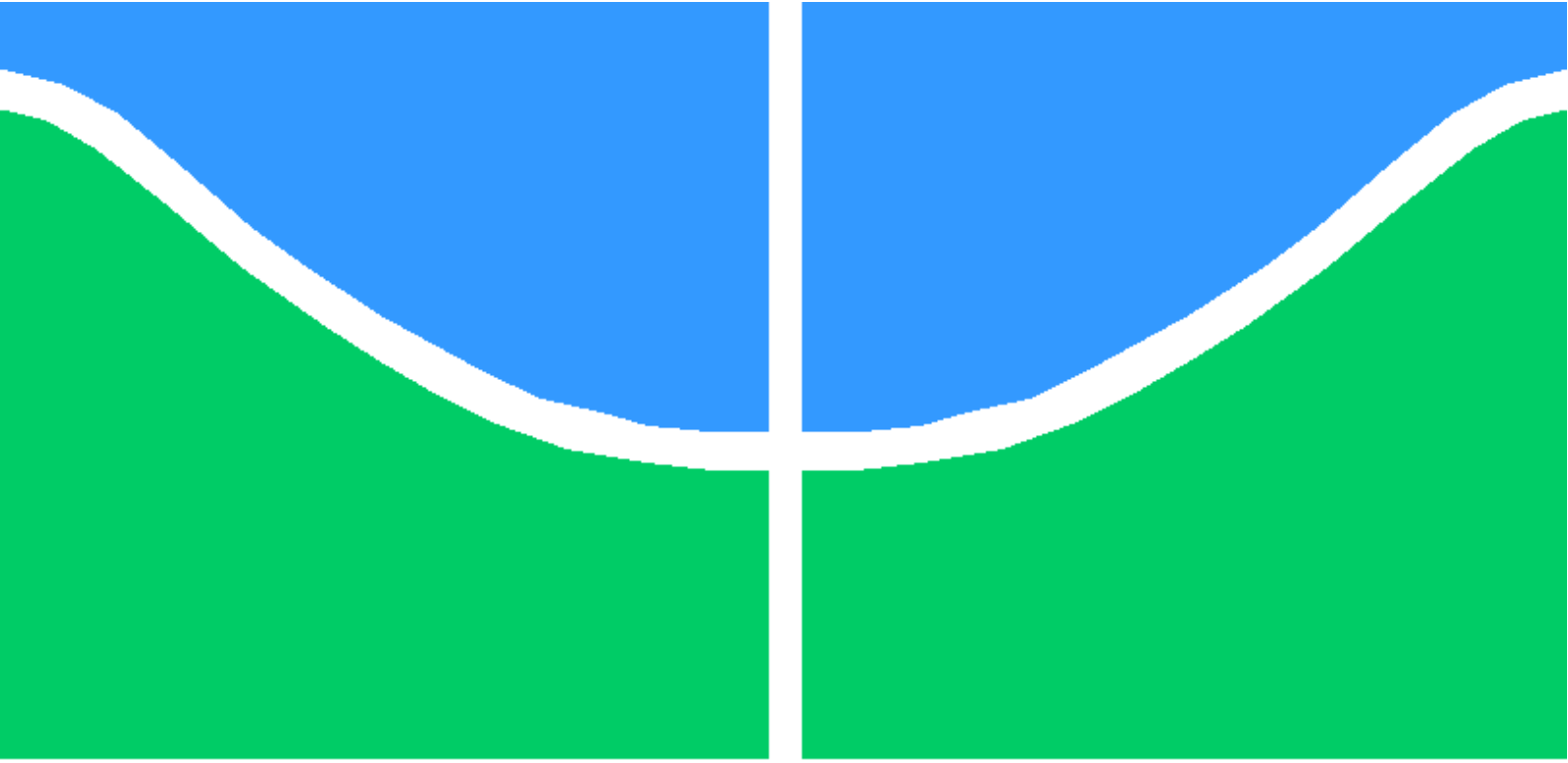

TRABALHO DE CONCLUSÃO DE CURSO

MODELAGEM DE TRANSISTORES DE FILMES FINOS ORGÂNICOS COM O MODELO DE FONTE VIRTUAL: UMA AVALIAÇÃO COMPARATIVA

Ricardo Moreno Taveira Coelho

Brasília, dezembro de 2015

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO

MODELAGEM DE TRANSISTORES DE FILMES FINOS ORGÂNICOS COM O MODELO DE FONTE VIRTUAL: UMA AVALIAÇÃO COMPARATIVA

Ricardo Moreno Taveira Coelho

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

Banca Examinadora

Prof. Stephan Michael Blawid, ENE/UnB _____
Orientador

Dr. Muthupandian Cheralathan, ENE/UnB _____
Co-orientador

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

Prof. Alexandre Ricardo Doares Romariz, ENE/UnB _____
Examinador interno

Dedicatória

Dedico este trabaho a minha família, que me conduziu por este caminho com muita paciência e determinação.

Ricardo Moreno Taveira Coelho

Agradecimentos

Gostaria de agradecer, primeiramente, ao professor Stefan Blawid, que me orientou e me ensinou durante este processo com muita paciência e compreensão. Agradeço também ao orientador Muthu por ter se dedicado tanto em ajudar, mesmo entrando no meio do processo.

Ricardo Moreno Taveira Coelho

RESUMO

Este projeto avalia a aplicabilidade do modelo de fonte virtual mvs desenvolvido pelo MIT para a modelagem de transistores orgânicos de filmes finos (OTFT's) para simulações. Para tanto, foi implementado o modelo usando a linguagem Verilog-A no simulador de circuito Qucs (Quite Universal Circuit Simulator). Os parâmetros desse modelo têm uma clara interpretação física, como a velocidade de injeção de carga e a mobilidade efetiva, e o número de parâmetros é limitado. Isso torna o mvs altamente adaptável, um recurso promissor para modelar transistores fabricados em plataformas com constantes mudanças. Para avaliar a performance do modelo de fonte virtual, as características I versus V de um transistor de filmes finos serão reproduzidas e o resultado obtido será comparado a outros modelos, como UMEM e MOSFET level 1. Para todas as simulações, o Qucs foi usado.

ABSTRACT

The present project evaluates the applicability of the Virtual Source Model MVS developed by MIT for the modeling of organic thin film transistors (OTFTs) for circuit simulation purposes. To this extent, a Verilog-A version of the model was implemented in a general circuit simulator (Quite Universal Circuit Simulator - Qucs). Model parameters in MVS have a clear physical interpretation, like injection velocity and effective mobility, and their number is limited. This makes the MVS highly adaptive, a promising feature for a model intended to be used to model transistors fabricated on emergent and rapidly changing technology platforms. To evaluate the performance of MVS, IV characteristics of OTFTs are fitted and the results are compared to fits obtained, employing different compact models, like UMEM and MOSFET level 1. Qucs was used for all simulations.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

1.1 Contextualização

A eletrônica é o ramo da engenharia que trata do processamento de sinais eletrônicos. Estes são formados sempre pelo movimento de cargas em um meio condutor, a corrente elétrica. A capacidade de controlar a corrente elétrica para obter resultados esperados é exatamente o conceito de eletrônica.

Primeiro, a invenção do diodo foi um marco na eletrônica. Este foi aperfeiçoado em 1906 por Lee de Forest quando foi introduzido um terceiro elemento no interior do diodo que permitia o controle da intensidade da corrente elétrica. Em 1947, um novo componente foi inventado, chamado de *transfer resistor*, posteriormente abreviado pra transistor. O objetivo deste novo componente era exatamente controlar a corrente que circula nesses transistores de forma mais eficiente e com uma economia de energia. Não tardou a se popularizar e progressivamente substituir as já antiquadas válvulas eletrônicas.

Para conseguir modelar os transistores modernos será usado um modelo compacto mvs (modelo de fonte virtual) desenvolvido pelo MIT (*Massachusetts Institute of Technology*) que prevê as características de saída desses dispositivos.

1.2 Definição do problema

Um transistor pode ser modelado de várias formas diferentes. Este trabalho apresentará o modelo de fonte virtual e apresentará uma comparação deste com dois outros modelos: UMEM (*unified model and parameter extraction method*) e MOSFET (*metal oxide semiconducor field effect transistor*).

1.3 Objetivos do projeto

O objetivo deste trabalho é apresentar o modelo de fonte virtual (mvs) e sua capacidade de reproduzir as curvas características de saída de um transistor de filmes finos e utilizar como interface para simulação o Qucs. Será verificada a compatibilidade dessas duas ferramentas (Qucs + mvs) para se aplicar em uma tecnologia ainda não publicada e a possibilidade do uso dessa ferramenta para futuros trabalhos. Será realizada ainda uma comparação entre o modelo citado com dois outros modelos diferentes, UMEM e MOSFET, com a intenção de verificar o seu desempenho.

1.4 Apresentação do manuscrito

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 e análise são discutidos no capítulo 4, seguido das conclusões no capítulo 5. Os anexos contêm material complementar.

Capítulo 2

Revisão Bibliográfica

2.1 Modelo de fonte virtual

Um transistor é um dispositivo de quatro terminais em que a corrente que passa entre dois terminais (I_{Ds}) é controlada pela tensão que existe entre outros dois terminais (V_{Gs}). Observando a figura 2.1, que mostra o modelo de um transistor do tipo MOSFET (metal oxide semicondutor field effect transistor), observa-se os terminais source (fonte), gate (porta), drain (dreno) e bulk (substrato) que não está mostrado na imagem abaixo pois, em geral, é aterrado.

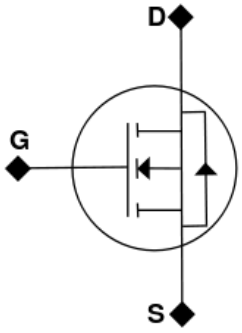

Figura 2.1: Modelo de um transistor MOSFET

Ao se aplicar uma tensão entre a porta e a fonte (V_{Gs}), é gerada uma corrente que passa entre o dreno e a fonte (I_{Ds}), caso este transistor seja denominado tipo "n". Se estiver trabalhando com um tipo "p", a corrente inverte o sentido e caminha da fonte para o dreno. Existe aqui um conceito que difere dos resistores, capacitores e indutores. Nesses três dispositivos, a conrrente de dois terminais é controlada pela tensão aplicada nesses mesmos terminais, e assim, chama-se de impedância a constante que relaciona a tensão com a corrente nesses dispositivos, e o seu inverso é chamado de condutância. No transistor, a corrente do dreno para a fonte (I_{Ds}) é controlada pela tensão V_{Gs}, que é da porta para fonte. Esta constante que relaciona a corrente com a tensão será chamada de transcondutância.

Para modelar este transistor, deve-se observar como a corrente varia com (V_{Gs}) e com (V_{Ds})

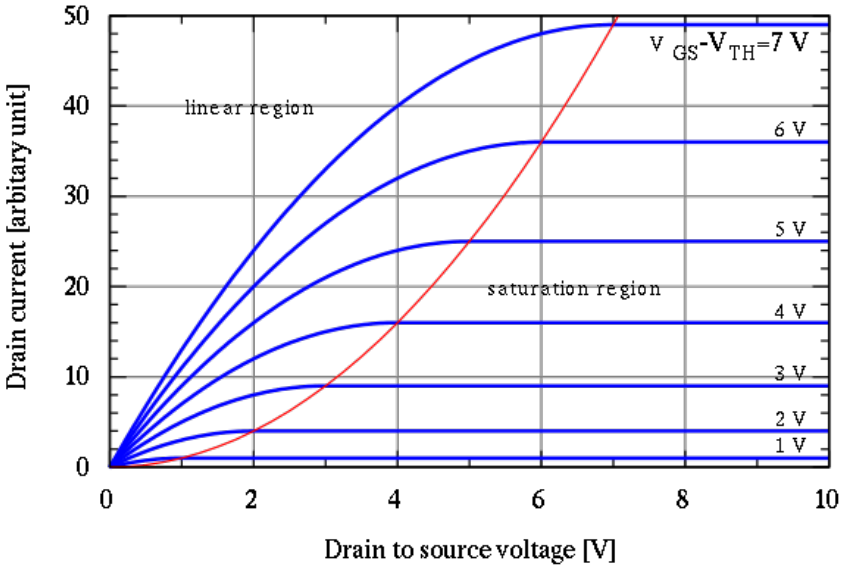

Figura 2.2: Curva azul: $I_D \times V_D$, curva vermelha: $I_D \times V_G$

assim como mostrado na figura 2.2. A curva azul mostra a relação entre o V_{Ds} e a corrente I_D, e a curva vermelha mostra a relação entre o V_{Gs} e a corrente I_D. Observa-se que o transistor tem três pontos de operação. O primeiro ponto está em corte e não conduz e ocorre quando o V_{Gs} é menor que a tensão limiar (V_T), o segundo conduz e tem uma resposta não linear a (V_{Ds}), porém muitas vezes esta relação é aproximada para uma relação linear, e a terceira é uma região de saturação em que a partir daquele valor de (V_{Ds}) a corrente é constante.

$$\frac{I_D}{W} = Q_n\left(V_D\right) \times v\left(V_D\right) \tag{2.1}$$

A corrente dividida pela largura do canal pode ser calculada de acordo com a equação 2.1. Em que $Qn(V_G)$ é a carga e $v(V_D)$ é a velocidade. Estes valores são obtidos através das seguintes equações:

$$Q_n\left(V_D\right) = -C_{ox}\left(V_G - V_T\right) \tag{2.2}$$

$$v\left(V_D\right) = \frac{\mu_{\text{eff}} V_D}{L} \tag{2.3}$$

$$v\left(V_D\right) = V_{Dsat} \tag{2.4}$$

μ_{eff} é a mobilidade do canal que define, através de um valor empírico, a facilidade que um elétron consegue se mover em um metal ou um semicondutor, L e W são o comprimento e a largura do canal e as equações 2.3 e 2.4 descrevem a velocidade na região linear e a velocidade na região de saturação, respectivamente.

O modelo de fonte virtual [1] tem o objetivo de reproduzir a dependência entre corrente de saída do transistor I_D e as tensões V_D e V_G. Analisando primeiro a dependência com a tensão V_D, que é representado na figura 2.2 pela curva azul, a equação 2.5 foi construída empiricamente para reproduzir o seu comportamento.

$$F_{sat} = \frac{V_D/V_{Dsat}}{[1 + \left(\frac{V_D}{V_{Dsat}}\right)^{\beta}]^{1/\beta}} \tag{2.5}$$

Em que

$$v\left(V_D\right) = F_{sat}\left(V_D\right) \times v_{sat} \tag{2.6}$$

Para testar seu comportamento, será necessária a divisão da curva em duas regiões: altos valores de V_D e baixos valores de V_D. Primeiro, para valores muito pequenos de V_D:

$$Fsat\left(V_d\right) \approx \frac{V_D}{V_{Dsat}} \tag{2.7}$$

$$v\left(V_D\right) \approx V_D \times \frac{V_{sat}}{V_D sat} \tag{2.8}$$

$$v\left(V_D\right) \approx VD\frac{V_{sat}}{V_{sat}L\mu_{\text{eff}}} \tag{2.9}$$

$$v\left(V_D\right) = \mu_{eff}\frac{V_D}{L} \tag{2.10}$$

Para grandes valores de V_D:

$$Fsat\left(V_D\right) \approx 1 \tag{2.11}$$

$$v\left(V_D\right) = V_{Dsat} \tag{2.12}$$

A velocidade pode ser analisada como uma relação entre a velocidade da região linear e a velocidade da região de saturação, sendo que o menor valor domina em relação a outra.

$$\frac{1}{v\left(V_D\right)} = \frac{1}{\mu_{eff}\frac{V_D}{L}} + \frac{1}{V_{Dsat}} \tag{2.13}$$

Reescrevendo a equação como uma função de $v(V_D)$.

$$v\left(V_D\right) = \frac{V_D/V_{Dsat}}{[1 + \left(\frac{V_D}{V_{Dsat}}\right)]} \tag{2.14}$$

A equação 2.14 difere da equação 2.5 apenas pelo fator beta que fornece um ajuste fino para a curva se adequar à resposta esperada. Portanto, esta função empírica fornece valores coerentes de velocidade para todas as duas regiões, tanto a de saturação como a linear, que representam bem o comportamento do transistor.

Existe uma série de parâmetros que são importantes para o modelo de fonte virtual. Alguns parâmetros influenciam diretamente no valor da corrente de saída I_D. Usando como exemplo a equação 2.1, o valor de W é fixo para um determinado transistor, portanto não deve ser variado em uma simulação. Da mesma forma, a carga $Q_n(V_D)$ depende da capacitância C_g, que é a capacitância da porta para o canal. Sendo assim não pode ser usado na simulação, pois depende da sua construção. Os outros dois parâmetros, entretanto, podem ser ajustados. A velocidade de injeção de carga tem influência tanto na corrente como no ponto de saturação da curva. O F_{sat} é a função apresentada na equação 2.5, que tem como um dos parâmetros no seu domínio a mobilidade das cargas.

$$V_{Dsat} = \frac{V_{x0} \times L_{eff}}{\mu_{\text{eff}}} \tag{2.15}$$

É possível, através da equação 2.15, perceber que esses mesmos parâmetros (mobilidade e velocidade) também influenciam na tensão do dreno de saturação. Esse parâmetro L_{eff} é o comprimento efetivo do canal e não pode ser ajustado para um determinado transistor. A Tensão de saturação tem influência no F_{sat} que, por sua vez, é diretamente proporcional a corrente do dreno.

O parâmetro beta é exclusivo do modelo de fonte virtual e serve para ajustar o joelho da curva apresentado na figura 2.2, assim como mencionado acima, para que a transição da região linear e a região de saturação seja feita de forma suave. Em geral, este valor é de aproximadamente 1.6 para transistores do tipo p e 1.8 para transistores do tipo n e é obtido empiricamente para cada curva.

Existe ainda a variável alpha, que deve ser achada empiricamente e também tem certa influência na corrente de dreno, ajudando na transição da curva da região linear para a corrente de saturação.

Assim como no modelo de um transistor MOSFET, o modelo de fonte virtual também possui uma tensão de limiar que determina a passagem da região de corte para a região linear (V_{T0}).

A corrente de saturação é um valor constante de I_D para qualquer valor de tensão maior que V_{Dsat}. Entretanto, existe uma inclinação na curva que faz a corrente crescer linearmente e no modelo de fonte virtual o parâmetro que regula esta inclinação é o δ.

O nome fonte virtual vem do seu ponto de estudo. Em um gráfico de energia versus posição existe uma barreira, onde no seu ponto máximo a carga pode ser aproximada para a equação 2.2 se o transistor for modelado corretamente. Este ponto é chamado de *virtual source*, marcado na figura 2.3 pelo ponto em que $x = x_0$.

Tabela 2.1: Parâmetros utilizados no modelo de fonte virtual

Parâmetro	Descrição
W	**Largura do transistor**
L	**Comprimento do transistor**
C_g	**Capacitância entre a porta e o canal**
δ	***Drain-induced-barrier-lowering* (DIBL)**
β	**Fator de saturação**
α	**Parâmetro empírico para transição entre forte e fraca inversão**
μ_{eff}	**Mobilidade do elétron**
v_{x0}	**velocidade de injeção**
$Vt0$	**tensão limiar**

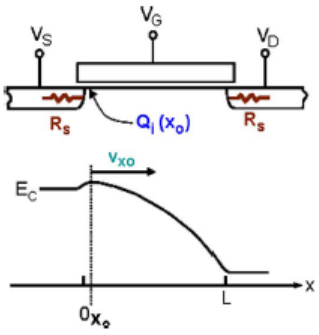

Figura 2.3: Gráfico de energia (E) versus posição (x)

2.2 Modelo UMEM

O modelo UMEM (*unified model and parameter extraction method*) é um modelo compacto implementado em AIM-spice e foi desenvolvido para modelar e extrair parâmetros de transistores de películas finas de a-SI:H, nas regiões linear e de saturação. A sua vantagem com relação a outros modelos criados anteriormente é que ele utiliza menos parâmetros que os que o antecederam (utilizavam 15 ou mais parâmetros), o que diminuiu o esforço computacional necessário. [2][3]

O AIM-Spice fornece, adicionalmente, um extrator de parâmetros que extrai de uma curva experimental valores para ajustar a simulação ao experimento real, o que otimiza o processo. Entretanto em alguns casos esse extrator é problemático e nem sempre os parâmetros que ajustam a curva de I_D versus V_D conseguem reproduzir a curva I_D versus V_G.[4][5]

2.3 Modelo MOSFET level 1

O MOSFET é um tipo de transistor de efeito de campo formado por um metal, um semicondutor e um óxido. Em geral os terminais são formados de polissilício sobre o canal e são separados deste por uma fina camada de dióxido de silício isolante.

Os semicondutores intrínsecos são cristais com nenhuma impureza ou defeito na rede. Nestes materiais não existem portadores de carga na temperatura 0 K uma vez que a banda de valência está completamente preenchida de elétrons e a banda de condução está vazia. Em temperaturas maiores que 0 kelvin, os pares elétron-buraco são gerados.

Além dos portadores intrínsecos, é possível criar portadores nos semicondutores propositalmente introduzindo impurezas no cristal. Este processo é chamado de dopagem e é a técnica mais comum para variar a condutividade do semicondutor. Quanto maior o número de portadores de carga na banda de condução, maior a corrente que pode ser gerada. [6]

Como apresentado na figura 2.2 o MOSFET tem três pontos de operação, a região de corte quando V_{Gs} é menor do que a tensão limiar, a região linear, quando V_{Gs} é maior que a tensão limiar porém é menor que V_{Ds}, e a região de saturação, quando V_{Gs} é maior que as tensões limiar e a tensão V_{Ds}. Os parâmetros comumente usados para modelar um MOSFET LEVEL 1 estão apresentados na tabela 2.2

2.4 OTFT - Organic thin film transistors

Os transistores de filmes finos têm um processo de fabricação muito menos complexo que os transistores de silício. Esta categoria envolve um grande número de diferentes transistores, que podem ser divididos em:

OFET (Organic Field-effect Transistors) - que funcionam similarmente ao MOSFET, em que um campo magnético é gerado em um meio dielétrico que separa a porta do semicondutor.

Tabela 2.2: Parâmetros utilizados no modelo MOSFET level 1

Parâmetro	Descrição
W	**Largura do canal**
L	**Comprimento do canal**
L_D	**Comprimento efetivo do canal**
W_D	**Largura efetiva do canal**
V_{T0}	**Tensão limiar**
C_{ox}	**Capacitância do óxido**
N_{sub}	**Dopagem do substrato**
R_S	**Resistência de contato da fonte**
R_D	**Resistência de contato do dreno**

OECT (Organic Electrochemical Transistors) - que operam induzindo uma reação de oxidação ou redução que influencia a corrente I_D, devido a tensão fornecida na porta.

CNTFET (Carbon Nanotube Field-effect Transistors) - são usados muitos nanotubos de carbono como semicondutores que conectam o eletrodo do dreno com a fonte. É visto como uma tecnologia promissora por conta das propriedades dos nanotubos de carbono.

Esses são apenas alguns exemplos de categorias. Os OTFT's têm a vantagem de ser muito mais baratos que os transistores de silício devido ao material utilizado e ao processo de fabricação, entretanto, seu funcionamento ainda está aquém do MOSFET, o que muitas vezes torna-o inviável de ser utilizado.

2.5 Verilog-A

Os modelos compactos descrevem de forma precisa o comportamento elétrico de dispositivos eletrônicos para prever as saídas de um circuito elétrico. Esses modelos são usados em simuladores como o SPICE. Com o desenvolvimento da tecnologia, novos efeitos físicos geram impacto no comportamento elétrico destes dispositivos e esses efeitos precisam ser, de alguma forma, modelados, adicionando equações nos modelos compactos. [7]

Verilog-A é uma linguagem padrão para modelagem de circuitos analógicos, é uma derivação do verilog-AMS, que trabalha com funções contínuas no tempo. Até então, as linguagens usadas para este tipo de modelagem eram o VHDL e o C.[8]

Uma grande vantagem é a facilidade de escrever as equações matemáticas que descrevem fisicamente os dispositivos, assim como altera-las, se necessário. Em outras linguagens, uma pequena alteração faria necessária uma revisão do código como um todo. O MATLAB é comumente usado para simular circuitos. Entretanto o seu código não pode ser diretamente usado em simuladores de circuitos, já o verilog-A é aceito em várias plataformas.

2.6 Qucs - Quite Universal Circuit Simulator

O Qucs é um simulador de circuitos produzido pela Qucs team com o qual o usuário pode montar um circuito em uma interface gráfica (GUI - Graphical User Interface) e simular pequenos sinais, grandes sinais e o comportamento de ruídos em circuitos. Após a simulação, o usuário pode ter acesso aos resultados de saída do circuito.

É possível, através do Qucs, apresentar o resultado de vários tipos de saídas: DC, AC, parâmetro-S, ruído e análise de transiente. É possível também importar modelos do SPICE para usar em simulações. Inclui uma lista grande de componentes e modelos disponíveis para uso, como diodos e transistores de diferentes tipos.

O Qucs funciona em vários sistemas operacionais, entre eles o Windows, MacOs, Solaris, NetBSD e FreeBSD, é de fácil operação e não necessita de licença para ser utilizado, por isso foi escolhido para este trabalho. [9]

Capítulo 3

Metodologia

3.1 Introdução

Primeiramente foi selecionado o Qucs (Quite Universal Circuit Simulator) para simular o modelo de fonte virtual através do código de fonte virtual fornecido pelo nanohub [1] que descreve o modelo e pode ser facilmente utilizado como um dispositivo neste simulador, fornecendo as características de saída.

Após a escolha da interface, é necessário mostrar que o modelo consegue reproduzir um transistor. Primeiramente foi utilizado dados do artigo [1], que fala sobre o modelo de fonte virtual e fornece as características de saída de uma simulação fornecendo os valores de cada parâmetro necessário para o modelo. Utilizando estes mesmos parâmetros, será comparado o resultado da simulação do Qucs com a simulação apresentada no artigo.

Para verificar a eficácia do modelo, foi registrado experimentalmente as características de saída de um transistor de filmes finos, e deve-se, empiricamente, utilizar-se dos parâmetros para que a curva seja reproduzida.

Até então, as curvas escolhidas para comparação foram de I_D versus V_D, e no intuito de fortalecer o modelo será comparado a curva I_D versus V_G de um gráfico, usando os mesmos parâmetros utilizados para reproduzir a curva fornecida pelo artigo [1].

Finalmente, este trabalho encerra comparando os resultados do modelo de fonte virtual com o modelo UMEM, tentando verificar qual dos dois consegue reproduzir de forma mais eficaz e quantos parâmetros são utilizados para tanto.

3.2 Simulação de um MOSFET no Qucs

Nesta seção, o leitor será introduzido à interface do Qucs. Uma simulação de um MOSFET será reproduzida para exemplificar o procedimento.

Primeiramente um projeto é criado, onde todas as figuras, circuitos, simulações ou códigos

utilizados serão relacionados a ele. Deve-se clicar em "*new*"e dar nome ao projeto.

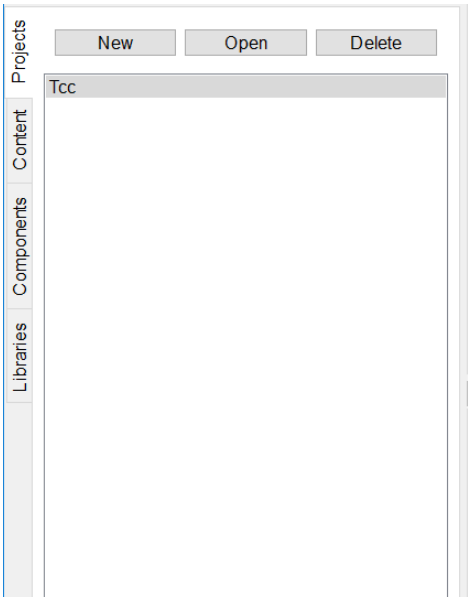

Figura 3.1: Criação de um projeto no Qucs

A interface de criação do circuito já está pronta pra ser utilizada. Para se escolher os componentes, deve-se clicar na aba "*components*", da figura 3.1, em que todas as opções estão separadas em categorias. Para esta simulação será necessário um transistor, algumas fontes DC e aterramentos.

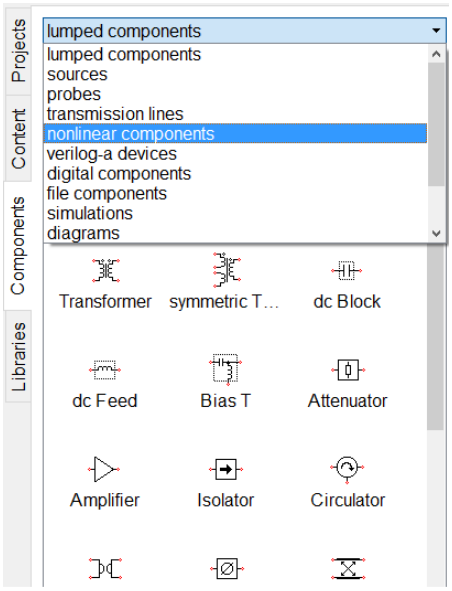

Figura 3.2: Opções de componentes no Qucs

Os transistores estão na aba "*non-linear*"components e as fontes estão na aba "*sources*". Ao selecionar cada componente, basta arrastar para a interface do programa onde se monta o circuito conectando tudo com fios.

Esta é uma montagem simples de um circuito que tem uma uma tensão V_{DD} fornecida por uma fonte de tensão (que neste caso é igual a tensão V_Ds) controlada por uma tensão V_{Gs}. Caso

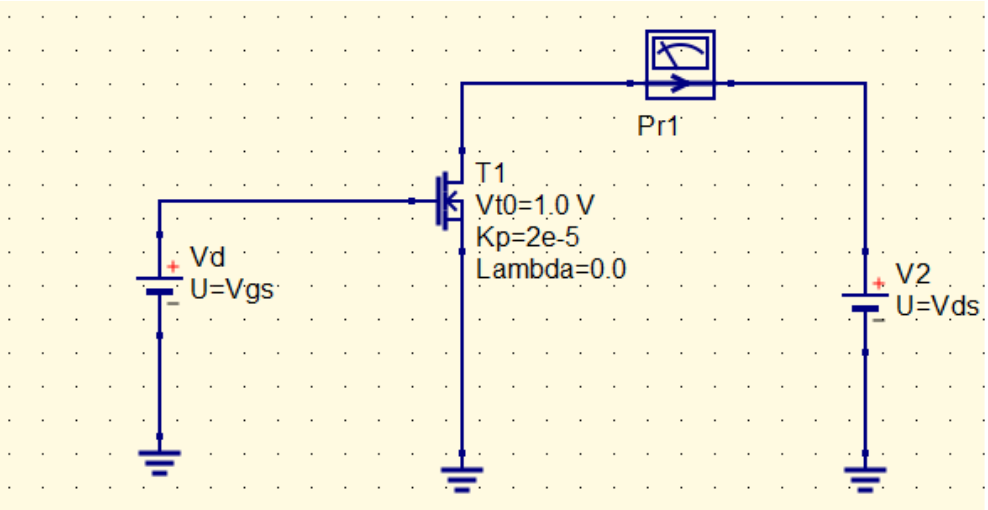

Figura 3.3: Circuito na interface do Qucs

seja necessário fazer alguma alteração em algum parâmetro do transistor, com um clique duplo no componente, uma janela apresentará todos os valores preestabelecidos para aquele transistor.

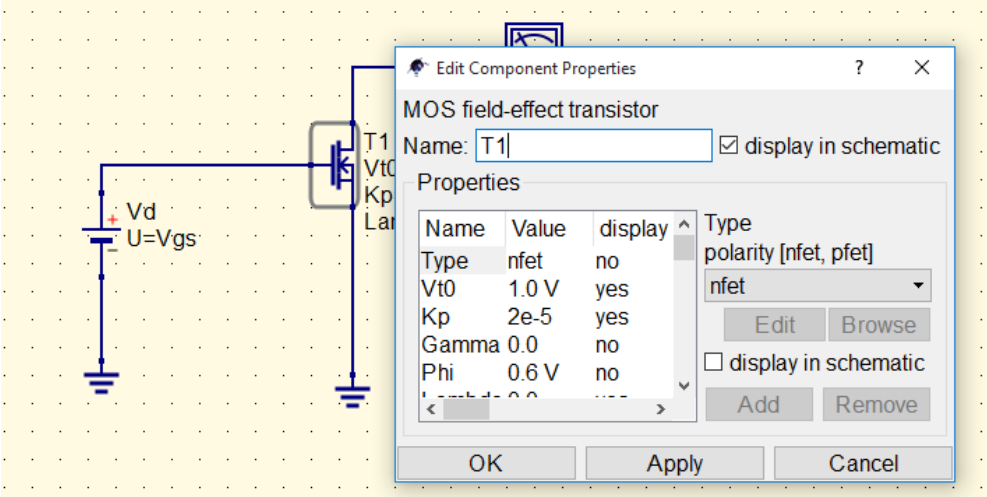

Figura 3.4: Parâmetros de um componente

Nos componentes apresentados na figura 3.2 existe uma aba de "*simulação*", onde se escolhe o tipo de simulação que o programa executará. Neste caso será escolhido simulação DC. Ainda preparando a simulação, para estudar o comportamento da corrente em função da tensão da porta, por exemplo, será escolhido um valor fixo para a função do dreno e deve-se variar a tensão da porta. Para tanto deve-se utilizar um "*parameter sweep*", também encontrado na mesma aba de "*simulação*", e determinar os valores iniciais e finais da variação, assim como a quantidade de pontos que a simulação terá.

Por fim, a simulação pode ser executada em "*simulate*", ou apenas pressionando F2, mas deve-se determinar qual a saída dessa simulação. Nos componentes da figura 3.2 existe uma aba denominada "*probes*", onde um amperímetro é escolhido e colocado na simulação para que se

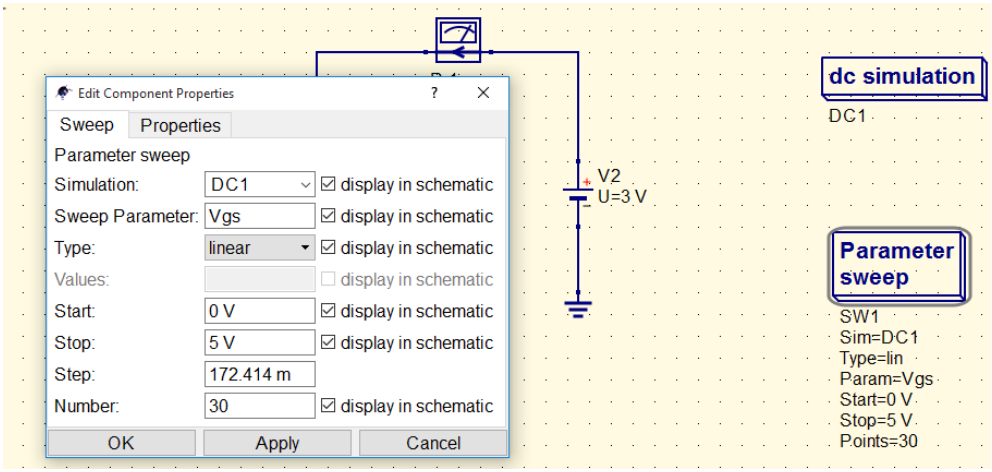

Figura 3.5: Ajustes para tornar uma constante uma variável

determine a saída.

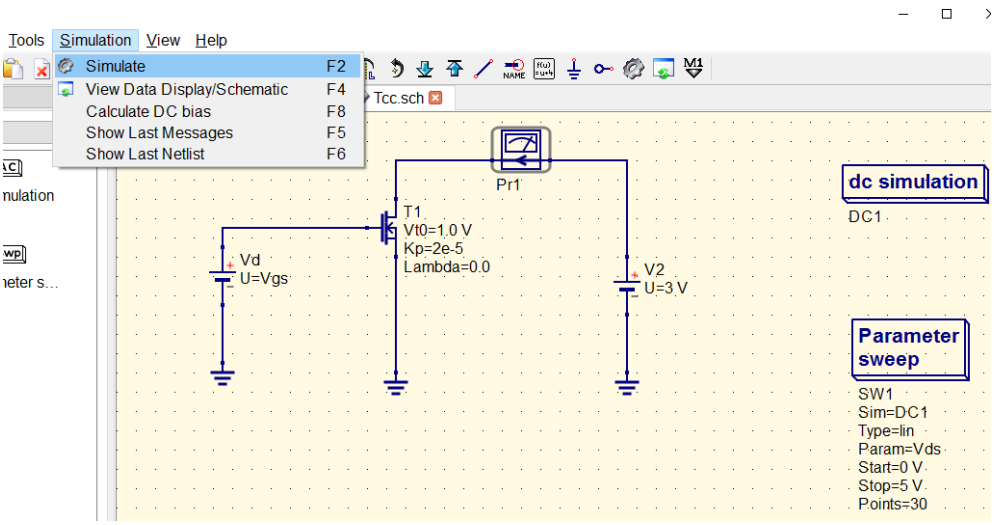

Figura 3.6: Execução da simulação do circuito

Ao executar a simulação, uma nova interface, é fornecida, e através de um "*componente*"obtido na aba "*diagrams*"pode-se visualizar o gráfico da forma que você preferir. Neste caso em particular, foi escolhido o plano cartesiano para representar a resposta da corrente I_D em função da tensão V_{Gs}.

Ao selecionar o "*cartesian*"como opção de gráfico, abrirá uma janela mostrando todas as saídas possíveis para este gráfico. Será selecionado neste caso a saída do amperímetro que foi utilizado no circuito da figura 3.6 e será gerada a curva representada na figura 3.7.

Existe a possibilidade, nesta interface de se importar dados. Ao selecionar "*project*"e "*import/export data*", abrirá uma janela onde você pode buscar o arquivo desejado em "*browse*"e selecionar "*convert*".

Esta curva pode ser exportada no formato "*.csv*". Este é um formato aceito em programas

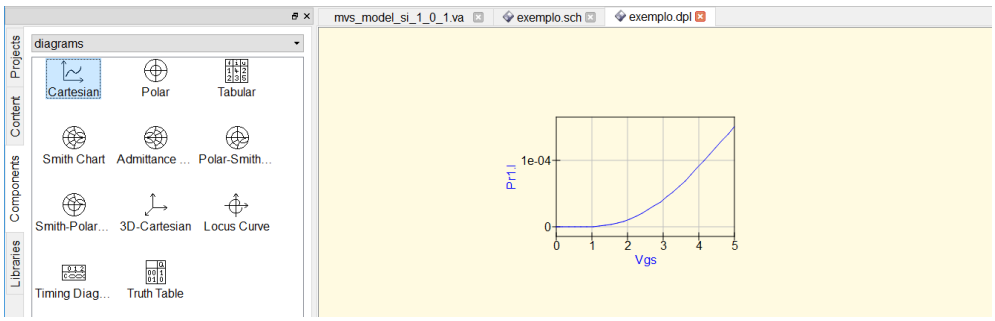

Figura 3.7: Corrente I_D em função da tensão V_{Gs}

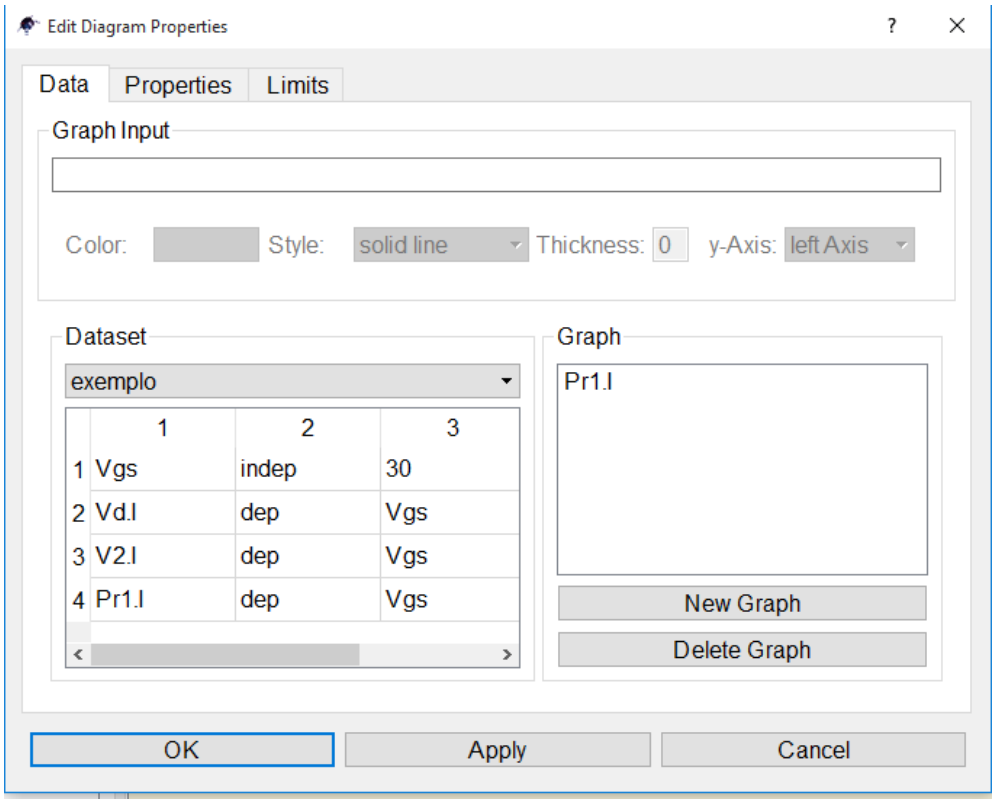

Figura 3.8: Opções de saída para a curva na simulação do Qucs

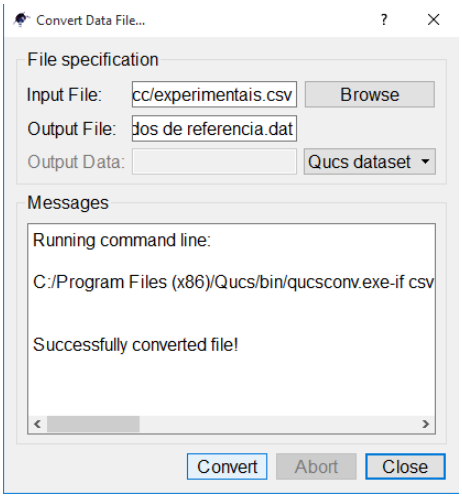

Figura 3.9: Importação de dados para a interface do Qucs

populares, como o excel e o Matlab. Para executar a exportação, é necessário selecionar a curva alvo e clicar em "*project*"e em seguida "*export to csv*".

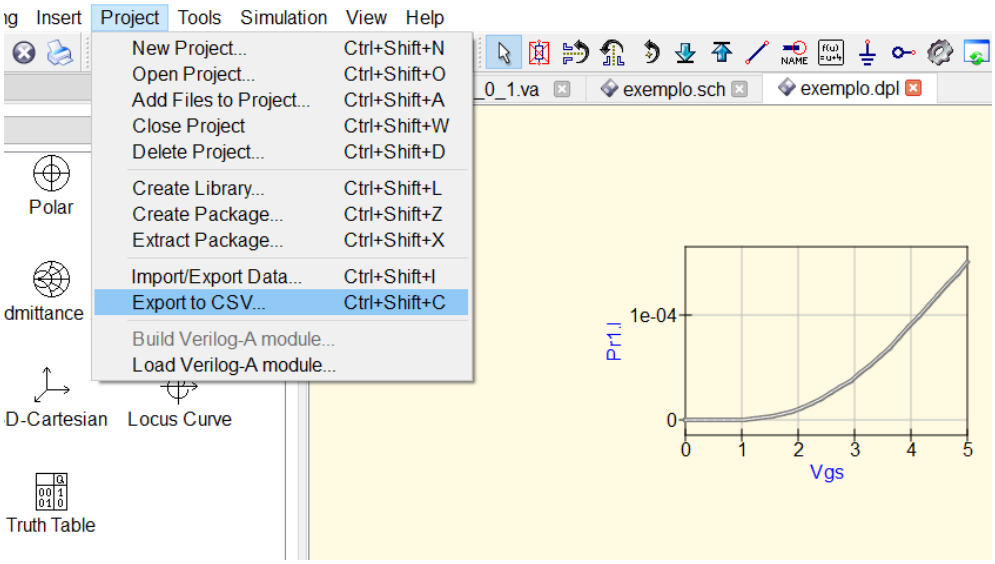

Figura 3.10: Exportar curvas no Qucs

3.3 Implementação do código em verilog-A para o modelo de fonte virtual

Primeiramente, serão comentados alguns trechos do código [10] para melhor entendimento do leitos. o arquivo completo está no anexo I.

```
‘include "\textit{constants.vams"
‘include "\textit{disciplines.vams"
```

São adicionadas bibliotecas. O "*constant.vams*"tem incluído o valor de constantes importantes para o código, como o valor da carga do elétron e da permissividade elétrica do vácuo. O disciplines.vams contém uma relação entre sinais físicos importantes para o verilog-A. As bibliotecas, a princípio, estavam na extensão .h, que é comumente aceita nos programas mais populares como o matlab, entretanto o Qucs não aceita este formato, portanto foi necessário uma biblioteca na extensão "*.vams*".

Em seguida, são definidos todos os parâmetros necessários para o código.

```
parameter real  version  = 1.01;  //mvs model version = 1.0.1
parameter integer  type  = 1  from [-1 : 1] exclude 0;  //Type of transistor. nFET type=1; pFET type=-1
parameter real  W           = 1e-4  from (0:inf);  //Transistor width [cm]
parameter real  Lgdr  = 80e-7  from (0:inf);  //Physical gate length [cm].
                                              //This is the designed gate length for litho printing.
parameter real  dLg  = 10.5e-7  from (0:inf);  //Overlap length including both source and drain sides [cm]
parameter real  Cg           = 2.2e-6  from (0:inf);  //Gate-to-channel areal capacitance
                                                      //at the virtual source [F/cm^2]
parameter real etov  = 1.3e-3  from (0:inf);  //Equivalent thickness of dielectric at S/D-G overlap [cm]
parameter real  delta  = 0.10  from [0:inf);  //Drain-induced-barrier-lowering (DIBL) [V/V]
parameter real  n0          = 1.5  from [0:inf);  //Subthreshold swing factor [unit-less]
                                                  //{typically between 1.0 and 2.0}
parameter real  Rs0  = 100  from (0:inf);  //Access resistance on s-terminal [Ohms-micron]
parameter real  Rd0  = 100  from (0:inf);  //Access resistance on d-terminal [Ohms-micron]
                                              //Generally, Rs0 = Rd0 for symmetric source and drain
parameter real  Cif  = 1e-12  from [0:inf);  //Inner fringing S or D capacitance [F/cm]
parameter real  Cof  = 2e-13  from [0:inf);  //Outer fringing S or D capacitance [F/cm]
parameter real  vxo  = 0.765e7  from (0:inf);  //Virtual source injection velocity [cm/s]
parameter real  mu          = 200  from (0:inf);  //Low-field mobility [cm^2/V.s]
parameter real  beta  = 1.7  from (0:inf);  //Saturation factor. Typ. nFET=1.8, pFET=1.6
parameter real       Tjun        = 298           from [173:inf);  //Junction temperature [K]
parameter real  phib  = 1.2;  //~abs(2*phif)>0 [V]
parameter real  gamma  = 0.0  from [0:inf);  //Body factor  [sqrt(V)]
parameter real  Vt0  = 0.486;  //Strong inversion threshold voltage [V]
parameter real  alpha  = 3.5;  //Empirical parameter for threshold voltage shift
                               //between strong and weak inversion.
parameter real  mc           = 0.2  from [0.01 : 10];  //Choose an appropriate value between 0.01 to 10
```

Os valores são padrões para se ter uma referência do que é escolhido. Vale ressaltar que o valor da constante "*type*"define o tipo de transistor que estará sendo usado, em que o valor "*1*"determina o transistor como tipo "*n*"e o valor "*-1*"determina o transistor como tipo "*p*". O código está todo comentado e ao lado de cada parâmetro existe a sua definição, assim como a unidade dele.

Uma parte importante do código é o calculo da corrente de dreno de saída do transistor. Como mencionado na revisão bibliográfica, a corrente pode ser calculada como carga multiplicada pela velocidade, e assim está exemplificado na equação do código.

```
//Total drain current
Id = Qinv_corr * vx0 * Fsat * W;
```

O W apresentado nesta equação não pode ser variado pois determina o valor da largura do canal, ele é fixo para um transistor, por isso ele costuma dividir o valor da corrente do outro lado

da igualdade. A velocidade apresentada como V_{xo} é um parâmetro do modelo e pode ser alterado como se desejar. A carga é calculada através das equações apresentadas a seguir:

```
//Charge at VS in saturation (Qinv)
     if (eta  <= ‘LARGE_VALUE) begin
Qinv_corr   =  Qref * ln( 1.0 + exp(eta) );
end
else begin
Qinv_corr = Qref * eta;
end
if (eta0 <= ‘LARGE_VALUE) begin
Qinv = Qref * ln( 1.0 + exp(eta0) );   // Compute charge w/ uncorrected intrinsic Vgs
                                       // for use later on in charge partitioning
end
else begin
Qinv = Qref * eta0;
end
```

E a variável definida por F_{sat} foi apresentada na equação 2.5, ele é apresentado no código da seguinte maneira:

```
//Transport equations
vx0 = vxo;
Vdsats = vx0 * Leff/ mu;
Vdsat = Vdsats * ( 1.0 - FF ) + phit * FF;  //  Saturation drain voltage for current
Vdratio = abs( Vdsi/ Vdsat);
Vdbeta = pow( Vdratio, beta);
Vdbetabeta  =  pow( 1.0 + Vdbeta, 1.0/ beta);
Fsat = Vdratio / Vdbetabeta;
```

3.4 Utilização do código Verilog-A no Qucs

Para conseguir executar uma simulação no Qucs usando o modelo de fonte virtual, é necessário compreender como criar um componente que responde de acordo com o modelo fornecido pelo código em Verilog-A. Primeiro o código escrito deve ser copiado para um script no Qucs e salvo com a extensão "*.va*", mantendo o nome previamente estabelecido pelo módulo.

Em seguida, deve-se abrir a aba "*project*"e selecionar a opção "*build verilog-A module*"que compilará através do ADMS e irá converte-lo para C++. O ADMS deve estar instalado junto com o Qucs. Ao finalizar este procedimento, uma nova interface de texto apresentará se a compilação foi executada com sucesso ou se houve erros.

Ao clicar em "*file*", usando a opção "*edit text simbol*", um símbolo será criado para o dispositovo da compilação. Nele, está identificado todas as entradas: porta (G), dreno (D), fonte (S) e bulk (B). Este símbolo pode ser editado para uma melhor visualização do circuito, porém neste projeto será utilizado o símbolo gerado pelo programa.

Após salvar, basta carregar este dispositivo para a lista de componentes selecionando em "*project*"a opção de "*load verilog-A module*".

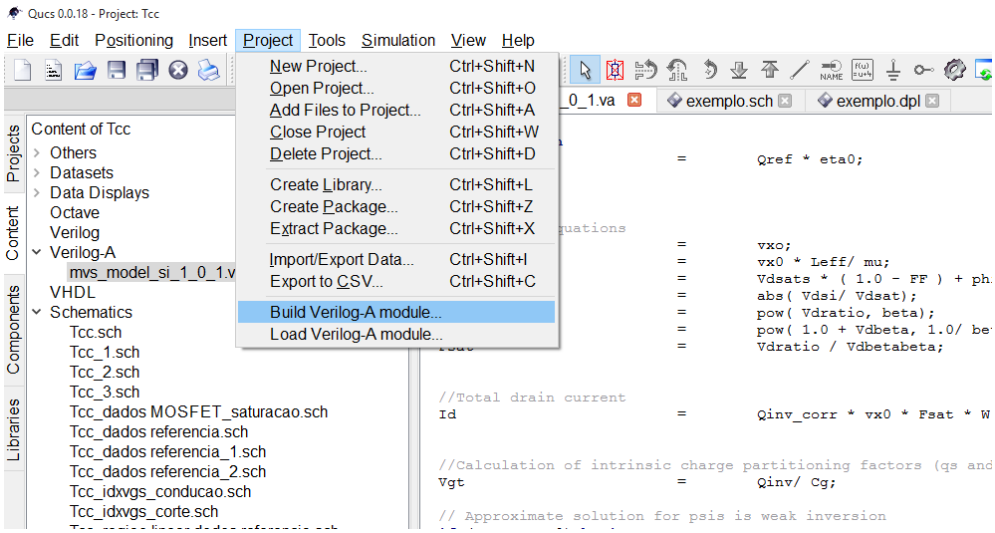

Figura 3.11: Compilando o código verilog-A

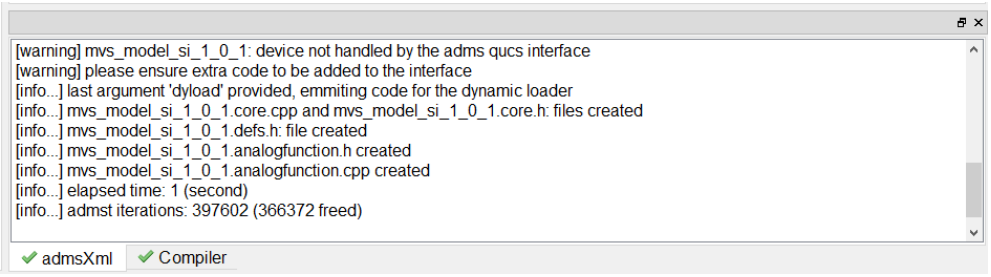

Figura 3.12: Interface de erros da compilação

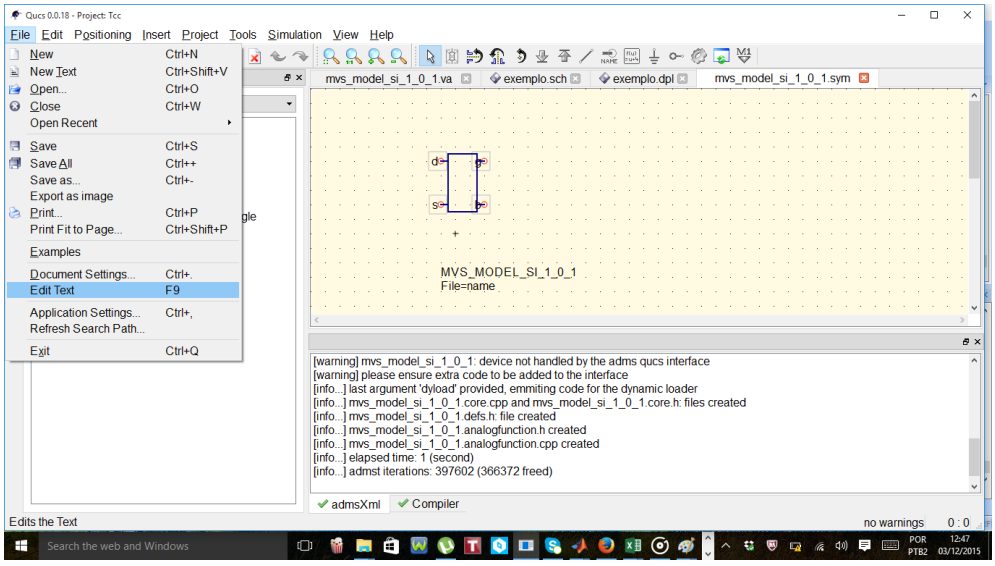

Figura 3.13: Dispositivo gerado a partir do código verilog-A

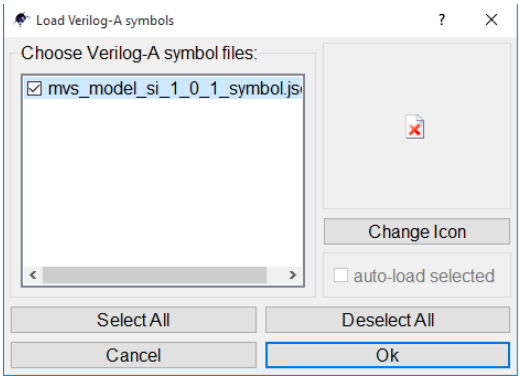

Figura 3.14: Carregando dispositivo

Um erro deverá aparecer aqui porque nenhum ícone foi escolhido para representar este dispositivo, porém isto não impede em nada de utilizar este componente.

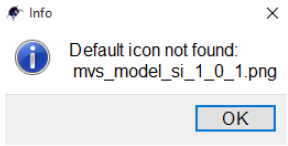

Figura 3.15: Mensagem de erro

Para escolher o componente, é necessário selecionar na aba da figura 3.2 a opção "*verilog – Auserdevices*" e selecionar o modelo do código de fonte virtual, que neste caso em particular, está sem ícone de representação.

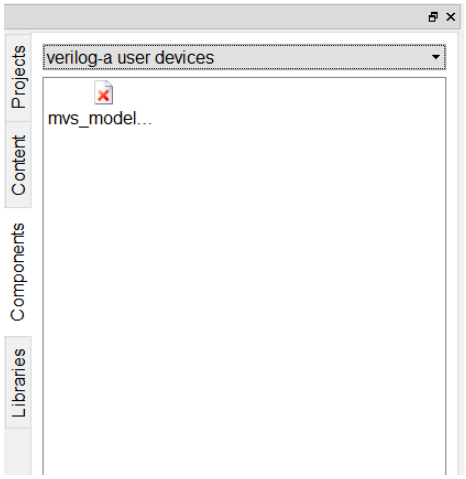

Figura 3.16: Dispositivo na interface Qucs

A partir daqui, a montagem do circuito pode ser executada com facilidade.

Capítulo 4

Resultados e análise

4.1 Verificação do modelo de fonte virtual através de simulações

Para verificação do modelo de fonte virtual, duas comparações serão feitas nesta seção. Primeiramente, utilizando dados obtidos do artigo [1], o gráfico de V_D versus I_D de um transistor p é apresentado como exemplo de uma simulação com o modelo de fonte virtual. Os mesmos parâmetros serão utilizados para tentar reproduzir a curva, utilizando o mesmo modelo para comprovar que o código em verilog-A está sendo executado de forma adequada com o Qucs. Para tanto, três etapas serão a base deste processo. Na primeira utilizaremos o mvs para comparar a região linear do transistor como pode-se observar na figura 4.3, obtendo uma semelhança de forma empírica sem se preocupar com os valores de referência.

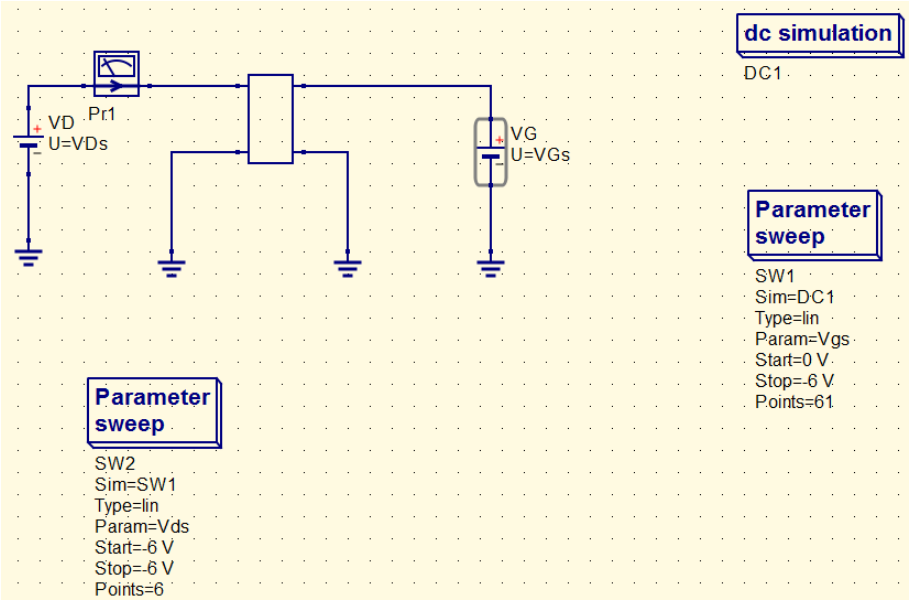

Figura 4.1: Dados de referência do modelo de fonte virtual

Na segunda etapa, o foco é reproduzir a região de saturação da curva 4.4, ainda de forma empírica, sem a preocupação de se utilizar os mesmos valores fornecidos no artigo de referência.

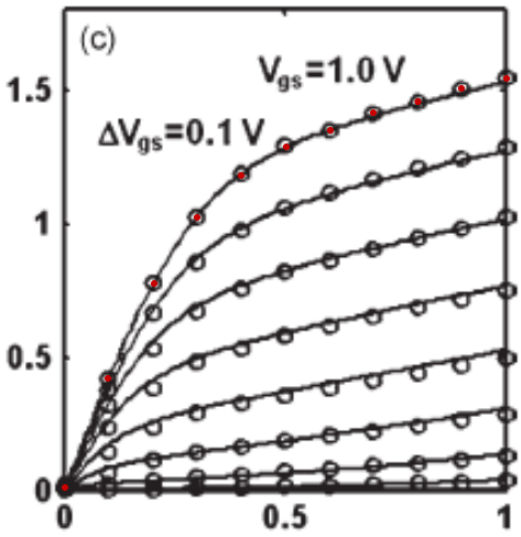

Figura 4.2: Dados de referência do modelo de fonte virtual

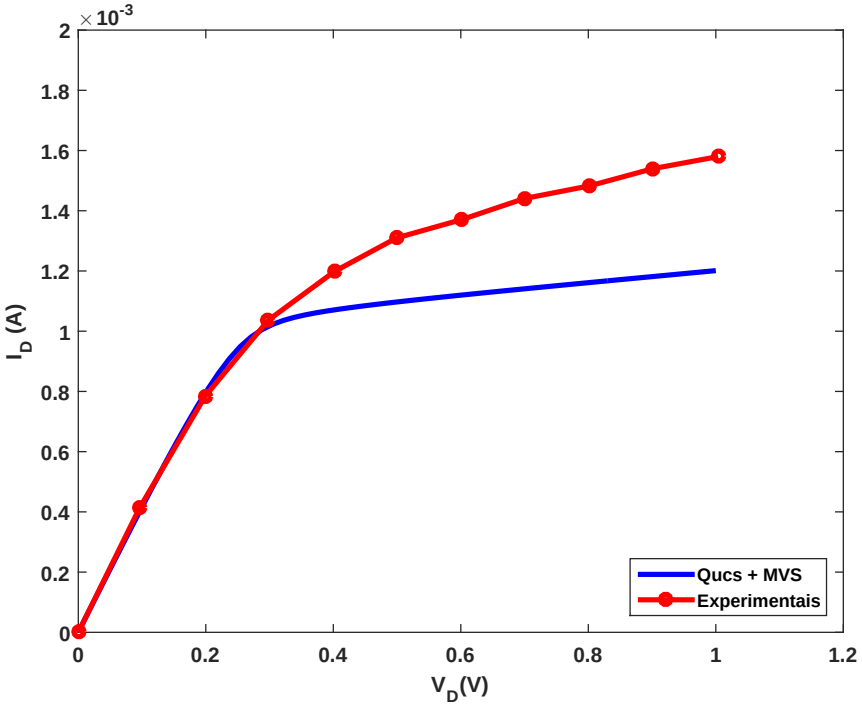

Figura 4.3: Reprodução da região linear. $V_G = 1$ V

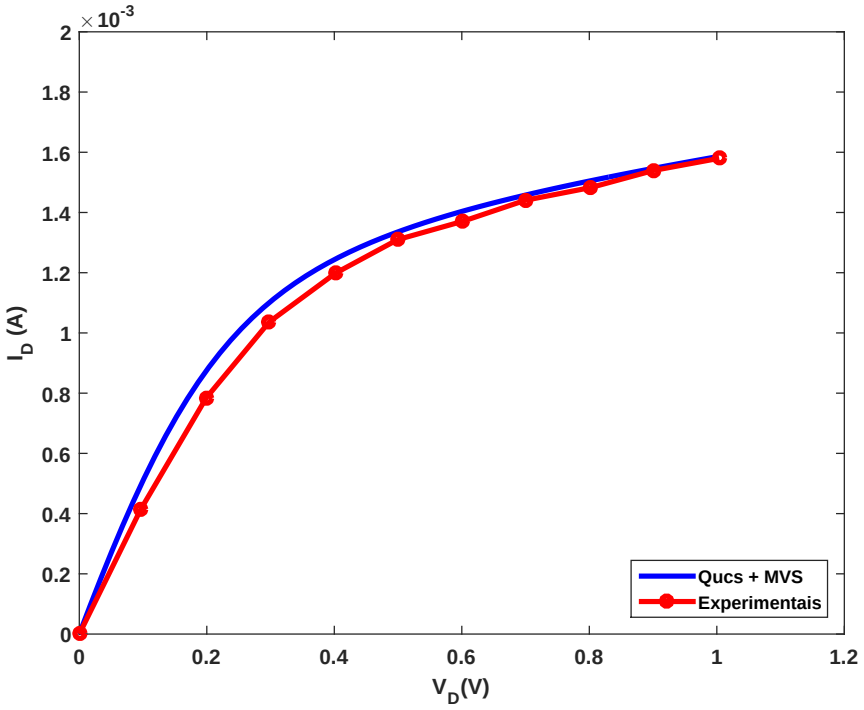

Figura 4.4: Reprodução da região de saturação. $V_G = 1$ V

Por fim, foram usados os mesmos valores utilizados pelo artigo [1] com o objetivo de se aproximar da curva como um todo 4.5. Como não foram informados todos os parâmetros, alguns ajustes foram feitos para se obter uma precisão maior.

O erro entre as duas curvas é bem pequeno, a reprodução feita no Qucs se aproximou bastante daquela apresentada no artigo, o que comprova que a simulação do Qucs com o código em verilog-A está condizente com aquele referido no artigo. Abaixo se encontra a tabela com os parâmetros que foram utilizados para a construção de cada curva.

4.2 Utilização do modelo de fonte virtual para reproduzir uma curva de um transistor de filmes finos

Nesta seção, o objetivo será verificar se o modelo de fonte virtual consegue reproduzir uma curva experimental [11] de um transistor de filmes finos que foi fornecido pela TUD (Technische Universität Dresden).

A mesma metodologia utilizada na seção anterior será repetida aqui. Primeiro a região linear será o foco da simulação, em seguida, a região de saturação e ,por fim, será reproduzida a curva como um todo.

Para se obter uma maior confiança no modelo, além da curva de I_D versus V_D, será também reproduzida a curva de I_D versus V_G. Os dados foram fornecidos para o mesmo transistor no mesmo circuito.

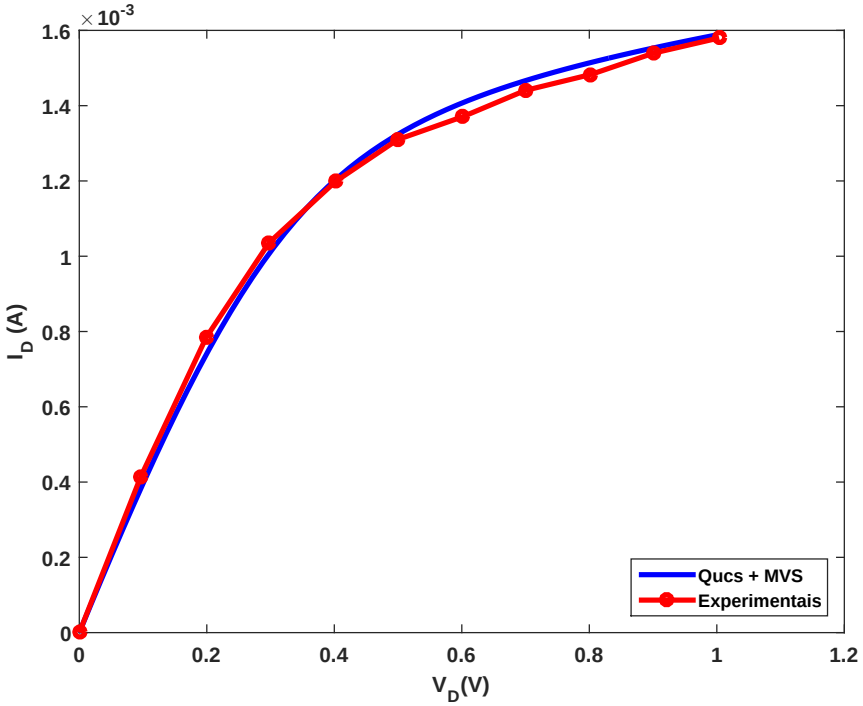

Figura 4.5: Reprodução do gráfico de referência. $V_G = 1\text{V}$

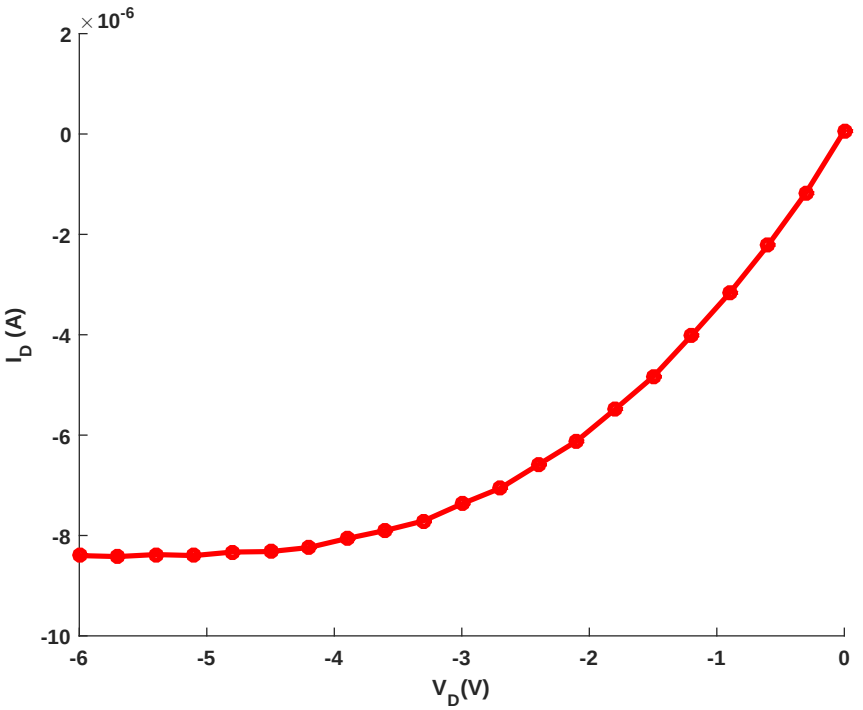

Figura 4.6: Gráfico de referência. V_G = -6

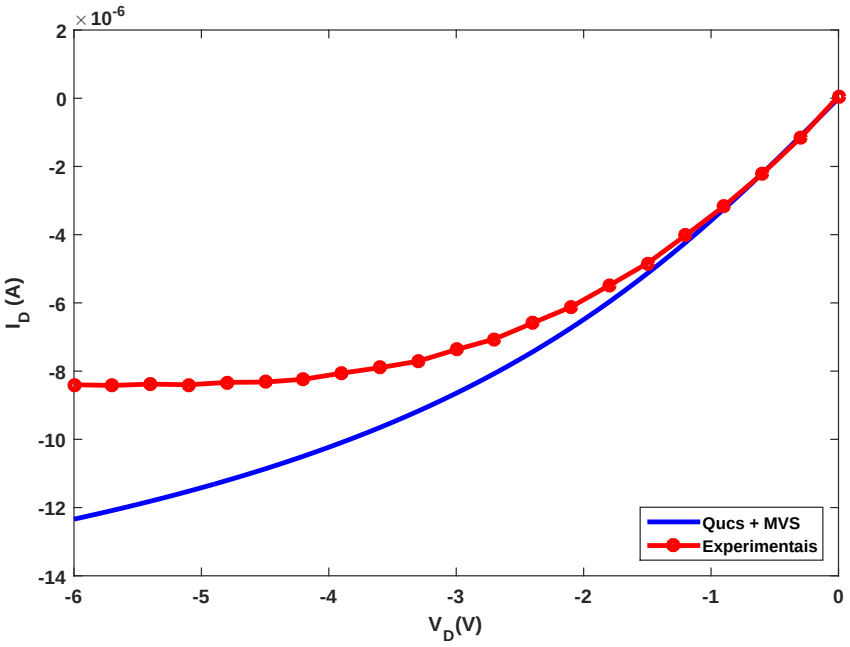

Figura 4.7: Reprodução da região linear dos dados experimentais. V_G = -6 V

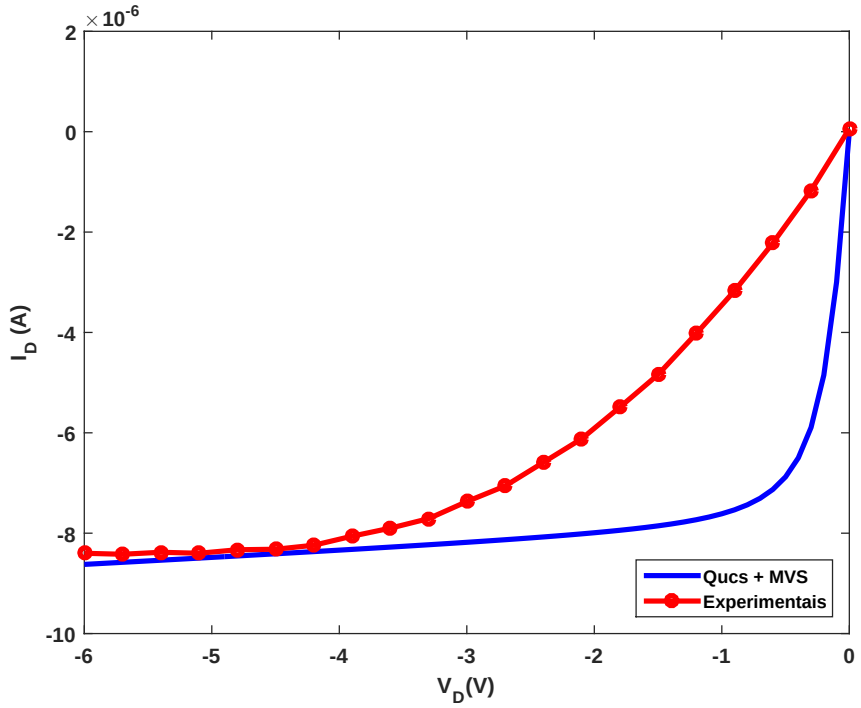

Figura 4.8: Reprodução da região de saturação dos dados experimentais. V_G = -6 V

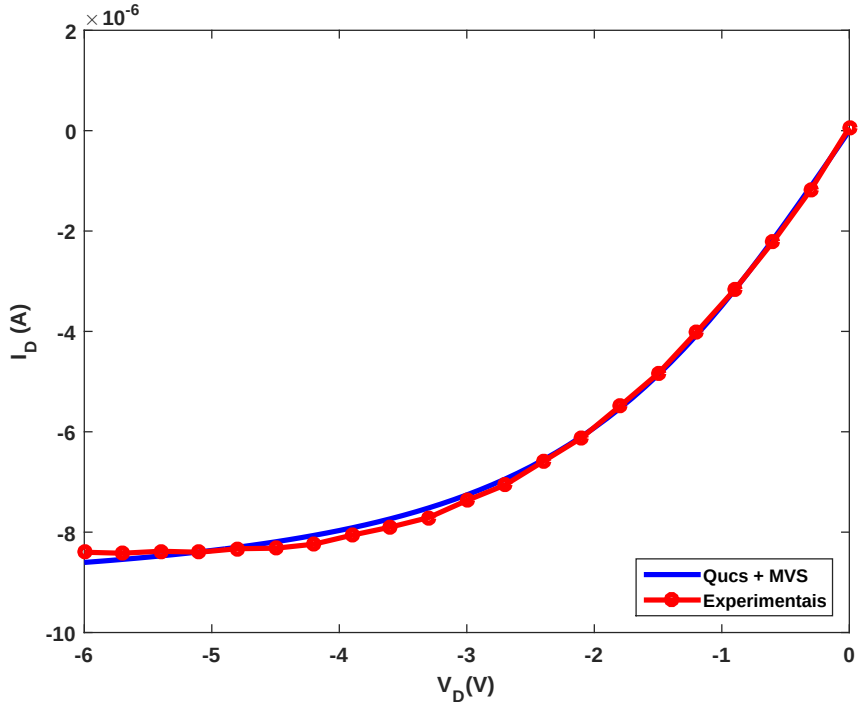

Figura 4.9: Reprodução do gráfico dos dados experimentais. V_G = -6 V

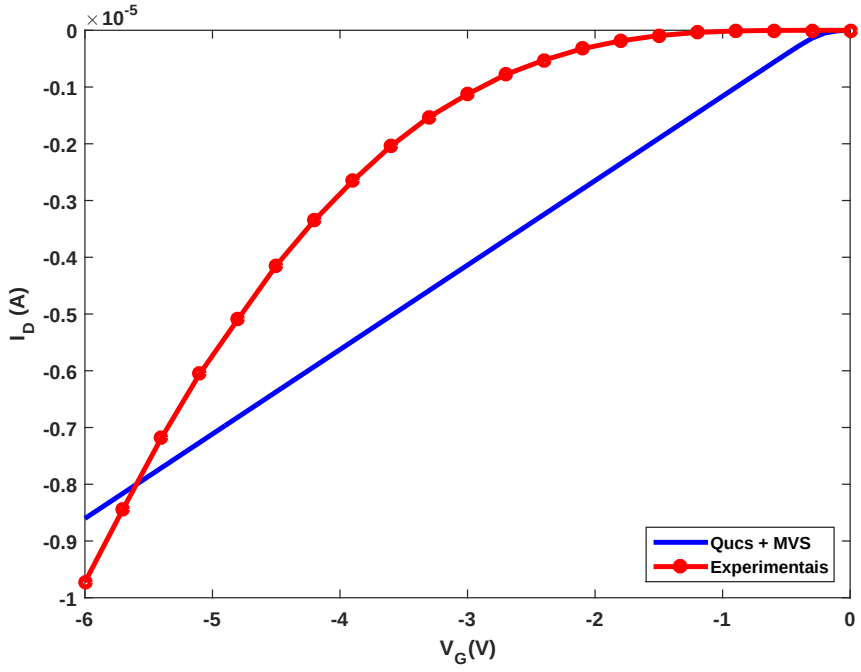

Figura 4.10: Reprodução da curva I_D versus V_G. V_D = -6 V

	Região Linear		Região de saturação		Curva completa	
	Referência	**Simulação**	**Ref.**	**Simu.**	**Ref.**	**Simu.**
$W[\mu m]$	1000	1000	1000	1000	1000	1000
$L\ [\mu m]$	3.5	3.5	3.5	3.5	3.5	3.5
$Delta\ [V/V]$	0.12	0.1	0.12	0.15	0.12	0.12
$V + xo\ [10^{-7}\ cm/s]$	1.4	1.2	1.4	1.2	1.4	1.4
$\mu_{\text{eff}}[cm^2/V\ s]$	250	250	250	200	250	250
β	1.8	1.7	1.8	1.8	1.8	1.8
V_{t0}		0.486		0.345		0.335

Tabela 4.1: Valores dos parâmetros utilizados para a construção das figuras 4.3,4.4 e 4.5

	Região Linear	**Região de saturação**	**Curva completa**
$W[10^-2\ m]$	0.1	0.1	0.1
$L\ [10^-2\ m]$	0.005	0.005	0.005
$Delta\ [V/V]$	0.1	0.1	0.01
$V_{xo}\ [cm/s]$	85	48	57
$\mu_{\text{eff}}[cm^2/V\ s]$	0.11	1	0.112
β	1.7	1.7	2.2
V_{t0}	0.1	0.1	0.28

Tabela 4.2: Valores dos parâmetros utilizados para a construção das figuras 4.7,4.8 e 4.9

Ao tentar reproduzir a curva de transferência usando os mesmos parâmetros que reproduziram a curva da figura 4.9, os resultados não foram satisfatórios. Como pode se observar na figura 4.10, a simulação não se aproximou da curva experimental, portanto novos parâmetros foram testados empiricamente para ajustar a curva.

Usando a mesma ideia que foi desenvolvida até aqui, primeiramente foi reproduzida a região de corte da curva e depois a região de condução.

É necessário observar que a princípio os parâmetros utilizados nas curvas das figuras 4.11 e 4.12, deveriam ser os mesmos da figura 4.9 entretanto a curva não se aproximou nem um pouco da experimental, portanto essas duas aproximações foram feitas também de forma empírica.

4.3 Resultados obtidos e comparação com os modelos UMEM, MOSFET level1

Para comparação dos modelos será apresentada a tentativa de cada modelo de reproduzir a curva experimental referida na seção anterior. Primeiro, a tentativa de cada um para a curva de I_D versus V_D.

Considerando as figuras 4.13 e 4.9, as três simulações se aproximam da curva experimental.

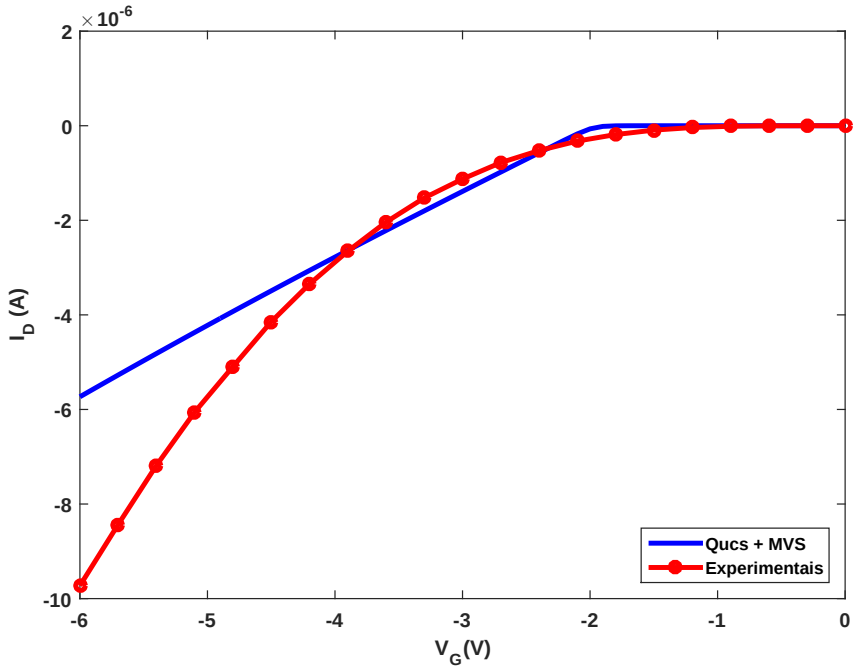

Figura 4.11: Reprodução da região de corte dos dados experimentais da curva I_D versus V_G. V_D = -6 V

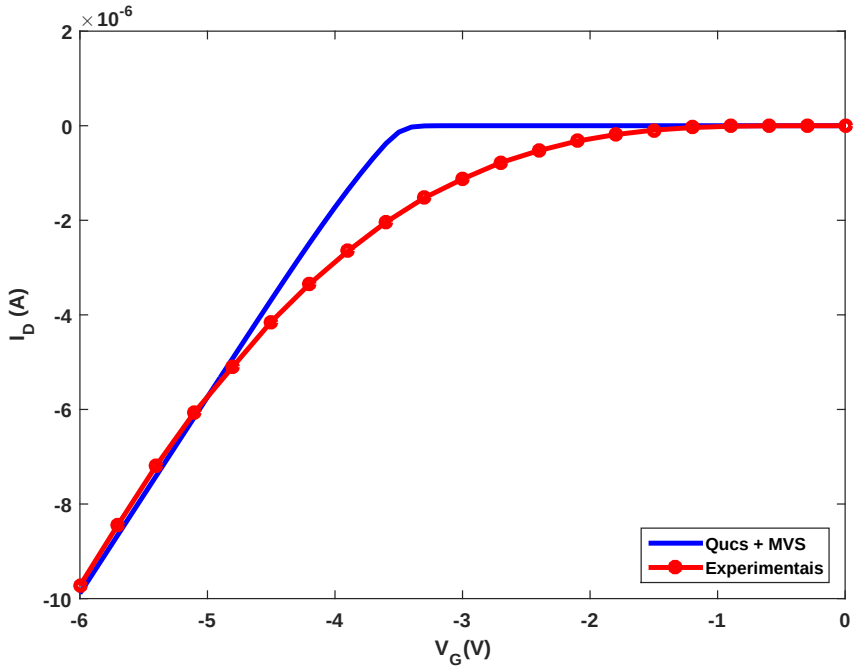

Figura 4.12: Reprodução da região de condução dos dados experimentais da curva I_D versus V_G. V_D = -6 V

	Região de corte	**Região de condução**
$W[10^-2\ m]$	0.1	0.1
$L\ [10^-2\ m]$	0.005	0.005
$Delta\ [V/V]$	0.04	0.03
$V_{xo}\ [cm/s]$	65	150
$\mu_{\text{eff}}[cm^2/V\ s]$	15	200
β	2	1.5
V_{t0}	3.5	3.8

Tabela 4.3: Valores dos parâmetros utilizados para a construção das figuras 4.11 e 4.12

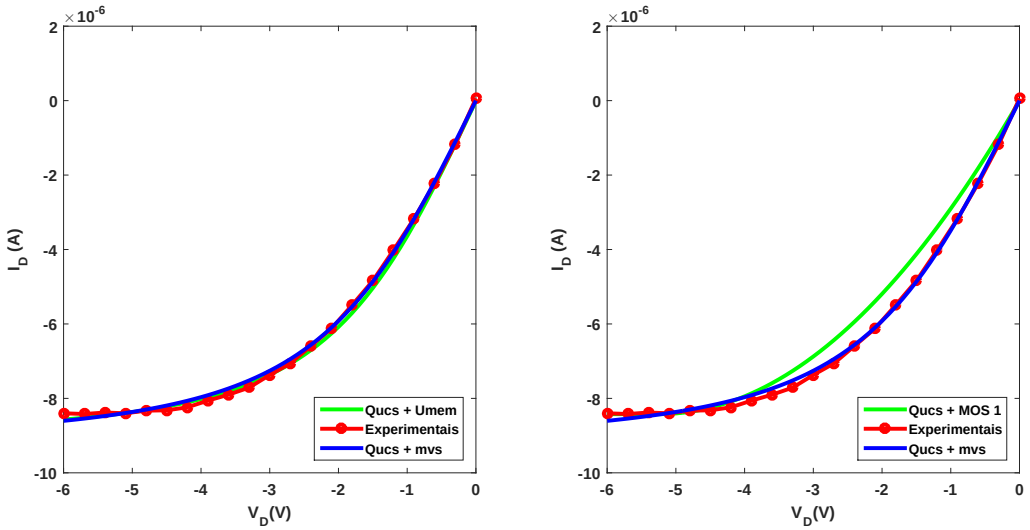

Figura 4.13: Comparação dos modelos UMEM e MOSFET level 1 com o mvs. V_G = -6 V

A simulação feita com o modelo de fonte virtual (mvs) tem uma aproximação com um erro praticamente nulo, entretanto se afasta dela na região de saturação. No modelo UMEM, o erro é maior na região linear, porém a distância entre as curvas na região de saturação cresce com menor intensidade. O MOSFET level 1 tem uma aproximação muito boa no começo e no final da curva, entretanto é a que mais se afasta dos dados experimentais na região de transição.

O modelo de fonte virtual consegue representar adequadamente os dados experimentais e necessita de menos parâmetros que os outros modelos, entretanto, ao analisar a curva de I_D versus V_G o modelo UMEM apresentou uma aproximação adequada para a curva completa e na simulação do mvs foi apenas conquistada a reprodução da região de corte e da região de condução separadamente, como apresentado nas figuras 4.11 e 4.12.

Tabela 4.4: Valores dos parâmetros utilizados para a construção da figura 4.13 com o modelo UMEM

Parâmetro	**Valor**
V_{DS} **[V]**	-6
V_{T} **[V]**	-3.95
γ_{a}	5
V_{aa} **[V]**	3
R [Ω]	3.7×10^5
α_{s}	1.165
m	2.2
λ **[1/V]**	-3×10^{-3}
ξ	7

Tabela 4.5: Valores dos parâmetros utilizados para a construção das figuras 4.13 e 4.14 com o modelo MOSFET level 1

Parâmetro	**Valor**
V_{T0} **[V]**	-0.76
$\mu_{\mathrm{eff}}[cm^2/V\ s]$	-0.1085
R_s,R_D **[Ohms]**	1000
N_{sub} **[1**/cm^3]	2.1 $10^1$4
NFS **[1**/cm^2 V]	1
K_p [A/V^2]	3.010 10^-8
m	2.2
C_{ox} [F/m^2]	2.773 10^{-7}
T_{ox} [nm]	30

Tabela 4.6: Valores dos parâmetros utilizados para a construção da figura 4.14 com o modelo UMEM

Parâmetro	**Valor**
V_{DS} **[V]**	-6
V_{T} **[V]**	-4
γ_{a}	6
V_{aa} **[V]**	3
R [Ω]	2×10^5
α_{s}	0.78
m	1.5
λ **[1/V]**	-3×10^{-3}
ξ	2

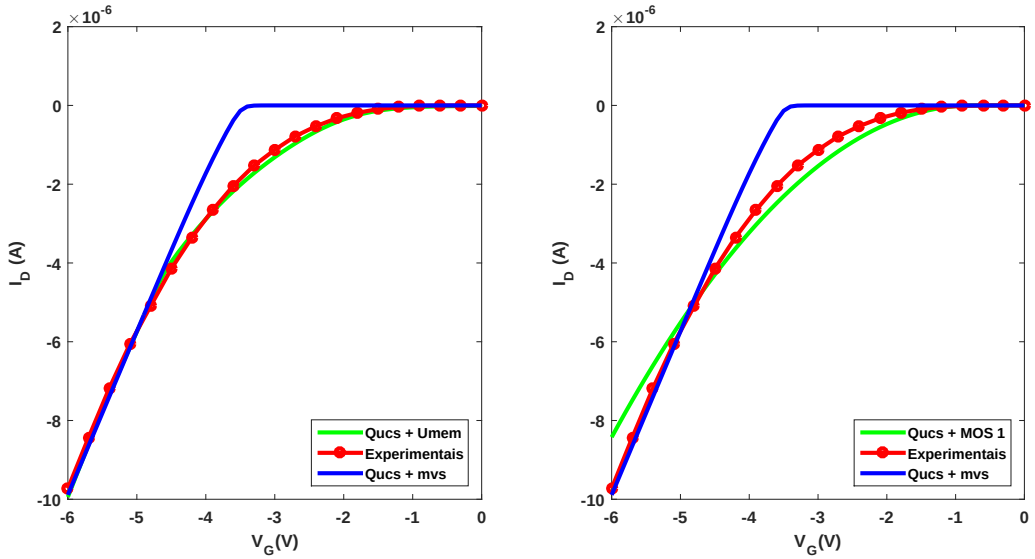

Figura 4.14: Reprodução da curva de I_D versus V_G para o modelo UMEM, MOSFET level 1 e mvs. V_D = -6

Capítulo 5

Conclusões

Como modelo, o mvs funcionou de forma eficaz e eficiente. As simulações da curva que mostram a dependência de I_D versus V_D conseguiram ser representadas com um erro muito pequeno e os parâmetros que são utilizados para essa aproximação é muito menor que a do modelo Umem. No modelo de fonte virtual são utilizados cinco parâmetros principais e no modelo Umem são necessários dez parâmetros.

Entretanto, para a simulação da curva de I_D versus V_G, o modelo Umem foi muito eficaz na reprodução, enquanto o modelo mvs conseguiu simular cada região da curva separadamente. Cabe então saber como utilizar cada modelo.

O modelo MOSFET, para simular os transistores de filmes finos, apresentou o maior erro na curva I_D versus V_D, entretanto conseguiu um erro muito menor que o modelo de fonte virtual na curva I_G versus V_G.

O mvs é um modelo compacto simples de ser utilizado, mas necessita de mais estudos quanto a característica de transferência que representa a dependência da corrente I_D com a tensão V_G. Além disso, deve ainda ser testado em outros protótipos de transistores de filmes finos para ter uma maior confiança no modelo.

Deve-se ainda questionar se esta tecnologia (Qucs + mvs) pode ser usada para projetar circuitos eletrônicos baseados em OTFT's, como por exemplo circuitos constituídos de componentes básicos do sistema RFID (Radio Frequency identification).

REFERÊNCIAS BIBLIOGRÁFICAS

[1] KHAKIFIROOZ, A.; NAYFEH, O. M.; ANTONIADIS, D. A simple semiempirical short-channel mosfet current voltage model continuous across all regions of operation and employing only physical parameters. *IEEE TRANSACTIONS ON ELECTRON DEVICES*, v. 56, p. 1674–1680, 2009.

[2] CERDEIRA, A. et al. New procedure for the extraction of basic a-siÇh tft model parameters in the linear and saturation regions. *SOLID STATE ELECTRONICS*, v. 49, p. 1077–1080, 2001.

[3] ESTRADA, M. et al. Accurate modeling and parameter extraction method for organic tfts. *SOLID STATE ELECTRONICS*, v. 49, p. 1009–1016, 2005.

[4] ESTRADA, M. et al. Mobility model for compact device modeling of otfts made with different materials. *SOLID STATE ELECTRONICS*, v. 52, p. 787–794, 2008.

[5] CERDEIRA, A.; nIGUEZ, M. E. B. I.; S.SOTO. Modeling the subthreshold region of otfts. *SOLID STATE ELECTRONICS*, p. 1033–1036, 2011.

[6] SERGIO, M. R. *Materiais e dispositivos eletrônicos*. [S.l.]: Editora Livraria da Fisica, 2004.

[7] MCANDREW1, C. C. et al. Best practices for compact modeling in verilog-a. *MICROELETRONIC ENGINEERING*, p. 383–396, 2015.

[8] CORAM, G. J. Howto (and how not to)write a compact model in verilog-a. *IEEE International Behavioral Modeling and Simulation Conference*, p. 97–106, 2004.

[9] QUCS Quite Universal Circuit Simulator. `http://qucs.sourceforge.net/`. Accessed: 2015-07-18.

[10] MVS Virtual source model. `https://nanohub.org/resources/vsmod`. Accessed: 2014-09-20.

[11] MOHAMMADI, S. *Untitled manuscript*. 2014. Unpublished.

ANEXOS

I. CÓDIGO DO MODELO DE FONTE VIRTUAL EM VERILOG-A

```
////////////////////////////////////////////////////////////////////////////////////////////////
//Copyright @ 2013 Massachusetts Institute of Technology (MIT)

//The terms under which the software and associated documentation (the Software) is provided are as the following:

//The Software is provided "as is", without warranty of any kind, express or implied,
//including but not limited to the warranties of merchantability, fitness for a particular
//purpose and noninfringement. In no event shall the authors or copyright holders be liable
//for any claim, damages or other liability, whether in an action of contract, tort or
//otherwise, arising from, out of or in connection with the Software or the use or other
//dealings in the Software.

//MIT grants, free of charge, to any users the right to modify, copy, and redistribute
//the Software, both within the user's organization and externally, subject to the
//following restrictions:

//1. The users agree not to charge for the MIT code itself but may charge for additions,
//extensions, or support.

//2. In any product based on the Software, the users agree to acknowledge the MIT VS
//Model Research Group that developed the software. This acknowledgment shall appear
//in the product documentation.

//3. The users agree to obey all U.S. Government restrictions governing redistribution
//or export of the software.

//4. The users agree to reproduce any copyright notice which appears on the software
//on any copy or modification of such made available to others.

//Agreed to by
//Dimitri A. Antoniadis, MIT
//May 27 2013
////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

// VerilogA for virtual-source (VS) based self-consistent transport/capacitance
//model for Si MOSFET
// transport model:  A. Khakifirooz, et al, p. 1674, T-ED 2009.
// charge model: L. Wei et al, p. 1263, T-ED 2012.
// Implemented on July 15, 2013 by S. Rakheja
// Modified on Sep. 19, 2013 by S. Rakheja

`include "constants.vams"
`include "disciplines.vams"

module mvs_model_si_1_0_1(d, g, s, b);
inout d, g, s, b;
electrical d, g, s, b;
electrical di, si;

// Original VS parameters
```

```
parameter real  version  = 1.01;  //MVS model version = 1.0.1
parameter integer  type  = 1  from [-1 : 1] exclude 0;  //type of transistor. nFET type=1; pFET type=-1
parameter real  W          = 1e-4  from (0:inf);  //Transistor width [cm]
parameter real  Lgdr  = 80e-7  from (0:inf);  //Physical gate length [cm].
                                              // This is the designed gate length for
//litho printing.
parameter real  dLg  = 10.5e-7  from (0:inf);  //Overlap length including both source and drain sides [cm]
parameter real  Cg          = 2.2e-6  from (0:inf);  //Gate-to-channel areal
                                                     //capacitance at the virtual
//source [F/cm^2]
parameter real etov  = 1.3e-3  from (0:inf);  //Equivalent thickness of
                                                //dielectric at S/D-G overlap [cm]
parameter real  delta  = 0.10  from [0:inf);  //Drain-induced-barrier-lowering (DIBL) [V/V]
parameter real  n0        = 1.5  from [0:inf);  //Subthreshold swing factor [unit-less]
                                                  //{typically between 1.0 and 2.0}
parameter real  Rs0  = 100  from (0:inf);  //Access resistance on s-terminal [Ohms-micron]
parameter real  Rd0  = 100  from (0:inf);  //Access resistance on d-terminal [Ohms-micron]
                                               // Generally, Rs0 = Rd0 for symmetric source and drain
parameter real  Cif  = 1e-12  from [0:inf);  //Inner fringing S or D capacitance [F/cm]
parameter real  Cof  = 2e-13  from [0:inf);  //Outer fringing S or D capacitance [F/cm]
parameter real  vxo  = 0.765e7  from (0:inf);  //Virtual source injection velocity [cm/s]
parameter real  mu      = 200  from (0:inf);  //Low-field mobility [cm^2/V.s]
parameter real  beta  = 1.7  from (0:inf);  //Saturation factor. Typ. nFET=1.8, pFET=1.6
parameter real    Tjun    = 298       from [173:inf);  //Junction temperature [K]
parameter real  phib  = 1.2;                      // ~abs(2*phif)>0 [V]
parameter real  gamma  = 0.0  from [0:inf);  //Body factor  [sqrt(V)]
parameter real  Vt0  = 0.486;                    // Strong inversion threshold voltage [V]
parameter real  alpha  = 3.5;                      // Empirical parameter for threshold voltage
                                                   //shift between strong and weak inversion.
parameter real  mc      = 0.2  from [0.01 : 10];  //Choose an appropriate value between
                                                      //0.01 to 10
                                                    // For, values outside of this range,
//convergence or accuracy of results is not
//guaranteed

parameter integer CTM_select  = 1     from [1 : inf);  //If CTM_select = 1, then classic DD-NVSAT
                                                 //model is used
                                                 // For CTM_select other than 1,blended
//DD-NVSAT and ballistic charge transport
//model is used
parameter real  $CC$  = 0         from [0:inf);  //Fitting parameter to adjust Vg-dependent
                                                 //inner fringe capacitances(Not used in
//this version)
parameter real  $nd$  = 0         from [0:inf);  //Punch-through factor [1/$V$]

`define  SMALL_VALUE  (1e-10)
`define  LARGE_VALUE  (40)

real Rs, Rd, Vds, Vgs, Vgsraw, Vgd, Vgdraw, Vbs, Vdsi, Vgsi, Vgdi, Vbsi, dir;
real Leff, me, S, phit;
real n, nphit, aphit, Vtpcorr, eVgpre, FFpre, ab, Vcorr, Vgscorr, Vbscorr, Vt0bs, Vt0bs0, Vtp, Vtp0;
real eVg, FF, eVg0, FF0, Qref, eta, eta0;
real Qinv, Qinv_corr, vx0, Vdsats, Vdsat,Vdratio, Vdbeta, Vdbetabeta, Fsat, Id ;
real Vgt, psis, Vgta, Vdsatq, Fsatq, x, den;
real qsc, qdc, qi, kq, kq2, kq4, tol, qsb, qdb, qs, qd, Qs, Qd;
real Qb, etai, Qinvi, dQinv, dibl_corr;
real Qinvs, Qinvd, Qsov, Qdov, Vt0x, Vt0y, Fs_arg, Fs, Fd_arg, Fd, FFx, FFy, Qsif, Qdif, Qg, a, Cofs, Cofd;


analog begin
```

```
//Voltage definitions
Vgsraw = type * ( V(g) - V(si) );
Vgdraw = type * ( V(g) - V(di) );
if (Vgsraw >= Vgdraw) begin
Vds = type * ( V(d) - V(s) );
Vgs = type * ( V(g) - V(s) );
Vbs = type * ( V(b) - V(s) );
Vdsi = type * ( V(di) - V(si) );
Vgsi = Vgsraw;
Vbsi = type * ( V(b) - V(si) );
dir = 1;
end
else begin
Vds = type * ( V(s) - V(d) );
Vgs = type * ( V(g) - V(d) );
Vbs = type * ( V(b) - V(d) );
Vdsi = type * ( V(si) - V(di) );
Vgsi = Vgdraw;
Vbsi = type * ( V(b) - V(di) );
dir = -1;
end


    //Parasitic element definition
Rs = 1e-4/ W * Rs0;                 //s-terminal resistance [ohms]
Rd = Rs;                           //d-terminal resistance [ohms] For symmetric source and
                                        //drain Rd = Rs.
      //Rd = 1e-4/ W * Rd0;          //  d-terminal resistance [ohms] {Uncomment for
                                    //asymmetric source and drain resistance.}
    Cofs = ( 0.345e-12/ etov ) * dLg/ 2.0 + Cof;  //  s-terminal outer fringing cap [F/cm]
    Cofd = ( 0.345e-12/ etov ) * dLg/ 2.0 + Cof;  //  d-terminal outer fringing cap [F/cm]
  Leff =  Lgdr - dLg;                                 //  Effective channel length [cm].
                                                  //After subtracting overlap lengths on s
//and d side

    phit = $vt(Tjun);           //Thermal voltage, kT/q [V]
  me = (9.1e-31) * mc;      //Carrier mass [Kg]
  n  = n0 + nd * Vds;       //Total subthreshold swing factor taking punchthrough into
                           //account [unit-less]
  nphit = n * phit;          //Product of n and phit [used as one variable]
  aphit = alpha * phit;      //  Product of alpha and phit [used as one variable]


         //Correct Vgsi and Vbsi
         //Vcorr is computed using external Vbs and Vgs but internal Vdsi, Qinv and Qinv_corr
//are computed with uncorrected Vgs, Vbs and corrected Vgs, Vbs respectively.

Vtpcorr = Vt0 + gamma * (sqrt(abs(phib - Vbs))- sqrt(phib))- Vdsi * delta; //Calculated from
                                                                          //extrinsic Vbs
eVgpre = exp(( Vgs - Vtpcorr )/ ( aphit * 1.5 ));  //Calculated from extrinsic Vgs
  FFpre = 1.0/ ( 1.0 + eVgpre );  //Only used to compute the correction factor
  ab = 2 * ( 1 - 0.99 * FFpre ) * phit;
  Vcorr = ( 1.0 + 2.0 * delta ) * ( ab/ 2.0 ) * ( exp( -Vdsi/ ab ));  //Correction to intrinsic Vgs
Vgscorr = Vgsi + Vcorr;  //Intrinsic Vgs corrected (to be used for charge and current computation)
Vbscorr = Vbsi + Vcorr;  //Intrinsic Vgs corrected (to be used for charge and current computation)
Vt0bs = Vt0 + gamma * (sqrt( abs( phib - Vbscorr)) - sqrt( phib ));  //Computed from corrected
                                                                   //intrinsic Vbs
Vt0bs0 = Vt0 + gamma * (sqrt( abs( phib - Vbsi)) - sqrt( phib ));  //  Computed from uncorrected
                                                                    //intrinsic Vbs
```

```
Vtp = Vt0bs - Vdsi * delta - 0.5 * aphit;  //Computed from corrected intrinsic Vbs and intrinsic Vds
  Vtp0 = Vt0bs0 - Vdsi * delta - 0.5 * aphit;  //Computed from uncorrected intrinsic Vbs and
                                               //intrinsic Vds
eVg = exp(( Vgscorr - Vtp )/ ( aphit ));  //Compute eVg factor from corrected intrinsic Vgs
FF = 1.0/ ( 1.0 + eVg );
eVg0 = exp(( Vgsi - Vtp0 )/ ( aphit ));  //Compute eVg factor from uncorrected intrinsic Vgs
FF0 = 1.0/ ( 1.0 + eVg0 );
Qref = Cg * nphit;
eta = ( Vgscorr - ( Vt0bs - Vdsi * delta - FF * aphit ))/ ( nphit );  //Compute eta factor from
                                                                      //corrected intrinsic Vgs
//and intrinsic Vds
eta0 = ( Vgsi - ( Vt0bs0 - Vdsi * delta - FFpre * aphit ))/ ( nphit );  //Compute eta0 factor
                                                                        //from uncorrected intrinsic
//Vgs and internal Vds.

// Using FF instead of FF0 in eta0 gives smoother capacitances.

//Charge at VS in saturation (Qinv)
     if (eta  <= `LARGE_VALUE) begin
Qinv_corr   =  Qref * ln( 1.0 + exp(eta) );
end
else begin
Qinv_corr = Qref * eta;
end
if (eta0 <= `LARGE_VALUE) begin
Qinv = Qref * ln( 1.0 + exp(eta0) ); //  Compute charge w/ uncorrected intrinsic Vgs
                                     //for use later on in charge partitioning
end
else begin
Qinv = Qref * eta0;
end


//Transport equations
vx0 = vxo;
Vdsats = vx0 * Leff/ mu;
Vdsat = Vdsats * ( 1.0 - FF ) + phit * FF;  //  Saturation drain voltage for current
Vdratio = abs( Vdsi/ Vdsat);
Vdbeta = pow( Vdratio, beta);
Vdbetabeta  =  pow( 1.0 + Vdbeta, 1.0/ beta);
Fsat = Vdratio / Vdbetabeta; //  Transition function from linear to saturation.
// Fsat = 1 when Vds>>Vdsat; Fsat= Vds when Vds<<Vdsat

//Total drain current
Id = Qinv_corr * vx0 * Fsat * W;


//Calculation of intrinsic charge partitioning factors (qs and qd)
Vgt = Qinv/ Cg;  //  Use charge computed from uncorrected intrinsic Vgs

// Approximate solution for psis is weak inversion
if (gamma == 0) begin
a  =  1.0;
if (eta0 <= `LARGE_VALUE) begin
             psis    =  phib + phit * ( 1.0 + ln( ln( 1.0 + `SMALL_VALUE + exp( eta0 ))));
end
else begin
             psis    = phib + phit * ( 1.0 + ln( eta0 ));
         end
end
```

```
else begin
         if (eta0 <= `LARGE_VALUE) begin
            psis = phib+(1.0-gamma)/(1.0+gamma)*phit*(1.0+ln(ln(1.0+`SMALL_VALUE+exp(eta0))));
         end
        else begin
             psis = phib + ( 1.0 - gamma )/ ( 1.0 + gamma ) * phit * ( 1.0 + ln( eta0 ));
         end
         a = 1.0 + gamma/ ( 2.0 * sqrt( abs( psis - ( Vbsi ))));
     end
Vgta = Vgt/ a;  //  Vdsat in strong inversion
Vdsatq = sqrt( FF0 * aphit * aphit + Vgta * Vgta); //  Vdsat approx. to extend to weak inversion;

  //  The multiplier of phit has strong effect on Cgd discontinuity at Vd=0.

// Modified Fsat for calculation of charge partitioning
//DD-NVSAT charge

Fsatq = abs( Vdsi/ Vdsatq )/ ( pow( 1.0 + pow( abs( Vdsi/ Vdsatq ), beta ), 1.0/ beta ));
x = 1.0 - Fsatq;
den = 15 * ( 1 + x ) * ( 1 + x );
qsc  = Qinv *(6 + 12 * x + 8 * x * x + 4 * x * x * x)/ den;
qdc = Qinv *(4 + 8 * x + 12 * x * x + 6 * x * x * x)/ den;
qi = qsc + qdc;  //  Charge in the channel


//QB charge
kq  =  0.0;
tol =  ( `SMALL_VALUE * vxo/ 100.0 ) * ( `SMALL_VALUE * vxo/ 100.0 ) * me/ ( 2 * `P_Q );
if (Vdsi <= tol) begin
kq2 = ( 2.0 * `P_Q/ me * Vdsi )/ ( vx0 * vx0 ) * 10000.0;
kq4 = kq2 * kq2;
qsb = Qinv * ( 0.5 - kq2/ 24.0 + kq4/ 80.0 );
qdb = Qinv * ( 0.5 - 0.125 * kq2 + kq4/ 16.0 );
end
else begin
kq = sqrt( 2.0 * `P_Q/ me * Vdsi )/ vx0 * 100.0;
kq2 = kq * kq;
qsb = Qinv * ( asinh( kq )/ kq - ( sqrt( kq2 + 1.0 ) - 1.0 )/ kq2);
qdb = Qinv * (( sqrt( kq2 + 1.0 )- 1.0 )/ kq2);
end


// Flag for classic or ballistic charge partitioning:
if (CTM_select == 1) begin //Ballistic blended with classic DD-NVSAT
qs = qsc; //Calculation of "ballistic" channel charge partitioning factors,
                           //qsb and qdb.
qd = qdc; //Here it is assumed that the potential increases parabolically
                           //from the
end     //virtual source point, where Qinv_corr is known to Vds-dvd at
                             //the drain.
else begin    //Hence carrier velocity increases linearly by kq (below) depending
                             //on the

qs = qsc * ( 1 - Fsatq * Fsatq ) + qsb * Fsatq * Fsatq;  //efecive ballistic mass of the carriers.
qd = qdc * ( 1 - Fsatq * Fsatq ) + qdb * Fsatq * Fsatq;
end


//Body charge based on approximate surface potential (psis) calculation with delta=0 using psis=phib
//in Qb gives continuous Cgs, Cgd, Cdd in SI, while Cdd is smooth anyway.
```

```
Qb =-type*W*Leff*(Cg*gamma*sqrt(abs(psis-Vbsi))+(a-1.0)/(1.0*a)*Qinv*(1.0-qi));


//DIBL effect on drain charge calculation.
//Calculate dQinv at virtual source due to DIBL only. Then:Correct the qd factor to reflect
//this channel charge change due to Vd
//Vt0bs0 and FF=FF0 causes least discontinuity in Cgs and Cgd but produces a spike in Cdd at
//Vds=0 (in weak inversion.  But bad in strong inversion)

etai  = ( Vgsi - ( Vt0bs0 - FF * aphit ))/ ( nphit );
if (etai <= `LARGE_VALUE) begin
Qinvi = Qref * ln( 1.0 + exp( etai ));
end
else begin
Qinvi = Qref * etai;
end
dQinv = Qinv - Qinvi;
dibl_corr = ( 1.0 - FF0 ) * ( 1.0 - Fsatq ) * qi * dQinv;
qd = qd - dibl_corr;


//Inversion charge partitioning to terminals s and d
Qinvs = type * Leff * (( 1 + dir ) * qs + ( 1 - dir ) * qd)/ 2.0;
Qinvd = type * Leff * (( 1 - dir ) * qs + ( 1 + dir ) * qd)/ 2.0;


//Outer fringing capacitance
Qsov = Cofs * ( V(g) - V(si) );
Qdov = Cofd * ( V(g) - V(di) );


//Inner fringing capacitance
Vt0x = Vt0 + gamma * ( sqrt( abs( phib - type * ( V(b) - V(si) ))) - sqrt(phib));
Vt0y = Vt0 + gamma * ( sqrt( abs( phib - type * ( V(b) - V(di) ))) - sqrt(phib));
Fs_arg = ( Vgsraw - ( Vt0x - Vdsi * delta * Fsat ) + aphit * 0.5 )/ ( 1.1 * nphit );
if (Fs_arg <= `LARGE_VALUE) begin
Fs = 1.0 + exp( Fs_arg );
FFx = Vgsraw - nphit * ln( Fs );
end
else begin
Fs = 0.0;  // Not used
FFx = Vgsraw - nphit * Fs_arg;
end
Fd_arg = ( Vgdraw - ( Vt0y - Vdsi * delta * Fsat ) + aphit * 0.5 )/ ( 1.1 * nphit );
if (Fd_arg <= `LARGE_VALUE) begin
Fd = 1.0 + exp( Fd_arg );
FFy = Vgdraw - nphit * ln( Fd );
end
else begin
Fd = 0.0;  // Not used
FFy = Vgdraw - nphit * Fd_arg;
end
Qsif = type * ( Cif + CC * Vgsraw ) * FFx;
Qdif = type * ( Cif + CC * Vgdraw ) * FFy;


//Partitioned charge
Qs = -W * ( Qinvs + Qsov + Qsif );  //  s-terminal charge
Qd = -W * ( Qinvd + Qdov + Qdif );  //  d-terminal charge
```

```
Qg = -( Qs + Qd + Qb ); //  g-terminal charge


//Sub-circuit initialization
     I(di,si)  <+  type * dir * Id;
     I(d,di) <+  ( V(d) - V(di) )/ Rd;
     I(si,s)  <+  ( V(si) - V(s) )/ Rs;

I(si,b)  <+  ddt( Qs ); //  charge term: node si to node b
I(di,b)  <+  ddt( Qd ); //  charge term: node di to node b
I(g,b)  <+  ddt( Qg ); //  charge term: node g to node b


end
endmodule
```

II. DESCRIÇÃO DO CONTEÚDO DO CD