UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE ELETRICIDADE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
ÊNFASE: SISTEMAS ELETRÔNICOS

ELISABETE NAKONECZNY MORAES

IMPLEMENTAÇÃO DE UM MODELO DO TRANSISTOR MOS EM VERILOG-AMS

CURITBA

2008

UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE ELETRICIDADE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
ÊNFASE: SISTEMAS ELETRÔNICOS

ELISABETE NAKONECZNY MORAES

IMPLEMENTAÇÃO DE UM MODELO DO TRANSISTOR MOS EM VERILOG-AMS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre, no Programa de Pós-Graduação em Engenharia Elétrica com ênfase em Sistemas Eletrônicos, Setor de Tecnologia, Universidade Federal do Paraná – UFPR.

Orientador: Oscar da Costa Gouveia Filho, Dr.

Curitiba

2008

Moraes, Elisabete Nakoneczny
Implementação de um modelo do transistor MOS em VERILOG-AMS / Elisabete Nakoneczny Moraes. - Curitiba, 2008.
xx, 95 f.: il.; tab.; graf.

Dissertação (mestrado) – Universidade Federal do Paraná. Setor de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.
Orientador: Oscar da Costa Gouveia Filho.

1. Engenharia auxiliada por computador. 2. Hardware – Linguagens descritivas. 3. Transistores de efeito de campo de semicondutores de oxido metálico. I. Gouveia Filho, Oscar da Costa. II. Título.

CDD 22 621.392

IMPLEMENTAÇÃO DO MODELO ACM DO TRANSISTOR MOS USANDO VERILOG-AMS

ELISABETE NAKONECZNY MORAES

Dissertação aprovada como requisito parcial para a obtenção do grau de Mestre no Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Paraná.

Prof. Oscar da Costa Gouveia Filho, Dr.
Orientador

Prof. Wilson Arnaldo Artuzi Junior, Ph.D.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora

Prof. Oscar da Costa Gouveia Filho, Dr. (UFPR)
Presidente

Christophe Frederic Lucien Bricout, Dr. (AXP Microeletrônica)

Prof. Márlio José do Couto Bonfim, Dr. (UFPR)

Prof. Wilson Arnaldo Artuzi Junior, Ph.D. (UFPR)

Curitiba, 13 de junho de 2008

“Deus dá a todos uma estrela.

Uns fazem da estrela um sol.

Outros nem conseguem vê-la.”

Helena Kolody

DEDICATÓRIA

Aos meus **P**ais,

DINORA e **G**REGORIO

AGRADECIMENTOS

À DOLPHIN INTEGRATION, em particular ao Sr. GILLES DEPEYROT pela concessão da licença do Simulador SMASH, tornando possível a concretização deste trabalho.

À POWER INFORMÁTICA, representada por WELLINGTON OUCHI, que colaborou de forma decisiva para proporcionar condições para que eu pudesse concluir este trabalho.

Aos engenheiros de suporte da MENTOR GRAPHICS, pelo cordial e eficiente atendimento nas questões relacionadas ao desenvolvimento do código.

Ao professor OSCAR DA COSTA GOUVEIA FILHO pela sua paciência e competência.

À RAQUEL, secretária do programa, pela sua prestatividade.

À doce e sempre professora FAIMARA, compartilhando sua experiência, competência e sabedoria.

Ao colega de turma e de trabalho VALFREDO, por não poupar esforços em me ajudar.

À CLARICE e EDILENE por estarem presentes nos momentos significativos da minha família.

À SUZETE, querida irmã, guiando a minha caminhada.

À CACILDA que exerceu com competência e brilhantismo as responsabilidades lhe conferidas.

Ao JULIMAR, adorável marido, pela sua sensibilidade e sabedoria.

À RHARA, minha preciosa filha.

À DEUS, pelo dom da vida.

SUMÁRIO

LISTA DE SIGLAS E ACRÔNIMOS

SIGLA	DEFINIÇÃO	SIGNIFICADO
AC	*Alternative (Alternating, Alternate) Current*	Corrente Alternada
ACM	*Advanced Compact Mosfet*	Modelo Compacto Avançado do Mosfet
AMS	*Analog Mixed Signal*	Sinais Analógicos e Mistos
BSIM	*Berkeley Short Channel IGFET Model*	Modelo de IGFET de Berkeley de Canal Curto
CAD	*Computer Aided Design*	Projeto Orientado por Computador
CAE	*Computer Aided Engineering*	Engenharia Orientada por Computador
CAM	*Compute Aided Manufacturing*	Manufatura Orientada por Computador
CI	---	Circuito Integrado
CPU	*Central Processing Unit*	Unidade Central de Processamento
DIBL	*Drain Induced Barrier Lowering*	Redução de Barreira Induzida pelo Dreno
DC	*Direct Current*	Corrente Direta
EECS	*Electrical Engineering Computer Science*	Departamento de Engenharia Elétrica e da Ciência da Computação
EDA	*Electronic Design Automation*	Automação de Projetos Eletrônicos
FORTRAN	*Formula Translation (Translator)*	Tradução de Fórmulas Tradutor de Fórmulas
HDL	*Hardware Description Language*	Linguagem de Descrição de *Hardware*
IEEE	*Institute of Electrical and Electronics Engineers*	Instituto de Engenheiros Elétricos e Eletrônicos
IGFET	*Insulating Gate Field Effect Transistor*	Transistor de Efeito de Campo de Porta Isolada
KCL	*Kirchhoff Current Law*	Lei de Kirchhoff das Correntes
KVL	*Kirchhoff Voltage Law*	Lei de Kirchhoff das Tensões

SIGLA	DEFINIÇÃO	SIGNIFICADO
LRM	*Language Reference Manual*	Manual de Referência da Linguagem
SPICE	*Simulated Program with Integrated Circuits Emphasis*	Programa de Simulação com ênfase em Circuitos Integrados
UCCM	*Unified Charge Control Model*	Modelo Unificado por Controle de Carga
VHDL	*VHSIC Hardware Description Language*	Linguagem de Descrição de *Hardware* tipo *VHSIC*
VHSIC	*Very High-Speed Integrated Circuit*	Circuitos Integrados de Velocidade Muito Alta

LISTA DE SÍMBOLOS

g	coeficiente de atenuação da função da onda do elétron no óxido
l	parâmetro de ajuste para modulação do comprimento do canal
s	efeito DIBL
q	parâmetro de entrada para ajuste da mobilidade
e_{Si}	permissividade elétrica do silício
μ_0	mobilidade dos portadores para o campo elétrico longitudinal transversal desprezíveis
μ_{ef}	mobilidade efetiva com o campo elétrico transversal
$Ø_S$	potencial de superfície
$Æ_{S0}$	potencial de superfície no equilíbrio
$Ø_t$	potencial termodinâmico
DL	comprimento da porção saturada do canal
C'_{OX}	capacitância do óxido por unidade de área
C_{bd}	capacitância substrato-dreno
C_{bg}	capacitância substrato-porta
C_{bs}	capacitância substrato-fonte
C_{db}	capacitância dreno-substrato
C_{dd}	capacitância de dreno
C_{dg}	transcapacitância dreno-porta
C_{ds}	transcapacitância dreno-fonte
C_{gb}	capacitância porta-substrato
C_{gd}	capacitância porta-dreno
C_{gs}	capacitância porta-fonte
g_{mB}	transcondutância de substrato
g_{mD}	transcondutância de dreno
g_{mG}	transcondutância de porta
g_{mS}	transcondutância de fonte
i	intensidade de corrente elétrica
I_D	corrente de dreno
I_F	corrente de saturação direta
I_R	corrente de saturação reversa

f	largura da banda de freqüência
k_B	constante de Boltzmann
L	comprimento nominal do canal
L_{eff}	comprimento efetivo do canal
n	fator de rampa
N_{ot}	densidade equivalente das trilhas de óxido
$N_t(E)$	densidade do óxido das trilhas por unidade de volume e energia
N_P	quantidade de transistores em paralelo
N_S	quantidade de transistores em série
q_e	carga elétrica do elétron
Q'_B	densidade de carga de depleção
Q'_I	densidade de carga de inversão
Q'_{ID}	densidade de carga de inversão no dreno
Q'_{IP}	densidade de carga de inversão na condição *pinch-off*
Q'_{IS}	densidade de carga de inversão na fonte
Q_B	carga total de depleção
Q_D	carga do dreno
Q_G	carga da porta
Q_I	carga total de inversão
Siw	densidade espectral de potência térmica
Sid	densidade espectral de potência
Si_{total}	densidade espectral de potência total
t	tempo
T	temperatura absoluta
u	tensão normalizada
$UCRIT$	campo elétrico crítico
v_{lim}	velocidade de saturação dos portadores
V_B.	tensão no terminal de substrato
V_C	tensão no canal
V_{CB}	tensão no canal referida ao substrato
V_{DB}	tensão dreno-substrato
V_{GB}	tensão porta-substrato

V_P	tensão de *pinch-off*
V_{P0}	tensão de *pinch-off* com $V_{DS} = 0$
V_{SB}	tensão fonte-substrato
V_{th0}	tensão de limiar no equilíbrio
x	coordenada na direção do comprimento do canal.
x_j	profundidade da junção
W	largura nominal do canal
W_{eff}	largura efetiva do canal

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A indústria de semicondutores a partir de 1960 passou a utilizar simuladores que são ferramentas computacionais que auxiliam o desenvolvimento de projetos eletrônicos permitindo a integração de todas as fases que envolvem a construção de um circuito integrado. A essência destas ferramentas fundamenta-se em bibliotecas de modelos de componentes eletrônicos adequadamente estruturados para se adaptarem às especificações, portanto a atualização destas bibliotecas deve ocorrer na mesma velocidade do processamento de semicondutores, uma vez que a cada geração, os modelos precisam ser adequados pela introdução de novos parâmetros ou nas equações que os descrevem. O uso da linguagem C ou FORTRAN para descrever os códigos dos modelos não permite a dinâmica requerida pela indústria de manufatura de circuitos integrados, de forma que o uso de linguagens padronizadas como o Verilog-AMS são essenciais para no desenvolvimento preciso de sistemas e componentes eletrônicos. O conteúdo desta dissertação apresenta o uso da linguagem Verilog-AMS na implementação do código fonte do modelo ACM do transistor MOS que foi originalmente desenvolvido em linguagem C, apresentando como características a conservação das cargas, equações simples e contínuas, número reduzido de parâmetros, dedicado para uso em simuladores. O processo de implementação do código em Verilog-AMS consiste em verificar a portabilidade do código desenvolvido pelo uso dos simuladores ELDO e SMASH e a realização de testes de validação, composto pela análise DC, AC, transiente e de ruído e a simulação de circuitos clássicos objetivando a comparação dos resultados com o código original em C.

Palavras chaves: Verilog-AMS, MOSFET, simulação, modelo compacto.

ABSTRACT

The industry of semiconductors from 1960 started to use simulators that are computational tools that assist the development of electronic projects allowing the integration of all the phases that involve the construction of an integrated circuit. The essence of these tools is based on libraries of models of electronic components adequately structuralized to attend the specifications, therefore the update of these libraries must occur in the same speed of the processing of semiconductors, at each new generation, the models need to be adjusted by the introduction of new parameters or changes in the equations that describe them. The use of language C or FORTRAN to describe the codes of the models does not allow the dynamics required by integrated circuits industry, in this case the use of standardized languages such as: Verilog-AMS is essential to the development of systems and electronic components. The content of this master thesis presents the use of the Verilog-AMS language in the implementation of the source code of ACM model transistor that was originally developed in C language. Its characteristics are resumed in loads conservation, simple and continuous equations, reduced number of parameters, and it is dedicated for use in simulators. The process of implementation of the code in Verilog-AMS consists of verifying the portability of the developed code, using ELDO and SMASH simulators and the accomplishment of validation tests is composed for DC, AC, transient and noise analysis and the simulation of classic circuits objectifying the comparison of the results with the original C code.

Key words: Verilog-AMS, MOSFET, simulation, compact model.

1 INTRODUÇÃO

1.1 HISTÓRICO

A crescente expansão do mercado dos segmentos de áudio, vídeo, comunicação sem fio, de novos produtos e de novas categorias de produtos faz com que a indústria micro-eletrônica esteja diante do seguinte desafio: a necessidade crescente de melhorar a exata descrição dos circuitos eletrônicos a fim de que possam ser simulados muito antes de se tornarem *chips* integrantes dos mais variados aparelhos, em espaços de tempo cada vez menores.

A simulação dos circuitos eletrônicos na fase do projeto deve encontrar especificações que convergem para o menor custo, menor tempo entre a concepção e o aparelho final, menor taxa de falhas, alta qualidade e confiabilidade, enfim, o melhor arranjo entre as variáveis, implicará em um produto de ponta, satisfação do mercado com retorno garantido para o fabricante.

Este conjunto de necessidades relacionadas à fabricação de dispositivos eletrônicos incentivou o desenvolvimento de ferramentas computacionais direcionadas ao projeto, desenvolvimento e à simulação de componentes eletrônicos capazes de gerenciar e integrar as fases de concepção, especificação, estruturação, verificação e fabricação de um circuito integrado (CI).

Já no final dos anos 1950, ferramentas computacionais começaram a serem usadas no suporte de projetos, mas foi entre as décadas de 1960 e 1970 que diversos simuladores foram

desenvolvidos: CIRCAL, SCEPTRE[1], ECAP, ASCAP, SPICE. Rapidamente as técnicas de simulação evoluíram para acompanhar o aumento da complexidade de circuitos crescendo em capacidade e popularidade, sendo o SPICE o mais popular entre eles [1].

Foi em 1972 que o Departamento de Engenharia Elétrica e da Ciência da Computação (EECS) da Universidade da Califórnia, Berkeley, desenvolveu a primeira versão do programa SPICE (*Simulation Program with Integrated Circuit Emphasis*). O SPICE foi escrito originalmente em FORTRAN e foi atualizado para o SPICE2, em 1975, e a versão atual, SPICE3 de 1985, escrita em linguagem C.

Os simuladores empregam técnicas matemáticas em computadores com o propósito de emular um processo ou operação do mundo real. Para tanto é necessário dispor de um modelo computacional que concilie a realidade às limitações do ambiente computacional.

Em 1990, a linguagem de descrição de *hardware* (HDL), passou a ser a base para a completa automação dos processos de fabricação do CI, capaz de formalizar a descrição de sistemas eletrônicos e integrar as informações pertinentes ao comportamento, à estrutura das conexões e sobre as características geométricas, dando origem ao diagrama de Gajski-Khan ou Y, indicado na Figura 1.1, no qual os eixos comportamental, estrutural e físico ou geométrico de um projeto eletrônico se estruturam [1].

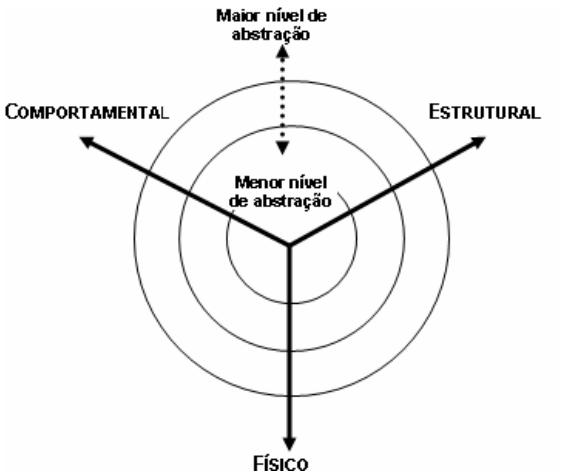


Figura 1.1 - Diagrama de Gajski-Khan

[1] SCEPTRE (*System for Circuit Evaluation and Predicition of Transient Radiation Effects*) é um programa de análise de circuitos genérico, com análise DC, AC e Trans, para redes lineares ou não lineares. Mais informações em: http://www.psych.mcgill.ca/perpg/stds/rk/linux-announce.html#toc-89 e o programa e documentação em ftp://novilux.fh-friedberg.de/pub/sceptre_linux/

A integração dos eixos por meio dos simuladores deu origem a uma ramificação na área da engenharia elétrica denominada de Automação de Projetos Eletrônicos (EDA) englobando a engenharia orientada por computador (CAE), projeto orientado por computador (CAD) e manufatura orientada por computador (CAM).

1.2 MOTIVAÇÃO

A base de funcionamento dos simuladores consiste na existência de bibliotecas de modelos computacionais de componentes eletrônicos e elétricos capazes de adaptar a realidade do comportamento à forma de processamento do simulador.

Os modelos computacionais atualmente disponíveis atendem a diversos propósitos, como por exemplo: para uso único em simulações, extração de informações ou leiaute do sistema. Estes modelos utilizam métodos científicos e matemáticos para descrever o comportamento real do componente de forma mais ou menos detalhada, dando origem a um conjunto extenso de modelos que precisam ser compatibilizados à arquitetura computacional do simulador.

Em geral as pessoas que desenvolvem os modelos não são as mesmas que desenvolvem os simuladores de circuitos. Os primeiros estão ligados principalmente a universidades e a empresas de projeto de circuitos integrados, já os últimos trabalham nas empresas fornecedoras de ferramentas de automação de projetos (EDA) [3].

A linguagem padrão para a implementação de modelos compactos em simuladores de circuitos tem sido C, desde o lançamento do SPICE3, em 1985 [2] e a implementação de modelos em C requer o conhecimento do código interno do simulador de circuitos, o que nem sempre está à disposição do desenvolvedor do modelo, por outro lado nem sempre os fabricantes de simuladores estão interessados na inclusão de novos modelos em seus produtos. Como conseqüência, nem todos os modelos estão instalados em todos os simuladores. Além disso, a correção de erros e a atualização dos modelos para atender avanços da tecnologia demoram a ser implementadas devido à complexidade do processo.

A Figura 1.2 ilustra o exposto no parágrafo anterior: a grande diversidade de modelos de transistores MOS utilizados em simuladores como o ACM, BSIM, EKV, MM11,

PSP que precisam ser codificados para cada uma das bibliotecas dos vários simuladores comerciais existentes, entre eles: ADS, AMS, ELDO, HSPICE, SMASH, SPECTRE.

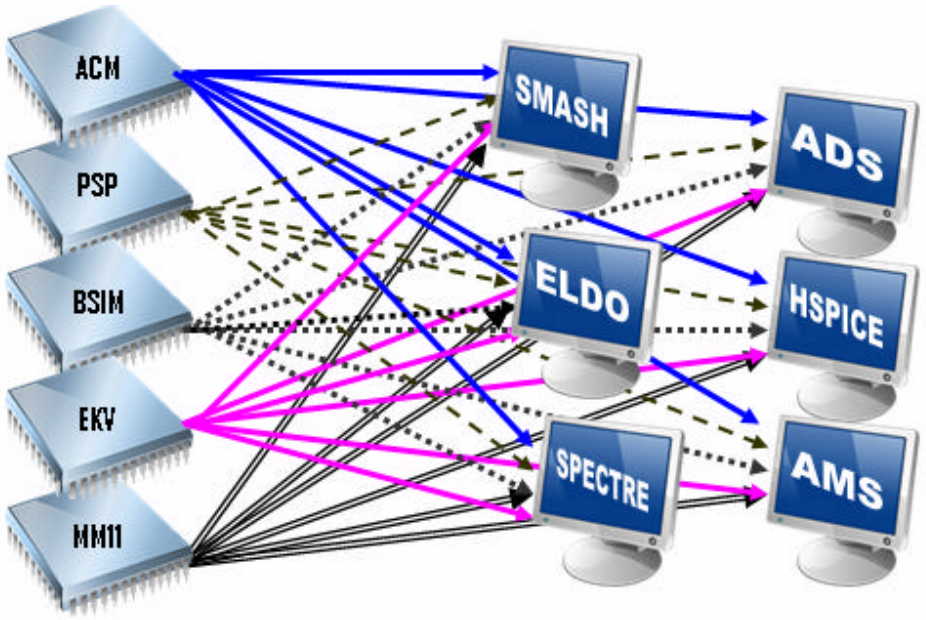


Figura 1.2 - Representação gráfica dos diferentes modelos de transistores adaptados para os diversos simuladores comerciais. Fonte: [27]

Portanto a padronização na linguagem usada para o desenvolvimento dos modelos compactos determina procedimentos e estruturas uniformizadas, que podem ser usadas em qualquer simulador comercial que suporte os padrões da linguagem em seu processador.

Esta padronização foi viabilizada por meio da linguagem de descrição de *hardware* que são específicas para descrição de estruturas e do comportamento do *hardware*, sendo exemplos de HDL's o Verilog-AMS e o VHDL-AMS.

No caso específico da linguagem Verilog-AMS, que é o objeto de estudo desta dissertação, ela proporciona uma interface única, como ilustra a Figura 1.3, entre os vários modelos de transistores e os simuladores comerciais, dispensando a codificação individualizada de cada modelo.

Figura 1.3 - Função que a linguagem Verilog-AMS exerce no contexto entre os vários modelos de transistores e simuladores comerciais.

1.3 OBJETIVOS

Tendo em vista as vantagens da linguagem Verilog-AMS e que a organização *Compact Model Council* (CMC) [2] incentiva o seu uso, o presente trabalho tem como objetivos:

- Implementar o código *Advanced Compact Mosfet Model* (ACM) do transistor MOS para a linguagem de descrição de *hardware* (HDL), utilizando a linguagem Verilog-AMS.
- Realizar testes de simulação para comprovar o funcionamento do código implementado.
- Verificar a portabilidade do código implementado utilizando os simuladores ELDO [6] e SMASH [7].

1.4 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está organizada em cinco capítulos. O Capítulo 2 descreve o modelo ACM do transistor MOS. No Capítulo 3 apresentam-se os fundamentos da linguagem de descrição de *hardware*, incluindo o Verilog-AMS e a metodologia adotada na implementação do código. No Capítulo 4 relatam-se os resultados obtidos, e as conclusões do trabalho são expostas no Capítulo 5.

2 DESCRIÇÃO DO MODELO ACM

2.1 MODELO COMPACTO DO TRANSISTOR

Modelos compactos descrevem o comportamento dos transistores por meio um modelo abstrato, conceitual, gráfico e/ou matemático utilizando técnicas, métodos e teoria fundamentada na física dos semicondutores e formam a base dos simuladores de circuitos eletrônicos.

Os modelos de componentes eletrônicos dedicados aos programas de simulação são recomendados pela organização denominada *Compact Model Council* (CMC) [8] que define, mantém e promove o uso de modelos padronizados permitindo com que as empresas do ramo de semicondutores sejam norteadas pela mesma base tecnológica. Segundo [2], a entidade incentiva o uso do Verilog-A à linguagem C para implementar os modelos compactos.

As simulações de circuitos eletrônicos são estruturadas em bibliotecas de dispositivos semicondutores que são desenvolvidas pela modelagem científica que é o processo de gerar:

Modelo abstrato: É uma construção teórica que representa o objeto em estudo usando um conjunto de variáveis com parâmetros interligados. Neste nível, o modelo representa as ações internas idealizadas pela estrutura de programação.

Modelo conceitual: É a representação de um fenômeno, dado ou teoria usando a lógica e a matemática por meio de funções, relações, tabelas, processos estocásticos, fórmulas, axiomas e regras de inferência.

Modelo gráfico: Desenhos, gráficos ou fluxogramas que representam a independência ou dependência entre as variáveis envolvidas.

Modelo matemático: É a representação de um sistema por meio de equações matemáticas.

Para os transistores, destacam-se dois grupos de modelos:

Modelo para o projeto do componente: Os modelos que se destinam ao projeto do componente único exploram a estrutura física, considerando os aspectos geométricos, mas também conhecimentos sobre a natureza do material semicondutor, sua pureza, elementos dopantes, entre outros.

Modelo para o projeto de circuitos eletrônicos: As ferramentas para projetos eletrônicos automatizados (*EDA tools*) usam modelos dos componentes para emular o funcionamento de um circuito, sendo necessário modelar nestas circunstâncias os efeitos do leiaute do transistor, como: comprimento e largura do canal, características de alimentação, capacitância parasita, indutância e resistência interna e efeitos da temperatura.

2.1.1 Modelos de Transistores para Simuladores Eletrônicos

Dependendo da aplicação do transistor, se para simulação e fabricação, ou só simulação, para sinal analógico ou digital, nível da tensão de trabalho, existirá um ou mais modelos que se adaptam melhor a essas necessidades como, por exemplo: o modelo SPICE *level*1 é um modelo ideal, aplicável somente para simulações e modelagens ideais, não poderá ser usado com propósito comercial ou de projeto. As mesmas características são aplicáveis para os modelos SPICE *level2* e *leve13*, que se diferenciam na quantidade de parâmetros que os definem.

Os modelos BSIM3, BSIM4, EKV, PSP são os mais conhecidos. O BSIM3 e BSIM4 [9] incluem grande quantidade de informação dos parâmetros do processo, e detalhes dos efeitos construtivos presentes na fabricação do CI, portanto assemelham-se com um transistor real, o EKV [10] é apropriado para projetos analógicos com tensões de operação abaixo da tensão de limiar e o modelo PSP, além de reunir as características da semelhança com o

transistor real dispõem de maiores detalhamentos sobre suas propriedades elétricas que podem ser melhor esclarecidas em [11] e [12].

O modelo ACM [19] do transistor MOS, dentro do contexto dos modelos compactos é adequado para uso em simuladores comerciais, pois requer uma quantidade reduzida de parâmetros de entrada, as equações matemáticas são capazes de descrever o transistor operando em qualquer região, seja na saturação, inversão fraca ou moderada.

2.2 MODELO COMPACTO ACM

O modelo ACM é caracterizado como um modelo físico compacto, que considera a conservação da carga, adequado para integrar as bibliotecas dos simuladores eletrônicos. A conservação da carga e a simetria fonte-dreno intrínseca do dispositivo que usa o substrato como referência, são propriedades essenciais do modelo, que inclui expressões únicas, contínuas e simples, que descrevem com boa precisão todas as características estáticas e dinâmicas do MOSFET desde inversão fraca até inversão forte.

Os efeitos de canal curto (sessão 2.2.5) são incluídos no modelo através de modificações no modelo de canal longo resultando em equações simplificadas capazes de representar com precisão tais efeitos. A região de saturação é adequadamente modelada permitindo a obtenção de um modelo único e consistente para todas as regiões de operação. O modelo para a condutância de saída e para as cargas e (trans)capacitâncias inclui os efeitos de saturação da velocidade dos portadores. A Figura 2.1 a seguir ilustra o transistor MOS em perspectiva com os terminais e dimensões adotadas para o desenvolvimento das equações do modelo ACM.

Para o desenvolvimento do modelo, o terminal do substrato é considerado como referência para os potenciais do transistor.

A caracterização dos parâmetros de entrada para a descrição do modelo ACM são apresentados na Tabela 2.1.

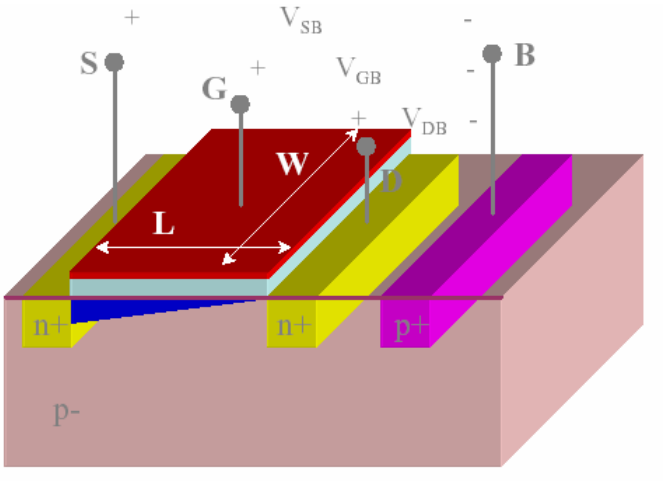


Figura 2.1 - Construção física do transistor MOS. Fonte: http://www.eletr.ufpr.br/ogouveia/te823/aulas.html aula 2.

Tabela 2.1 Parâmetros de entrada do modelo ACM.

NOME	SÍMBOLO	DESCRIÇÃO	UNIDADE	*DEFAULT*	
				NMOS	PMOS
μ_0	μ_0	Mobilidade dos portadores na condição de campo transversal e longitudinal desprezíveis.(*carrier mobility*)	m^2 /Vs	436.8×10^{-4}	149.4×10^{-4}
t_{OX}	t_{OX}	Espessura do óxido	m	7.7×10^{-9}	7.7×10^{-9}
vto	v_{to}	Tensão de limiar	V	0.486	-0.719
gamma	g	Coeficiente de efeito de corpo	sqrt(V)	0.77	0.375
phi	$\varnothing_S$	Potencial de superfície	V	0.862	0.81
v_{lim}	v_{lim}	Velocidade deslocamento dos portadores	m/s	$1.613 \times 10^{+5}$	$1.169 \times 10^{+5}$
theta	q	Coeficiente de redução da mobilidade devida ao campo elétrico transversal	V^{-1}	0.0593	0.0509
xj	x_j	Profundidade da junção	m	1×10^{-7}	1×10^{-7}
sigma	s	Coeficiente para DIBL	m^2	268.8×10^{-15}	1.04×10^{-15}

Nas próximas sessões, as principais equações do modelo ACM serão apresentadas tendo como referência: [19],[20] e [23].

2.2.1 Fundamentação

Os princípios físicos que fundamentam as equações do modelo ACM estão baseados no modelo da aproximação de folha carga[2] e na aproximação linear de canal gradual[3] que permite relacionar o potencial de superfície $\emptyset_S$ e a densidade de carga de inversão Q'_I, aplicada a uma porção da camada de inversão.

A corrente de dreno em um modelo de canal longo, supondo que a dopagem do substrato é uniforme, incluindo as componentes de deriva e difusão e mobilidade dos portadores como constante resulta em:

$$I_D = m_o W\left(-Q_I^{'}\frac{df_S}{dx} + f_t\frac{dQ_I^{'}}{dx}\right) \qquad (2.1)$$

A relação entre o potencial de superfície $\emptyset_S$ e a densidade de carga de inversão Q'_I é expressa por:

$$dQ_I^{'} = nC_{OX}^{'}df_S \qquad (2.2)$$

onde: μ_0, W, $\emptyset_t$, n, C'_{OX} e x indicam respectivamente: mobilidade dos portadores para o campo elétrico longitudinal e transversal desprezíveis, largura do canal, potencial termodinâmico, fator de rampa, capacitância do óxido por unidade de área e a coordenada na direção do comprimento do canal. As variáveis relacionadas à geometria do componente podem ser visualizadas na Figura 2.2.

[2] A espessura do canal é infinitesimal.
[3] O gradiente de potencial na direção da fonte para o dreno é muito maior que o gradiente de porta para substrato.

2.2.2 Cálculo da Corrente de Dreno

A partir da substituição de (2.2) em (2.1) e integrando-se ao longo do canal chega-se ao valor da corrente de dreno:

$$I_D = I_F - I_R \tag{2.3}$$

Considerações e o desenvolvimento estão apresentados em [19] de onde obtém-se a equação final da corrente de dreno em função das cargas de inversão:

$$I_{F(R)} = \mu_0 n C'_{OX} \frac{W}{L} \frac{f_t^2}{2} \left[\left(\frac{Q'_{IS(D)}}{n C'_{OX} f_t} \right)^2 - \frac{2 Q'_{IS(D)}}{n C'_{OX} f_t} \right] \tag{2.4}$$

onde: $I_{F(R)}$, L, $Q'_{IS(D)}$ e Q'_{IP} são respectivamente: corrente de saturação direta (reversa), comprimento do canal, densidade de carga de inversão na fonte (dreno) e densidade de carga de inversão na condição *pinch-off* .

2.2.3 Relação entre a Tensão no Canal e a Carga de Inversão

O uso da modelagem unificada por controle de carga (*Unified Charge Control Model* - UCCM) resulta na expressão que relaciona a tensão no canal e a densidade de carga de inversão:

$$V_P - V_C = f_t \left[\frac{Q'_{IP} - Q'_I}{n . C'_{OX} . f_t} + \ln \left(\frac{Q'_I}{Q'_{IP}} \right) \right] \tag{2.5}$$

onde: V_P é a tensão de *pinch-off*, V_C é a tensão no canal, Q'_{IP} é densidade da carga de inversão na condição *pinch-off* e ϕt é o potencial termodinâmico. Todas as tensões são referenciadas ao substrato.

Porém (2.5) não pode ser resolvida analiticamente para Q'_I, sendo aproximada pelo seguinte conjunto de expressões:

$$q = \ln\left[1 + \frac{e^{u-1}}{1 + k(u).\ln(1 + e^{u-1})}\right] \tag{2.6}$$

com

$$k(u) = 1 - \frac{84.4839}{u^2 + 150.8640} \tag{2.7}$$

onde

$$u = \frac{V_P - V_{S(D)}}{f_t} \tag{2.8}$$

e

$$q = -\frac{Q'_{IS(D)}}{n.C'_{OX}.f_t} \tag{2.9}$$

2.2.4 Cálculo das Cargas Totais, Capacitâncias e Transcondutâncias

O desenvolvimento a seguir integra o conjunto formal de equações do código ACM, porém na implementação do modelo na linguagem Verilog-AMS, o cálculo explícito das capacitâncias e transcondutâncias não foram necessários, pois a linguagem Verilog-AMS ao realizar a integração numérica utiliza como variável de integração o tempo. Esta particularidade é usada em conjunto com a função operador derivação que dispensam a efetiva realização dos cálculos.

Portanto os cálculos que envolvem a operação derivação podem ser realizados sem o cálculo explícito das grandezas elétricas envolvidas na operação. A exemplificação do que foi exposto no parágrafo anterior está detalhada no item 3.3 desta dissertação.

Como a estrutura física do transistor MOS é composta por quatro terminais: dreno (D), porta (G), fonte (S) e substrato (B) e a fim de que o modelo conserve a carga, faz-se necessário realizar o equacionamento das cargas elétricas em todos os terminais, assim a

Figura 2.2 auxilia o entendimento da disposição física dos terminais e das cargas elétricas no transistor tipo NMOS.

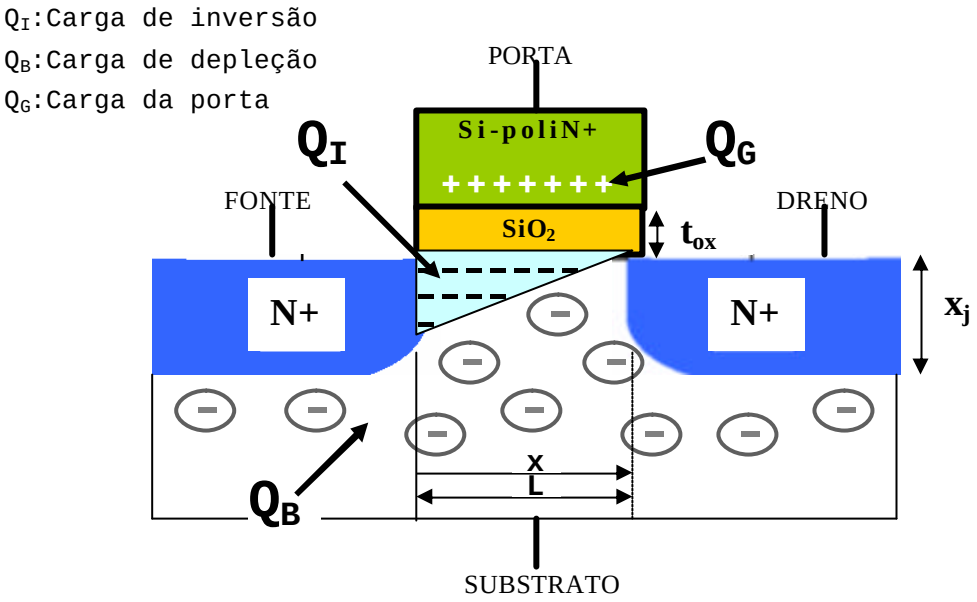


Figura 2.2 - Perfil do transistor NMOS, indicando os tipos de cargas. Q_I:carga de inversão, Q_B:carga de depleção e Q_G: carga da porta

Com base na Figura 2.2 as cargas de inversão e de depleção respectivamente, são expressas por:

$$Q_I = W\int_0^L Q'_I .dx \tag{2.10}$$

$$Q_B = W\int_0^L Q'_B .dx \tag{2.11}$$

Para um modelo que respeita a conservação da carga:

$$Q_S + Q_D = Q_I \tag{2.12}$$

Sendo Q_S a carga da fonte, Q_D a carga do dreno e Q_I a carga de inversão, Q_B a carga de depleção. Q'_I e Q'_B são respectivamente as cargas de inversão e depleção por unidade de área.

A carga do dreno e da fonte é definida em função do comprimento do canal L e da coordenada na direção do comprimento do canal x, conforme indica as equações (2.13) e (2.14).

$$Q_D = W\int_0^L \frac{x}{L}.Q'_I.dx \tag{2.13}$$

$$Q_S = W\int_0^L \left(1-\frac{x}{L}\right)Q'_I.dx \tag{2.14}$$

A partir da solução das integrais (2.10), (2.11), (2.13) e (2.14) chega-se às expressões das cargas nos terminais para o transistor que são apresentadas na Tabela 2.2.

As capacitâncias intrínsecas são calculadas tendo [17] como referência e são obtidas de acordo com a formação indicada a seguir:

$$C_{KJ} = -\frac{\partial Q_K}{\partial V_J} \tag{2.15}$$

$$C_{JJ} = \frac{\partial Q_J}{\partial V_J} \tag{2.16}$$

onde: Q_K e Q_J são quaisquer das cargas Q_G, Q_B, Q_D ou Q_S e V_J é qualquer potencial V_{GB}, V_{SB} V_{DB} ou V_B. As expressões (2.15) e (2.16) dão origem a um total de dezesseis (trans)capacitâncias das quais nove são usadas e estão apresentadas na Tabela 2.2.

Tabela 2.2 Cargas e (trans)capacitâncias para o MOSFET canal longo

VARIÁVEL	EXPRESSÃO
Q_I	$WL\left[\frac{2}{3}\frac{Q_F^{'2}+Q_F^{'}Q_R^{'}+Q_R^{'2}}{Q_F^{'}+Q_R^{'}}\right]+nC_{OX}^{'}.f_t$
Q_B	$-\frac{n-1}{n}Q_I-WL\frac{g^2C_{OX}^{'}}{2(n-1)}$
Q_D	$WL\left[\frac{6Q_R^{'3}+12Q_F^{'}Q_R^{'2}+8Q_F^{'2}Q_R^{'}+4Q_F^{'3}}{15\left(Q_F^{'}+Q_R^{'}\right)^2}+\frac{nC_{OX}^{'}f_t}{2}\right]$
Q_S	Q_I-Q_D
C_{gs}	$\frac{2}{3}WLC_{OX}^{'}\left[1-\frac{Q_R^{'2}}{\left(Q_F^{'}+Q_R^{'}\right)^2}\right]\left(1+\frac{nC_{OX}^{'}f_t}{Q_F^{'}}\right)$
C_{gd}	$\frac{2}{3}WLC_{OX}^{'}\left[1-\frac{Q_F^{'2}}{\left(Q_F^{'}+Q_R^{'}\right)^2}\right]\left(1+\frac{nC_{OX}^{'}f_t}{Q_R^{'}}\right)$
C_{gb}	$\frac{n-1}{n}\left(C_{OX}-C_{gs}-C_{gd}\right)$
C_{dd}	$\frac{2}{15}WLnC_{OX}^{'}\left[\frac{3Q_R^{'3}+9Q_F^{'}Q_R^{'2}+8Q_F^{'2}Q_R^{'}}{\left(Q_F^{'}+Q_R^{'}\right)^3}\right]\left(1+\frac{nC_{OX}^{'}f_t}{Q_R^{'}}\right)$
C_{ds}	$-\frac{4}{15}WLnC_{OX}^{'}\left[\frac{Q_F^{'3}+3Q_R^{'}Q_F^{'2}+Q_R^{'2}Q_F^{'}}{\left(Q_F^{'}+Q_R^{'}\right)^3}\right]\left(1+\frac{nC_{OX}^{'}f_t}{Q_F^{'}}\right)$
C_{dg}	$\frac{\left(C_{dd}-C_{ds}\right)}{n}$
C_{db}	$(n-1)\frac{\left(C_{dd}-C_{ds}\right)}{n}$
C_{bs}	$(n-1)C_{gs}$
C_{bd}	$(n-1)C_{gd}$

onde: $Q_F=Q_{IS}^{'}-nC_{OX}^{'}f_t$

$Q_R=Q_{ID}^{'}-nC_{OX}^{'}f_t$

As transcondutâncias, que são as derivadas da corrente em relação às respectivas tensões, indicadas pela conjunto de expressões (2.17), são contínuas e permitem uma rápida convergência na análise DC, caso seja usada uma linguagem de programação diferente do Verilog-AMS, uma vez que as referidas equações não precisaram ser implementadas no código e são equacionadas considerando todos os parâmetros que representam a variação da corrente em função das tensões nos terminais do transistor.

$$g_{mS} = -\frac{\partial i_D}{\partial V_S} \qquad g_{mG} = -\frac{\partial i_D}{\partial V_G} \qquad g_{mD} = -\frac{\partial i_D}{\partial V_D} \qquad g_{mB} = -\frac{\partial i_D}{\partial V_B} \tag{2.17}$$

onde: g_{mS}, g_{mG}, g_{mD} e g_{mB} são as transcondutâncias de fonte, porta, dreno e substrato respectivamente.

2.2.5 Efeitos de Segunda Ordem

Também denominados como efeitos de canal curto e ocorrem quando as dimensões do canal se aproximam de seus valores mínimos[4], surgem novos efeitos que modificam as características estáticas e dinâmicas do transistor. Os efeitos de segunda ordem que são considerados na modelagem do código ACM são: saturação da velocidade, a modulação do comprimento do canal (*channel length modulation* - CLM), redução de barreira induzida pelo dreno (*drain induced barrier lowering* – DIBL).

Os detalhes sobre os efeitos são encontrados em [19] e [24].

[4] Comprimento do canal for menor que aproximadamente 2µm.

2.2.5.1 Modulação do comprimento do canal

A modulação do comprimento do canal engloba genericamente as contribuições do gradiente do campo elétrico longitudinal, sobretudo nas vizinhanças do dreno, este efeito é modelado de forma unidimensional por meio da redução do comprimento efetivo do canal em função do aumento do potencial do dreno, devido ao avanço da região de depleção.

$$\Delta L = l\ L_C.\ln\left[1+\frac{(V_{DS}-V'_{DS})}{L_C.UCRIT}\right] \tag{2.18}$$

onde: V'_{DS} é dado por uma função contínua tal que $V'_{DS} = V_{DS}$ para $V_{DS} < V_{DSSAT}$ e $V'_{DS} = V_{DSSAT}$ para $V_{DS} > V_{DSSAT}$, l é um parâmetro de ajuste e L_C é dado por:

$$L_C = \sqrt{\frac{e_{Si}.x_j}{C'_{OX}}} \tag{2.19}$$

onde: e_{Si} é a permissividade elétrica do silício e x_j é a profundidade da junção e *UCRIT* é o campo elétrico crítico dado por:

$$UCRIT = \frac{v_{\lim}}{m_o} \tag{2.20}$$

onde: v_{lim} é a velocidade de saturação e μ_o é a mobilidade dos portadores no canal na condição em que o campo elétrico transversal e longitudinal são desprezíveis.

2.2.5.2 Saturação da velocidade para os portadores

Embora constante para baixos valores do campo elétrico longitudinal (na direção do comprimento do canal), a velocidade dos portadores de carga tende a saturar à medida em que este campo aumenta. Uma vez que este campo corresponde ao gradiente do potencial elétrico na direção do eixo x, tal efeito será tão mais pronunciado quanto mais curto for o canal. Conseqüentemente, a corrente de dreno no transistor saturado apresentaria valores menores que os previstos pela teoria do MOSFET canal longo. Para levar em conta esta não idealidade, a expressão que calcula a carga de inversão na saturação é corrigida pela expressão (2.21):

$$m_{Sat} = \frac{m_o}{1 + \frac{m_o}{v_{\lim}} . \frac{1}{n.C'_{OX}} . \frac{dQ'_I}{dx}} \qquad (2.21)$$

onde: μ_o é a mobilidade dos portadores no canal na condição em que o campo elétrico transversal e longitudinal são desprezíveis e v_{lim} é a velocidade de saturação dos portadores.

O efeito da saturação da velocidade para os portadores ao ser considerado no cálculo das transcapacitâncias resulta nas equações que estão apresentadas na Tabela 2.3.

Tabela 2.3 Expressões das Transcapacitâncias considerando o efeito da saturação da velocidade dos portadores

VARIÁVEL	EXPRESSÃO
C_{gs}	$C_{gso}-\left(C_{gso}+C_{gdo}\right)\frac{s}{n}-\frac{1}{3n}\left(1-\frac{2\left(q_f^2+q_r^2\right)}{\left(q_f+q_r\right)^2}\right)\frac{L_{eq}}{v_{\lim}}g_{ms}$
C_{gd}	$C_{gdo}-\left(C_{gdo}+C_{gso}\right)\frac{s}{n}+\frac{1}{3n}\left(1-\frac{2\left(q_f^2+q_r^2\right)}{\left(q_f+q_r\right)^2}\right)\frac{L_{eq}}{v_{\lim}}g_{md}$
C_{gb}	$\frac{n-1}{n}\left(C_{OX}-C_{gso}-C_{gdo}\right)+\left((n-1)C_{OX}+C_{gso}+C_{gdo}\right)\frac{2s}{n}+\frac{1}{3}\left(1-\frac{2\left(q_f^2+q_r^2\right)}{\left(q_f+q_f\right)^2}\right)\frac{L_{eq}}{nv_{\lim}}g_{mb}$
C_{dd}	$C_{ddo}+\left(C_{dso}-C_{ddo}\right)\frac{s}{n}+\left(\frac{2}{15}\frac{3q_r^3+11q_r^2\,q_f+14q_f^2\,q_r+2q_f^3}{\left(q_f+q_r\right)^3}-\frac{1}{2}\right)\frac{L_{eq}}{v_{\lim}}g_{md}$
C_{ds}	$C_{dso}+\left(C_{ddo}-C_{dso}\right)\frac{s}{n}+\left(\frac{2}{15}\frac{3q_r^3+11q_r^2\,q_f+14q_f^2\,q_r+2q_f^3}{\left(q_f+q_r\right)^3}-\frac{1}{2}\right)\frac{L_{eq}}{v_{\lim}}g_{ms}$
C_{dg}	$\frac{\left(C_{ddo}-C_{dso}\right)}{n}-\left(\frac{2}{15}\frac{3q_r^3+11q_r^2\,q_f+14q_f^2\,q_r+2q_f^3}{\left(q_f+q_r\right)^3}-\frac{1}{2}\right)\frac{L_{eq}}{v_{\lim}}g_{mg}$
C_{db}	$\frac{\left(C_{ddo}-C_{dso}\right)}{n}\left(n-1-2s\right)-\left(\frac{2}{15}\frac{3q_r^3+11q_r^2\,q_f+14q_r^2\,q_f+2q_f^3}{\left(q_f+q_r\right)^3}-\frac{1}{2}\right)\frac{L_{eq}}{v_{\lim}}g_{mb}$
C_{bs}	$(n-1)C_{gs}$
C_{bd}	$(n-1)C_{gd}$

2.2.5.3 Redução de barreira induzida pelo dreno (Drain Induced Barrier Lowering – DIBL)

Em transistores de canal curto, a influência dos potenciais de fonte e dreno sobre a carga de depleção Q_B é significativa, ficando apenas uma parcela desta carga (Q^*_B) controlada efetivamente pelo potencial de porta. Um aumento do potencial de dreno faz a região de depleção associada à difusão de dreno avançar mais sob o óxido e a razão Q^*_B/Q_B torna-se menor, este efeito é denominado DIBL. Este efeito é empiricamente modelado através de uma modificação do fator de corpo, que passa a ser função do comprimento do canal e da profundidade da região de depleção associadas à junção de dreno e de fonte.

$$V_P = V_{P0} + \frac{s}{n}\left(V_{SB} + V_{DB}\right) \tag{2.22}$$

onde: V_P é a tensão de *pinch-off*, V_{P0} é a tensão de *pinch-off* no equibíbrio ($V_{DB}=V_{SB}=0$), s é o efeito DIBL e n é o fator de rampa.

2.2.5.4 Mobilidade com o campo transversal

O movimento superficial dos portadores é afetado pelo campo elétrico perpendicular associado ao potencial da porta. Uma modelagem simples para o fenômeno pode ser obtida substituindo a mobilidade normal μ_o pela mobilidade dada por:

$$m_{ef} = \frac{m_0}{1+q\ n.\left(V_{P0} + f_{S0}\right)} \tag{2.23}$$

onde: μ_{ef} é a mobilidade efetiva, μ_o é a mobilidade dos portadores no canal para o campo elétrico transversal e longitudinal desprezíveis, V_{P0} é a tensão de *pinch-off* no equilíbrio, q é um dos parâmetros de entrada e f_{S0} é o potencial de superfície no equilíbrio.

2.2.6 Ruído Térmico e 1/f

O código do modelo ACM desenvolvido em linguagem C disponível no simulador SMASH não possui o cálculo do ruído térmico e 1/f implementado, desta forma, para que este cálculo pudesse ser implementado no código em Verilog-AMS, foram usadas as equações que estão disponíveis em [21], sendo fórmulas simples e que são válidas para as regiões linear, de saturação e sublimiar de operação do transistor.

Tendo em vista que a implementação do ruído no código em Verilog-AMS precisa ser validada e que o modelo em C no simulador SMASH não dispõe desta possibilidade, adotaram-se como referência de comparação os resultados que foram obtidos por [21] e que estão apresentados na Figura 2.3.

Tais resultados foram encontrados a partir da simulação de um transistor SPICE NLEV=2,3 , com W=200µm e L=5µm, saturado, sendo que os parâmetros de entrada do transistor não foram informados.

De acordo com [21], os resultados do ruído térmico foram obtidos na freqüência de 25 kHz, para que os efeitos do ruído 1/f fossem minimizados e as medidas do ruído 1/f foram tomadas na freqüência de 1Hz.

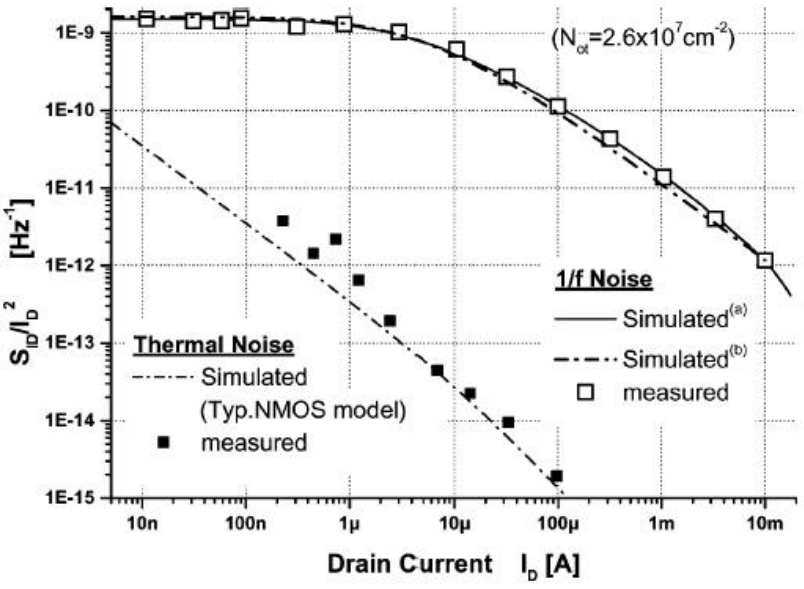


Figura 2.3 - Gráfico das curvas obtidas por [21] e que foram usadas como referência para a simulação do ruído com o código em Verilog-AMS ACM NMOS.

2.2.6.1 Formulação do ruído térmico

A equação usada na implementação do ruído térmico para o modelo ACM em Verilog-AMS:

$$Siw = -4.k_B.T.m_0.\frac{W}{L}.Q_I \tag{2.24}$$

Onde:

Siw = densidade espectral de potência (A^2/Hz)

k_B = constante de Boltzmann (1,38x10^{-23} C.V/K)

T = temperatura absoluta (K)

μ_0 = mobilidade do portador (m^2/V.s)

W = largura do canal (μm)

L = comprimento do canal (μm)

Q_I = carga total de inversão (C)

2.2.6.2 Formulação do ruído 1/f

A equação utilizada na implementação do cálculo do ruído 1/f do modelo ACM em Verilog-AMS é:

$$Sid = \frac{{q_e}^2.Not.m_0}{L^2.n.C'_{OX}}.Id.\frac{1}{f}\ln\left(\frac{n.C'_{OX}.f_t - Q'_{IS}}{n.C'_{OX}.f_t - Q'_{ID}}\right) \tag{2.25}$$

Onde:

Sid = densidade espectral de potência (A^2/Hz)

q_e = carga elétrica do elétron (1,602x10^{-19} C)

N_{ot} = densidade equivalente das trilhas de óxido (cm^{-2})

μ_0 = mobilidade do portador (cm^2/V.s)

L = comprimento do canal (m)

n = fator de rampa

$C'ox$ = capacitância do óxido por unidade de área(F/m^2)

Id = corrente de dreno (A)

f_t = potencial termodinâmico (V)

Q'_{IS} = densidade de carga da fonte (C/m^2)

Q'_{ID} = densidade de carga da fonte (C/m^2)

f = largura da banda de freqüência (Hz)

A variável N_{ot} é calculada da seguinte forma:

$$N_{ot} = \frac{k_B.T.N_t(E)}{g} \tag{2.26}$$

Onde:

$N_t(E)$ = densidade do óxido das trilhas por unidade de volume e energia ($cm^{-3}.eV^{-1}$)

g = coeficiente de atenuação da função da onda do elétron no óxido (cm^{-1})

3 VERILOG-AMS

3.1 LINGUAGEM DE DESCRIÇÃO DE HARDWARE

O rápido crescimento de produtos eletrônicos no mercado requer dos fabricantes de circuitos integrados (CI's) a solução de uma equação complexa com muitas variáveis, entre elas o tempo total de desenvolvimento das etapas de um projeto, ilustrada na Figura 3.1 e o respectivo custo. A solução que o fabricante encontrar para esta equação irá ou não definir a sua liderança no mercado, portanto esta dinâmica determina a necessidade da adoção de metodologias eficientes na execução do projeto, durante o desenvolvimento e fabricação do CI [29].

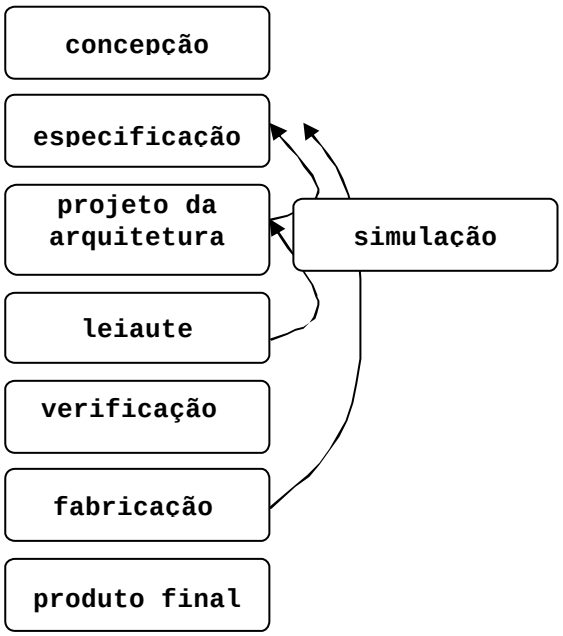


Figura 3.1 - Exemplo de metodologia usada no desenvolvimento de um CI ou sistemas [29].

Uma das soluções é o aumento no nível de integração dos CI's, que por sua vez determina o uso de metodologias e ferramentas de projeto, desenvolvimento e fabricação compatíveis com a complexidade que envolve as fases de desenvolvimento do CI. Outra forma de agilizar o processo de fabricação é a utilização de modelos compactos padronizados compatíveis com a maioria dos simuladores comerciais, por meio de bibliotecas padrões que permitam a atualização e distribuição na medida em que as modificações ocorram.

Estas necessidades puderam ser atendidas a partir de 1980, quando as linguagens tradicionais de programação como o C [30] começaram a ser substituídas pelas linguagens de descrição de *hardware* (HDL). Atualmente as HDL's padronizadas disponíveis no mercado são o Verilog-AMS [4],[5] e o VHDL-AMS, sendo o Verilog-AMS o objeto de estudo desta dissertação.

Portanto o uso do Verilog-AMS na implementação de um modelo, torna-o compatível com todos os simuladores que suportam esta linguagem, desta forma poderá ser distribuído diretamente aos usuários sem a necessidade da intermediação dos vendedores de EDA, ao contrário da linguagem C que ao ser usada na implementação dos modelos, os tornam funcionais para um conjunto restrito de simuladores comerciais.

Com relação à abrangência, a eletrônica analógica em relação à digital enfrenta maiores adversidades na padronização de bibliotecas, pois há uma grande diversidade nas características dos sinais analógicos, que podem assumir formas e intensidades diferenciadas de acordo com a freqüência de operação ou tensão de alimentação.

Enfim o sinal analógico é essencialmente complexo pela sua própria natureza, implicando em limitações quando comparado ao sinal digital. Neste contexto, o uso das HDL's no desenvolvimento de CI´s analógicos passa a ser fundamental pelas vantagens que pode proporcionar, tais como: a reutilização das bibliotecas, a experiência dos projetistas, minimização dos custos e do tempo de projeto, e principalmente por se tratar de uma linguagem padronizada.

3.1.1 Fundamentos da Linguagem de Descrição de *Hardware*

As linguagens computacionais usadas na representação de componentes ou circuitos eletrônicos utilizam o conceito de malhas e nós, na qual a forma da representação é por entidades primitivas constituídas por ramos e nós em que a lei de Kirchhoff da Corrente

(LKC) e da tensão (KVL) são válidas, sendo o SPICE um exemplo. A segunda forma usada, considera um nível de abstração maior, usando equações diferenciais, funções de transferência, procedimentos de controle e estruturas de atribuição que são características das linguagens de programação estruturadas como C ou FORTRAN [31].

Segundo [31] as premissas básicas para uma linguagem de descrição de comportamento analógico são:

- Suportar sinais analógicos e digitais e múltiplos domínios do comportamento.
- Abrangência nos tipos de estruturas descritivas, como equações diferenciais não lineares, funções de transferência, tabelas de dados.
- Permitir a múltipla descrição do comportamento em vários níveis de detalhamento para uma dada entidade (gráfico de fluxo, terminais, malha, componente, estrutura física).
- Diferenciação da descrição do circuito e condições de simulação.
- Compatibilidade na semântica da linguagem para possibilitar o uso simultâneo dos domínios analógico, digital e de sinais mistos.
- As estruturas de construção (especificações, esquemáticos) devem ser explícitas com o propósito da documentação do projeto.
- Bibliotecas de modelos padronizados.
- Estruturas de programação consistentes.
- Integração de mecanismos para descrição física (geométrica) do componente final.

Conforme exposto, uma HDL é capaz de descrever de forma abrangente, as estruturas e os comportamentos. Em uma descrição estrutural, detalha-se a interconexão entre os componentes que fazem parte do circuito sendo usada como entrada para uma simulação da mesma forma que uma entrada esquemática, já a descrição comportamental descreve o funcionamento de cada um dos componentes do circuito [1].

Portanto o diferencial de uma HDL é a sua capacidade de descrever simultaneamente o comportamento de componentes individuais e como estão interligados [14], sendo processada seqüencialmente nas estruturas individuais e de forma concorrente entre as estruturas individuais, enquanto uma linguagem tradicional de programação é essencialmente seqüencial.

Em termos gerais o objetivo principal da HDL é caracterizar a síntese e propiciar a simulação de projetos eletrônicos.

Síntese: Capacidade do simulador de traduzir o código da linguagem HDL para uma linguagem mais próxima da implementação, ou seja, realiza transformações entre níveis de especificação do sistema, até a realização física deste sistema, conforme ilustra a Figura 3.2, introduzindo detalhes estruturais e/ou geométricos. Está relacionada com a tecnologia de construção deste sistema que por sua vez é formada por bibliotecas específicas.

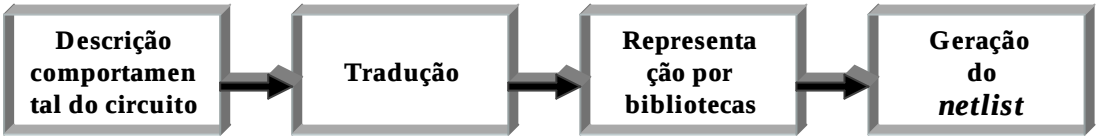


Figura 3.2 - Fluxo das etapas do processo de síntese[14].

Simulação: Pelo uso de uma ferramenta de automação de projetos eletrônicos (EDA), sinais de entrada são aplicados como estímulos e o(s) sinal(ais) de saída são observados sem a necessidade da construção física do circuito [13], além de checar ou extrair dados pertinentes à dinâmica[5] do funcionamento.

3.2 FUNDAMENTOS DA LINGUAGEM

3.2.1 Histórico

Verilog é o acrônimo de VERIfying LOGic, idealizada por Philip Moorby e Prabhu Goel da empresa *Automated Integrated Design Systems,* posteriormente *Gatway Design*

[5] Propriedade denominada de "*timming*", seu uso é mais freqüência no projeto de circuitos digitais.

Automation entre 1983 e 1984. Inicialmente era uma linguagem proprietária, mas em 1991 passou a ser de domínio público. Atualmente é a empresa *Cadence Design Systems* que detém os direitos da linguagem e que foi a responsável pela sua abertura e conseqüentemente a sua difusão.

Cronologicamente o domínio digital foi o precursor na padronização da linguagem, denominada como Verilog HDL (*Hard Description Language*) e ocorreu em dezembro de 1995 pela norma IEEE 1364-1995. Em junho de 1996 devido à implementação de melhorias no Verilog-95, voltou a ser revista pela IEEE, estas novas extensões na linguagem foram padronizadas pela norma IEEE 1364-2001 conhecida como Verilog-A (*Analog*). Na última e atual versão, padronizada como norma IEEE 1364-2005, pequenas correções foram realizadas e a melhoria mais significativa foi a inclusão de sinais analógicos e mistos. As melhorias a partir deste projeto são chamadas por Verilog-AMS (*Analog and Mixed Signal*) abrangendo os domínios: digital, analógico e sinais mistos (quando coexistem os dois sinais), grandezas da mecânica, ótica, termodinâmica e mecânica dos fluidos.

Verilog-AMS é uma linguagem de descrição de *hardware*, formada a partir do Verilog-HDL e Verilog-A que oferece meios de especificar sistemas eletrônicos em vários níveis de abstração (Figura 1.1), seja no estágio inicial do projeto ou nas fases finais de simulação do circuito. É uma linguagem simples, disponível em um único simulador, podendo ser compartilhada entre projetistas de circuitos digitais e de circuitos analógicos.

A organização de um projeto que demanda pelo compartilhamento entre sinais analógicos, digitais e mistos é facilitada pela abrangência da linguagem, pois a objetividade é maior na estruturação de blocos em que o comportamento é formado por sinais mistos, caso contrário cada sinal deverá ser simulado separadamente.

Os simuladores AMS são compostos por dois gerenciadores de simulação (*kernel*) [32], um deles responsável pela simulação lógica: simulação orientada por evento (*event-driven simulation)* e o outro para a simulação em tempo contínuo: *continuous-time simulation*.

Tendo como base [34], a simulação AMS é composta pela existência de uma rede hierarquizada de processos paralelos que trocam mensagens sob o controle do gerenciador de simulação que atualiza concorrentemente os sinais e variáveis. As atribuições da estrutura dos sinais independentes não afetam os sinais alvos imediatamente, somente serão atualizados no próximo ciclo de simulação. O gerenciador de simulação retoma o processamento quando todos os processos definidos pelo usuário tornam-se suspensos, seja pela execução de uma estrutura de espera ou até alcançar a última estrutura de processamento. De forma resumida, o gerenciador de simulação atualiza os sinais e variáveis e os suspende enquanto o processo do

usuário prossegue. Se o tempo do próximo mais novo evento “tn” é igual ao tempo da simulação corrente “tc”, o processo do usuário executa um novo ciclo.

3.2.2 Características

A semântica da linguagem Verilog-AMS tem sua raiz fundamentada em C, o que a torna fácil de aprender e usar, principalmente para os usuários de C.

Com base em [13] e [35] algumas das características da linguagem:

- Os projetos podem ser decompostos hierarquicamente.
- Capacidade de descrever diferentes tipos de comportamentos: linear, não linear, integro-diferencial, simulação orientada por eventos analógicos (*analog event driven*).
- Disponibiliza recursos de semântica para lidar com a descontinuidade.
- A estruturação da linguagem permite com que o projetista defina o fluxo dos sinais, isto implica em formulações mais estáveis e robustas.
- Permite a interdisciplinaridade entre as áreas elétrica e mecânica.
- Cada elemento do projeto possui distintamente uma interface de conexão com outros elementos e uma precisa especificação funcional.
- Interoperabilidade dos módulos entre as HDL’s. Módulos podem ser construídos em uma HDL e invocar ou ser invocado por outra HDL.
- Modelos compactos podem ser atualizados de forma independente da versão do simulador.
- Facilita ao uso da metodologia descendente (*top-down*).
- Executa automaticamente a interface entre modelos analógicos e digitais mesmo quando seus terminais não são compatíveis.

O Verilog-AMS dispõe de um conjunto de estruturas (entre outras) relacionadas à semântica, que tipificam as características AMS da linguagem entre elas a capacidade da descrição do comportamento dos componentes analógicos ou digitais.

A listagem representada na Figura 3.3 exemplifica a estrutura básica de programação em Verilog-AMS, de um módulo que descreve o comportamento de uma fonte de tensão DC, formada pelos terminais **`t1`** e **`t2`** , que pode ser fixa ou variável, dependente do valor do

parâmetro **mode**, a intensidade é função do parâmetro **bias**, sendo restringida a um intervalo de valores apropriado. A seguir são apresentados elementos da sintaxe essenciais na construção dos módulos AMS.

- **discipline:** relaciona domínios tais como: elétrico, magnético, térmico, cinemático, rotacional que estarão presentes naquela estrutura de programação. No exemplo da Figura 3.3, a linha 3 define que **t1** e **t2** são terminais de natureza elétrica, como poderiam ser de natureza térmica (**thermal**).
- **nature**: define qualitativamente e quantitativamente as grandezas do domínio analógico, como a sua unidade (ampére, volt, kelvin), a letra usada para acessá-la (*access function*), tolerância absoluta e a grandeza resultante no caso de uma operação de integração ou derivação. Por exemplo: a letra **I** maiúscula para a **discipline** corrente, cuja unidade é ampére (A), seu **abstol** default é $1x10^{-12}$ e caso a operação de integração seja realizada a grandeza resultante é definida por: **idt_nature= Charge** .
- **'include**: é uma diretiva de compilação em que arquivos externos, como as bibliotecas padrão, são acessadas como arquivos durante todo o processamento.
- **module**: é a unidade básica de projeto e programação, acomoda a descrição de um componente até um sistema, no exemplo listado são as linhas 2 a 13.
- **parameter**: é atribuído para o identificador um valor constante, não deve mudar no curso da simulação, que será utilizado em seu lugar, pode ser real ou inteiro. A sintaxe permite definir um intervalo no qual o parâmetro pode estar contido, no exemplo as linhas 5 e 6 mostram esta característica.
- *system tasks*: são funções que retornam valores reais, tendo como entrada os valores atribuídos aos parâmetros.

  Ex1.: <ktonq = **$vt(($temperature** (tnom))>, esta estrutura está calculando o valor do potencial termodinâmico pelo uso da função **$vt()**, sendo **tnom um parâmetro** convertido de ºCelsius para kelvin por meio da *system task*: **$temperature().**

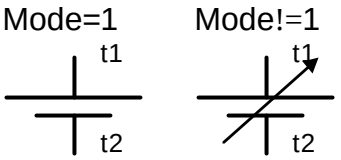
  Ex2.: Na linha 9 , em que o potencial do sinal de saída entre os terminais **t1** e **t2** é uma função do tempo absoluto na unidade segundos:

  <V(t1,t2) <+ **$abstime** >
- *access function*: uma das características qualitativas que formam a **nature** , como uma disciplina é associada a um sinal de entrada e/ou saída.

Ex.:< `vds = V(t1,t2)`>, a variável **`vds`** é igual a diferença do potencial do nó **`t1`** e **`t2`**.

- **`<+`** (*contribution operator*): como um sinal de saída recebe o valor de uma expressão que indica o comportamento daquele sinal, sempre associado a uma disciplina.

  <**`output_signal <+ f (input_signal)`**>, este operador está sendo usado na linha 9 e 11.




```
1-/////////////// MÓDULO SOURCE  ///////
2-module source(t1, t2);
3-electrical t1, t2;
4-inout t1, t2;

5-parameter integer mode  = 1   from [-1:1] exclude 0;
6-parameter real    bias  = 1.0 from [1:3.5);
7-analog begin
8-      if (mode  ==  1)
9-              V(t1,t2)     <+  $abstime;
10-     else
11-             V(t1,t2)    <+   bias;
12-end
13-endmodule
```

Figura 3.3 - Listagem de um módulo que descreve uma estrutura de comportamento em Verilog-AMS .

3.2.3 Metodologia de Projetos

Durante o projeto de um sistema eletrônico, as representações comportamental, estrutural e física (Figura 1.1) podem ser usadas em diferentes níveis de abstração de acordo com o tipo de objeto utilizado para isso. Tendo como base [36], os níveis de abstração para componentes analógicos se organizam conforme Tabela 3.1.

Tabela 3.1 Representação do projeto e níveis de abstração [36].

NÍVEL	FORMATO DO COMPORTAMENTO	ESTRUTURA DO COMPONENTE	OBJETO FÍSICO
Básico	Equações diferenciais, Equações analíticas, diagramas ExI	Resistor, indutor, capacitor	Células analógicas e digitais
Semicondutor	Equações diferenciais, diagramas, funções de transferência	Diodo, TBJ, MOSFET	Módulos, unidades
Blocos Funcionais	Algoritmos, fluxogramas, conjunto de instruções	Conversores, de/modulador, filtros	*Microchips*
AMS	Sinais mistos	Sensores, atuadores, transdutores	Placa de circuito impresso, módulos *multichip*

3.2.3.1 Bottom-up e Top-down

Conforme [1] as metodologias de projeto são seqüências de transformações que partem de uma descrição e/ou especificação de um sistema para chegar a uma descrição final validada deste sistema, que é suficiente e necessária para o processo de fabricação. A descrição final contém informações relativas aos eixos estrutural e geométrico do projeto, sendo função dos componentes primitivos do nível de abstração no qual é fabricado o sistema. Este nível de fabricação pode ser o das máscaras, no caso de CI's, ou do leiaute quando circuito impresso, no caso de sistemas implementados com componentes discretos, ou mesmo níveis mais abstratos, no caso de uma implementação com subsistemas adquiridos de fornecedores externos, o que é comum em empresas ditas integradoras.

Uma metodologia de projeto particular pode empregar um processo descendente (*top-down*) ou ascendeste (*bottom-up*) de concepção, ou ainda uma combinação de ambos.

Um processo descendente de projeto é aquele no qual uma seqüência de transformações de síntese e de validação é utilizada, partindo de um nível *i* e chegando a um nível *i-j* (processo de refinamento). No processo ascendente a seqüência de transformações de síntese e de validação parte de um nível *i* para chegar a um nível *i+j* (processo de composição).

Usualmente, a metodologia de projeto realiza uma combinação dos processos ascendente e descendente sendo de forma descendente até que se atinja um nível N nas quais as entidades primitivas sejam elas estruturais ou comportamentais, têm uma implementação conhecida em termos de primitivas estruturais do nível de fabricação. Estas primitivas do nível N formam uma biblioteca de funções ou de células, e sua concepção é normalmente realizada de forma ascendente.

A implementação do modelo ACM em Verilog-AMS consiste em uma representação do mais baixo nível hierárquico de abstração no desenvolvimento de um projeto, considerado como uma entidade primitiva.

3.3 DESENVOLVIMENTO DO CÓDIGO EM VERILOG-AMS

A literatura padrão da linguagem Verilog-AMS é o Manual de Referência da Linguagem versão 2.2 - Novembro 2004 (*Language Reference Manual – LRM*) [4] desenvolvido pela organização denominada *Accellera Organization Inc.* [5], sendo responsável pelo desenvolvimento de metodologias para a obtenção de padrões, objetivando a melhora da produtividade de projetistas da indústria eletrônica.

Os simuladores utilizados para a implementação do código fonte em Verilog-AMS apresentam as seguintes especificações:

- **SMASH**- release 5.9.0-5.9.3 – novembro 2007
- **ELDO**: ICstudio v2006.2_3.1 (2006.2a)
  DA-IC – Design Architec-IC v2006.2_3.1 (2006.2a)
  EZwave - Release v2.6_3.1 (AMS2006.2b Production)

3.3.1 Estruturação do Código em Verilog-AMS

A visão geral do programa em Verilog-AMS do código do transistor ACM assume a estrutura representada na Figura 3.4. A divisão em módulos está sendo usada como uma

forma de organização, uma vez que as linguagens de descrição de *hardware* são caracterizadas por módulos que serão processados de forma concorrente.

Figura 3.4 - Estrutura simplificada do programa em Verilog-AMS do código ACM.

A partir do modelo descrito no capítulo 2, foi usado o simulador SMASH para desenvolver o código fonte em Verilog-AMS que neste caso recebe a extensão `vams`. A implementação do código ACM foi realizada com base na estrutura indicada na Figura 3.4:

Primeiro módulo: destina-se à declaração dos parâmetros de entrada e parâmetros dependentes do transistor indicados na Tabela 2.1, como comprimento, largura, temperatura além da definição das variáveis. A sintaxe da linguagem prevê a existência de variáveis locais e globais, porém no desenvolvimento do código foram usadas variáveis globais, uma vez que o simulador em uso não estava com este recurso implementado na versão usada.

Segundo módulo: calculam-se os parâmetros e variáveis internas definindo as características físicas do transistor, como: fator de rampa, tensão de *pinch-off* , carga de inversão , tensão de saturação e a corrente de dreno entre outras grandezas, usando as equações matemáticas que definem o modelo ACM, sendo que as principais são apresentadas na sessão 2.2. Este módulo não apresentou nenhuma particularidade em relação à sintaxe do Verilog-AMS, no caso do simulador SMASH houve limitações decorrentes da implementação parcial da linguagem como o uso da

estrutura *Procedural Assignments* exemplificada pela sintaxe: *analog function*, em que cálculos auxiliares são executados para que os resultados sejam usados na rotina principal, mas que pode ser implementada no simulador ELDO[6].

Terceiro módulo: calculam-se as cargas elétricas do dreno, fonte, porta e substrato para determinação da corrente AC. Este módulo ao ser comparado com o código original, foi reduzido significativamente em número de linhas. Tal redução é porque em C, as capacitâncias são calculadas explicitamente pelo uso da operação de derivação para serem usadas no cálculo da corrente AC, mas na linguagem Verilog-AMS, a operação derivação é realizada diretamente pelo operador *time derivative* ddt(), aplicando a regra da cadeia no conjunto de equações (3.1).

$$\begin{aligned} i(t) &= C.(dV/dt) \\ C_{\text{intrínseca}} &= dQ/dV \end{aligned} \qquad (3.1)$$

Na listagem da Figura 3.5 detalha-se a execução do operador *time derivative* que substituiu o cálculo explícito das capacitâncias usadas para a determinação da corrente AC.

Inicialmente o laço condicional atribui o correto sinal à corrente DC e às cargas, tendo como premissa a forma de como o transistor está polarizado e o tipo do transistor se NMOS ou PMOS. Concluída esta verificação, a corrente AC é calculada pelo uso do operador derivação que é aplicado às cargas, realizando implicitamente o cálculo por meio de (3.1).

```
if (drain3 == 0) begin
    I(d,s) <+   material* idc;
    Qd     = material* qd;
    Qg     = material* qg;
    Qs     = material* qs;
    Qb     = material* qb;
end    else begin
       if ( drain3== 1)  begin
              I(d,s) <+ -material* idc;
       Qd     = -material*qg;
       Qg     = -material*qd;
       Qs     = -material*qs;
       Qb     = -material*qb;
       end
end
//////////////Corrente AC ///////////////
    I(d,s) <+  ddt(Qd);
    I(g,b) <+  ddt(Qg);
    I(s,b) <+  ddt(Qs);
```

Figura 3.5 - Listagem de parte do código em Verilog-AMS que realiza o cálculo da corrente AC.

[6] Para efeito do código final em Verilog-AMS, quando usado no simulador SMASH, este recurso foi "comentado".

Quarto módulo: apresentado na Figura 3.6, são calculados os ruídos térmico e 1/f de forma simplificada utilizando as equações que foram apresentadas no item 2.2.6 aplicadas às funções **`flicker_noise()`**(linha 6) e **`white_noise()`**(linha 11).

```
1. ////////RUÍDO 1/f/////////////////
2. begin:Flicker_Noise
3. Not =   (ktonq * ntraps/ e_gamma)*1e+4;
4. Sid =   ((`P_Q*`P_Q*Not*uef*xi)/(l*l*n*cox))*
5. ln((ncoxktonq-qis)/(ncoxktonq-qid));
6. I(d,s) <+   flicker_noise(Sid,1,"Sflicker");
7. end //Flicker

8. ////////RUÍDO TÉRMICO/////////////////
9. begin:White_Noise
10.    Siw   =  -4*ktonq*uef*w/l*((2/3*(qis*qis+qis*qid+qid*qid))-
   (ncoxktonq*(qis+qid)))/(qis+qid-2*ncoxktonq);

11.    I(d,s)  <+  white_noise(Siw,"Swhite");
12.    end //White(thermal)
```

Figura 3.6 - Listagem referente ao Módulo 4 da estrutura do código em Verilog AMS

3.4 EXEMPLOS EM VERILOG-AMS

Conforme foi apresentado no item 3.1.1, o Verilog-AMS pode ser usado na construção de módulos primitivos e este por sua vez utilizado em conjunto com qualquer outra linguagem de simulação desde que o Verilog-AMS seja suportado pelo simulador em questão. A seguir serão mostrados dois exemplos de programação, no primeiro será usada somente a linguagem Verilog-AMS, e no segundo exemplo o Verilog-AMS é chamado pelo SPICE.

3.4.1 Primeiro Exemplo: Verilog-AMS

Na seqüência é apresentado o *netlist* para obter da característica de saída da corrente de dreno em função da tensão de dreno-substrato (Id=f(db)) para um transistor PMOS modelo ACM usando o Verilog-AMS.

O *netlist* a apresentado na Figura 3.7 configura-se em dois módulos: *source*, e *top*. Basicamente a função do **`module source`** , que foi previamente comentado na sessão 3.2.2, é a de desempenhar a função de uma fonte DC fixa ou variável, conforme o valor do parâmetro que define esta condição. O **`module top`** é o que realiza as conexões entre o módulo *source* e o módulo do transistor. O código fonte ACM está na biblioteca de nome **`acm.vams,`** que é incluída por meio da diretiva de compilação **`'include<>`** que também vincula as bibliotecas **`disciplines`** e **`constants`**.

Comentários sobre o *netlist*:

Inicialmente é sinalizado para o simulador SMASH que a linguagem que será usada é o Verilog-AMS, por meio do *tag* **`>>> VERILOG`** .

`module source`: descreve o comportamento de um componente, que no caso é a fonte de tensão contínua, que se interliga por dois terminais (*ports*) denominados **`t1`** e **`t2`**, que admitem o fluxo de sinal tanto no sentido 'entrando' como 'saindo', são declarados como **`inout`**. A descrição do componente deve ficar encapsulada na estrutura **`analog begin`** , sendo necessário para se caracterizar como uma descrição de comportamento o uso da estrutura *contribution:* **`<+`** vinculada a um sinal de potencial ou de fluxo (*output_signal* **`<+`** f(*input_signals*)).

`module top`: atribui os valores dos parâmetros aos módulos e os interliga. A inexistência da declaração dos terminais de entrada e saída indica que é o próprio módulo **`top`** que gerencia os sinais de potencial e de fluxo. A passagem dos parâmetros é feita pela associação posicional[7], ou seja, o novo valor numérico do parâmetro de entrada é atribuído na ordem em que foi declarado no código fonte ou

[7] *positional association* de acordo com [29], *named association* é a outra forma de associar os parâmetros.

no módulo primitivo. Como o código fonte do modelo ACM possui como parâmetros *default* os de um transistor NMOS, é necessário a passagem dos parâmetros relativos ao PMOS por meio do operador do operador ‘tarefa # ’ na mesma seqüência que foi escrito no código fonte, salienta-se que a forma usada para a passagem dos parâmetros é específica para o simulador SMASH.

```
//Curva Característica Id=f(Vdb)@ Vsb Transistor PMOS///////
>>> VERILOG
                                                  Mode=1      Mode != 1
`include "disciplines.vams"                                t1          t1
`include "constants.vams"
`include "acm.vams"

`timescale 1s / 1fs                                        t2          t2
////////////// MÓDULO SOURCE  ///////////////
module source(t1, t2);
electrical t1, t2;
inout t1, t2;

parameter integer mode  = 1;    //fonte variável no tempo
parameter real    bias  = 1.0;  //nivel DC

analog begin
              if (mode  ==  1)
              V(t1,t2)     <+  $abstime;   // fonte variável
              else
              V(t1,t2)    <+   bias;       // fonte fixa
              End

endmodule

// MÓDULO TOP (Id=f(Vdb)) PMOS /////////

module top();
electrical p1, p2, p3, p4;
electrical gnd;
ground gnd;

parameter integer   modeVar  =  1;     //fonte variável
parameter integer   modeFix  =  2;     //fonte fixa
parameter real      bias3    = 3.0;    //amplitude 3 volts
parameter real      bias0    = 0.0;    //amplitude 0 volts

acm # (-1, 1.0e-6, 10.0e-6, 4.0e-9, 7.7e-9, 1.0e-7,1.04e-15, 27.0,
0.375, 4.567, -0.719, 0.81, 149.4e-4, 1.169e5, 0.0509, 2.0)
topMOS(p1,p2,p3,p4);

source # (modeVar)          source1 (p1,gnd);    //drain-variável
source # (modeFix,bias0)    source2 (p2,gnd);    //gate-fixo
source # (modeFix,bias3)    source3 (p3,gnd);    //source-fixo
source # (modeFix,bias3)    source4 (p4,gnd);    //bulk-fixo

endmodule
```

Figura 3.7 - Listagem usando somente Verilog-AMS para gerar curva de saída Id=f(Vdb) @ Vsb para um transistor PMOS W=10µm L=1µm.

3.4.2 Segundo Exemplo: Verilog-AMS + SPICE

A função do circuito a seguir é o traçado da característica de saída de dois transistores NMOS modelo ACM, um deles em Verilog-AMS (**topnmos**) e o outro em linguagem C (**M1**) em função tensão de porta-substrato, repetindo esta análise para valores da tensão de fonte entre 0,5 V a 2V a cada 0,5V usando o SPICE para invocar o módulo escrito em Verilog-AMS.

Comentários sobre o *netlist*:

Como este exemplo contém duas linguagens, a cada início da descrição, deve se sinalizado para o simulador SMASH que a linguagem que será usada é o Verilog-AMS, por meio do *tag* **>>> VERILOG** seguida pelo tag >>> SPICE.

module topN: Este módulo descreve o sub-circuito **topN** composto por um transistor ACM denominado de **topnmos**, não houve a passagem de parâmetros porque estão sendo usado os parâmetros *default* do transistor e que neste caso estão definidos no código fonte (**.vams**). O código fonte do transistor ACM está implícito na estrutura por meio da diretiva **`include "acm.vams"** .

Módulo SPICE: Para o restante do *netlist*, a sintaxe do SPICE passa a ser usada, sendo a letra "X" para definir um sub-circuito **topN**, seguido pela passagem dos parâmetros do transistor ACM NMOS , finalmente é feita a polarização dos transistores e as diretivas do tipo de análise a ser realizada, que neste caso se resume na análise DC, onde Vgb varia de 0V a 3V a cada 20mV, podendo ser realizada para valores diferentes da tensão de fonte, configurando-se em um *sweep* de 0,5V a 2V a cada 0,5V. O resultado da análise DC é traçado em um mesmo gráfico para as duas curvas da corrente de dreno.

```
//// Cacacterística de saída Id=f(Vgb) @ Vsb  //// ACM NMOS

/////SINTAXE SMASH///////////////
M1 D1 G1 S1 0 ACMN L=1u W=10u

 >>> VERILOG

`timescale 1ns / 1ps
`include "disciplines.vams"
`include "constants.vams"
`include "acm.vams"

module topN(d1,g1,s1,b1);
    electrical d1,g1,s1,b1;
    electrical gnd;
    inout d1,g1,s1,b1;
    ground b1;
    acm topnmos(d1,g1,s1,b1);
endmodule
>>> SPICE

////Módulo SPICE
Xacm  D1 G1 S1 0 acm

.MODEL ACMN NMOS LEVEL=10
+TOX=7.7E-9   UO=436.8         PHI=0.862         VTO=0.486
+GAMMA=0.601  SIGMA=268.8E-18  THETA=0.0593      VMAX=1.613E5
+XJ=1E-7      pclm=1.645       PB=0.86  LD=4E-9  DW=-0.1E-6

//Polarização dos transistores
Vd1 D1 0 3
VG1 G1 0 0
Vs1 S1 0 vsx

//Diretivas da análise
.DC LIN VG1.Value 0 3 20m
.param vsx=0.5
.paramsweep vsx 0.5 2.0 0.5
.Trace  DC    I(D1_VE_ISPICE)  ID(M1)
```

Figura 3.8 - Listagem usando o SPICE para chamar módulo em Verilog-AMS para obter a curva de saída Id=f(Vgb) @ Vsb para um transistor NMOS W=10µm L=1µm.

4 RESULTADOS

A simulação do modelo transistor MOS ACM ocorreu pela combinação das análises tradicionalmente realizadas em um circuito eletrônico que é composto pela análise DC, análise de sinais transientes no domínio do tempo, análise para pequenos sinais AC e ruído, acrescida dos mesmos testes de validação que foram executados no modelo ACM quando o mesmo foi desenvolvido [19].

4.1 DESENVOLVIMENTO DO PROCESSO DE VALIDAÇÃO

A validação do código ACM, implementado em Verilog-AMS, foi realizada nos simuladores ELDO e SMASH de forma que constatou-se a portabilidade do código modelado em Verilog-AMS, comprovando a premissa que a linguagem permitiria esta versatilidade.

As análises que somente dependiam de um único transistor foram realizadas no SMASH, as quais limitaram-se à análise DC do código ACM e para a validação de circuitos onde eram construídos a partir do transistor MOS, foi utilizado o simulador ELDO.

A Tabela 4.1 a seguir apresenta um resumo das análises realizadas para a validação do código ACM.

Tabela 4.1 Relação dos testes realizados para validação do modelo.

ESQUEMÁTICO/*NETLIST* SIMULADO	TIPO DE ANÁLISE	SIMULADOR
Polarização NMOS / PMOS	DC	SMASH
Inversor	DC - AC - Tran	ELDO
Associação Série-Paralela	DC	ELDO
Amostragem e Retenção	Tran	ELDO
Capacitor Chaveado	Tran	ELDO
Simetria das Capacitâncias	Tran	ELDO
Espelho de Corrente	Ruído	ELDO

4.1.1 Característica Corrente x Tensão do Transistor MOS – Análise DC

Na análise DC do transistor MOS, verificou-se o comportamento da corrente de dreno em função das tensões de dreno, porta e fonte, para um transistor NMOS e PMOS, tecnologia 0,35µm, W=10µm e L=1µm.

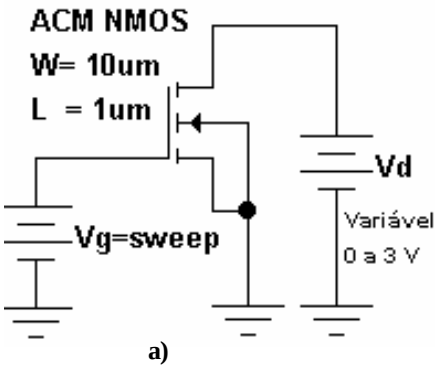
As figuras a seguir mostram corrente de saída Id e as transcondutâncias do transistor em função da tensão de polarização Vdb, Vgb e Vsb para o código em C[8] e em Verilog-AMS.

4.1.1.1 Característica de saída Id = f(Vdb) @ Vgb - NMOS

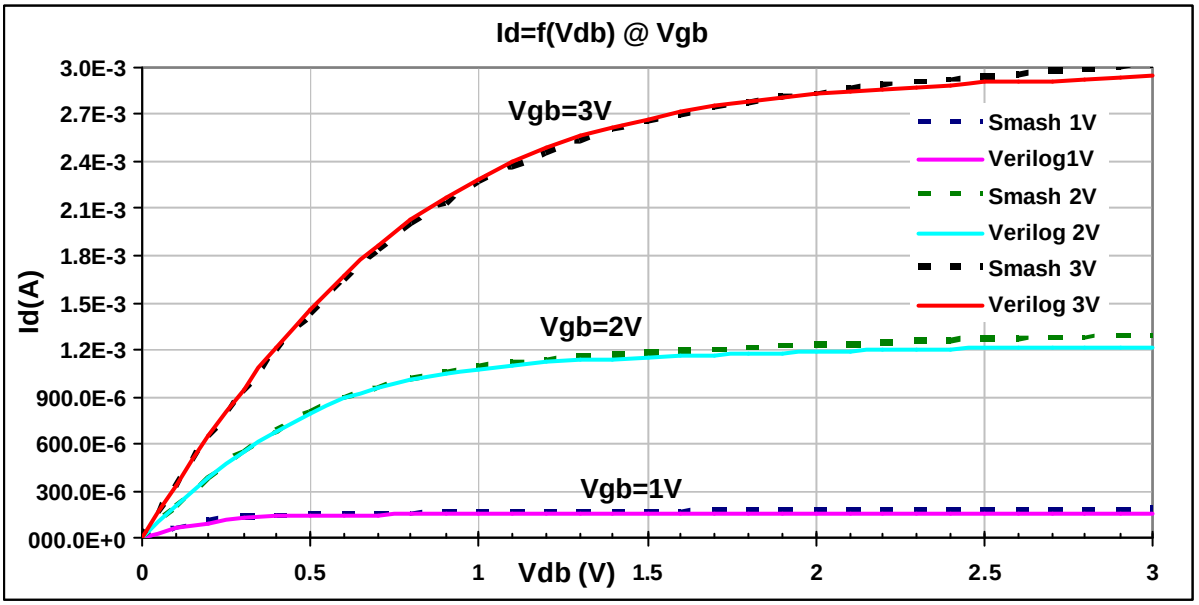
Nesta análise estão sendo comparadas as famílias de curvas da corrente de dreno, geradas pela simulação do modelo ACM em C e em Verilog-AMS.

[8] Linguagem C ou código original indica que o modelo foi implementado na linguagem C, sendo disponível somente no simulador SMASH.

A tensão de dreno variou de 0 a 3 V, a tensão de porta assume a cada simulação consecutiva, tensões entre 0 e 3V a cada 1V. O resultado da simulação é apresentado na Figura 4.1.



a)



b)

Figura 4.1 - a) Circuito simulado b) Gráfico Id x Vdb para Vgb variando de 1 a 3 V e Vsb = 0 V. Transistor NMOS W=10µm e L=1µm.

Os resultados obtidos pela simulação entre o código original e o desenvolvido em Verilog-AMS (Figura 4.1) apresentam-se praticamente sobrepostas na região de operação linear (triodo), ocorrendo uma maior diferença na região de saturação para Vgb=1 V, conforme pode ser visualizado na Figura 4.2, onde são mostrados os percentuais de erro entre o código em C e o em Verilog-AMS.

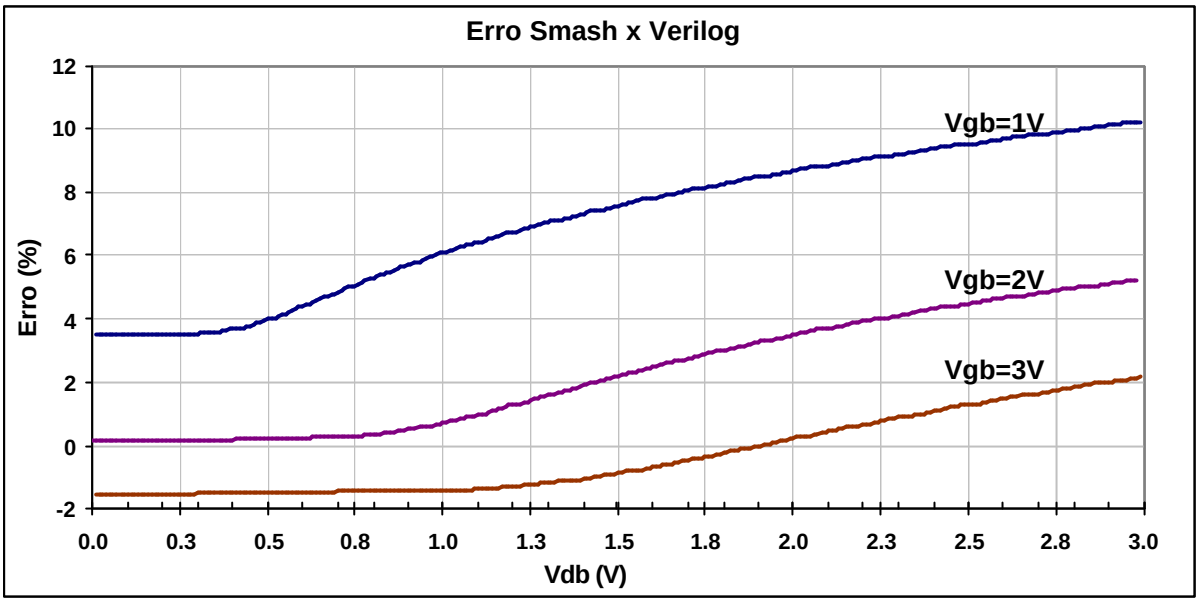
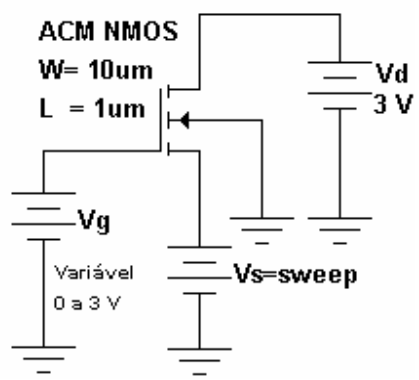


Figura 4.2 - Gráfico apresentado o percentual de erro entre o código original e o em Verilog-AMS para a simulação apresentada na Figura 4.2.

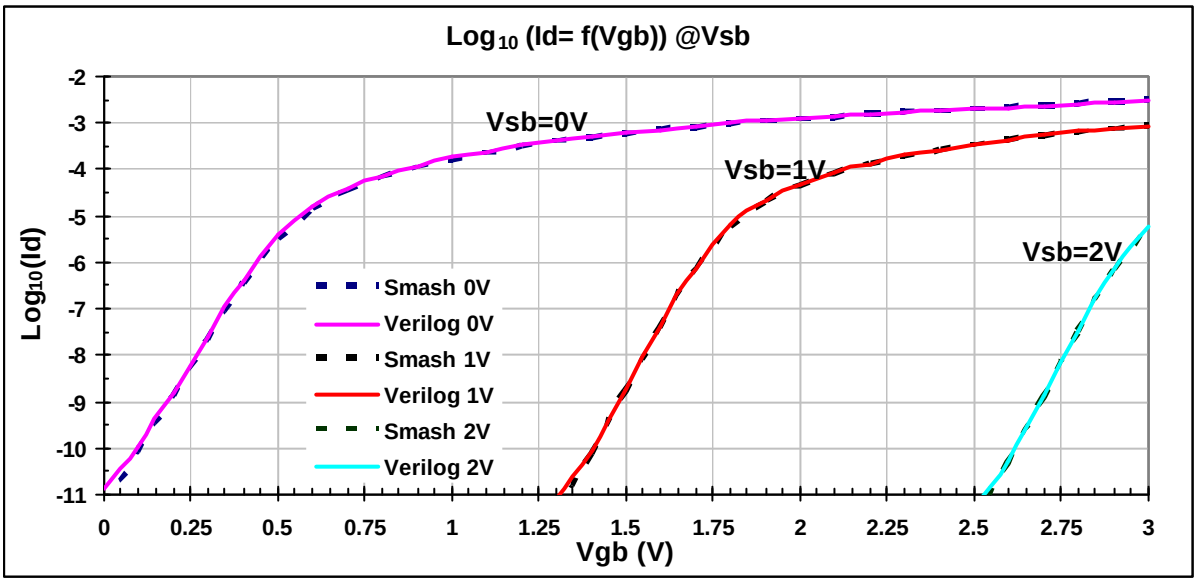
Após minuciosa verificação do código pela conferência das equações, pela repetição da simulação utilizando diferentes métodos numéricos e a alteração nos tempos de controle da precisão da simulação, constatou-se que os resultados das simulações não se modificaram pela mudança do método numérico ou pela variação do passo do tempo. Assim sendo, tem-se a possibilidade de que o código fonte em linguagem C seja o provável motivo das diferenças verificadas.

4.1.1.2 Característica de saída $\log_{10}$ Id =(f (Vgb)) @ Vsb NMOS

A simulação compara os resultados entre os códigos C e em Verilog-AMS da característica DC para o logaritmo na base dez da corrente de dreno em função da tensão de porta e Vsb varia de 0 a 2V a cada 1V. Nota-se, neste caso, a sobreposição das curvas dos dois códigos em todas as regiões de operação.



a)

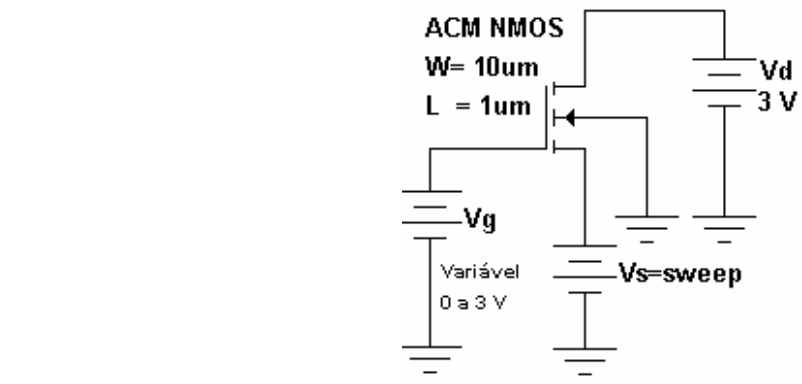


b)

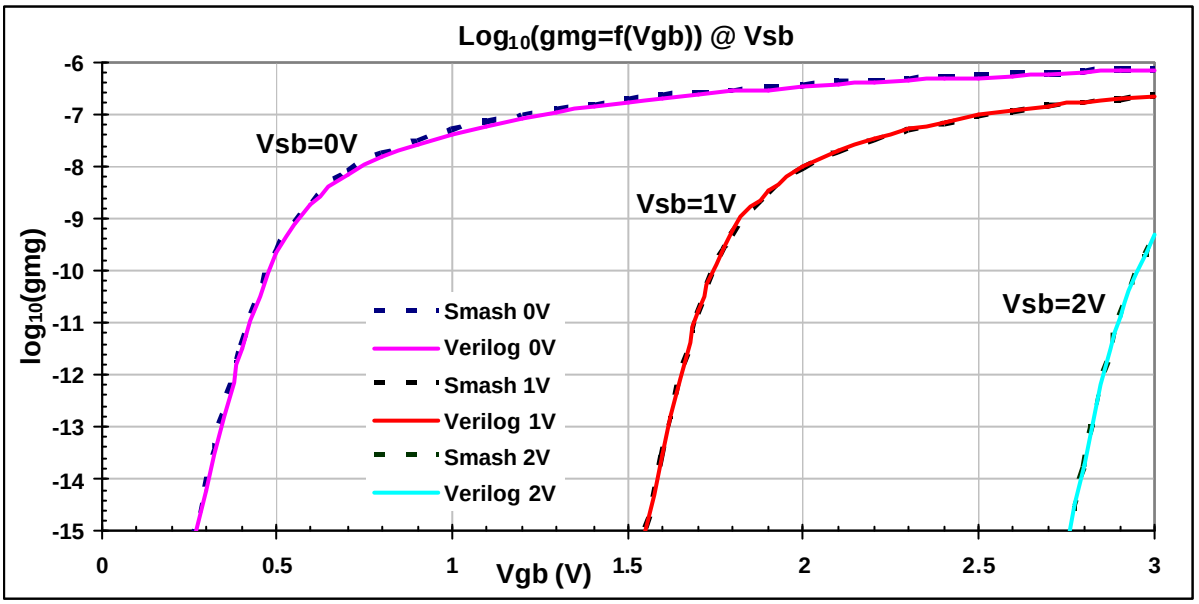
Figura 4.3 - a) Circuito simulado b)Gráfico da característica $\log_{10}$(Id) x Vgb para Vsb variando de 0 a 2 V e Vdb = 3V. Transistor NMOS W=10µm e L=1µm.

4.1.1.3 Característica da transcondutância gmg em fonte comum NMOS

A Figura 4.4 a seguir mostra o logaritmo na base dez da transcondutância gmg do código ACM original e em Verilog-AMS, na saturação, em função de Vgb. O conjunto de curvas é dado pela variação de Vsb de 0 a 2V a cada 1V.



a)



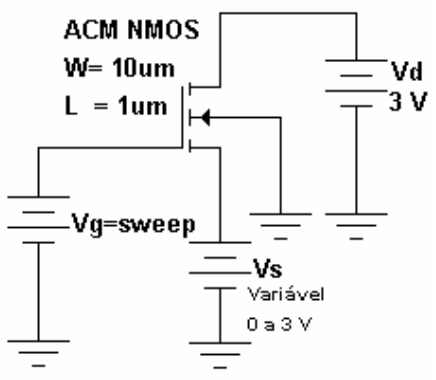
b)

Figura 4.4 - a) Circuito simulado b)Gráfico log_{10}(gmg) x Vgb com Vsb variando de 0 a 2V e Vd = 3V. Transistor NMOS W=10µm L=1µm.

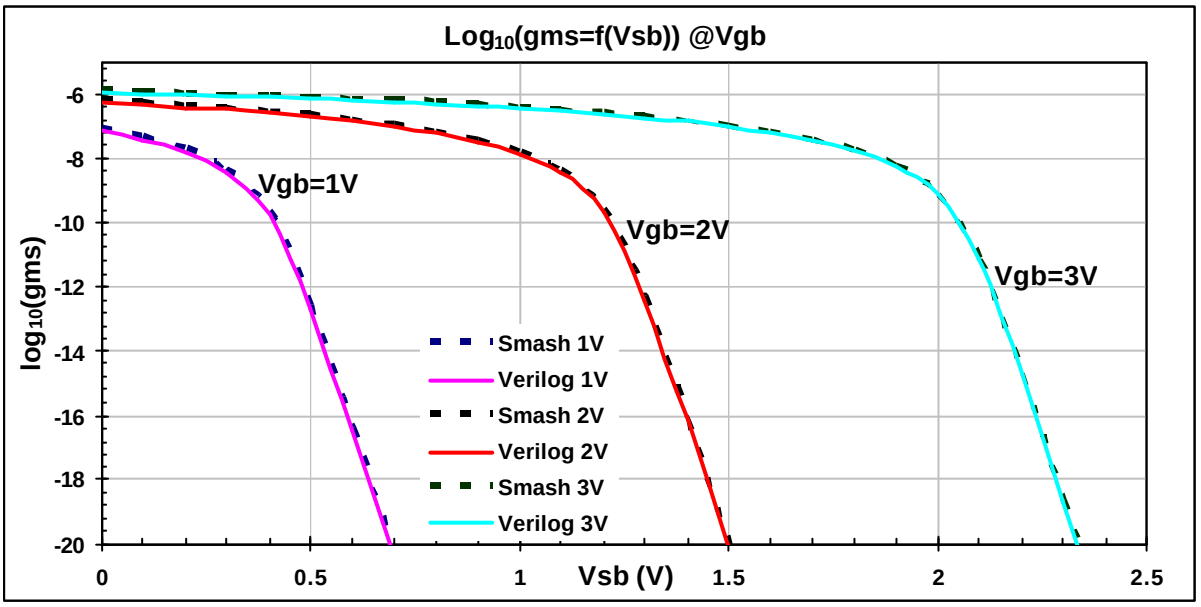
A sobreposição das curvas indica a concordância de funcionamento entre os códigos C e em Verilog-AMS.

4.1.1.4 Característica da transcondutância gms em porta comum NMOS

O logaritmo na base dez da transcondutância de fonte em função da tensão de fonte para o código em C e em Verilog-AMS do transistor NMOS na saturação é mostrado na Figura 4.5. O conjunto de curvas é obtido variando de Vgb de 1 a 3 V.



a)



b)

Figura 4.5 - a) Circuito simulado b) Gráfico $\log_{10}$(gms) x Vsb com Vgb variando de 1 a 3 V e Vd = 3V. Transistor NMOS W=10µm L=1µm.

O gráfico da simulação indica que os códigos estão gerando os mesmos valores da transcondutância gms.

4.1.2 Análise DC Aplicada em Circuitos

Este item refere-se à montagem de circuitos onde o transistor ACM escrito em Verilog-AMS é um dos componentes e o desenvolvimento destas simulações foi feito no simulador ELDO, por apresentar os recursos da linguagem implementados na versão utilizada durante o desenvolvimento deste trabalho.

A análise DC foi realizada para os circuitos: Inversor, Associação Série-Paralela e Rede Divisora de Corrente para funcionamento na região linear, utilizando o transistor MOS, modelo ACM, implementado em Verilog-AMS, com tecnologia 0,35µm com dimensões que estão indicadas no desenho esquemático do respectivo circuito.

4.1.2.1 Análise DC do inversor

Circuito clássico que permite verificar o se os transistores estão funcionando corretamente uma vez que a curva de transferência depende da correta implementação do modelo ACM dos transistores PMOS e NMOS. As dimensões dos transistores empregados nesta associação são: PMOS W= 30µm L=1µm e um ACM NMOS onde W= 10µm e L=1µm. O circuito simulado é o da Figura 4.6.

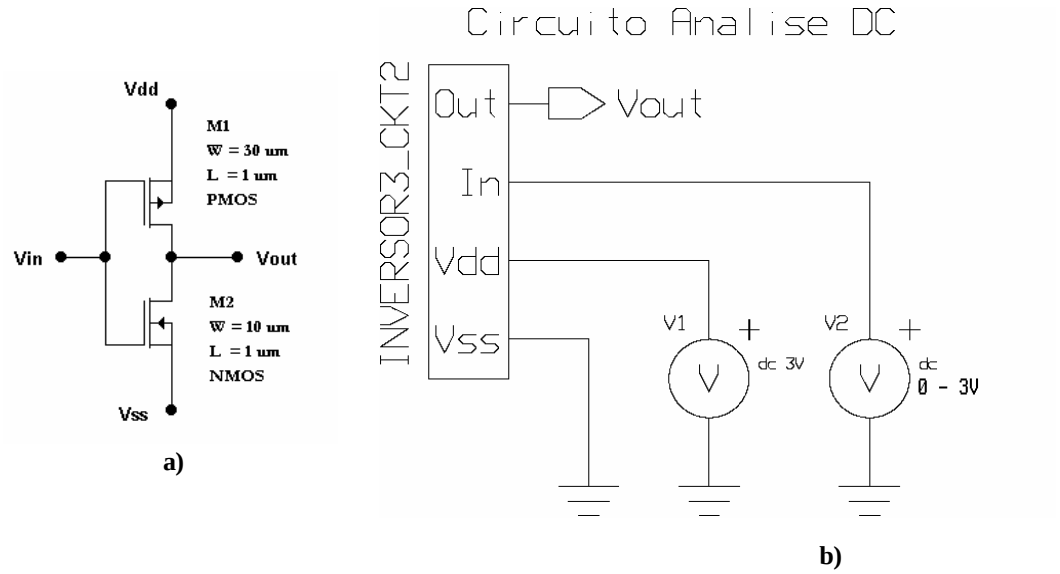


Figura 4.6 - a) Detalhe do sub-circuito inversor. b) Circuito utilizado na simulação.

O resultado da simulação no ELDO, Figura 4.7 que utiliza o código em Verilog-AMS está sendo comparado com o resultado obtido pelo código original simulado no SMASH. As curvas obtidas apresentam-se sobrepostas em grande parte das regiões de operação. A diferença que ocorre entre as curvas advém do erro constatado quando verificado o funcionamento do transistor ACM nas condições indicadas na sessão 4.1.1.1, sendo que este erro acaba se propagando ao longo do ponto de operação.

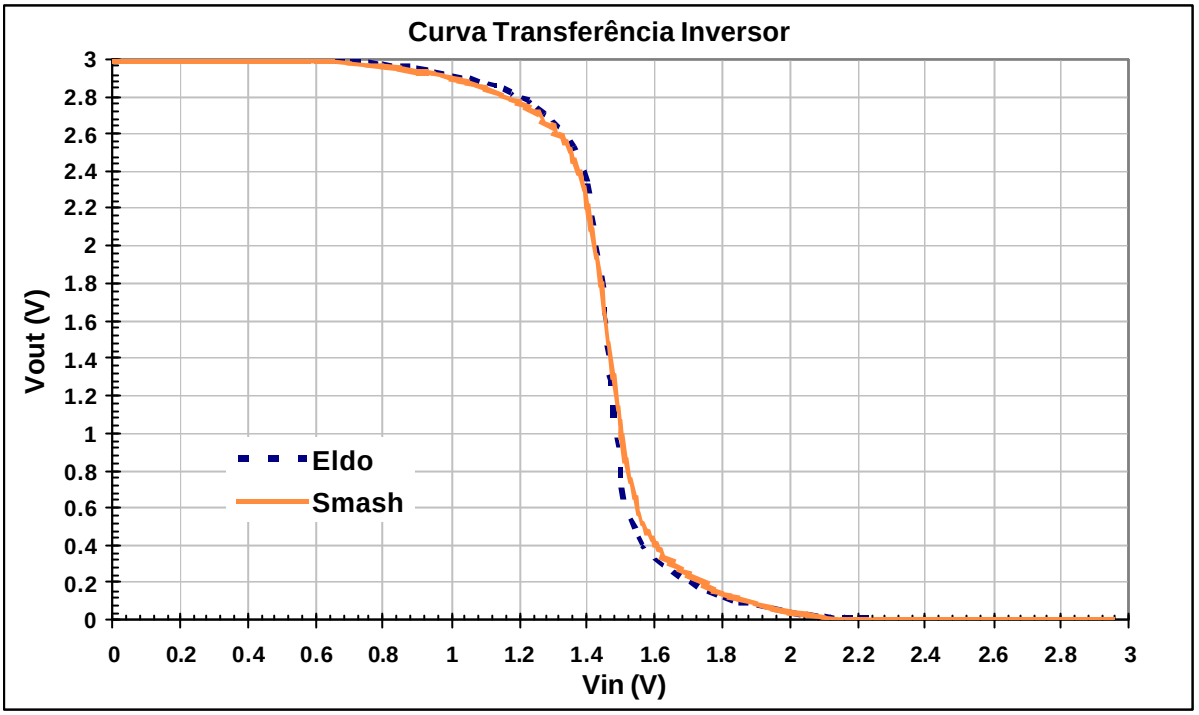


Figura 4.7 - Característica de transferência do inversor.

4.1.2.2 Análise DC da associação série-paralela

A associação série-paralela é um arranjo utilizado como uma alternativa para o projeto de circuitos analógicos [25] em que as dimensões de um único transistor não são compatíveis com as dimensões máximas permitidas para uma a tecnologia de fabricação.

A seguir apresenta-se circuito da associação série-paralelo utilizando simultaneamente transistores MOS em série e em paralelo, representado pelo sub-circuito "SERPARAL_CKT1" e um transistor MOS unitário equivalente ao do sub-circuito montado. As equações (4.1) e (4.2) relacionam as dimensões com a quantidade de transistores para resultar na largura ou comprimento equivalente a de um único transistor.

O cálculo das dimensões equivalentes fica definido pelas equações:

$$Weff = N_P.(W + DW) \tag{4.1}$$

$$Leff = N_S.(L + DL) \tag{4.2}$$

Onde N_P é a quantidade de transistores em paralelo e N_S em série e W+DW e L+DL, respectivamente, a largura e o comprimento efetivo de cada transistor.

O desenho da Figura 4.8b é composto por duas células em paralelo, sendo que a estrutura de nome SERIEPARAL_CKT1 é o sub-circuito formado pelos dezesseis transistores NMOS, modelo ACM cada um com dimensão W=10µm e L=2µm interligados conforme o esquema da Figura 4.8a, e a célula M3_CKT2 é composta por um transistor com dimensões equivalente a da estrutura série paralela. Ambas as células estão submetidas à mesma polarização DC.

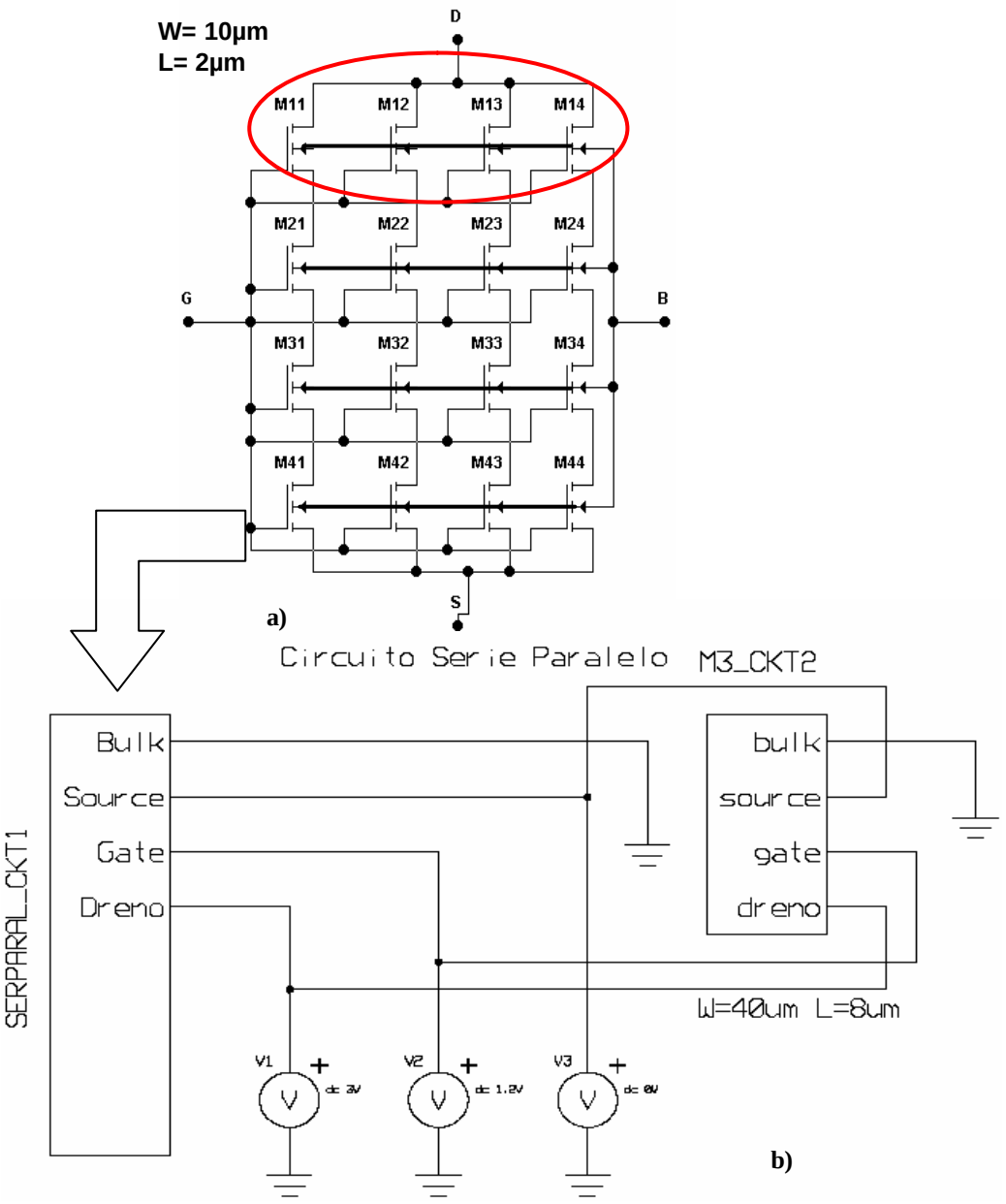


Figura 4.8 - a) Associação série-paralela constituída por 16 transistores ACM NMOS com W=10µm L=2µm. b) Estrutura simulada formada pelo sub-circuito e pelo transistor unitário equivalente W=40µm L=8µm.

Na Figura 4.9 é representada: a soma das correntes parciais dos transistores M11, M12, M13 e M14 (Eldo Soma Id_parciais) destacados na Figura 4.8a , a corrente para um transistor com dimensões equivalente ao do circuito série paralelo simulado no ELDO (Eldo Id M3) e a corrente no transistor modelado em Verilog-AMS com dimensões equivalentes, simulado no SMASH.

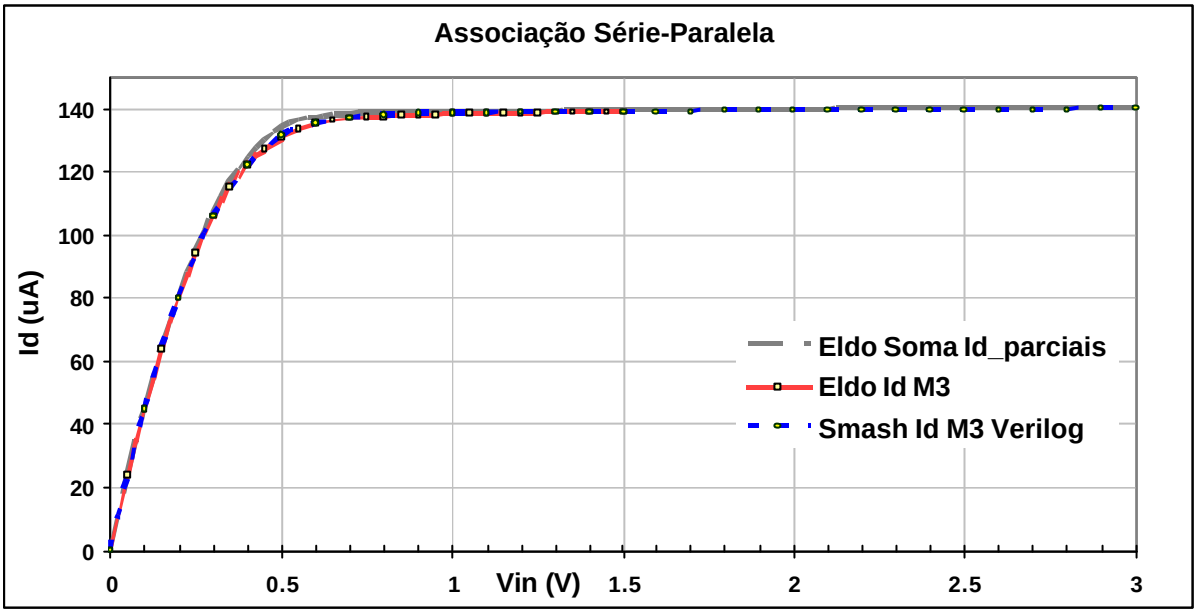


Figura 4.9 - Comparação entre a soma das correntes parciais relativas aos 4 transistores em paralelo a e a corrente do transistor equivalente M3 da Figura 4.8.

Nota-se a sobreposição dos resultados em grande parte da extensão da curva, havendo uma diferença no joelho do resultado da soma das correntes parciais (Eldo Soma Id_parciais), justificada pelo fato de que esta curva é o resultado da soma das correntes parciais de cada transistor, que não estão submetidos aos efeitos de canal curto, ao contrário dos outros dois resultados.

4.1.2.3 Rede divisora de corrente para operação na região linear

O circuito representado pela Figura 4.10 é utilizado em circuitos D/A e A/D, filtros programáveis entre outros sendo sua função é realizar a divisão binária das correntes a partir da corrente de referência (Iref). O resultado desta simulação, usando o circuito da Figura 4.11 é apresentado por meio da normalização entre o valor da corrente parcial obtida e o valor de

referência. O objetivo desta análise é além de comprovar a equivalência entre o código em C e o em Verilog-AMS é constatar que para pequenas intensidades de corrente e, pela a simetria entre o dreno e a fonte a corrente não sofre variações significativas para pequenas intensidades.

A simulação deste teste foi feita para o código em Verilog-AMS no simulador ELDO, sendo o resultado expresso pela Figura 4.12 e o código em C foi realizada no simulador SMASH.

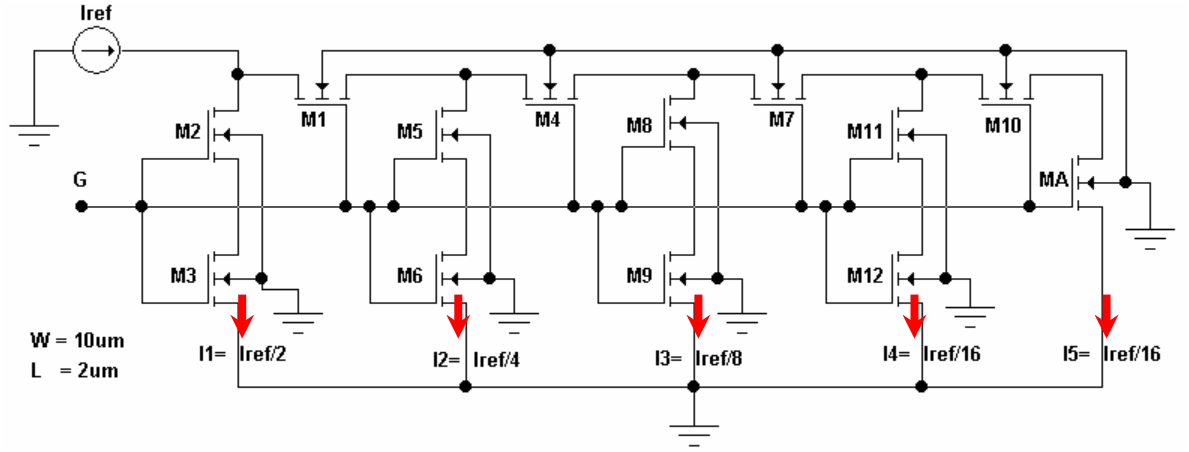


Figura 4.10 - Esquemático da interligação dos transistores código ACM para operação na região linear.

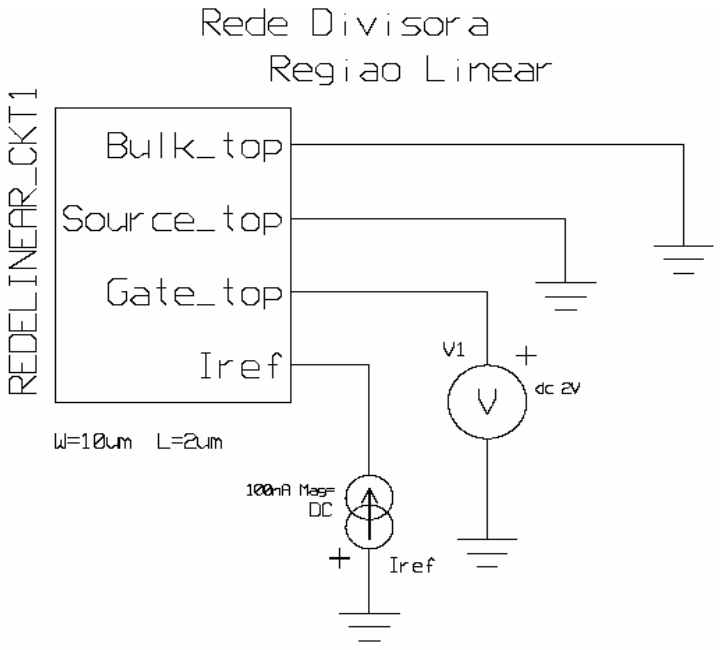


Figura 4.11 - Circuito simulado compondo a rede divisora para operar na região linear.

A validação deste teste ocorreu pela imposição de onze valores para a corrente de referência (Iref) no intervalo entre 100nA e 100µA. Para cada valor da corrente de referência, a análise DC foi feita, resultando em valores de correntes parciais (I1 a I5) muito próximos

aos indicados na Figura 4.10. Tais valores foram normalizados e apresentados nos gráficos que seguem.

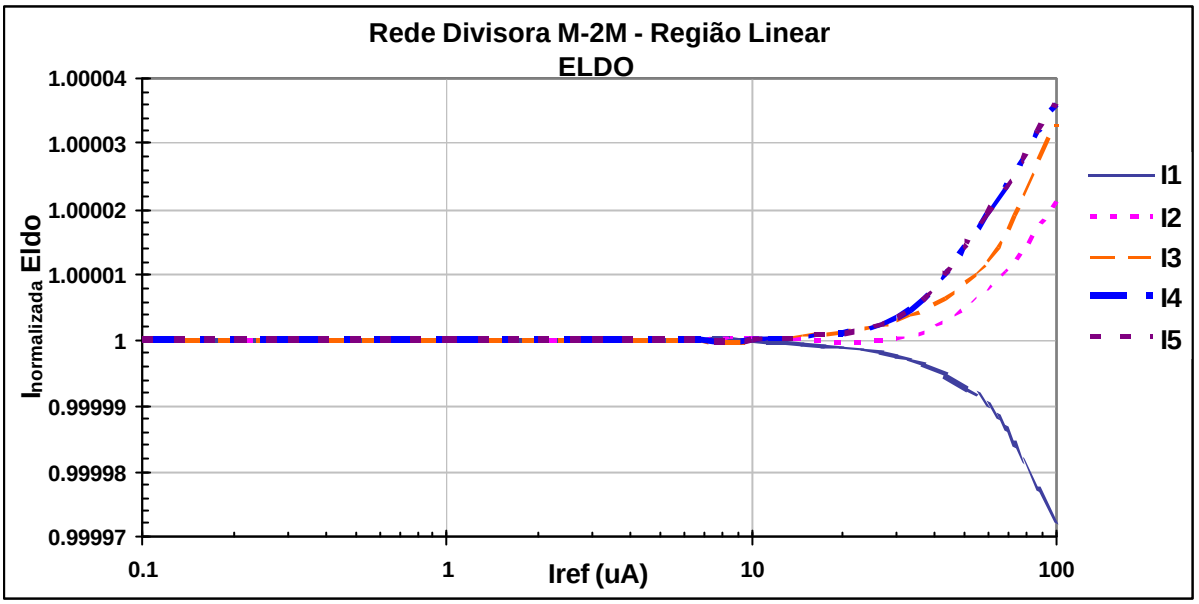


Figura 4.12 - Gráfico das correntes normalizadas ($I_{simulada}/I_{ref}$) para o circuito da rede divisora linear simulada no ELDO.

Simulação realizada no simulador SMASH, utilizando o código em C, o resultado é apresentado na Figura 4.13.

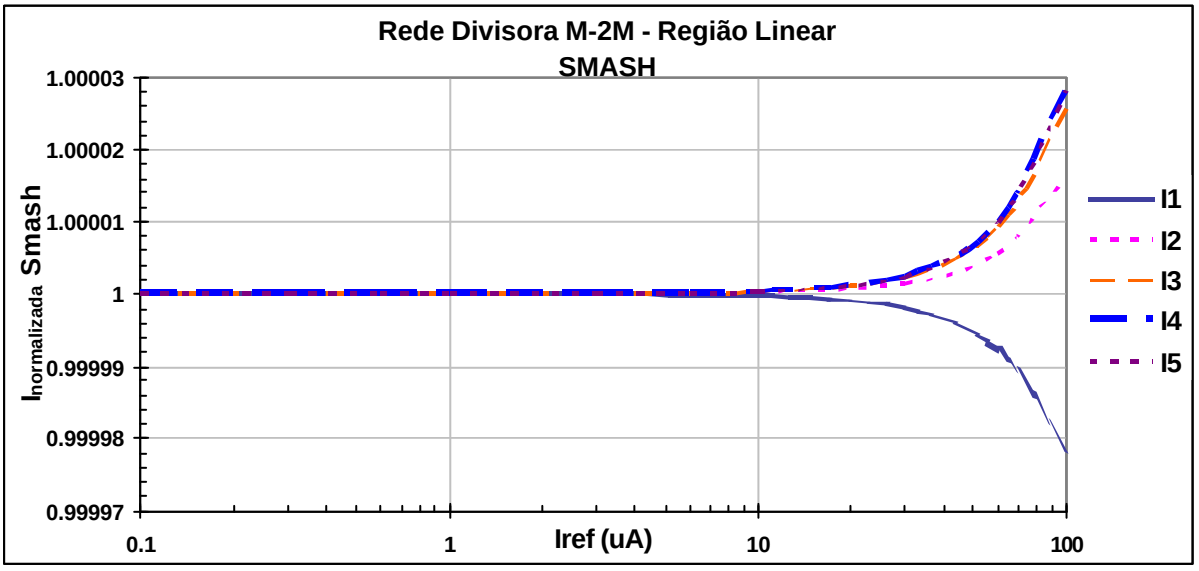


Figura 4.13 - Gráfico das correntes normalizadas ($I_{simulada}/I_{ref}$) para o circuito da rede divisora linear simulada no SMASH.

Os resultados obtidos são consistentes tendo em vista que há a coincidência dos resultados obtidos.

4.1.3 Análise Transiente

A análise transiente permite verificar se o modelo das cargas e capacitâncias está corretamente implementado pela linguagem, ou seja, o cálculo das derivadas das cargas que no modelo escrito em linguagem C foi desenvolvido explicitamente é substituído pelo uso do operador *time derivative* – ddt( ) disponível na linguagem Verilog-AMS. Este cálculo é feito diretamente pelo simulador tendo como passo a base de tempo, como o modelo ACM baseia-se na conservação das cargas e as equações são contínuas para todas as regiões de operação, o processamento ocorre por meio da regra da cadeia aplicada às equações da corrente no tempo e da capacitância intrínseca conforme o conjunto de equações (3.1). Os resultados das simulações comprovam que o processamento da HDL ocorre da forma esperada, isto porque o bloco que calcula as capacitâncias por meio das derivadas explícitas da carga que existe no código C, foi retirado quando o código foi escrito em Verilog-AMS.

As simulações transientes foram realizadas utilizando o método trapezoidal de integração numérica, aplicada nos circuitos: inversor, amostragem e retenção, capacitor chaveado e na comprovação da simetria das capacitâncias Cgd e Cgs.

4.1.3.1 Análise transiente do inversor

Por ser um circuito de simples construção e análise, o inversor foi novamente utilizado para análise transiente sendo aplicado na entrada uma fonte de tensão pulsada - Vin conforme representação na Figura 4.14.

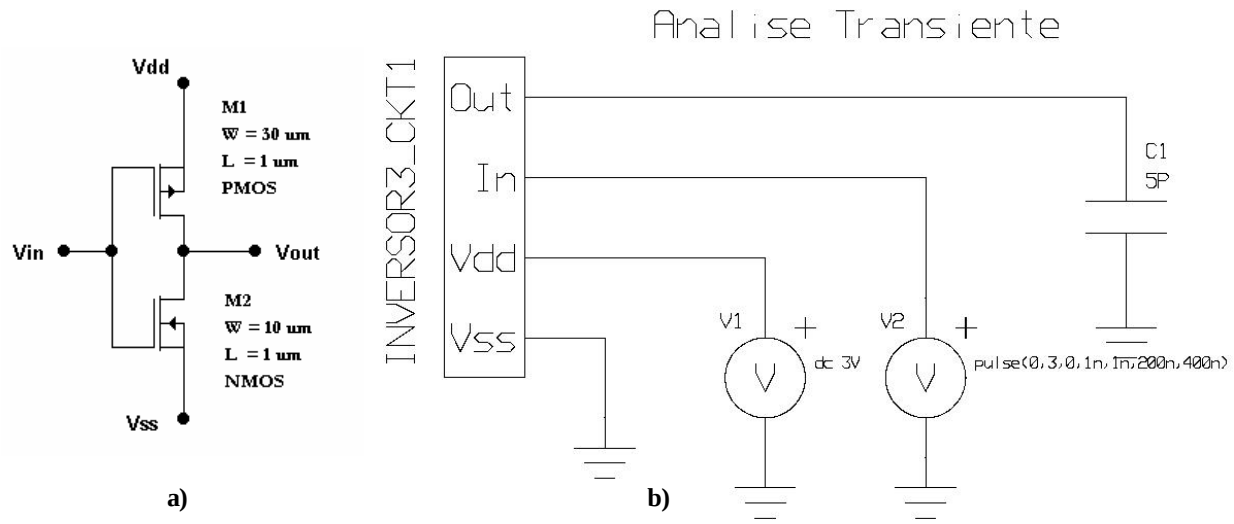


Figura 4.14 - a) Detalhe do sub-circuito do inversor: M1=PMOS W=30µm L=1µm e M2=NMOS W=10µm L=1µm. b) Circuito simulado para análise transiente.

O resultado obtido na saída corresponde à inversão do sinal aplicado à entrada, conforme pode ser observado na Figura 4.15.

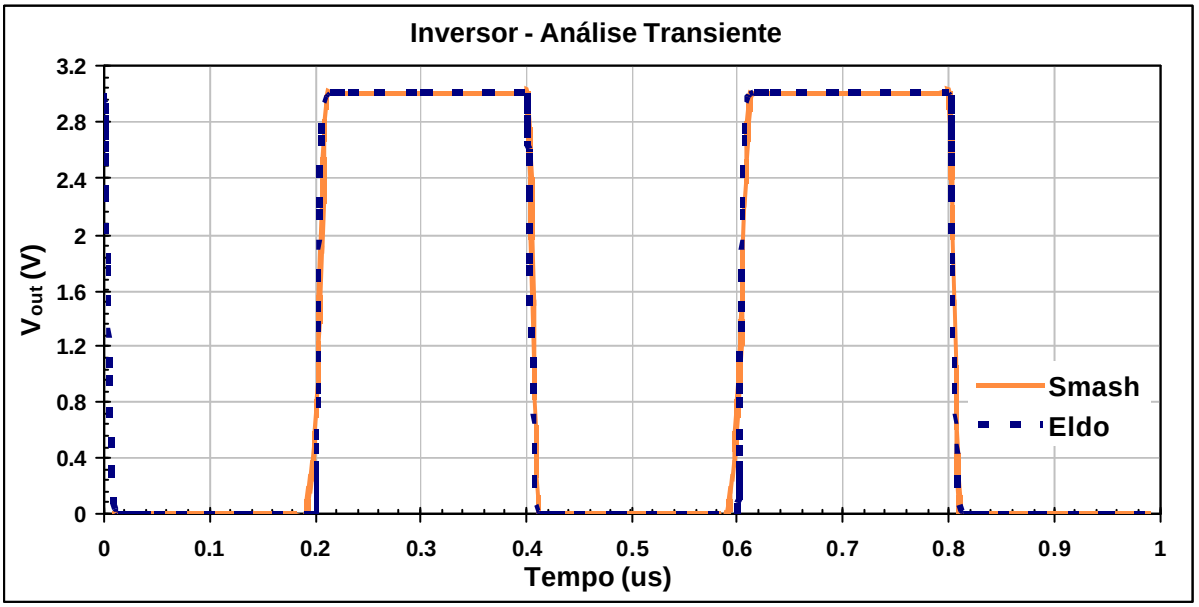


Figura 4.15 - Tensão na carga C1=5pF do circuito inversor submetido a uma fonte de pulsos de período 400ns.

O mesmo circuito da Figura 4.14 teve o capacitor alterado para 1pF e a fonte passou para período igual a 10ns. O resultado da simulação é apresentado na Figura 4.16.

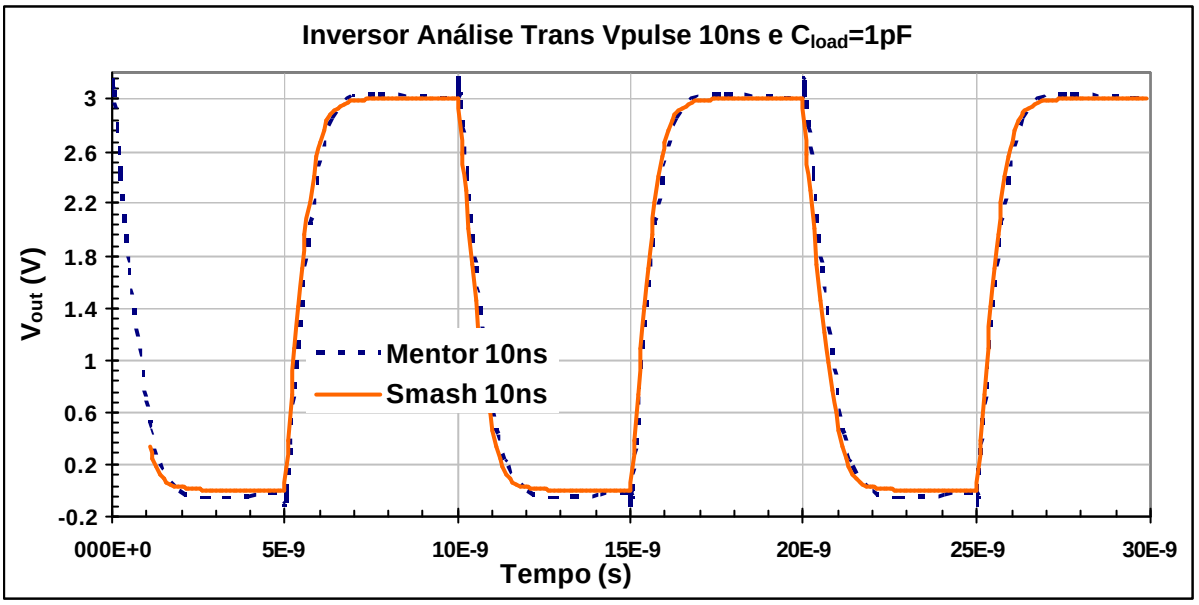


Figura 4.16 - Tensão na carga C1=1pF do circuito inversor submetido a uma fonte de pulsos de período 10ns.

Mesmo em condições desfavoráveis de funcionamento, o circuito apresenta resultados compatíveis com a faixa de freqüências a que foi submetido.

4.1.3.2 Circuito de amostragem e retenção

Esta análise foi realizada em um circuito básico de amostragem e retenção que permite verificar a conservação de carga do modelo ACM. O transistor usado para esta simulação foi um NMOS com W=100µm e L=15µm. A simulação consiste no capacitor inicialmente descarregado submetido a um trem de pulsos aplicado no terminal de porta do transistor ACM, de modo que cada pulso transfere-se uma quantidade de carga para o capacitor. A estrutura do circuito em questão é representada na Figura 4.17.

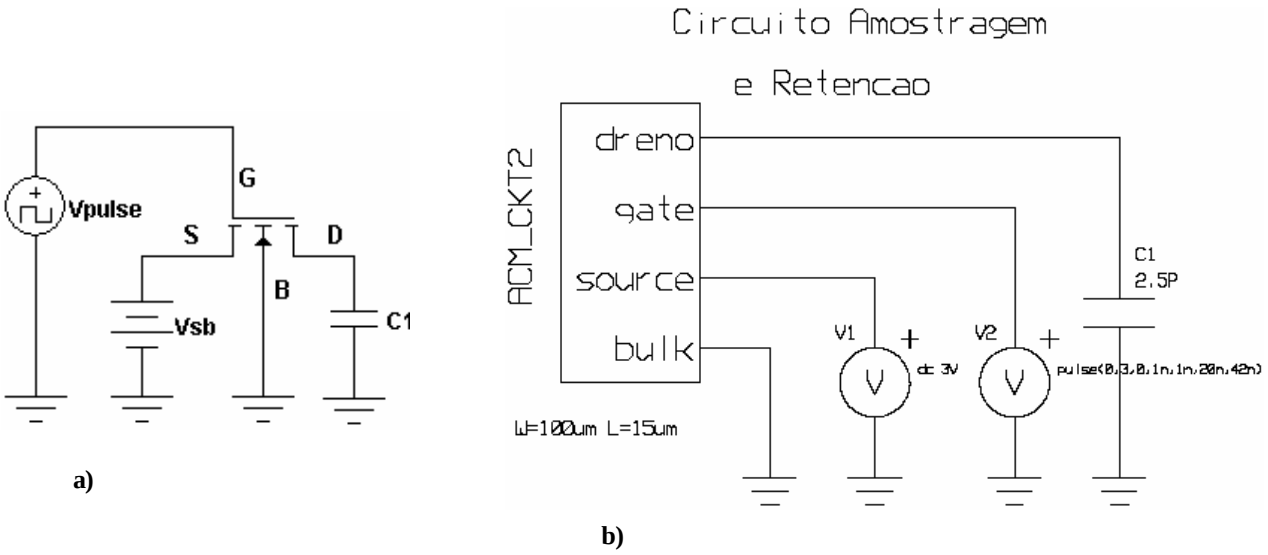


Figura 4.17 - a) Detalhe do circuito de amostragem e retenção. b) Circuito usado para realização da simulação.

O resultado para a análise transiente apresentado na Figura 4.18, resultou no comportamento esperado para o respectivo circuito, cabendo ressaltar de que a implementação das derivadas no código é feita pelo uso do operador ddt( ), confirmando a correta implementação.

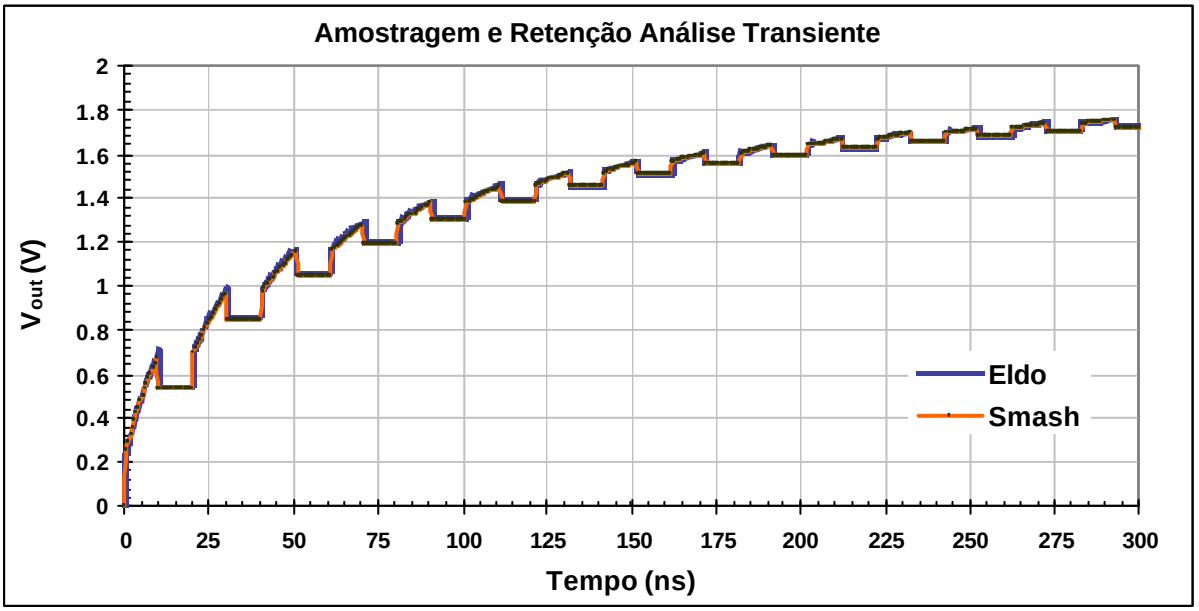


Figura 4.18 - Tensão de saída do circuito da Figura 4.17.

4.1.3.3 Capacitor chaveado

Este circuito mostra a forma com que ocorre a distribuição das cargas ao longo do circuito formado pelos dois transistores. As entradas definidas pelos terminais de porta de cada um dos transistores ficam submetidos a tensões pulsadas, sendo elas defasadas entre si. A simulação foi realizada para dois valores de capacitâncias de carga C_{load} = 5pF na ordem de grandeza da capacitância de porta e C_{load} = 20pF, valor muito superior a da capacitância de porta. O circuito simulado está na Figura 4.19.

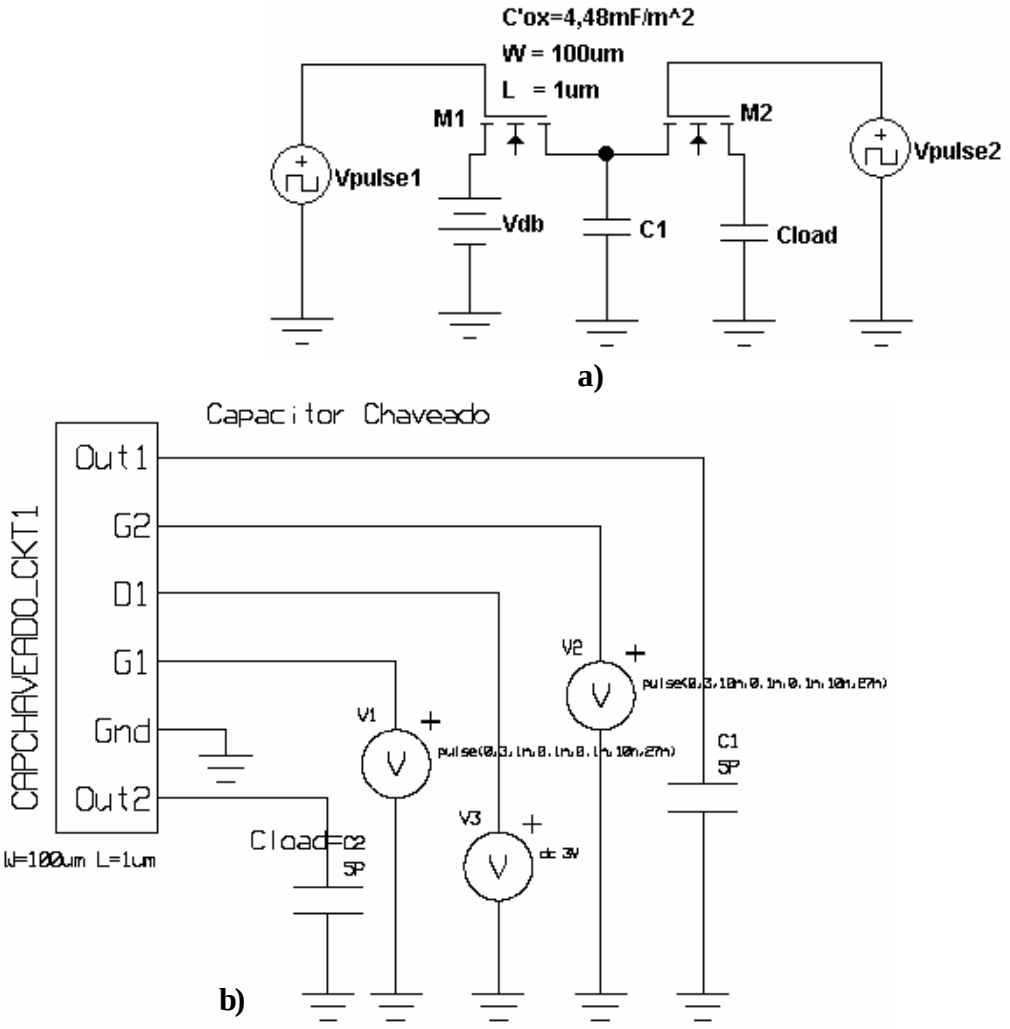


Figura 4.19 - a) Detalhe do circuito do capacitor chaveado. b) Circuito utilizado na simulação do capacitor chaveado.

O resultado para a distribuição das cargas para C_{load} = 5 pF, é apresentado na Figura 4.20.

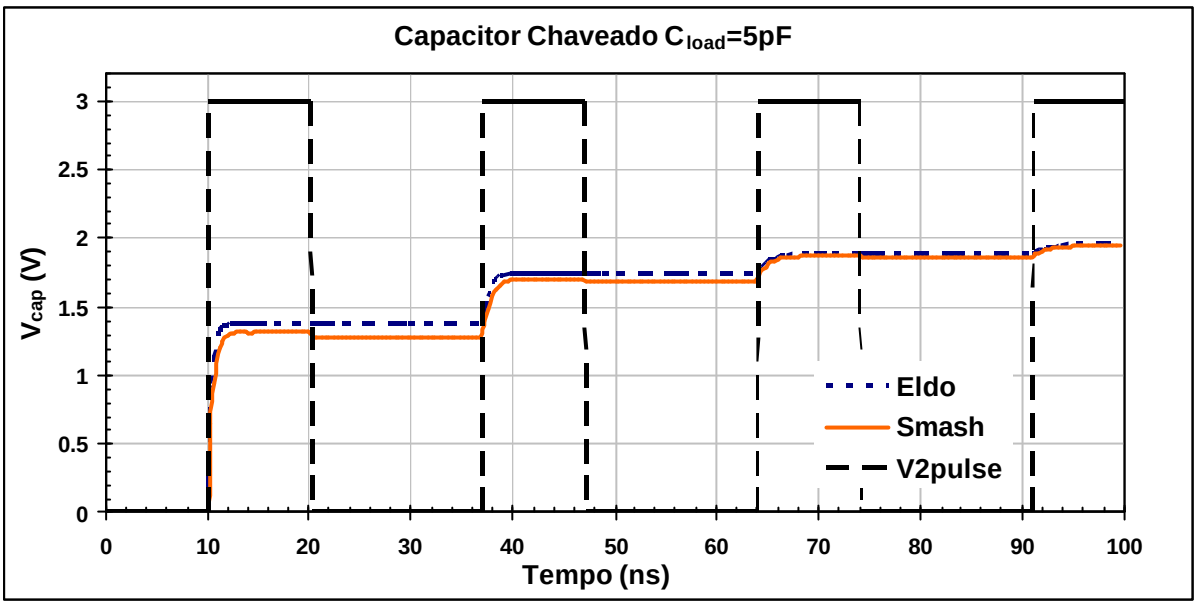


Figura 4.20 -Tensão no capacitor C_{load} de valor 5pF.

O resultado para a distribuição das cargas para C_{load} = 20 pF, é apresentado na Figura 4.21.

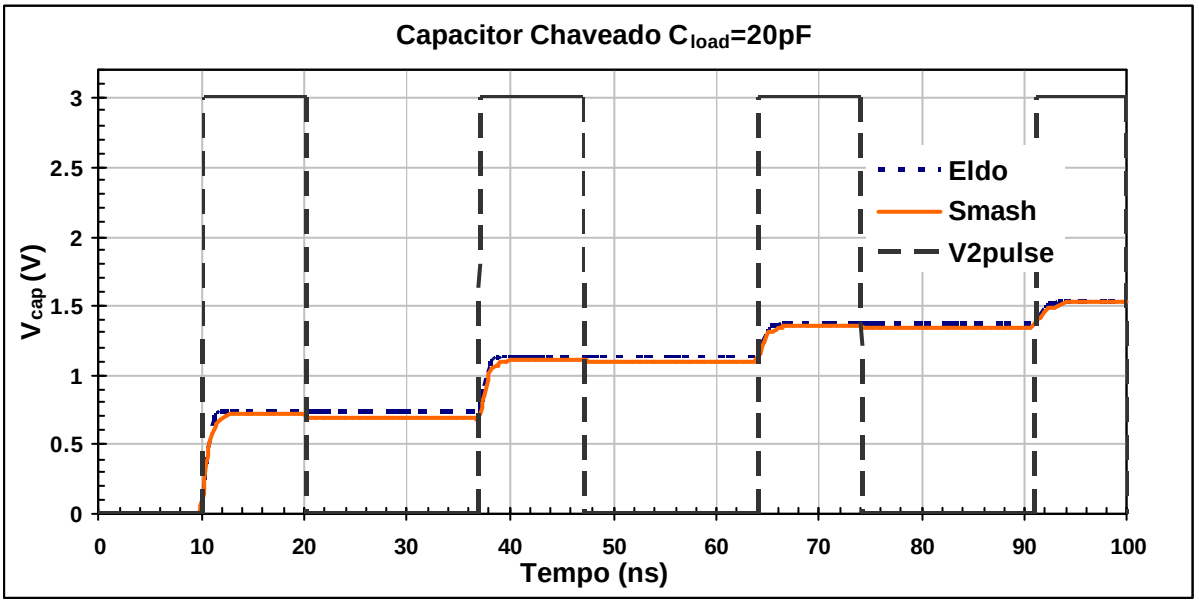


Figura 4.21 - Tensão no capacitor de C_{load} de valor 20pF.

A comparação entre as simulações resulta em comportamentos semelhantes entre o modelo implementado e o original, sendo que as diferenças verificadas, atribui-se ao erro numérico.

4.1.3.4 Simetria das capacitâncias Cgd e Cgs

A simulação foi realizada para a condição inicial em que o capacitor conectado ao terminal de dreno esteja carregado com 3V, e o capacitor da fonte descarregado. Aplica-se um trem de pulsos de curto intervalo a fim de verificar a igualdade entre as tensões Vd e Vs. O transistor utilizado NMOS W=10µm L=1.5µm. O esquema e o circuito relativos a este teste são indicados na Figura 4.22 abaixo.

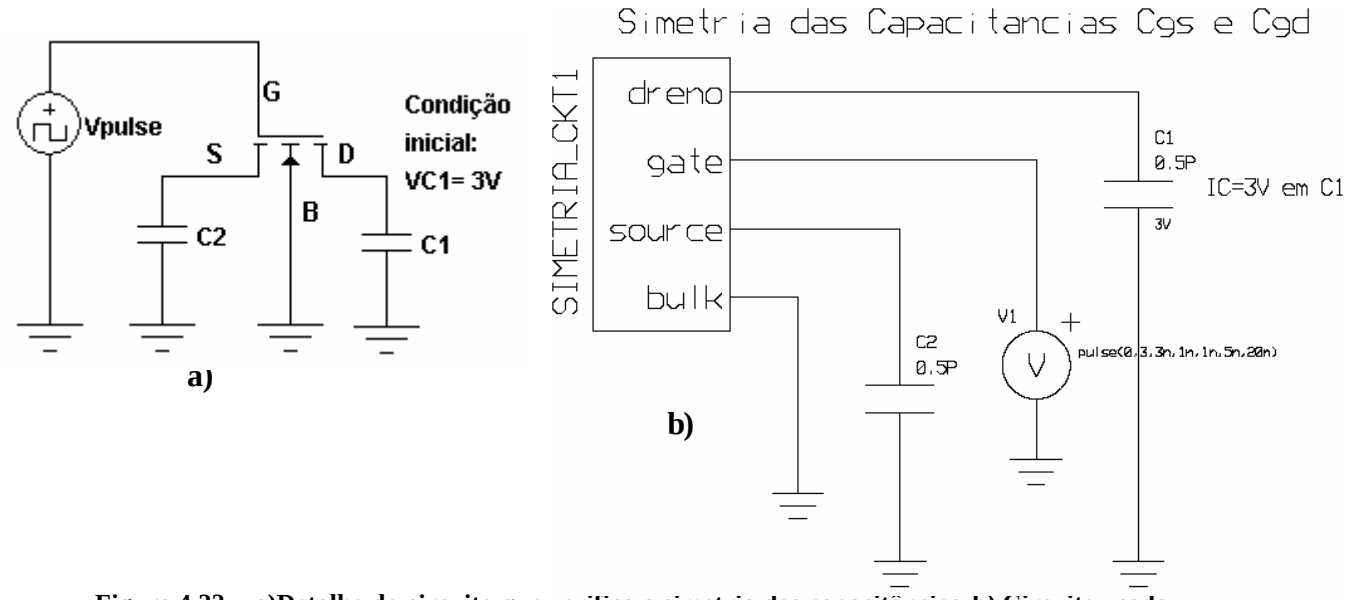


Figura 4.22 - a)Detalhe do circuito que verifica a simetria das capacitâncias. b) Circuito usado para simular o teste simetria das capacitâncias.

A seguir estão apresentados os resultados obtidos pela simulação com o código em Verilog-AMS (ELDO) e com o código original (SMASH).

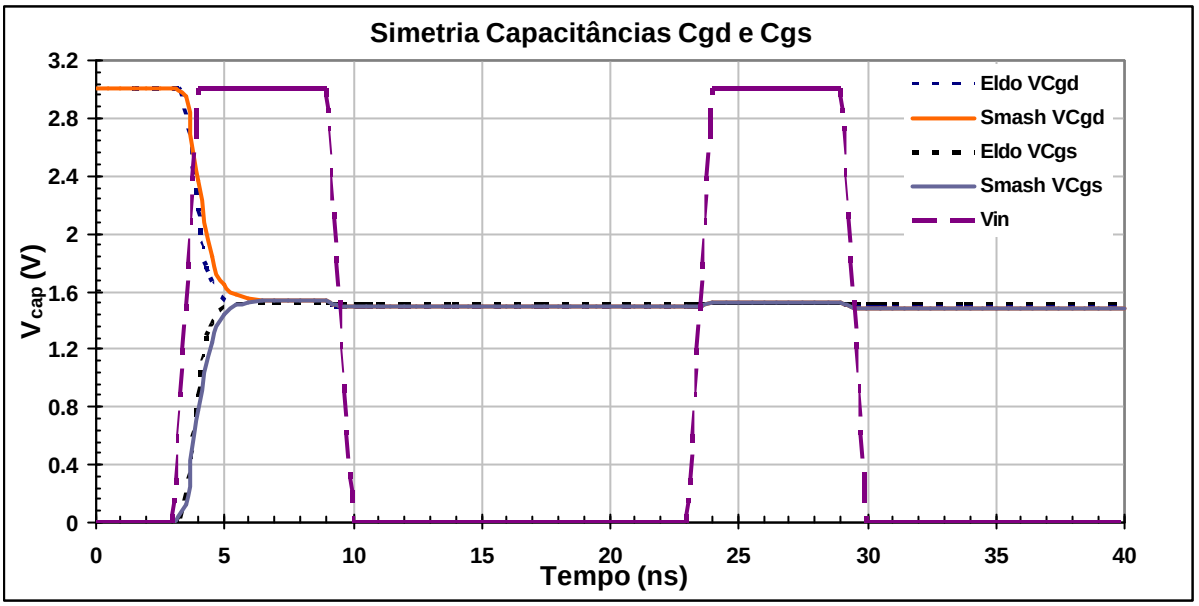


Figura 4.23 - Composição dos resultados obtidos pela simulação do código em Verilog-AMS e o código C.

O resultado obtido pela simulação apresentado na Figura 4.23 indica a coincidência dos resultados entre os códigos em Verilog-AMS e o código C.

4.1.4 Análise AC

A verificação da análise AC foi realizada pela simulação do circuito do inversor.

4.1.4.1 Análise AC do inversor

Esta simulação foi feita com uma fonte AC de módulo 1V e fase 0º e nível DC igual ao obtido pela análise DC representada pela Figura 4.6 e resultado na Figura 4.7. O circuito simulado está a seguir representado (Figura 4.24). A fim de verificar o comportamento do circuito em condições mais próximas às reais, um resistor de valor 1kO foi colocado em série com a fonte AC. O resultado do módulo em dB desta simulação está representado na Figura 4.25 e a fase na Figura 4.26 para as duas condições de funcionamento do circuito.

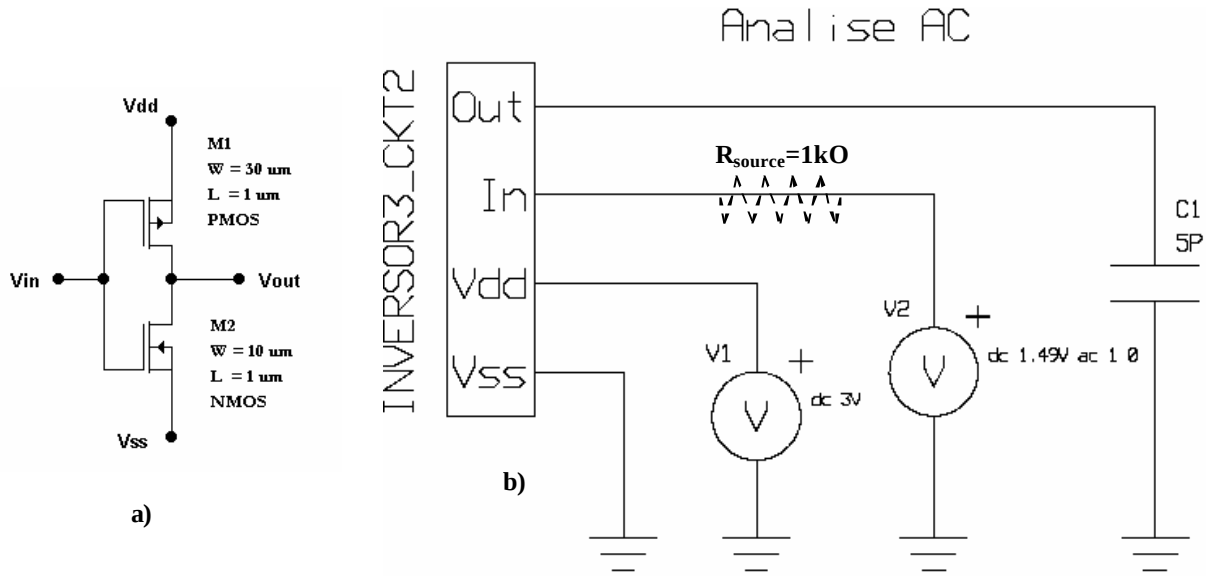


Figura 4.24 - a) Circuito do inversor b) Circuito usado na simulação da análise AC do inversor. O símbolo do resistor tracejado indica uma segunda condição de operação.

O resultado do módulo em dB da análise AC realizada pelo código em Verilog-AMS e o código C é apresentado na Figura 4.25.

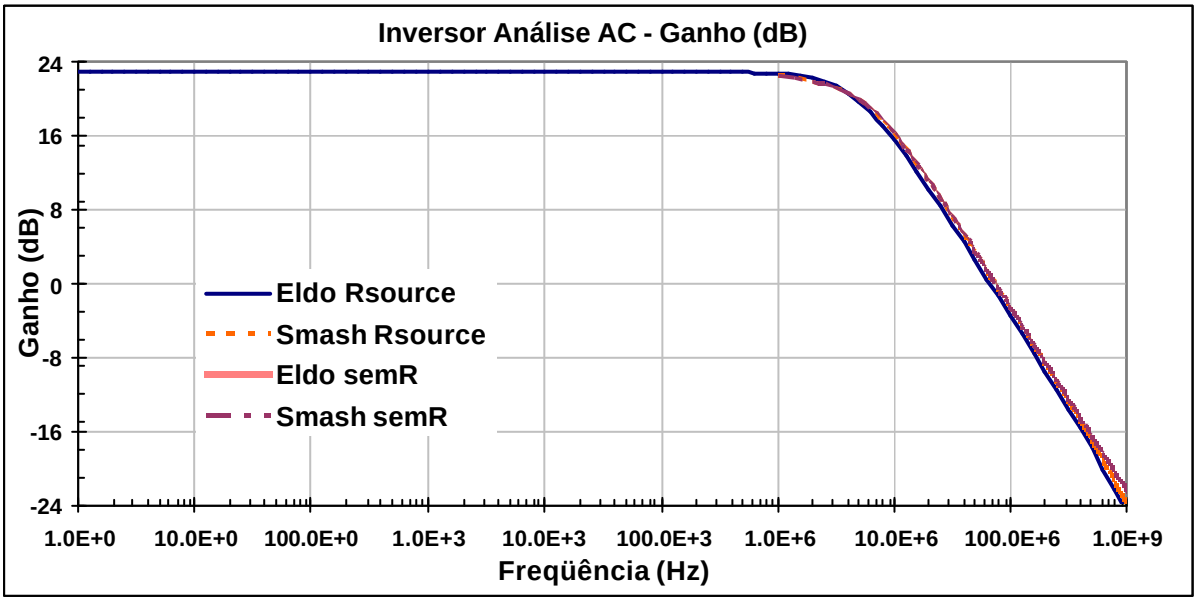


Figura 4.25 - Resultado do módulo em dB da análise de pequenos sinais no circuito inversor.

Resultado de como a fase se comporta para as situações de operação com o resistor de fonte (R_{source}) e sem o resistor é mostrado na Figura 4.26.

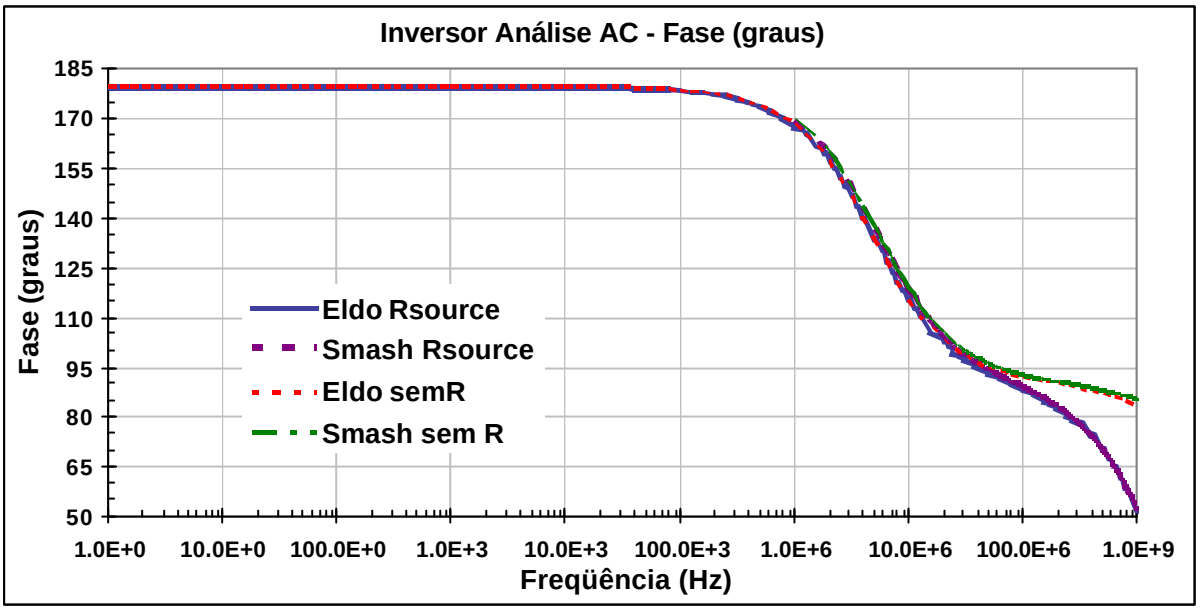


Figura 4.26 Resultado da fase da análise de pequenos sinais no circuito inversor.

Os resultados das condições de funcionamento com e sem resistor, apresentam-se de forma semelhante tanto em módulo como na fase, havendo uma alteração na fase nas altas freqüências (a partir de aproximadamente 30MHz).

4.1.5 Ruído Térmico e 1/f

A formulação matemática para a o cálculo da densidade espectral de potência teve como referência [21], [22] e o memorial de cálculo apresentado no item 2.2.6. A linguagem Verilog-AMS dispõem de funções dedicadas para esta análise, facilitando o cálculo do ruído térmico e 1/f. O circuito usado para a simulação do ruído está na Figura 4.27 abaixo a seguir.

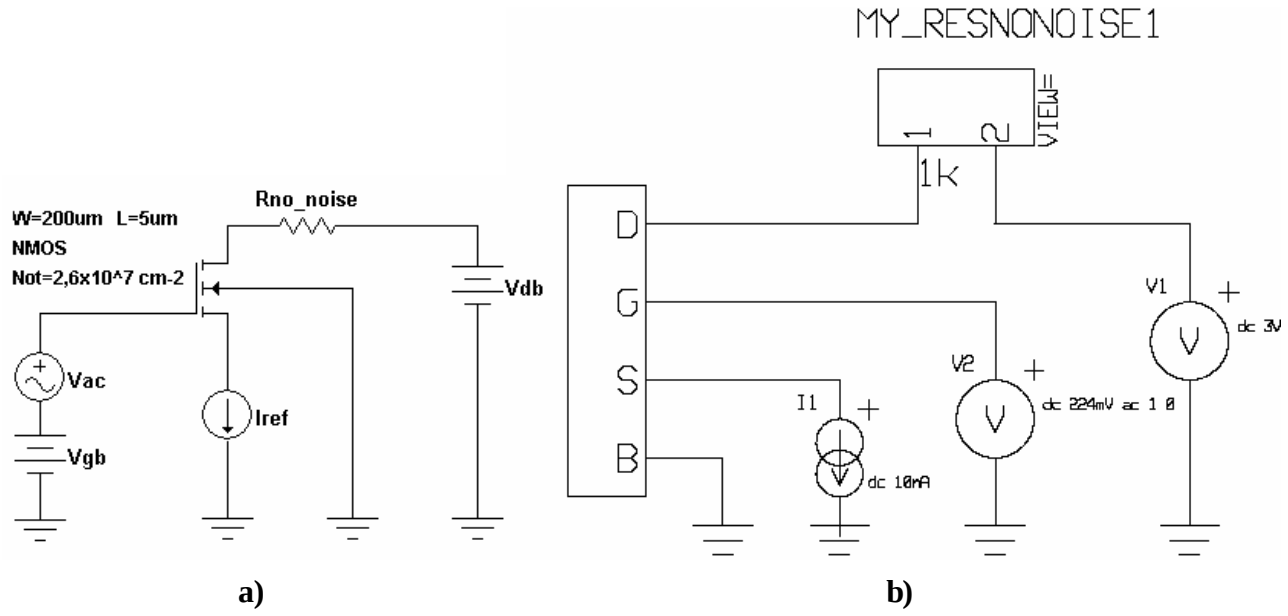


Figura 4.27 - a)Detalhe do circuito para determinação do ruído. b) Circuito utilizado para realização a simulação do ruído térmico e 1/f. W=200µm L=5µm NMOS.

4.1.5.1 Metodologia usada para a obtenção dos resultados finais do ruído

Conforme apresentado na sessão 2.2.6, foram usados como referência de comparação os resultados apresentados por [21], porém o modelo do transistor empregado na respectiva simulação foi o SPICE NLEV=2,3 , W=200µm L=5µm, N_{ot}=2,6x10^{7} cm^{-2} , com os parâmetros de entrada não divulgados.

O transistor simulado nesta etapa foi um modelo ACM NMOS saturado, com mesmas dimensões ao do que tomou-se como referência.

Os resultados gráficos foram obtidos por meio de dois procedimentos: O primeiro deles foi a obtenção da curva teórica calculada aplicada às equações (2.24) e (2.25) dos ruídos térmico e 1/f respectivamente utilizando o MatLab. O segundo procedimento foi estruturado nos resultados das simulações feitas no simulador ELDO, utilizando o código em Verilog-AMS.

1. Neste cálculo teórico, usando o Matlab, adotou-se para efeito de simplificação, o fator de rampa (*n*) igual a 1,3 , banda de freqüência igual a 1Hz, e como o transistor deve estar saturado, o valor adotado para *qid* foi zero. Na equação (4.3) retirada de [23], a variável Q'_{IS} foi isolada e reescrita como uma função do segundo grau conforme (4.4).

$$I_F = m_{ef}.n.C'_{OX}.\frac{W}{L}.\frac{f_t^{\,2}}{2}\left[\left(\frac{Q'_{IS}}{n.C'_{OX}.f_t}\right)^2 - 2.\left(\frac{Q'_{IS}}{n.C'_{OX}.f_t}\right)\right] \qquad (4.3)$$

$$\left(\frac{Q'_{IS}}{n.C'_{OX}.f_t}\right)^2 - 2.\left(\frac{Q'_{IS}}{n.C'_{OX}.f_t}\right) - \frac{I_F}{m.n.C'_{OX}.\frac{W}{L}.\frac{f_t^{\,2}}{2}} = 0 \qquad (4.4)$$

Onde:

I_F = corrente direta de saturação (A)

μ_{ef} = mobilidade efetiva do portador (m^2/V.s)

n = fator de rampa

$C'ox$ = capacitância do óxido por unidade de área (F/m^2)

W = largura do canal (µm)

L = comprimento do canal (µm)

f_t = potencial termodinâmico (V)

Q'_{IS} = densidade de carga da fonte (C/m^2)

2. O valor de Q'_{IS} foi calculado e substituído nas equações (2.24) e (2.25) que determina o ruído térmico e 1/f respectivamente, com isto as curvas individuais foram obtidas por meio da variação da corrente de referência no intervalo entre 1nA e 10mA.
3. O valor do ruído total foi calculado por meio da equação (4.5):

$$Si_{total} = \sqrt{Sid^2 + Siw^2} \qquad (4.5)$$

Os procedimentos de 1 a 3 foram realizados para a obtenção das curvas relativas aos valores teóricos calculados no Matlab, tendo como referência as equações (2.24) e (2.25).

A simulação do código em Verilog-AMS no simulador ELDO ocorre a partir do procedimento 4.

4. O circuito da Figura 4.28 foi empregado no simulador ELDO, para quinze valores de corrente de referência no intervalo entre 1nA e 10mA. Nesta etapa os resultados da simulação representam a composição do ruído térmico e do ruído 1/f, gerando o valor do S_{itotal}.

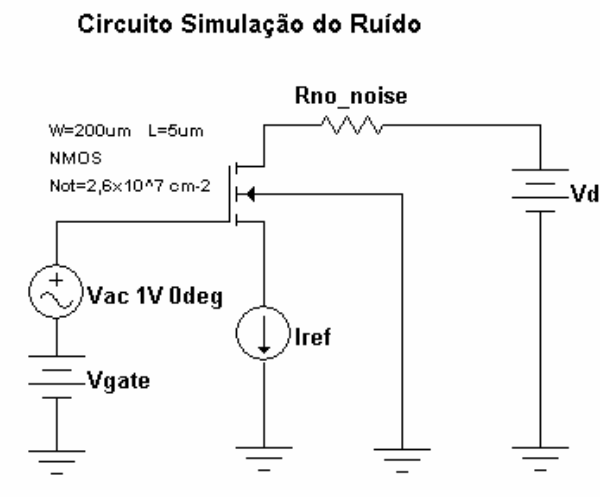


Figura 4.28 - Circuito usado para realização da simulação do ruído térmico e 1/f.

5. Repetiu-se o procedimento da etapa 4, porém foi imposta a condição de que o valor da variável N_{ot} ser igual zero na equação do ruído 1/f (2.25), tal situação implicou no valor de SPD relativa a 1/f ser igual a zero, portando o resultado apresentado na simulação passou a ser uma função exclusiva do ruído térmico S_{iw}.

6. O valor relativo ao ruído 1/f, foi obtido por meio da equação (4.6).

$$Sid = \sqrt{Si_{total}^{\ 2} - Siw^2} \qquad (4.6)$$

Os resultados dos procedimentos de 1 a 6 foram agrupados em gráficos que indicam a condição do cálculo e da simulação para os ruídos térmico (Figura 4.29), 1/f (Figura 4.30) e total (Figura 4.29) respectivamente.

Gráfico do ruído térmico calculado e simulado, sendo o valor simulado obtido igualando o parâmetro N_{ot} a zero na equação (2.25).

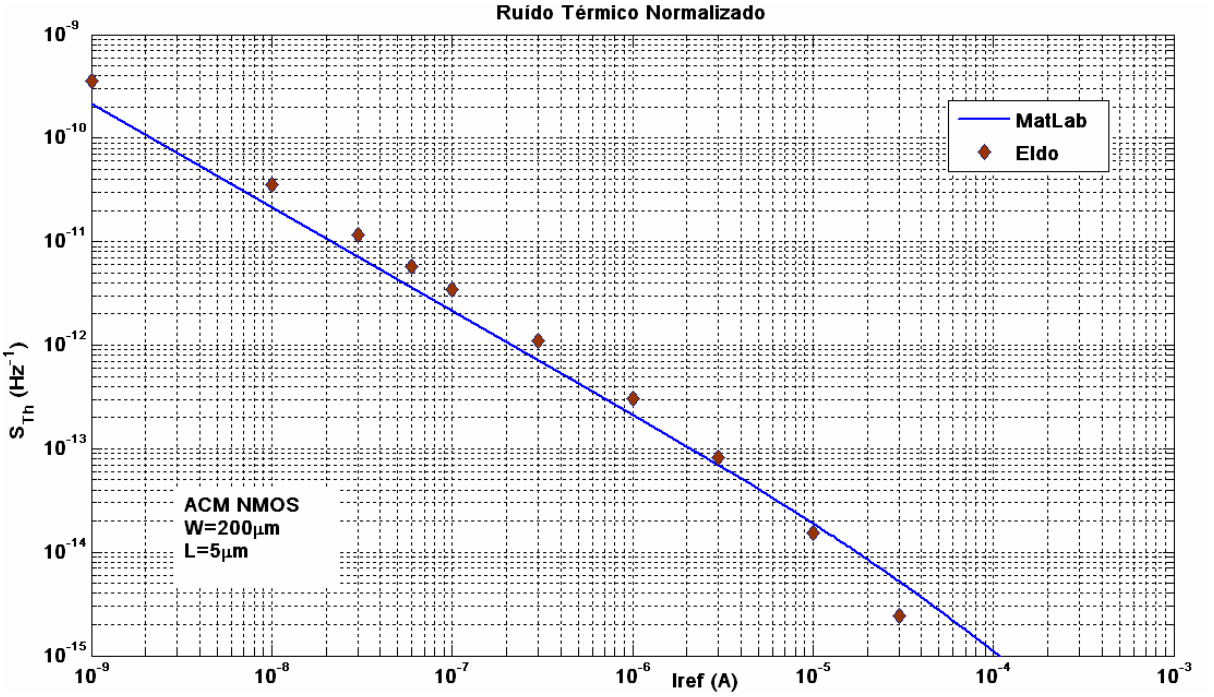


Figura 4.29 - Curva do ruído térmico normalizado para modelo ACM NMOS saturado W = 200µm L=5µm.

Gráfico do ruído 1/f calculado e simulado a partir do cálculo utilizando (4.6).

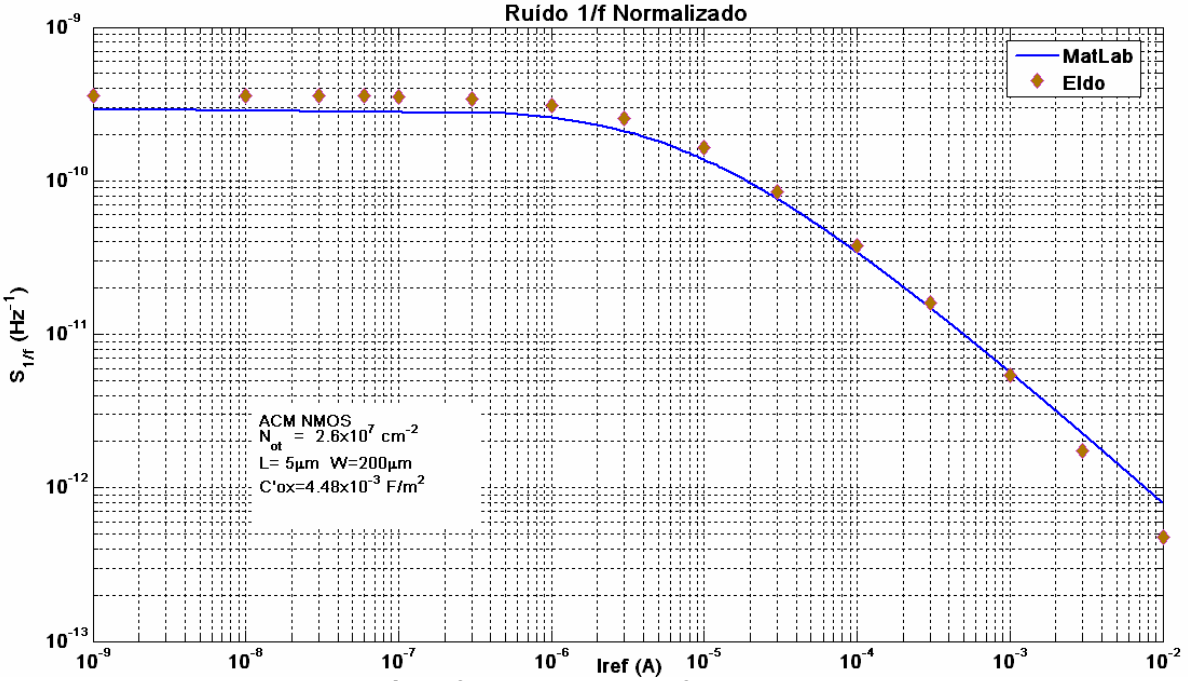


Figura 4.30 - SPD do ruído 1/f normalizado para f = 1Hz para o transistor ACM NMOS saturado – W = 200µm L = 5 µm.

Gráfico do ruído total calculado e simulado.

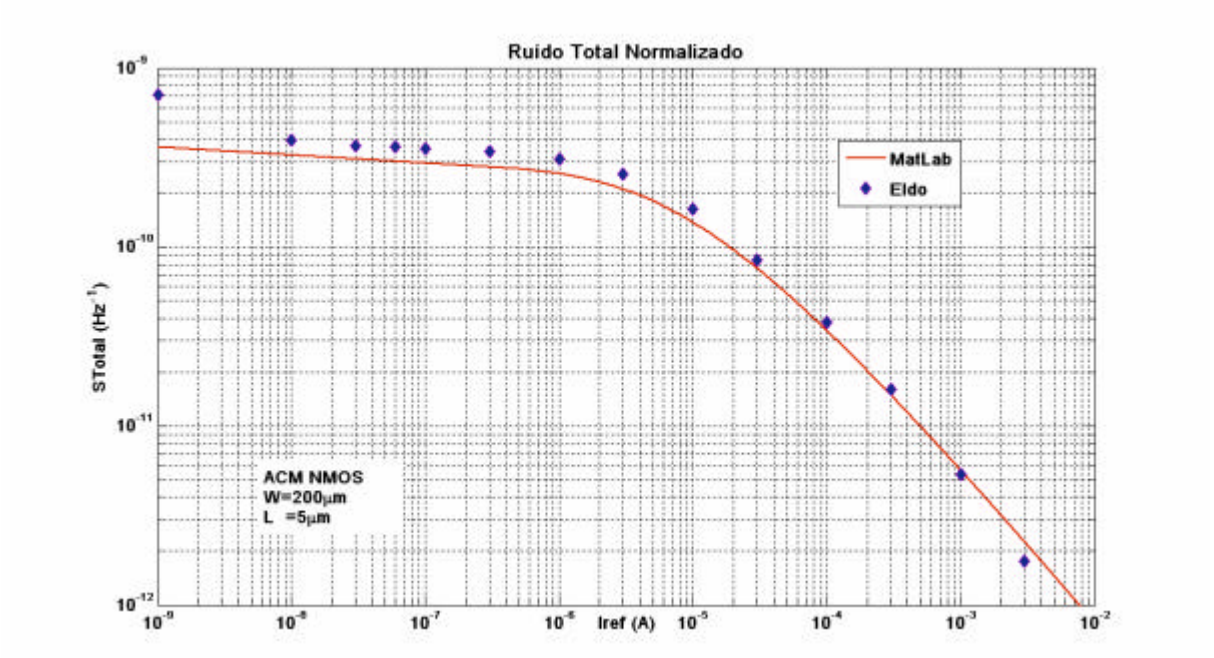


Figura 4.31 - Curva do ruído total normalizado para modelo ACM NMOS saturado W=200µm L=5µm.

A composição dos ruídos térmico e 1/f calculados e simulados estão indicados em um único gráfico, representado pela Figura 4.32. Com o objetivo de facilitar a comparação entre os resultados obtidos por [21] e o simulado a partir do código em Verilog-AMS, decidiu-se pela repetição da Figura 2.3 que por conveniência está sendo apresentada na Figura 4.33.

Gráfico da composição do ruído térmico e 1/f calculado e simulado.

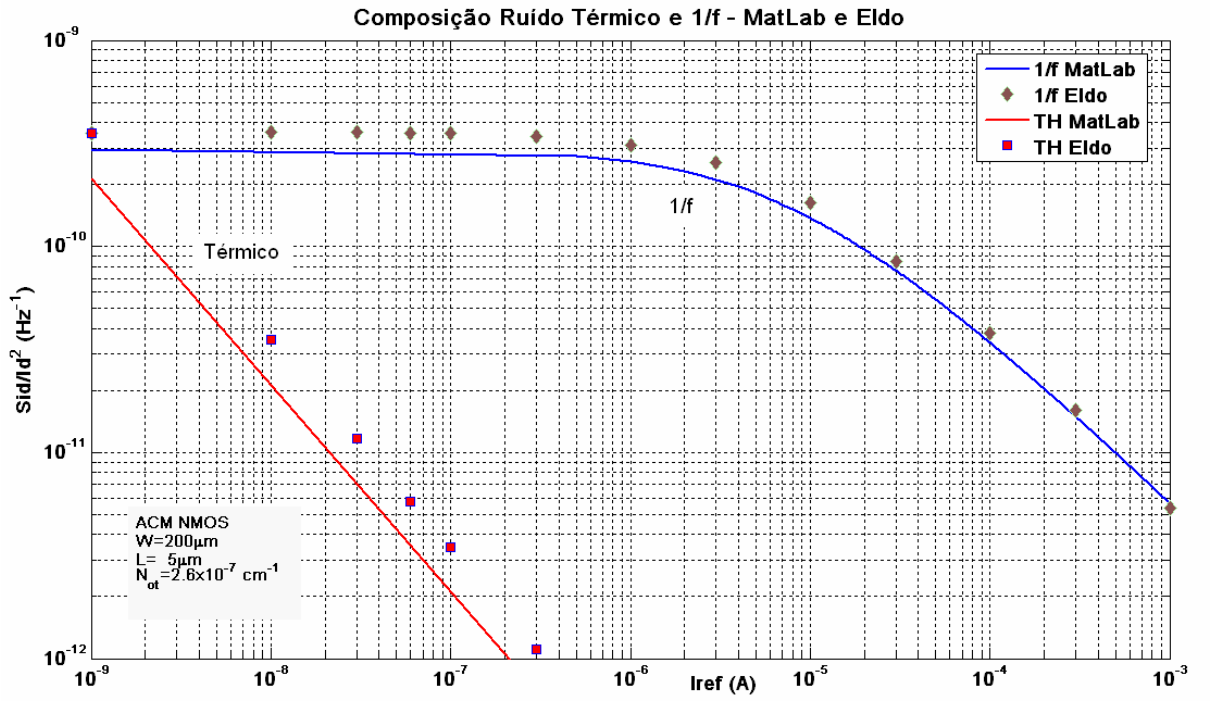


Figura 4.32 - Resultado da composição dos ruídos térmico e 1/f calculado (MatLab) e simulado (Eldo).

Gráfico usado como referência para a comparação dos resultados.

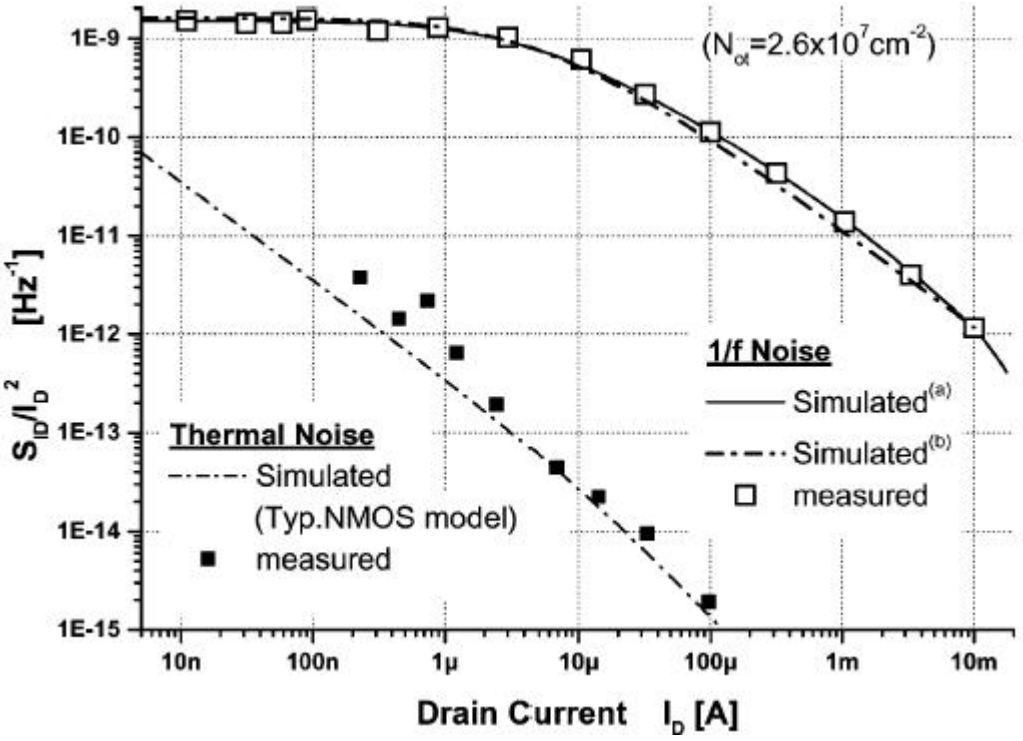


Figura 4.33 – Gráfico dos resultados obtidos por [21].

Pela comparação entre os resultados, conclui-se que os valores relativos ao código em Verilog-AMS são consistentes pela semelhança no formato das curvas, apesar da diferença numérica entre elas, que é atribuída ao uso de parâmetros diferentes para cada um dos transistores simulados (SPICE e ACM).

4.2 AVALIAÇÃO DOS RESULTADOS

Quanto aos simuladores, constatou-se que o simulador ELDO, cuja plataforma é o sistema operacional LINUX, atendeu às necessidades da operacionalidade das funções que estão descritas no Manual de Referência da Linguagem, suportando as funções mais apropriadas para as necessidades da programação do código fonte, o que não ocorreu com o SMASH. Esta limitação do simulador interferiu na forma de comparação dos resultados das simulações, pois os testes de validação que necessitavam da operação do transistor em conjunto com outros componentes não puderam ser simulados, impedindo com que o resultado das simulações com código escrito em Verilog-AMS pudesse ser comparado nos dois simuladores.

Isto justifica o motivo pelo qual ao ser usado o simulador SMASH, foi usado o código ACM escrito em linguagem C e não o escrito em Verilog-AMS como forma de comparação entre as simulações.

Quanto à maneira com que os simuladores manipulam os dados sobre o circuito elétrico, o ELDO é composto pela interface *DA-IC* (apêndice B.2 - que permite com que os circuitos elétricos possam ser desenhados em forma de esquemas, utilizando símbolos gráficos disponibilizados pelas bibliotecas padrão. Após os esquemáticos terem sido montados o simulador os converte em *netlist*, que poderão ser simulados diretamente do terminal usando comandos do sistema LINUX, sendo também possível optar por toda a descrição no formato *netlist*, sem ter que desenhá-lo. No simulador SMASH, a descrição do circuito e dos estímulos são todos no formato de *netlist*.

Quanto à forma de apresentação dos resultados, ambos os simuladores dispõem de ambiente gráfico que permitem a manipulação dos resultados.

Tendo como base os resultados obtidos nas simulações com o SMASH e o ELDO, constatamos a portabilidade do código ACM em Verilog-AMS, apesar das limitações encontradas no conjunto de funções pertinentes à linguagem Verilog-AMS suportadas pelo SMASH. A diferença de suporte da linguagem entre os simuladores implicou em códigos que estão escritos de forma diferenciada para cada um para poder aproveitar os recursos disponíveis da linguagem no ELDO, contudo é possível afirmar que o código é portável aos dois simuladores pela consistência dos resultados obtidos nos testes de validação.

Quanto aos resultados dos testes de validação, nos deparamos com um erro considerável na simulação da característica de saída Id=f(Vdb) e que não pôde ser esclarecido, com base no conjunto de ações que foram feitas. Para os demais testes de validação, ao serem comparados com o código em linguagem C, indicam que o código desenvolvido em Verilog-AMS comporta-se de forma consistente.

Apesar do tempo de processamento do código em Verilog-AMS ser superior ao código em C, faz-se o esclarecimento de que existem ferramentas computacionais que geram o código em C a partir do Verilog-AMS, desta forma o modelo é distribuído para qualquer simulador sem a preocupação com a plataforma em C, exemplo destas ferramentas podem ser encontradas em [38] e [39].

As funções e operações suportadas pela linguagem Verilog-AMS permitiu que a descrição do código ACM do transistor MOS fosse otimizada evitando operações matemáticas explícitas, esta versatilidade reduziu significativamente a quantidade de linhas de programação, sendo 2080 linhas no código em C e 750 linhas no código em Verilog-AMS, esta diferença corresponde a 64%.

5 CONCLUSÃO

5.1 COMENTÁRIOS FINAIS

O conteúdo apresentado nos capítulos 2 e 3 apresentam a síntese de parte da teoria que fundamentou o desenvolvimento deste trabalho e procura apresentar os aspectos que foram vivenciados durante o processo da implementação e validação do modelo ACM.

Constatou-se que no simulador SMASH a linguagem Verilog-AMS não está totalmente implementada. Desta forma o conjunto de testes de validação não puderam ser igualmente comparados, restrição que também implicou na escrita final do código ACM desprovido de recursos e facilidades que a linguagem disponibiliza para este objetivo específico. O código atualmente desenvolvido poderá receber melhorias que estarão vinculadas à implementação da linguagem no simulador SMASH.

Os resultados obtidos ao longo do desenvolvimento deste trabalho nos permitem concluir que a fundamentação da linguagem Verilog-AMS como uma HDL normalizada é consistente nos aspectos que estiveram ao alcance do que os simuladores permitiam desenvolver e simular tendo como base o Manual de Referência da Linguagem.

O simulador ELDO oferece recursos que estão organizados em ambientes diferenciados permitindo chegar até ao desenvolvimento do leiaute do componente em estudo. A existência de muitas interfaces acaba tornando a sua utilização complexa porque exige do usuário o conhecimento do conjunto para relacionar as características de cada interface para escolher aquela(s) que irá(ão) atender às necessidades. Por outro lado, o vínculo entre os ambientes garante que o projeto eletrônico nos diversos níveis de abstração esteja efetivamente concordante nas especificações. Cabe ressaltar que o ELDO é formado por recursos que permitem desenhar o circuito e não só escrevê-lo em forma de *netlist*, recurso

vantajoso para o projetista, pois é um apoio visual à análise e desenvolvimento. Quanto ao simulador SMASH, a sua análise fica prejudicada em função da linguagem Verilog-AMS não estar integralmente implementada.

No contexto geral, a linguagem Verilog-AMS é versátil e fácil de aprender, permitindo a portabilidade entre os simuladores e a redução significativa na quantidade de linhas de programação.

5.2 SUGESTÃO PARA FUTUROS TRABALHOS

Após a avaliação positiva do desempenho da linguagem Verilog-AMS, sugere-se a implementação de rotinas para a extração de parâmetros do modelo ACM, cálculo do ruído correlacionado, construção de módulos básicos de sistemas analógicos como amplificadores, filtros, fontes.

REFERENCIAL TEÓRICO

[1] WAGNER, Flávio Rech; JANSCH-PÔRTO, Ingrid; WEBWER, Raul Fernando; WEBWE, Taisy Silva**, Métodos de validação de sistemas digitai,**. Campinas: Unicamp, 1988.

[2] CORAM, Geoffrey J., **How to (and how not to) write a compact model in Verilog-A**, Proc. 2004 IEEE International Behavioral Modeling and Simulation Conference (BMAS 2004)

[3] MIERZWINSKI, M.; O'HALLORAN, P.; TROYANOVSKY, B. e DUTTON, R.; **Changing the Paradigm for Compact Model Integration in Circuit Simulators Using Verilog-A**; Nanotech 2005, pp. 376 – 379.

[4] **Verilog-AMS Language Reference Manual**, version 2.2, Accellera, 2004.

[5] http://www.accellera.org

[6] http://www.mentor.com/products/ic_nanometer_design/custom_design_simulation/eldo/index.cfm

[7] http://www.dolphin.fr/medal/smash/smash_overview.php

[8] Compact Model Council. http://www.geia.org/index.asp?bid=597

[9] http://www-device.eecs.berkeley.edu/~bsim3/ visitado em 26/04/08 as 9:31h

[10] http://helpme.scudc.scu.edu/hspice2001/hspice_and_qrg/hspice_2001_2-174.html#pgfId-192149 visitado em 26/04/08 as 9:16h

[11] http://pspmodel.asu.edu

[12] **Introduction to PSP**. Workshop on Compact modeling in Annaheim CA, May 2005; Technical Proceedings, pp. 19-24 Disponível em http://pspmodel.asu.edu/downloads/IntroPSP.pdf. visitado em 29/04/08 às 11:07h

[13] WAKERLY, John F**. Digital design, principles and practices,** Upper Saddle River, NJ. Pearson Prentice Hall, 4th edition, 2006.

[14] **Verilog Synthesis Tutorial Part-I**, Disponível em http://www.asic-world.com/verilog/synthesis1.html. visitado em 06/04/08 às 17:21h

[15] PÊCHEUX, François, LALLEMENT, Christophe; VACHOUX, Alain. **VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems,** IEEE Transactions on computer aided design of integrated circuits and systems, vol24, nº 2, fevereiro 2005.

[16] SMITH, Douglas J. **VHDL & Verilog compared &contrasted – plus modeled example written in VHDL,Verilog and C,** 33rd Design Automation Conference, 1996.

[17] TISIVIDS, Y., **Operation and Modeling of the MOS Transistor**, Second Edition, Mc Graw-Hill, New York, 1999.

[18] SEDRA, Adel S.; SMITH, Kenneth C., **Microeletrônica**, 4ª ed. Pearson Education do Brasil, São Paulo, 2000.

[19] GOUVEIA Fº, Oscar da. C., **Um modelo compacto do transistor MOS para simulação de circuitos**, Doctor Thesis, Florianópolis, Santa Catarina, Brasil, 1999.

[20] GOUVEIA Fº, Oscar da. C, CUNHA A. I. A, SCHNEIDER M. C. and GALUP-MONTORO C. **Advanced compact model for short-channel MOS transistors**, IEEE 2000 Custom Integrated Circuits Conference, pp. 209-212, Orlando, May 2000.

[21] ARNAUD, A. e GALUP-MONTORO C., **Consistent noise models for analysis and design of CMOS circuits**, *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol.51, pp.1909-1915, October 2004.

[22] ARNAUD, A. e GALUP-MONTORO C., **A compact model for flicker noise in MOS transistors for analog circuit design**, *IEEE Transactions on Electron Devices*, vol.50, pp.1815-1818, August 2003.

[23] GALUP-MONTORO C., SCHNEIDER M. C. e CUNHA A. I. A, **A current-based MOSFET model for integrated circuit design**, Chapter 2 in Low-Voltage/Low-Power Integrated Circuits and Systems, Edited by E. Sánchez-Sinencio and A. C. Andreou, pp. 7-55, IEEE Press, New Jersey, 1999.

[24] CUNHA, Ana Isabela Araújo. **Um modelo do transistor MOS para projetos de circuitos integrados,** Tese de doutorado. Universidade Federal de Santa Catarina, Florianópolis 1996.

[25] GIRARDI, Alessandro Gonçalves. **Uma ferramenta para automação da geração do leiaute de circuitos analógicos sobre uma matriz de transistores MOS pré-**

difundidos, Porto Alegre: PPGC da UFRGS, 2003. Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul - Programa de Pós-Graduação, Porto Alegre, BR - RS, 2003.

[26] CORAM, Geoffrey J., **MOS Model 11 Verilog-A implementation level 11010**, Disponível em www.designers-guide.org. visitado em 08/08/2006 14:48h.

[27] CORAM, Geoffrey J., **Verilog-A: An introduction for compact modelers,** Analog Devices. Apresentação em PPT. Workshop (Montreux 2006).

[28] WAGNER, Flávio Rech;JANSCH-PÔRTO, Ingrid; WEBWER, Raul Fernando; WEBWE, Taisy Silva. **Métodos de validação de sistemas digitais**, Campinas: Unicamp, 1988.

[29] FITZPATRICK, Dan; MILLER, Ira, **Analog Behavioral modeling with the Verilog-A,** Nova York, Springer Science+Business Media, Inc. 1998.

[30] KUNDERT, Kenneth S.; ZINKE, Olaf , **The designer's guide to Verilog-AMS**, Springer Science+Business Media. 1st ed. 2004.

[31] KURKER, Christopher M; PAULOS, John J.; COHEN, Brian S. and COOLEY, Edmond S. **Development of an analog hardware description language,** Custom Integrated Circuits conference. IEEE 1990

[32] KUNDERT, Ken; CHANG, Henry; JEFFERIES, Dan; LAMANT, Gilles; MALAVASI, Enrico; SENDIG, Fred, **Design of mixed-signal systems on chip,** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. March 2001.

[33] CABRAL, Maria Izabel Cavalcanti, **Simulação de redes ATM,** disponível em. http://www.mc21.fee.unicamp.br/alberti/tese_cap4.pdf. visitado em 14/01/08 às 10:34h

[34] KAZMIERSKI, Tom J. e JUNAID, Hessa Al, **Synchronization of analogue and digital solvers in mixed-signal simulation on a SystemC platform,** In: Forum on Specification and Design Languages, 23-26 September 2003, Frankfurt, Germany.

[35] GRABINSKI, Wladyslaw, TOMASZEWSKI, Daniel, LEMAITRE, Laurent, JAKUBOWSKI, Andrzej. **Standardization of the compact model coding: non-fully depleted SOI MOSFET example,** Journal of Telecomunications and Information Technology, pág. 135-141 1/2005.

[36] GAJSKI, Daniel D., **Principles of digital design**, University of California. Upper Saddle River, New Jersey 07458, Prentice Hall. 1997.

[37] Philips Research Press Release, March 30, 2005 Philips and Penn State University, **Sophisticated transistor model for advanced CMOS to the CMC for standardization**, http://www.research.philips.com/newscenter/archive/2005/050330-cmoscompactmodel.html, visitado em 26/04/08 as 9:03h

[38] CHAUDHARY, Vivek et al., **Automatic Generation of Compact Semiconductor Device Models using Paragon and ADMS,** Behavioral Modeling and Simulation Conference, 2004. BMAS 2004. Proceedings of the 2004 IEEE International Volume , Issue , 21-22 Oct. 2004 Page(s): 107 – 112

[39] LAMAITRE, Laurent; McANDREW, Colin; HAMM, Steve, **ADMS - Automatic Device Model Synthesizer,** Custom Integrated Circuits Conference, 2002. Proceedings of the IEEE 2002. Page(s): 27 - 30

APÊNDICE A

A.1 - CÓDIGO FONTE EM VERILOG AMS DO MODELO ACM

Nesta sessão é apresentado o código do modelo ACM em Verilog-AMS sendo que a sua estrutura contempla o seu funcionamento nos simuladores SMASH e ELDO. É importante ressaltar que alguns recursos que a linguagem oferece, não puderam ser aplicados ou foram parcialmente utilizados nesta versão, tendo em vista que cada simulador não tem implementado todo o conjunto de funções e diretivas que estão padronizados no *Language Reference Manual (LRM)* Versão 2.2 – Novembro 2004 [4]. Como o intuito do trabalho era a verificação da portabilidade, optou-se pela apresentação do código contendo a sintaxe que permite o funcionamento em ambos os simuladores com pequenas modificações.

Tais modificações estão caracterizadas pelos comentários: "/// INCLUIR NO SMASH ///", "/// OMITIR NO SMASH ///""/// INCLUIR NO ELDO ///", "/// OMITIR NO ELDO ///"

CÓDIGO FONTE

```
1. ////////////////////////////////////////////////////////////////////////////
2. ////    Versao que contempla o calculo do ruido, tendo como base o paper:
3. ////    "Consistent Noise Models for Analysis and Design of CMOS Circuits"
4. ////    Alfredo Arnaud e Carlos Galup-Montoro
5. ////    eq. 6 flicker
6. ////    eq. 17 white
7. ////////////////////////////////////////////////////////////////////////////
8. ////    Código que rodou corretamente no ELDO em 14 de janeiro 2008   ///
9. ////    e que foi adaptado para rodar também no SMASH                 ///
10.        ////////////////////////////////////////////////////////////////////////////
11.        //
12.        //      Alterações para uso no SMASH:
13.        //          1)Eliminada a função "limexp"
14.        //          2)Eliminada a função "analog function"
15.        //          3)Inclusão do laço if-else para o cálculo do "uf" e "ur"
16.        //          4)Inclusão do laço "pseudo safelog"
17.        //          5)Mudança das diretivas include.h para include.vams
18.        //          6)Cálculo da corrente "xi" no bloco "Charges_Density_Current"
```

```
19.  ////////////////////////////////////////////////////////////////
20.  //        INCLUIR ELDO   -   OMITIR SMASH
21.  //
22.  //`include "constants.h"
23.  //`include "disciplines.h"
24.  //
25.  ////////////////////////////////////////////////////////////////
26.  //         INCLUIR SMASH   -   OMITIR ELDO
27.  //
28.  `include "disciplines.vams"
29.  `include "constants.vams"
30.  //
31.  ////////////////////////////////////////////////////////////////


32.  `timescale 1s / 1fs

33.  `define EPSLON  3.45E-11  //permissividade elt SiO2->default tese F/m

34.  `resetall

35.  module acm (d,g,s,b);
36.  inout d,g,s,b;
37.  (* desc     = "drain terminal" *) electrical d;        //description attribute
38.  (* desc     = "gate terminal" *) electrical g;         //description attribute
39.  (* desc     = "source terminal" *) electrical s;       //description attribute
40.  (* desc     = "bulk terminal" *) electrical b;         //description attribute

41.  branch (d,s)canal1;

42.  //current transGMS,transGMD,transGMG,transGMB;  //Permite acessar variáveis
     internas,mais apropriado para o SMASH

43.  ////////////////////////////////////////////////////////////////////////////////////////
44.  //
45.  //                    PARAMETERS FOR ACM CODE
46.  //
47.  ////////////////////////////////////////////////////////////////////////////////////////

48.  parameter integer type  =   1          from [-1:1] exclude 0;  // NMOS = 1   PMOS = -1
49.  parameter real l          =   1.0e-6     from (1.0e-9:1.0e-3);   // channel length [m]
50.  parameter real w          =   10.0e-6    from (1.0e-9:1.0e-3);   // channel width [m]
51.  parameter real ld         =   4.0e-9     from (1.0e-12:1.0e-6); // coeficiente encurtamento
     canal [?]
52.  parameter real tox        =   7.7e-9     from (10.0e-15:10.0e-6); // oxide thicknesses [m]
53.  parameter real xj         =   1.0e-7     from (10.0e-9:10.0e-6);  // profundidd juncao [m]
54.  parameter real sigma1     =   268.8e-18  from (1.0e-20:1.0e-10)  // coefic. p/ DIBL [m2]
55.  parameter real tnom       =   27.0       from [-273.15:inf);      // environment temperature
     [ºC]
56.  parameter real gamma      =   0.601      from (0.0:1.0);          // body effect [sqrt(V)]
57.  parameter real pclm       =   1.645      from (0.0:10.0);         // chanel length modulation
```

```
58.     parameter real vth0     =  0.486       from (-3.0:3.0)exclude 0; // tensao de limiar [V]
59.     parameter real phi      =  0.862       from (0.0:1.0);           // potencial superficie [V]
60.     parameter real u0       =  436.8e-4    from (1.0e-6:1.0);        // mobility [m2/V.s]
61.     parameter real vmax     =  1.613e5     from (1.0e3:10.0e9); //
62.     parameter real theta    =  0.0593      from (0.0:0.1);       // coeficiente de reducao
    mobilidade [V-1]
63.     parameter real k        =  2.0         from [1.0:3.0);       //

64.     ////////////////////////////////////////////////////////////////////////////////////////////////
65.     //                    PARAMETERS NOISE EVALUATION
66.     ////////////////////////////////////////////////////////////////////////////////////////////////
67.     //parameter real ntraps     =  4e16 ;    //density of oxide traps per unit volume and unit
    energy [cm-3eV-1]
68.     //parameter real e_gamma  =  1e8 ;    //attenuation coefficient of the electron wave function
    in the oxide  [cm-1]
69.     parameter real Not              =  1e7;       //density of oxide traps [cm-2]

70.     ////////////////////////////////////////////////////////////////////////////////
71.     //                    DEPENDENT PARAMETERS
72.     ////////////////////////////////////////////////////////////////////////////////
73.     parameter real leff       =  l - 2.0*ld; // It was introduced in this code,it isn't in'C' code
74.     parameter real sqgamma  =  gamma * gamma;
75.     parameter real sigma      =  sigma1/(leff*leff);
76.     parameter real weff       =  w;        //ALLIAS
77.     ////////////////////////////////////////////////////////////////////////////////
78.     //     INCLUIR ELDO - OMITIR SMASH //
79.     //     AUXILIARY FUNCTIONS       //
80.     //     Calculo do "uf" e "ur"     //
81.     ////////////////////////////////////////////////////////////////////////////////
82.     /*
83.     analog function real safelog1;
84.     input mantissa1,mantissa2;
85.     real  mantissa1,mantissa2;
86.     real  mantissa;

87.     begin
88.     mantissa=  mantissa1/mantissa2;
89.     if  (mantissa < 1.0e-80)
90.     safelog1 = ln (1.0e-80);
91.     else
    92. safelog1 = ln (mantissa);
93.     end
94.     endfunction // safelog

95.     analog function real safelog2;
96.     input mantissa1;
97.     real  mantissa1;

98.     begin
99.     if  (mantissa1 < 0)
100.    safelog2 = ln (1.0e-80);
```

```
101.    else
102.    safelog2 = ln (mantissa1);
103.    end
104.    endfunction // safelog

105.    analog function real safesqrt;
106.    input radicando1;
107.    real  radicando1;
108.    begin
109.    if  (radicando1 < 1.0e-30)
110.    safesqrt = ln (1.0e-30);
111.    else
112.    safesqrt = ln (radicando1);
113.    end
114.    endfunction // safesqrt
115.    */
116.    ////////////////////////////////////////////////
117.    //          GLOBAL VARIABLES
118.    ////////////////////////////////////////////////
119.    //      Terminals Voltage Evaluation
120.    ////////////////////////////////////
121.    integer material;
122.    real vd,vg,vs,vb;
123.    real v1,v2,v3,v4;
124.    real vgb,vsb,vdb,vds,vgs;
125.    real drain3;
126.    ////////////////////////////////////
127.    //      VPinch Off Evaluation
128.    ////////////////////////////////////
129.    real ktonq,sqktonq;
130.    real vsb1,vdb1,vgb1,vgb2;
131.    real vpo2,vpo1,vpo,vp;
132.    real n;

133.    ////////////////////////////////////////////////////////////////////////////////////////
134.    //      Transcondutance Evaluation
135.    //      Definição das variáveis para o cálculo explícito da transcondutân
136.    //  cia, encontra-se no arquivo:"CalculoTranscondutância.txt"
137.    ////////////////////////////////////////////////////////////////////////////////////////
138.    ////////////////////////////////////////////////////////
139.    //      SECOND PART OF ACM CODE
140.    ////////////////////////////////////////////////////////
141.    real cox,ncoxktonq;
142.    real uf,kuf,qf0,qf,qis;
143.    real ucrit;
144.    real uef,qo,ksiqo;
145.    real qmin;
146.    real delNum,delDen;
147.    real del,result;
148.    real vdssat,vdsl;
```

```
149.    real lc,delta_l,lmin,l1,le;
150.    real vdsat,ur,kur;
151.    real qa,ksiqa,qidsat;
152.    real vdl,qr0,qr,qid;
153.    real xi1,xi2,beta;
154.    real xi;

155.    ////////////////////////////////////////////////////////////
156.    //      THIRD PART OF ACM CODE
157.    ////////////////////////////////////////////////////////////
158.    //      Charges Evaluation
159.    //      Definição das variáveis para o cálculo explícito da capacitân
160.    //  cia, encontra-se no arquivo:"CalculoCapacitância.txt"
161.    ////////////////////////////////////

162.    real qg, qb, qd, qs, qi;
163.    real x, y, x2, y2, x3, y3, xmaisy2, xmaisy3;
164.    real I;
165.    real qi_dl;
166.    real Qi,Qd,Qg,Qs,Qb;

167.    ////////////////////////////////////////////////////////////
168.    //      FOURTH PART OF ACM CODE
169.    ////////////////////////////////////////////////////////////
170.    //      Noise Evaluation
171.    ////////////////////////////////////////
172.    real Sid;//Not;
173.    real Siw;
174.    ////////////////////////////////////////////////////////////
175.    //
176.    //  Sequence of ACM MOS==> "Drain" "Gate" "Source" "Bulk"
177.    //                          ___________o D=drain (terminal 1)
178.    //                         |
179.    //          G=gate o----|<-------o B=bulk (terminal 4)
180.    //          (terminal2)  |___________o S=source (terminal 3)
181.    //
182.    ////////////////////////////////////////////////////////////
183.    `resetall
184.    analog begin
185.    material=((type==1)?1:-1);

186.    v1=V(d);
187.    v2=V(g);
188.    v3=V(s);
189.    v4=V(b);
190.    ////////////////////////////////////////////////////////////////////////////////
191.    //  Verification of MOS type - v1=V(d),v2=V(g),v3=V(s),v4=V(b)
192.    ////////////////////////////////////////////////////////////////////////////////
193.    if (material == 1) begin
194.    vd  =   v1;
```

```
195.    vs  =   v3;
196.    drain3 = 0;       //==>significa que Vd>Vs ou seja Id  positiva
197.    if(v3 > v1)begin
198.    vd     = v3;
199.    vs     = v1;
200.    drain3 = 1;  //==> Vs > Vd  that implies in negative Id
201.    end
202.    vds =   vd-vs;
203.    vgs =   v2-vs;
204.    vsb =   vs-v4;
205.    vdb =   vd-v4;
206.    vgb =   v2-v4;
207.    end else begin
208.    if  (material == -1)begin
209.    vd  =   v3;
210.    vs  =   v1;
211.    drain3  =1;           //==>significa que Vs>Vd ou seja Id negativa
212.    if(v3 > v1)begin
213.    vd  =   v1;
214.    vs  =   v3;
215.    drain3 = 0;     //==>significa que Vd>Vs ou seja Id positiva
216.    end
217.    vds =   vs-vd;
218.    vgs =   vs-v2;
219.    vsb =   v4-vs;
220.    vdb =   v4-vd;
221.    vgb =   v4-v2;
222.    end//2o. if
223.    end//1o. if


224.    if ((material == -1)  &   (vth0 > 0.0))begin
225.    $display("vt0 com valor incorreto");
226.    end


227.    begin:definition_Vpinchoff

228.    ktonq      = $vt((tnom +`P_CELSIUS0));   //thermodinamic potential - thermal voltage(PHIt)
229.    sqktonq    = ktonq*ktonq;
230.    vsb1       = 0.5 * (vsb + phi + sqrt((vsb + phi) * (vsb + phi) + 4 * sqktonq ));
231.    vdb1       = 0.5 * (vdb + phi + sqrt((vdb + phi) * (vdb + phi) + 4 * sqktonq ));
232.    vgb1       = vgb - abs(vth0) + phi + gamma * sqrt(phi);
233.    vgb2       = 0.5 * (vgb1 + sqrt((vgb1 * vgb1) + 4.0 * sqktonq));
234.    vpo2       = sqrt(vgb2 + sqgamma / 4.0) - gamma/2.0;
235.    vpo1       = vpo2*vpo2;
236.    vpo        = vpo1 - phi;
237.    n          = 1.0 + gamma/(2.0 * sqrt(phi + vpo2));//slop factor
238.    vp         = vpo + (sigma / n) * (vsb1 + vdb1);
239.    end //Pinchoff_Evaluation
```

```
240.    begin:Charges_Density_Current

241.    cox             = `EPSLON/tox;    //oxide capacitance per unit area
242.    ncoxktonq   = n * cox * ktonq;
243.    uf              = (vp - vsb) / ktonq + 2.0;
244.    kuf             = 1.0 - 84.4839/((uf * uf) + 150.864);
245.    ////////////////////////////////////////////////////////////////////////////////////
246.    //    "limexp" -> ELDO que permite a utilizacao unica do:
247.    //  qf0 = 1.0 + ln(1.0 + exp(uf - 1.0)/(1.0 + kuf * ln(1.0 + exp(uf - 1.0))))
   //EQUAÇÃO ORIGINAL
248.    qf0     = 1.0 + ln(1.0 + limexp(uf - 1.0)/(1.0 + kuf * safelog2(1.0 + limexp(uf -
   1.0))));// INCLUIR ELDO - OMITIR SMASH
249.    ////////////////////////////////////////////////////////////////////////////////////
250.    //    INCLUIR SMASH  -  OMITIR ELDO
251.    //    Verificação 'manual' de "uf" utilizando if-else
252.    ////////////////////////////////////////////////////////////////////////////////////
253.    /*
254.    if (uf > 100.0) begin
255.    qf0     = uf -ln(1.0 + kuf * (uf - 1.0));
256.    end
257.    else
258.    if (uf < -100.0) begin
259.    qf0     = 1.0;
260.    end
261.    else
262.    qf0     = 1.0 + ln(1.0 + exp(uf - 1.0)/(1.0 + kuf * ln(1.0 + exp(uf - 1.0))));
263.    */
264.    ////////////////////////////////////////////////////////////////////////////////////
265.    qf      = -qf0*ncoxktonq;
266.    qis     = qf + ncoxktonq;     //==>Q'IS
267.    uef     = u0 /(1.0 + theta * (n * vpo + phi)); //efective mobility
268.    ucrit   = vmax/uef;
269.    qo      = cox * leff * n * ucrit;
270.    ksiqo   = ncoxktonq / qo;    // allias for epsilon in C code
271.    ////////////////////////////////////////////////////
272.    //    CALCULO DO VDSSAT
273.    //    Analog Function "safesqrt" -> ELDO
274.    //    qmin    = 2.0 * ksiqo * qis * (1.0 - qis / (2.0 * ksiqo * qo)) / (1.0 - (qis -
   ncoxktonq)/qo + safesqrt(1.0 - 2.0 * (qis -ncoxktonq)/qo + (ncoxktonq/qo)
   *(ncoxktonq/qo))); // INCLUIR ELDO -OMITIR SMASH//
275.    qmin    = 2.0 * ksiqo * qis * (1.0 - qis / (2.0 * ksiqo * qo)) / (1.0 - (qis -
   ncoxktonq)/qo + sqrt(1.0 - 2.0 * (qis -ncoxktonq)/qo + (ncoxktonq/qo) *(ncoxktonq/qo)));
276.    ////////////////////////////////////////////////////////////
277.    //    INCLUIR SMASH - OMITIR ELDO
278.    //    Pseudo SafeLog
279.    ////////////////////////////////////////////////////////////
280.    //    Expressao original do Vdssat
```

```
281.  //  vdssat   = ktonq * ((qmin - qis)/ncoxktonq - safelog(2.0* epsilon * (1.0 - qis/(2.0 *
   epsilon * qo))/(1.0 - (qis-ncoxktonq)/qo + sqrt(1.0 - 2.0 * (qis-ncoxktonq)/qo +
   (ncoxktonq/qo) * (ncoxktonq/qo)))));
282.  /*
283.  delNum   =   2.0* ksiqo * (1.0 - qis/(2.0 * ksiqo * qo));
284.  delDen   =   (1.0 - (qis -ncoxktonq)/qo + sqrt(1.0 - 2.0 * (qis -ncoxktonq)/qo +
   (ncoxktonq/qo) * (ncoxktonq/qo)));
285.  del        =   delNum/delDen;

286.  if ( del < 1e-40) begin
287.  result  = ln(1e-40);
288.  end
289.  else
290.  result  = ln(del);
291.  */

292.  ////////////////////////////////////////////////////////////////////////////////
293.  //     Expressão adaptada para Vdssat - ELDO
294.  //     vdssat   = ktonq * ((qmin - qis)/ncoxktonq - safelog1(2.0* ksiqo * (1.0 - qis/(2.0
   * ksiqo * qo)),(1.0 - (qis-ncoxktonq)/qo + safesqrt(1.0 - 2.0 * (qis-ncoxktonq)/qo +
   (ncoxktonq/qo) * (ncoxktonq/qo)))));
295.  ////////////////////////////////////////////////////////////////////////////////
296.  //     Expressão adaptada para Vdssat - SMASH
297.  vdssat   = ktonq * ((qmin - qis)/ncoxktonq - result);
298.  ////////////////////////////////////////////////////////////////////////////////
299.  vdsl     = (vdb-vsb)/(pow(1.0+pow((vdb-vsb)/vdssat,2.0*k),1.0/k/2.0));
300.  ////////////////////////////////////////////////////
301.  //         CALCULO DO VDSAT
302.  ////////////////////////////////////////////////////
303.  lc        = sqrt(`EPSLON * xj/cox);
304.  delta_l  = pclm * lc * ln(1.0 + (vds - vdsl)/(lc * ucrit));
305.  lmin     = leff/10.0;
306.  l1        = leff - delta_l;
307.  le         = 0.5 * (l1 + sqrt(l1 * l1 + lmin * lmin));//effective lenght
308.  qa         = cox * le * n * ucrit;
309.  ksiqa     = ncoxktonq/qa;
310.  ////////////////////////////////////////////////////////////////////////
311.  //     Analog Function "safesqrt" INCLUIR ELDO
312.  // qidsat = 2.0 * ksiqa * qis * (1.0 - qis/(2.0*ksiqa * qa))/(1.0 - (qis - ncoxktonq)/qa +
   safesqrt(1.0 - 2.0 * (qis-ncoxktonq)/qa + (ncoxktonq/qa) * (ncoxktonq/qa)));//INCLUIR
   ELDO
313.  qidsat = 2.0 * ksiqa * qis * (1.0 - qis/(2.0*ksiqa * qa))/(1.0 - (qis - ncoxktonq)/qa +
   sqrt(1.0 - 2.0 * (qis-ncoxktonq)/qa + (ncoxktonq/qa) * (ncoxktonq/qa)));//INCLUIR
   SMASH
314.  ////////////////////////////////////////////////////////////////////////
315.  //     Expressão Original do Vdsat
316.  //         vdsat = ktonq * ((qidsat - qis)/ncoxktonq - safelog(2* ksiqa * (1 -
   qis/(2*ksiqa * qa))/(1 - (qis-ncoxktonq)/qa + sqrt(1 - 2 * (qis-ncoxktonq)/qa +
   (ncoxktonq/qa) * (ncoxktonq/qa)))));
```

```
317.  ////////////////////////////////////////////////////////////
318.  //  INCLUIR SMASH - OMITIR ELDO
319.  //  Pseudo SafeLog
320.  ////////////////////////////////////////////////////////////
321.  /*
322.  delNum   =   2.0* ksiqa * (1.0 - qis/(2.0 * ksiqa * qa));
323.  delDen    =   (1.0 - (qis -ncoxktonq)/qa + sqrt(1.0 - 2.0 * (qis -ncoxktonq)/qa +
   (ncoxktonq/qa) * (ncoxktonq/qa)));
324.  del         =   delNum/delDen;

325.  if ( del < 1e-40)
326.  result= ln(1e-40);
327.  else
328.  result= ln(del);
329.  */
330.  ////////////////////////////////////////////////////////////////////////////
331.  //      Expressão adaptada para Vdsat - ELDO
332.  //   vdsat   = ktonq * ((qidsat - qis)/ncoxktonq - safelog1(2.0* ksiqa * (1.0 -
   qis/(2.0*ksiqa * qa)),(1.0 - (qis-ncoxktonq)/qa + safesqrt(1.0 - 2.0 * (qis-ncoxktonq)/qa +
   (ncoxktonq/qa) * (ncoxktonq/qa)))));
333.  ////////////////////////////////////////////////////////////////////////////
334.  //      Expressão adaptada para Vdsat - SMASH
335.  vdsat = ktonq * ((qidsat - qis)/ncoxktonq - result);
336.  ////////////////////////////////////////////////////////////////////////////
337.  vdl     = (vdb-vsb)/pow(1.0+pow((vdb-vsb)/vdsat,2.0*k),1.0/k/2.0) + vsb;
338.  ur      = (vp - vdl) / ktonq + 2.0;
339.  kur     = 1.0 - 84.4839/((ur * ur) + 150.864);
340.  qr0     = 1.0 + ln(1.0 + limexp(ur - 1.0)/(1.0 + kur * safelog2(1.0 + limexp(ur -
   1.0))));//ONLY ELDO
341.  ////////////////////////////////////////////////////////////////////////////////////
342.  ////Verificacao 'manual' de "ur" utilizando if-else -> SMASH
343.  ////////////////////////////////////////////////////////////////////////////////////

344.  /*
345.  if (ur > 100)begin
346.  qr0    = ur -ln(1.0 + kur * (ur - 1.0));
347.  end
348.  else
349.  if (ur < -100) begin
350.  qr0   = 1.0;
351.  end
352.  else
353.  qr0   = 1.0 + ln(1.0 + exp(ur - 1.0)/(1.0 + kur * ln(1.0 + exp(ur - 1.0))));
354.  */
355.  qr      = -qr0*ncoxktonq;   //==>Q'ID
356.  qid     = qr + ncoxktonq;
357.  xi1     = (qf * qf - qr * qr) / (2.0 * n);
358.  xi2     = 1.0/(1.0 + sqrt((qf - qr)*(qf - qr)+ (qa/10.0)*(qa/10.0))/qa);
359.  beta    = (uef * w) / (cox * le);
360.  xi      = beta * xi1 * xi2;              // drain current = corrente de dreno
```

```
361.    ////////////////////////////////////////////////////////////////////////////////
362.    //
363.    //      O bloco que define I(d,s) nesta posição impede que a análise transiente no
        ELDO
364.    //  seja realizada corretamente, pois no cálculo das cargas para a definicao
365.    //  das correntes ac, faz-se novamente a atribuição de I(d,s) o que implica em atribuir
366.    //  valores diferentes para a mesma variavel.
367.    //  Desta forma o cálculo da Idreno é feita junto com atribuições corretas das cargas,
368.    //  MAS...., no SMASH o uso de "analog operators" (como '<+' = contrbution (item
        3.2),
369.    //  livro Designer's Guide pag169) resulta em erro conforme item 4.6 pag178 do
        mesmo livro,
370.    //  A fim de evitar tais erros, retirou-se as 'constributions' de dentro do laço das
        cargas
371.    //  e contornou-se o cálculo da corrente de dreno pela expressão: I(canal1)<+
        material*xi
372.    //
373.    ////////////////////////////////////////////////////////////////////
374.    //  INCLUIR SMASH - OMITIR ELDO
375.    ////////////////////////////////////////////////////////////////////
376.    I(canal1)   <+ material*xi;

377.    end//charges and drain current

378.    ////////////////////////////////////////////////////////////////////////////////
379.    //
380.    //      Transcondutance Evaluation      RETIRADA PARA CALCULO DO
        TRANSIENTE
381.    //      BLOCO DIFERENCIADO DO SMASH
382.    //      Esta parte do codigo esta identificada por:
383.    //      CalculoTranscondutancia.txt dentro do diretorio "Documentcao ACM 510_0"
384.    ////////////////////////////////////////////////////////////////////////////////


385.    ////////////////////////////////////////////
386.    //      THIRD PART
387.    ////////////////////////////////////////////
388.    //      Charges Evaluation
389.    ////////////////////////////////////////////
390.    begin:Charges_Evaluation
391.    I       = xi/w/vmax;
392.    x       = (qf + I);
393.    y       = (qr + I);
394.    x2      = pow(qf+I,2.0);
395.    y2      = pow(qr+I,2.0);
396.    x3      = pow(qf+I,3.0);
397.    y3      = pow(qr+I,3.0);
398.    xmaisy2   = (x + y) * (x + y);//= x2+2.0.x.y+y2= pow(qf+I,2.0)+
        2.0*(qf+I)*(qr+I)+pow(qr+I,2.0)
```

```
399.    xmaisy3   = xmaisy2 * (x +
        y);//=x3+y3+3.0.x2.y+3.0.x.y2=pow(qf+I,3.0)+pow(qr+I,3.0)+3.0*pow(qf+I,2.0)*(qr+I)+
        3.0*(qf+I)*pow(qr+I,2.0)

400.    qi1       = w * le * ((2.0/3.0)*((x2 + x * y + y2)/(x + y)) + ncoxktonq - I);
401.    ////////////////////////////////////////////////////
402.    //      CHARGES: qi,qd, qg, qs, qb
403.    ////////////////////////////////////////////////////
404.    qi_dl     = w * delta_l * qid;
405.    qi        = qi1 + qi_dl;
406.    qb        = -(n - 1.0) * qi/n - w * leff *cox * sqgamma/2.0/(n - 1.0);
407.    qg        = -qb - qi;

408.    qd1       = w*le*le/leff *((6.0* y3 + 12 * y2 * x + 8.0* y * x2 + 4.0 * x3)/(15.0 *
        xmaisy2) + (ncoxktonq - I)/2.0);
409.    qd_dl     = w * (leff*leff - le*le)/2.0/leff * qid;

410.    qd        = qd1 + qd_dl;
411.    qs        = qi - qd;
412.    end//Charges_Evaluation
413.    ////////////////////////////////////////////////////////////////////////////////////
414.    //      CHARGES EVALUATION (linha 1525 p. 24 CÓDIGO 'C')
415.    ////////////////////////////////////////////////////////////////////////////////////
416.    //      Small_Signs
417.    ////////////////////////////////////////////////////
418.    /*
419.    if (drain3 == 0) begin
420.    ////////////////////////////////////////////////////////////
421.    //  INCLUIR ELDO - OMITIR SMASH
422.    ////////////////////////////////////////////////////////////

423.    I(d,s) <+   material* xi;
424.    I(g,s) <+ 0;
425.    I(b,s) <+ 0;
426.    Qd     = qd;
427.    Qg     = qg;
428.    Qb     = qb;
429.    end else begin
430.    if ( drain3== 1)  begin
431.    ////////////////////////////////////////////////////////////
432.    //  INCLUIR ELDO - OMITIR SMASH
433.    ////////////////////////////////////////////////////////////
434.    I(d,s) <+ -material* xi;
435.    I(g,s) <+ 0;
436.    I(b,s) <+ 0;
437.    Qd     = -qd;
438.    Qg     = -qg;
439.    Qb     = -qb;
440.    end //2º if
441.    end //1º if
```

```
442.    ////////////////////////////////////////////////
443.    //      AC Current
444.    ////////////////////////////////////////////////
445.    I(d,s) <+  ddt(Qd);
446.    I(g,s) <+  ddt(Qg);
447.    I(b,s) <+  ddt(Qb);
448.    */
449.    ////////////////////////////////////////////////////////////////////////////////////////
450.    //  Capacitances Evaluation     RETIRADA PARA CALCULO DO TRANSIENTE
451.    //
452.    //          BLOCO DIFERENCIADO DO SMASH
453.    //  Esta parte do codigo esta identificada por:
454.    //  CalculoCapacitancia.txt dentro do diretorio "Documentacao ACM 510_0"
455.    //
456.    ////////////////////////////////////////////////////////////////////////
457.    //
458.    //           FOURTH PART - NOISE EVALUATION
459.    //
460.    ////////////////////////////////////////////////////////////////////////
461.    //             INCLUIR ELDO - OMITIR SMASH
462.    ////////////////////////////////////////////////////////////////////////   /*
463.    begin:Flicker_Noise    //      EQ 06 Paper
464.    ////////////////////////////////////////////////////////////
465.    ///Not passou a ser um parâmetro de entrada e não mais calculado
466.    ////////////////////////////////////////////////////////////

467.    //Not     =   (ktonq * ntraps/ e_gamma)*1e+4;
468.    Sid     =   ((`P_Q * `P_Q* Not* uef* xi* 1e4) / (l*l*n*cox))*ln((ncoxktonq-qis) /
        (ncoxktonq-qid));//fator 1e4 resulta da conversao de cm-2 para m-2

469.    I(d,s)  <+   flicker_noise(Sid,1,"Sflicker");

470.    end //Flicker

471.    begin:White_Noise       //      EQ 17 Paper
472.    Siw     =   -4*(`P_K*(tnom
        +`P_CELSIUS0))*uef*w/l*((2/3*(qis*qis+qis*qid+qid*qid))-
        (ncoxktonq*(qis+qid)))/(qis+qid-2*ncoxktonq);

473.    I(d,s)  <+  white_noise(Siw,"Swhite");

474.    end //White(thermal)*/

475.    end         // analog
476.    endmodule   // acm
```

APÊNDICE B

O simulador ELDO é uma ferramenta de automação de projetos eletrônicos (*Electronic Design Automation – EDA*), que usa o LINUX como sistema operacional e, consiste em um grande conjunto de interfaces dedicadas a cada nível de abstração do projeto eletrônico. Para o desenvolvimento do trabalho foram usados os ambientes: *ICstation* , *Design Architech* (*DA IC*) e *EZwave.*

B.1 - *ICSTUDIO*

É o ambiente em que os diagramas esquemáticos são organizados (Figura B.1) nele é criado o projeto contendo as bibliotecas que serão necessárias à montagem do circuito, tendo em vista o tipo da tecnologia de fabricação. Além das bibliotecas padrão, outras podem ser criadas caso sejam necessárias. Para a simulação dos circuitos usando o código em Verilog-AMS do transistor ACM foi criada uma biblioteca para armazenar este novo componente, e foi este processo que permitiu verificar a portabilidade da linguagem.

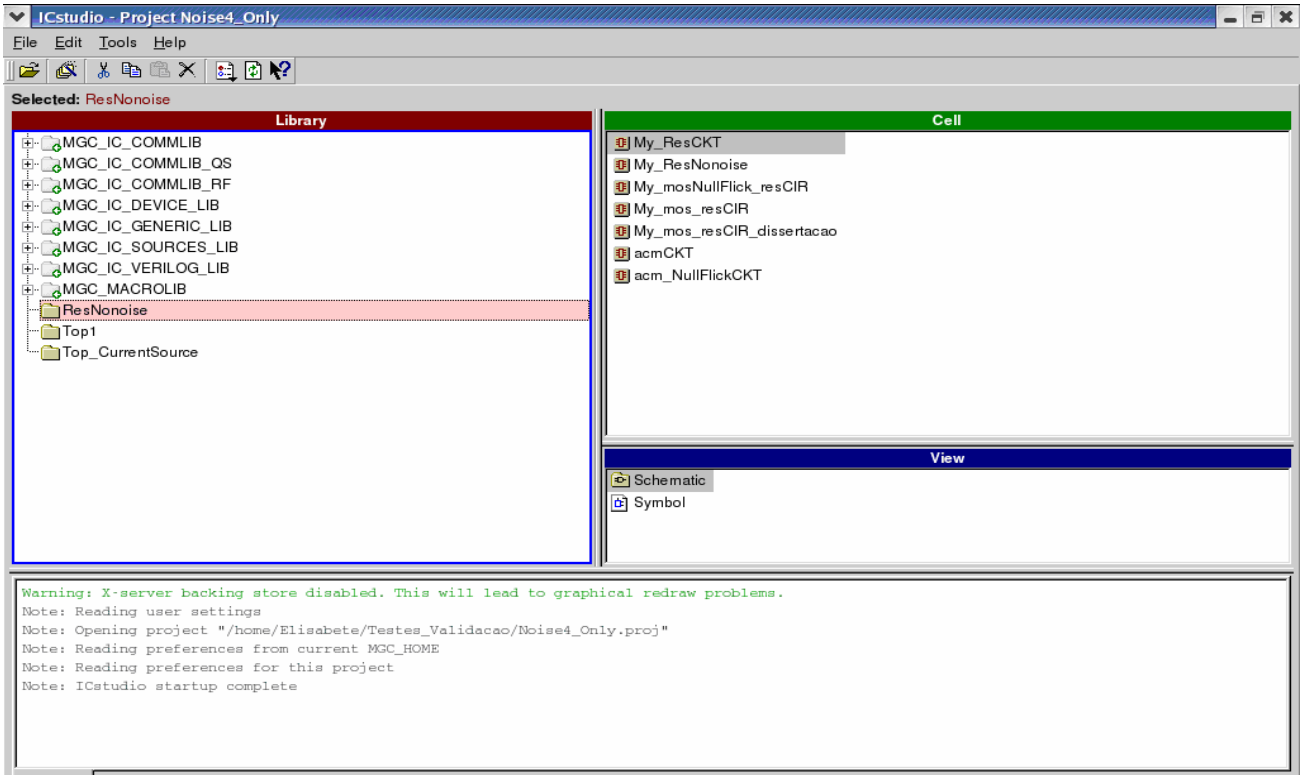


Figura B.1 - Aspecto do ambiente de trabalho *ICStudio*.

B.2 - *DA IC*

Ambiente indicado pela Figura B.3 onde o desenho do circuito esquemático é feito, nele as bibliotecas selecionadas no *ICstudio* são disponibilizadas para serem usadas na construção do circuito a ser simulado.

A partir da importação do código fonte realizada no ambiente do *ICStudio*, um símbolo gráfico é gerado, que é apresentado na Figura B.2, podendo ser editado, para melhor corresponder à simbologia normalmente usada. Os parâmetros de entrada do transistor são representados junto ao símbolo, podendo também ser editados.

Uma das características deste simulador, nesta fase, é a possibilidade de usar a conceituação da linguagem de descrição de *hardware* e as metodologias de construção *top down* e *botton up*, isto ocorre porque o circuito pode ser elaborado a partir da construção de sucessivos sub-circuitos que foram anteriormente criados. É neste ambiente, que seleciona-se o tipo de análise a ser realizada no circuito, ou seja, o circuito passa para a fase de simulação.

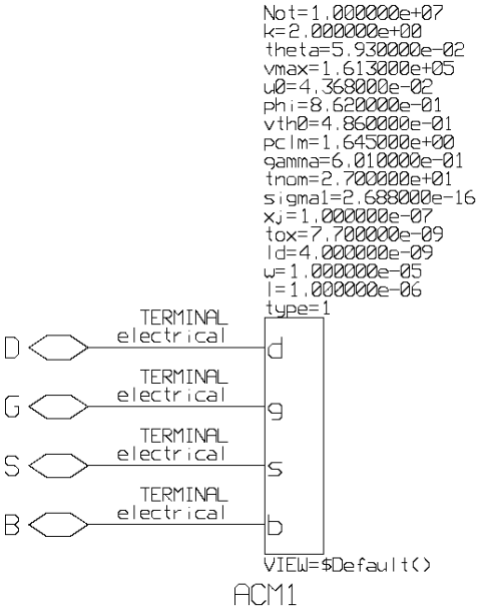


Figura B.2 - Símbolo atribuído ao transistor ACM após compilação do código em Verilog-AMS disponibilizado no ambiente *DA IC*.

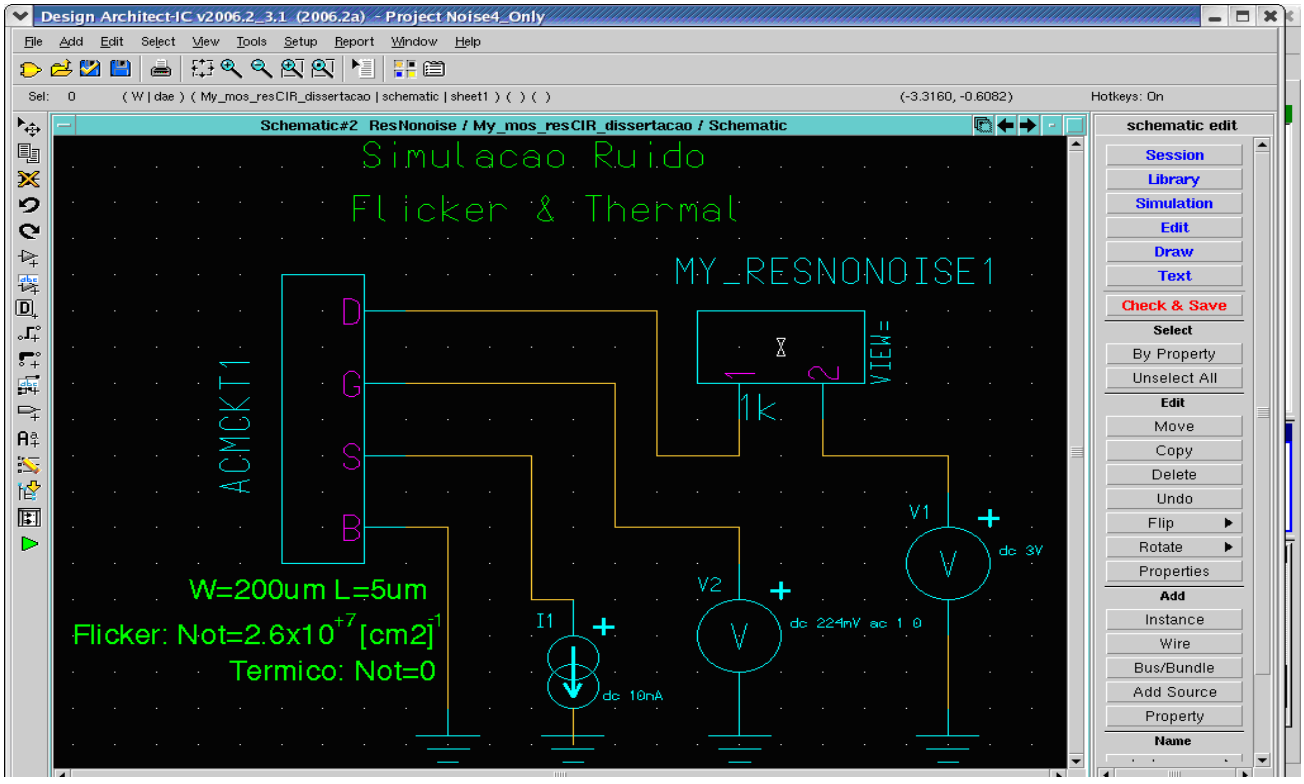


Figura B.3 - Aspecto do ambiente de trabalho *DA IC*.

B.3 - *EZWAVE*

Ambiente de trabalho destinado à visualização dos resultados gráficos decorrente das simulações, como pode ser visualizado na Figura B.4. Disponibiliza recursos que permitem realizar cálculos e obter formas de onda derivadas das originais.

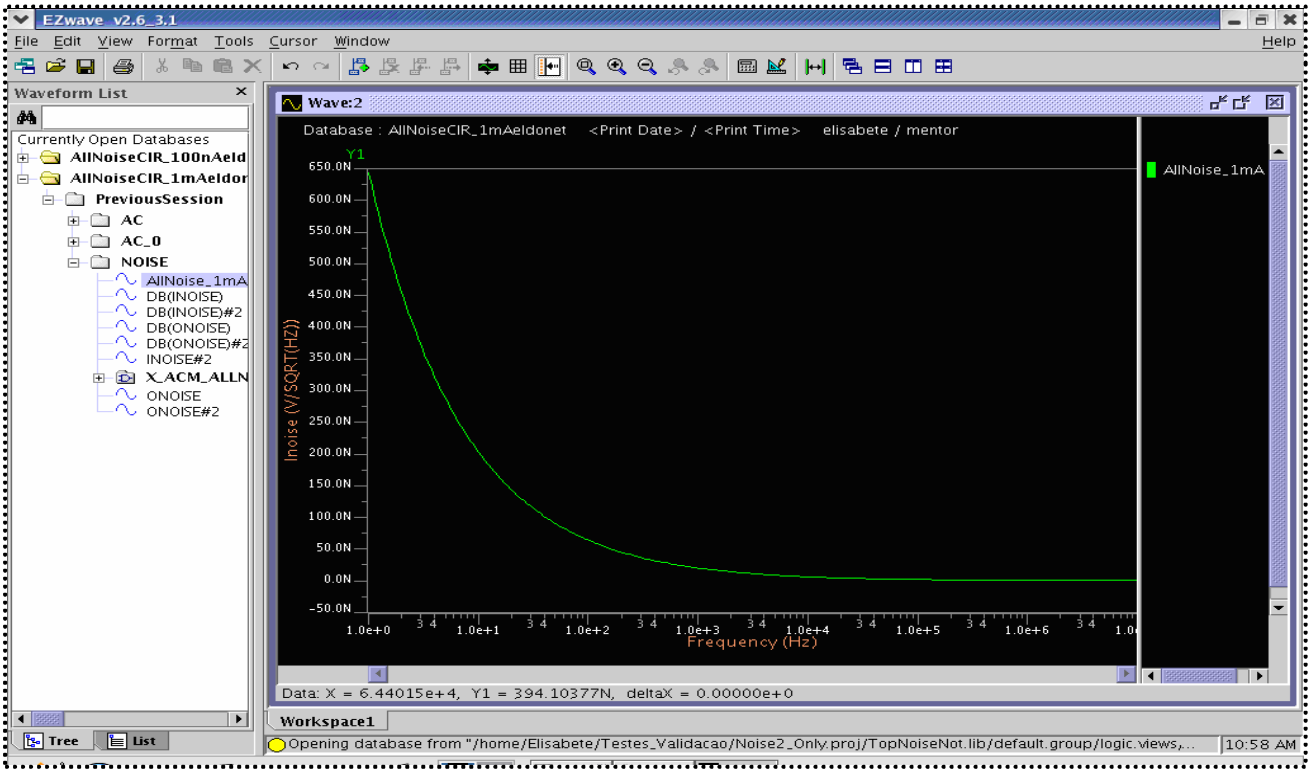


Figura B.4 - Aspecto do ambiente de trabalho *EZwave*.

B.4 - *TIPOS DE ARQUIVOS GERADOS*

O simulador ELDO gera para cada esquemático desenhado o respectivo *netlist*, que pode ser usado para realizar as simulações diretamente do terminal, sem a necessidade de estar nos ambientes anteriormente citados, estes *netlists* possuem a extensão **`<arquivo>.cir`** , **`<arquivo>.chi`** e **`<arquivo>.spi`**:

`<file>.cir` *:* arquivo de controle principal do ELDO, contém o *netlist* do circuito, comandos dos estímulos e simulação. É um arquivo compatível com o SPICE.

`<file>.chi` *:* arquivo **`log`** de saída compatível com o SPICE, contém dados em ASCII, resultados e mensagens de erro.

`<file>.spi` *:* arquivo que contém *netlist*, comandos de simulação compatível com o SPICE