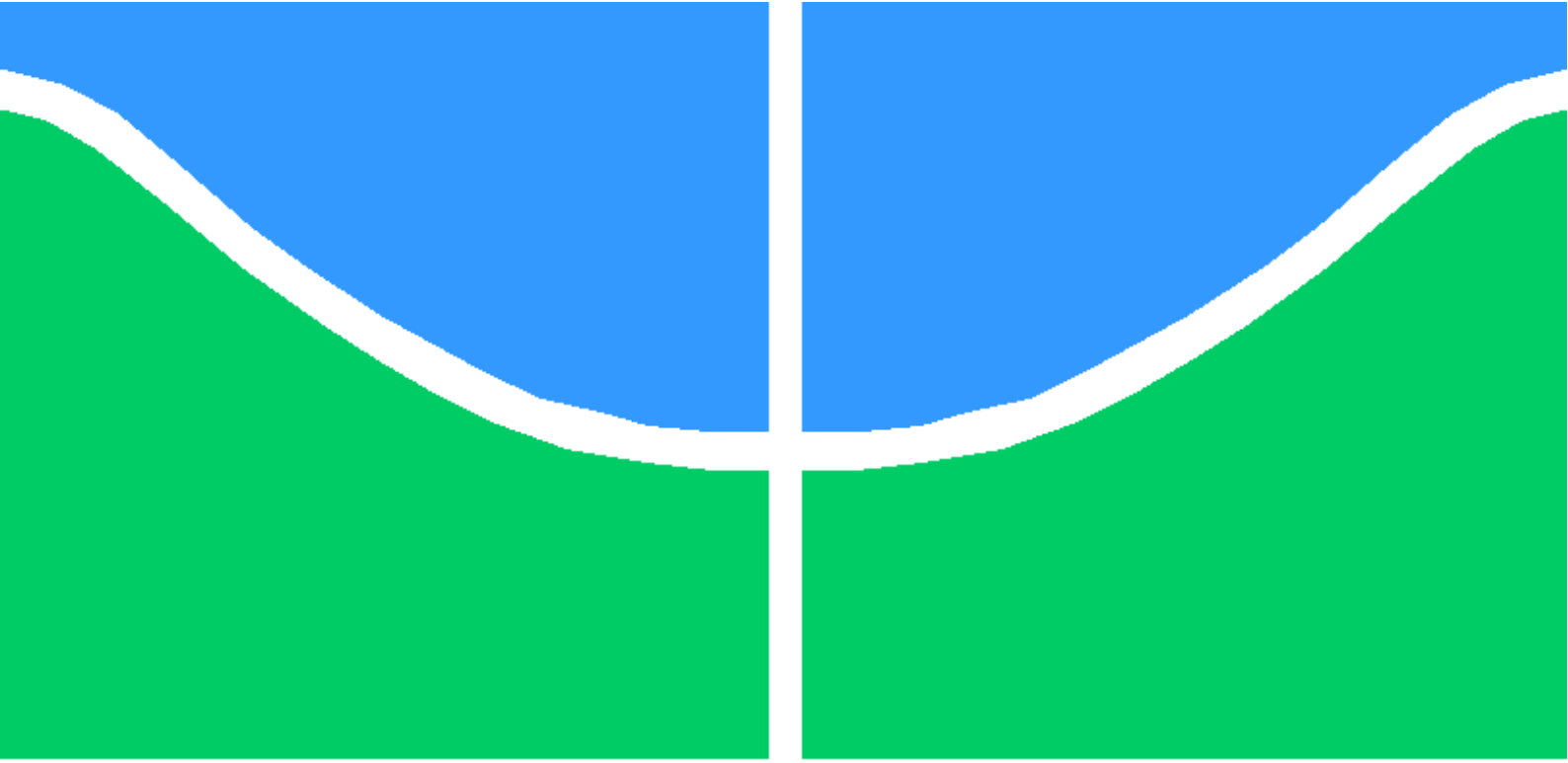


TRABALHO DE CONCLUSÃO DE CURSO

AUTOMATIZAÇÃO DE SIMULAÇÕES DE CIRCUITOS ELETRÔNICOS BASEADOS EM TRANSISTORES VARIÁVEIS

João Pedro Leite Nunes

Brasília, dezembro de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO

AUTOMATIZAÇÃO DE SIMULAÇÕES DE CIRCUITOS ELETRÔNICOS BASEADOS EM TRANSISTORES VARIÁVEIS

João Pedro Leite Nunes

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Stefan Michael Blawid, ENE/UnB _____
Orientador

Prof. José Edil G. de Medeiros, ENE/UnB _____
Examinador interno

Prof. Daniel Chaves Café, ENE/UnB _____
Examinador interno

Dedicatória

À minha mãe e ao meu pai.

João Pedro Leite Nunes

Agradecimentos

Agradeço primeiramente ao meu orientador, Prof. Stefan Michael Blawid, pela paciência e por ter sanado minhas dúvidas durante a realização deste Trabalho. Ao Prof. Muthupandian Cheralathan por ter motivado este Trabalho e por toda ajuda prestada durante sua execução. Aos bons professores do departamento. Ao meu pai e à minha mãe por todo o apoio. Ao meu irmão pela torcida. A todas as grandes amizades pelos bons momentos vividos durante o curso. E à Fê pela parceria.

João Pedro Leite Nunes

RESUMO

O presente Trabalho de Conclusão de Curso apresenta uma abordagem no sentido de automatizar a simulação de circuitos eletrônicos baseados em transistores variáveis. Propõe-se a criar não apenas uma série de rotinas Matlab, mas toda uma sistematização do processo, incluindo-se a criação de Tutoriais visando sua implementação por parte de outros alunos. A validação do uso da abordagem proposta é feita pela comparação direta entre os resultados obtidos por meio dela e os resultados obtidos em Referências Bibliográficas pertinentes.

ABSTRACT

The present Completion of Course Work introduces an approach to automate the simulation of electronic circuits based on variable transistors. It intends to create not only a series of Matlab routines, but a whole systematization of the process, including the creation of Tutorials for its implementation by other students. The proposed approach's usage is validated through the direct comparison between the results obtained through it and the results obtained in relevant Bibliographic References.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

FET	Transistores de Efeito de Campo
CNT	Transistores de Nanotubo de Carbono
CCAM	Modelo Semifísico de Transistores CNT
QUCS	Quite Universal Circuit Simulator
RF	Rádio-Frequência
MOS	Óxido Metálico Semicondutor
CML	Lógica de Modo de Corrente
RO	Oscilador em Anel
DC	Corrente Direta
AC	Corrente Alternada

Capítulo 1

Introdução

1.1 Contextualização

O desenvolvimento de dispositivos eletrônicos baseados em tecnologias emergentes visa substituir ou, ao menos, complementar dispositivos baseados em transistores de efeito de campo (field-effect transistors, FETs) de tecnologias largamente difundidas, como silício (Si) ou silício-germânio (SiGe) [1].

Dentre as promissoras possibilidades, o desenvolvimento de FETs de nanotubos de carbono (carbon nanotubes, CNT) destaca-se devido à relativa maturidade, evidenciada pela já relevante literatura, e ao potencial apresentado nessas obras, com destaque para sua característica nanométrica quase unidimensional (1D) [2]. Entretanto, mesmo demonstrando enorme potencial, a tecnologia CNT apresenta, também, grandes desafios e dificuldades para sua implementação, sendo uma das principais a dificuldade de obtenção de nanotubos puramente semicondutivos [3].

Portanto, para encorajar e promover o desenvolvimento da tecnologia CNT, assim como das demais, a fase de geração de benchmarks de circuitos faz-se necessária. Benchmarks de circuito, obtidos a partir de circuitos padrões (também chamados de circuitos de benchmark), são conjuntos de dados que expressam características de interesse desses circuitos padrões. Por exemplo, para o Buffer CML, um circuito de benchmark abordado neste Trabalho, duas características de interesse são as curvas de transferência e as curvas de resposta à onda quadrada. Os dados que expressam essas características são os benchmarks desse circuito.

É na fase de geração de benchmarks que se obtêm, experimentalmente, dados que atestam suas capacidades e limitações para diferentes combinações de parâmetros físico-estruturais e, ultimamente, sua viabilidade tanto do ponto de vista da performance e características elétricas quanto do ponto de vista econômico [1]. Por sua vez, para que possamos gerar tais benchmarks, precisamos adotar um modelo físico ou semi-físico dos CNTFETs. No presente estudo, optou-se por utilizar o modelo CCAM (modelo compacto semi-físico dos nanotubos de carbono), disponível em código Verilog-A [4].

A fase de benchmarks é composta por simulações que se subdividem em dois grupos: si-

mulações que contêm variações correlatas dos parâmetros, onde os circuitos são compostos por elementos de interesse (no caso deste Trabalho, transistores CNTFET) idênticos e simulações que contêm variações não-correlatas dos parâmetros, onde os circuitos são compostos por elementos de interesse diferentes. O primeiro grupo de simulações, também chamado de *Corner Analysis* (Análise de Canto) permite obter gráficos como o mostrado na Figura 1.1, retirada de [1]. Nele, para cada uma das simulações, os valores dos parâmetros são fixos e a figura de mérito desejada (na Figura 1.1, tempo de subida) pode ser expressa por um número. Já no segundo grupo de simulações, representado pela Figura 1.2, os valores de um parâmetro de interesse obedecem a alguma distribuição estatística (representada na Figura 1.2a por X, uma variável aleatória) e o que se deseja é mapear a distribuição estatística resultante de alguma figura de mérito de interesse (representada na Figura 1.2b por Y(X), uma variável aleatória função de X). O presente Trabalho abordou apenas as simulações do tipo correlatas.

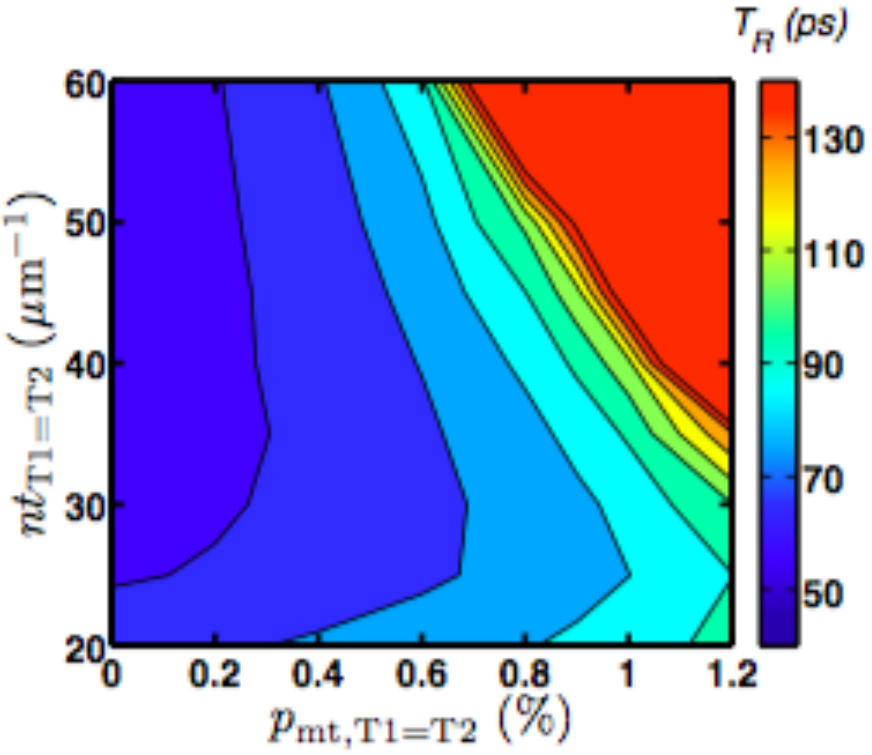
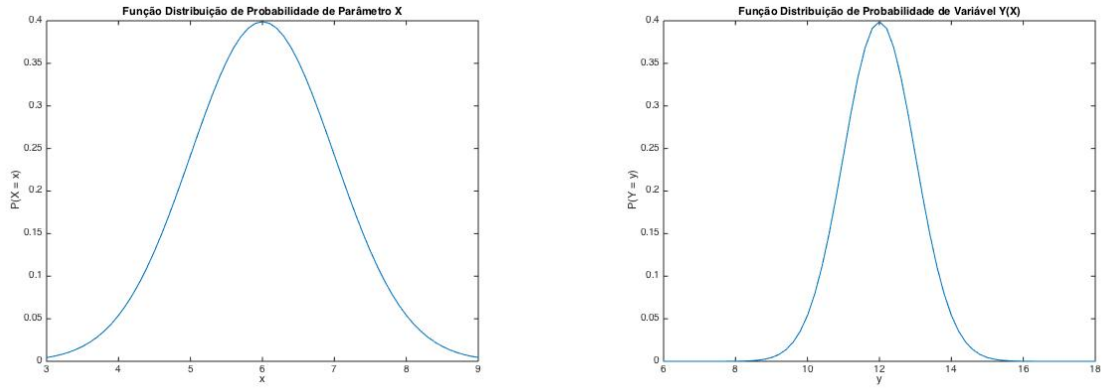


Figura 1.1: Exemplo de Simulação Correlata.

A não automatização da fase de geração de benchmarks pode torná-la demorada a ponto de ser praticamente inviável, prejudicando, consequentemente, a previsão do impacto da variabilidade dos parâmetros físico-estruturais. Suponha-se, por exemplo, que se deseje simular um circuito contendo 2 transistores idênticos e variando-se 10 parâmetros de interesse, cada um podendo assumir 10 valores possíveis. Para isso, seriam necessárias 10^{10} simulações, tornando a automatização desse processo uma necessidade. Em especial, para tecnologias emergentes, essa automatização é muitas vezes inexistente. Nesse sentido, a criação de um algoritmo capaz de realizar tal automatização mostra-se uma importante oportunidade de avanço técnico-científico.



(a) (b)

Figura 1.2: Representação de Simulação Não-Correlata.

No caso específico do trabalho reportado em [1],o qual motivou a realização deste Trabalho de Conclusão de Curso, a metodologia não automatizada consistia num método demorado e ineficiente de produção de planilhas contendo diferentes cartas de modelo. Parte de uma dessas planilhas é mostrada na Figura 1.3 a título de ilustração. A partir dela, é possível imaginar que grande parte do esforço empregado na realização do trabalho tenha sido concentrado na confecção de planilhas semelhantes, daí a necessidade de criar-se um algoritmo de automatização dessa tarefa.

1.2 Objetivo

O presente trabalho se propõe a criar um software de interface gráfica em ambiente Matlab capaz de automatizar a fase de geração de benchmarks de circuitos eletrônicos em geral e de circuitos compostos por transistores CNTFET em particular, bem como um guia para sua utilização. Esse software utilizará como simulador o simulador de código aberto QUCS e terá sua aplicação comprovada e validada pela comparação com os resultados obtidos por Muthupandian Cheralathan, Martin Claus e Stefan Blawid em [1]. A aplicação desse software deve ser capaz de aprofundar o conhecimento que temos dessa e de outras tecnologias emergentes e ajudar-nos a comparar seus resultados com os das tecnologias consolidadas. Mais especificamente, deve ser capaz de automatizar o trabalho realizado por Muthupandian Cheralathan em [1], diminuindo o tempo despendido na fase de geração de benchmarks e aumentando sua eficiência e a organização de seus resultados. Além disso, o software deve ser adaptável para outra situações, tais como: simulações com variações não-correlatas dos parâmetros e simulações de circuitos contendo dispositivos baseados em outras tecnologias.

1.3 Organização

No capítulo 2 é feita uma revisão bibliográfica sobre o tema de estudo. Em seguida, o capítulo 3 descreve a metodologia empregada no desenvolvimento do projeto. Resultados são discutidos

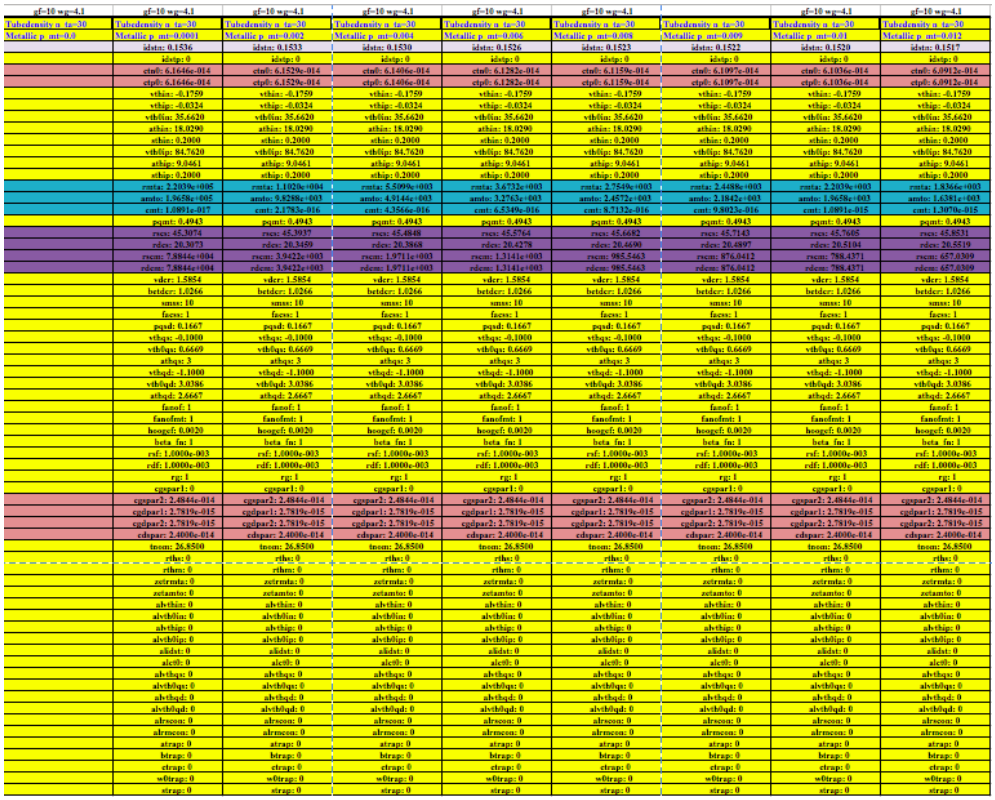


Figura 1.3: Parte de uma das planilhas elaboradas como parte do trabalho reportado em [1].

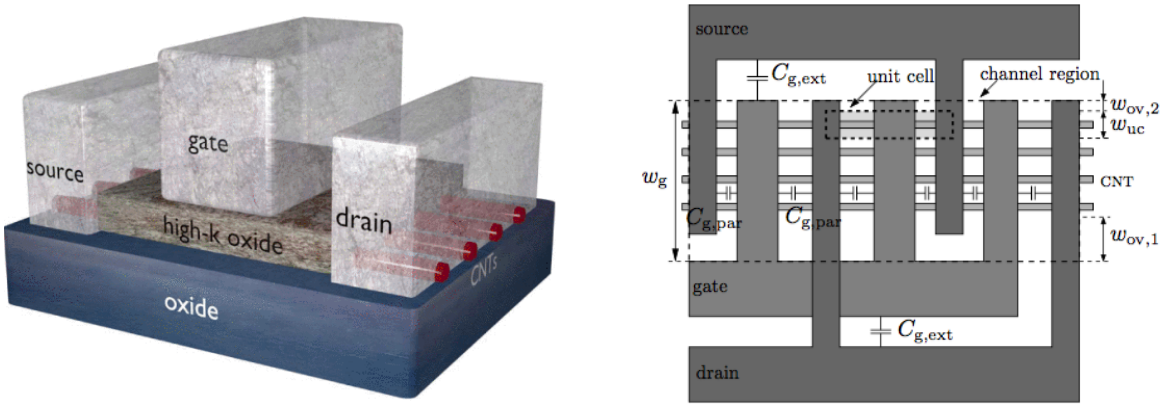
no capítulo 4, seguidos das conclusões no capítulo 5. Os anexos contêm material complementar.

Capítulo 2

Revisão Bibliográfica

2.1 Transistor CNTFET

Embora não seja o escopo deste Trabalho apresentar um estudo detalhado das propriedades físico-químicas de um CNTFET, é importante para a compreensão de alguns de seus parâmetros físico-estruturais apresentar algumas de suas estruturas típicas. Uma estrutura típica de um CNTFET com um dedo e com múltiplos nanotubos de carbono pode ser vista na Figura 2.1a [4]. Uma outra estrutura típica de um CNTFET, com múltiplos dedos e com múltiplos nanotubos de carbono pode ser vista na Figura 2.1b [11]. Nota-se que, enquanto na primeira há a formação de apenas um canal por tubo, na segunda há a formação de múltiplos canais por tubo.



(a) (b)

Figura 2.1: Estruturas típicas de CNTFETs com um dedo e com múltiplos nanotubos de carbono (a) e com múltiplos dedos e múltiplos nanotubos de carbono (b).

Comparados a outras tecnologias, transistores de efeito de campo de nanotubos de carbono (CNTFETs) possuem algumas propriedades importantes para determinadas aplicações. Em particular, sua característica nanométrica quase unidimensional (1D) e sua consequente característica de transporte também quase unidimensional proporcionam não apenas uma alta capacidade de

transporte de carga mas também uma relação linear entre corrente de dreno e tensão de entrada (tensão porta-fonte) sob determinadas condições. Espera-se que tal linearidade seja benéfica para aplicações nos mais diversos campos. [4]

O design de Sistemas RF (Sistemas de Rádio-Frequência), por exemplo, depende criticamente do design de seus blocos de circuito, tais como Amplificadores de Potência e Osciladores. O design desses blocos de circuito é realizado em simuladores utilizando modelos compactos que representam os dispositivos reais em um chip. Os pré-requisitos para o uso desses modelos compactos são rigorosos já que eles precisam descrever com precisão o comportamento de dispositivos não-lineares para grandes intervalos de valores de polarização, de frequência e de temperatura enquanto se mantêm suficientemente simples para não extrapolar o tempo de simulação dos circuitos para além do razoável. Mais especificamente, a descrição precisa do comportamento de pequenos sinais desses dispositivos requer a obtenção da primeira derivada das correntes e cargas em relação às tensões em seus terminais, enquanto a descrição do comportamento de grandes sinais de formas de onda no domínio do tempo requer a obtenção de derivadas de quinta ordem das correntes e cargas.[4]

2.2 O Modelo CCAM

Inserido nesse contexto, surge o modelo CCAM detalhado em [4], implementado em Verilog-A para permitir seu uso irrestrito nos mais variados simuladores de circuitos. O desenvolvimento desse modelo utilizou uma abordagem semi-física, driblando a limitação imposta pelo conhecimento ainda limitado da física de dispositivos CNT. Tal abordagem permitiu a obtenção dos pré-requisitos mencionados na seção anterior.

O circuito equivalente de grandes sinais do modelo CCAM é mostrado na Figura 2.2. Foge do escopo deste trabalho analisar sua obtenção e descrever matematicamente todos os seus parâmetros. No entanto, há dentre esses parâmetros, de natureza elétrica, alguns mais relevantes para este trabalho e que merecem, portanto, uma abordagem mais cuidadosa, assim como sua relação com os parâmetros físico-estruturais e com a geometria do transistor. Tais parâmetros são evidenciados pela Figura 2.2.

Cartas do modelo (Modelcards) CCAM calibradas forma publicadas na plataforma nanoHUB, mantida pela Network for Computational Nanotechnology [5]. Uma carta de modelo pode ser entendida como o conjunto de valores dos parâmetros elétricos, tais como os evidenciados na Figura 2.2, obtidos para um determinado conjunto de valores dos parâmetros físico-estruturais de um CNTFET. No caso específico deste Trabalho, as cartas de modelo foram geradas a partir da variação do número de dedos da porta (gate fingers) (n_{gf}), da largura da porta (W_g), da densidade de nanotubos de carbono (n_t) e da porcentagem de tubos metálicos (p_{mt}) e da aplicação de relações de proporcionalidade específicas. De maneira geral, esses também são os principais parâmetros variados no âmbito de outros estudos, como [1].

A densidade de tubos semicondutores e metálicos no canal segue de:

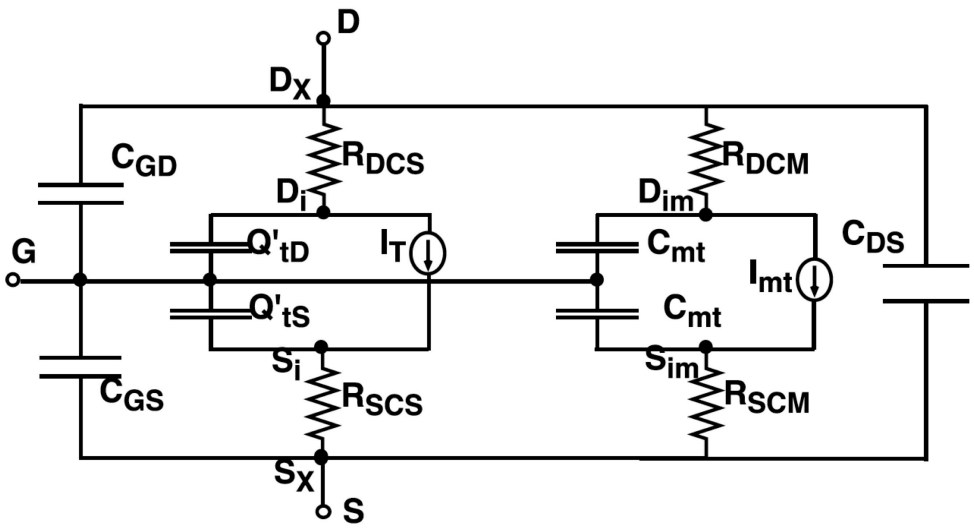


Figura 2.2: Circuito Equivalente de Grandes Sinais Reduzido do Modelo CCAM.

$$n_{semi} = n_t \times (1 - p_{mt}) \tag{2.1}$$

$$n_{mt} = n_t \times p_{mt} \tag{2.2}$$

O número efetivo total de tubos (N_t) no canal equivale à soma do número de tubos semicondutores (N_{semi}) e de tubos metálicos (N_{mt}):

$$N_t = N_{semi} + N_{mt} \tag{2.3}$$

$$N_{semi} = n_{gf} \times W_g \times n_{semi} \tag{2.4}$$

$$N_{mt} = n_{gf} \times W_g \times n_{mt} \tag{2.5}$$

$$\tag{2.6}$$

Os parâmetros intrínsecos da parte semicondutora do canal (corrente de carga $\dot{Q}_t \sim C_{tn0}$, corrente de transferência I_t e resistências seriais $R_{d/s,cs}$) são proporcionais a N_{semi}, enquanto os parâmetros intrínsecos da parte metálica do canal (C_{mt}, I_{mt} e $R_{d/s,cm}$) são proporcionais a N_{mt}.

A proporcionalização das capacitâncias extrínsecas depende apenas da geometria do transistor e corresponde às relações dadas por:

$$\frac{C_{gs}}{C_{gs,ref}} = \frac{n_{gf}}{n_{gf,ref}} \times \frac{1 + W_g \times a_{ref}}{1 + W_{g,ref} \times a_{ref}} \tag{2.7}$$

$$\frac{C_{gd}}{C_{gd,ref}} = \frac{n_{gf}}{n_{gfref}} \times \frac{W_g}{W_{gref}} \tag{2.8}$$

$$\frac{C_{ds}}{C_{ds,ref}} = \frac{\frac{3}{2} + n_{gf}}{\frac{3}{2} + n_{gf,ref}} \tag{2.9}$$

Alguns parâmetros elétricos de um CNTFET, modelo CCAM, com comprimento do canal $L_g = 0,18\mu m$, número de dedos da porta $n_{gf} = 8$, largura da porta $W_g = 50\mu m$, densidade de nanotubos de carbono $n_t = 8\,\mu\text{m}$ e percentual de tubos metálicos $p_{mt} = 30\%$, retirados da carta de modelo correspondente, são mostrados na Tabela 2.1 [1].

Tabela 2.1: Parâmetros elétricos de um CNTFET, modelo CCAM, com $L_g = 0,18\mu m$, $n_{gf} = 8$, $W_g = 50\mu m$, $n_t = 8\,\mu\text{m}$ e $p_{mt} = 30\%$.

(a) Capacitâncias

$C_{\text{gs,ref}}$	67.6 fF
a_{ref}	$0.0666\,\mu\text{m}^{-1}$
$C_{\text{gd,ref}}$	54.2 fF
$C_{\text{ds,ref}}$	20.0 fF
$C_{\text{tn0,ref}}$	112 fF
$C_{\text{mt,ref}}$	42.5 fF

(b) Resistências

$R_{\text{scs,ref}}$	$24.9\,\Omega$
$R_{\text{dcs,ref}}$	$11.1\,\Omega$
$R_{\text{scm,ref}}$	$10.1\,\Omega$
$R_{\text{dcm,ref}}$	$10.1\,\Omega$

2.3 Design de um Buffer CML baseado em tecnologia CNT

Atualmente, um Buffer CML operando a $2,4GHz$ baseado em tecnologia MOS de $0,18\mu m$ pode ser facilmente realizado. Num futuro próximo, devido ao avanço no desenvolvimento técnico-científico, a realização desse mesmo dispositivo será possível também em tecnologia CNT de $0,18\mu m$, possibilitando, assim, uma comparação direta entre ambas as tecnologias [1].

Para fins de comparação e validação dos resultados, optou-se por seguir uma metodologia similar à utilizada em [1] para o design de um Buffer CML contendo transistores CNTFET, como o mostrado na Figura 2.3[1], proporcionalizado para um comprimento de canal de $0,18\mu m$ e operando sob uma tensão de alimentação $V_{dd} = 1,8V$. Os parâmetros de design são a corrente de cauda I_t, a resistência de carga R_l e a largura da porta W_{ef}.

O modelo CCAM prevê que o CNTFET conduz corrente razoavelmente bem quando a tensão porta-fonte $V_{gs} \approx 0V$. Portanto, o valor mínimo da tensão da porta desses transistores deve ser igual ao valor da tensão da fonte comum do estágio amplificador diferencial:

$$V_{g,min} = V_s \tag{2.10}$$

O que implica em:

$$V_{gs,min} = 0 \tag{2.11}$$

A frequência de trânsito máxima pode ser obtida matematicamente dos parâmetros de admitância Y do Buffer CML. Essa frequência de trânsito máxima é atingida para $V_{gs} \approx 0,4V$ como

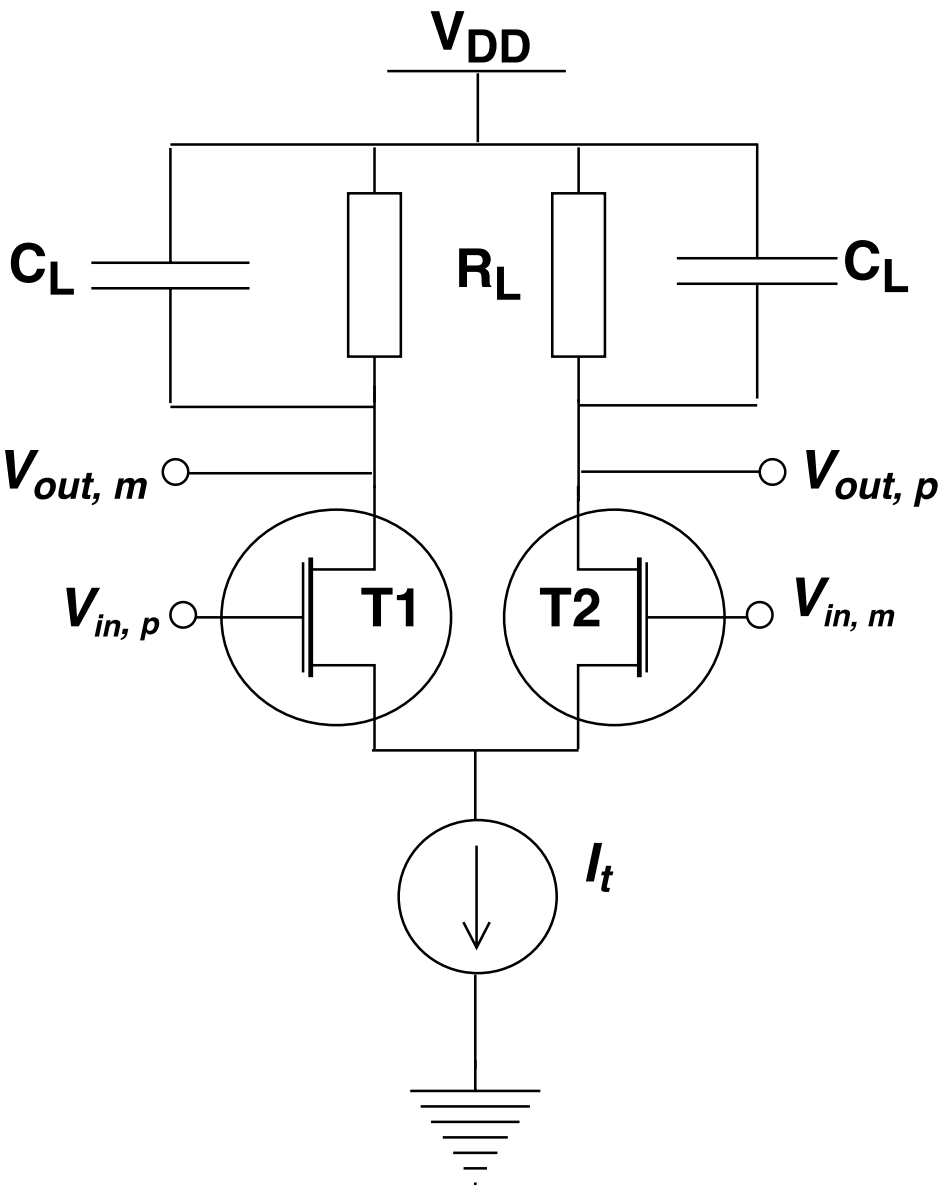


Figura 2.3: Buffer CML contendo transistores CNTFET

pode ser visto na Figura 2.4[1]. Logo, $V_{gs,max} = 0,4V$, o que leva a:

$$\begin{aligned} V_{in,dif} &= V_{gs,max} - V_{gs,min} \\ V_{in,dif} &= 0,4V \end{aligned} \tag{2.12}$$

Tal situação é atingida quando um dos transistores se encontra em região de corte e o outro se encontra em região de saturação. Por exemplo, $V_{gs,2} = 0V$ e $V_{gs,1} = 0,4V$.

Para garantir um ganho de tensão diferencial de grandes sinais $A_{V,dif} = 1,5$, deve-se ter uma tensão diferencial de saída $V_{out,dif} = 0,6V$. Pode-se encontrar a tensão de dreno do transistor em saturação utilizando:

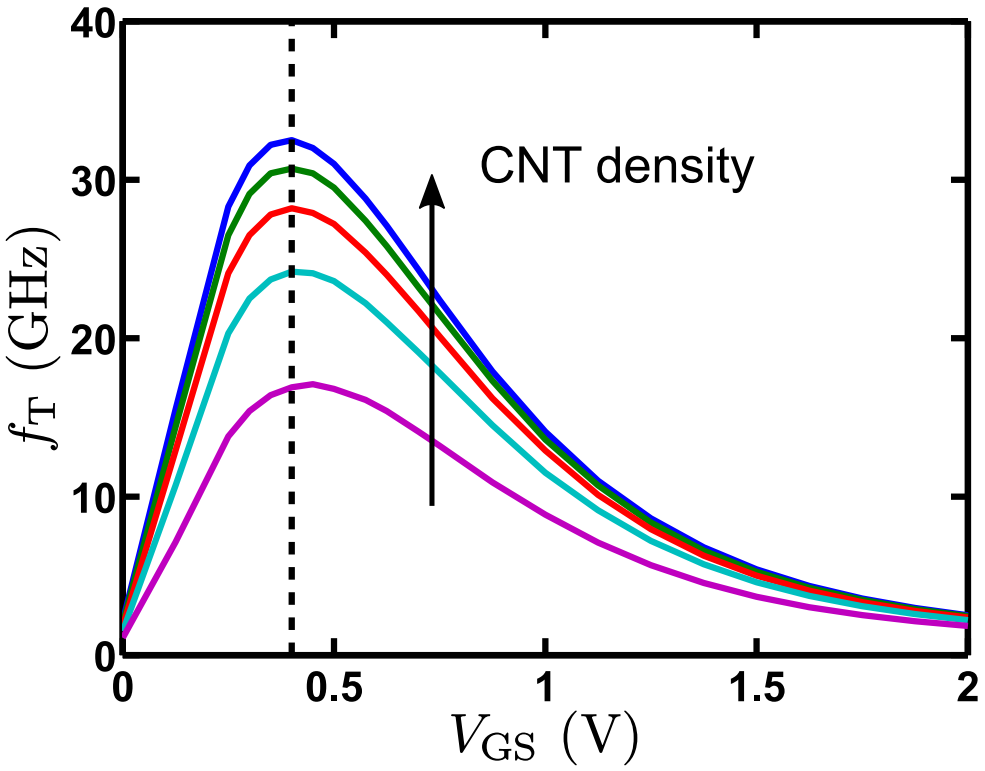


Figura 2.4: Dependência da frequência de trânsito do transistor CNTFET, modelo CCAM, em relação à tensão porta-fonte V_{gs}

$$V_{out,dif} = |V_{d,1} - V_{d,2}| \tag{2.13}$$

Como o Buffer CML possui propriedade inversora de tensão, a Equação 2.13 se torna:

$$\begin{aligned} V_{out,dif} &= V_{d,2} - V_{d,1} \\ V_{d,1} &= V_{d,2} - V_{out,dif} \\ V_{d,1} &= 1,8 - 0,6 \\ V_{d,1} &= 1,2V \end{aligned} \tag{2.14}$$

Como o estágio fonte-comum também é realizado com CNTFETs, considera-se que a tensão de dreno do transistor em saturação é igualmente dividida entre esse estágio e o estágio amplificador diferencial, o que fornece $V_s = 0,6V$ e, utilizando-se a Equação 2.10, obtém-se $V_{g,min} = 0,6V$. Por sua vez, a tensão de porta máxima é dada por:

$$\begin{aligned} V_{g,max} &= V_{gs,max} + V_s \\ V_{g,max} &= 0,4 + 0,6 \\ V_{g,max} &= 1V \end{aligned} \tag{2.15}$$

E pode-se encontrar a tensão de nível comum da porta:

$$\begin{aligned} V_{cm} &= \frac{(V_{g,min} + V_{g,max})}{2} \\ V_{cm} &= \frac{(0,6+1)}{2} \\ V_{cm} &= 0,8V \end{aligned} \tag{2.16}$$

A configuração das tensões aplicadas sobre o transistor em saturação nos permite encontrar a densidade de corrente por largura de canal máxima de um transistor CNTFET $J_{ds,max}$. Como pode ser visto na Figura 2.5[1], $J_{ds,max} = 1,1\mu A$ para $V_{gs} = 0,4V$ e $V_{ds} = 0,6V$.

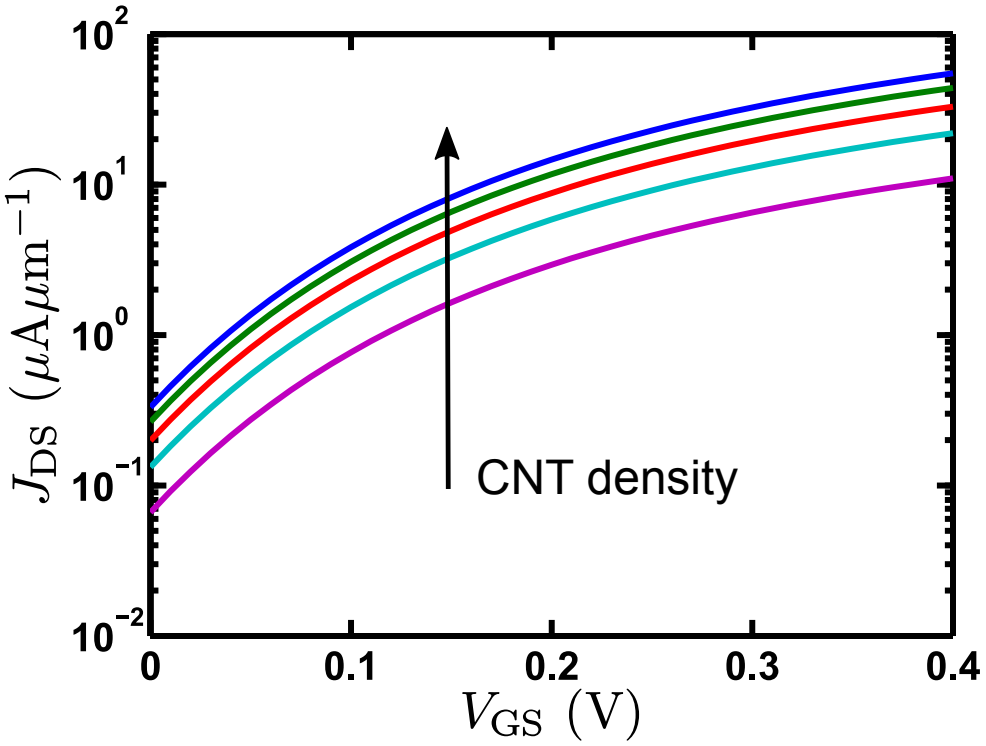


Figura 2.5: Densidade de corrente por largura de canal máxima $J_{ds,max}$ de um único tubo para $V_{ds} = 0,6V$ e V_{gs} variando de $0V$ a $0,4V$.

Supondo uma corrente de cauda $I_t = 1,8mA$, encontramos os demais parâmetros de design, R_l e W_{ef} por meio das Equações 2.17 e 2.18.

$$\begin{aligned} R_l &= \frac{(V_{dd} - V_{d,1})}{I_t} \\ R_l &= \frac{(1,8-1,2)}{1,8 \times 10^{-3}} \\ R_l &= 333,33\Omega \end{aligned} \tag{2.17}$$

$$
\begin{aligned}
W_{ef} &= \frac{I_t}{n_t \times J_{ds,max}} \\
W_{ef} &= \frac{1,8 \times 10^{-3}}{\frac{4}{10^{-6}} \times 1,1 \times 10^{-6}} \\
W_{ef} &= 41\mu m
\end{aligned}
\tag{2.18}
$$

O atraso da porta pode ser aproximado por:

$$
\tau \approx C_l \times R_l \tag{2.19}
$$

[5]

Queremos um atraso muito menor do que a metade do inverso da frequência de corte máxima. Escolhemos um valor dez vezes menor:

$$
\tau = 0,05 \times \frac{1}{f_{t,max}} \tag{2.20}
$$

Para uma frequência de trânsito máxima de interesse, $f_{t,max} = 2,4GHz$ e:

$$
\begin{aligned}
\tau &= 0,05 \times \frac{1}{2,4 \times 10^9} \\
\tau &= 21ps
\end{aligned}
\tag{2.21}
$$

Igualando ambas as equações, podemos determinar o valor da capacitância de carga equivalente, C_l:

$$
\begin{aligned}
C_l &= \frac{\tau}{R_l} \\
C_l &= \frac{21 \times 10^{-12}}{R_l} \\
C_l &= 63fF
\end{aligned}
\tag{2.22}
$$

Alguns parâmetros elétricos de um CNTFET, modelo CCAM, com comprimento do canal $L_g = 0,18\mu m$, número de dedos da porta $n_{gf} = 10$, largura da porta $W_g = 4,1\mu m$, densidade de nanotubos de carbono $n_t = 40\,\mu m$ e percentual de tubos metálicos $p_{mt} \approx 0\%$ (escolhido $p_{mt} = 0,01\%$), retirados da carta de modelo correspondente, são mostrados na Tabela 2.2.

2.4 Oscilador em Anel

Quando se compara a performance de portas implementadas em diferentes tecnologias, é importante não incluir na comparação parâmetros como fator de carga, *fan-in* e *fan-out*. Uma maneira

Tabela 2.2: Parâmetros elétricos de um CNTFET, modelo CCAM, com $L_g = 0,18\mu m$, $n_{gf} = 10$, $W_g = 4,1\mu m$, $n_t = 40\,\mu m$ e $p_{mt} = 0,01\%$.

(a) Capacitâncias

$C_{\mathrm{gs,ref}}$	24.8 fF
$C_{\mathrm{gd,ref}}$	5.6 fF
$C_{\mathrm{ds,ref}}$	24.0 fF
$C_{\mathrm{tn0,ref}}$	82.1 fF
$C_{\mathrm{mt,ref}}$	0 fF

(b) Resistências

$R_{\mathrm{scs,ref}}$	33.9 Ω
$R_{\mathrm{dcs,ref}}$	15.2 Ω
$R_{\mathrm{scm,ref}}$	59.1 kΩ
$R_{\mathrm{dcm,ref}}$	59.1 kΩ

uniforme de medir o tempo de propagação de uma onda em uma porta, de modo a permitir a comparação de tecnologias é, portanto, desejável. O circuito padrão, ou circuito de benchmark, para medir atraso é o Oscilador em Anel (RO) [7].

Um Oscilador em Anel é produzido pela conexão em cascata de um número ímpar n de estágios inversores em loop, o que resulta na longa topologia de cadeia mostrada na Figura 2.6. O atraso T_i de cada inversor resulta na frequência de clock:

$$f_{clk} = \frac{1}{n \times T_i} \tag{2.23}$$

[6]

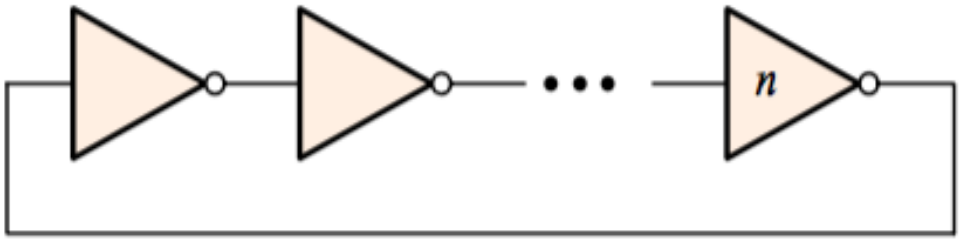


Figura 2.6: Topologia de um Oscilador em Anel.

Devido ao número ímpar de inversões, esse circuito não possui um ponto de operação estável e, por isso, oscila.

Comportamento oscilatório está presente em todos os sistemas físicos, especialmente eletrônicos e ópticos. Em sistemas de comunicação de rádio-frequência e de ondas de luz, osciladores são usados para tradução de informação e seleção de canal. Osciladores também estão presente em todos os sistemas de eletrônica digital, os quais requerem referência de tempo (sinal de clock) para sincronizar operações [8]. Tipicamente, um Oscilador em Anel precisa de pelo menos cinco estágios para ser operacional [7]. A implementação de um RO a partir de transistores CNTFET em uma frequência de operação de 500 MHz foi relatada em [9].

Para que um Oscilador em Anel oscile de fato, é necessário realizar a excitação desse circuito, aplicando um impulso em sua entrada. No esquemático do RO utilizado em [1] disponível em

anexo, essa excitação é realizada utilizando-se uma fonte DC de 0,6 V em série com uma chave controlada por tempo que abre 0,01 ns após iniciada a simulação.

Capítulo 3

Metodologia

O trabalho foi realizado em sistema operacional macOS Sierra versão 10.12.15 e utilizando-se os softwares Matlab versão R2014b (8.4.0.150421) e QUCS versão 0.0.18. O QUCS, em especial, possui um processo de instalação sensivelmente mais complicado em sistema operacional macOS. Por isso, optou-se pela criação de um tutorial, disponível em anexo, para auxiliar nesse processo.

O primeiro passo para a realização do trabalho foi compreender o funcionamento do QUCS. Esse funcionamento é evidenciado na Figura 3.1. Primeiramente, monta-se o componente de circuito relativo ao transistor CNTFET a partir do código Verilog-A de seu modelo CCAM. Em seguida, cria-se o esquemático do circuito cujo funcionamento deseja-se investigar e inicia-se sua simulação. Ao iniciar a simulação de um circuito, o QUCS gera um arquivo de netlist a partir do esquemático que é, então, utilizado em seu simulador backend para a geração dos resultados. Portanto, conclui-se que, para automatizar a fase de geração de benchmarks, deve-se, basicamente, automatizar a geração das netlists de circuito para várias combinações de parâmetros físico-estruturais dos transistores CNTFET.

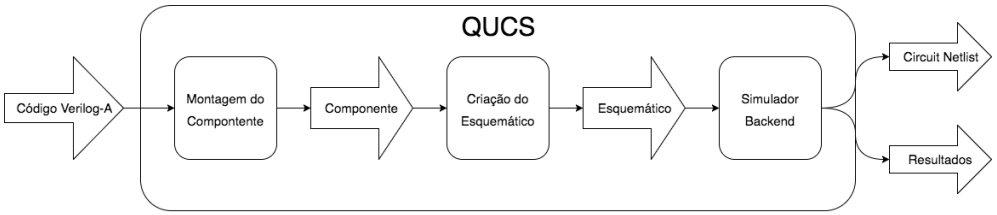


Figura 3.1: Funcionamento do QUCS

A criação do Automated Simulator, portanto, seguiu nessa direção, adicionando também métodos simples de organização dos resultados, importante para criação de material de referência, para uma melhor compreensão por parte de futuros usuários e para aumentar a velocidade da análise dos resultados. Seu código, assim como um tutorial, está disponível em anexo e seu funcionamento é evidenciado na Figura 3.2.

Por fim, para a análise dos resultados e sua consolidação gráfica, utilizaram-se funções Matlab já desenvolvidas pela comunidade de desenvolvedores e usuários do QUCS, adaptadas para que se obtivessem os resultados desejados. Essas funções adaptadas estão disponíveis em anexo. Os

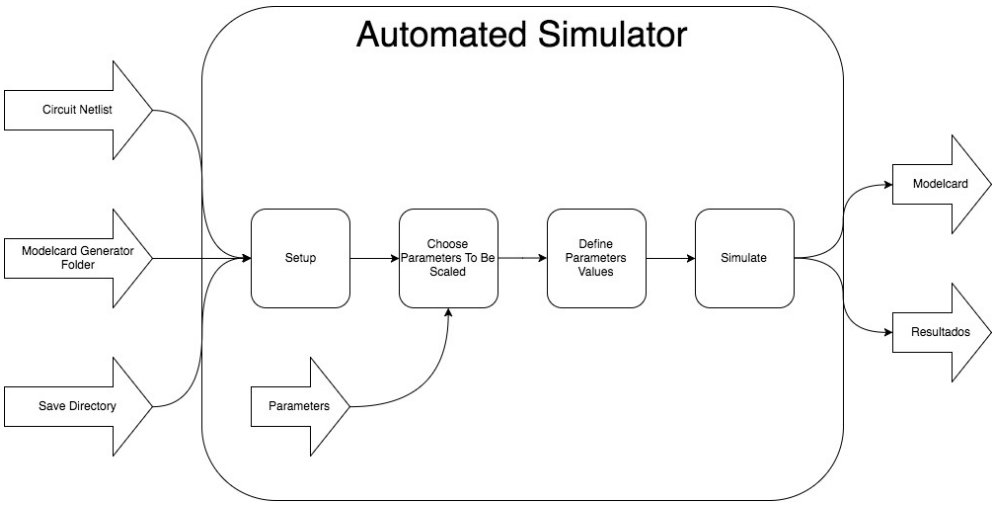


Figura 3.2: Funcionamento do Automated Simulator

resultados foram comparados com os obtidos por Muthupandian Cheralathan, Martin Claus e Stefan Blawid em [1].

O fluxo de trabalho simplificado mostrado na Figura 3.3 mostra, portanto, a metodologia geral deste trabalho, bem como como sua condução pode ser subdividida e compreendida, além de como futuros trabalhos baseados no Automated Simulator poderão ser conduzidos: primeiramente, cria-se o circuito a ser investigado utilizando-se o QUCS, depois simula-se esse circuito para variadas combinações de parâmetros dos componentes de interesse utilizando-se o Automated Simulator e, por fim, analisam-se os resultados.

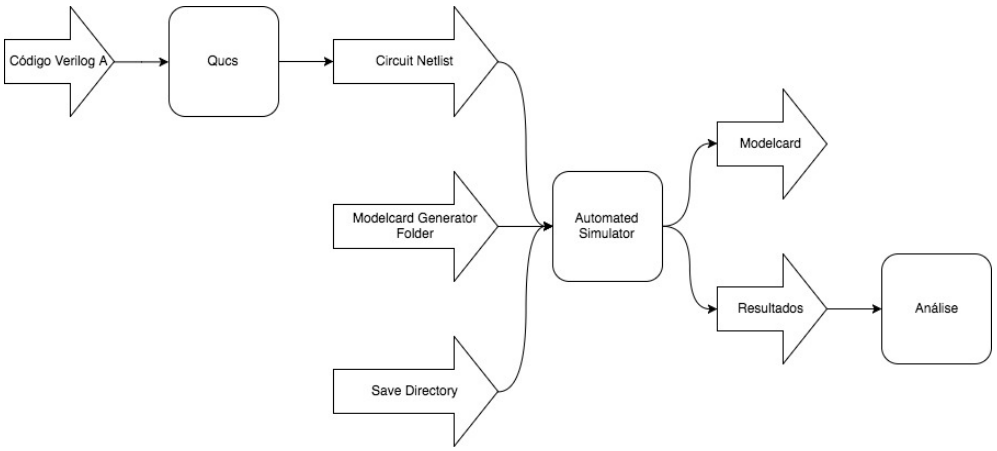


Figura 3.3: Fluxo de Trabalho Geral

Capítulo 4

Resultados

4.1 Introdução

Nesta seção são apresentados os resultados experimentais obtidos pela aplicação da metodologia apresentada no capítulo Metodologia. Esses resultados e a praticidade de sua obtenção em comparação a metodologias adotadas anteriormente [1] corroboram o alcance do objetivo deste Trabalho: automatizar o processo de simulação de circuitos eletrônicos. Esse sucesso é auferido pela comparação direta entre os resultados obtidos e os resultados expostos em [1].

Os resultados experimentais englobam os dados obtidos pela simulação automática dos circuitos eletrônicos abordados no capítulo Referências Bibliográficas e incluem, em alguns casos, sua manipulação. Entre os resultados encontram-se: tabelas com valores de parâmetros elétricos de transistores; curvas características de transistores; curvas de frequência de transição para transistores; curvas de saída e de resposta ao degrau de um Buffer CML para diferentes valores dos parâmetros de referência de seus transistores.

4.2 Modelo CCAM de Referência

Um primeiro passo, necessário para a validação dos resultados, é a adoção do modelo CCAM de referência empregado em [1]. Isso significa que, para se obter os mesmos resultados que [1], é necessário que se adote o mesmo modelo de referência para os transistores CNTFET empregados neste Trabalho, com as mesmas combinações de valores dos parâmetros físico-estruturais e, consequentemente, de parâmetros elétricos que os transistores CNTFET utilizados em [1]. Esse modelo de referência teve os valores de alguns de seus parâmetros elétricos exposto no capítulo Referências Bibliográficas, na Tabela 2.2.

4.3 Características do CNTFET

Após a adoção de um modelo CCAM de referência e a validação de seu uso, parte-se para a caracterização do transistor CNTFET utilizando-se tal modelo. Tal caracterização envolve a obtenção de suas curvas características e de suas curvas de frequência de transição.

4.3.1 Curvas Características do CNTFET

Para obterem-se curvas características do transistor CNTFET semelhantes às obtidas em [1] são empregados 2 esquemáticos de circuito distintos, como mostrado na Figura 4.1.

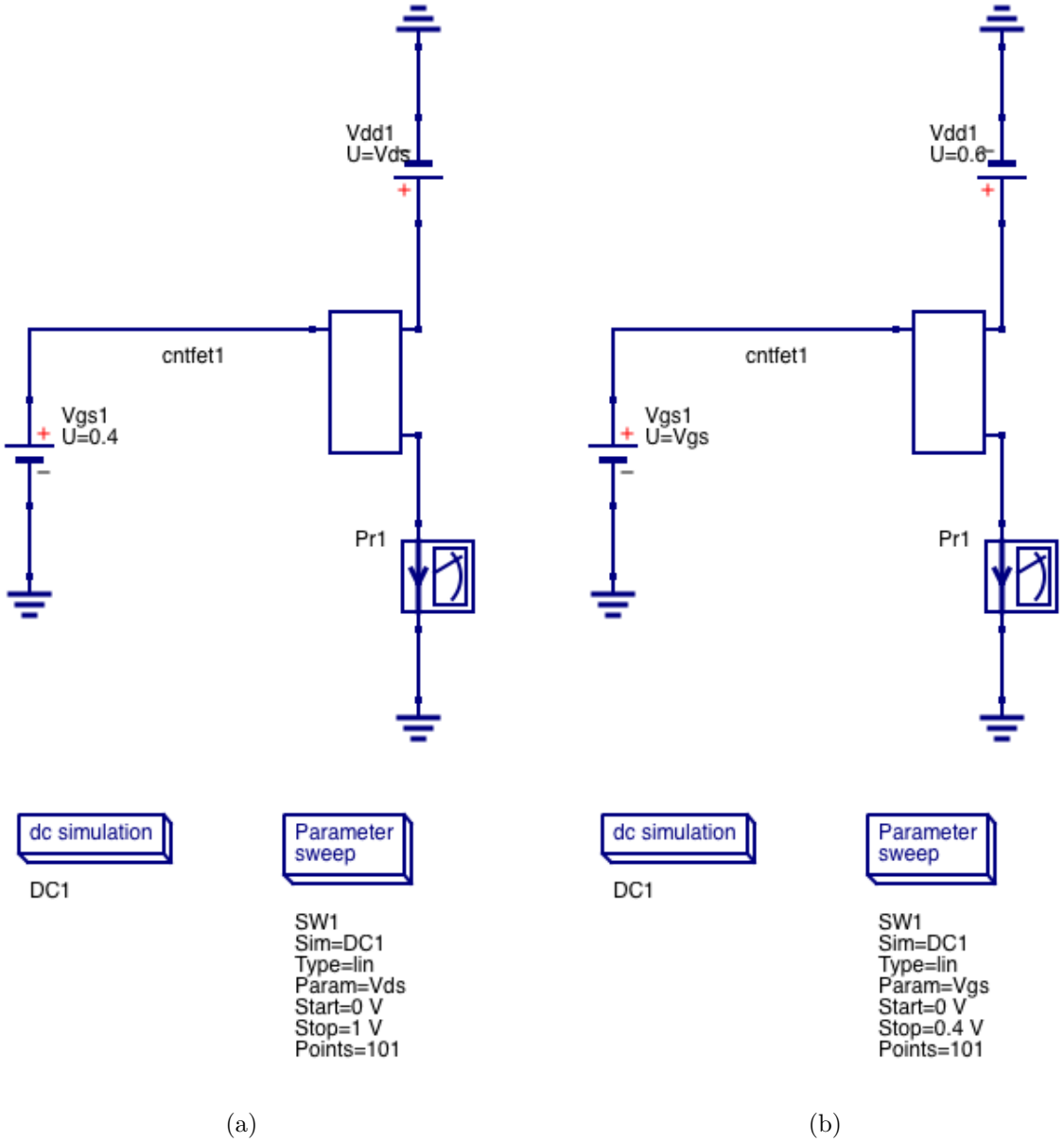
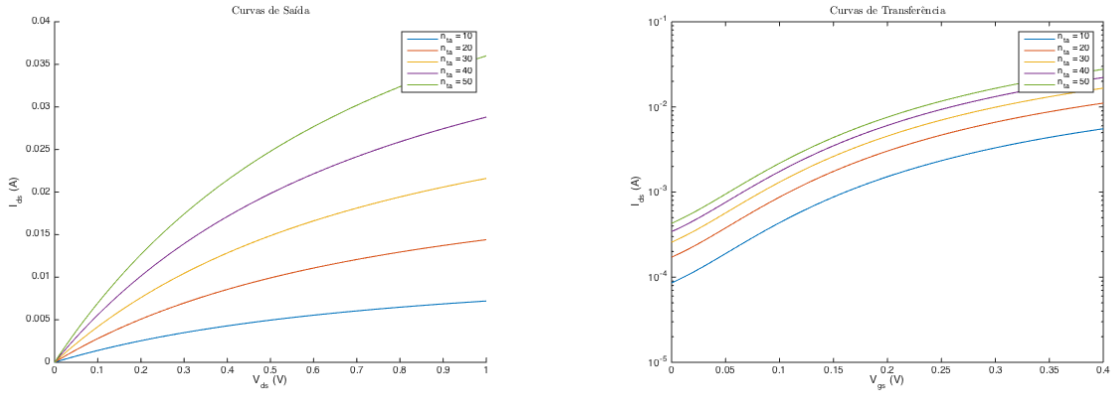


Figura 4.1: Esquemáticos dos circuitos para obtenção de curva de saída para $V_{gs} = 0,4V$ (a) e de curva de transferência para $V_{ds} = 0,6V$ (b).

O esquemático (4.1a) permite obter a curva de saída do transistor CNTFET para uma tensão porta-fonte constante $V_{gs} = 0,4V$ e uma tensão dreno-fonte V_{ds} variando de $0V$ a $1V$ com passos de $0,01V$. O esquemático (4.1b) permite a obtenção da curva de transferência do transistor CNTFET para uma tensão dreno-fonte constante $V_{ds} = 0,6V$ e uma tensão porta-fonte V_{gs} variando de $0V$ a $0,4V$ com passos de $0,004V$.

Utiliza-se o Automated Simulator em cada um dos esquemáticos da Figura 4.1 para variar os parâmetros de interesse de forma a obter combinações idênticas às utilizadas em [1]. Em relação ao modelo CCAM de referência adotado, isso significa definir o número de dedos da porta $n_{gf} = 10$ e o percentual de tubos metálicos $p_{mt} \approx 0\%$ (escolhido $p_{mt} = 0,01\%$). Deve-se também variar a densidade de tubos n_{ta} de $10\,\mu m^{-1}$ a $50\,\mu m^{-1}$ com passos de $10\,\mu m^{-1}$.

Aplicando-se os algoritmos de análise disponíveis em anexo aos resultados obtidos com o uso do Automated Simulator, geramos as curvas de saída e de transferência do transistor CNTFET, como mostrado na Figura 4.2.



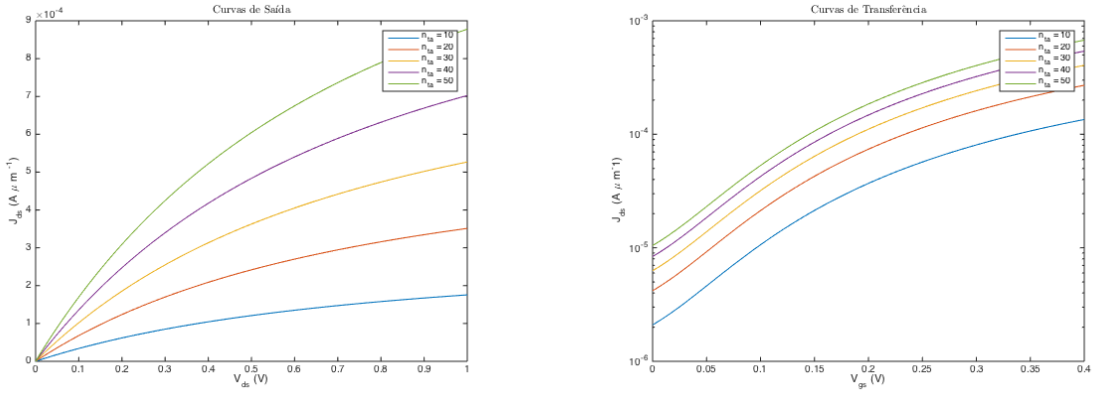
(a) (b)

Figura 4.2: Curvas de saída ($I_{ds} \times V_{ds}$) com $V_{gs} = 0,4V$ (a) e curvas de transferência em escala semi-logarítmica ($I_{ds} \times V_{gs}$) com $V_{ds} = 0,6V$ (b) para diferentes valores da densidade de tubos (n_{ta}) dos transistores CNTFET.

Essas curvas são proporcionais às apresentadas em [1], que mostram a densidade de corrente por largura do canal para um único dedo da porta $J_{ds} = \frac{I_{ds}}{W_{ef}}$ ao invés da corrente I_{ds}. Para se obter curvas idênticas, deve-se dividir os valores das correntes pela largura efetiva da porta ($w_g \times n_{gf}$), como mostrado na Figura 4.3.

4.4 Buffer CML

Depois das fases de simulação e validação do uso do Automated Simulator para a caracterização do transistor CNTFET, modelo CCAM, progride-se para as fases de simulação e validação do uso do Automated Simulator para caracterização de circuitos mais elaborados contendo tais transistores, como o Buffer CML, abordado no capítulo Referências Bibliográficas. Tal caracterização envolve a obtenção das curvas de transferência e de resposta ao degrau do Buffer CML.



(a) (b)

Figura 4.3: Curvas de densidade de saída ($J_{ds} \times V_{ds}$) com $V_{gs} = 0,4V$ (a) e curvas de densidade de transferência em escala semi-logarítmica ($J_{ds} \times V_{gs}$) com $V_{ds} = 0,6V$ (b) para diferentes valores da densidade de tubos (n_{ta}) dos transistores CNTFET.

Para obterem-se curvas de transferência e de resposta ao degrau do Buffer CML semelhantes às obtidas em [1] são empregados dois esquemáticos de circuito distintos, como mostrado na Figura 4.4.

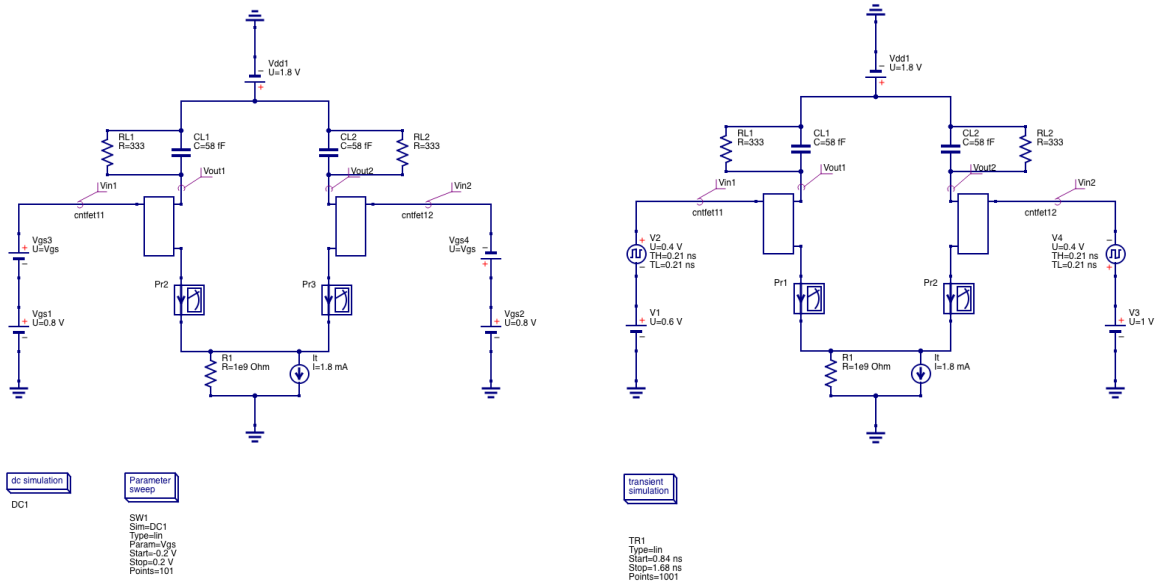
O esquemático (4.4a) permite obter a curva de transferência do Buffer CML para uma tensão de entrada $V_{in,dif}$ variando de $-0,2V$ a $0,2V$. O esquemático (4.1b) permite a obtenção da curva de transferência do transistor CNTFET para uma tensão dreno-fonte constante $V_{ds} = 0,6V$ e uma tensão porta-fonte V_{gs} variando de $0V$ a $0,4V$ com passos de $0,004V$.

Utiliza-se o Automated Simulator em cada um dos esquemáticos da Figura 4.4 para variar os parâmetros de interesse de forma a obter combinações idênticas às utilizadas em [1]. Em relação ao modelo CCAM de referência adotado, isso significa definir o número de dedos da porta $n_{gf} = 10$. Deve-se também variar o percentual de tubos metálicos p_{mt} de aproximadamente 0% (escolhido $p_{mt} = 0,01\%$) a $1,2\%$ com passos de $1,19\%$ e a densidade de tubos n_{ta} de $40\,\mu m^{-1}$ a $60\,\mu m^{-1}$ com passos de $20\,\mu m^{-1}$.

Aplicando-se os algoritmos de análise disponíveis em anexo aos resultados obtidos com o uso do Automated Simulator, geramos as curvas de transferência e de resposta à onda quadrada do Buffer CML, como mostrado nas Figuras 4.5 e 4.6.

4.5 Oscilador em Anel

Depois das fases de simulação e validação do uso do Automated Simulator para a caracterização do Buffer CML, progride-se para as fases de simulação e validação do uso do Automated Simulator para caracterização de circuitos mais elaborados contendo esse estágio, como o Oscilador em Anel (RO), abordado no capítulo Referências Bibliográficas. Tal caracterização envolve a obtenção das curvas de resposta em frequência de ROs com circuito aberto de três e de cinco estágios e das



(a) (b)

Figura 4.4: Esquemáticos dos circuitos para obtenção de curva de transferência ($V_{out} \times V_{in,diff}$) (a) e de curva de resposta à onda quadrada (b).

.

curvas de saída de ROs de cinco estágios.

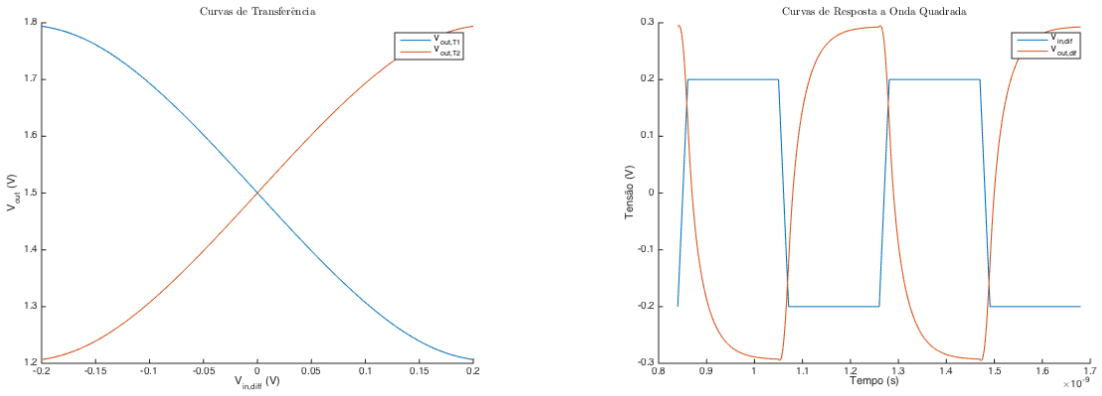
4.5.1 Curvas de Resposta em Frequência de ROs com Circuito Aberto

Para obterem-se curvas de resposta em frequência de ROs com circuito aberto de três e de cinco estágios semelhantes às obtidas em [1] são empregados dois esquemáticos de circuito distintos, como mostrado na Figura 4.7.

O esquemático (4.7a) permite obter a curva de resposta em frequência de um RO de três estágios Buffer CML com circuito aberto para uma tensão de entrada $V_{in,dif}$ variando de $-0,2V$ a $0,2V$. O esquemático (4.7b) permite obter a curva de resposta em frequência de um RO de cinco estágios Buffer CML com circuito aberto para uma tensão de entrada $V_{in,dif}$ variando de $-0,2V$ a $0,2V$.

Utiliza-se o Automated Simulator em cada um dos esquemáticos da Figura 4.7 para variar os parâmetros de interesse de forma a obter combinações idênticas às utilizadas em [1]. Em relação ao modelo CCAM de referência adotado, isso significa definir o número de dedos da porta $n_{gf} = 10$, o percentual de tubos metálicos p_{mt} como aproximadamente 0% (escolhido $p_{mt} = 0,01\%$) e a densidade de tubos $n_{ta} = 40\,\mu\text{m}$.

Aplicando-se os algoritmos de análise disponíveis em anexo aos resultados obtidos com o uso do Automated Simulator, geramos as curvas de resposta em frequência de ROs com circuito aberto de três e de cinco estágios Buffer CML, como mostrado na Figura 4.8.



(a) (b)

Figura 4.5: Curvas de Transferência ($V_{out} \times V_{in,dif}$) com $V_{in,dif}$ variando de $-0,2V$ a $0,2V$ (a) e Curvas de Resposta à Onda Quadrada (b).

4.5.2 Curvas de Saída de ROs de Cinco Estágios Buffer CML com Circuito Fechado

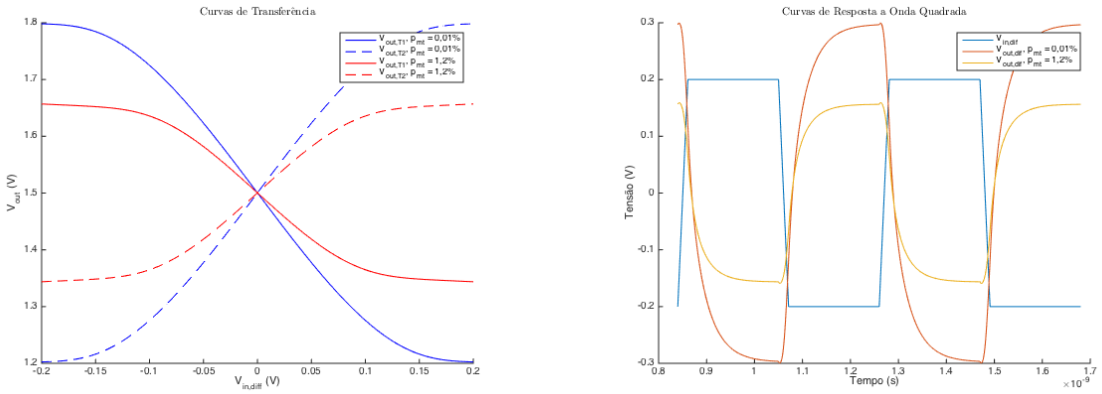
Nesta seção, o simulador QUCS apresentou algumas instabilidades. A primeira delas foi a dificuldade de encontrar um ponto de operação DC a partir dos valores genéricos dos parâmetros internos do modelo CCAM, o que ocasionava um truncamento numérico da simulação. Para superar essa dificuldade, foi necessário atualizar manualmente em ambiente QUCS os valores dos parâmetros internos do modelo CCAM utilizado, fornecendo, assim, valores convergentes para o início da simulação. Em relação ao modelo CCAM de referência adotado, isso significa definir o número de dedos da porta $n_{gf} = 10$, a largura da porta $W_g = 4.1\mu m$, a densidade de tubos $n_{ta} =$ 40 µm e o percentual de tubos metálicos p_{mt} aproximadamente 0% (escolhido $p_{mt} = 0,01\%$).A partir daí, prosseguiu-se normalmente com a metodologia empregada anteriormente.

Para obterem-se curvas de saída de ROs de cinco estágios Buffer CML com circuito fechado semelhantes às obtidas em [1] é empregado apenas um esquemático de circuito, como mostrado na Figura 4.9.

Utiliza-se o Automated Simulator no esquemático mostrado na Figura 4.9 para variar os parâmetros de interesse de forma a obter combinações idênticas às utilizadas em [1]. Em relação ao modelo CCAM atualizado, isso significa variar o percentual de tubos metálicos p_{mt} de aproximadamente 0% (escolhido $p_{mt} = 0,1\%$) a $2,2\%$ com passos de $0,1\%$ e a densidade de tubos n_{ta} de 40 µm a 60 µm com passos de 20 µm. As curvas obtidas encontram-se na Figura 4.10a.

Novamente, o simulador QUCS apresentou instabilidades. Dessa vez, devido à dificuldade de encontrar um ponto de operação DC para valores maiores do percentual de tubos metálicos, o que também ocasionava um truncamento numérico da simulação. Percebeu-se que, para superar esse problema, bastava utilizar a mesma versão do circuito utilizada em [1], ou seja, uma versão do circuito não dividido em subcircuitos. Esse esquemático encontra-se disponível em anexo.

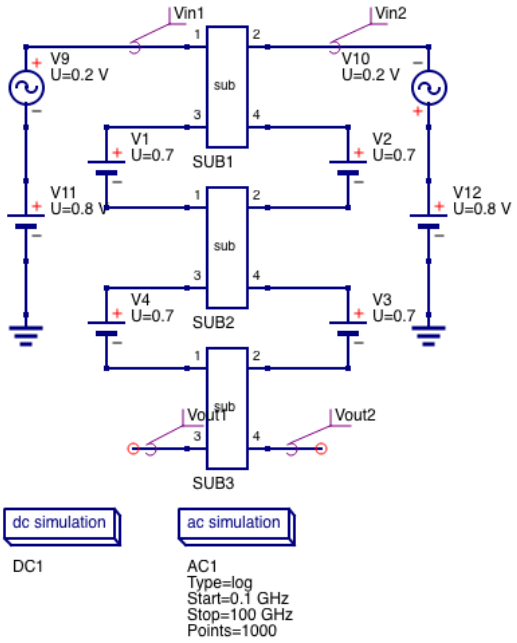
Obtêm-se as curvas restantes por meio da utilização do Automated Simulator no esquemático



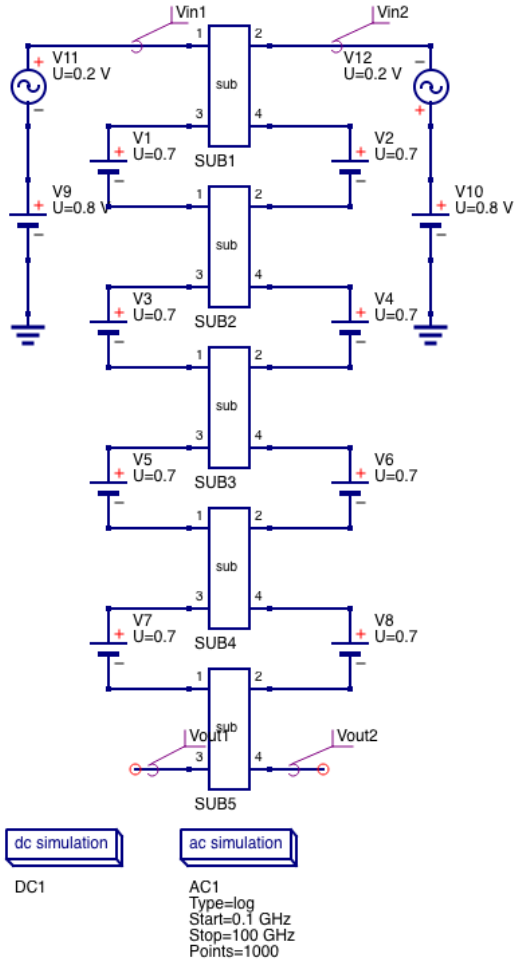
(a) (b)

Figura 4.6: Curvas de Transferência ($V_{out} \times V_{in,dif}$) com $V_{in,dif}$ variando de $-0,2V$ a $0,2V$, $n_{ta} = 60\,\mu m$ e p_{mt} variando de aproximadamente 0% (escolhido $p_{mt} = 0,01\%$) a $0,12\%$ (a) e Curvas de Resposta à Onda Quadrada com $n_{ta} = 60\,\mu m$ e p_{mt} variando de aproximadamente 0% (escolhido $p_{mt} = 0,01\%$) a $0,12\%$ (b).

disponível em anexo, variando-se o percentual de tubos metálicos p_{mt} para os valores de aproximadamente 0% (escolhido $p_{mt} = 0,01\%$), $1,2\%$ e $2,2\%$. Essas curvas encontram-se na Figura 4.10b.



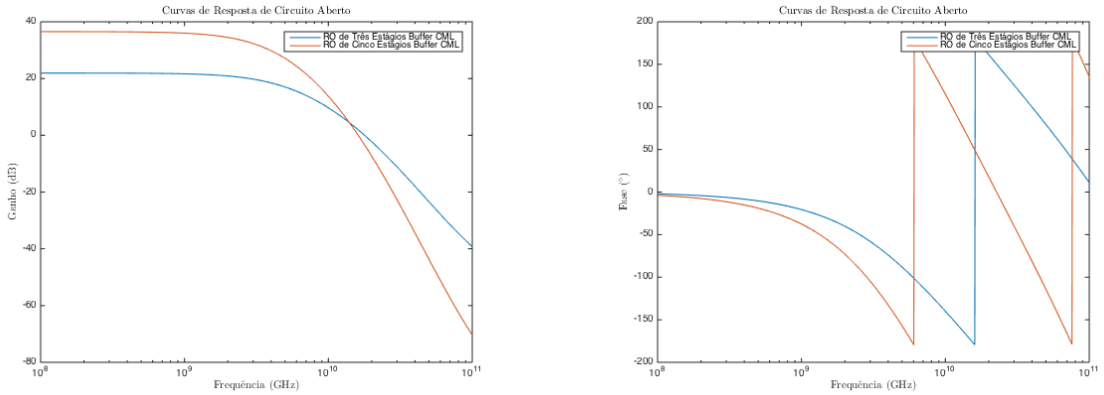
(a)



(b)

Figura 4.7: Esquemáticos dos circuitos para obtenção das curvas de resposta em frequência de um RO de três estágios Buffer CML com circuito aberto (a) e de um RO de cinco estágios Buffer CML com circuito aberto (b).

.



(a) (b)

Figura 4.8: Esquemáticos dos circuitos para obtenção das curvas de resposta em frequência de um RO de três estágios Buffer CML com circuito aberto (a) e de um RO de cinco estágios Buffer CML com circuito aberto (b).

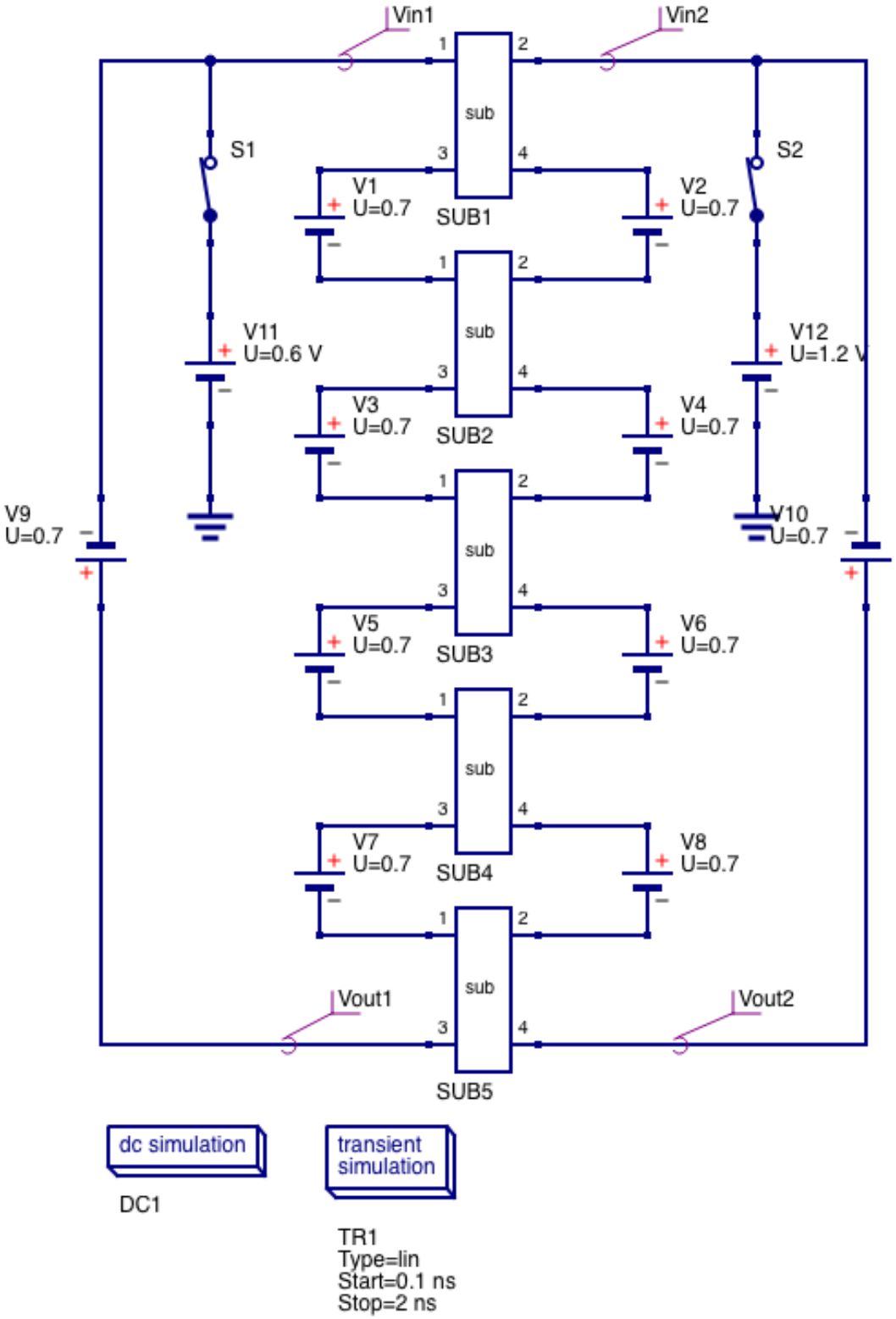
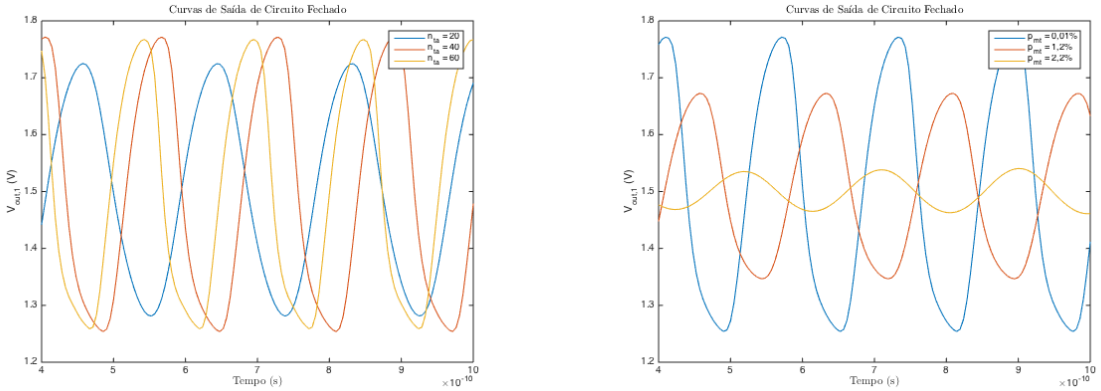


Figura 4.9: Esquemático de circuito para obtenção das curvas de saída de ROs de cinco estágios Buffer CML com circuito fechado.



(a) (b)

Figura 4.10: Curvas de saída de ROs de cinco estágios Buffer CML com circuito fechado para diferentes valores da densidade de tubos n_{ta} (a) e do percentual de tubos metálicos p_{mt} (b).

Capítulo 5

Conclusões

5.1 Conclusão

A geração de benchmarks de circuitos padrões (circuitos de benchmark), tais como o Buffer CML e o Oscilador em Anel, contendo dispositivos ainda não inteiramente caracterizados é um dos principais desafios para a concepção e o design desses dispositivos. Nesse sentido, o presente Trabalho de Conclusão de Curso propôs a sistematização e a automatização de grande parte dessa tarefa. Pode-se dizer que os objetivos traçados no início do Trabalho foram satisfatoriamente atingidos.

Embora ainda haja um longo caminho pela frente, a solução proposta e adotada durante o decurso do Trabalho foi capaz de reduzir consideravelmente o tempo despendido para a execução das tarefas propostas. Foi capaz, também, de reproduzir resultados de difícil reprodução de maneira muito mais célere e organizada, gerando Tutoriais com vistas à implementação de um modelo duradouro de trabalho a ser seguido, espera-se, por outros alunos de graduação.

A validação desse novo método foi feita pela comparação direta entre os resultados obtidos neste Trabalho e os resultados obtidos anteriormente por Muthupandian Cheralathan, Martin Claus e Stefan Blawid em [1]. Como esperado, há uma completa correlação entre ambos, o que indica que, de fato, a metodologia está pronta, salvo pequenas diferenças entre modelos de dispositivos, para ser aplicada em outros trabalhos.

5.2 Trabalhos Futuros

Como mencionado, há ainda um longo caminho a ser percorrido. Mais especificamente, para que a metodologia aqui apresentada possa de fato ser amplamente adotada em outros trabalhos, é preciso que se ataque algumas questões principais, dentre as quais se destacam:

- Atualização do software Automated Simulator para variação dos parâmetros de instâncias individualizadas dos dispositivos de interesse;

- Adaptação do Gerador de Cartas de Modelo (Modelcard Generator) para outros modelos de dispositivos e criação de um Tutorial para facilitar sua utilização;
- Otimização de métodos de arquivamento de resultados de simulações, facilitando sua consulta posterior.
- Criação de uma interface gráfica de análise dos resultados;

Sugere-se fortemente que essas questões sejam resolvidas na ordem apresentada. Em particular, a atualização do Automated Simulator para variação dos parâmetros de instâncias individualizadas dos dispositivos, de modo a permitir análises de variabilidade não-correlata, é de extrema importância para a simulação de situações próximas à real, de variações aleatórias nos parâmetros dos dispositivos durante sua fabricação.

5.3 Considerações Finais

De modo geral, o desenvolvimento deste Trabalho constituiu um imenso desafio, em especial no tocante à programação do Automated Simulator. Fica patente que a familiaridade com esse tipo de ferramenta é cada vez mais necessária, principalmente na área de tecnologia, como é o caso do Departamento de Engenharia Elétrica. A documentação precisa dos passos adotados em cada trabalho também constitui uma ferramenta cuja adoção é fundamental para a aceleração da obtenção de futuros resultados.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CHERALATHAN, M.; CLAUS, M.; BLAWID, S. Projected tolerances of carbon nanotube current-mode logic to process variability. *IEEE Transactions on Circuits and Systems II: Express Briefs*, PP, n. 99, p. 1–1, 2017.

[2] LUO, J. et al. Compact model for carbon nanotube field-effect transistors including nonidealities and calibrated with experimental data down to 9-nm gate length. *IEEE Transactions on Electron Devices*, v. 60, n. 6, p. 1834–1843, June 2013.

[3] TULEVSKI, G. S.; FRANKLIN, A. D.; AFZALI, A. High purity isolation and quantification of semiconducting carbon nanotubes via column chromatography. *ACS Nano*, v. 7, n. 4, p. 2971–2976, 2013.

[4] SCHRÖTER, M. et al. A semiphysical large-signal compact carbon nanotube fet model for analog rf applications. *IEEE Transactions on Electron Devices*, v. 62, n. 1, p. 52–60, Jan 2015.

[5] SCHROTER, M.; HAFERLACH, M.; CLAUS, M. *CCAM Compact Carbon Nanotube Field-Effect Transistor Mode*. Oct 2015.

[6] MICHAL, V. On the low-power design, stability improvement and frequency estimation of the cmos ring oscillator. In: *Proceedings of 22nd International Conference Radioelektronika 2012*. [S.l.: s.n.], 2012. p. 1–4.

[7] RABAEY, J. M. *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[8] MANDAL, M.; SARKAR, B. C. Ring oscillators: Characteristics and applications. In: . [S.l.: s.n.], 2010.

[9] PESETSKI, A. A. et al. A 500 mhz carbon nanotube transistor oscillator. *Applied Physics Letters*, v. 93, n. 12, p. 123506, 2008.

[10] BRINSON, M. E.; JAHN, S. Compact device modeling for established and emerging technologies with the qucs gpl circuit simulator. In: *2009 MIXDES-16th International Conference Mixed Design of Integrated Circuits Systems*. [S.l.: s.n.], 2009. p. 39–44.

[11] CLAUS, M.; SCHRÖTER, M. Design study of cnt transistor layouts for analog circuits. v. 3, p. 566–569, 01 2009.

[12] CLAUS, M. et al. High-frequency benchmark circuit design for a sub 50 nm cntfet technology. In: *2013 SBMO/IEEE MTT-S International Microwave Optoelectronics Conference (IMOC)*. [S.l.: s.n.], 2013. p. 1–5.

[13] SCHROTER, M. et al. Carbon nanotube fet technology for radio-frequency electronics: State-of-the-art overview. *IEEE Journal of the Electron Devices Society*, v. 1, n. 1, p. 9–20, Jan 2013.

[14] MOTHES, S.; CLAUS, M.; SCHRÖTER, M. Toward linearity in schottky barrier cntfets. *IEEE Transactions on Nanotechnology*, v. 14, n. 2, p. 372–378, March 2015.

[15] NUNES, J. P. L. Circuitos digitais nanoeletrônicos. In: . [S.l.: s.n.], 2015.

[16] CARDOSO, R. F. T. *Análise do impacto dos parâmetros da tecnologia e do desenho no ganho de amplificadores de um estágio baseados nos transistores de nanotubos de carbono.*

[17] JÚNIOR, P. A. de A. *Projeto de um amplificador de baixo ruído e de um misturador de frequências para um transceptor Zigbee (2,4 GHz).*

[18] CAMPOS, R. S. *Modelagem de um transceptor Zigbee utilizando a linguagem Verilog-AMS.*

[19] RAZAVI, B. *Fundamentos de microeletrônica*. [S.l.]: LTC.

[20] ALLEN, P.; HOLBERG, D. *CMOS Analog Circuit Design*. [S.l.]: Oxford University Press, 2002. (Oxford series in electrical and computer engineering).

[21] VOINIGESCU, S. *High-Frequency Integrated Circuits*. [S.l.]: Cambridge University Press, 2013. (The Cambridge RF and Microwave Engineering Series).

[22] NANOHUB. Disponível em: <https://nanohub.org/>.

[23] QUCS. Disponível em: <http://qucs.sourceforge.net/>.

[24] MATLAB Documentation.

ANEXOS

I. DIAGRAMAS ESQUEMÁTICOS

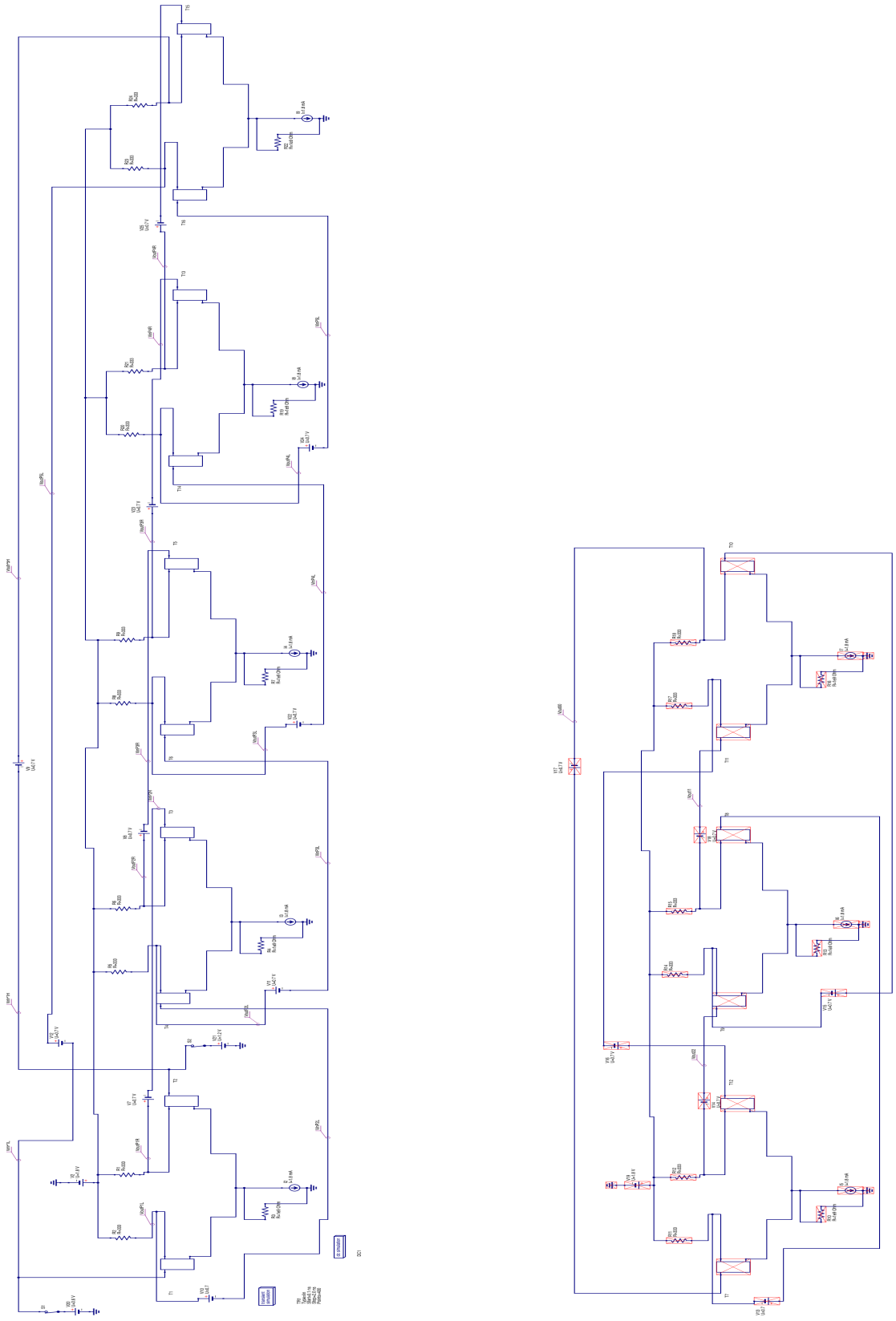


Figura I.1: Esquemático do RO de cinco estágios CML Buffer utilizado em [1].

II. CÓDIGOS DO AUTOMATED SIMULATOR

```
function launchAutomatedSimulator()

setup;

end
```

```
function setup()

setupScreen = figure('Name','setup',...
    'IntegerHandle','off',...
    'MenuBar','none',...
    'Visible','off');

guidata(setupScreen,struct('circuitNetlist','/Users/jpleite/.qucs/netlist.↙
txt',...
↙
'modelcardGeneratorFolder','/Users/jpleite/Documents/Estudo/TCC/02_Methods↙
/Matlab/ModelCard Generator',...
↙
'saveDirectory','/Users/jpleite/Documents/Estudo/TCC/04_Results/Simulation↙
s'));

setupScreenData = guidata(setupScreen);

title = uicontrol('Parent',setupScreen,...
    'Style','text',...
    'String','Automated Simulator',...
    'FontSize',24,...
    'FontWeight','bold',...
    'Units','normalized',...
    'Position',[0.25 0.85 0.5 0.1],...
    'Tag','title');

setupPanel = uipanel('Title','Setup',...
    'FontSize',20,...
    'FontWeight','bold',...
    'Position',[.1 .1 .8 .7]);

chooseCircuitNetlist = uicontrol('Parent',setupPanel,...
    'Style','text',...
    'String','Choose Circuit Netlist:',...
    'FontSize',12,...
    'FontWeight','bold',...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.1 0.85 0.5 0.1],...
    'Tag','chooseCircuitNetlist');

circuitNetlist = uicontrol('Parent',setupPanel,...
    'Style','edit',...
    'String',setupScreenData.circuitNetlist,...
    'FontSize',10,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.15 0.75 0.5 0.1],...
    'Tag','circuitNetlist');

chooseCircuitNetlistButton = uicontrol('Parent',setupPanel,...
    'Style','pushbutton',...
```

```
    'String','...',...
    'FontSize',10,...
    'Units','normalized',...
    'Position',[0.7 0.75 0.05 0.1],...
    'Tag','chooseCircuitNetlistButton',...
    'Callback', {@chooseCircuitNetlistButton_Callback, circuitNetlist});

chooseModelcardGeneratorFolder = uicontrol('Parent',setupPanel,...
    'Style','text',...
    'String','Choose Modelcard Generator Folder:',...
    'FontSize',12,...
    'FontWeight','bold',...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.1 0.6 0.5 0.1],...
    'Tag','chooseModelcardGeneratorFolder');

modelcardGeneratorFolder = uicontrol('Parent',setupPanel,...
    'Style','edit',...
    'String',setupScreenData.modelcardGeneratorFolder,...
    'FontSize',10,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.15 0.5 0.5 0.1],...
    'Tag','modelcardGeneratorFolder');

chooseModelcardGeneratorFolderButton = uicontrol('Parent',setupPanel,...
    'Style','pushbutton',...
    'String','...',...
    'FontSize',10,...
    'Units','normalized',...
    'Position',[0.7 0.5 0.05 0.1],...
    'Tag','chooseModelcardGeneratorFolderButton',...
    'Callback', {@chooseModelcardGeneratorFolderButton_Callback,↙
modelcardGeneratorFolder});

chooseSaveDirectory = uicontrol('Parent',setupPanel,...
    'Style','text',...
    'String','Choose Save Directory:',...
    'FontSize',12,...
    'FontWeight','bold',...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.1 0.35 0.5 0.1],...
    'Tag','chooseSaveDirectory');

saveDirectory = uicontrol('Parent',setupPanel,...
    'Style','edit',...
    'String',setupScreenData.saveDirectory,...
    'FontSize',10,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.15 0.25 0.5 0.1],...
```

```
    'Tag','saveDirectory');

chooseSaveDirectoryButton = uicontrol('Parent',setupPanel,...
    'Style','pushbutton',...
    'String','...',...
    'FontSize',10,...
    'Units','normalized',...
    'Position',[0.7 0.25 0.05 0.1],...
    'Tag','chooseSaveDirectoryButton',...
    'Callback', {@chooseSaveDirectoryButton_Callback,saveDirectory});

nextButton = uicontrol('Parent',setupPanel,...
    'Style','pushbutton',...
    'String','Next',...
    'FontSize',10,...
    'Units','normalized',...
    'Position',[0.8 0.05 0.15 0.15],...
    'Tag','nextButton',...
    'Callback', {@nextButton_Callback,setupScreen});

setupScreen.Visible = 'on';

end

function chooseCircuitNetlistButton_Callback(hObject,eventdata,↙
circuitNetlist)
    data = guidata(hObject);
    [circuitNetlistFileName,circuitNetlistPathName] = uigetfile('*.txt');
    if (circuitNetlistPathName ~= 0) && (circuitNetlistFileName ~= 0)
        data.circuitNetlist = [circuitNetlistPathName,↙
circuitNetlistFileName];
        circuitNetlist.String = data.circuitNetlist;
    end
    guidata(hObject,data);
end

function chooseModelcardGeneratorFolderButton_Callback(hObject,eventdata,↙
modelcardGeneratorFolder)
    data = guidata(hObject);
    modelcardGeneratorFolderPathName = uigetdir;
    if modelcardGeneratorFolderPathName ~= 0
        data.modelcardGeneratorFolder = modelcardGeneratorFolderPathName;
        modelcardGeneratorFolder.String = data.modelcardGeneratorFolder;
    end
    guidata(hObject,data);
end

function chooseSaveDirectoryButton_Callback(hObject,eventdata,↙
saveDirectory)
    data = guidata(hObject);
    saveDirectoryPathName = uigetdir;
    if saveDirectoryPathName ~= 0
        data.saveDirectory = saveDirectoryPathName;
```

```
        saveDirectory.String = data.saveDirectory;
    end
    guidata(hObject,data);
end

function nextButton_Callback(hObject,eventdata,setupScreen)
    data = guidata(hObject);
    chooseParametersToBeScaled(setupScreen,data);
end
```

```
function chooseParametersToBeScaled(setupScreen, setupScreenData)

chooseParametersToBeScaledScreen = figure↙
('Name','chooseParametersToBeScaled',...
    'IntegerHandle','off',...
    'MenuBar','none',...
    'Visible','off');

guidata(chooseParametersToBeScaledScreen,setupScreenData);

setupScreen.Visible = 'off';

chooseParametersToBeScaledScreenData = guidata↙
(chooseParametersToBeScaledScreen);

chooseParametersToBeScaledScreenData.parameters = {'n_gf', 'w_g', 'l_g',↙
'l_gsd', 'l_sd',...
    'd_cnt', 'n_ta', 'p_mt', 'h_mg', 'h_msd', 'h_m1', 't_ox', 'w_ms',↙
'w_mg',...
    'w_do', 'eps_is', 'eps_ox', 'eps_ins'};

guidata(chooseParametersToBeScaledScreen,↙
chooseParametersToBeScaledScreenData);

title = uicontrol('Parent',chooseParametersToBeScaledScreen,...
    'Style','text',...
    'String','Automated Simulator',...
    'FontSize',24,...
    'FontWeight','bold',...
    'Units','normalized',...
    'Position',[0.25 0.85 0.5 0.1],...
    'Tag','title');

chooseParametersToBeScaledPanel = uipanel('Title','Choose Parameters To Be↙
Scaled',...
    'FontSize',20,...
    'FontWeight','bold',...
    'Position',[.1 .1 .8 .7]);

chooseParametersToBeScaled = uicontrol('Parent',↙
chooseParametersToBeScaledPanel,...
    'Style','listbox',...
    'String',chooseParametersToBeScaledScreenData.parameters,...
    'Fontsize',10,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.1 0.25 0.8 0.65],...,
    'Max',2,...
    'Tag','chooseParametersToBeScaled',...
    'Callback', @chooseParametersToBeScaled_Callback);

nextButton = uicontrol('Parent',chooseParametersToBeScaledPanel,...
    'Style','pushbutton',...
```

```
    'String','Next',...
    'FontSize',10,...
    'Units','normalized',...
    'Position',[0.8 0.05 0.15 0.15],...
    'Tag','nextButton',...
    'Callback', {@nextButton_Callback,chooseParametersToBeScaledScreen});

chooseParametersToBeScaledScreen.Visible = 'on';

end

function chooseParametersToBeScaled_Callback(hObject,eventdata)
    data = guidata(hObject);
    for counter = 1:size(hObject.Value,2)
        data.parametersToBeScaled{counter} = data.parameters{hObject.Value↙
(counter)};
    end
    guidata(hObject,data);
end

function nextButton_Callback(hObject,eventdata,↙
chooseParametersToBeScaledScreen)
    chooseParametersToBeScaledScreenData = guidata↙
(chooseParametersToBeScaledScreen);
    for parametersToBeScaledCounter = 1:size↙
(chooseParametersToBeScaledScreenData.parametersToBeScaled,2)
        defineParametersValues(chooseParametersToBeScaledScreen,↙
chooseParametersToBeScaledScreenData,parametersToBeScaledCounter);
        uiwait;
        chooseParametersToBeScaledScreenData = guidata↙
(chooseParametersToBeScaledScreen);
        guidata(chooseParametersToBeScaledScreen,↙
chooseParametersToBeScaledScreenData);
    end
    simulate(chooseParametersToBeScaledScreen,↙
chooseParametersToBeScaledScreenData);
end
```

```
function defineParametersValues(chooseParametersToBeScaledScreen,↙
chooseParametersToBeScaledScreenData, parametersToBeScaledCounter)

defineParametersValuesScreen = figure('Name','defineParametersValues',...
    'IntegerHandle','off',...
    'MenuBar','none',...
    'Visible','off');

guidata(defineParametersValuesScreen,↙
chooseParametersToBeScaledScreenData);

defineParametersValuesScreenData = guidata(defineParametersValuesScreen);

guidata(defineParametersValuesScreen,defineParametersValuesScreenData);

defineParameterValuesPanel = uipanel('Title','Define Parameter Values',...
    'FontSize',20,...
    'FontWeight','bold');

defineParameterValues = uicontrol('Parent',defineParameterValuesPanel,...
    'Style','text',...
    'String',defineParametersValuesScreenData.parametersToBeScaled↙
{parametersToBeScaledCounter},...
    'FontSize',12,...
    'FontWeight','bold',...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.1 0.85 0.5 0.1],...
    'Tag','defineParameterValues');

parameterInitialValue = uicontrol('Parent',defineParameterValuesPanel,...
    'Style','edit',...
    'String','Initial Value',...
    'FontSize',10,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.15 0.45 0.15 0.1],...
    'Tag','parameterInitialValue',...
    'Callback',{@parameterInitialValue_Callback,↙
parametersToBeScaledCounter});

parameterFinalValue = uicontrol('Parent',defineParameterValuesPanel,...
    'Style','edit',...
    'String','Final Value',...
    'FontSize',10,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.4 0.45 0.15 0.1],...
    'Tag','parameterFinalValue',...
    'Callback',{@parameterFinalValue_Callback,↙
parametersToBeScaledCounter});

parameterStep = uicontrol('Parent',defineParameterValuesPanel,...
```

```
    'Style','edit',...
    'String','Step',...
    'FontSize',10,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[0.65 0.45 0.15 0.1],...
    'Tag','parameterStep',...
    'Callback',{@parameterStep_Callback,parametersToBeScaledCounter});

nextButton = uicontrol('Parent',defineParameterValuesPanel,...
    'Style','pushbutton',...
    'String','Next',...
    'FontSize',10,...
    'Units','normalized',...
    'Position',[0.8 0.05 0.15 0.15],...
    'Tag','nextButton',...
    'Callback', {@nextButton_Callback,chooseParametersToBeScaledScreen,↙
defineParametersValuesScreen});

defineParametersValuesScreen.Visible = 'on';

end

function nextButton_Callback(hObject,eventdata,↙
chooseParametersToBeScaledScreen,defineParametersValuesScreen)
    data = guidata(hObject);
    guidata(chooseParametersToBeScaledScreen,data);
    defineParametersValuesScreen.Visible = 'off';
    uiresume;
end

function parameterInitialValue_Callback(hObject,eventdata,counter)
    data = guidata(hObject);
    data.parameterInitialValue(counter) = str2num(hObject.String);
    guidata(hObject,data);
end

function parameterFinalValue_Callback(hObject,eventdata,counter)
    data = guidata(hObject);
    data.parameterFinalValue(counter) = str2num(hObject.String);
    guidata(hObject,data);
end

function parameterStep_Callback(hObject,eventdata,counter)
    data = guidata(hObject);
    data.parameterStep(counter) = str2num(hObject.String);
    guidata(hObject,data);
end
```

```
function simulate(chooseParametersToBeScaledScreen,↙
chooseParametersToBeScaledScreenData)

simulateScreen = figure('Name','simulate',...
    'IntegerHandle','off',...
    'MenuBar','none',...
    'Visible','off');

guidata(simulateScreen,chooseParametersToBeScaledScreenData);

chooseParametersToBeScaledScreen.Visible = 'off';

simulateScreenData = guidata(simulateScreen);

guidata(simulateScreen,simulateScreenData)

title = uicontrol('Parent',simulateScreen,...
    'Style','text',...
    'String','Automated Simulator',...
    'FontSize',24,...
    'FontWeight','bold',...
    'Units','normalized',...
    'Position',[0.25 0.85 0.5 0.1],...
    'Tag','title');

simulatePanel = uipanel('Title','Simulate',...
    'FontSize',20,...
    'FontWeight','bold',...
    'Position',[.1 .1 .8 .7]);

simulate = uicontrol('Parent',simulatePanel,...
    'Style','text',...
    'String','The Setup has been completed.',...
    'FontSize',16,...
    'FontWeight','bold',...
    'Units','normalized',...
    'Position',[0.1 0.55 0.8 0.2],...
    'Tag','setup');

simulateButton = uicontrol('Parent',simulatePanel,...
    'Style','pushbutton',...
    'String','Simulate',...
    'FontSize',14,...
    'Units','normalized',...
    'Position',[0.25 0.1 0.5 0.2],...
    'Tag','simulateButton',...
    'Callback', @simulateButton_Callback);

simulateScreen.Visible = 'on';

end

function simulateButton_Callback(hObject,eventdata)
```

```
    data = guidata(hObject);
    automatedSimulator(data);
end
```

```
function automatedSimulator(data)

data = data;
circuitNetlist = data.circuitNetlist;
modelcardGeneratorFolder = data.modelcardGeneratorFolder;
saveDirectory = data.saveDirectory;

workingDirectory = cd(modelcardGeneratorFolder);

modelcard = modelcardGenerator(data.parametersToBeScaled,data.↙
parameterInitialValue,data.parameterFinalValue,data.parameterStep);

cd(workingDirectory);

originalNetlist = getOriginalNetlist(circuitNetlist);

simulationsDirectory = createSimulationsDirectory(originalNetlist,↙
saveDirectory);

cd(simulationsDirectory);

save('modelcard','modelcard');
save('simulationData','data');

cd(workingDirectory);

[originalPATH, updatedPATH] = getPATHS(':/usr/local/bin');

updatePATH(updatedPATH);

parametersValuesCombinationsFieldnames = fieldnames(modelcard);
parametersNames = fieldnames(modelcard.↙
(parametersValuesCombinationsFieldnames{1}));

for counter = 1:size(fieldnames(modelcard),1)

        parametersValues = struct2cell(modelcard.↙
(parametersValuesCombinationsFieldnames{counter}));

        modifiedNetlist = modifyNetlist(originalNetlist,parametersNames,↙
parametersValues);

        saveModifiedNetlist(simulationsDirectory,↙
parametersValuesCombinationsFieldnames{counter},modifiedNetlist);

        !qucsator -i netlist.txt -o simulationResults.dat;

        cd(workingDirectory);

end

updatePATH(originalPATH);
```

```
end
```

```
function originalNetlist = getOriginalNetlist(circuitNetlist)

originalNetlist = fileread(circuitNetlist);

end
```

```
function simulationsDirectory = createSimulationsDirectory↙
(originalNetlist,saveDirectory)

    circuitNameEndPoint = regexp(originalNetlist,'.sch','once')-1;
    circuitNameStartPoint = max(regexp(originalNetlist(1:↙
circuitNameEndPoint),'/'))+1;
    circuitName = originalNetlist(circuitNameStartPoint:↙
circuitNameEndPoint);
    cd(saveDirectory);
    date = datetime;
    mkdir([circuitName ' ' datestr(date)]);
    simulationsDirectory = [saveDirectory '/' circuitName ' ' datestr↙
(date)];

end
```

```
function [originalPATH, updatedPATH] = getPATHS(qucsatorDirectory)

originalPATH = getenv('PATH');
updatedPATH = [originalPATH qucsatorDirectory];

end
```

```
function updatePATH(updatedPATH)

setenv('PATH', updatedPATH);

end
```

```
function modifiedNetlist = modifyNetlist(originalNetlist,parametersNames,↙
parametersValues)

numberOfCNTFET = size(regexp(originalNetlist,'cntfet1:'),2);
parametersNamesLocations = cell(size(parametersNames,1),size↙
(parametersNames,2));

for counter1 = 1:size(parametersNames,1)
    parametersNamesLocations{counter1,1} = regexp(originalNetlist,[' '↙
parametersNames{counter1}]);
end

modifiedNetlist = originalNetlist;

for counter1 = 1:size(parametersNamesLocations,1)
    for counter2 = 1:numberOfCNTFET
        modificationStartPoint = parametersNamesLocations{counter1,1}↙
(counter2)+regexp(modifiedNetlist(parametersNamesLocations{counter1,1}↙
(counter2):size(modifiedNetlist,2)),'"','once');
        modificationEndPoint = modificationStartPoint+regexp↙
(modifiedNetlist(modificationStartPoint:size(modifiedNetlist,↙
2)),'"','once')-2;
        modifiedNetlist = [modifiedNetlist(1:modificationStartPoint-1)↙
num2str(parametersValues{counter1,1}) modifiedNetlist↙
(modificationEndPoint+1:size(modifiedNetlist,2))];
        for counter3 = counter1:size(parametersNames,1)
            parametersNamesLocations{counter3,1} = regexp(modifiedNetlist,↙
[' ' parametersNames{counter3}]);
        end
    end
end

end
```

```
function saveModifiedNetlist(saveDirectory,↙
parametersValuesCombinationsFieldnames,modifiedNetlist)

cd(saveDirectory);
mkdir(parametersValuesCombinationsFieldnames);
cd(parametersValuesCombinationsFieldnames);
fileID = fopen('netlist.txt','w+');
fprintf(fileID,modifiedNetlist);
fclose(fileID);

end
```

III. CÓDIGOS DE ANÁLISE DOS RESULTADOS

```
folder = uigetdir;
data.n_ta10 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta20 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta30 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta40 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta50 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;

outputCurves = figure('Name','outputCurves',...
'IntegerHandle','off');
plot(data.n_ta10(1).data,data.n_ta10(2).data, data.n_ta20(1).data, data.↙
n_ta20(2).data,data.n_ta30(1).data, data.n_ta30(2).data, data.n_ta40(1).↙
data, data.n_ta40(2).data, data.n_ta50(1).data, data.n_ta50(2).data);
legend('n_{ta} = 10','n_{ta} = 20','n_{ta} = 30','n_{ta} = 40','n_{ta} =↙
50');
title('Curvas de Sa\''ida','Interpreter','latex');
xlabel('V_{ds} (V)');
ylabel('I_{ds} (A)');

currentDensityOutputCurves = figure↙
('Name','currentDensityOutputCurves',...
'IntegerHandle','off');
plot(data.n_ta10(1).data,data.n_ta10(2).data/41, data.n_ta20(1).data,↙
data.n_ta20(2).data/41,data.n_ta30(1).data, data.n_ta30(2).data/41, data.↙
n_ta40(1).data, data.n_ta40(2).data/41, data.n_ta50(1).data, data.n_ta50↙
(2).data/41);
legend('n_{ta} = 10','n_{ta} = 20','n_{ta} = 30','n_{ta} = 40','n_{ta} =↙
50');
title('Curvas de Sa\''ida','Interpreter','latex');
xlabel('V_{ds} (V)');
ylabel('J_{ds} (A \mu m ^{-1})');
```

```
folder = uigetdir;
data.n_ta10 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta20 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta30 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta40 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta50 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;

transferCurves = figure('Name','transferCurves',...
'IntegerHandle','off');
semilogy(data.n_ta10(1).data,data.n_ta10(2).data, data.n_ta20(1).data,↙
data.n_ta20(2).data,data.n_ta30(1).data, data.n_ta30(2).data, data.n_ta40↙
(1).data, data.n_ta40(2).data, data.n_ta50(1).data, data.n_ta50(2).data);
legend('n_{ta} = 10','n_{ta} = 20','n_{ta} = 30','n_{ta} = 40','n_{ta} =↙
50');
title('Curvas de Transfer\^encia','Interpreter','latex');
xlabel('V_{gs} (V)');
ylabel('I_{ds} (A)');

transferCurrentDensityCurves = figure↙
('Name','transferCurrentDensityCurves',...
'IntegerHandle','off');
semilogy(data.n_ta10(1).data,data.n_ta10(2).data/41, data.n_ta20(1).data,↙
data.n_ta20(2).data/41,data.n_ta30(1).data, data.n_ta30(2).data/41, data.↙
n_ta40(1).data, data.n_ta40(2).data/41, data.n_ta50(1).data, data.n_ta50↙
(2).data/41);
legend('n_{ta} = 10','n_{ta} = 20','n_{ta} = 30','n_{ta} = 40','n_{ta} =↙
50');
title('Curvas de Transfer\^encia','Interpreter','latex');
xlabel('V_{gs} (V)');
ylabel('J_{ds} (A \mu m^-1)');
```

```
folder = uigetdir;
data.p_mt0_0001 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_012 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;
transferCurves = figure('Name','transferCurves',...
'IntegerHandle','off');
title('Curvas de Transfer\^encia','Interpreter','latex');
xlabel('V_{in,diff} (V)');
ylabel('V_{out} (V)');
hold on;
vInDiff = data.p_mt0_0001(1).data;
plot(vInDiff,data.p_mt0_0001(3).data,'b',vInDiff,data.p_mt0_0001(2).↙
data,'--b',vInDiff,data.p_mt0_012(3).data,'r',vInDiff,data.p_mt0_012(2).↙
data,'--r');
legend('V_{out,T1}, p_{mt} = 0,01%','V_{out,T2}, p_{mt} = 0,01%','V_{out,↙
T1}, p_{mt} = 1,2%','V_{out,T2}, p_{mt} = 1,2%');
```

```
folder = uigetdir;
data.p_mt0_0001 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_012 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;
squareWaveResponseCurves = figure('Name','squareWaveResponseCurves',...
'IntegerHandle','off');
title('Curvas de Resposta a Onda Quadrada','Interpreter','latex');
xlabel('Tempo (s)','Interpreter','latex');
ylabel('Tens~ao (V)','Interpreter','latex');
hold on;
vInDiff = (data.p_mt0_0001(14).data-data.p_mt0_0001(13).data)/2;
vOutDiff.p_mt0_0001 = (data.p_mt0_0001(18).data-data.p_mt0_0001(17).data)↙
/2;
vOutDiff.p_mt0_012 = (data.p_mt0_012(18).data-data.p_mt0_012(17).data)/2;
plot(data.p_mt0_0001(1).data,vInDiff,data.p_mt0_0001(1).data,vOutDiff.↙
p_mt0_0001,data.p_mt0_0001(1).data,vOutDiff.p_mt0_012);
legend('V_{in,dif}','V_{out,dif}, p_{mt} = 0,01%', 'V_{out,dif}, p_{mt} =↙
1,2%');
```

```
folder = uigetdir;
data.threeCMLBufferStage = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.fiveCMLBufferStage = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;

frequency = data.threeCMLBufferStage(1).data;
vInDiff = (data.threeCMLBufferStage(13).data-data.threeCMLBufferStage(12).data)/2;
vOutDiff.threeCMLBufferStage = (data.threeCMLBufferStage(23).data-data.threeCMLBufferStage(22).data)/2;
vOutDiff.fiveCMLBufferStage = (data.fiveCMLBufferStage(35).data-data.fiveCMLBufferStage(34).data)/2;
gain.threeCMLBufferStage = abs(vOutDiff.threeCMLBufferStage./vInDiff);
gainDB.threeCMLBufferStage = 10*log(gain.threeCMLBufferStage);
gain.fiveCMLBufferStage = abs(vOutDiff.fiveCMLBufferStage./vInDiff);
gainDB.fiveCMLBufferStage = 10*log(gain.fiveCMLBufferStage);
phase.threeCMLBufferStage = radtodeg(angle(vOutDiff.threeCMLBufferStage./vInDiff));
phase.fiveCMLBufferStage = radtodeg(angle(vOutDiff.fiveCMLBufferStage./vInDiff));

openLoopGainResponseCurves = figure
('Name','openLoopGainResponseCurves',...
'IntegerHandle','off');
semilogx(frequency,gainDB.threeCMLBufferStage,frequency,gainDB.fiveCMLBufferStage);
title('Curvas de Resposta de Circuito Aberto','Interpreter','latex');
xlabel('Frequ\^encia (GHz)','Interpreter','latex');
ylabel('Ganho (dB)','Interpreter','latex');
legend('RO de Tr?s Est?gios Buffer CML','RO de Cinco Est?gios Buffer CML');

openLoopPhaseResponseCurves = figure
('Name','openLoopPhaseResponseCurves',...
'IntegerHandle','off');
semilogx(frequency,phase.threeCMLBufferStage,frequency,phase.fiveCMLBufferStage);
title('Curvas de Resposta de Circuito Aberto','Interpreter','latex');
xlabel('Frequ\^encia (GHz)','Interpreter','latex');
ylabel('Fase ($^{\circ}$)','Interpreter','latex');
legend('RO de Tr?s Est?gios Buffer CML','RO de Cinco Est?gios Buffer CML');
```

```
folder = uigetdir;
data.n_ta20 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta40 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.n_ta60 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;

folder = uigetdir;
data.p_mt0_0001 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_012 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_022 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;

time = data.n_ta20(1).data;

closedLoopNtaOutputCurves = figure('Name','closedLoopNtaOutputCurves',...
'IntegerHandle','off');
plot(time,data.n_ta20(18).data,time,data.n_ta40(18).data,time,data.n_ta60↙
(18).data);
title('Curvas de Sa\''ida de Circuito Fechado','Interpreter','latex');
xlabel('Tempo (s)','Interpreter','latex');
ylabel('V_{out,1} (V)');
legend('n_{ta} = 20','n_{ta} = 40','n_{ta} = 60');
axis([0.4*10^-9 1*10^-9 1.2 1.8]);

closedLoopPmtOutputCurves = figure('Name','closedLoopPmtOutputCurves',...
'IntegerHandle','off');
plot(time,data.p_mt0_0001(27).data,time,data.p_mt0_012(27).data,time,data.↙
p_mt0_022(27).data);
title('Curvas de Sa\''ida de Circuito Fechado','Interpreter','latex');
xlabel('Tempo (s)','Interpreter','latex');
ylabel('V_{out,1} (V)');
legend('p_{mt} = 0,01%','p_{mt} = 1,2%','p_{mt} = 2,2%');
axis([0.4*10^-9 1*10^-9 1.2 1.8]);
```

IV. TUTORIAIS

Qucs

Quite Universal Circuit Simulator

Tutorial – Instalando Qucs em Mac OS

versão 1.0

Universidade de Brasília

João Pedro Leite Nunes
Universidade de Brasília

ENE-LDCI Tutorial Qucs – Mac OS João Pedro

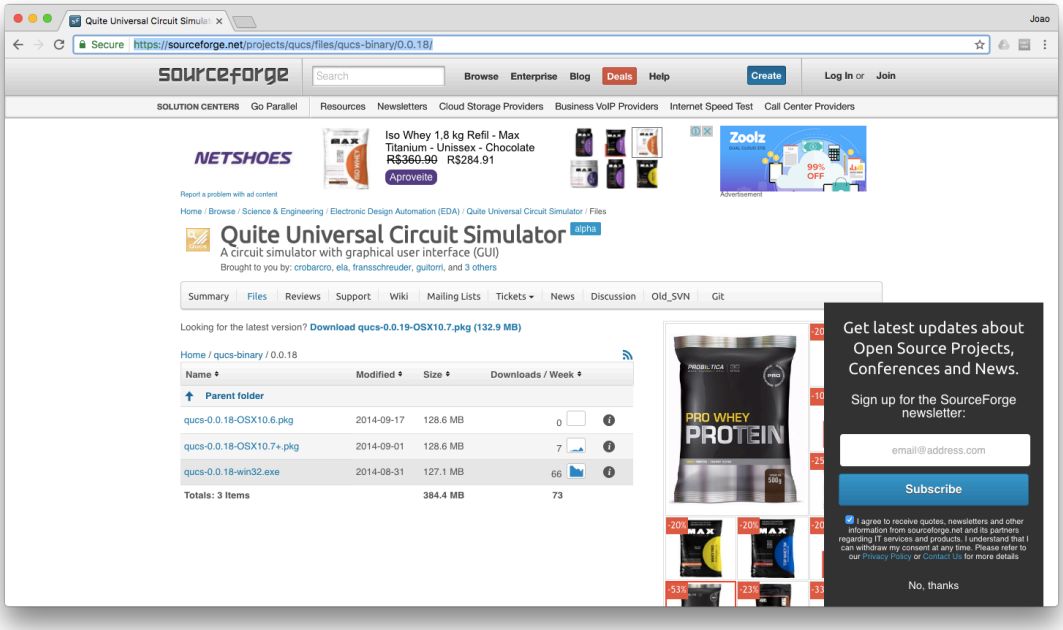
Introdução

O método de instalação do software Qucs (Quite Universal Circuit Simulator) em plataformas Mac OS é diferente do método de instalação do mesmo software em plataformas Windows, em especial para a simulação de circuitos contendo componentes descritos em Verilog-a. Este Tutorial visa guiar o usuário através dos passos necessários para a instalação do Qucs com todas as funcionalidades pertinentes.

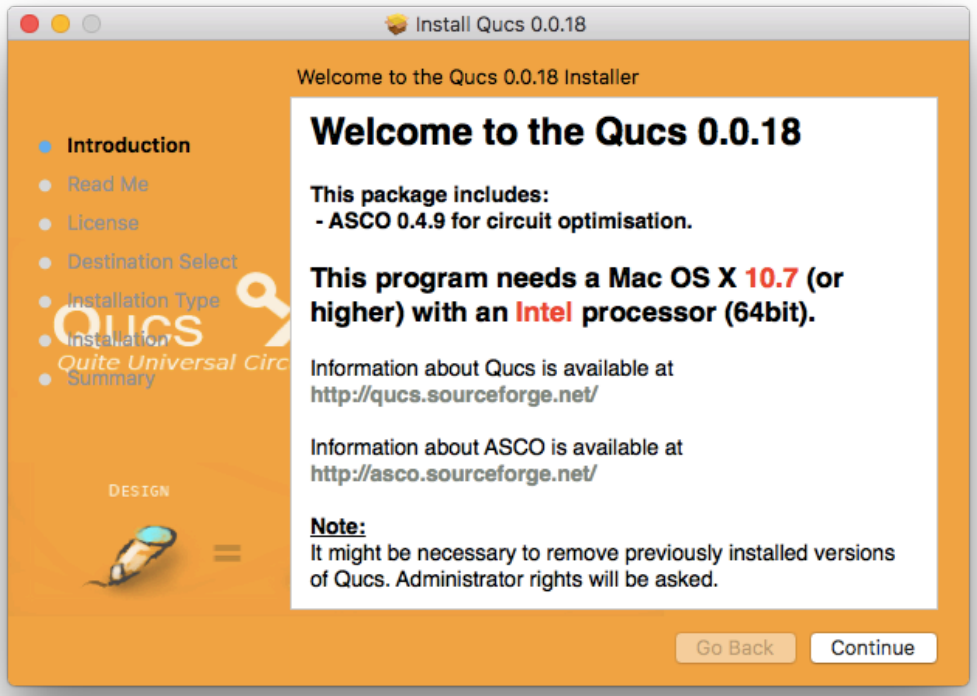
Instalação do Qucs

O Qucs é um software livre de simulação de circuitos que conta também com uma interface gráfica (GUI) para o desenho dos esquemáticos.

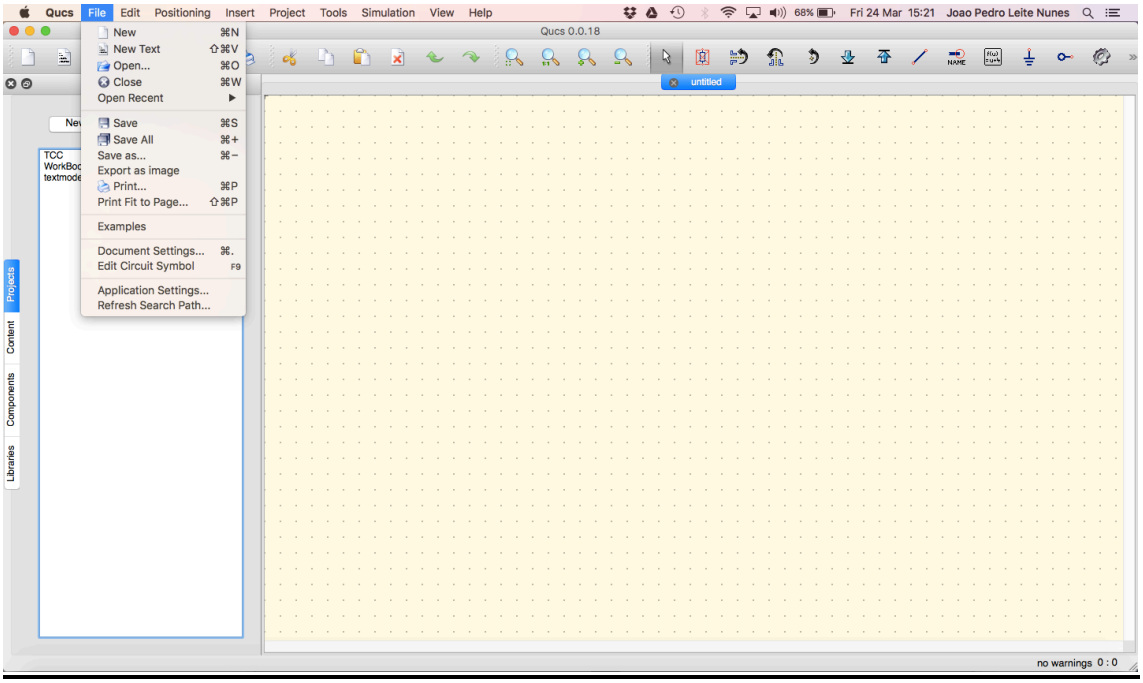
1. Abra algum navegador de internet (Google Chrome, Apple Safari, Mozilla Firefox) e acesse o link: https://sourceforge.net/projects/qucs/files/qucs-binary/0.0.18/. Escolha o arquivo do instalador correspondente ao seu sistema operacional para inciar seu download.



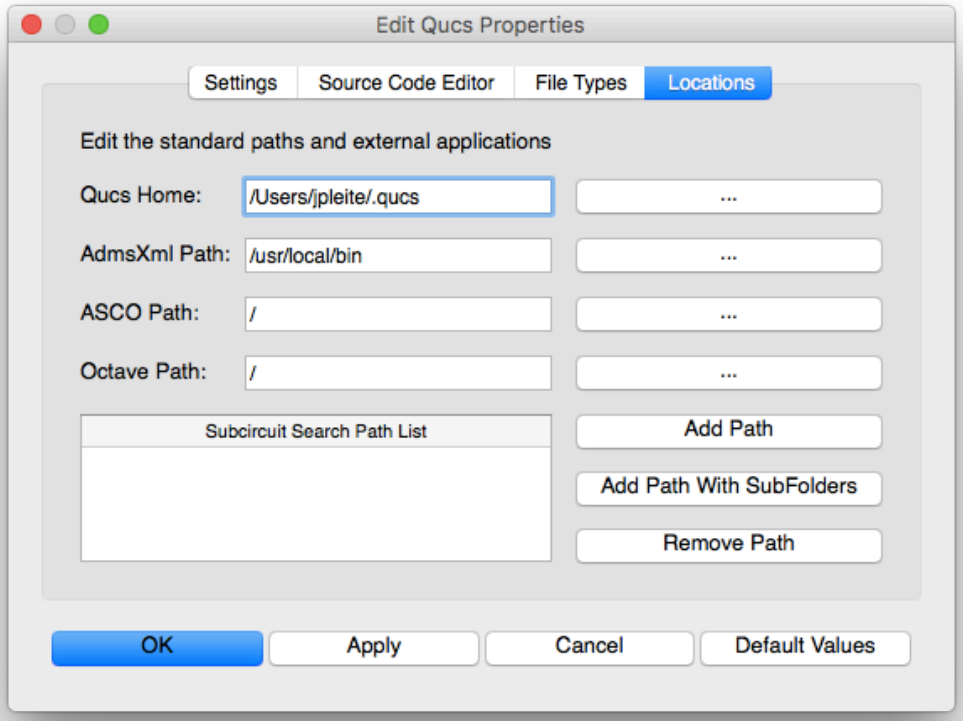
2. Execute o instalador. Ele irá guiá-lo pelo resto do processo.



3. Abra o Qucs. Clique no menu "File" e em seguida no submenu "Application Settings".



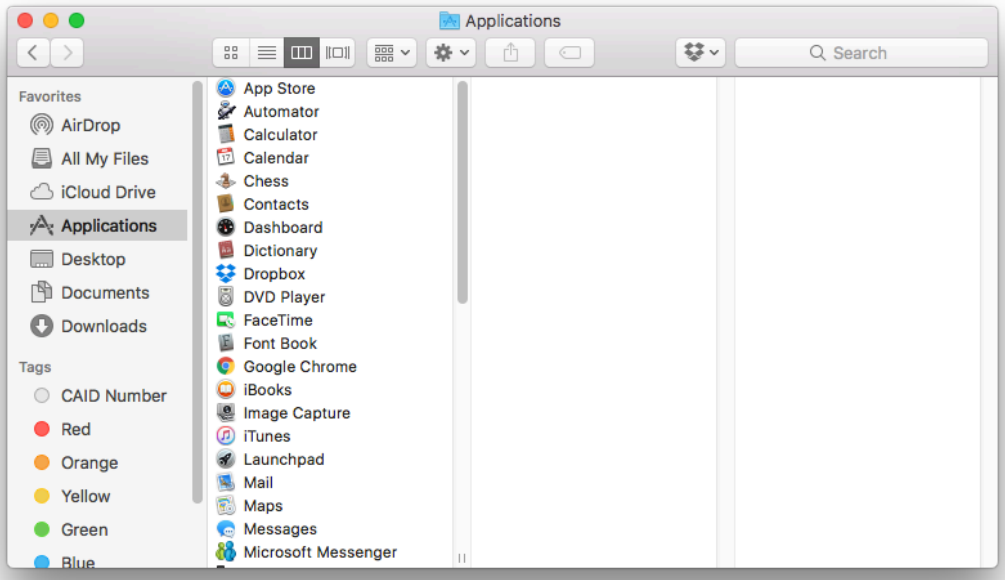
4. Na janela "Edit Qucs Properties", clique em "Locations" e atualize o campo "AdmsXml Path" para que ele fique igual ao da figura.



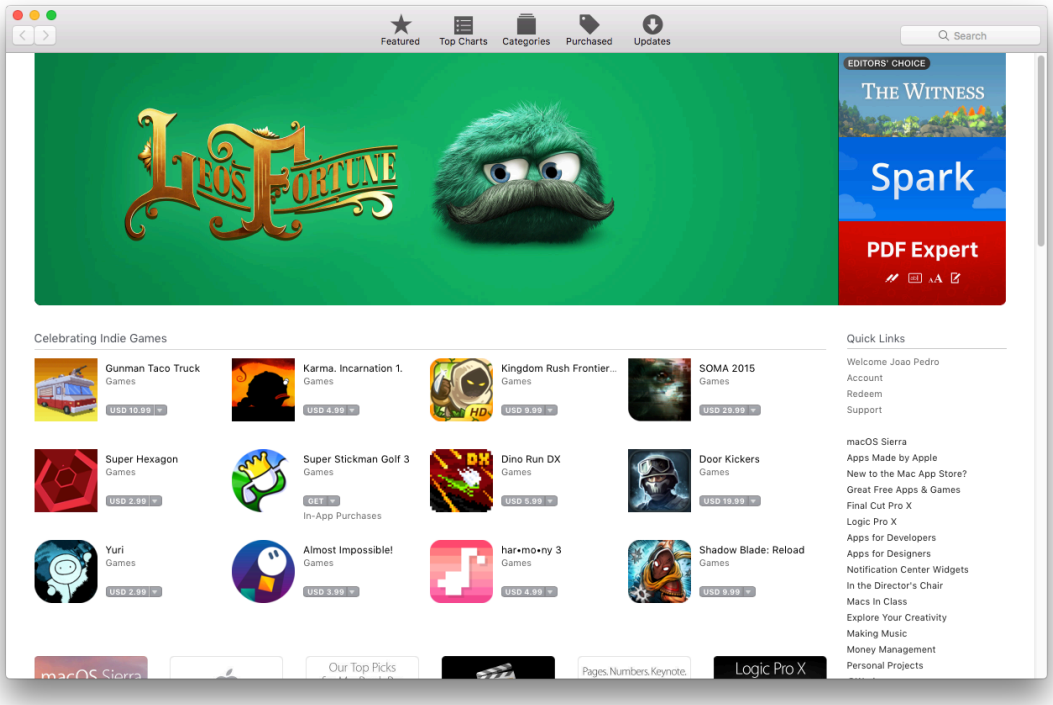
Instalação do Xcode

O Xcode é um software Apple que possibilitará o uso de Command Line Tools, necessárias para a instalação do AdmsXml, que possibilita a simulação de componentes descritos em Verilog-A.

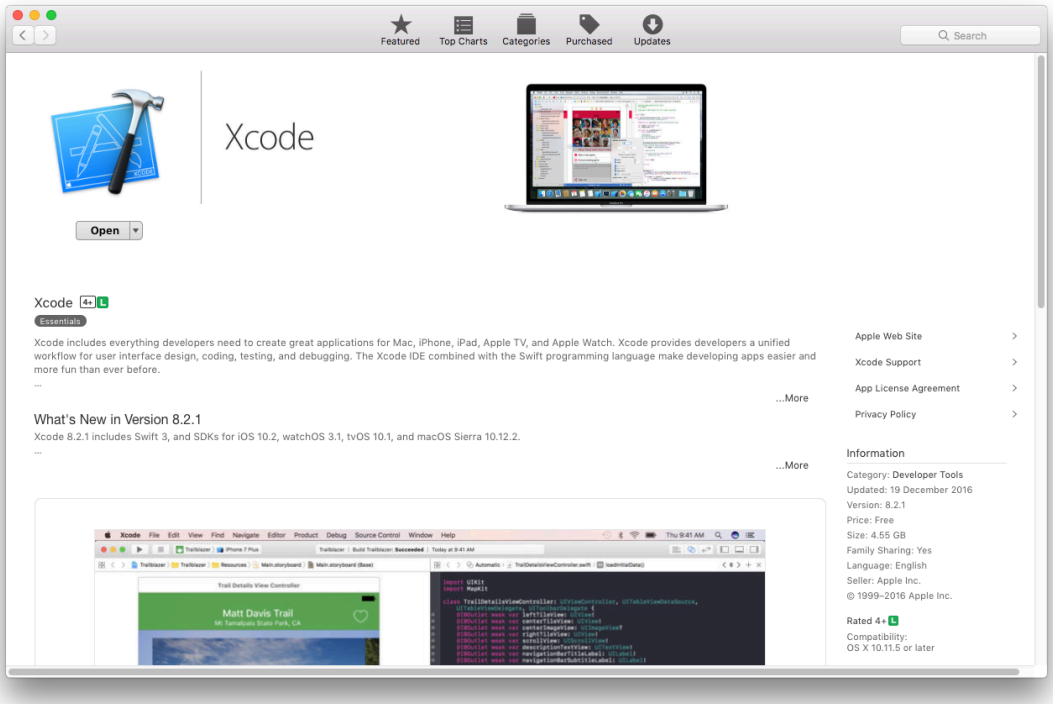
1. Vá para a pasta Applications do seu sistema e execute o aplicativo App Store.



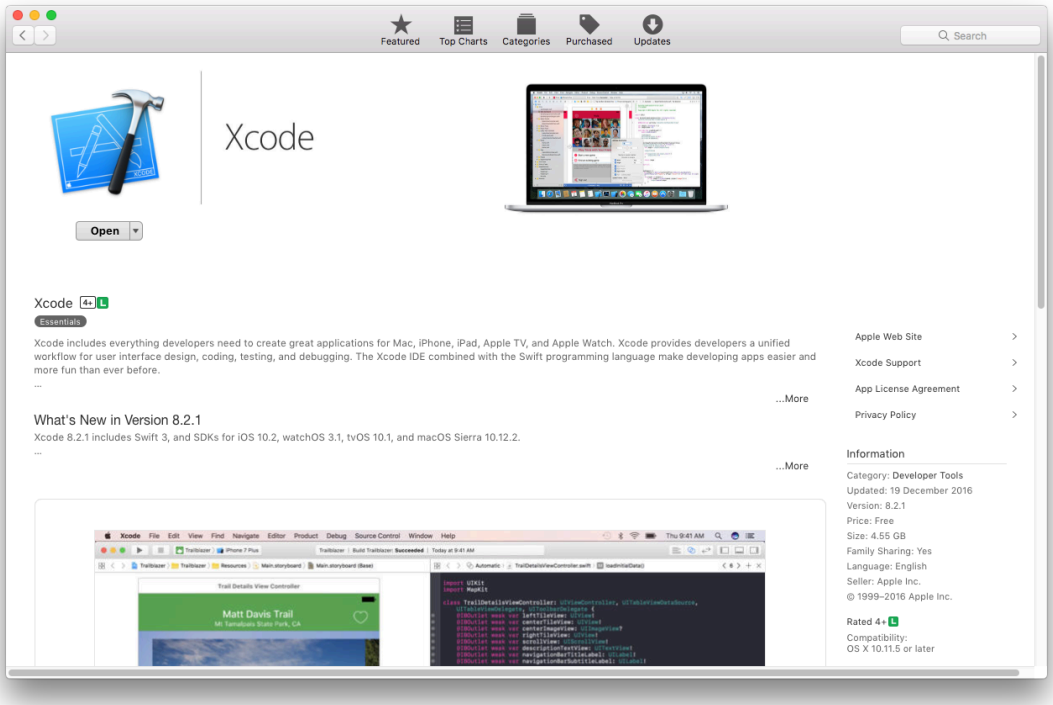
2. Na janela inicial da App Store, clique no campo Search, no canto superior direito, e digite "Xcode".



3. Clique no botão Get abaixo do ícone do Xcode para iniciar a instalação do Xcode.



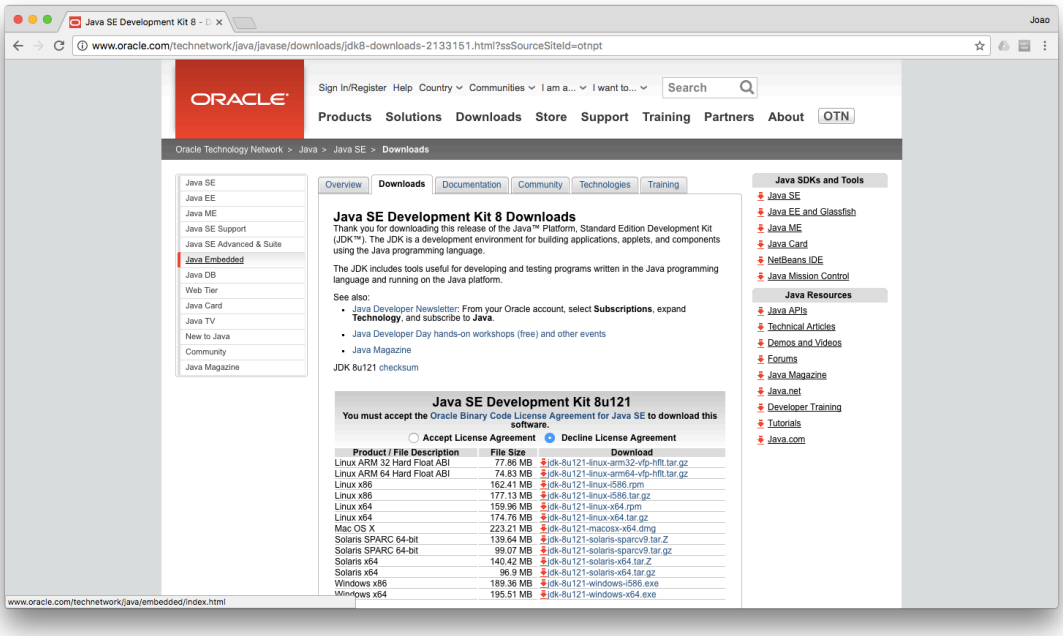
4. Concluída a instalação, execute o Xcode. Ele irá guiá-lo pelos resto do processo.



Instalação do Java SE Development Kit

O Java SE Development Kit é um software Oracle, também necessário para a instalação do AdmsXml.

1. Abra algum navegador de internet (Google Chrome, Apple Safari, Mozilla Firefox) e acesse o link: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html?ssSourceSiteId=otnpt. Clique em "Accept License Agreement", e escolha o arquivo do instalador correspondente ao seu sistema operacional para inciar seu download.

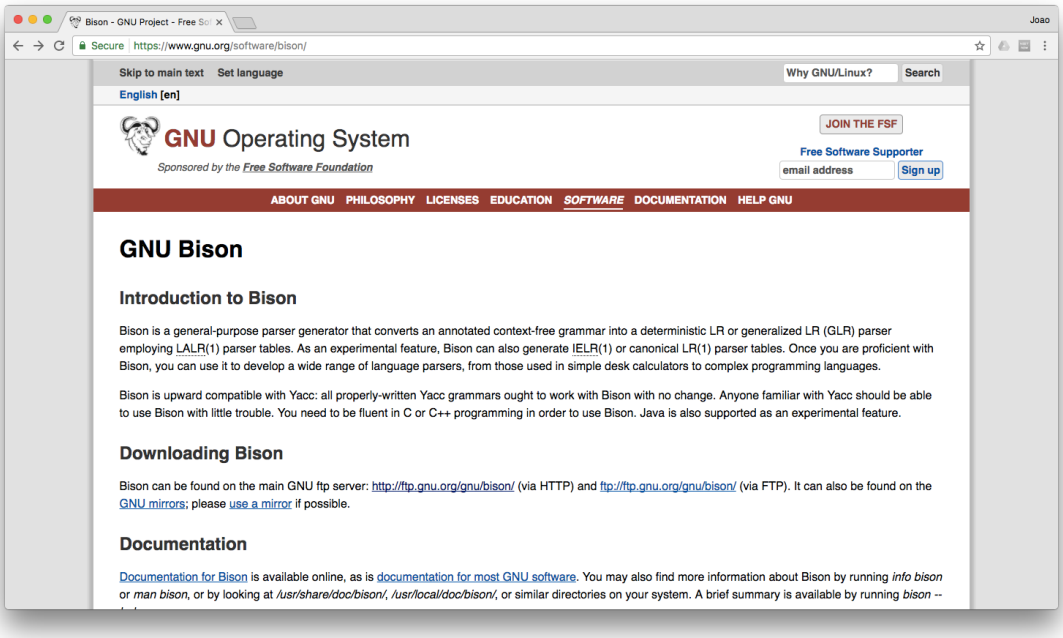


2. Execute o instalador. Ele irá guiá-lo pelo resto do processo.

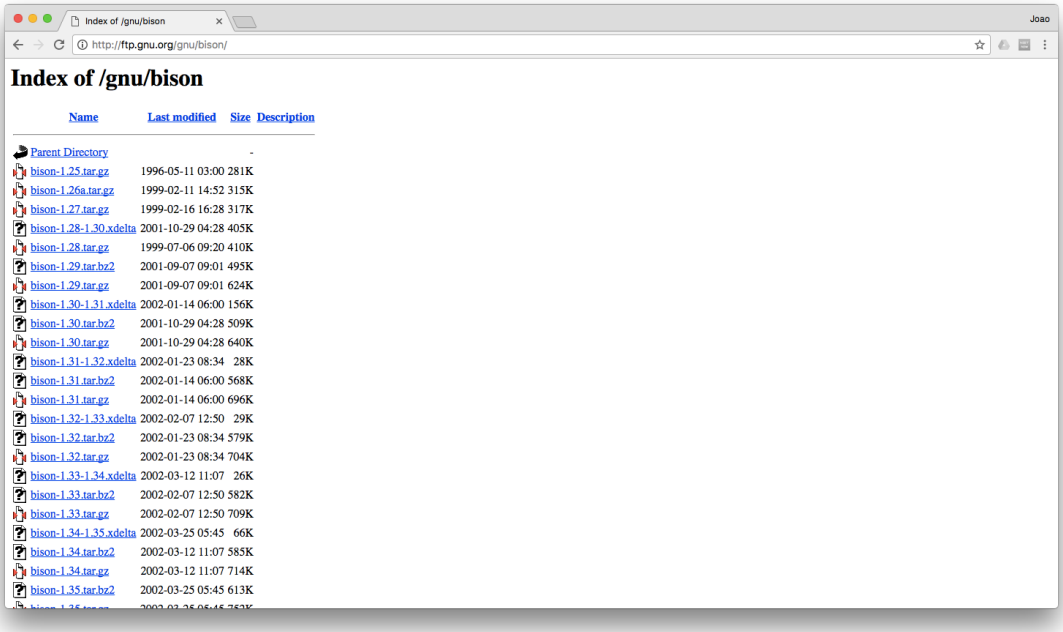
Instalação do Bison

O Bison é um software GNU necessário para a instalação do AdmsXml. O Mac OS já vem com uma versão obsoleta do Bison (2.3), portanto é necessária a instalação de uma versão atualizada.

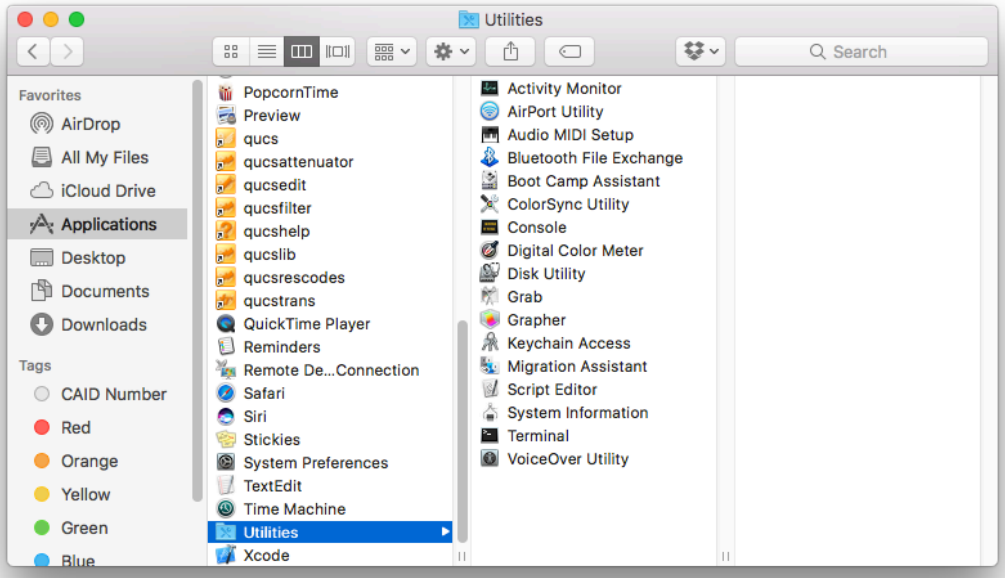
1. Abra algum navegador de internet (Google Chrome, Apple Safari, Mozilla Firefox) e acesse o link: https://www.gnu.org/software/bison/. Na seção "Downloading Bison", clique em algum dos links para prosseguir ao download do Bison.



2. Clique no link correspondente ao arquivo "bison-3.0.tar.gz" para iniciar o download. Salve o arquivo na pasta Downloads.



3. Vá para a pasta Applications/Utilities e execute o aplicativo Terminal.



4. No aplicativo Terminal, digite os seguintes comandos:

cd Downloads

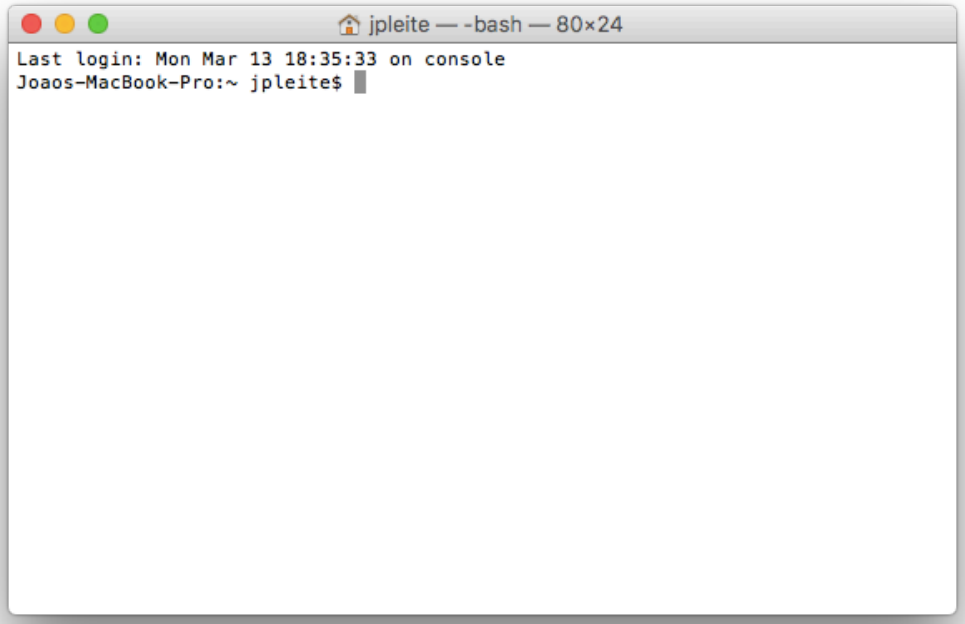
tar xvfz bison-3.0.tar.gz

cd bison-3.0

./configure

make install

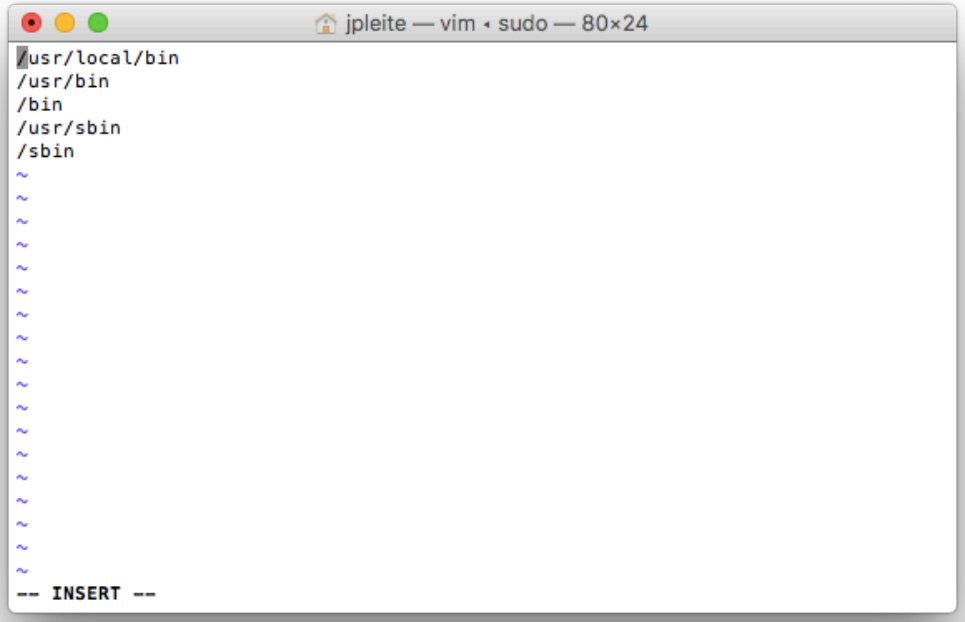
sudo vim /etc/paths



5. Insira sua senha de usuário para poder utilizar o vim para editar o arquivo paths. Esse arquivo indica a ordem das pastas na qual seu computador procura pelos programas para executá-los. Tecle "i" para entrar no modo Insert do vim. No modo insert, o vim funciona como um editor de textos padrão. Edite o arquivo para que ele fique igual ao da figura. Ao terminar, tecle "esc" e insira o comando:

   :wq

   Esse comando salva o arquivo e encerra o vim.

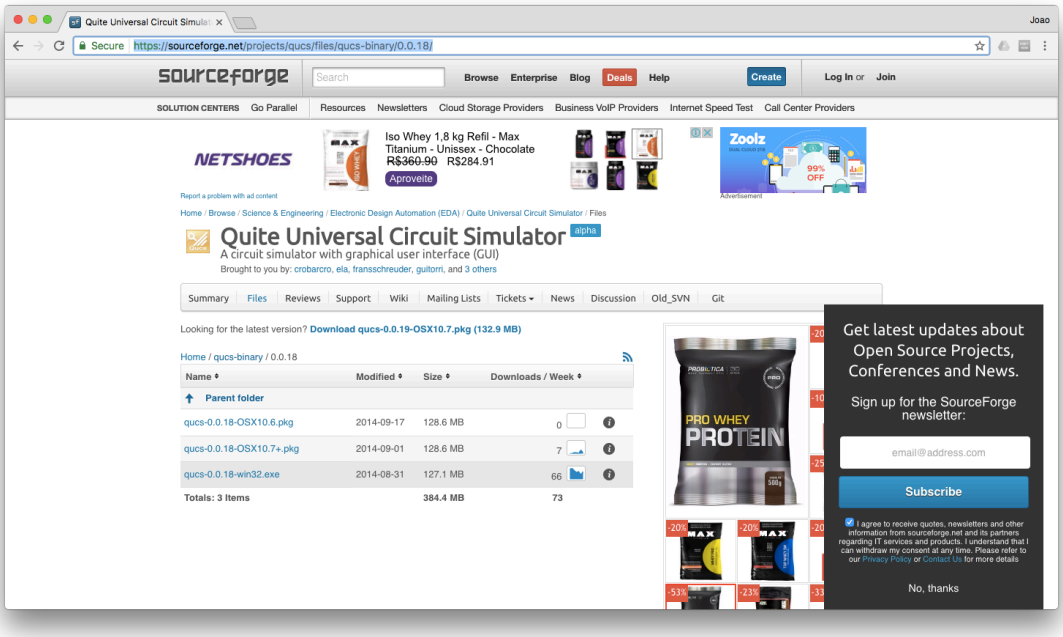


6. Feche o aplicativo Terminal.

Instalação do AdmsXml

O software AdmsXml é um software necessário para que o Qucs simule componentes de circuito descritos em Verilog-a.

1. Abra algum navegador de internet (Google Chrome, Apple Safari, Mozilla Firefox) e acesse o link: https://sourceforge.net/projects/mot-adms/. Clique no botão "Download" para iniciar o download do arquivo. Salve o arquivo na pasta Downloads.



2. Vá para a pasta Applications/Utilities e execute o aplicativo Terminal.

3. No aplicativo Terminal, digite os seguintes comandos:

```
cd Downloads

tar xvfz adms-2.3.5.tar.gz

cd adms-2.3.5

./configure

make install
```

4. Feche o aplicativo Terminal.

Automated Simulator

Simulador Automatizado de Circuitos

Tutorial – Utilizando o Automated Simulator

versão 1.0

Universidade de Brasília

João Pedro Leite Nunes
Universidade de Brasília

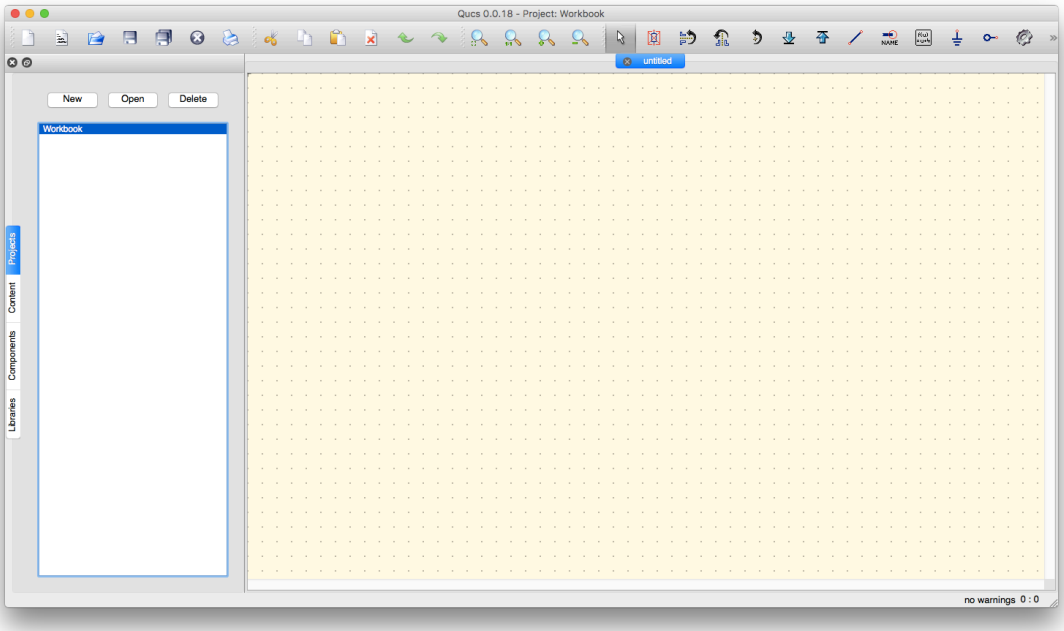
Introdução

A utilização do Automated Simulator para automatizar a fase de geração de benchmarks de circuitos eletrônicos visa simplificar e acelerar essa fase. No entanto, ela pode se tornar confusa caso o usuário não tenha familiaridade com o software. Este Tutorial visa guiar o usuário através de um exercício básico de obtenção de curvas de transferência de um transistor CNTFET, modelo CCAM, para diferentes valores de seu parâmetro de percentual de tubos metálicos (p_{mt}) e, com isso, oferecer uma primeira abordagem prática à utilização do Automated Simulator.

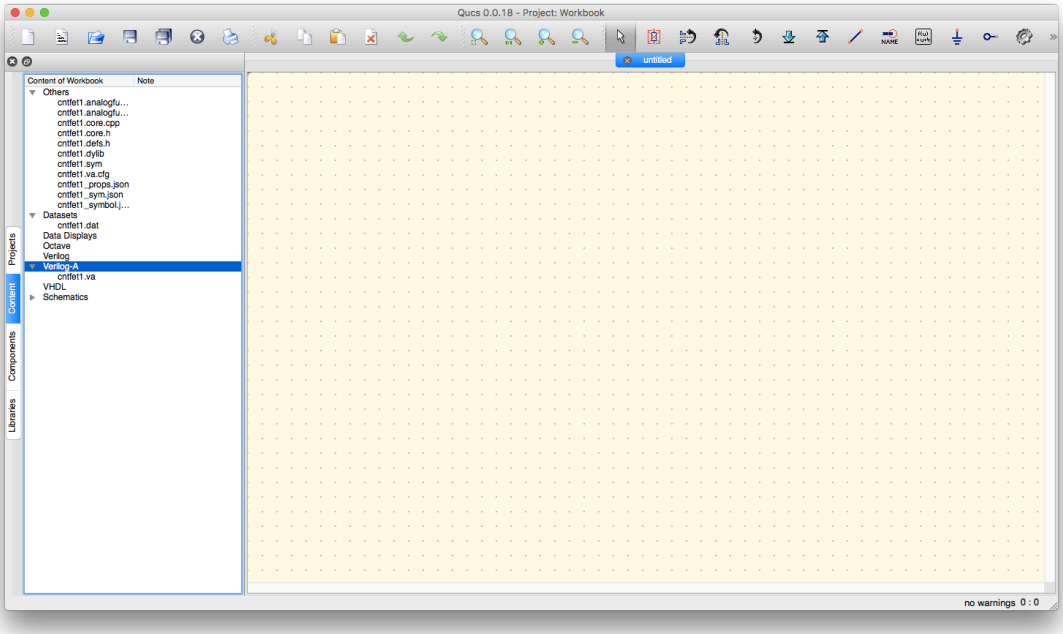
Geração da Netlist do Circuito

O primeiro passo para utilizar o Automated Simulator é gerar, em ambiente QUCS, a Netlist do circuito eletrônico que se deseja simular.

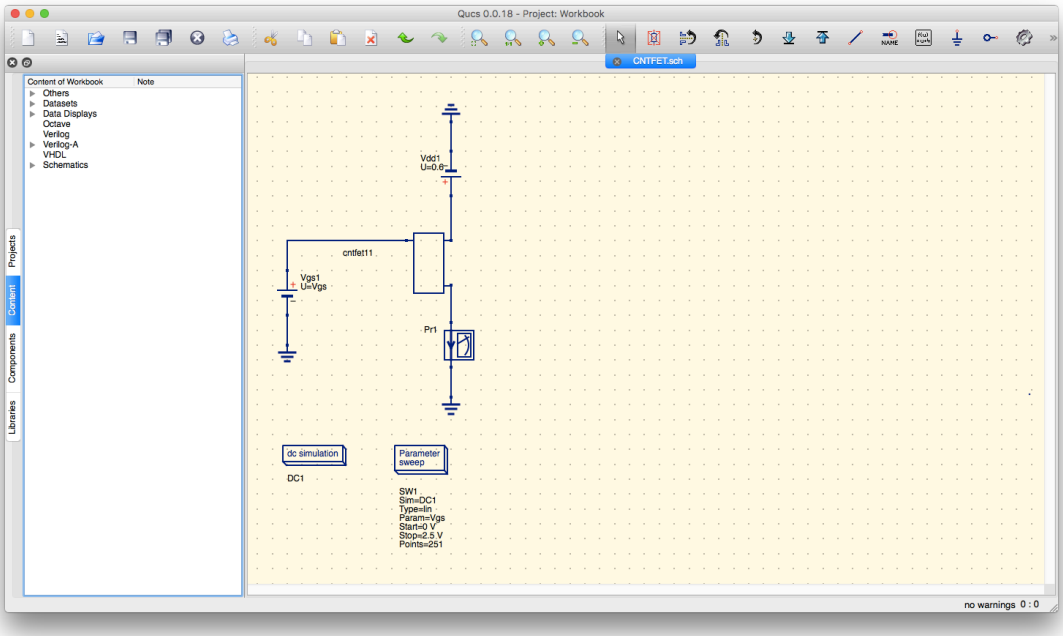
1. Siga o Tutorial – Getting Started with Qucs do Prof. Stefan Michael Blawid e do Dr. Muthupandian Cheralathan até a seção "Setting Up Schematics", primeira subseção, para configurar o ambiente QUCS. Ignore o resto da seção "Setting Up Schematics".



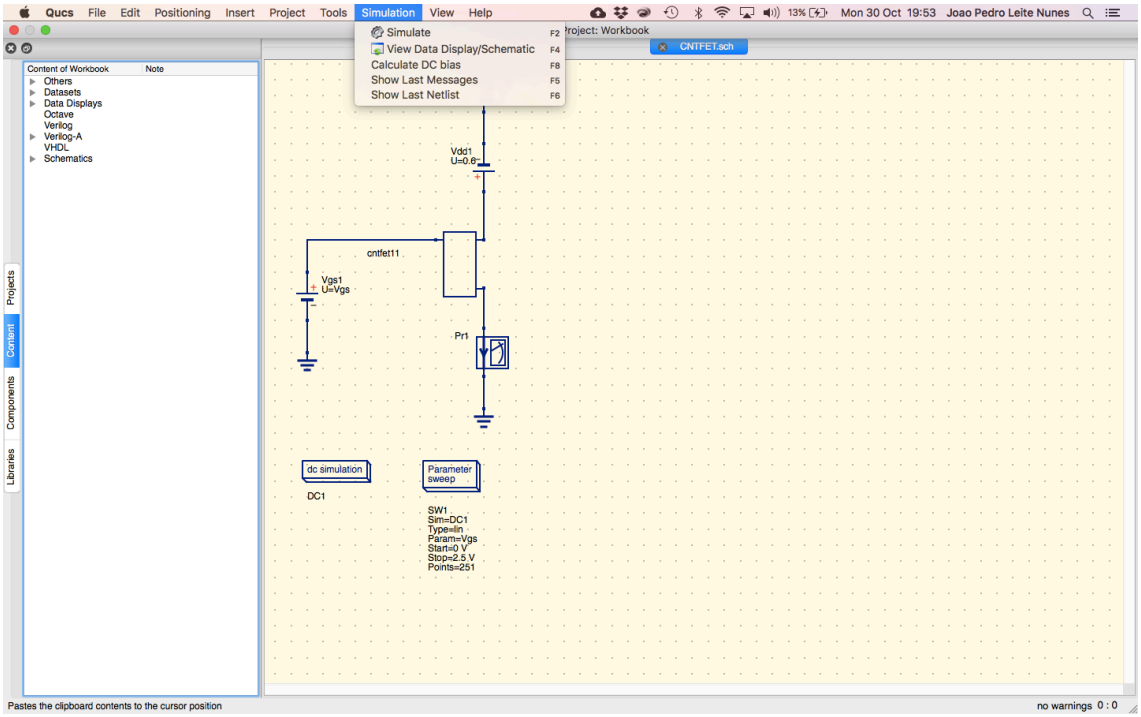
2. Siga o Tutorial – Getting Started with Qucs do Prof. Stefan Michael Blawid e do Dr. Muthupandian Cheralathan a partir da seção "Building Verilog-A Module" para configurar o componente do transistor CNTFET, modelo CCAM que será utilizado.



3. Construa o esquemático do circuito que se deseja simular. Neste tutorial, será utilizado o esquemático ilustrado abaixo.



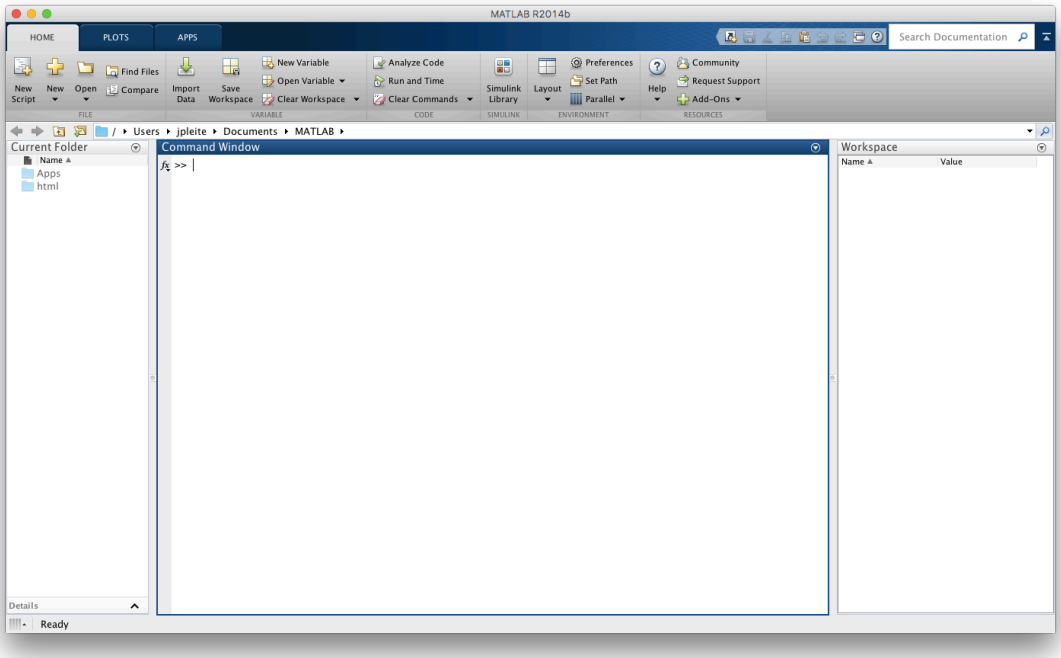
4. Simule uma vez o esquemático do circuito para gerar a Netlist correspondente e encerre o QUCS.



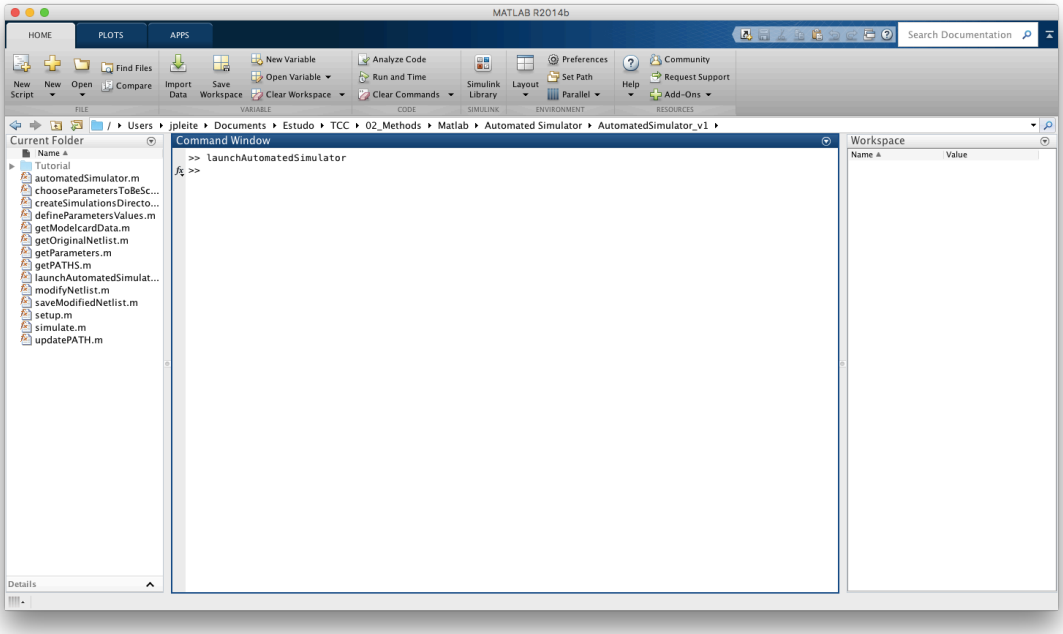
Utilização do Automated Simulator

Após a fase de geração da Netlist em ambiente QUCS, passamos à fase de utilização do Automated Simulator em ambiente Matlab.

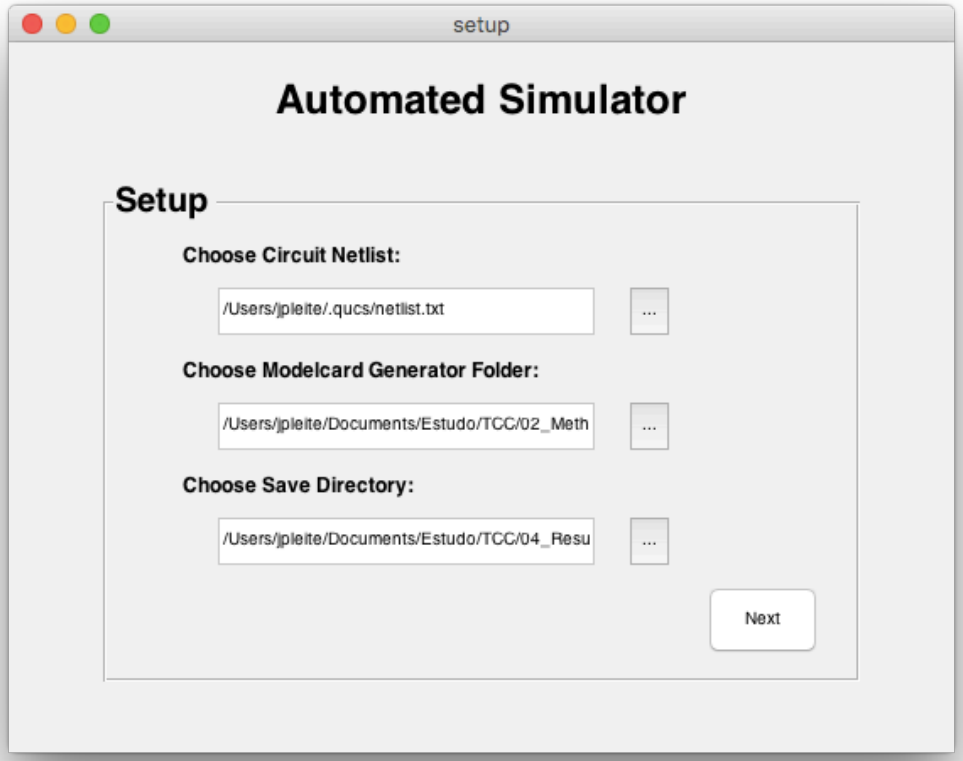
1. Execute o aplicativo Matlab.



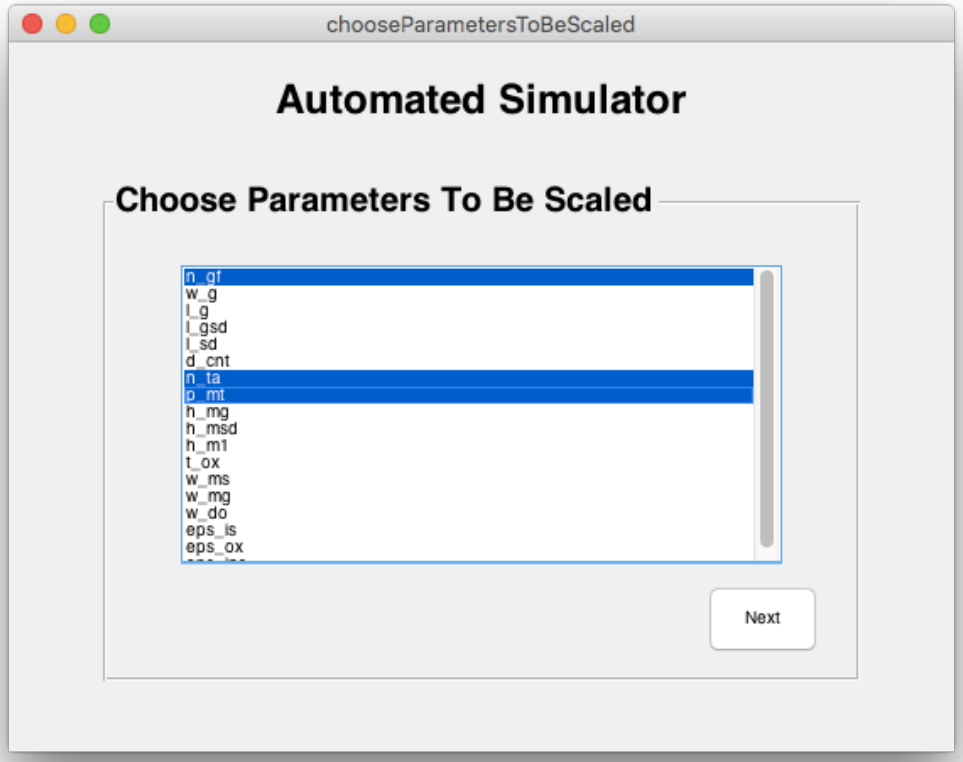
2. Navegue para a pasta do Automated Simulator e execute o comando "launchAutomatedSimulator".



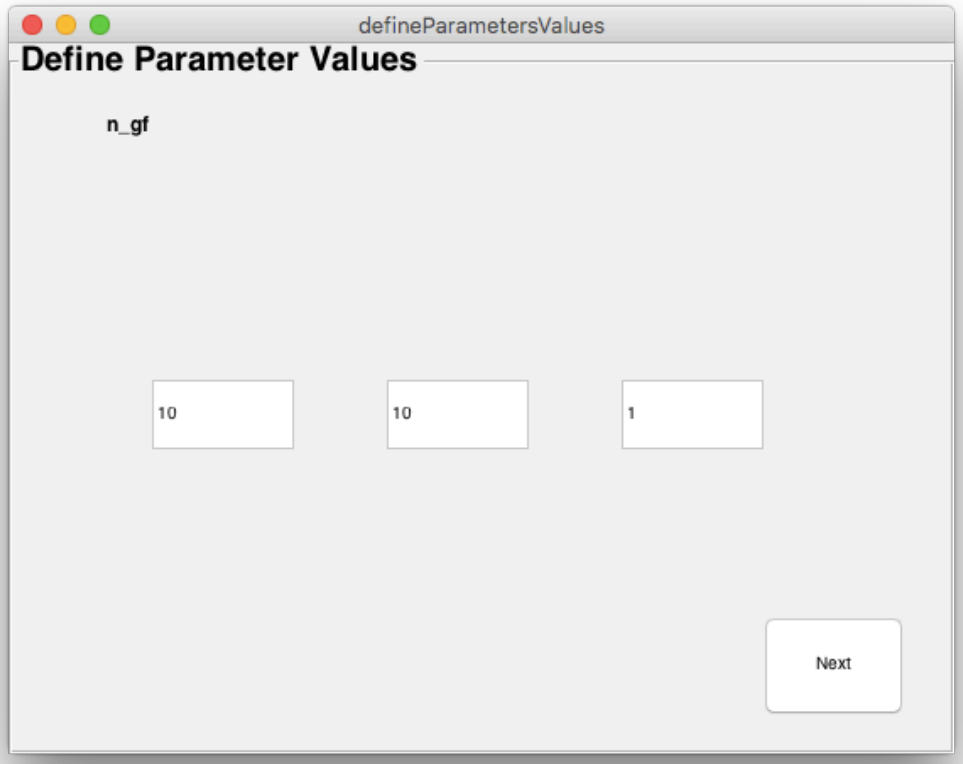
3. Na janela "Setup" do Automated Simulator, escolha a Netlist do circuito que se deseja simular, a pasta em que se encontra o Gerador de Carta de Modelo e um diretório para salvar os dados da simulação e em seguida clique em "Next".

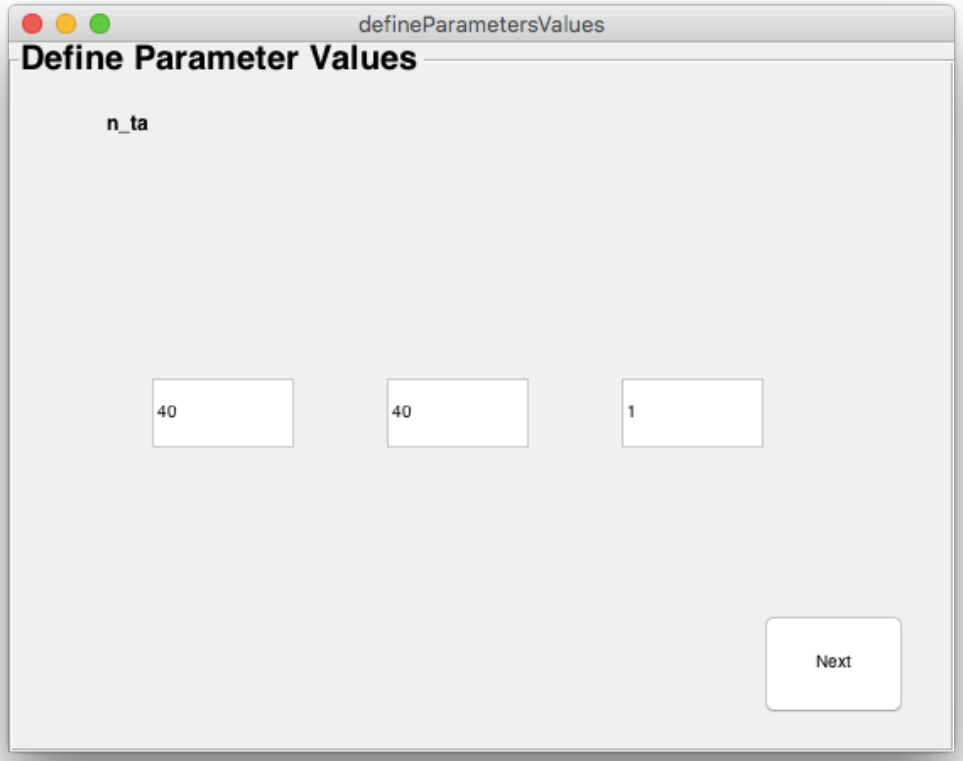


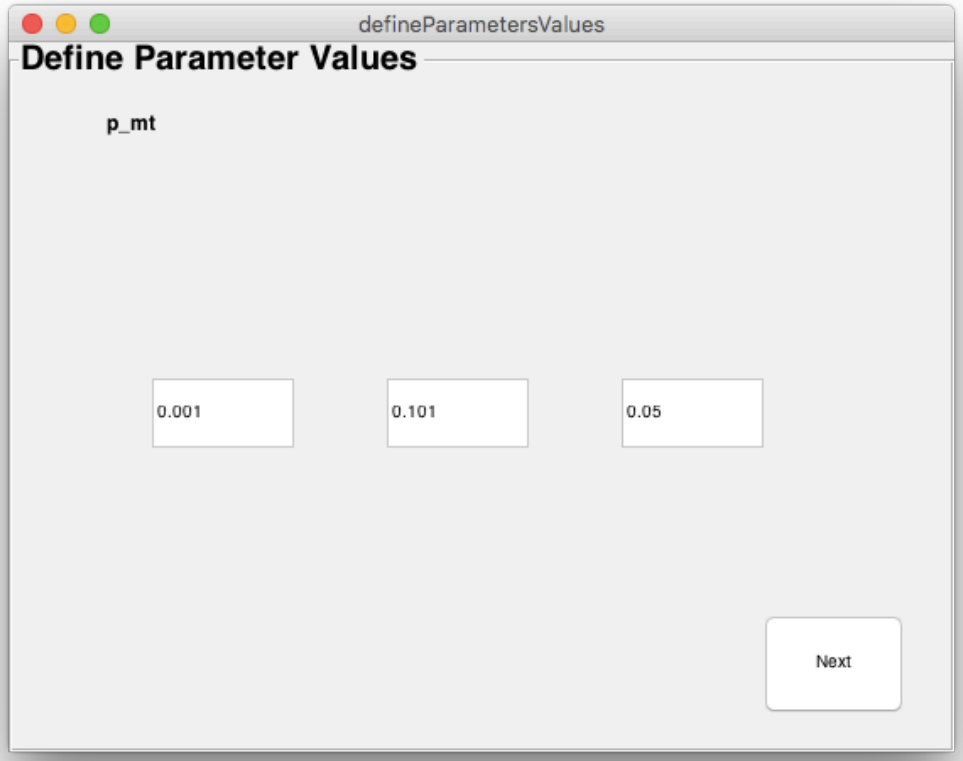
4. Na janela "Choose Parameters To Be Scaled" do Automated Simulator, escolha os parâmetros dos transistores que se deseja variar e clique em "Next". No exemplo deste Tutorial, será utilizado um transistor com os parâmetros de referência definidos em [1] proporcionalizado para um número de dedos da porta (n_{gf}) igual a 10 e uma densidade de tubos (n_{ta}) igual a 40. Seu percentual de tubos metálicos (p_{mt}) será variado de aproximadamente 0% para aproximadamente 10% com um passo de 5%.



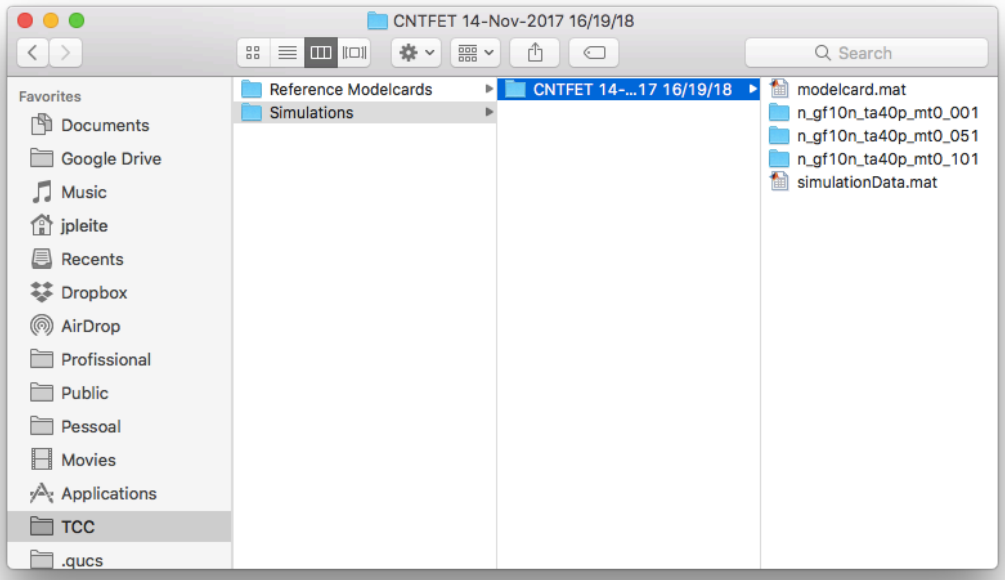
5. Nas janelas "Define Parameters Values" do Automated Simulator referentes aos parâmetros selecionados, insira os valores correspondentes e clique em "Next".




defineParametersValues
Define Parameter Values
n_ta
40
40
1
Next


defineParametersValues
Define Parameter Values
p_mt
0.001
0.101
0.05
Next

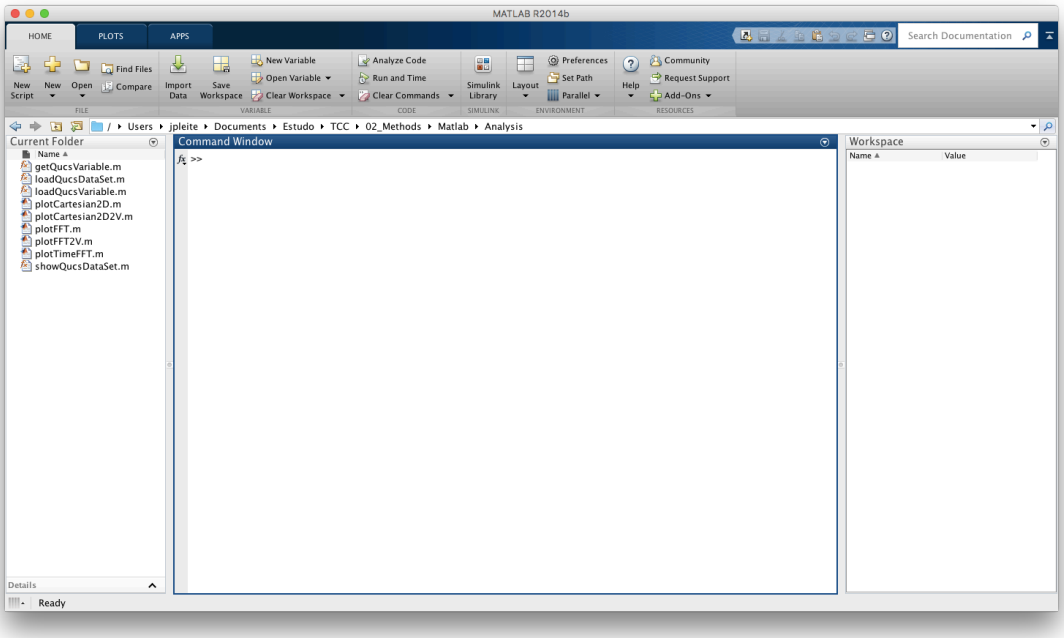
6. Na janela "Simulate" do Automated Simulator, clique em "Simulate" e feche a janela. Os resultados serão disponibilizados no diretório escolhido no passo 3 desta seção.


CNTFET 14-Nov-2017 16/19/18
Search
Favorites
Documents
Google Drive
Music
jpleite
Recents
Dropbox
AirDrop
Profissional
Public
Pessoal
Movies
Applications
TCC
.qucs
Reference Modelcards
Simulations
CNTFET 14-...17 16/19/18
modelcard.mat
n_gf10n_ta40p_mt0_001
n_gf10n_ta40p_mt0_051
n_gf10n_ta40p_mt0_101
simulationData.mat

Análise dos Resultados

De posse dos resultados da simulação, pode-se partir para a etapa de análise dos resultados, também em ambiente Matlab. Para tanto, é necessário ter concluído o Tutorial - Using QUCS in Textmode, disponível em http://qucs.sourceforge.net/docs/tutorial/textmode.pdf e possuir os arquivos utilizados nele.

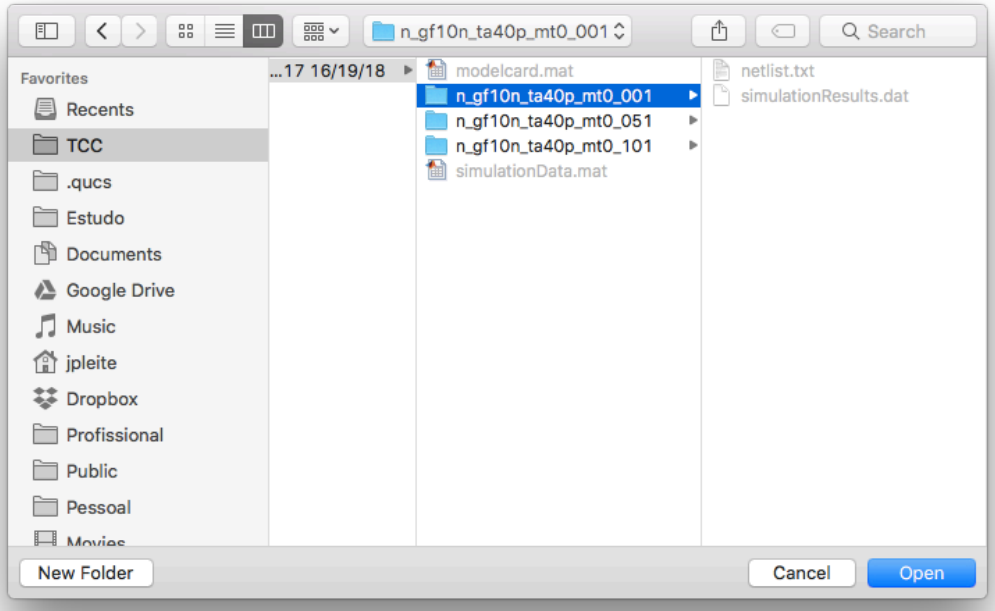
1. Navegue para a pasta onde foram salvos os arquivos utilizados no Tutorial - Using QUCS in Textmode.

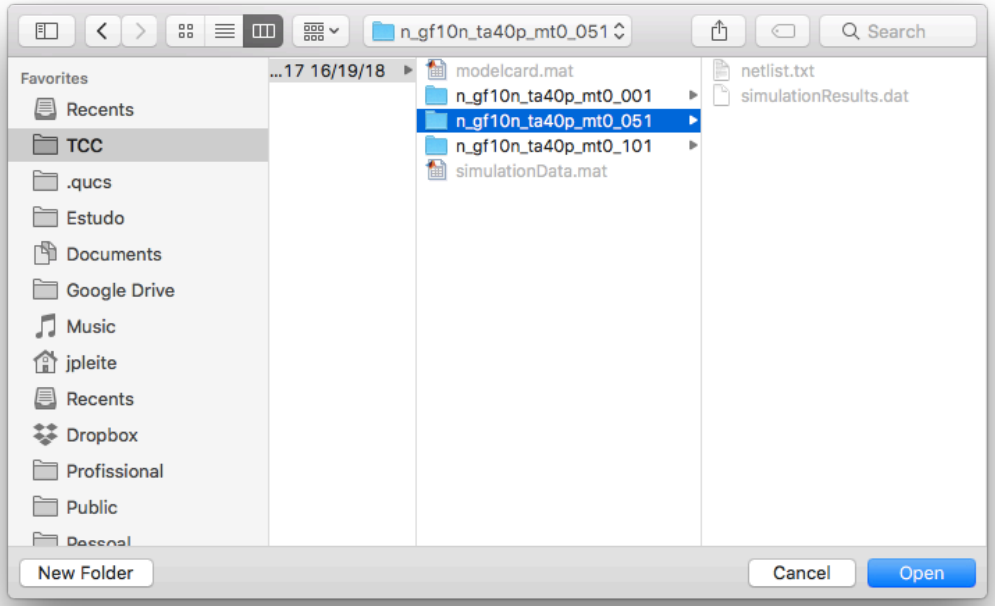


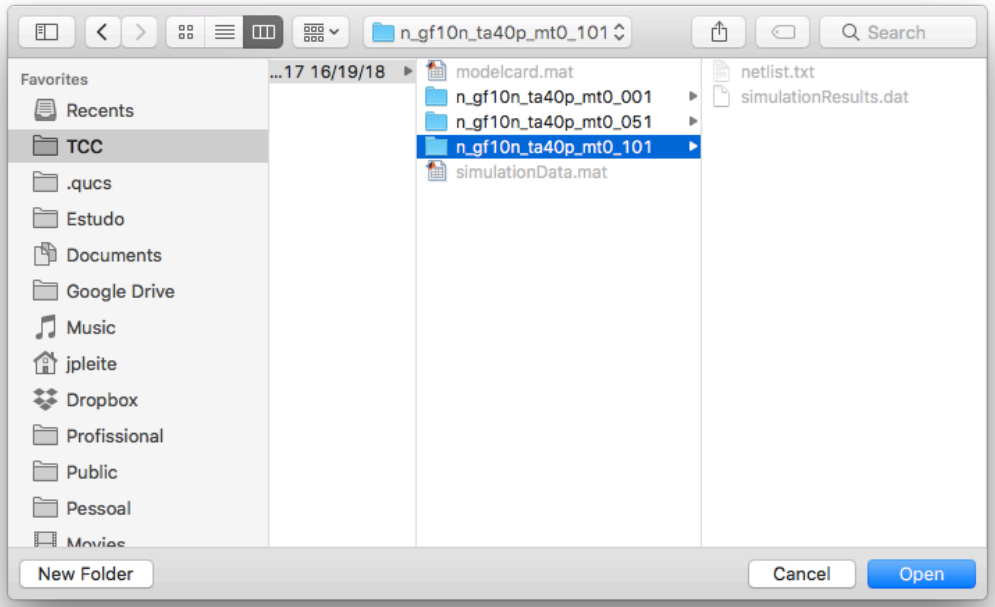
2. Execute os comandos:

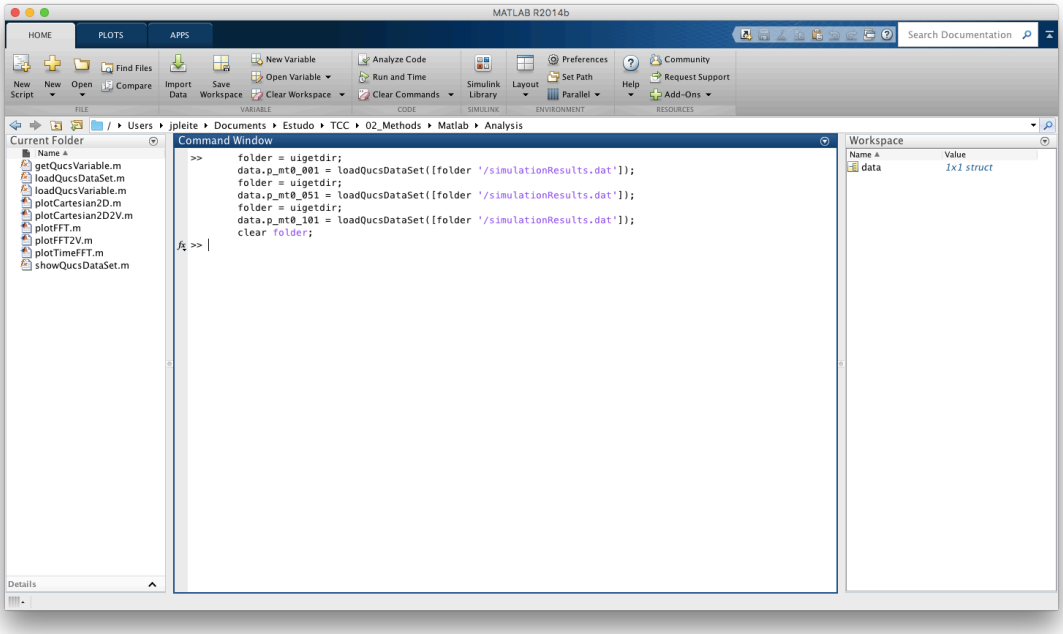
```
folder = uigetdir;
data.p_mt0_001 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_051 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_101 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;
```

Eles permitem criar a variável "data", que contém os dados de simulação para os diferentes valores do percentual de tubos metálicos do transistor, a partir da escolha das pastas onde os respectivos resultados estão salvos.

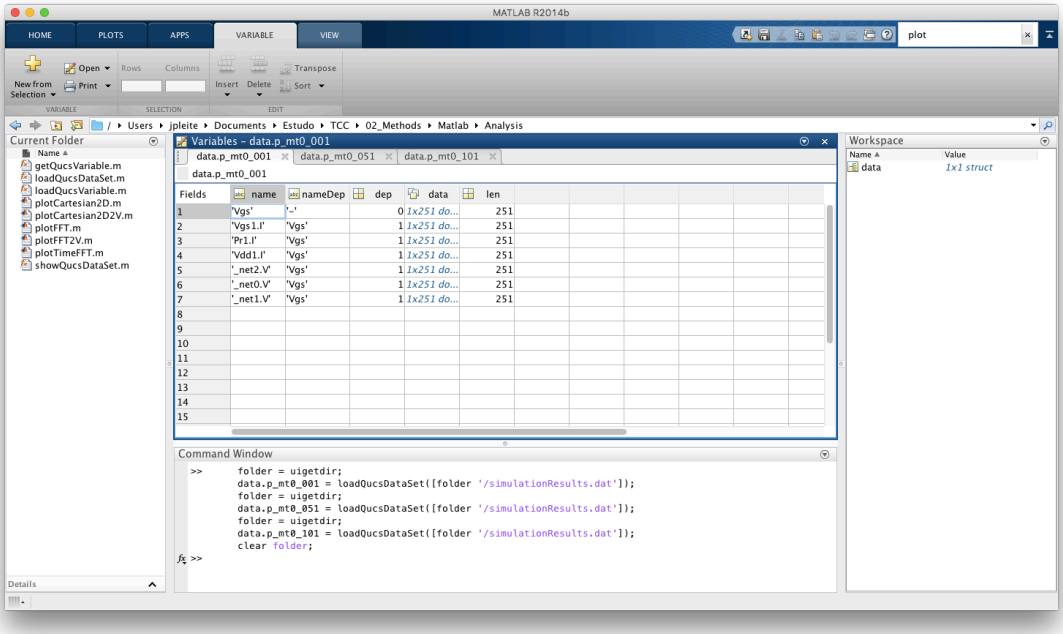


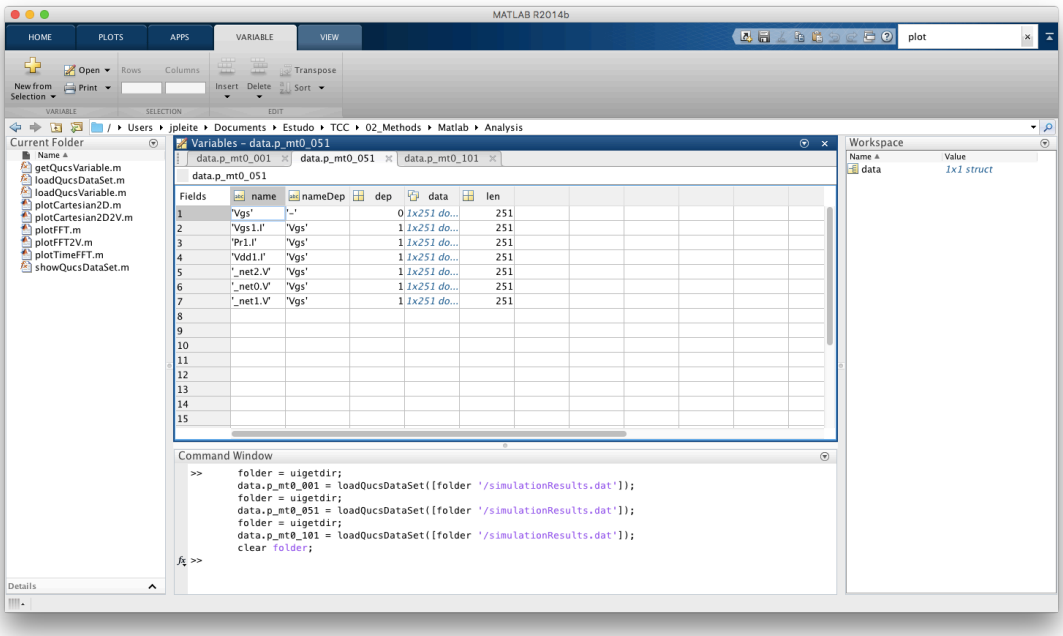

n_gf10n_ta40p_mt0_051
Search
Favorites
Recents
TCC
.qucs
Estudo
Documents
Google Drive
Music
jpleite
Recents
Dropbox
Profissional
Public
Pessoal
...17 16/19/18
modelcard.mat
n_gf10n_ta40p_mt0_001
n_gf10n_ta40p_mt0_051
n_gf10n_ta40p_mt0_101
simulationData.mat
netlist.txt
simulationResults.dat
New Folder
Cancel
Open

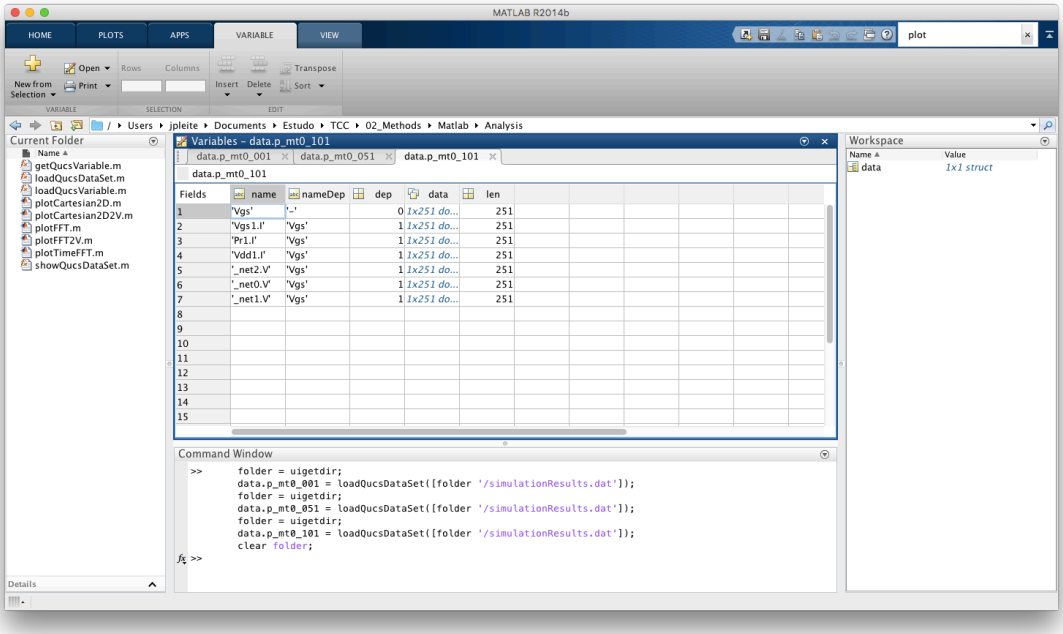

n_gf10n_ta40p_mt0_101
Search
Favorites
Recents
TCC
.qucs
Estudo
Documents
Google Drive
Music
jpleite
Dropbox
Profissional
Public
Pessoal
...17 16/19/18
modelcard.mat
n_gf10n_ta40p_mt0_001
n_gf10n_ta40p_mt0_051
n_gf10n_ta40p_mt0_101
simulationData.mat
netlist.txt
simulationResults.dat
New Folder
Cancel
Open



3. Abra todas as instâncias de "data" para visualizar melhor a organização dos dados de simulação. No exemplo deste Tutorial, deseja-se obter as curvas de transferência do transistor CNTFET para diferentes valores de seu percentual de tubos metálicos. Logo, deseja-se obter as 3 curvas de I_D x V_{GS} num mesmo gráfico. Pode-se observar que I_D é o campo número 3 e V_{GS} é o campo número 1 para cada uma das instâncias de "data".



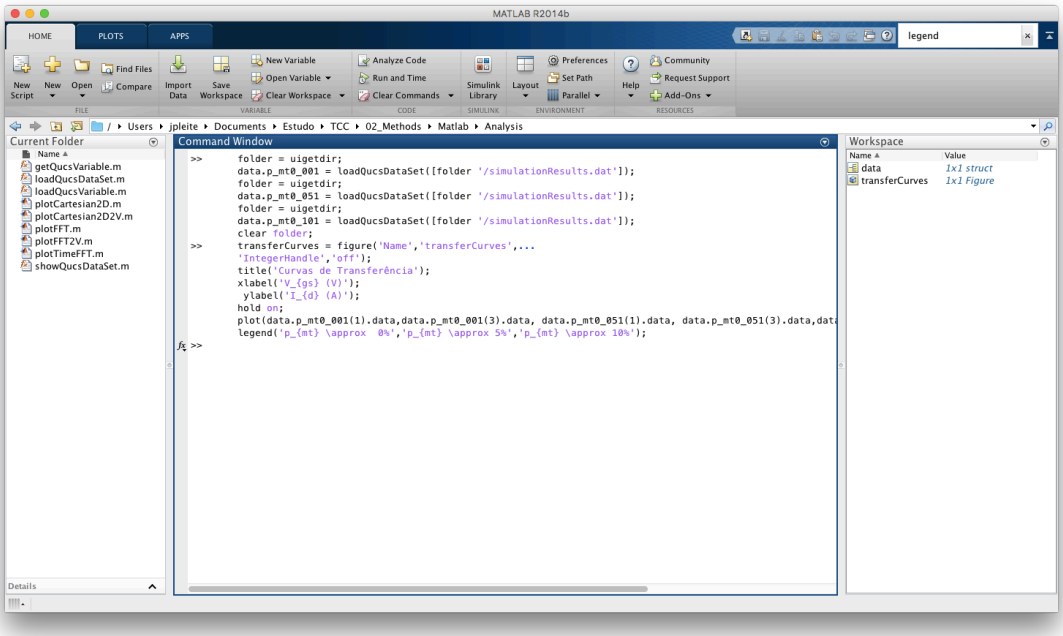

MATLAB R2014b
HOME
PLOTS
APPS
VARIABLE
VIEW
plot
Users ▸ jpleite ▸ Documents ▸ Estudo ▸ TCC ▸ 02_Methods ▸ Matlab ▸ Analysis
Current Folder
getQucsVariable.m
loadQucsDataSet.m
loadQucsVariable.m
plotCartesian2D.m
plotCartesian2D2V.m
plotFFT.m
plotFFT2V.m
plotTimeFFT.m
showQucsDataSet.m
Variables - data.p_mt0_051
data.p_mt0_001
data.p_mt0_051
data.p_mt0_101
Fields
name
nameDep
dep
data
len
'Vgs' '-' 0 1x251 do... 251
'Vgs1.I' 'Vgs' 1 1x251 do... 251
'Pr1.I' 'Vgs' 1 1x251 do... 251
'Vdd1.I' 'Vgs' 1 1x251 do... 251
'_net2.V' 'Vgs' 1 1x251 do... 251
'_net0.V' 'Vgs' 1 1x251 do... 251
'_net1.V' 'Vgs' 1 1x251 do... 251
Workspace
Name
Value
data
1x1 struct
Command Window
folder = uigetdir;
data.p_mt0_001 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_051 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_101 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;

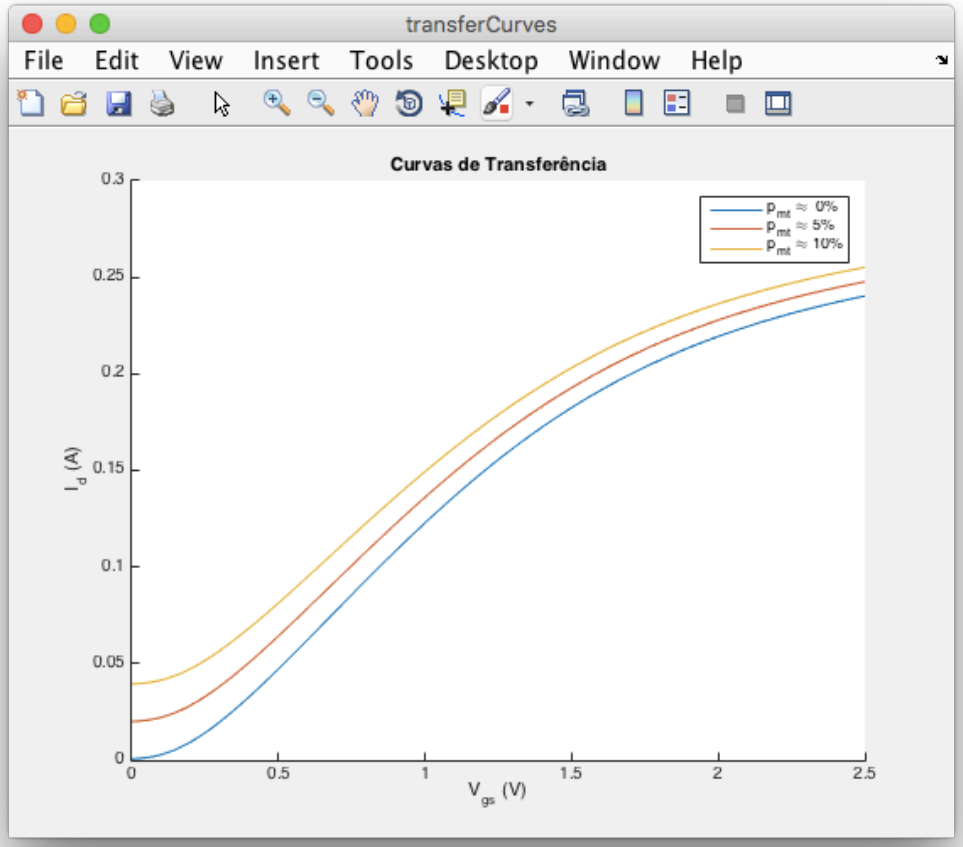

MATLAB R2014b
HOME
PLOTS
APPS
VARIABLE
VIEW
plot
Users ▸ jpleite ▸ Documents ▸ Estudo ▸ TCC ▸ 02_Methods ▸ Matlab ▸ Analysis
Current Folder
getQucsVariable.m
loadQucsDataSet.m
loadQucsVariable.m
plotCartesian2D.m
plotCartesian2D2V.m
plotFFT.m
plotFFT2V.m
plotTimeFFT.m
showQucsDataSet.m
Variables - data.p_mt0_101
data.p_mt0_001
data.p_mt0_051
data.p_mt0_101
Fields
name
nameDep
dep
data
len
'Vgs' '-' 0 1x251 do... 251
'Vgs1.I' 'Vgs' 1 1x251 do... 251
'Pr1.I' 'Vgs' 1 1x251 do... 251
'Vdd1.I' 'Vgs' 1 1x251 do... 251
'_net2.V' 'Vgs' 1 1x251 do... 251
'_net0.V' 'Vgs' 1 1x251 do... 251
'_net1.V' 'Vgs' 1 1x251 do... 251
Workspace
Name
Value
data
1x1 struct
Command Window
folder = uigetdir;
data.p_mt0_001 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_051 = loadQucsDataSet([folder '/simulationResults.dat']);
folder = uigetdir;
data.p_mt0_101 = loadQucsDataSet([folder '/simulationResults.dat']);
clear folder;

4. Feche as janelas das instâncias de "data" e execute os comandos:

```
transferCurves = figure('Name','transferCurves',...
'IntegerHandle','off');
title('Curvas de Transferência');
xlabel('V_{gs} (V)');
 ylabel('I_{d} (A)');
hold on;
plot(data.p_mt0_001(1).data,data.p_mt0_001(3).data, data.p_mt0_051(1).data, data.p_mt0_051(3).data,data.p_mt0_101(1).data, data.p_mt0_101(3).data);
legend('p_{mt} \approx  0%','p_{mt} \approx 5%','p_{mt} \approx 10%');
```

Eles geram o gráfico das curvas de transferência do transistor CNTFET para os valores aproximados de 0%, 5% e 10% do percentual de tubos metálicos.




transferCurves
File Edit View Insert Tools Desktop Window Help
Curvas de Transferência
$p_{mt} \approx 0\%$
$p_{mt} \approx 5\%$
$p_{mt} \approx 10\%$
I_d (A)
V_{gs} (V)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.5
1
1.5
2
2.5