Universidade Federal do Pampa
Curso de Bacharelado em Engenharia Elétrica

Uma Ferramenta de Auxílio ao Projeto de Circuitos Integrados Analógicos Integrada ao Ambiente Cadence Virtuoso

Rodrigo da Silva Moraes

Alegrete - RS, 2023

Rodrigo da Silva Moraes

Uma Ferramenta de Auxílio ao Projeto de Circuitos Integrados Analógicos Integrada ao Ambiente Cadence Virtuoso

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia Elétrica como requisito parcial para a obtenção do grau de Bacharel em Engenharia Elétrica.

Universidade Federal do Pampa

Orientador: Prof. Dr. Lucas Compassi Severo

Alegrete - RS

2023

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

M827f Moraes, Rodrigo
Uma ferramenta de auxílio ao projeto de circuitos integrados analógicos integrada ao ambiente Cadence Virtuoso / Rodrigo Moraes.
83 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA ELÉTRICA, 2023.
"Orientação: Lucas Severo".

1. Microeletrônica. 2. Dimensionamento de transistores. 3. Circuitos Integrados Analógicos. 4. Python. 5. Cadence SKILL. I. Título.

RODRIGO DA SILVA MORAES

UMA FERRAMENTA DE AUXÍLIO AO PROJETO DE CIRCUITOS INTEGRADOS ANALÓGICOS INTEGRADA AO AMBIENTE CADENCE VIRTUOSO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia Elétrica.

Dissertação defendida e aprovada em: 06 de julho de 2023.

Banca examinadora:

_____

Prof. Dr. Lucas Compassi Severo

Orientador

UNIPAMPA

_____

Prof. Dr. Alessandro Gonçalves Girardi

UNIPAMPA

_____

Prof. Dr. Paulo César Comassetto de Aguirre

UNIPAMPA

Assinado eletronicamente por **ALESSANDRO GONCALVES GIRARDI**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 06/07/2023, às 17:06, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **LUCAS COMPASSI SEVERO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 06/07/2023, às 17:08, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **PAULO CESAR COMASSETTO DE AGUIRRE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 06/07/2023, às 17:10, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1167598** e o código CRC **E8C09DC3**.

Agradecimentos

Agradeço a Deus, aos meus pais e minha família, que me apoiaram durante os anos da graduação.
Ao meu orientador, Prof. Lucas Severo, pela orientação desde os projetos de iniciação científica que foram de grande influência para minha aprendizagem, por todos os conselhos, pela paciência e ajuda nesse período.

Aos meus amigos Luan e Jhosepher, que me acompanharam desde o início desta trajetória, e se tornaram grandes amigos.

À minha querida Suzian, que esteve comigo nesta etapa importante em minha vida.

Resumo

Este trabalho apresenta uma ferramenta para o dimensionamento de transistores CMOS baseada na polarização e simulação elétrica, integrada ao ambiente de projetos Cadence Virtuoso ®, a partir do uso das linguagens de programação Python e Cadence SKILL, e do simulador Cadence Spectre ®. Visando o auxílio de projetos de circuitos integrados analógicos, a ferramenta possui fluxo de projeto customizável utilizando funções básicas, e é compatível com as tecnologias TSMC de 180 nm e 65 nm.
A metodologia utilizada no desenvolvimento desta ferramenta é descrita, e como resultado, é demonstrada a aplicação do software desenvolvido no projeto de quatro topologias básicas de circuitos integrados analógicos.

Palavras-chaves: Microeletrônica. Dimensionamento de transistores. Python. Circuitos Integrados Analógicos, Cadence SKILL.

Abstract

This work presents a CMOS transistor sizing tool based on biasing and electrical simulation, integrated in the Cadence Virtuoso ®design environment, using the Python and Cadence SKILL programming languages, and the Cadence Spectre® electrical simulator. Aiming to aid analog integrated circuits projects, the sizing tool has a customizable project flow using basic functions, and it is compatible with the TSMC 180 nm and 65 nm technology. It is described the methodology used in the development of this tool, and as results, it is shown the application of the developed software in the project of four basic analog integrated circuit topologies.

Keywords: Microelectronics. Transistor sizing tool. Python. Analog integrated circuits, Cadence SKILL.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CMOS	Metal-Óxido Semicondutor Complementar
MOSFET	Transistor de Efeito de Campo Metal-Óxido Semicondutor Complementar
V_{GS}	Tensão entre os terminais gate e source
V_{DS}	Tensão entre os terminais drain e source
V_{BS}	Tensão entre os terminais bulk e source
V_{DD}	Tensão no terminal dreno
V_{TH}	Tensão de limiar (Threshold)
V_{OV}	Tensão de overdrive
CI	Circuito Integrado
CAD	Computer Aided Design
GBW	Gain-bandwidth product

Sumário

1 Introdução

Transistores são dispositivos semicondutores usados em circuitos eletrônicos, tendo como funções principais amplificar e chavear sinais elétricos. Estes dispositivos, considerados como uma das maiores invenções do século 20, revolucionaram a eletrônica, permitindo a miniaturização de circuitos e um avanço tecnológico significativo em todo mundo. Em 1956, os físicos americanos John Bardeen, Walter Brattain e William Shockley receberam o prêmio Nobel da física pela "*Pesquisa com Semicondutores e a descoberta do Efeito transistor*"(SHOCKLEY; BARDEEN; BRATTAIN, 1956). Na microeletrônica, transistores são componentes indispensáveis em projetos de circuitos integrados, em especial os MOSFETs.

Devido ao seu baixo custo e boa performance, a tecnologia CMOS é muito utilizada na indústria, e se baseia no uso de pares de MOSFETs do tipo N e P para realizar funções lógicas, podendo ser usados em processadores, microcontroladores, memórias, entre outros. Em circuitos integrados analógicos de rádio-frequência, transistores são úteis em projetos de amplificadores de potência, amplificadores de baixo ruído, osciladores controlados por tensão, etc.

A demanda por circuitos integrados com cada vez mais transistores para realizar um número maior de funções causou a redução progressiva na tecnologia CMOS (ABBAS; OLIVIERI, 2016), tendo dimensões mínimas que passaram de dezenas de micrômetros para unidades de nanômetros em menos de 50 anos. Gordon Moore, cofundador da Intel Corp., fez uma observação em 1965 de que o número de componentes em um circuito integrado dobra a cada ano (MOORE, 2006).

As dimensões e tensões de operação alteram as características elétricas dos transistores, e consequentemente as especificações dos circuitos analógicos. Os parâmetros de dimensionamento destes transistores dependem principalmente da largura (W) e do comprimento (L) do canal, essas dimensões afetam diretamente as regiões de operação do transistor, que controlam o fluxo de corrente elétrica que passa pelo mesmo. Desta forma, é necessário obter o dimensionamento adequado para que o circuito alcance as especificações do projeto de forma precisa.

Ao contrário de circuitos digitais, circuitos integrados analógicos exploram a física do processo de fabricação, manipulando quantidades precisas como tensões, correntes, cargas, e proporções contínuas de parâmetros como resistência e capacitância. Devido a isto, efeitos que não são considerados críticos em circuitos digitais tornam-se problemas no projeto de circuitos integrados analógicos (LEENAERTS; GIELEN; RUTENBAR, 2001).

Para circuitos digitais, existem diversas ferramentas para a automação do design, enquanto circuitos analógicos não possuem muitas ferramentas para o auxílio de projetos,

dependendo na grande maioria das vezes somente da experiência do projetista do circuito (MINA; JABBOUR; SAKR, 2022).

Neste contexto, este trabalho tem como objetivo o desenvolvimento de uma ferramenta para o dimensionamento de transistores CMOS, de modo a auxiliar o projeto de circuitos integrados nas tecnologias CMOS sub-micrométricas. Esta ferramenta baseia-se no uso das linguagens de programação *Python* e Cadence SKILL, bem como o simulador *Cadence Spectre®*.

1.1 Objetivos

Desenvolver uma ferramenta capaz de dimensionar de forma semi-automática transistores PMOS e NMOS em diferentes processos de fabricação, para serem utilizados em circuitos analógicos.

1.2 Organização do trabalho

Este trabalho está organizado em 5 capítulos. No capitulo 2 é realizada uma fundamentação teórica e bibliográfica relacionada aos principais tópicos que envolvem o tema. No capítulo 3, é apresentada a metodologia do trabalho, formulação do tema, descrevendo as etapas da ferramenta de dimensionamento de transistores, bem como o exemplo do projeto de um transistor NMOS. No capítulo 4, são apresentados os resultados obtidos a partir do método utilizado, aplicando o software desenvolvido nos projetos de diferentes topologias de amplificadores. Por fim, no capítulo 5 ressaltam-se as conclusões e propostas para trabalhos futuros.

2 Fundamentação Teórica e Revisão Bibliográfica

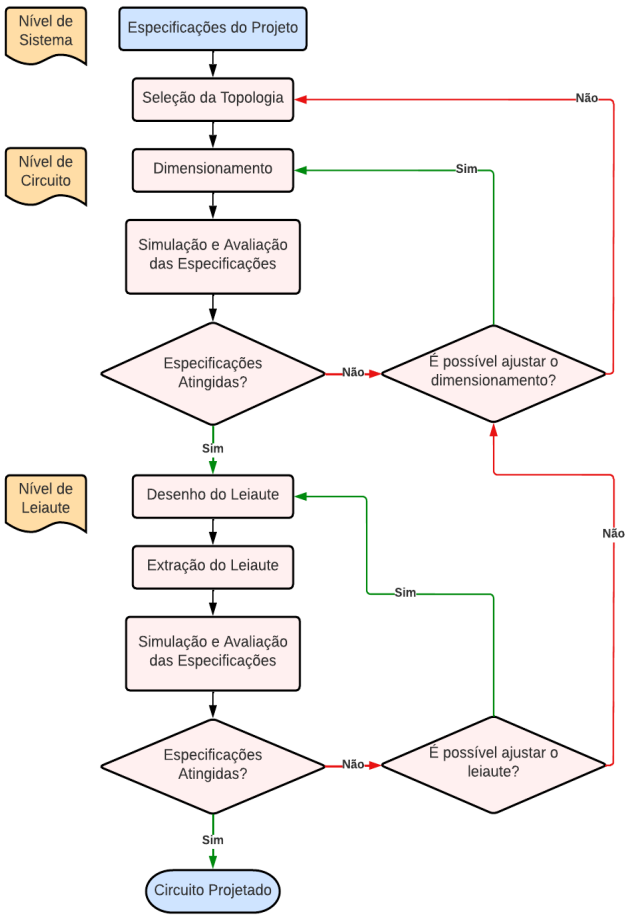
O termo CMOS é derivado de Metal Óxido Semicondutor Complementar, referindo-se à composição construtiva dos MOSFETs (Transistor de Efeito de Campo Metal Óxido Semicondutor), seu elemento principal. Esta tecnologia é muito utilizada no projeto de circuitos integrados, devido a seu baixo consumo de energia, baixo custo e boa performance. De modo geral, existem três tipos de circuitos integrados: analógicos, digitais e mistos. Circuitos analógicos operam com sinais contínuos (em tempo e amplitude), enquanto circuitos digitais trabalham com sinais discretos. Circuitos mistos, são circuitos que possuem tanto partes analógicas quanto digitais em um único chip.
Este trabalho está concentrado no âmbito de circuitos integrados analógicos, portanto, os demais tipos de circuitos não serão abordados no decorrer do trabalho.

2.1 Projeto de Circuitos Integrados Analógicos

Atualmente, grande parte do fluxo de projeto de circuitos analógicos é feito manualmente por um projetista. Algumas ferramentas computacionais de auxílio de projeto (CAD) são utilizadas no processo, como simuladores e editores de leiaute. Desta forma, cabe ao projetista compreender a aplicação e topologia do circuito a ser implementado. O projeto de circuitos integrados analógicos é, de modo geral, dividido em três níveis de abstração, que vão desde as definições das especificações que o circuito deve possuir para desempenhar suas funções, até o desenho físico do circuito (BALKIR; DÜNDAR; ÖGRENCI, 2003). Estes níveis são chamados de nível de sistema (maior abstração), nível de circuito (nível de abstração intermediário) e nível de leiaute (baixa abstração). A Figura 1 mostra o fluxograma do projeto de circuitos analógicos integrados.

No nível de sistema, o circuito analógico é tratado como um bloco funcional, onde as especificações do circuito são determinadas. Para nível de circuito, o bloco é tratado como um esquemático, em que os componentes devem ser dimensionados de acordo com as especificações definidas no nível de sistema. No nível de *layout*, nível de menor abstração, é apresentado o desenho físico do circuito, onde o tamanho dos componentes calculados no nível anterior são utilizados, levando em consideração algumas restrições impostas pela fábrica de circuitos integrados (SEVERO; GIRARDI, 2012).
Este trabalho se concentra no nível de circuito, mais precisamente na etapa do dimensionamento. Segundo o fluxo de projeto de circuitos integrados mostrado na Figura 1, após a escolha da topologia, parte-se ao dimensionamento. Os transistores CMOS, por sua vez,

Figura 1 – Fluxo de projeto de circuitos integrados analógicos.



Fonte: Autor

são os elementos principais nesses circuitos. Neste contexto, o dimensionamento adequado dos transistores CMOS torna-se imprescindível.

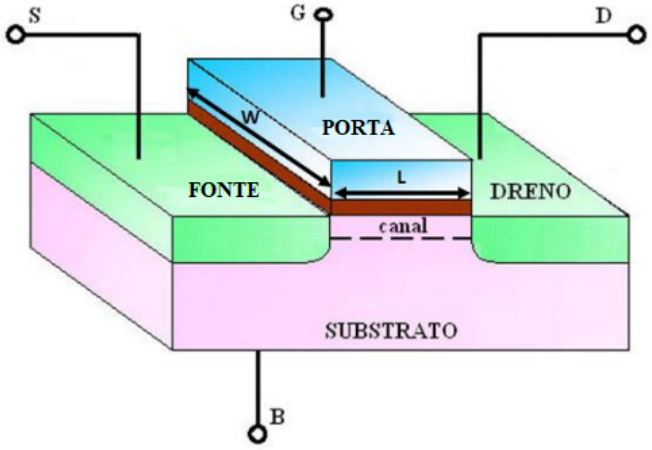
A alta demanda e complexidade em projetos de circuitos integrados analógicos exige o uso de ferramentas CAD (*Computer Aided Design*), em que os circuitos são projetados de forma manual em uma abordagem *top-down*, através dos níveis de abstração do sistema (SEVERO; NOIJE, 2016). Embora não existam ferramentas comerciais que automatizam todo o projeto de CIs, existem metodologias e ferramentas que automatizam parte do projeto, como por exemplo, o uso de uma inteligência artificial (MARTENS; GIELEN, 2008), e heurísticas de otimização (SEVERO et al., 2012).

2.2 Transistores CMOS

Transistores de efeito de campo MOS, são dispositivos que possuem quatro terminais: fonte (source), porta (gate), dreno (drain) e substrato ou corpo (bulk). A operação básica do MOSFET ocorre por meio do fluxo de corrente elétrica entre os terminais dreno e fonte, onde a condutividade é controlada através da tensão aplicada na porta. Há dois tipos de

MOSFETs: os de canal N (NMOS) e os de canal P (PMOS).
A construção destes transistores é dada a partir do uso de três camadas de materiais, chamadas: Metal, Óxido e Semicondutor. A camada de metal atualmente foi substituída por silício policristalino (polissilício), silício altamente dopado que apresenta características de um condutor.
A Figura 2 mostra a estrutura física de um transistor MOS. Percebe-se que, o MOSFET é um dispositivo simétrico, ou seja, a região de difusão do dreno é idêntica à região da fonte. Sendo assim, os terminais fonte e dreno são uma mera questão de referência. Para transistores do tipo N, adota-se a fonte como sendo o terminal de menor potencial, e para o tipo P o de maior potencial.

Figura 2 – Estrutura de um transistor MOS.



Fonte: (SEVERO; GIRARDI, 2011)

Como pode ser visto na Figura 2, o canal do transistor possui um valor de largura W e de comprimento L. As dimensões do canal influenciam diretamente a passagem de cargas elétricas e todas as características elétricas do dispositivo.

2.2.1 Comportamento do Transistor MOS

As equações que representam o comportamento da corrente que passa pelo dreno em um transistor NMOS são mostradas abaixo:

$$I_D = \mu_0 C_{OX} \frac{W}{L}[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}] \tag{2.1}$$

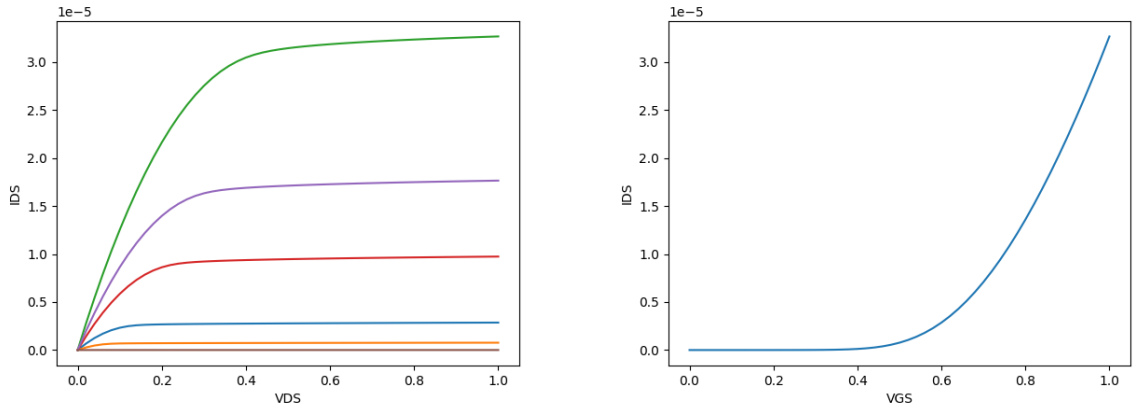
$$I_D = \mu_0 C_{OX} \frac{W}{L}\frac{(V_{GS} - V_T)^2}{2}(1 + \lambda V_{DS}) \tag{2.2}$$

A equação 2.1 demonstra a operação do transistor na região do triodo, quando $0 < V_{DS} \leq V_{GS} - V_T$, e a equação 2.2 mostra a região de saturação, quando $0 < V_{GS} - V_T \leq$

V_{DS}. O transistor opera na região de corte ($I_D = 0$) quando $V_{GS} \leq V_T$ ou $V_{GS} - V_T \leq 0$. Como pode ser observado nas equações 2.1 e 2.2, a corrente I_D é proporcional a largura W e inversamente proporcional ao comprimento L do canal. Ou seja, a corrente elétrica é diretamente proporcional a relação W/L.

Os gráficos na Figura 3 demonstram o comportamento das curvas características de um dispositivo MOSFET do tipo N. Estas curvas relacionam o comportamento da corrente entre os terminais de dreno e source (I_{DS}) com as tensões V_{GS} e V_{DS}.

Figura 3 – Curvas características I_{DS} x V_{DS} (a) e I_{DS} x V_{GS} (b)



(a) (b)

A Figura 3a mostra as regiões de operação de transistor MOSFET para V_{GS} entre $250mV$ e $1V$, região de sublimiar (Se $V_{GS} - V_T < 0$, uma pequena corrente flui através do transistor), região do triodo (I_{DS} varia linearmente com V_{DS}) e região de saturação (I_{DS} é aproximadamente constante).

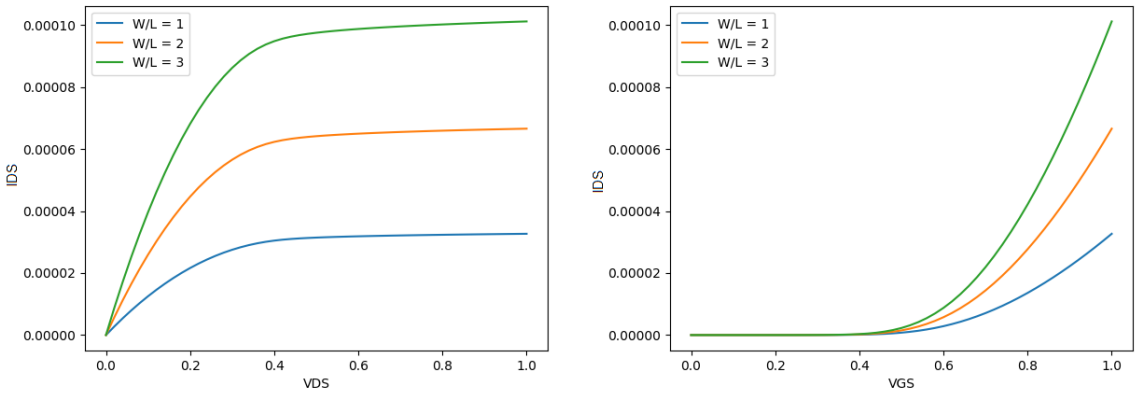
A tensão V_{GS} controla a inversão de portadores de carga na região do canal. Quando V_{GS} é pequena ($V_{GS} < V_T$), o transistor opera em inversão fraca (*Weak inversion*). Para valores maiores de V_{GS}, o transistor opera em inversão forte (*Strong inversion*). A transição entre estes níveis de inversão é chamada de inversão moderada (*Moderate inversion*). O nível de inversão pode ser avaliado a partir da diferença entre as tensões V_{GS} e V_T, chamada de tensão de *overdrive* (V_{OV}), representada na equação 2.3 (GIRARDI; AGUIRRE; SEVERO, 2022).

$$V_{OV} = V_{GS} - V_T \tag{2.3}$$

Na Figura 3b é possível observar a região de sublimiar, em que I_{DS} é quase nulo para tensões menores que a tensão de limiar V_T. Para V_{GS} maior que V_T, a corrente I_{DS} aumenta quadraticamente.

Como mostrado nas equações 2.1 e 2.2, a corrente I_{DS} varia linearmente com a relação W/L. Este comportamento pode ser visto na Figura 4:

Figura 4 – Proporcionalidade entre a corrente I_{DS} e a relação W/L para $W/L = 1, 2$ e 3.



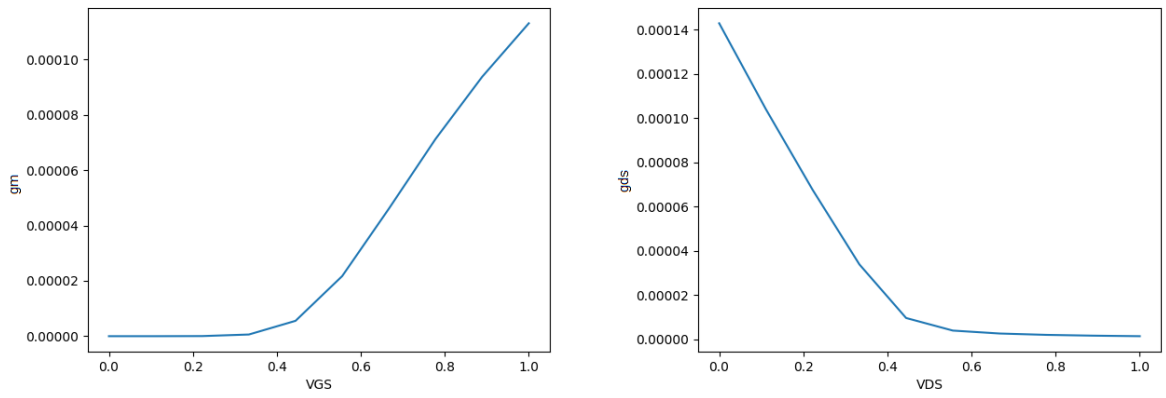
(a) (b)

Outras características significativas a serem analisadas são as variações entre corrente I_{DS} e as tensões V_{GS} e V_{DS}, importantes para análise de pequenos sinais do dispositivo. Estas variações, denominadas transcondutância de gate (gm) e condutância de saída (gds) podem ser calculadas com as equações 2.4 e 2.5, a partir das derivadas parciais de I_{DS}. As curvas típicas de gm e gds são mostradas na Figura 5:

$$gm = \frac{\partial I_{DS}}{\partial V_{GS}} \tag{2.4}$$

$$gds = \frac{\partial I_{DS}}{\partial V_{DS}} \tag{2.5}$$

Figura 5 – Transcondutância de gate (a) e condutância de saída (b).

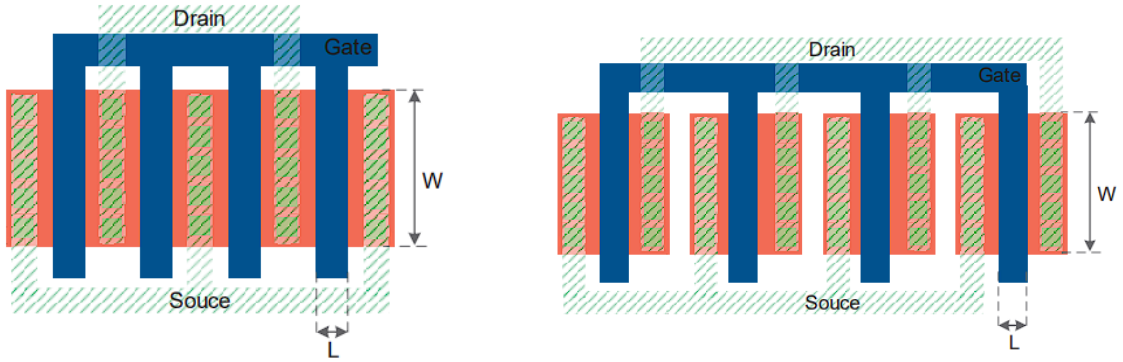


(a) (b)

2.2.2 Multiplicidade e *fingers*

Algumas características que devem ser citadas devido a sua importância no projeto de transistores são o multiplicidade (M) e os *fingers*(N_F).
Devido à razão $I_D \propto \frac{W}{L}$, a densidade da corrente I_D é baixa quando L possui um valor relativamente alto. Neste caso, aumentar a largura W consequentemente aumenta as capacitâncias parasitas. Uma solução é a partir do uso de transistores em paralelo.
O uso de MOSFETs em paralelo conectados pelo terminal gate é definido através da multiplicidade, onde o número de transistores associados em paralelo é dado por M.
O número de fingers (N_F) possui função semelhante a multiplicidade, porém os transistores utilizam a mesma difusão e compartilham os terminais internos.
A capacitância de dreno e *source* para o terminal *bulk* tende a ser melhor em dispositivos multifinger do que em dispositivos com múltiplos associados em paralelo. Isto ocorre devido ao compartilhamento da difusão.
A Figura 6 mostra exemplos de transistores multifinger e em paralelo.

Figura 6 – Transistor multifinger (a) e em paralelo (b).



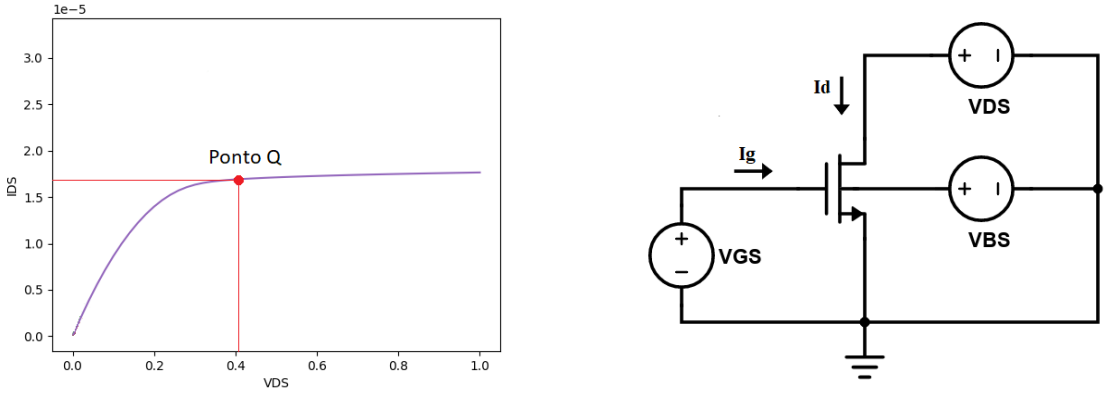
(a) (b)

Fonte: (SEVERO; NOIJE, 2019)

2.2.3 Polarização do Transistor

Polarizar um transistor é setar o seu ponto de operação em corrente contínua, dentro de suas curvas características. Este ponto de operação, é chamado de ponto quiescente, também chamado de ponto Q ou *bias*.
A polarização do transistor é relacionada diretamente com as tensões aplicadas em seus terminais V_{GS}, V_{DS} e V_{BS}. Na Figura 7 (a), é mostrado o ponto Q de um MOSFET. A corrente I_D é fixada em um ponto em que há uma tensão V_{DS} correspondente, polarizando o transistor. A figura 7 (b) mostra que, aplicando determinadas tensões V_{GS}, V_{DS} e V_{BS}, uma corrente I_D é obtida.

Figura 7 – Polarização do transistor. (a) Gráfico de $I_{DS} \times V_{DS}$ mostrando o ponto Q do transistor, e (b) tensões aplicadas nos terminais.



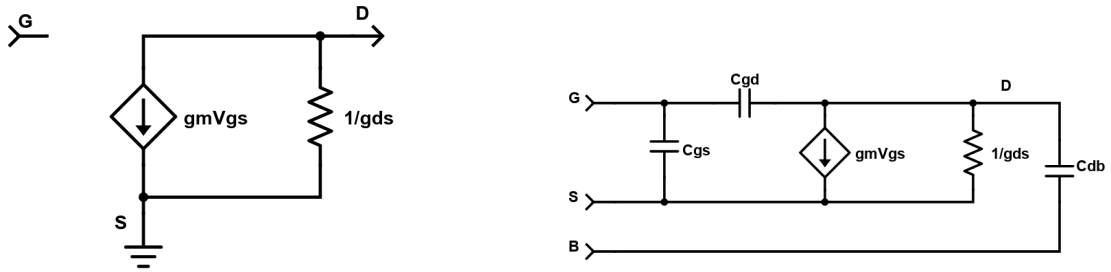
(a) (b)

Fonte: Autor

2.2.4 Modelo de pequenos sinais

O modelo de pequenos sinais representa uma aproximação linear do comportamento dos transistores a partir de uma representação simplificada de seu circuito equivalente, polarizado em seu ponto de operação (RODRIGUEZ et al., 2014).
A Figura 8 mostra o esquemático dos modelos de pequenos sinais simplificado e considerando capacitâncias parasitas para um transistor NMOS em baixa frequência.

Figura 8 – Modelos de pequenos sinais: (a) NMOS simplificado, e (b) NMOS com capacitâncias parasitas.



(a) (b)

Fonte: Autor

2.3 Síntese do capítulo

Neste capítulo foi abordada uma introdução ao projeto de circuitos integrados analógicos, bem como dispositivos MOSFETs e suas principais características físicas e elétricas, as equações que demonstram seu comportamento e as curvas características correspondentes.

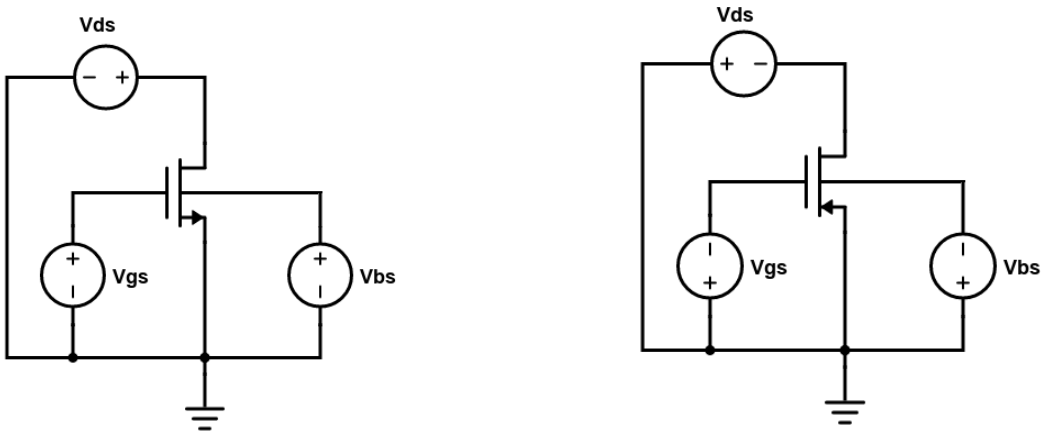
3 Metodologia

A modelagem de transistores CMOS não é uma tarefa fácil quando se trata de processos de fabricação em escalas sub-micrométricas, devido a efeitos de segunda ordem e da complexidade do processo de fabricação. Como consequência, não é possível projetar um transistor diretamente utilizando as equações 2.1 e 2.2.
De modo a desenvolver uma estratégia genérica para o dimensionamento de transistores CMOS, uma ferramenta baseada no uso do simulador comercial Cadence Spectre® é proposta.

3.1 Ferramenta para o dimensionamento de transistores CMOS

A ferramenta proposta é estruturada pela implementação de um algoritmo na linguagem de programação Python para automatizar a simulação SPICE de ponto de operação, utilizando o simulador elétrico Cadence Spectre®.
A simulação SPICE de ponto de operação utiliza as informações do transistor, como a largura (W) e o comprimento (L) do canal, além das tensões V_{DS}, V_{GS} e V_{BS}, entre outros parâmetros.
A Figura 9 mostra o esquemático criado para os transistores NMOS e PMOS para a simulação SPICE.

Figura 9 – Diagramas dos transistores NMOS (a) e PMOS (b)



(a) (b)

Fonte: Autor

Em ambos os transistores, fontes de tensão foram conectadas em seus terminais gate, source, drain e bulk com valores não atribuídos inicialmente, que serão determinados

a partir das especificações do projeto do circuito integrado.
As simulações SPICE retornam arquivos com os resultados da simulação de ponto de operação, em que estes dados podem ser processados e utilizados no dimensionamento dos transistores. Estes resultados incluem todos os parâmetros do modelo de pequenos sinais dos transistores.
O simulador Cadence Spectre® foi escolhido para esta ferramenta por ser um dos principais simuladores no mercado, onde atualmente é muito usado por projetistas de circuitos integrados e não requer a configuração de um ambiente completo de simulação para seu uso.
O algoritmo foi implementado de forma iterativa a partir das equações 3.1 e 3.2, utilizando os resultados da simulação de ponto de operação. Em outras palavras, após o usuário identificar as especificações, o arquivo SPICE é simulado e retorna o arquivo com os resultados da simulação. A partir destes resultados, a equação 3.1 é calculada comparando o erro do parâmetro desejado e do mesmo parâmetro porém na iteração atual, e caso o erro na iteração atual seja maior que o erro máximo, a equação 3.2 recalcula o parâmetro variável, repetindo a simulação até que algum dos critérios seguintes sejam atendidos: o erro na iteração atual for menor que o erro máximo admitido, o número máximo de iterações foi atingido, ou o tamanho máximo ou mínimo do transistor foi atingido e o resultado não foi alcançado.

$$Erro = \frac{|Par_{atual} - Par_{ref}|}{|Par_{ref}|} \tag{3.1}$$

$$Var_{n+1} = \frac{Par_{ref}}{Par_{atual}} \times Var_n \tag{3.2}$$

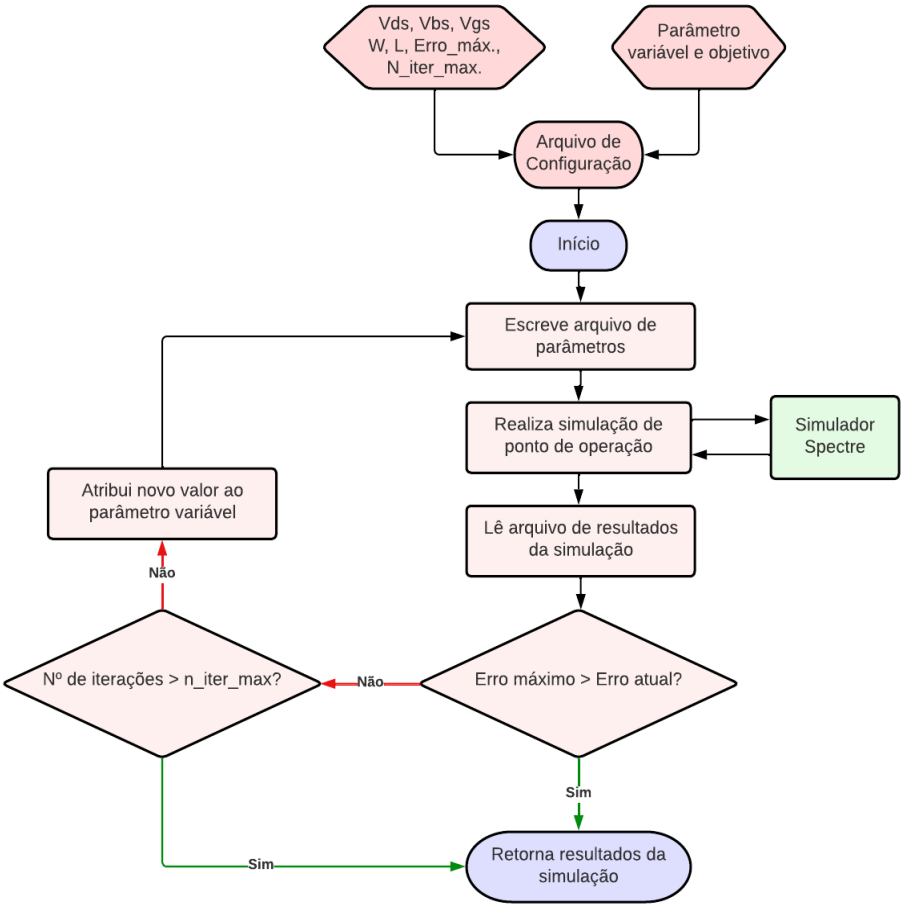
Onde Par_{ref} é o parâmetro objetivo (parâmetro a ser buscado), Par_{atual} é o parâmetro objetivo na iteração atual, Var_n o parâmetro variável na iteração atual, e Var_{n+1} o valor do parâmetro variável na próxima iteração.
O erro é calculado a cada iteração, e caso seja maior que o erro máximo especificado pelo usuário, o parâmetro Var é recalculado, como mostrado na equação 3.2 e é feita a simulação novamente, até que algum dos critérios de parada sejam atingidos.
As dimensões de largura e comprimento do transistor são delimitadas a partir da tecnologia usada, em que a tecnologia de 180 nm possui os seguintes limites:

Tabela 1 – Limites superior e inferior das dimensões de MOSFETs na tecnologia CMOS de 180 nm.

Parâmetro	Mínimo	Máximo
W (µm)	0.22	900
L (µm)	0.18	20.0

A Figura 10 ilustra o fluxograma do funcionamento da ferramenta proposta.

Figura 10 – Fluxograma ilustrando o processo de dimensionamento da ferramenta.



Fonte: Autor

3.1.1 Parâmetros importantes para o projeto de um transistor CMOS

Para o projeto de transistores CMOS, alguns parâmetros são necessários na metodologia apresentada. Enquanto um dos parâmetros é variável e será ajustado durante as simulações, outros parâmetros devem ser fixados. Os parâmetros importantes para o projeto são listados abaixo:

- $Technology$: Tecnologia de fabricação usada no projeto.
- $Model\_name$: Nome do modelo do transistor.
- V_{GS}: Tensão entre os terminais gate e source.
- V_{DS}: Tensão entre os terminais dreno e source.
- V_{BS}: Tensão entre os terminais bulk e source.
- NF: Número de fingers.
- M: Multiplicidade.

- $Iter_{MAX}$: Número máximo de iterações.
- $Erro_{MAX}$: Erro máximo aceitável para o parâmetro buscado.
- Par_{ref}: Parâmetro a ser buscado.
- Var: Parâmetro que será variado.

Os parâmetros que podem ser variados são W e L, enquanto os que podem ser buscados são definidos como os parâmetros I_D, gm, gds, e gmb, entre outros.

3.1.2 Implementação do código-fonte

A implementação do código-fonte foi feita a partir do uso da linguagem de programação Python. As bibliotecas utilizadas são bibliotecas as padrões *sys* e *os*, que permitem o uso de comandos do sistema operacional a partir de códigos em Python. Nenhuma biblioteca externa foi usada para o código principal. Para a interface gráfica, foram utilizadas as bibliotecas *matplotlib, pandas* e *PyQt*.
O simulador Spectre® é executado a partir da linha de comandos do sistema operacional, em que é indicado o arquivo no formato SPICE a ser simulado, e o formato dos arquivos de saída da simulação. Neste caso, é necessário usar a formatação no padrão ASCII para que os dados possam ser manipulados nos códigos Python. Segundo o guia de usuário do simulador Spectre®(CADENCE DESIGN SYSTEMS, 2002), o comando que executa o simulador pode ser visto abaixo:

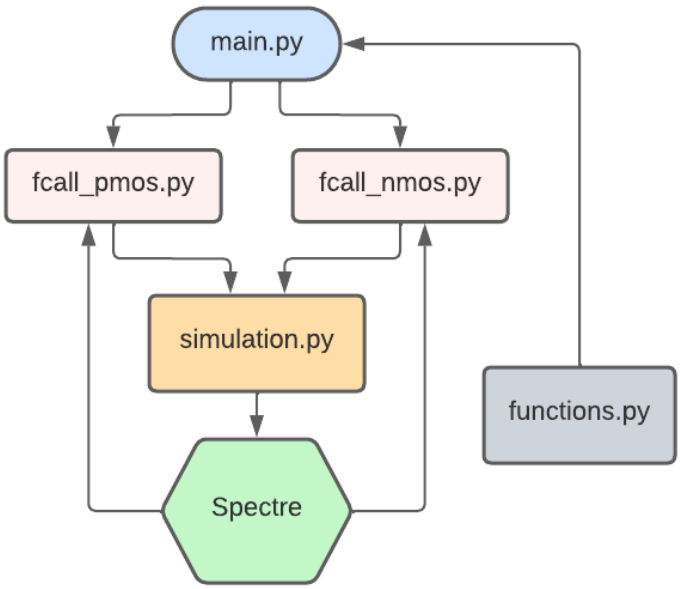
Figura 11 – Comando que executa o simulador Spectre®.

```
spectre -64 netlist.scs -format psfascii
```

O código-fonte da ferramenta foi divido em 5 arquivos diferentes, para facilitar a organização. Estes arquivos são: *main.py*, $fcall\_nmos.py$, $fcall\_pmos.py$, *simulation.py* e $functions.py$. O arquivo *main.py* é o arquivo principal da ferramenta, pois a partir dele que os outros arquivos serão chamados. $fcall\_nmos.py$ e $fcall\_pmos.py$ possuem as funções de custo (eq. 3.1) e de ajuste (eq. 3.2) em um loop até que alguma das condições de parada ocorra, para os transistores NMOS e PMOS, respectivamente. O arquivo *simulation.py* é usado para executar o simulador Spectre®, e retorna os dados de saída das simulações. Por fim, $functions.py$ possui funções que são usadas pelos outros arquivos. A representação da relação entre os arquivos é mostrada na Figura 12.

Após a execução da ferramenta, um arquivo .CSV com os resultados é gerado, contendo todos os parâmetros obtidos na simulação.

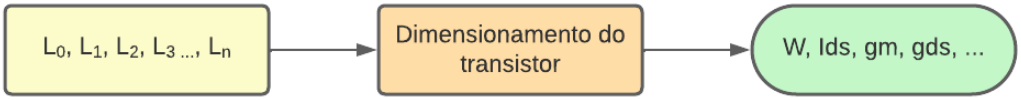
Figura 12 – Relação entre os arquivos da ferramenta.



3.2 Ferramenta de varredura e busca de parâmetros

A partir das funcionalidades da primeira ferramenta, foi possível desenvolver uma segunda ferramenta capaz de buscar o parâmetro desejado e realizar uma varredura em um determinado intervalo de um segundo parâmetro. Desta forma, é possível realizar uma busca do parâmetro desejado, obtendo diferentes resultados.
Esta ferramenta possui a opção de traçar gráficos após obter os resultados.
Os parâmetros que resultam da varredura são: *L, W,* I_{DS}*, gm, gds, gmb, cgd, cgs, cdb, cds, gm/*I_D*, gm/gds, gmb/gds*, e *gmb/gm*. Já os parâmetros disponíveis para a realização da varredura, são: *W, L,* V_{DS}, V_{BS}, e V_{GS}, em que o objetivo é um dos seguintes parâmetros: I_{DS}*, gm, gds,* e *gmb*.
A Figura 13 ilustra a operação da ferramenta.

Figura 13 – Fluxograma mostrando o processo de varredura da ferramenta.

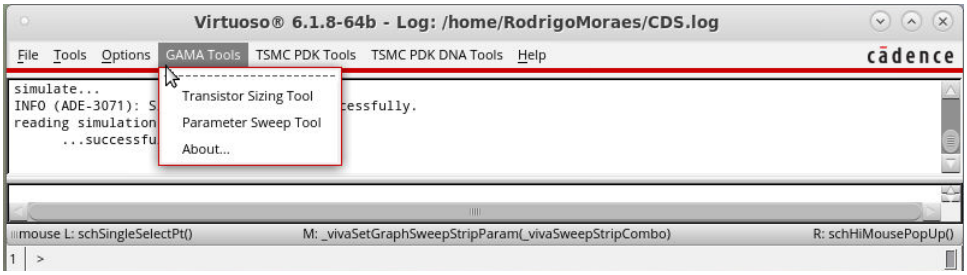


Esta ferramenta é extremamente útil no projeto de circuitos integrados analógicos, pois torna-se possível visualizar os *trade-offs* em relação à mudança da magnitude de diversos parâmetros úteis para o projeto.

3.3 Interface gráfica

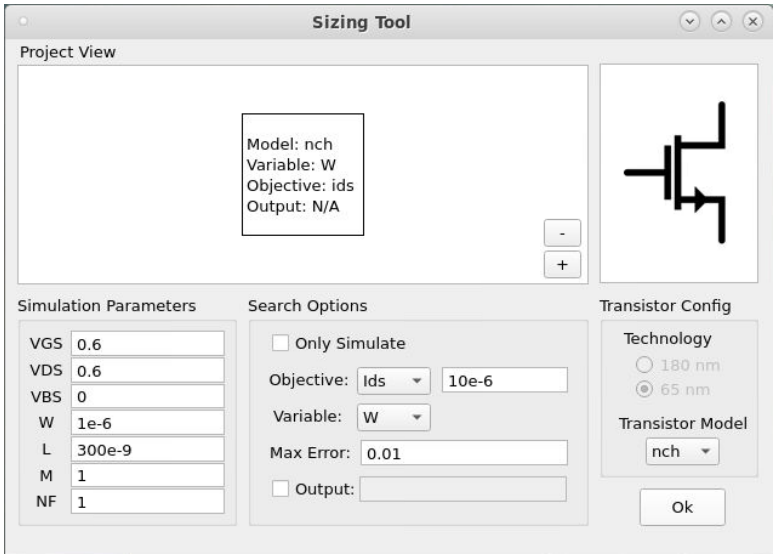
Para facilitar e incentivar o uso da ferramenta por parte dos projetistas, uma interface gráfica foi desenvolvida para cada ferramenta. Ambas as ferramentas foram integradas ao terminal do software Cadence Virtuoso®, tornando-as de fácil acesso. O código na linguagem SKILL pode ser visto no Apêndice A.2.

Figura 14 – Visualização do terminal *CIW* do software Cadence Virtuoso®



A integração foi feita por meio da linguagem de programação Cadence SKILL®, desenvolvida pela Cadence Design Systems, com o objetivo de ser uma linguagem eficiente e flexível para o auxílio de ferramentas CAD (BARNES, 1990).
Na Figura 15, a interface gráfica da ferramenta de dimensionamento é mostrada. O painel principal "Project View", mostra algumas informações importantes sobre a sequência dos projetos de dimensionamento. Vários projetos podem ser adicionados em sequência, podendo utilizar o parâmetro de saída do projeto anterior como objetivo do subsequente, como por exemplo, no caso do primeiro transistor ser projetado para um certo parâmetro, como gm ou gds, e na sequência um segundo transistor possui dependências em relação ao primeiro, como a corrente I_{DS}.

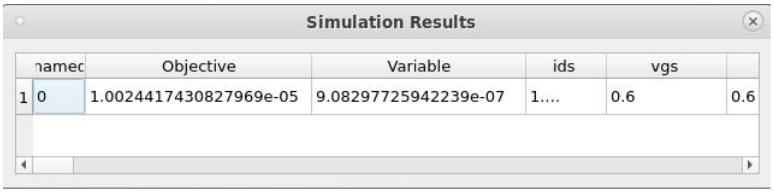
Figura 15 – Visualização da ferramenta de dimensionamento de transistores.



Os resultados são exibidos em uma janela que abre quando o processo de busca do parâmetro é finalizado, onde é exibida uma tabela com todos os parâmetros do transistor

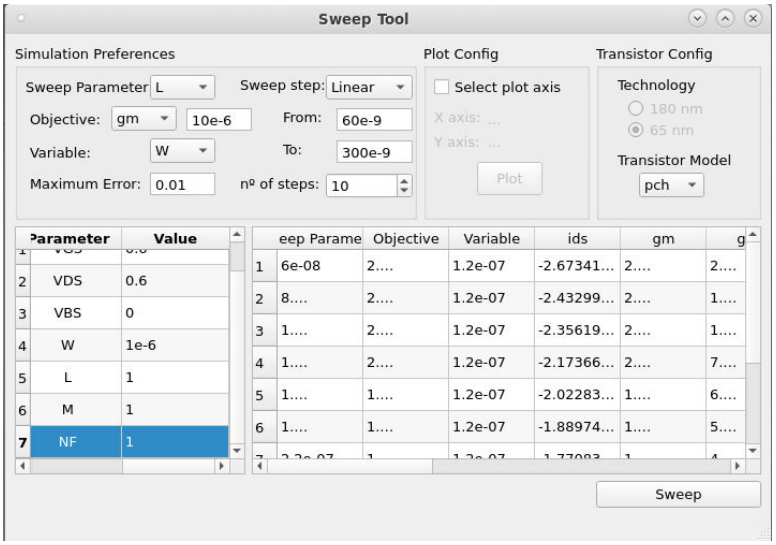
obtidos após a simulação, como pode ser visto na Figura 16.

Figura 16 – Janela mostrando os resultados obtidos do dimensionamento.



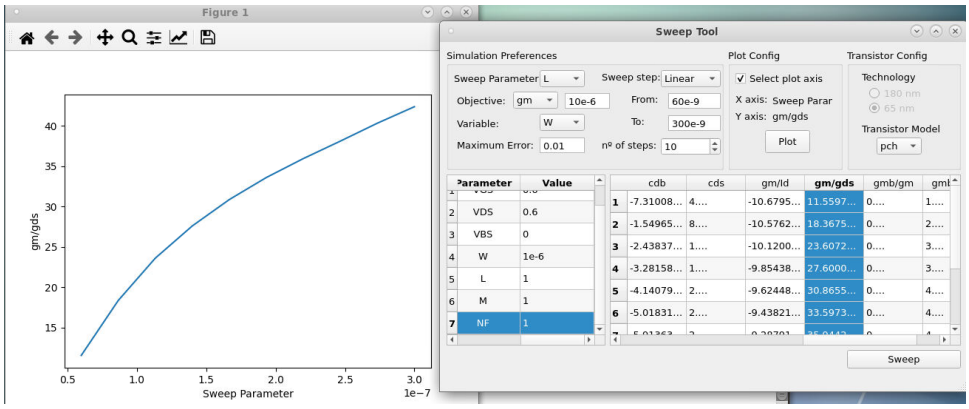
A Figura 17 mostra a interface gráfica da ferramenta de varredura. Diferente da ferramente anterior, a visualização dos resultados é vista diretamente na janela principal.

Figura 17 – Visualização da ferramenta de varredura de parâmetros.



Esta ferramenta possui a função de visualização de gráficos utilizando os dados obtidos após a varredura. Na Figura 18, é mostrado o gráfico da relação $\frac{gm}{gds}$xL. A seleção de parâmetros foi feita com base em funções de seleção definidas pelo clique do mouse.

Figura 18 – Janela da ferramenta de varredura e gráfico gerado.



3.4 Exemplo de utilização da ferramenta

Com o objetivo de demonstrar o funcionamento da ferramenta, nesta seção será projetado um transistor NMOS na tecnologia CMOS de 180 nm. O objetivo será projetar o transistor para que possua uma corrente de 10 mA, usando o parâmetro W como variável. A ferramenta pode ser utilizada pela interface gráfica desenvolvida, ou diretamente via código Pyhton. Nesta seção, serão demonstradas as duas formas. Após o uso da ferramenta de dimensionamento, será usada a ferramenta de varredura para analisar a mudança da corrente I_D de acordo com a variação de L.

3.4.1 Ferramenta de dimensionamento

A ferramenta de dimensionamento é recomendada para o dimensionamento de transistores quando já existem valores definidos para L e para as fontes de polarização definidos para o projeto. A Figura 19 mostra o trecho do código-fonte em que é chamada a função *main*, onde os parâmetros do projeto são inseridos.

Figura 19 – Trecho do código-fonte que mostra os parâmetros na função principal.

```
main.main('180nm', 'nch', target_name='ids', target_value=10e-3,
max_error=0.01, L=180e-9, NF=1, M=1, VGS=0.6, VDS=0.6, VBS=0,
parameter_calc_name='W')
```

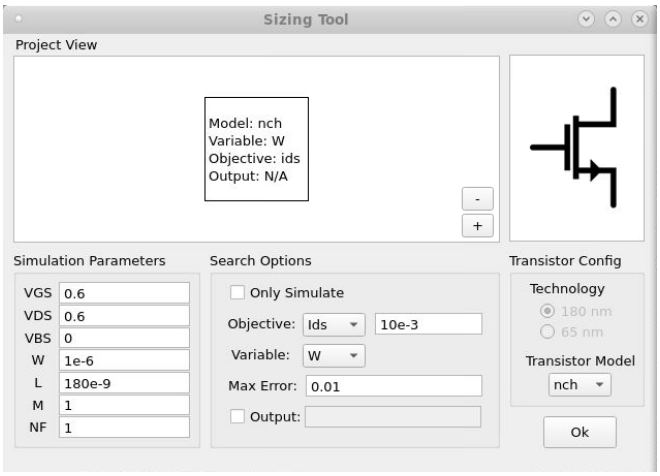
Os dados inseridos como parâmetros na função *main* podem ser vistos na tabela 2.

Tabela 2 – Parâmetros usados para o exemplo de design.

Parâmetro	Valor
$I_{DS}(mA)$	10
V_{GS} (V)	0.6
V_{DS} (V)	0.6
V_{BS} (V)	0
W (µm)	Variável a ser obtida
L (µm)	0.18
Número de Fingers	1
Multiplicador	1
Erro máximo (%)	1

A Figura 20 mostra a janela da ferramenta de dimensionamento, com os dados mostrados na tabela 2, que é equivalente ao trecho do código-fonte mostrado na Figura 19. Nota-se que, embora W seja a variável do projeto, o valor de 1 µm é inserido. Neste caso, $W = 1$ µm é o valor inicial da busca, que irá variar de acordo com as equações 3.1 e 3.2, até alcançar o objetivo especificado.

Figura 20 – Interface gráfica da ferramenta de dimensionamento, com os dados do exemplo proposto.

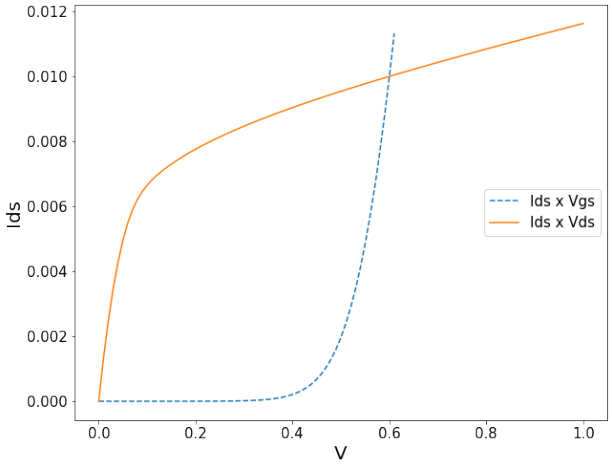


Após 4 iterações a simulação é finalizada, e os parâmetros obtidos podem ser observados na tabela 3. A Figura 21 mostra o gráfico das curvas características $V_{DS} \times I_D$ e $V_{GS} \times I_D$ do transistor projetado.

Tabela 3 – Resultados obtidos para o exemplo de design.

Parâmetros de projeto	Valor
W (µm)	757.68
I_D (mA)	10
Alguns parâmetros disponíveis	**Valor**
V_{TH} (mV)	556.87
V_{DSAT} (mV)	85.49
$gm(mS)$	130.56
$gds(mS)$	4.52
nº de iterações	4

Figura 21 – Curvas características do transistor projetado.



Como resultado, obtém-se a corrente $I_D = 10\ mA$, bem como outros parâme-

tros, como as tensões V_{TH}, V_{DSAT}, e a largura do canal W, a partir das especificações estabelecidas inicialmente no projeto.

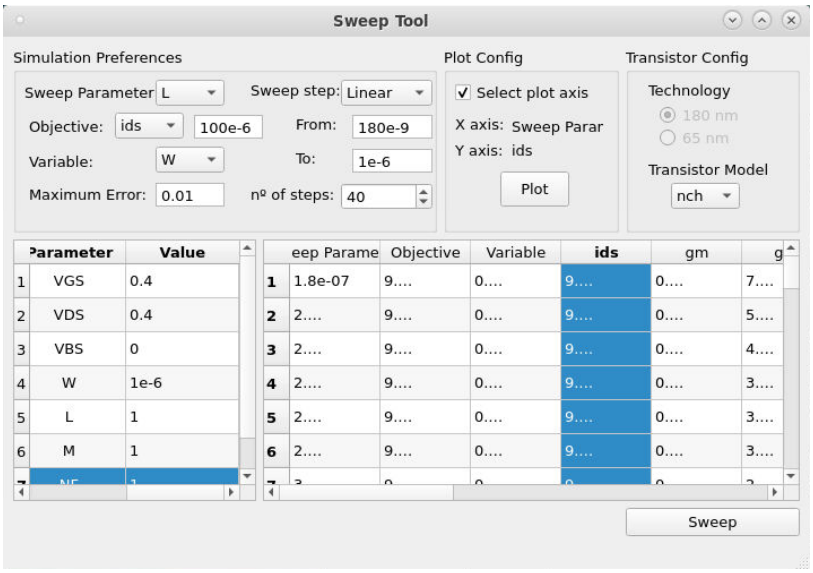
3.4.2 Ferramenta de varredura

Na ferramenta de varredura será analisado o dimensionamento de um transistor NMOS para manter $I_D = 100\,\mu A$ fixo para diferentes valores de L. Na Figura 22, a janela da ferramenta de varredura é mostrada com os dados mostrados na tabela 4 inseridos.

Tabela 4 – Parâmetros usados para o exemplo da ferramenta de varredura.

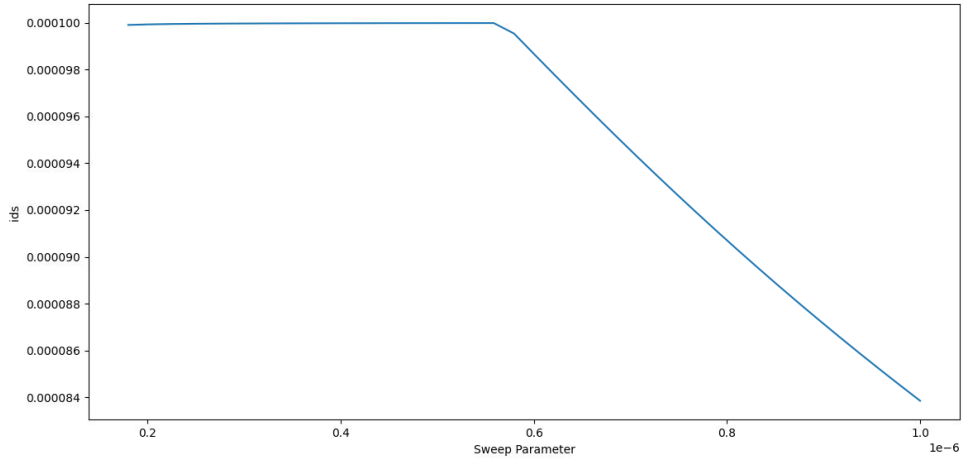
Parâmetro	Valor
$I_{DS}(\mu A)$	100
$V_{GS}(V)$	0.4
$V_{DS}(V)$	0.4
$V_{BS}(V)$	0
$L_{min}(\mu m)$	0.18
$L_{max}(\mu m)$	1
Multiplicador	1
Número de Fingers	1
Erro máximo (%)	1

Figura 22 – Interface gráfica da ferramenta de varredura, com os dados do exemplo proposto.



Como mostrado na Figura 23, a corrente $I_{DS} \approx 100\,\mu A$ é alcançada e mantida constante para $L < 600\,nm$.
Outros parâmetros podem ser analisados com esta ferramenta, tornando-a extremamente útil para o auxílio de projetos de circuitos integrados analógicos.

Figura 23 – Gráfico de I_D x L do exemplo desta seção.



3.5 Síntese do capítulo

Neste capítulo foi demonstrada a metodologia e funcionamento da ferramenta proposta neste trabalho, definindo suas funcionalidades, ferramentas usadas e por fim, um exemplo do design de um transistor NMOS, apresentando o resultado com as curvas características do mesmo.

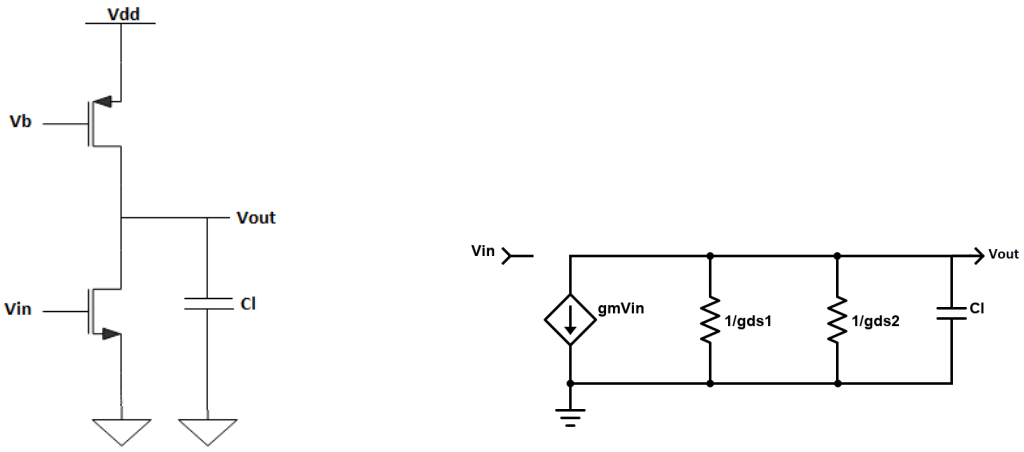
4 Projeto e Resultados

Nesta seção, é demonstrada a aplicação das ferramentas de dimensionamento de transistores CMOS. Foram desenvolvidos quatro projetos no total, o primeiro sendo um amplificador de fonte comum, o segundo um amplificador baseado em inversor (*push-pull*), o terceiro um amplificador diferencial, e o quarto, um transdutor negativo.

4.1 Amplificador Common-Source

O amplificador common-source é uma das topologias básicas de amplificadores. Ele consiste de dois transistores, um PMOS e um NMOS, onde o transistor PMOS polariza a corrente, e o transistor NMOS amplifica o sinal.
A Figura 29 (a) mostra o esquemático do amplificador common-source, e a Figura 29 (b) mostra o circuito equivalente para pequenos sinais.

Figura 24 – Circuito do amplificador Common-Source: (a) Esquemático, e (b) Circuito equivalente para pequenos sinais.



(a) (b)

A partir da análise do circuito equivalente de pequenos sinais, podemos obter as equações do ganho Av e do GBW para o amplificador:

$$A_V = \frac{-gm_1}{gds_1 + gds_2 + j\omega C_L} \tag{4.1}$$

$$GBW = \frac{gm_1}{2\pi C_L} \tag{4.2}$$

Considerando o projeto na tecnologia CMOS de 65 nm, a tabela 5 mostra as seguintes especificações definidas para o projeto:

Tabela 5 – Especificações do projeto do amplificador common-source.

Parâmetro	Valor
$V_{DD}(V)$	1,2
$GBW(MHz)$	1
$C_L(pF)$	10
$A_V(dB)$	30

A partir dos dados mostrados na tabela 5, podemos determinar alguns parâmetros que serão os objetivos de busca da ferramenta. Isolando o parâmetro gm na equação 4.2, podemos estimar o valor da transcondutância necessário.

$$gm_1 = 2\pi C_L GBW = 62,83\,\mu S$$

Sabendo que $Av = 30\ dB = 31,62\ V/V$, e considerando $gds_1 = gds_2 = gds$, é possível determinar gds, isolando o termo na equação 4.1:

$$gds = \frac{gm_1}{2A_V} \approx 1\,\mu S$$

Desta forma, temos que a razão $gm/gds \approx 63$.
Nesta topologia, a função do transistor PMOS é polarizar a corrente I_D, portanto $I_{D_1} = I_{D_2}$.
Com estas informações, é possível projetar o circuito com a ferramenta proposta. O primeiro passo é encontrar a relação $gm/gds = 63$ para o transistor NMOS. Para isto, é usada a ferramenta de varredura para encontrar a razão gm/gds em função de L.
Para este caso, W é usado como variável, o parâmetro de variação é L ($60\,nm < L < 1\,\mu m$), e o objetivo é $gm = 63\,\mu S$.
A Figura 25 mostra que, entre 800 nm e 1 µm encontramos a relação desejada. Mais especificamente, o valor de $L = 879\,nm$ é escolhido por estar mais próximo do valor da razão gm/gds desejado.

Para estes parâmetros, temos que $I_D = 6,14\,\mu A$. O próximo passo então é buscar a corrente I_D para o transistor PMOS, que possua $gds \approx 1\,\mu S$.
Na Figura 26, pode ser visto que para $L > 400\,nm$ temos que gds é aproximadamente o valor almejado.

Os parâmetros obtidos para o projeto são mostrados na tabela 6.

Para validar a solução obtida com a ferramenta, o circuito foi simulado no software Cadence Virtuoso.
Através da análise DC do ponto de operação, podemos observar na Figura 27 os parâmetros após a simulação, mostrando as correntes e tensões coerentes com os valores esperados.

Figura 25 – Gráfico da relação gm/gds em função de L para o amplificador common-source.

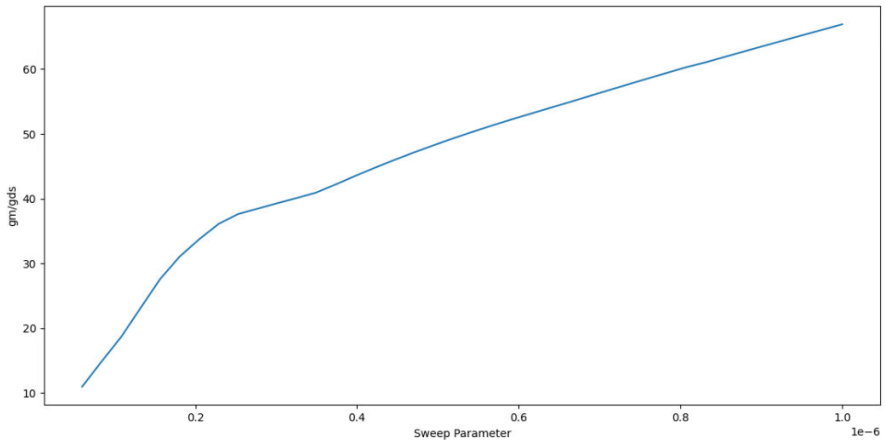


Figura 26 – Gráfico de gds x L do transistor PMOS para o amplificador common-source.

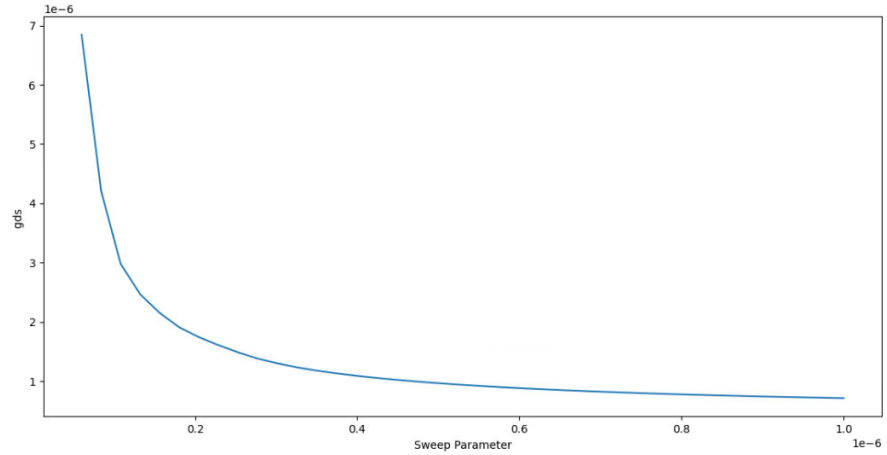


Tabela 6 – Resultados obtidos para o projeto do amplificador common-source.

Parâmetro	NMOS	PMOS
gm(µS)	62,6	Não se aplica
gds(µS)	0,99	1
I_D(µA)	6,14	6,14
L(µm)	0,88	0,47
W(µm)	1,31	1,52

A análise AC resulta no gráfico da resposta em frequência do circuito, como mostra a Figura 28. O gráfico indica que o ganho Av é aproximadamente 30 dB para baixas frequências, e é próximo a zero quando a frequência é igual ao GBW.

Como pode ser visto na Figura 28, o ganho é estável em aproximadamente 30dB até cerca de 10kHz.

Figura 27 – Amplificador Common-Source após a similação de ponto de operação no software Cadence Virtuoso®.

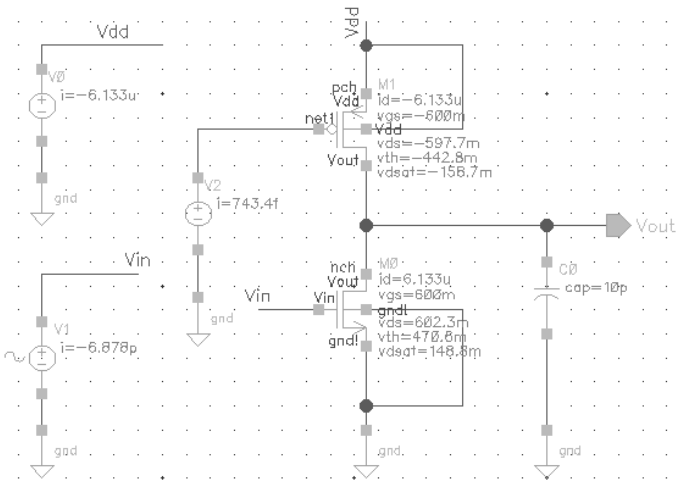
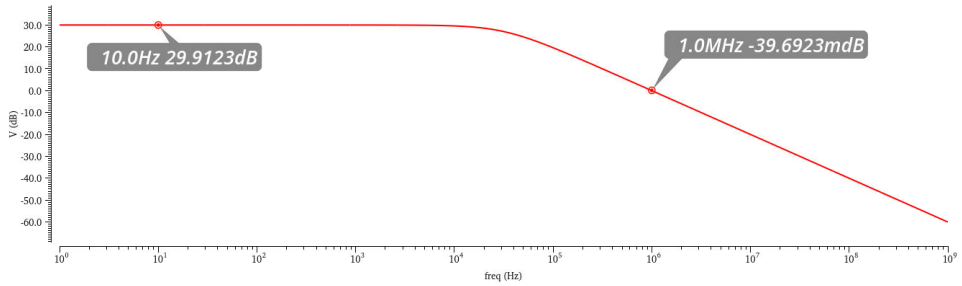


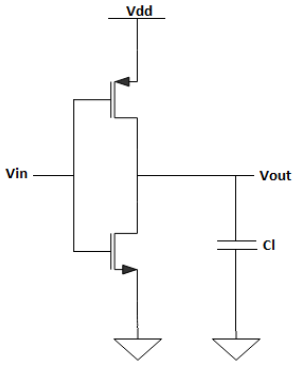
Figura 28 – Resposta em frequência do amplificador common-source.



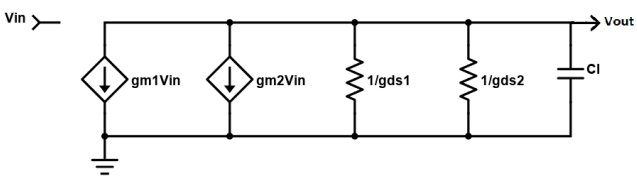
4.2 Amplificador Push-Pull

O amplificador push-pull utiliza dois transistores, PMOS e NMOS, ambos ligados ao mesmo sinal de entrada. Na Figura 29 (a), o esquemático do circuito é mostrado, e na Figura 29 (b) é mostrado o circuito equivalente para pequenos sinais.

Figura 29 – Circuito do amplificador push-pull: (a) Esquemático, e (b) Circuito equivalente para pequenos sinais.



(a)



(b)

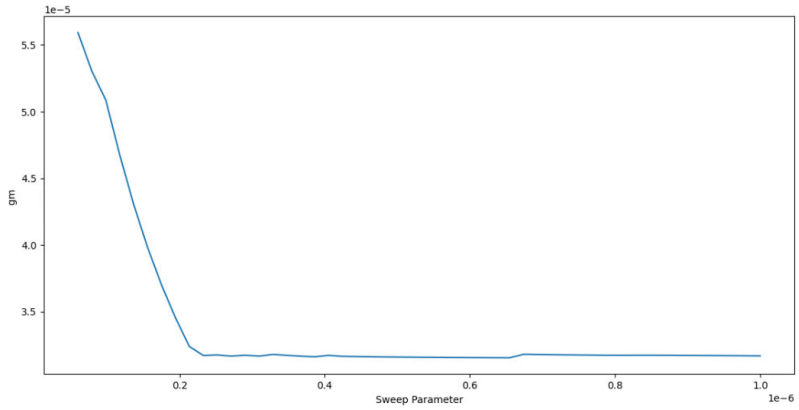
Diferentemente do amplificador common-source, o ganho desta topologia depende da transcondutância dos transistores PMOS e NMOS. A análise do modelo de pequenos sinais resulta nas equações 4.3 e 4.4:

$$Av = \frac{-(gm_1 + gm_2)}{gds_1 + gds_2 + j\omega C_L} \tag{4.3}$$

$$GBW = \frac{gm_1 + gm_2}{2\pi C_L} \tag{4.4}$$

Assim como no projeto anterior, este também será desenvolvido com a tecnologia CMOS de 65 nm. Os parâmetros para este projeto são os mesmos utilizados no amplificador common-source, que podem ser vistos na tabela 5. Considerando $gm_1 = gm_2 = gm$, e $gds_1 = gds_2 = gds$, temos que $gm = 31.41\,\mu S$, e $gds = 1\,\mu S$. Inicialmente, o transistor NMOS foi dimensionado, com o objetivo de buscar $gm = 31.41\,\mu S$. Na Figura 30, pode ser observada a relação gm x L, em que só é alcançado o valor de gm desejado quando $L \approx 230\,nm$. A partir desta curva, o transistor NMOS é dimensionado, em que $L = 232\,nm$, e $I_D = 3.53\,\mu A$.

Figura 30 – Gráfico da relação gm x L para o transistor NMOS.



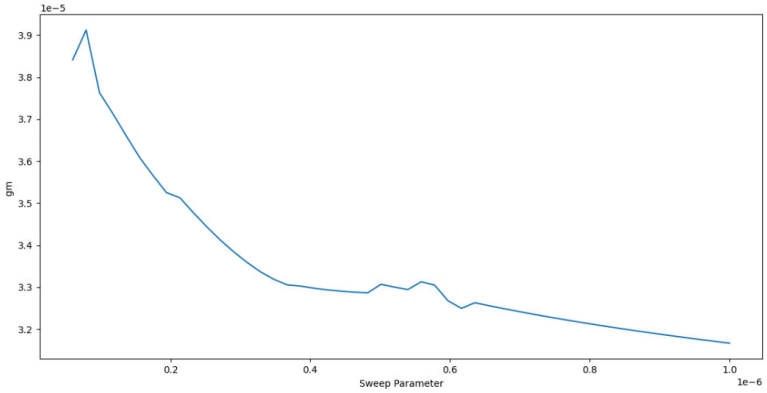
O transistor PMOS é dimensionado da mesma forma, já que os parâmetros gm, gds e I_D devem ser os mesmos. A Figura 31 mostra a curva gm x L do transistor PMOS. Para $L = 194\,nm$, obtém-se aproximadamente os valores desejados de gm e gds.

Os parâmetros obtidos para os transistores NMOS e PMOS do amplificador push-pull podem ser vistos na tabela 7.

Tabela 7 – Resultados obtidos para o projeto do amplificador push-pull.

Parâmetro	NMOS	PMOS
$gm(\mu S)$	32,04	35,67
$gds(\mu S)$	1,10	0,98
$I_D(\mu A)$	3,59	3,59
$L(nm)$	232	194
$W(nm)$	131	339

Figura 31 – Gráfico de gm x L para o transistor PMOS do amplificador push-pull.



A Figura 32 mostra o circuito do amplificador push-pull após a simulação de ponto de operação, e a Figura 33 exibe a resposta em frequência do circuito.

Figura 32 – Amplificador push-pull após a simulação de ponto de operação no software Cadence Virtuoso®.

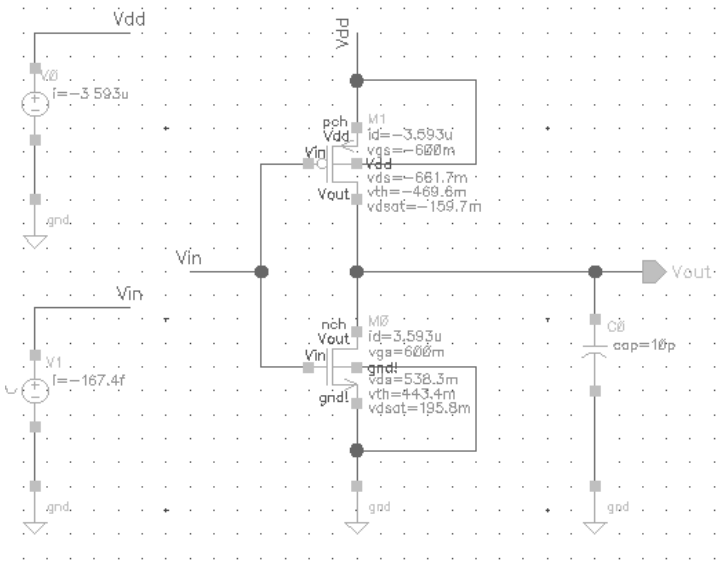
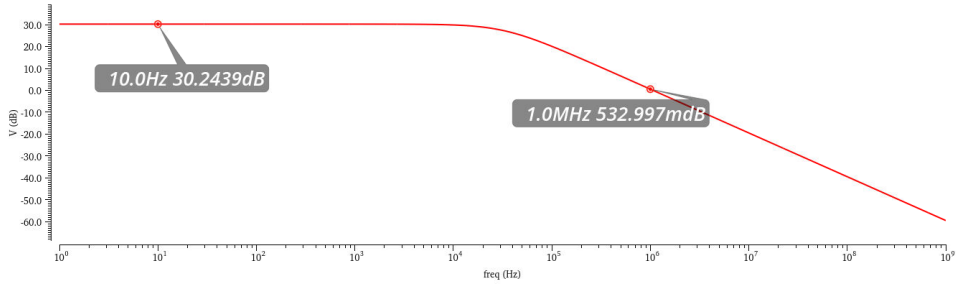


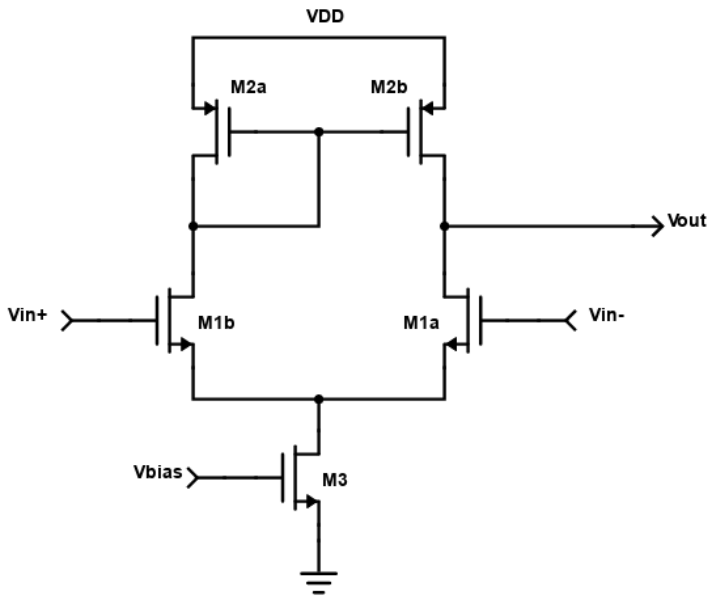
Figura 33 – Resposta em frequência do amplificador push-pull projetado.



4.3 Amplificador Diferencial

O amplificador diferencial, é uma topologia muito usada em circuitos analógicos, como filtros, osciladores, etc. De modo geral, consiste de dois pares de transistores PMOS e NMOS, em que o par de transistores PMOS funcionam como um espelho de corrente, enquanto os transistores NMOS operam o a transcondutância. Um quinto transistor é usado como fonte de corrente, como pode ser visto na Figura 34. Na análise de pequenos

Figura 34 – Esquemático do amplificador diferencial.



sinais, as fontes DC são consideradas iguais a zero. A tensão V_{DD} se comporta como um curto-circuito e é substituída por um terminal *ground*. O transistor $M3$ é visto como uma fonte de corrente, que neste caso, age como um circuito aberto. As tensões V_{GS} nos transistores $M1_a$ e $M1_b$ são iguais, portanto serão representadas como V_{GS1}. Este circuito é representado na forma vista na Figura 35 (a).

Na Figura 35 (b), é representado o circuito equivalente do modelo de pequenos sinais. Nota-se que, usando a Lei de Kirchhoff das tensões na malha com as tensões V_{IN}, V_{GS} e V_X nos transistores $M1_a$ e $M1_b$, obtemos as equações 4.5 e 4.6.

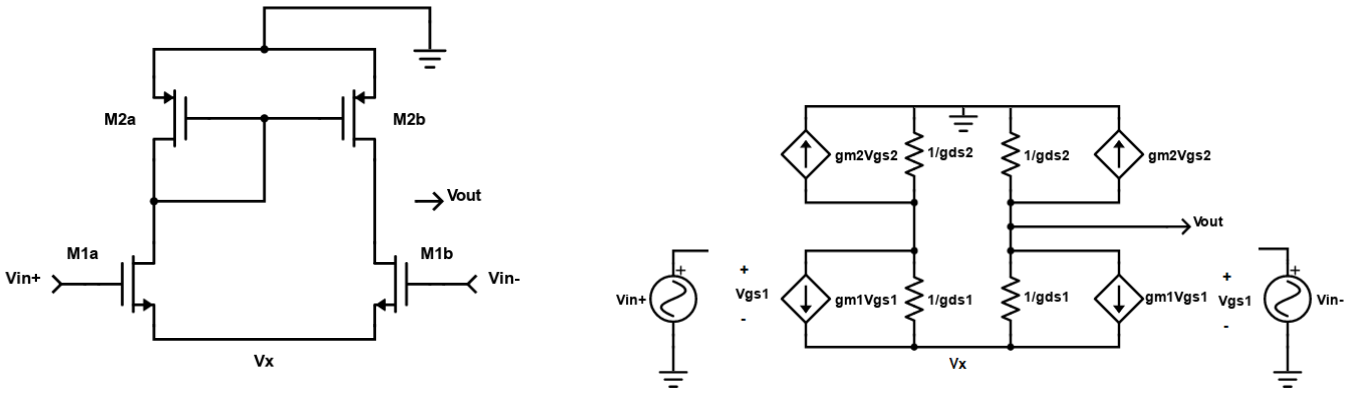
$$V_{IN+} - V_{GS} = V_X \tag{4.5}$$

$$V_{IN-} - V_{GS} = V_X \tag{4.6}$$

Substituindo a tensão V_X e sabendo que $V_{IN+} = -V_{IN-}$, temos a relação 4.7.

$$2V_{IN+} = 2V_{GS} \tag{4.7}$$

Figura 35 – Circuito equivalente na análise de pequenos sinais do amplificador diferencial.



(a) (b)

Analisando as correntes que chegam ao nó que possui a tensão V_X, temos que a corrente neste nó é a soma das correntes $I_{DS_{M1a}}$ e $I_{DS_{M1b}}$. Assim, conseguimos obter a relação 4.8.

$$I_{DS_{M1a}} + I_{DS_{M1b}} = gm_{1a}V_{GS1} + gm_{1b}V_{GS1} = 0 \tag{4.8}$$

Sabendo que $gm_{1a} = gm_{1b}$, temos que $V_{GS1} = 0$. Da equação 4.7, obtemos que $V_{IN} = 0$. Dessa forma, nota-se que das equações 4.5 e 4.6, $V_X = 0$. Assim, concluímos que o nó onde está a tensão V_X se comporta como um terminal *ground*. O circuito então pode ser simplificado para ser analisado separadamente, de modo que o amplificador diferencial terá um comportamento aproximadamente igual ao do amplificador de fonte comum. As equações 4.1 e 4.2 que representam o ganho e GBW do amplificador de fonte comum, então, se tornam válidas para o amplificador diferencial.
Este projeto será realizado para os ganhos de 26dB e 40dB respectivamente, de modo a variar os resultados em relação aos projetos anteriores. As especificações dos projetos do amplificador diferencial são mostradas na tabela 8.

Tabela 8 – Parâmetros utilizados para o projeto do amplificador diferencial.

Parâmetro	26 dB	40 dB
$V_{DD}(V)$	1,2	1,2
$GBW(MHz)$	4	10
$C_L(pF)$	2	5

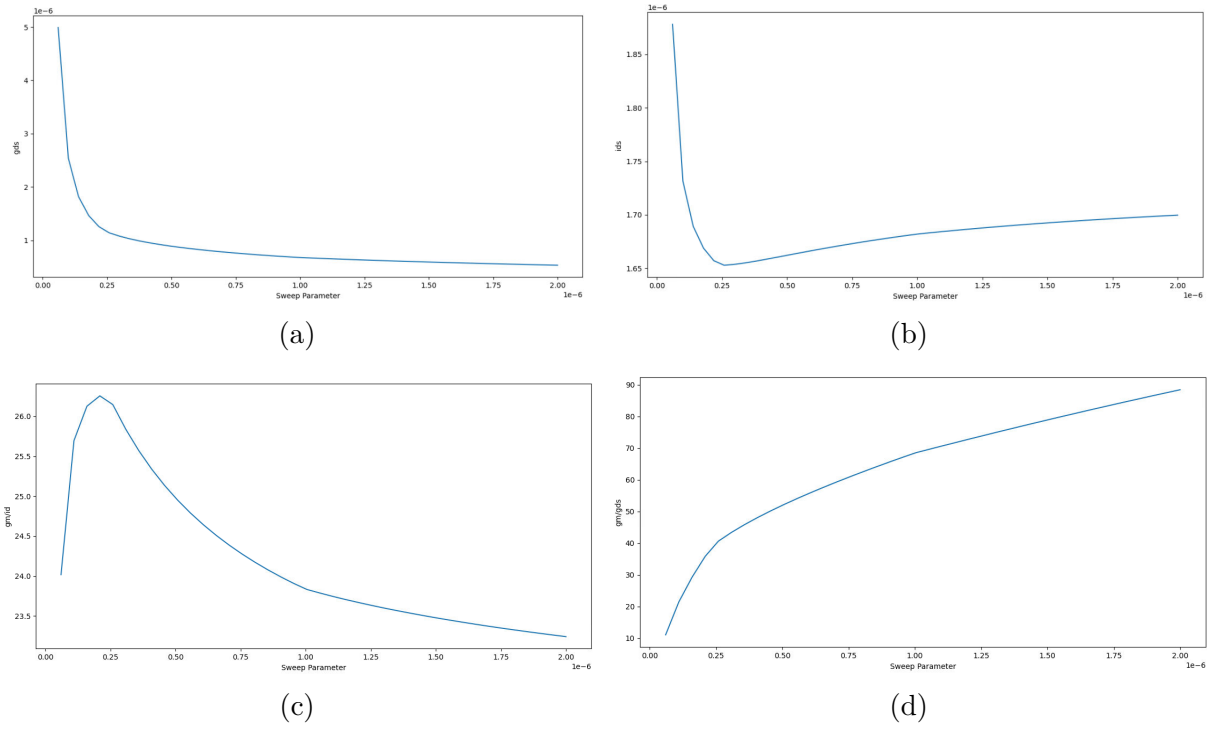
Inicialmente, para o projeto com $A_V = 26$ dB, é necessário determinar os parâmetros gm dos transistores $M1_a$ e $M1_b$, além de gds dos transistores $M1_a$, $M1_b$, $M2_a$ e $M2_b$. Os parâmetros gm, gds_{M1} e gds_{M2} portanto, são:

$$gm_{M1} = 50,2\,\mu S$$

$$gds_{M1} + gds_{M2} \approx 2,5\,\mu S$$

Inicialmente, os transistores $M1_a$ e $M1_b$ são dimensionados com a ferramenta de varredura, a fim de buscar gm_{M1}. Então, é necessário determinar os valores de gds e I_D para projetar os transistores $M2_a$ e $M2_b$.
A Figura 36, mostram as relações entre I_D x L e gds x L para os transistores $M1_a$ e $M1_b$.

Figura 36 – Relações entre (a) gds x L, (b) I_D x L, (c) gm/I_D x L, e (d) gm/gds x L para os transistores $M1_a$ e $M1_b$.



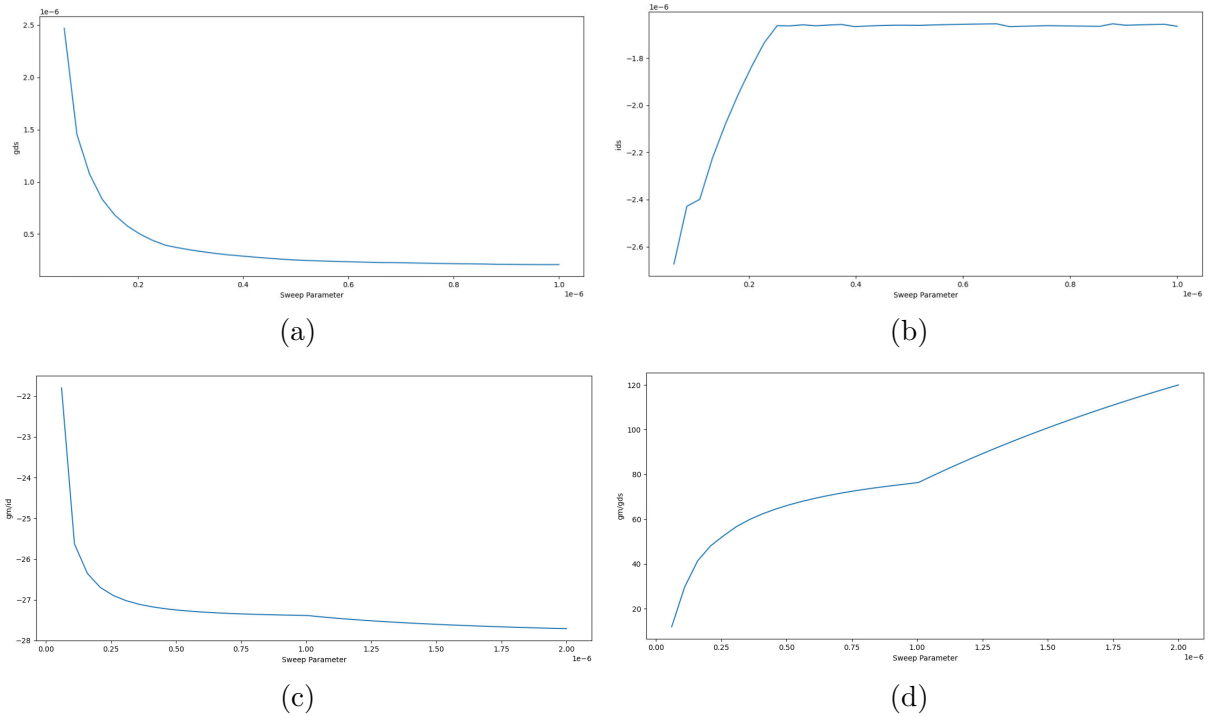
Somente a partir destas curvas, não é possível determinar os valores de W e L para $M1_a$ e $M1_b$, sendo necessário analisar as curvas de $M2_a$ e $M2_b$ para escolher os parâmetros mais adequados.
Os gráficos de gds x L e I_D x L para $M2_a$ e $M2_b$ podem ser vistos na Figura 37.

Analisando as curvas obtidas para os transistores, é possível escolher os parâmetros de dimensionamento.
A corrente no transistor $M3$ é igual a soma das correntes em $M1_a$ e $M1_b$. Portanto, $M3$ é dimensionado para $I_{D3} = 2I_D$.
Os resultados obtidos são mostrados na tabela 9.

A resposta em frequência do amplificador diferencial projetado é vista na Figura 38.

Considerando o projeto para um ganho $A_V = 40$ dB, os parâmetros gm e gds

Figura 37 – Relações entre (a) gds x L, (b) I_D x L, (c) gm/I_D x L, e (d) gm/gds x L para os transistores $M2_a$ e $M2_b$.



(a) (b)

(c) (d)

Tabela 9 – Resultados obtidos para o projeto do amplificador diferencial para um ganho $A_V = 26$ dB.

Parâmetro	$M1_{a,b}$	$M2_{a,b}$	$M3$
W(µm)	12,7	0,12	0,24
L(nm)	125	250	200
I_D(µA)	1,99	1,99	3,98
gm(µS)	52,6	Não se aplica	Não se aplica
gds(µS)	2,07	0,45	Não se aplica

devem ser recalculados. Os valores obtidos são mostrados abaixo.

$$gm_{M1} = 31.14\,\mu\text{S}$$

$$gds_{M1} + gds_{M2} = 0.31\,\mu\text{S}$$

De forma semelhante, os transistores $M1_a$ e $M1_b$ são projetados inicialmente para o valor de gm calculado. A corrente $I_D = 13.86\,\mu\text{A}$ então é obtida, e os transistores $M2_a$ e $M2_b$ são projetados com o parâmetro objetivo sendo I_D. Com curvas obtidas, é possível obter o dimensionamento para os transistores a partir dos valores de gds. A figura 39 mostra as curvas gds x L para (a) $M1_a$ e $M1_b$, (b) $M2_a$ e $M2_b$.O transistor $M3$ é projetado para $2I_D$.

Devido ao GBW neste projeto ser maior que anteriormente, é necessário levar em consideração as capacitâncias parasitas presentes no circuito. Desta forma, o valor da

Figura 38 – Resposta em frequência do amplificador diferencial com um ganho $A_V = 26$ dB.

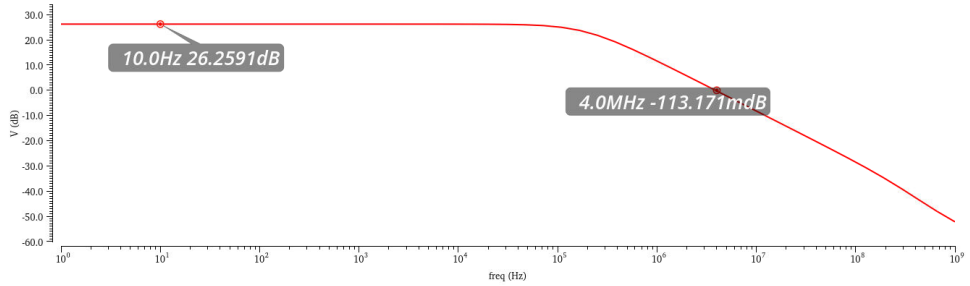
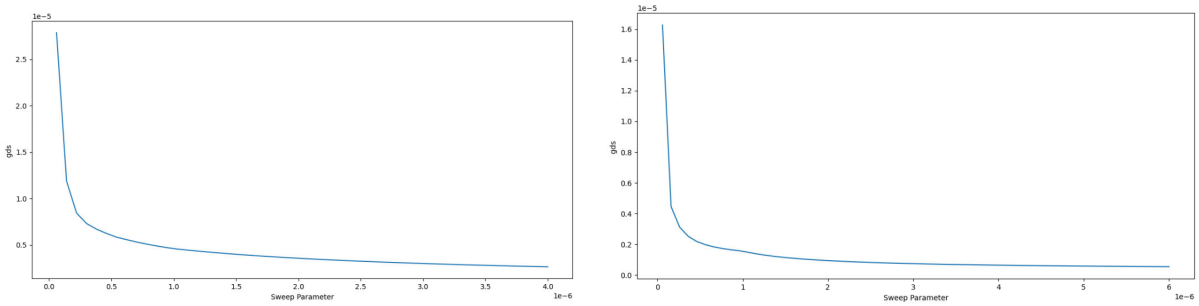


Figura 39 – Curvas gds x L para o amplificador diferencial de $A_V = 40$ dB para os transistores (a) $M1_a$ e $M1_b$, (b) $M2_a$ e $M2_b$.



(a) (b)

capacitância total será:

$$C_{TOT} = C_L + C_{PAR} \tag{4.9}$$

Onde $C_{TOT} = 5pF$, C_L é a carga capacitiva e C_{PAR} é a soma das capacitâncias parasitas presentes.

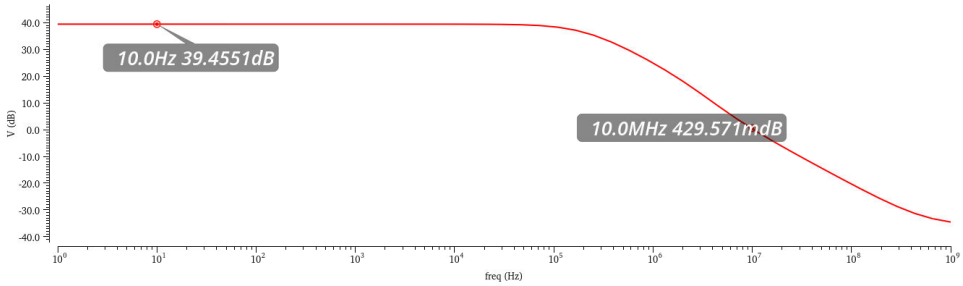
As capacitâncias parasitas podem ser obtidas diretamente nas ferramentas desenvolvidas, e é determinado que $C_{PAR} \approx 3pF$. Assim, pela equação 4.9, o valor da carga capacitiva deve ser ajustado para $C_L = 2pF$.

Os parâmetros obtidos após os dimensionamentos são exibidos na tabela 10. A figura 40 mostra a resposta em frequência obtida.

Tabela 10 – Resultados obtidos para o projeto do amplificador diferencial com um ganho $A_V = 40$ dB.

Parâmetro	$M1_{a,b}$	$M2_{a,b}$	$M3$
W(µm)	305	28,9	1,15
L(µm)	4,5	4,1	0,06
I_D(µA)	13,86	13,86	27,72
gm(µS)	322	Não se aplica	Não se aplica
gds(µS)	2,21	1,18	Não se aplica

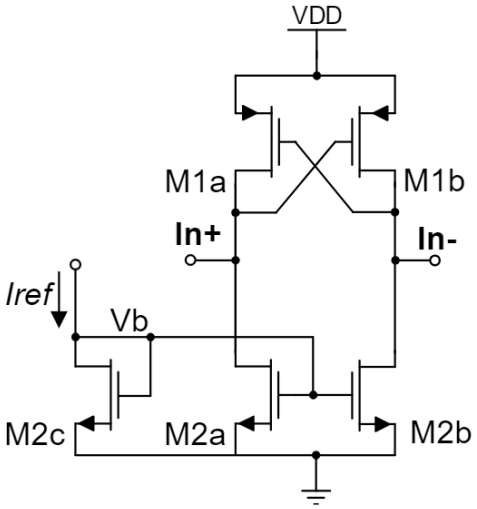
Figura 40 – Resposta em frequência para o amplificador diferencial com um ganho $A_V = 40$ dB.



4.4 Transdutor Negativo

O transdutor negativo de acoplamento cruzado é um circuito muito importante para diversas aplicações. Em amplificadores e filtro de ultra-baixa tensão, são empregados para melhorar a largura de banda e o ganho de tensão (GIRARDI; AGUIRRE; SEVERO, 2022). A Figura 41 mostra o esquemático do transdutor negativo de acoplamento cruzado. O circuito do transdutor negativo foi projetado para as tecnologias de 180nm e 65nm, a fim de comparar os resultados.

Figura 41 – Esquemático do circuito de um transdutor negativo de acoplamento cruzado.



Fonte: (GIRARDI; AGUIRRE; SEVERO, 2022).

Nesta topologia, os transistores PMOS $M1_a$ e $M1_b$ implementam a transcondutância negativa, enquanto os transistores NMOS $M2_a$, $M2_b$ e $M2_c$ operam como fontes de corrente para polarizar o transcondutor. Para o modelo de pequenos sinais, a transcondutância negativa pode ser obtida a partir da equação 4.10:

$$gm_{neg} = -gm_1 + gds_1 + gds_2 \tag{4.10}$$

Para obter o valor de gm_{neg}, é necessário reescrever a equação 4.10 como função

de gm_1. A estratégia para o projeto deste circuito consiste em assumir que ambos os transistores possuem a mesma corrente de dreno ($I_{D_{M1}} = I_{D_{M2}} = I_{ref}$), e utilizar os parâmetros gm/I_D e gm/gds, como mostrado nas equações 4.11, 4.12 e 4.13.

$$gds_1 = \left[\frac{1}{\left(\frac{gm}{gds}\right)_1}\right] gm_1 \tag{4.11}$$

$$gds_2 = \left[\frac{1}{\left(\frac{gm}{gds}\right)_2}\right] gm_2 = \left[\frac{\left(\frac{gm}{I_D}\right)_2}{\left(\frac{gm}{gds}\right)_2}\right] I_{D_{M2}} \tag{4.12}$$

$$I_{D_{M2}} = I_{D_{M1}} = \left[\frac{1}{\left(\frac{gm}{I_D}\right)_1}\right] gm_1 \tag{4.13}$$

A partir destas suposições, podemos obter gm_{neg} na equação 4.14 substituindo as equações 4.11, 4.12 e 4.13 em 4.10.

$$gm_1 = -\frac{gm_{neg}}{\left[1 - \frac{1}{\left(\frac{gm}{gds}\right)_1} - \frac{\left(\frac{gm}{I_D}\right)_2}{\left(\frac{gm}{gds}\right)_2\left(\frac{gm}{I_D}\right)_1}\right]} \tag{4.14}$$

Desta forma, os transistores PMOS devem ser projetados primeiro, de modo a obter gm, e em seguida os transistores NMOS são projetados a partir da corrente I_{DS} obtida no transistor PMOS.

Para este projeto, é necessário realizar o dimensionamento dos transistores em sequência. O transistor PMOS é simulado com o parâmetro objetivo sendo a transcondutância $gm = 10\,\mu S$, e é indicado que existe um parâmetro de saída da função, neste caso sendo a corrente I_{DS}. Em seguida, o transistor NMOS é dimensionado, com o objetivo de buscar a corrente I_{DS} encontrada na simulação anterior.

A Figura 42 demonstra o método de dimensionamento usado para os transistores PMOS e NMOS necessários para o projeto do transdutor negativo de acoplamento cruzado. O trecho do código-fonte que mostra os parâmetros da função *main* pode ser visto mo Apêndice A.3.

Com a finalidade de comparação do projeto do transdutor negativo com outro semelhante encontrado na literatura, o objetivo será a obtenção de uma transcondutância igual a $-10\,\mu S$. Para isto, serão adotados os seguintes parâmetros para as simulações:

O projeto desenvolvido na ferramenta de dimensionamento pode ser visto na Figura 43, considerando a tecnologia de 180 nm. Foram usados modelos de transistores low-V_T.

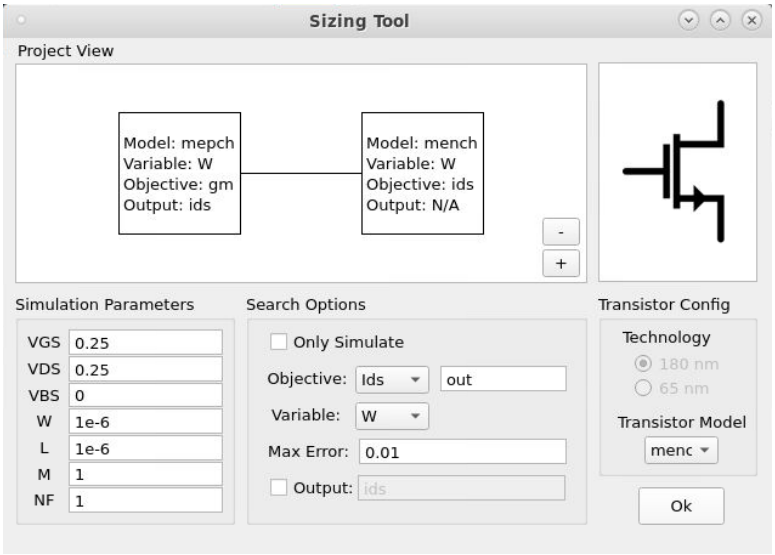
Figura 42 – Ilustração do processo utilizado para o projeto do transdutor negativo de acoplamento cruzado.

Fonte: Autor

Tabela 11 – Parâmetros do projeto do transdutor negativo para a tecnologia de 180 nm.

Parâmetro	PMOS	NMOS
$V_{GS}(V)$	0,25	0,25
$V_{DS}(V)$	0,25	0,25
$V_{BS}(V)$	0	0
W(µm)	Variável	Variável
L(µm)	1	1
gm(µS)	Objetivo	Não se aplica
I_{DS}(nA)	Não se aplica	Objetivo
$Erro_{MAX}(\%)$	1	1

Figura 43 – Projeto do transdutor negativo de acoplamento cruzado na ferramenta de dimensionamento.



Para o projeto na tecnologia de 65 nm, foi analisada a curva de gm x L gerada na ferramenta de varredura. É definido que $L \approx 305\,\text{nm}$ para todos os transistores. O restante do projeto é feito de forma semelhante ao da tecnologia de 180 nm.

Os resultados obtidos no projeto do transdutor negativo de acoplamento cruzado podem ser vistos na Tabela 12, onde os parâmetros obtidos são comparados com outros encontrados

na literatura, de modo a demonstrar a eficiência da ferramenta de dimensionamento de transistores.

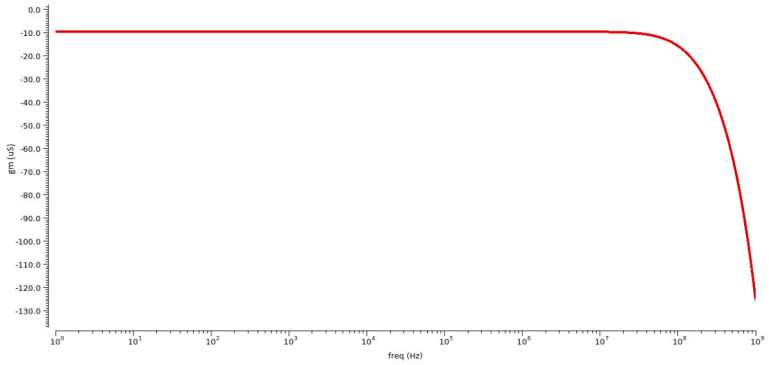
Tabela 12 – Comparação dos resultados obtidos com resultados da literatura.

Parâmetro	180 nm	65 nm	[1]
W_1	**2,**12 µm	9,76 µm	2,17 µm
W_2	**5,**12 µm	2,23 µm	5,9 µm
I_{ref}	**556,**99 nA	353 nA	580,5 nA

[1] - (GIRARDI; AGUIRRE; SEVERO, 2022).

A resposta em frequência da transcondutância negativa é mostrada na Figura 44. Como pode ser observado, o valor de $-10\,\mu S$ é mantido constante até cerca de 25 MHz para todos os casos.

Figura 44 – Resposta em frequência do transdutor negativo de acoplamento cruzado projetado.



4.5 Síntese do Capítulo

Este capítulo apresentou a aplicação das ferramentas de dimensionamento de transistores CMOS desenvolvidas em diversos projetos. Foram demostrados desde a análise dos circuitos propostos, os respectivos projetos, até os resultado obtidos. As tabelas 6, 7, 9 e 12 mostram os parâmetros obtidos após o dimensionamento dos transistores, e para todos os projetos, é mostrada a resposta em frequência, validando o fluxo de projeto da ferramenta.

5 Conclusão

Neste trabalho, foi apresentada uma ferramenta de dimensionamento de transistores CMOS baseada nas linguagens de programação Python e Cadence SKILL, e no simulador Cadence Spectre®.

Primeiramente, uma breve revisão bibliográfica sobre projetos de circuitos analógicos foi apresentada, no qual é verificado a necessidade de ferramentas para o auxílio do projeto de transistores. Partindo deste contexto, uma ferramenta para o dimensionamento de MOSFETs foi apresentada.

Com a utilização de uma função de custo e uma função de ajuste, foi possível automatizar as simulações dos transistores NMOS e PMOS de modo que haja uma busca pelo parâmetro desejado, ocorrendo de forma iterativa. A ferramenta de dimensionamento automático de transistores CMOS proposta possui atualmente compatibilidade com as tecnologias de fabricação CMOS de 180 nm e 65 nm. Devido ao fato da ferramenta usar o simulador e não usar equações, é possível implementar o projeto em qualquer tecnologia, bastando apenas fornecer os modelos e configurações.

Todos os circuitos propostos foram projetados, e a partir do fluxo de projeto da ferramenta proposta, foram obtidos resultados satisfatórios.

Como sugestão de trabalhos futuros, é interessante desenvolver mais funcionalidades para a ferramenta, como funções para automatizar o dimensionamento dos componentes de blocos analógicos, e adicionar a compatibilidade com outras tecnologias de fabricação CMOS.

Referências

ABBAS, Z.; OLIVIERI, M. Optimal transistor sizing for maximum yield in variation-aware standard cell design. *International Journal of Circuit Theory and Applications*, v. 44, n. 7, p. 1400–1424, 2016. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cta.2167>. Citado na página 1.

BALKIR, S.; DÜNDAR, G.; ÖGRENCI, A. *Analog VLSI Design Automation*. CRC Press, 2003. (VLSI Circuits). ISBN 9781135515430. Disponível em: <https://books.google.com.br/books?id=I02uDwAAQBAJ>. Citado na página 3.

BARNES, T. Skill: a cad system extension language. In: *27th ACM/IEEE Design Automation Conference*. [S.l.: s.n.], 1990. p. 266–271. ISSN 0738-100X. Citado na página 16.

CADENCE DESIGN SYSTEMS. *Spectre® Circuit Simulator User Guide*. 555 River Oaks Parkway, San Jose, CA 95134, USA, 2002. Citado na página 14.

GIRARDI, A.; AGUIRRE, P. C.; SEVERO, L. Design techniques for ultra-low voltage analog circuits using cmos characteristic curves: a practical tutorial. *Journal of Integrated Circuits and Systems*, v. 17, n. 1, 2022. Citado 3 vezes nas páginas 6, 34 e 37.

LEENAERTS, D.; GIELEN, G.; RUTENBAR, R. Cad solutions and outstanding challenges for mixed-signal and rf ic design. In: *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281)*. [S.l.: s.n.], 2001. p. 270–277. Citado na página 1.

MARTENS, E.; GIELEN, G. Classification of analog synthesis tools based on their architecture selection mechanisms. *Integration*, v. 41, n. 2, p. 238–252, 2008. ISSN 0167-9260. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167926007000417>. Citado na página 4.

MINA, R.; JABBOUR, C.; SAKR, G. E. A review of machine learning techniques in analog integrated circuit design automation. *Electronics*, v. 11, n. 3, 2022. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/11/3/435>. Citado na página 2.

MOORE, G. E. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, v. 11, n. 3, p. 33–35, 2006. Citado na página 1.

RODRIGUEZ, S. et al. A comprehensive graphene fet model for circuit design. *IEEE Transactions on Electron Devices*, v. 61, n. 4, p. 1199–1206, 2014. Citado na página 9.

SEVERO, L. C. et al. Simulated annealing to improve analog integrated circuit design: Trade-offs and implementation issues. In: _____. *Simulated Annealing - Single and Multiple Objective Problems*. IntechOpen, 2012. p. 261–283. Disponível em: <https://doi.org/10.5772/2565>. Acesso em: 23 Jun 2023. Citado na página 4.

SEVERO, L. C.; GIRARDI, A. G. *Um framework para o dimensionamento automático de blocos analógicos básicos integrados*. Dissertação (Trabalho de Conclusão de Curso) — Universidade Federal do Pampa, 2011. Citado na página 5.

SEVERO, L. C.; GIRARDI, A. G. *Uma Ferramenta para o Dimensionamento Automáticode Circuitos Integrados Analógicos Considerando Análisede Produtividade*. Dissertação (Dissertação de Mestrado) — Universidade Federal do Pampa, 2012. Citado na página 3.

SEVERO, L. C.; NOIJE, W. A. M. V. *Metodologia Baseada em Otimização para o Projeto de Circuitos Analógicos CMOS de Ultra-Baixa Tensão Considerando os Efeitos da Variabilidade*. 114 p. Monografia (Doutorado) — Universidade de São Paulo, São Paulo, 2016. Citado na página 4.

SEVERO, L. C.; NOIJE, W. V. *ULV and ULP active-RC filters combining single-stage OTA and negative input transconductance for low energy RF receivers*. Tese (Doutorado) — Universidade de São Paulo, São Paulo, 2019. Disponível em: <https://teses.usp.br/teses/disponiveis/3/3140/tde-20032019-101049/publico/LucasCompassiSeveroCorr19.pdf>. Acesso em: 17 jan 2023. Citado na página 8.

SHOCKLEY, W.; BARDEEN, J.; BRATTAIN, W. *The Nobel Prize in Physics 1956*. 1956. Disponível em: <https://www.nobelprize.org/prizes/physics/1956/summary/>. Citado na página 1.

Apêndices

APÊNDICE A – CÓDIGO-FONTE

A.1 Código-fonte da ferramenta desenvolvida

O código-fonte dos arquivos principais da ferramenta são listados a seguir.

A.1.1 main.py

```
import fcall_nmos
import fcall_pmos
import simulation
import functions
import os

home_directory = os.path.expanduser( '~' )

def main(technology, model_name, no_op = False, out_param=None, **kwargs):

    pmos, nmos = functions.model(technology, model_name)

    if nmos == True:
        functions.edit_input(str(home_directory) + '/Circuit
        Sizing Tool/files/inputs/input_nmos.txt', kwargs)
        input_dict, target_name, target_value, max_error, par_calc_name,
        par_calc_val, nit, size = functions.inputs(str(home_directory) +
        '/Circuit Sizing Tool/files/inputs/input_nmos.txt')

        if no_op == False:
            variables, values, par_ref, w = fcall_nmos.calc_w(target_name,
            target_value, max_error, par_calc_name, par_calc_val, input_dict,
            size, nit, technology)
            functions.results(technology, target_name, par_ref, par_calc_name,
            w, variables, values)

        elif no_op == True:
            variables, values = simulation.simulate(technology, input_dict)
            target_index = functions.search_index(target_name, variables)
```

```
            par_ref = values[target_index]
            functions.results(technology, target_name, par_ref, par_calc_name,
            par_calc_val, variables, values)

if out_param != None:
out_n = functions.out_parameter(out_param, variables, values)
return out_n

    if pmos == True:
        functions.edit_input(str(home_directory) + '/Circuit Sizing
        Tool/files/inputs/input_pmos.txt', kwargs)
        input_dict, target_name, target_value, max_error, par_calc_name,
        par_calc_val, nit, size = functions.inputs(str(home_directory) +
        '/Circuit Sizing Tool/files/inputs/input_pmos.txt')

        if no_op ==  False:
            if target_name == 'ids':
                target_value = -target_value

            variables, values, par_ref, w = fcall_pmos.calc_w(target_name,
            target_value, max_error, par_calc_name, par_calc_val, input_dict,
            size, nit, technology)
            functions.results(technology, target_name, par_ref, par_calc_name,
            w, variables, values, pmos=True)

        elif no_op == True:
            variables, values = simulation.simulate(technology, input_dict,
            pmos=True)
            target_index = functions.search_index(target_name, variables)
            par_ref = values[target_index]
            functions.results(technology, target_name, par_ref, par_calc_name,
            par_calc_val, variables, values, pmos=True)

        if out_param != None:
            out_p = functions.out_parameter(out_param, variables, values)
            return out_p
```

A.1.2 simulation.py

```
import os
import sys

home_directory = os.path.expanduser( '~' )

def simulate(technology, kwargs, pmos=False):
    if pmos == False:
        fileTxt_n = open(str(home_directory) + "/Circuit Sizing
        Tool/files/parametros_n.txt", 'w')

        var = []

        for key, value in zip(kwargs.keys(),kwargs.values()):
            var.append(str(key) + "=" + str(value))

        sys.stdout = fileTxt_n
        print("//parametros da simulacao")
        print("simulator lang=spectre")
        print("parameters", end=" ")
        print(*var)
        sys.stdout = sys.__stdout__
        fileTxt_n.close()

        if technology == '180nm':
            os.system('spectre -64 ' + str(home_directory) + '/Circuit\
            Sizing\ Tool/netlist/tsmc180/nmos_dc180.scs -format psfascii')
            arq = open(str(home_directory) + '/Circuit Sizing
            Tool/netlist/tsmc180/nmos_dc180.raw/dcOpInfo.info', 'r')
        elif technology == '65nm':
            os.system('spectre -64 ' + str(home_directory) + '/Circuit\
            Sizing\ Tool/netlist/tsmc65/nmos_dc65.scs -format psfascii')
            arq = open(str(home_directory) + '/Circuit Sizing
            Tool/netlist/tsmc65/nmos_dc65.raw/dcOpInfo.info', 'r')

    if pmos == True:
        fileTxt_p = open(str(home_directory) + "/Circuit Sizing
        Tool/files/parametros_p.txt", 'w')
```

```
    var = []

    for key, value in zip(kwargs.keys(),kwargs.values()):
        var.append(str(key) + "=" + str(value))

    sys.stdout = fileTxt_p
    print("//parametros da simulacao")
    print("simulator lang=spectre")
    print("parameters", end=" ")
    print(*var)
    sys.stdout = sys.__stdout__
    fileTxt_p.close()

    if technology == '180nm':
        os.system('spectre -64 ' + str(home_directory) + '/Circuit\
        Sizing\ Tool/netlist/tsmc180/pmos_dc180.scs -format psfascii')
        arq = open(str(home_directory) + '/Circuit Sizing
        Tool/netlist/tsmc180/pmos_dc180.raw/dcOpInfo.info', 'r')
    elif technology == '65nm':
        os.system('spectre -64 ' + str(home_directory) + '/Circuit\
        Sizing\ Tool/netlist/tsmc65/pmos_dc65.scs -format psfascii')
        arq = open(str(home_directory) + '/Circuit Sizing
        Tool/netlist/tsmc65/pmos_dc65.raw/dcOpInfo.info', 'r')


variables = []
values = []
index = 0


text = arq.readlines()
for line in text:
    for word in line.split():
        index = word.find("FLOAT")
        index2 = word.find("INT")
        if index != -1:
            variables.append(line.split()[0])
        if index2 != -1:
```

```
                variables.append(line.split()[0])
    for i in range(442, 544):
        values.append(text[i])
    variables = [i.replace('"', '') for i in variables]

    return variables, values
```

A.1.3 fcall_nmos.py

```
import simulation as sm

def calc_w(par, ref, max_erro, var_name, var_value, inputs, size, nit,
technology):
erro = 999999999
w = float(var_value)
par_ref = ref
i = 0
while erro > max_erro and i < nit:
    w_last_it = w

    inputs.update({var_name:w})
    variables, values = sm.simulate(technology, inputs)
    par_index = variables.index(str(par))
    par_value = float(values[par_index])

    erro = abs(par_value - par_ref)/abs(par_ref)
    w = par_ref/par_value*w

    if var_name == 'L':
        if float(inputs['L']) <= size[0]:
            w = size[0]
        if float(inputs['L']) >= size[1]:
            w = size[1]
    if var_name == 'W':
        if float(inputs['W']) <= size[2]:
            w = size[2]
        if float(inputs['W']) >= size[3]:
            w = size[3]
```

```
    if w_last_it == w:
        print("Parameters out of range")
        return variables, values, par_value, w
        break

    i += 1
    print(i)

return variables, values, par_value, w
```

A.1.4 fcall_pmos.py

```
import simulation as sm

def calc_w(par, ref, max_erro, var_name, var_value, inputs, size, nit, technology):
erro = 999999999
w = float(var_value)
par_ref = ref
i = 0
while erro > max_erro and i < nit:
    w_last_it = w

    inputs.update({var_name:w})
    variables, values = sm.simulate(technology, inputs, pmos=True)
    par_index = variables.index(str(par))
    par_value = float(values[par_index])

    erro = abs(par_value - par_ref)/abs(par_ref)
    w = par_ref/par_value*w

    if var_name == 'L':
        if float(inputs['L']) <= size[0]:
            w = size[0]
        if float(inputs['L']) >= size[1]:
            w = size[1]
    if var_name == 'W':
        if float(inputs['W']) <= size[2]:
```

```
            w = size[2]
        if float(inputs['W']) >= size[3]:
            w = size[3]

    if w_last_it == w:
        print("Parameters out of range")
        return variables, values, par_value, w
        break

    i += 1
    print(i)

return variables, values, par_value, w
```

A.1.5 sweep.py

```
import numpy as np
import pandas as pd

import os, sys
home_directory = os.path.expanduser( '~' )
sys.path.append(str(home_directory) + '/Circuit Sizing Tool/files')

import main

def sweep(VarMin, VarMax, steps, sweepType, sweepVar, technology,
transistorModel, targetName, targetValue, maxError,w, l, nf, m, vgs,
vds, vbs, variable):

if sweepType == 'linear':
    sweep_variable = np.linspace(VarMin, VarMax, steps)
if sweepType == 'logarithmic':
    sweep_variable = np.logspace(VarMin, VarMax, steps)
parameter_defined = False

for step in sweep_variable:
    if sweepVar == 'L':
        main.main(technology, transistorModel, target_name=targetName,
```

```
    target_value=targetValue, max_error=maxError, W=w, L=step, NF=nf,
    M=m, VGS=vgs, VDS=vds, VBS=vbs, parameter_calc_name=variable)
if sweepVar == 'W':
    main.main(technology, transistorModel, target_name=targetName,
    target_value=targetValue, max_error=maxError,W=step, L=l, NF=nf,
    M=m, VGS=vgs, VDS=vds, VBS=vbs, parameter_calc_name=variable)
if sweepVar == 'VGS':
    main.main(technology, transistorModel, target_name=targetName,
    target_value=targetValue, max_error=maxError,W=w, L=l, NF=nf,
    M=m, VGS=step, VDS=vds, VBS=vbs, parameter_calc_name=variable)
if sweepVar == 'VDS':
    main.main(technology, transistorModel, target_name=targetName,
    target_value=targetValue, max_error=maxError,W=w, L=l, NF=nf,
    M=m, VGS=vgs, VDS=step, VBS=vbs, parameter_calc_name=variable)
if sweepVar == 'VBS':
    main.main(technology, transistorModel, target_name=targetName,
    target_value=targetValue, max_error=maxError,W=w, L=l, NF=nf,
    M=m, VGS=vgs, VDS=vds, VBS=step, parameter_calc_name=variable)

outputs = open(str(home_directory) + '/Circuit Sizing
Tool/files/outputs/out_sweep.txt', 'r')
out_txt = outputs.readlines()
outputs.close()

parameters = []
values = []

if parameter_defined == False:
    for line in out_txt:
        if '#' not in line:
            parameters.append(line.split("=")[0])
    parameters = [i for i in parameters if i != '\n']
    parameters[0] = 'Objective'
    parameters[1] = 'Variable'
    parameters.append('Sweep Parameter')
    dataframe = pd.DataFrame(columns = parameters)
parameter_defined = True

for line in out_txt:
```

```
        if '#' not in line:
            if line.split("=")[0] != '\n':
                values.append(line.split("=")[1][:-1])
    if values != 0:
        values.append(step)
        dataframe.loc[len(dataframe)] = values

dataframe.to_csv(str(home_directory) + '/Circuit Sizing
Tool/files/outputs/out_sweep.csv')
```

A.2 Código do arquivo SKILL para a integração da ferramenta no ambiente Virtuoso®

```
procedure( sizingTool()
        printf("Running Sizing Tool...")
        system("LD_LIBRARY_PATH=/lib /bin/python3 ~/SizingTool.py 65")

)
procedure( sweepTool()
        printf("Running Sweep Tool...")
        system("LD_LIBRARY_PATH=/lib /bin/python3 ~/SweepTool.py 65")

)


;#########################Criar menu no terminal CIW#############################
procedure(CreateCIWMenu()
let( (item1 item2 item3)
;; create a couple of menu items
item1 = hiCreateMenuItem( ?name 'item1 ?itemText "Transistor Sizing Tool"
          ?callback "sizingTool"
        )
item2 = hiCreateMenuItem( ?name 'item2 ?itemText "Parameter Sweep Tool"
          ?callback "sweepTool()"
        )

item3 = hiCreateMenuItem( ?name 'item3 ?itemText "About..."
         ?callback "aboutFunction()"
        )
```

```
;; create a menu that includes the menu items and return a list of the
;; pulldown menus
hiCreatePulldownMenu('AmitMenu "GAMA Tools" list(item1 item2 item3))
)
)

procedure(AmitAddMenu(args)
unless(boundp('AmitMenu)
CreateCIWMenu()
)
list(AmitMenu)
); procedure

procedure(AmitAddToCIW()
unless(boundp('AmitMenu)
CreateCIWMenu()
)
hiInsertBannerMenu(hiGetCIWindow() AmitMenu  3 )
)

;; create a userMenuTrigger trigger that automatically adds the menu
deRegUserTriggers("maskLayout" nil 'AmitAddMenu)
; may need to initialise CIW menus before adding
ciwMenuInit()
AmitAddToCIW()
```

A.3 Código fonte usado no projeto do transdutor negativo de acoplamento cruzado

```
out_p = main.main('180nm', 'mepch', target_name='gm', %target_value=10e-6,
max_error=0.01, L=1e-6, NF=1, M=1, VGS=0.25, VDS=0.25, VBS=0,
parameter_calc_name='W', out_param='ids')

main.main('180nm', 'mench', target_name='ids', target_value=-float(out_p),
max_error=0.01, L=1e-6, NF=1, M=1, VGS=0.25, VDS=0.25, VBS=0,
parameter_calc_name='W')
```

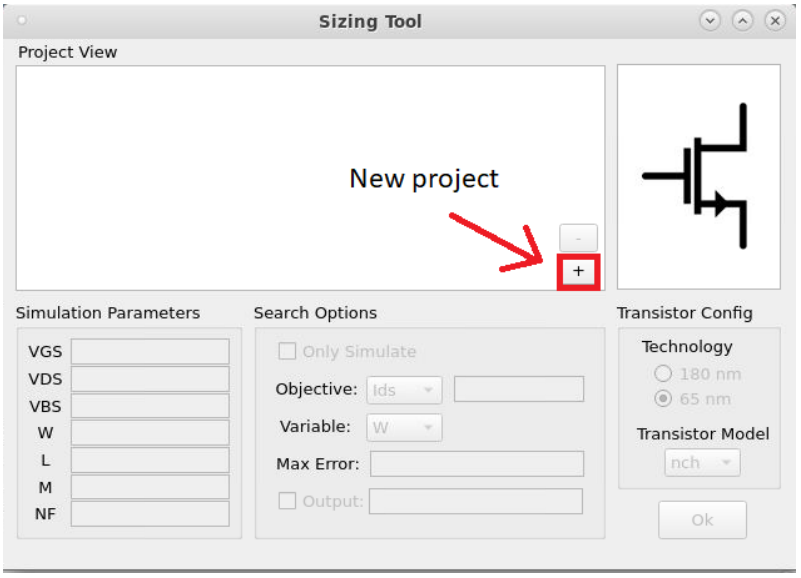
Anexos

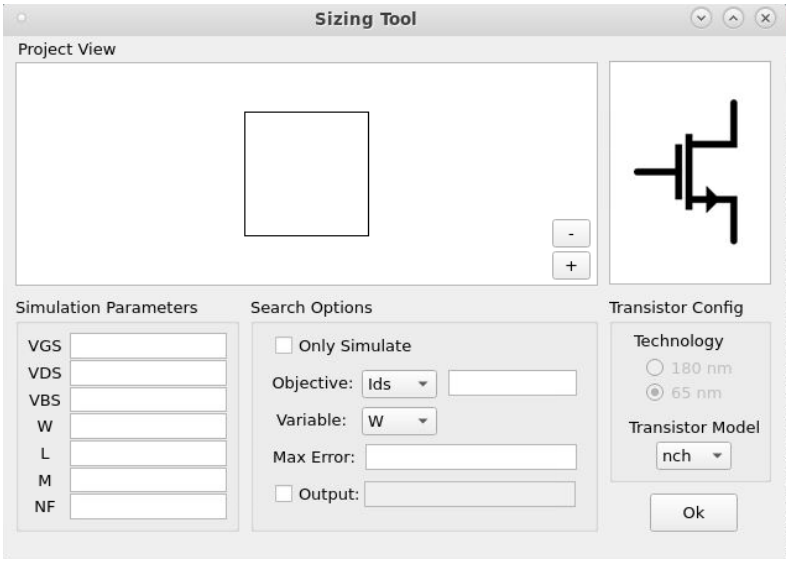
ANEXO A – Manual de uso da ferramenta (em inglês)

Sizing Tool Tutorial

To open the software, use the Cadence Virtuoso CIW menu.

The Sizing Tool window will be shown. To start a new project, click the “+” button:





After that, all the fields will be enabled. You can fill the text fields with the parameters of the transistor you want to simulate in the “Simulation Parameters” tab.
In the “Search Options” tab you will find the following parameters:

- **Only Simulate**: if the box is checked, the software will simulate the transistors with the parameters entered, but will not search for any specific parameter.
- **Objective**: You can select the parameter you want to search and then write the target value.
- **Variable**: This is the parameter that will be varying during the simulations.
- **Max Error**: The maximum error accepted for the project.
- **Output**: You can write the name of a parameter you want to use in the next simulation. See the examples for further explanation.

In the right side of the window, there is the transistor configurations tab. The technology cannot be switched after opening the software, since it depends directly on the virtuoso environment. If you want to change the technology, you must open the respective virtuoso environment for that technology. The transistor models depend on the technology used, for example, for 65 nm, the available models are “nch” (n-channel) and “pch” (p-channel).

If the parameters are not correctly entered, it may cause an error in the simulation process. If that is the case, you will see the following message:

If everything is typed correctly, the simulation process will occur with no major problems.

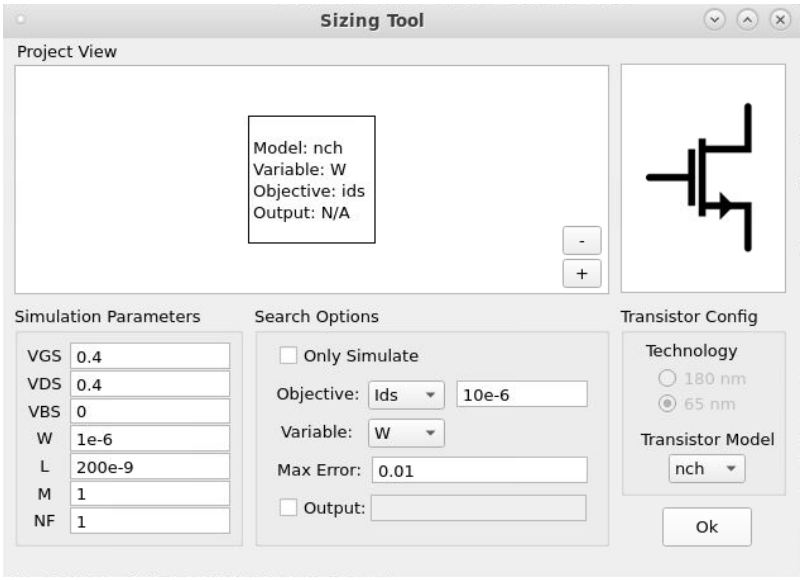
Examples:

1) In this example, a single transistor will be simulated, using the following parameters:

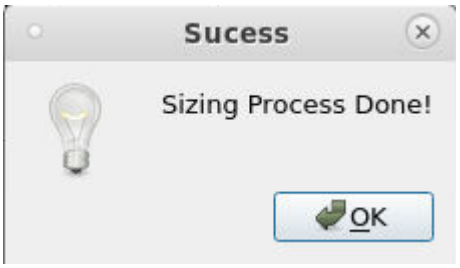
VGS	0.4 V
VDS	0.4 V
VBS	0 V
W	1e-6 m
L	200e-9 m
M	1
NF	1

The objective will be to find the value of **W** in a **NMOS** transistor on which we can get the current **Ids** = 10e-6 A.

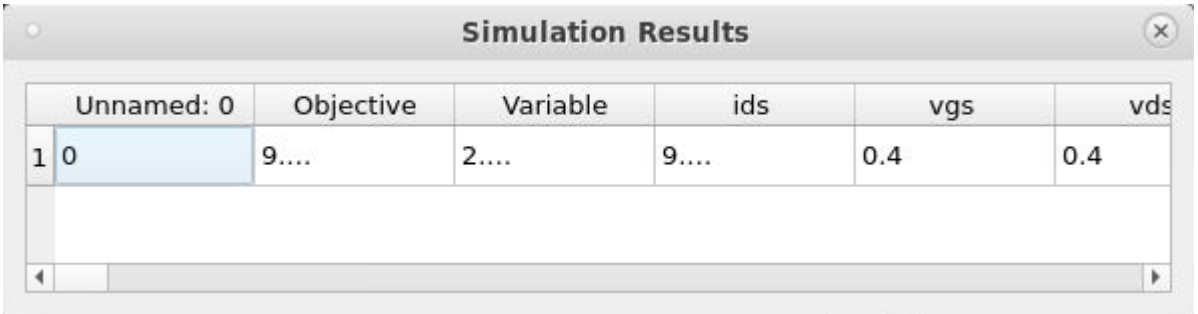
After typing the parameters into the respective fields, click the button “Ok”. The preview will show some parameters of the sizing process:



When the process is finished, a message will appear, confirming the successful search.



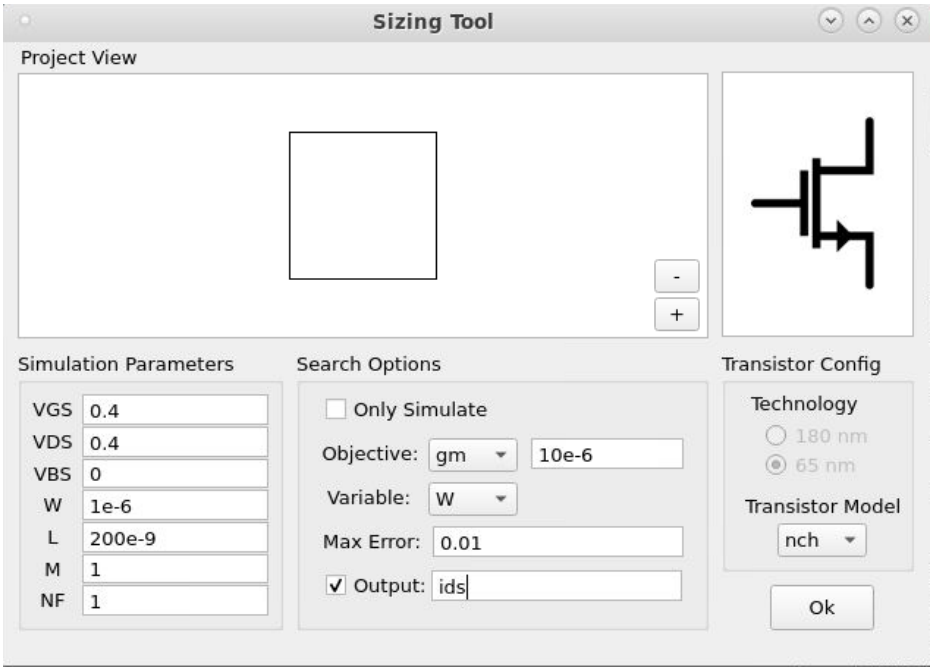
Then, the simulation results will be shown in a table:



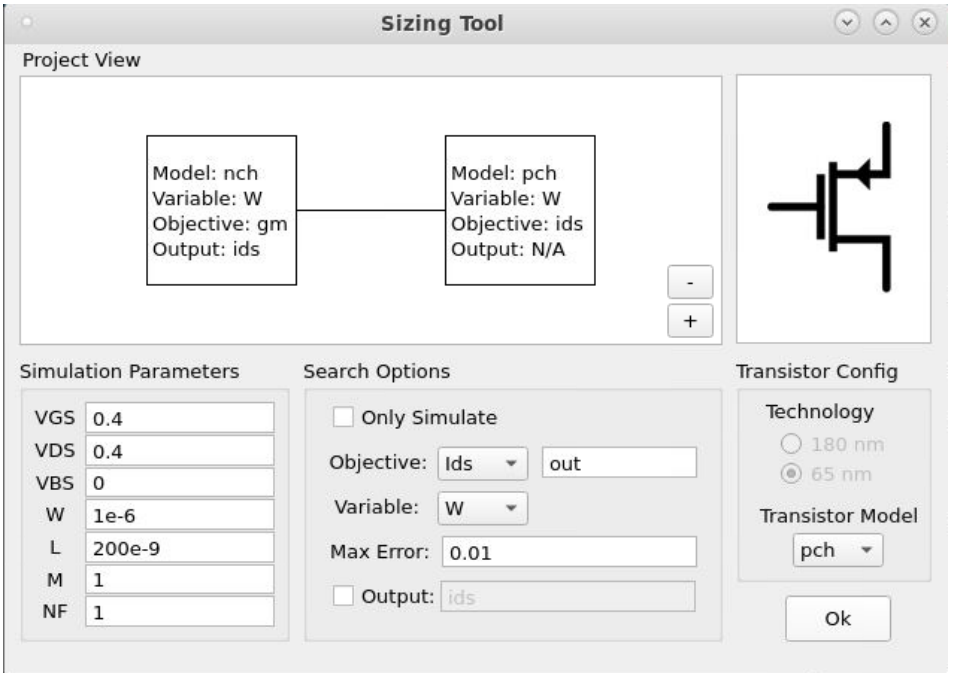
The file containing these values is located in the path “Circuit Sizing Tool/files/outputs”.

2) In this second example, we will simulate 2 transistors, a NMOS and a PMOS transistors, using the output parameter of the NMOS as the objective for the PMOS.

Using the same parameters as the first example, the project steps can be seen below:



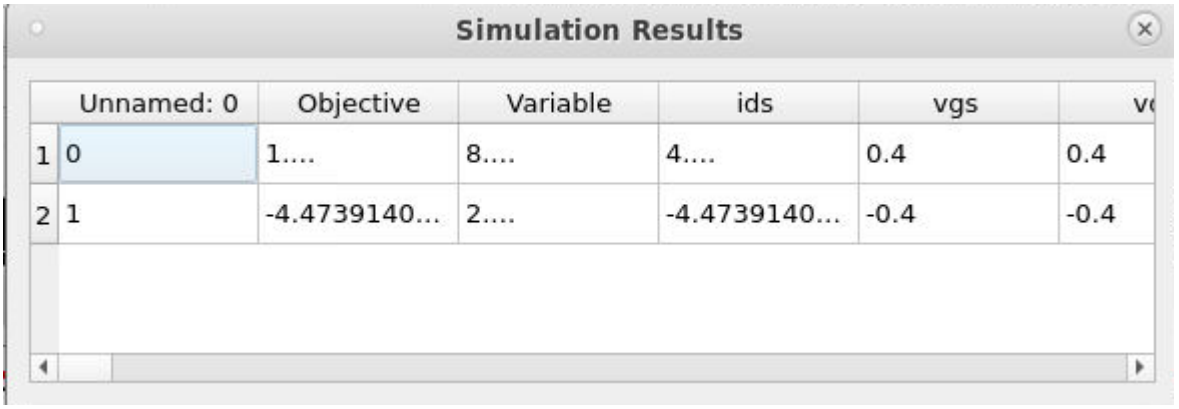
The NMOS transistor is set to output the value of the parameter **Ids**. To attach more steps to the project, click on the “+” button.



The project will be shown as the figure above.

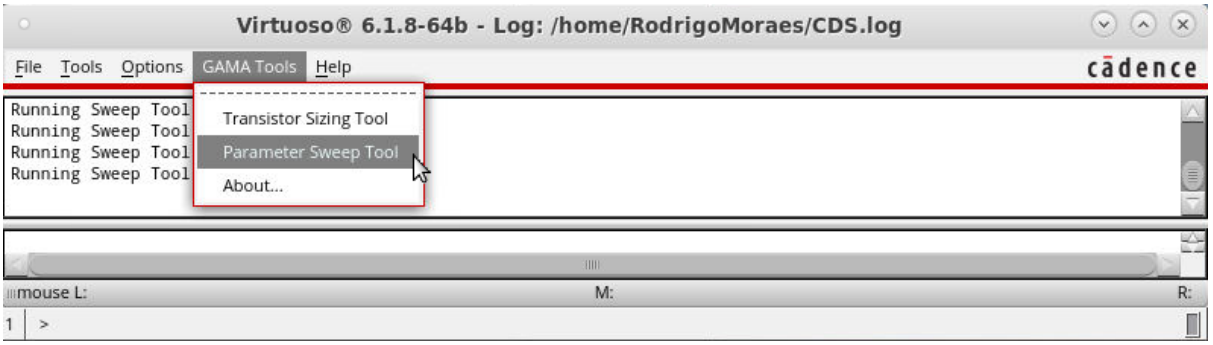
Notice that, to use the output value of the simulation done before, it is necessary to write “out” in the objective text field.

The results can be seen below:

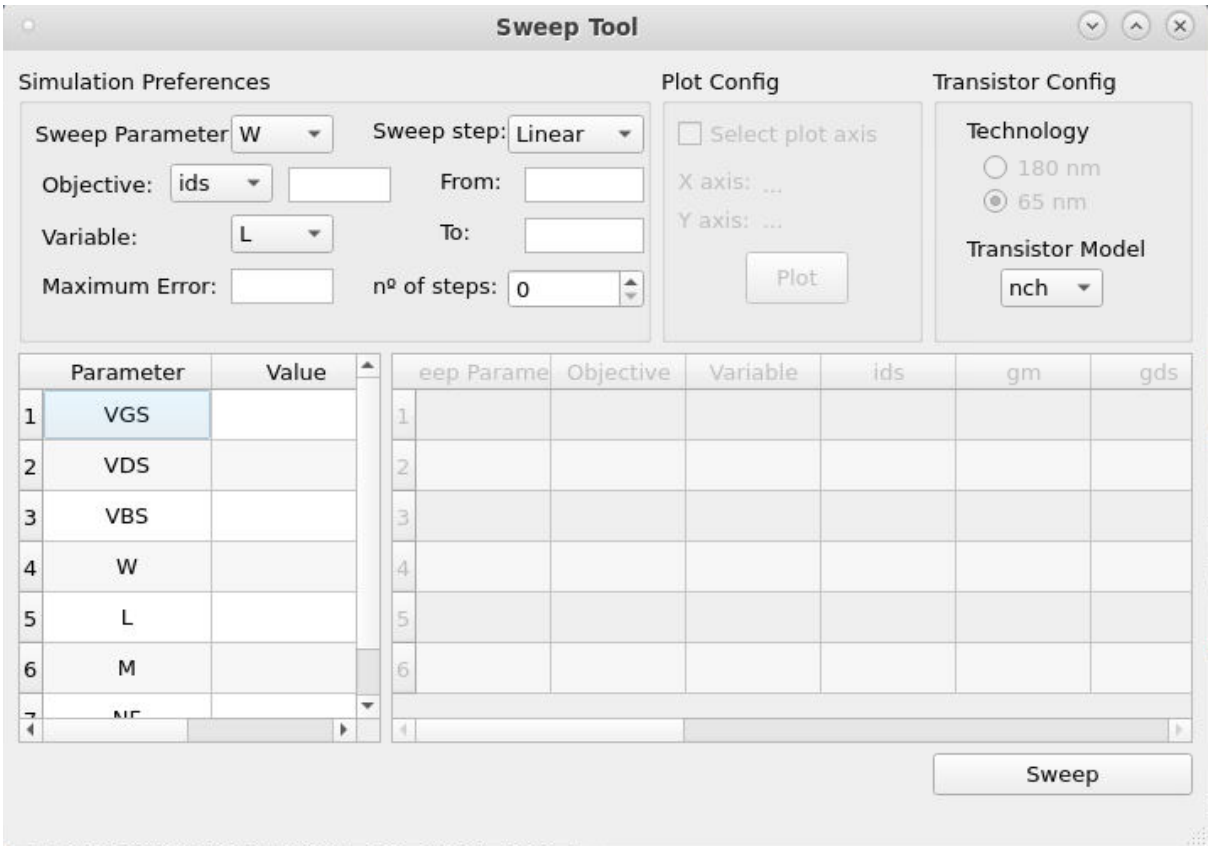


Sweep Tool Tutorial

To open the software, use the Cadence Virtuoso CIW menu.



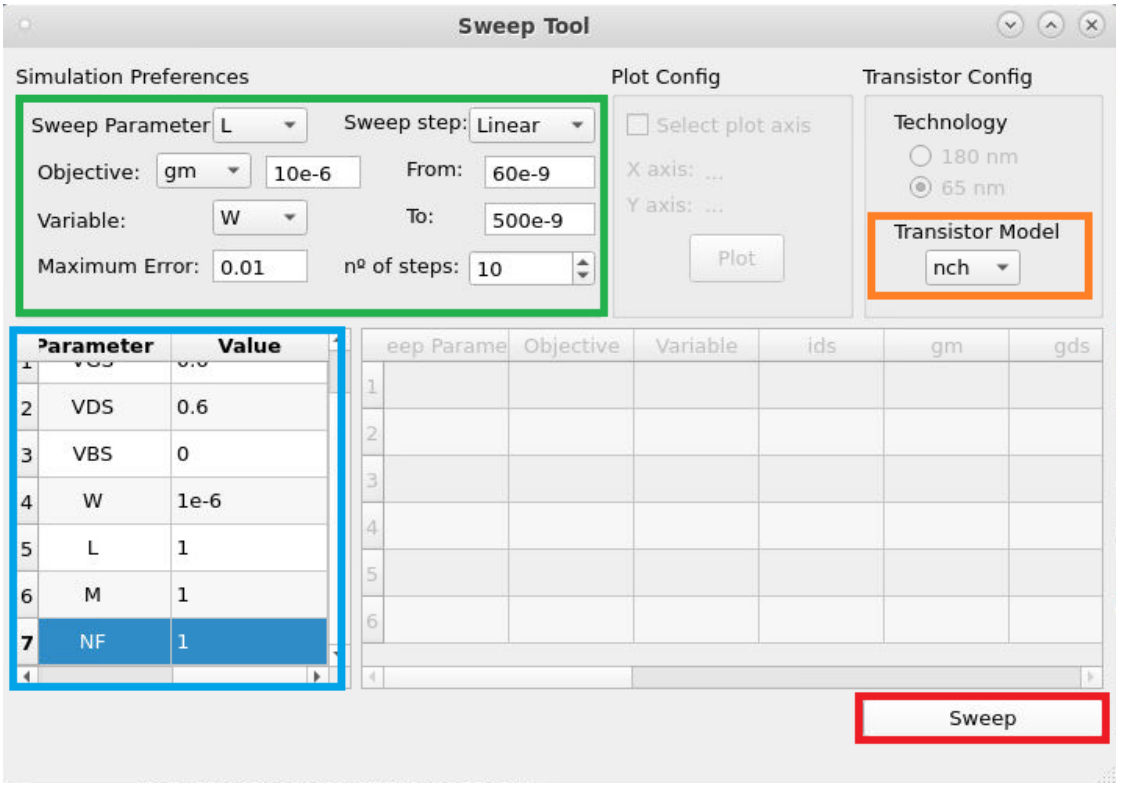
The Sweep Tool window will be shown.



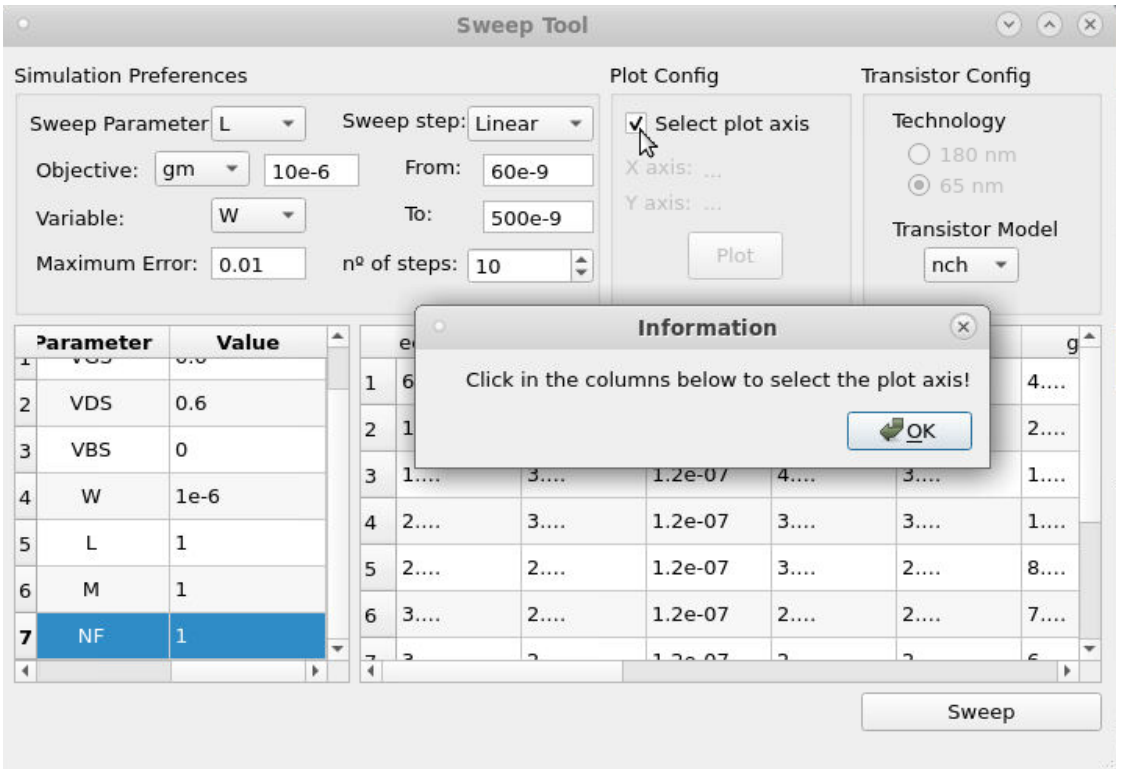
To start a new project, fill the empty spaces with the specifications of the project. An example is shown below.
For this example, we will use the following specifications:

Vdd = 0.8 V
gm = 10e-6 uS
Variable = W
Sweep Parameter = L

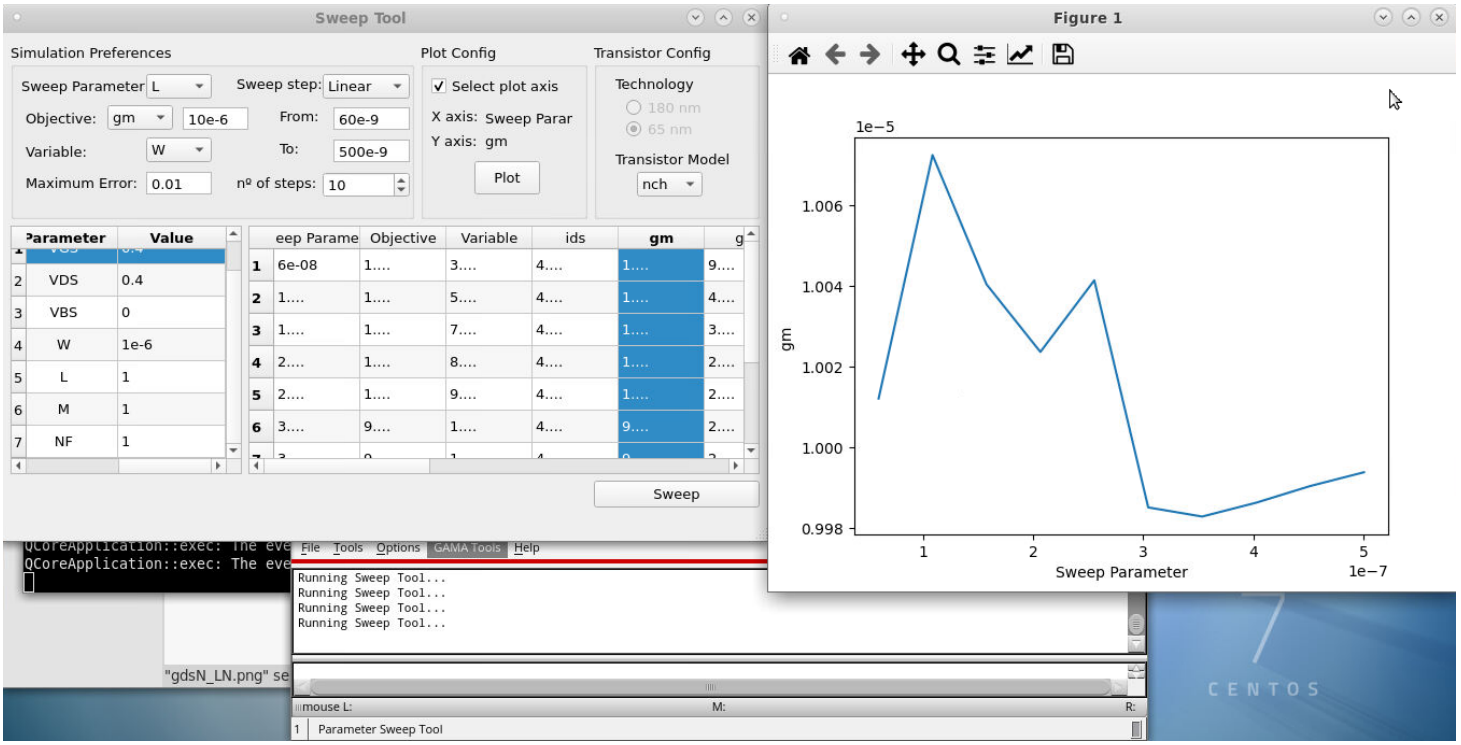
Transistor Model = nch
n of steps = 10
maximum error = 1%
60e-9 **<= L <=** 500e-9



The **green** rectangle shows the simulation preferences, on which the objective, maximum error and number of steps are specified, as well as the initial and last value of the sweep parameter;
In the **blue** area are the transistor specifications, used in the simulations;
The **orange** rectangle shows the transistor model that will be simulated;
And the **red** rectangle highlights the “Sweep” button, which starts the simulations.
After the simulation is done, click in the “Select plot axis” checkbox:



To select the plot axis, select the columns that are now available to be clicked. The selected axis are shown above the “Plot” button.
After selecting the parameters to be plotted, click on the “Plot” button. The results are shown below:



Other parameters can be selected to be plotted, showing the relationship between various parameters.