n', slash, slash, modelsD, slash,
modelo);

% coloca os parâmetros
fprintf(arq,'.Param   ');
for i = 1:length(x)
    if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
    end
    fprintf(arq,'X%i= %1.3f  ', i, x(i));
end;
fprintf(arq,'\r\n');
fprintf(arq,'* ( ');
for i = 1:length(x)
    fprintf(arq,'%1.3f  ', x(i));
end;
fprintf(arq,')\r\n');
```

```
% coloca as constantes
nconst = length(const);
if (nconst > 0)
    fprintf(arq,'.Param   ');

    for i = 1:nconst
        if (mod(i, 7) == 0) fprintf(arq,'\r\n.Param   ');
        end
        fprintf(arq,'M%i= %1.2f  ', i, const(i));
    end;
    fprintf(arq,'\r\n');
end;
fprintf(arq,'\r\n');

%SL = dV/dt
%a variação que ocorrerá será de Vdd/2 V. Portanto, o tempo de analise
%será de (Vdd/2)/SlewRate, teoricamente
t = ((Vdd+Vss)/2)/(SlewRate*1e-6); %tempo encontrado em s

fprintf(arq, 'Vdd vd 0 DC %.2fV\r\n', Vdd);
fprintf(arq,'Vss vs 0 DC %.2fV\r\n',Vss);
fprintf(arq,'Vgnd gnd 0 DC %.2fV\r\n',(Vdd+Vss)/2);
fprintf(arq,'Vpulse in 0 PULSE (%.8f %.2f 10ns 1ns 1ns %.2fus %.2fms)\r\n',
Vof, Vdd, 20*t, 30*t);
fprintf(arq,'Vip ip 0 %.2f \r\n', (Vcmax+Vcmin)/2);
fprintf(arq,'\r\n');

% comando para simulação
fprintf(arq,'.TRAN 10ns %.2fus 0ns %.3fus\r\n',20*t,t/50);
fprintf(arq,'\r\n');

% comandos para medidas
fprintf(arq,'.MEAS TRAN maximo max V(out)\r\n');
fprintf(arq,'.MEAS TRAN minimo min V(out)\r\n');
fprintf(arq,'.MEAS TRAN slewrate DERIV V(out) when
V(out)=''(maximo+minimo)/2'' \r\n');

%nova simulação
fprintf(arq,'.ALTER\r\n');
fprintf(arq,'Vpulse in 0 PULSE (%.8f %.2f 10ns 1ns 1ns %.2fus %.2fms)\r\n',
Vof, Vss, 20*t, 30*t);
fprintf(arq,'.TRAN 10ns %.2fus 0ns %.3fus\r\n',20*t,t/50);

end
```

APÊNDICE C

Descrição em Linguagem SPICE da Topologia DoisEstagios

******Twostage Amplifier**********
*Neste código é possível perceber a presença de uma variável chamada de Wpadrão, presente no arquivo de descrição do modelo do circuito, o qual se *refere ao comprimento mínimo de construção para o dreno ou source de um transistor da fonte que tecnologia AMS 0,35µm. Esse valor é *usado para cálculos das áreas de dreno/source e perímetros de dreno/source.
M1 1 in 4 vs MODN L = 'X1*1u' W = 'X5*1u' AD = 'X5*1u*Wpadrao/2' PD = 'X5*1u + Wpadrao' AS = 'X5*1u*Wpadrao/2'
+PS ='X5*1u + Wpadrao'
M2 3 ip 4 vs MODN L = 'X1*1u' W = 'X5*1u' AD = 'X5*1u*Wpadrao/2' PD = 'X5*1u + Wpadrao' AS = 'X5*1u*Wpadrao/2'
+PS = 'X5*1u + Wpadrao'
M3 1 1 vd vd MODP L = 'X2*1u' W = 'X6*1u' AD = 'X6*1u*Wpadrao/2' PD = 'X6*1u + Wpadrao' AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao'
M4 3 1 vd vd MODP L = 'X2*1u' W = 'X6*1u' AD = 'X6*1u*Wpadrao/2' PD = 'X6*1u + Wpadrao' AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao'
M5 4 bias vs vs MODN L = 'X3*1u' W = 'X7*1u' AD = 'X7*1u*Wpadrao/2' PD = 'X7*1u + Wpadrao' AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao' M = 'round(M1)'
M6 out 3 vd vd MODP L = 'X2*1u' W = 'X6*1u' AD = 'X6*1u*Wpadrao/2' PD = 'X6*1u + Wpadrao' AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao' M = '2*round(X8)/round(M1)'
M7 out bias vs vs MODN L = 'X3*1u' W = 'X7*1u' AD = 'X7*1u*Wpadrao/2' PD = 'X7*1u + Wpadrao' AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao' M= 'round(X8)'
M8 3 vs 5 vd MODP L = 'X4*1u' W = 'X9*1u' AD = 'X9*1u*Wpadrao/2' PD = 'X9*1u + Wpadrao' AS = 'X9*1u*Wpadrao/2'
+PS = 'X9*1u + Wpadrao'
Mbn bias bias vs vs MODN L = 'X3*1u' W = 'X7*1u' AD = 'X7*1u*Wpadrao/2' PD = 'X7*1u + Wpadrao' AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao'
Cc 5 out 'X10*M2*1p'
Cl out gnd 'M2*1p'
Ibb vd bias DC '(10^X11)*1u'
.include param
.end

Descrição em Linguagem SPICE da Topologia ClassAB

******CMOS Amplifier with class AB output stage**********
*Neste código é possível perceber a presença de uma variável chamada de Wpadrão, presente no arquivo de descrição do modelo do circuito, o qual se *refere ao comprimento mínimo de construção para o dreno ou source de um transistor da fonte que tecnologia AMS 0,35µm. Esse valor é *usado para cálculos das áreas de dreno/source e perímetros de dreno/source.
M1 1 in 2 vs MODN L = 'X1*1u' W = 'X6*1u' AD = 'X6*1u*Wpadrao/2' PD = 'X6*1u + Wpadrao' AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao'
M2 3 ip 2 vs MODN L = 'X1*1u' W = 'X6*1u' AD = 'X6*1u*Wpadrao/2' PD = 'X6*1u + Wpadrao' AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao'
M3 1 1 vd vd MODP L = 'X2*1u' W = 'X7*1u' AD = 'X7*1u*Wpadrao/2' PD = 'X7*1u + Wpadrao' AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao' M = 'round(X11)'
M4 3 1 vd vd MODP L = 'X2*1u' W = 'X7*1u' AD = 'X7*1u*Wpadrao/2' PD = 'X7*1u + Wpadrao' AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao' M = 'round(X11)'
M5 2 4 vs vs MODN L = 'X3*1u' W = 'X8*1u' AD = 'X8*1u*Wpadrao/2' PD = 'X8*1u + Wpadrao' AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao' M = '2*round(X11)'
M6 out 3 vd vd MODP L = 'X2*1u' W = 'X7*1u' AD = 'X7*1u*Wpadrao/2' PD = 'X7*1u + Wpadrao' AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao' M = 'round(X13)'
M7 out 6 vs vs MODN L = 'X3*1u' W = 'X8*1u' AD = 'X8*1u*Wpadrao/2' PD = 'X8*1u + Wpadrao' AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao' M = 'round(X13)'
M8 3 vs 7 vd MODP L = 'X4*1u' W = 'X9*1u' AD = 'X9*1u*Wpadrao/2' PD = 'X9*1u + Wpadrao' AS = 'X9*1u*Wpadrao/2'
+PS = 'X9*1u + Wpadrao'
M9 vd 3 6 vs MODN L = 'X5*1u' W = 'X10*1u' AD = 'X10*1u*Wpadrao/2' PD = 'X10*1u + Wpadrao' AS = 'X10*1u*Wpadrao/2'
+PS = 'X10*1u + Wpadrao' M = 'round(X12)'
M10 6 4 vs vs MODN L = 'X3*1u' W = 'X8*1u' AD = 'X8*1u*Wpadrao/2' PD = 'X8*1u + Wpadrao' AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao' M = 'round(X12)'

```
M11 5 5 vd vd MODP   L = 'X2*1u'   W = 'X7*1u'   AD = 'X7*1u*Wpadrao/2'   PD = 'X7*1u + Wpadrao'   AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao'
M12 5 5 4 vs MODN   L = 'X5*1u'   W = 'X10*1u'   AD = 'X10*1u*Wpadrao/2'   PD = 'X10*1u + Wpadrao'   AS = 'X10*1u*Wpadrao/2'
+PS = 'X10*1u + Wpadrao'
M13 4 4 vs vs MODN   L = 'X3*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao'
Cc 7 out 'X14*M1*1p'
Cl out gnd 'M1*1p'
.include param
.end
```

Descrição em Linguagem SPICE da Topologia OTA

```
******Operational transcondutance amplifier**********
*Neste código é possível perceber a presença de uma variável chamada de Wpadrão, presente no arquivo de descrição do modelo do
circuito, o qual se *refere ao comprimento mínimo de construção para o dreno ou source de um transistor da fonte que tecnologia AMS
0,35µm. Esse valor é *usado para cálculos das áreas de dreno/source e perímetros de dreno/source.
M1 7 ip 8 vs MODN   L = 'X1*1u'  W = 'X5*1u'   AD = 'X5*1u*Wpadrao/2'   PD = 'X5*1u + Wpadrao'   AS = 'X5*1u*Wpadrao/2'
+PS = 'X5*1u + Wpadrao'
M2 2 in 8 vs MODN   L = 'X1*1u'  W = 'X5*1u'   AD = 'X5*1u*Wpadrao/2'   PD = 'X5*1u + Wpadrao'   AS = 'X5*1u*Wpadrao/2'
+PS = 'X5*1u + Wpadrao'
M3 3 4 vs vs MODN   L = 'X2*1u' W = 'X6*1u'   AD = 'X6*1u*Wpadrao/2'   PD = 'X6*1u + Wpadrao'   AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao'
M4 4 4 vs vs MODN   L = 'X2*1u' W = 'X6*1u'   AD = 'X6*1u*Wpadrao/2'   PD = 'X6*1u + Wpadrao'   AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao'
M5 out 3 4 vs MODN   L = 'X2*1u' W = 'X6*1u'   AD = 'X6*1u*Wpadrao/2'   PD = 'X6*1u + Wpadrao'   AS = 'X6*1u*Wpadrao/2'
+PS = 'X6*1u + Wpadrao'
MNi 8 Vgn vs vs MODN   L = 'X3*1u'   W = 'X7*1u'   AD = 'X7*1u*Wpadrao/2'   PD = 'X7*1u + Wpadrao'   AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao'   M = 'round(M1)'
MNi2 Vgn Vgn vs vs MODN   L = 'X3*1u'   W = 'X7*1u'   AD = 'X7*1u*Wpadrao/2'   PD = 'X7*1u + Wpadrao'   AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao'
M6 vd 6 7 vd MODP   L = 'X4*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'   PS
= 'X8*1u + Wpadrao'
M7 vd 1 2 vd MODP   L = 'X4*1u' W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'   PS
= 'X8*1u + Wpadrao'
M8 vd 6 6 vd MODP   L = 'X4*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao'   M = Mx
M9 vd 1 1 vd MODP   L = 'X4*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'   PS
= 'X8*1u + Wpadrao'   M = Mx
M10 6 7 out vd MODP   L = 'X4*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'   PS
= 'X8*1u + Wpadrao'   M = Mx
M11 1 2 3 vd MODP   L = 'X4*1u'W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'   PS
= 'X8*1u + Wpadrao'   M = Mx
Cl out gnd 'M2*1p'
Ibb vd Vgn DC '(10^X9)*1u'
.param Mx = 'round(X10)'
.include param
.end
```

Descrição em Linguagem SPICE da Topologia FoldedCascode

```
******Folded-Cascode Amplifier**********
M1 2 ip 5 vs MODN   L = 'X1*1u'   W = 'X7*1u'   AD = 'X7*1u*Wpadrao/2'   PD = 'X7*1u + Wpadrao'   AS = 'X7*1u*Wpadrao/2'   +PS
= 'X7*1u + Wpadrao'
M2 3 in 5 vs MODN   L = 'X1*1u'   W = 'X7*1u'   AD = 'X7*1u*Wpadrao/2'   PD = 'X7*1u + Wpadrao'   AS = 'X7*1u*Wpadrao/2'
+PS = 'X7*1u + Wpadrao'
M3 vd 1 2 vd MODP   L = 'X2*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'   +PS
= 'X8*1u + Wpadrao'   M = 'round(M1)'
```

```
M4 vd 1 3 vd MODP   L = 'X2*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao'   M = 'round(M1)'
M5 5 7 vs vs MODN   L = 'X3*1u'   W = 'X9*1u'   AD = 'X9*1u*Wpadrao/2'   PD = 'X9*1u + Wpadrao'   AS = 'X9*1u*Wpadrao/2'
+PS = 'X9*1u + Wpadrao'   M = 'round(M1)'
M6 10 4 vs vs MODN   L = 'X4*1u'   W = 'X10*1u'   AD = 'X10*1u*Wpadrao/2'   PD = 'X10*1u + Wpadrao'   AS = 'X10*1u*Wpadrao/2'
+PS = 'X10*1u + Wpadrao'
M7 8 4 vs vs MODN   L = 'X4*1u'   W = 'X10*1u'   AD = 'X10*1u*Wpadrao/2'   PD = 'X10*1u + Wpadrao'   AS = 'X10*1u*Wpadrao/2'
+PS = 'X10*1u + Wpadrao'
M8 out Vp1 10 vs MODN   L = 'X5*1u'   W = 'X11*1u'   AD = 'X11*1u*Wpadrao/2'   PD = 'X11*1u + Wpadrao'   AS = 'X11*1u*Wpadrao/2'
+PS = 'X11*1u + Wpadrao'
M9 4 Vp1 8 vs MODN   L = 'X5*1u'   W = 'X11*1u'   AD = 'X11*1u*Wpadrao/2'   PD = 'X11*1u + Wpadrao'   AS = 'X11*1u*Wpadrao/2'
+PS = 'X11*1u + Wpadrao'
M10 3 Vp2 out vd MODP   L = 'X6*1u'   W = 'X12*1u'   AD = 'X12*1u*Wpadrao/2'   PD = 'X12*1u + Wpadrao'   AS= 'X12*1u*Wpadrao/2'
+PS = 'X12*1u + Wpadrao'
M11 2 Vp2 4 vd MODP   L = 'X6*1u'   W = 'X12*1u'   AD = 'X12*1u*Wpadrao/2'   PD = 'X12*1u + Wpadrao'   AS= 'X12*1u*Wpadrao/2'
+PS = 'X12*1u + Wpadrao'
M12 Vp1 Vp1 vs vs MODN   L = 'X13*1u'   W = 'X14*1u'   AD = 'X14*1u*Wpadrao/2'   PD = 'X14*1u + Wpadrao'   As = 'X14*1u*Wpadrao/2'
+PS = 'X14*1u + Wpadrao'
M13 Vp2 7 vs vs MODN   L = 'X3*1u'   W = 'X9*1u'   AD = 'X9*1u*Wpadrao/2'   PD = 'X9*1u + Wpadrao'   AS = 'X9*1u*Wpadrao/2'
+PS = 'X9*1u + Wpadrao'
M14 vd Vp2 Vp2 vd MODP   L = 'X15*1u'   W = 'X16*1u'   AD = 'X16*1u*Wpadrao/2'   PD = 'X16*1u + Wpadrao'   AS = 'X16*1u*Wpadrao/2'
+PS = 'X16*1u + Wpadrao'
M15 vd 1 Vp1 vd MODP   L = 'X2*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao'
Mbn 7 7 vs vs MODN   L = 'X3*1u'   W = 'X9*1u'   AD = 'X9*1u*Wpadrao/2'   PD = 'X9*1u + Wpadrao'   AS = 'X9*1u*Wpadrao/2'
+PS = 'X9*1u + Wpadrao'
Mbp vd 1 1 vd MODP   L = 'X2*1u'   W = 'X8*1u'   AD = 'X8*1u*Wpadrao/2'   PD = 'X8*1u + Wpadrao'   AS = 'X8*1u*Wpadrao/2'
+PS = 'X8*1u + Wpadrao'

Cl out gnd 'M2*1p'
Ibb 1 7 DC '10^X17*1u'

.include param
                .end
```

ANEXOS

Descrição em Linguagem SPICE do Modelo da Tecnologia AMS 0,35µm

```
.param Wpadrao = 1.7u
* ---------------------------------------------------------------------
.MODEL MODN NMOS LEVEL=53 MODTYPE=ELDO
* ---------------------------------------------------------------------
************************ SIMULATION PARAMETERS ************************
* ---------------------------------------------------------------------
* format    : ELDO, AccusimII, Continuum
* model     : MOS BSIM3v3
* process   : C35
* revision : 4.0;
* extracted : B10866 ; 2002-12; ese(5487)
* doc#      : ENG-182 REV_6
* ---------------------------------------------------------------------
*                    TYPICAL MEAN CONDITION
* ---------------------------------------------------------------------
*
+THMLEV =0
*       *** Flags ***
+NOIMOD =3 FLKLEV =0
```

+MOBMOD =1.000e+00 CAPMOD =2.000e+00 VERSION=3.240e+00 NQSMOD =0.000e+00

+DERIV =1

* *** Threshold voltage related model parameters ***

+K1 =5.0296e-01

+K2 =3.3985e-02 K3 =-1.136e+00 K3B =-4.399e-01

+NPEAK =2.611e+17 VTH0 =4.979e-01

+VOFF =-8.925e-02 DVT0 =5.000e+01 DVT1 =1.039e+00

+DVT2 =-8.375e-03 KETA =2.032e-02

+PSCBE1 =1.000e+30 PSCBE2 =1.000e-06

+DVT0W =1.089e-01 DVT1W =6.671e+04 DVT2W =-1.352e-02

* *** Mobility related model parameters ***

+UA =4.705e-12 UB =2.137e-18 UC =1.000e-20

+U0 =4.758e+02

* *** Subthreshold related parameters ***

+DSUB =5.000e-01 ETA0 =1.415e-02 ETAB =-1.221e-01

+NFACTOR=4.136e-01

* *** Saturation related parameters ***

+EM =4.100e+07 PCLM =6.948e-01

+PDIBLC1=3.571e-01 PDIBLC2=2.065e-03 DROUT =5.000e-01

+A0 =2.541e+00 A1 =0.000e+00 A2 =1.000e+00

+PVAG =0.000e+00 VSAT =1.338e+05 AGS =2.408e-01

+B0 =4.301e-09 B1 =0.000e+00 DELTA =1.442e-02

+PDIBLCB=3.222e-01

* *** Geometry modulation related parameters ***

+W0 =2.673e-07 DLC =3.0000e-08

+DWC =9.403e-08 DWB =0.000e+00 DWG =0.000e+00

+LL =0.000e+00 LW =0.000e+00 LWL =0.000e+00

+LLN =1.000e+00 LWN =1.000e+00 WL =0.000e+00

+WW =-1.297e-14 WWL =-9.411e-21 WLN =1.000e+00

+WWN =1.000e+00

* *** Temperature effect parameters ***

+AT =3.300e+04 UTE =-1.800e+00

+KT1 =-3.302e-01 KT2 =2.200e-02 KT1L =0.000e+00

+UA1 =0.000e+00 UB1 =0.000e+00 UC1 =0.000e+00

+PRT =0.000e+00

* *** Overlap capacitance related and dynamic model parameters ***

+CGSO =1.200e-10 CGDO =1.200e-10 CGBO =1.100e-10

+CGDL =1.310e-10 CGSL =1.310e-10 CKAPPA =6.000e-01

+CF =0.000e+00 ELM =5.000e+00

+XPART =1.000e+00 CLC =1.000e-15 CLE =6.000e-01

+NOFF =1.000e+00 VOFFCV =0.000e+00

* *** Parasitic resistance and capacitance related model parameters ***

+RDSW =3.449e+02

+CDSC =0.000e+00 CDSCB =1.500e-03 CDSCD =1.000e-03

+PRWB =-2.416e-01 PRWG =0.000e+00 CIT =4.441e-04

* *** Process and parameters extraction related model parameters ***

+TOX =7.575e-09 NGATE =0.000e+00

```
+NLX    =1.888e-07
+XL     =0.000e+00 XW     =0.000e+00
*       *** Substrate current related model parameters ***
+ALPHA0 =2.600e-06 ALPHA1 =5.000e+00 BETA0  =2.100e+01
*       *** Noise effect related model parameters ***
+AF     =1.507e+00 KF     =2.170e-26 EF     =1.000e+00
+NOIA   =1.121e+19 NOIB   =5.336e+04 NOIC   =-5.892e-13
*       *** Common extrinsic model parameters ***
+ALEV   =2         RLEV   =2
+RD     =0.000e+00 RS     =0.000e+00 RSH    =7.000e+01
+RDC    =0.000e+00 RSC    =0.000e+00 LD     =-5.005e-08
+WD     =9.403e-08
+LDIF   =0.000e+00 HDIF   =8.000e-07 WMLT   =1.000e+00
+LMLT   =1.000e+00 DEL    =0.000e+00 XJ     =3.000e-07
+DIOLEV =4         JS     =5.100e-07 JSW    =0.600e-12
+IS     =0.000e+00 N      =1.000e+00
+DCAPLEV=2         CBD    =0.000e+00 CBS    =0.000e+00
+CJ     =8.400e-04 CJSW   =2.500e-10 FC     =0.000e+00
+MJ     =3.400e-01 MJSW   =2.300e-01 TT     =0.000e+00
+XTI    =2.026e+00 PB     =6.900e-01 PBSW   =6.900e-01
* ---------------------------------------------------------------------
.MODEL MODP PMOS LEVEL=53 MODTYPE=ELDO
* ---------------------------------------------------------------------
************************* SIMULATION PARAMETERS ************************
* ---------------------------------------------------------------------
* format    : ELDO, AccusimII, Continuum
* model     : MOS BSIM3v3
* process   : C35
* revision : 4.1;
* extracted : C64685 ; 2002-12; ese(5487)
* doc#      : ENG-182 REV_6
* ---------------------------------------------------------------------
*                    TYPICAL MEAN CONDITION
* ---------------------------------------------------------------------
*
+THMLEV =0
*       *** Flags ***
+NOIMOD =3 FLKLEV =0
+MOBMOD =1.000e+00 CAPMOD =2.000e+00 VERSION=3.24e+00 NQSMOD =0.000e+00
+DERIV  =1
*       *** Threshold voltage related model parameters ***
+K1     =5.9959e-01
+K2     =-6.038e-02 K3     =1.103e+01 K3B    =-7.580e-01
+NPEAK  =9.240e+16 VTH0   =-6.915e-01
+VOFF   =-1.170e-01 DVT0   =1.650e+00 DVT1   =3.868e-01
+DVT2   =1.659e-02 KETA   =-1.440e-02
+PSCBE1 =1.000e+30 PSCBE2 =1.000e-06
```

```
+DVT0W  =1.879e-01 DVT1W  =7.335e+04 DVT2W  =-6.312e-03
*       *** Mobility related model parameters ***
+UA     =5.394e-10 UB     =1.053e-18 UC     =1.000e-20
+U0     =1.482e+02
*       *** Subthreshold related parameters ***
+DSUB   =5.000e-01 ETA0   =2.480e-01 ETAB   =-3.917e-03
+NFACTOR=1.214e+00
*       *** Saturation related parameters ***
+EM     =4.100e+07 PCLM   =3.184e+00
+PDIBLC1=1.000e-04 PDIBLC2=1.000e-20 DROUT  =5.000e-01
+A0     =5.850e-01 A1     =0.000e+00 A2     =1.000e+00
+PVAG   =0.000e+00 VSAT   =1.158e+05 AGS    =2.468e-01
+B0     =8.832e-08 B1     =0.000e+00 DELTA  =1.000e-02
+PDIBLCB=1.000e+00
*       *** Geometry modulation related parameters ***
+W0     =1.000e-10 DLC    =2.4500e-08
+DWC    =3.449e-08 DWB    =0.000e+00 DWG    =0.000e+00
+LL     =0.000e+00 LW     =0.000e+00 LWL    =0.000e+00
+LLN    =1.000e+00 LWN    =1.000e+00 WL     =0.000e+00
+WW     =1.894e-16 WWL    =-1.981e-21 WLN   =1.000e+00
+WWN    =1.040e+00
*       *** Temperature effect parameters ***
+AT     =3.300e+04 UTE    =-1.300e+00
+KT1    =-5.403e-01 KT2   =2.200e-02 KT1L   =0.000e+00
+UA1    =0.000e+00 UB1    =0.000e+00 UC1    =0.000e+00
+PRT    =0.000e+00
*       *** Overlap capacitance related and dynamic model parameters   ***
+CGSO   =8.600e-11 CGDO   =8.600e-11 CGBO   =1.100e-10
+CGDL   =1.080e-10 CGSL   =1.080e-10 CKAPPA =6.000e-01
+CF     =0.000e+00 ELM    =5.000e+00
+XPART  =1.000e+00 CLC    =1.000e-15 CLE    =6.000e-01
+NOFF   =1.000e+00 VOFFCV =0.000e+00
*       *** Parasitic resistance and capacitance related model parameters ***
+RDSW   =1.033e+03
+CDSC   =2.589e-03 CDSCB  =2.943e-04 CDSCD  =4.370e-04
+PRWB   =-9.731e-02 PRWG   =1.477e-01 CIT    =0.000e+00
*       *** Process and parameters extraction related model parameters ***
+TOX    =7.754e-09 NGATE  =0.000e+00
+NLX    =1.770e-07
+XL     =0.000e+00 XW     =0.000e+00
*       *** Substrate current related model parameters ***
+ALPHA0 =1.000e-09 ALPHA1 =1.500e+00 BETA0  =3.250e+01
*       *** Noise effect related model parameters ***
+AF     =1.461e+00 KF     =1.191e-26 EF     =1.000e+00
+NOIA   =5.245e+17 NOIB   =4.816e+03 NOIC   =8.036e-13
*       *** Common extrinsic model parameters ***
+ALEV   =2        RLEV   =2
```

+RD =0.000e+00 RS =0.000e+00 RSH =1.290e+02

+RDC =0.000e+00 RSC =0.000e+00 LD =-7.130e-08

+WD =3.449e-08

+LDIF =0.000e+00 HDIF =8.000e-07 WMLT =1.000e+00

+LMLT =1.000e+00 DEL =0.000e+00 XJ =3.000e-07

+DIOLEV =4 JS =2.800e-07 JSW =3.700e-13

+IS =0.000e+00 N =1.000e+00

+DCAPLEV=2 CBD =0.000e+00 CBS =0.000e+00

+CJ =1.360e-03 CJSW =3.500e-10 FC =0.000e+00

+MJ =5.400e-01 MJSW =4.600e-01 TT =0.000e+00

+XTI =1.973e+00 PB =1.020e+00 PBSW =1.020e+00

* ---

* ---

* Owner: austriamicrosystems