INTEGRATED CIRCUITS LABORATORY

Deni Germano Alves Neto

ADVANCED COMPACT MODEL 2V0 REPORT

Florianópolis

August 31, 2023

LIST OF FIGURES

Figure 1 – Equivalent circuit of the ACM model. 4
Figure 2 – Temperature variation of the five parameters (from top to bottom): $V_{T0}(T)$, $I_S(T)$, n, $\zeta(T)$ and $\sigma(T)$) and respective fitting curves for the NMOS transistor from the 0.18 μm CMOS process in Table 9. 13
Figure 3 – Noise simulation results of the power spectral density (PSD) for SVT n-channel transistor with $W/L = 5$ μm/0.18 μm from 0.18 μm CMOS process. Operating points: in triode $V_D = 13$ mV and saturation $V_D = 1.8$ V; in weak inversion (WI) $V_G = 100$ mV, moderate inversion (MI) $V_G = V_{T0} = 528$ mV, strong inversion (SI) $V_G = 1$ V. 15
Figure 4 – Circuit for the Gummel symmetry test. 16
Figure 5 – Derivatives of I_D with respect to V_X, first to the fifth order derivatives, for V_G = 1.8 V and V_{CM} = 100 mV. 17
Figure 6 – DC characteristics $I_D \times V_{DS}$ with V_{GB} = 1.8 V. 32

LIST OF TABLES

Table 1 – Device input variables. 5
Table 2 – DC parameters. 5
Table 3 – Junction capacitance parameters. 5
Table 4 – Process parameters. 5
Table 5 – Mismatch parameters. 6
Table 6 – Temperature related parameters. 6
Table 7 – Noise parameter. 6
Table 8 – Model internal variables. 7
Table 9 – Dimensions and parameters of the transistor used in the simulations. 7

CONTENTS

1 INTRODUCTION 4

2 VARIABLES AND PARAMETERS 5

2.1 VARIABLES 5

2.2 PARAMETERS 5

2.3 MODEL INTERNAL VARIABLES 7

3 MODEL EQUATIONS IN VERILOG-A 8

3.1 DRAIN CURRENT WITH THE VELOCITY SATURATION PHENOMENON 8

3.2 TRANSCONDUCTANCES 9

3.3 DYNAMIC MODEL 10

3.3.1 Channel linearity factor 10

3.3.2 Total charges 10

3.3.3 Intrinsic capacitances 10

3.3.4 Extrinsic capacitances 11

3.3.4.1 Overlap capacitances 11

3.3.4.2 Junction capacitances 11

3.4 MISMATCH 12

3.5 TEMPERATURE 13

3.6 NOISE 14

3.7 SYMMETRY AND CONTINUITY TESTS OF DC CHARACTERISTICS . 16

References 18

APPENDIX A – VERILOG-A CODE 19

APPENDIX B – CONTINUITY BETWEEN THE TRIODE AND THE SATURATION REGIONS 31

1 INTRODUCTION

The Advanced compact MOSFET (ACM) model is a charge-based physical model [1]-[2]. All the large signal characteristics (currents and charges) and the small signal parameters ((trans)conductances and (trans)capacitances) are given by single-piece expressions in all regions of operation. ACM model preserves the structural source-drain symmetry of the transistor. It is also charge-conserving and has explicit equations for the MOSFET 8 (trans)capacitances, as shown on the equivalent circuit in Figure 1.

Some advantages of the model over the BSIM [3] model are the use of simple expressions to describe all regions of operation, the symmetry of the MOSFET, and a smaller number of device parameters. Moreover, all five parameters have a solid physical basis.

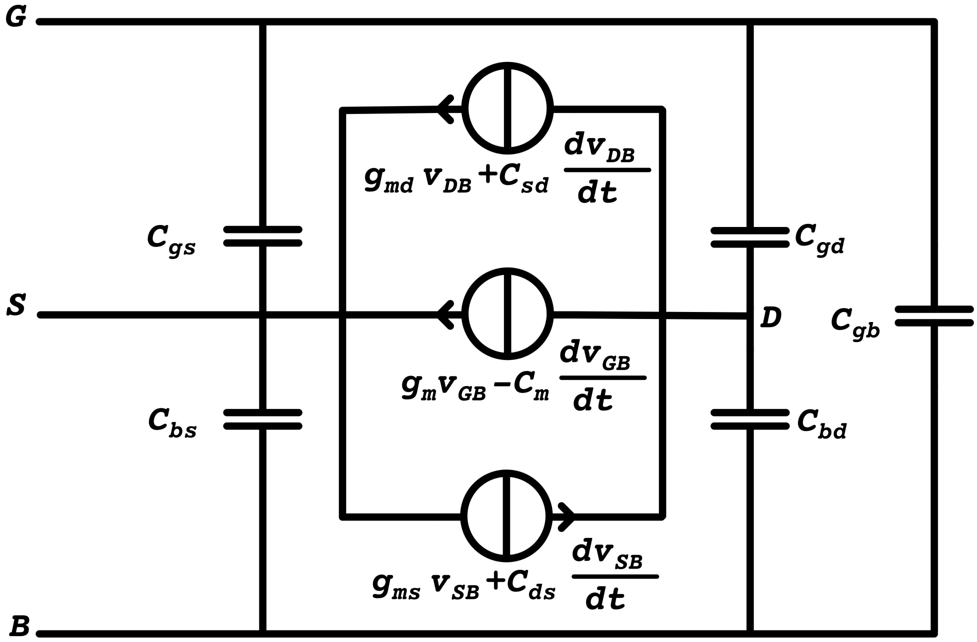

Figure 1 – Equivalent circuit of the ACM model.

2 VARIABLES AND PARAMETERS

2.1 VARIABLES

Table 1 – Device input variables.

NAME	DESCRIPTION	UNIT	DEFAULT
W	channel width	m	1E-6
L	channel length	m	1E-6
AD	drain diffusion area	m^2	2E-12
AS	source diffusion area	m^2	2E-12
PD	drain diffusion perimeter	m	10E-6
PS	source diffusion perimeter	m	10E-6
m	Number of transistor in parallel	-	1

2.2 PARAMETERS

Table 2 – DC parameters.

NAME	DESCRIPTION	UNIT	DEFAULT
VT0	threshold voltage	V	0.5
IS	specific current	A	5E-6
n	slope factor	-	1.3
Sigma	DIBL coefficient	-	0.03
Zeta	velocity saturation related parameter	-	0.05

Table 3 – Junction capacitance parameters.

NAME	DESCRIPTION	UNIT	DEFAULT
cj0	Zero bias junction capacitance	F/m^2	4e-3
cj0sw	Side-wall zero bias junction capacitance	F/m^2	4e-3
xd_mj	Drain(Source) bottom junction capacitance gradient coefficient	-	0.5
xd_mjsw	Side-wall drain(Source) bottom junction capacitance gradient coefficient	-	0.5
phi_zero	Bulk-junction built-in potential	V	0.6
phi_zerosw	Side-wall bulk-junction built-in potential	V	0.6

Table 4 – Process parameters.

NAME	DESCRIPTION	UNIT	DEFAULT
xj	Junction depth	-	150E-9
LD	Gate overlap length	-	30E-9
tox	oxide thickness	m	4E-9

Table 5 – Mismatch parameters.

NAME	DESCRIPTION	UNIT	DEFAULT
aVT0	Area related threshold voltage mismatch	V m	0
aK	Area related gain mismatch	m	0

Table 6 – Temperature related parameters.

NAME	DESCRIPTION	UNIT	DEFAULT
alphaVT0	VT0 fitting temperature coefficient	-	-0.4E-3
alphaIS	IS fitting temperature coefficient	-	1.5
alphasigma	Sigma fitting temperature coefficient	-	0.3E-6
alphazeta	Zeta fitting temperature coefficient	-	0.2E-3

Table 7 – Noise parameter.

NAME	DESCRIPTION	UNIT	DEFAULT
N_ot	Equivalent density of oxide traps	cm^{-2}	1.079E12

2.3 MODEL INTERNAL VARIABLES

Table 8 – Model internal variables.

NAME	DESCRIPTION	UNIT
umob	carrier mobility	m^2/Vs
Cox	oxide capacitance per unit area	F/m^2
e0	vacuum permittivity	F/m
eox	silicon dioxide permittivity	F/m
esi	silicon permittivity	F/m
VP	pinch-off voltage	V
PhiT	thermal voltage	V
gm	gate transconductance	A/V
gms	source transconductance	A/V
gmd	drain transconductance	A/V
alpha	channel linearity factor	-
QI	total inversion charge	C
QB	total bulk charge	C
QG	total gate charge	C
QD	total drain charge	C
QS	total source charge	C
QID	drain charge density	C
QIS	source charge density	C
qD	normalized drain charge density	C
qS	normalized source charge density	C
Cgs	gate-source capacitance	F
Cgd	gate-drain capacitance	F
Csd	source-drain capacitance	F
Cds	drain-source capacitance	F
Cgb	gate-bulk capacitance	F
Cbd	bulk-drain capacitance	F
Cbs	bulk-source capacitance	F
Cgd_ov	gate-drain overlap capacitance	F
Cgs_ov	gate-source overlap capacitance	F
Cjdb	drain-bulk junction capacitance	F
Cjsb	source-bulk junction capacitance	F

Table 9 – Dimensions and parameters of the transistor used in the simulations.

Transistor	W/L [μm/μm]	I_S [μA]	V_{T0} [mV]	n	σ	ζ
NMOS	5/0.18	5.52	528	1.37	0.025	0.056

3 MODEL EQUATIONS IN VERILOG-A

3.1 DRAIN CURRENT WITH THE VELOCITY SATURATION PHENOMENON

The ACM considers both the drift and diffusion components of the current and the saturation velocity phenomenon [2], yielding (1).

$$I_D = I_S \frac{(q_S + q_D + 2)}{1 + \zeta\,|q_S - q_D|}(q_S - q_D) \tag{1}$$

The normalized inversion charge densities at source (q_S) and drain (q_D) are defined as the inversion charge densities $Q_{IS(D)}$ normalized to the thermal charge density $-nC_{ox}\phi_t$ as given in (2).

$$q_{S(D)} = \frac{Q_{IS(D)}}{-nC_{ox}\phi_t} \tag{2}$$

where C_{ox} is the the oxide capacitance per unit area, n is the slope factor and ϕ_t is the thermal voltage.

The specific current I_S, given in (3), is related to the gate width W and length L, and to technology parameters. μ is the carrier mobility.

$$I_S = \mu C_{ox} n \frac{\phi_t^2}{2}\frac{W}{L} \tag{3}$$

Parameter ζ is a short-channel parameter associated with the velocity saturation (v_{sat}) phenomenon. It is defined by (4) as the ratio of a diffusion-related velocity to the saturation velocity.

$$\zeta = \frac{\mu\phi_t/L}{v_{sat}} \tag{4}$$

The maximum drain current I_{Dsat} that can flow through the channel occurs when the velocity of the carriers is saturated and is related to the drain charge density Q_{chsat} as

$$I_{Dsat} = -W v_{sat} Q_{sat} \tag{5}$$

From (1) and (5), the saturated charge density normalized to the thermal charge density is given by

$$q_{sat} = q_{S(D)} + 1 + \frac{1}{\zeta} - \sqrt{(1+\frac{1}{\zeta})^2 + \frac{2q_{S(D)}}{\zeta}} \tag{6}$$

The unified charge-control model (UCCM) [2] expresses the relationship between the terminal voltages and the normalized charge densities at the source and drain of a long channel transistor. In the UCCM, the source and drain normalized charge densities q_S and q_D are calculated using (7)

$$\frac{V_P - V_{S(D)B}}{\phi_t} = q_{S(D)} - 1 + \ln q_{S(D)} \tag{7}$$

Applying (7) to the source and drain, we obtain the symmetrical expression (8), that links q_S and q_D with the potential drop V_{DS} along the channel.

$$\frac{V_{DS}}{\phi_t} = q_S - q_D + \ln\left(\frac{q_S}{q_D}\right) \tag{8}$$

For a long channel transistor, it is clear from (8), that $q_D \to 0$ when $V_{DS} \to \infty$. The classical way to deal with velocity saturation using (8) is to calculate the drain-to-source saturation voltage substituting q_D by its saturation value. In the present model, we avoid the definition of a saturation voltage substituting $q_{S(D)}$ by $q_{S(D)} - q_{Dsat}$ in (8), getting (9)

$$\frac{V_{DS}}{\phi_t} = q_S - q_D + \ln\left(\frac{q_S - q_{sat}}{q_D - q_{sat}}\right) \tag{9}$$

Equation (9) has the necessary properties to model the effect of the saturation velocity of the carriers using single-piece equations. In effect, $V_{DS} = 0$ for $q_S = q_D$, $q_D \to q_{Dsat}$ when $V_{DS} \to \infty$, and if we interchange q_S and q_D, V_{DS} changes sign.

The pinch-off voltage V_P is linearly approximated by (10). The introduction of σ, the drain-induced barrier lowering (DIBL) coefficient, at both source and drain, keeps the device symmetry. V_{T0} is the equilibrium threshold voltage.

$$V_P = \frac{V_{GB} - V_{T0} + \sigma V_{DB} + \sigma V_{SB}}{n} \tag{10}$$

3.2 TRANSCONDUCTANCES

Transconductances, defined as the derivatives of the current with respect to the control voltages, are crucial for reliable and efficient convergence of the DC analysis.

$$g_m = \frac{\partial I_D}{\partial V_G}; g_{ms} = -\frac{\partial I_D}{\partial V_S}; g_{md} = \frac{\partial I_D}{\partial V_D}; \tag{11}$$

$$\begin{aligned} g_m = \frac{2I_S}{n\phi_t\left[1+\zeta(q_S-q_D)\right]^2} \Bigg\{ & \left[1 + q_S + \frac{\zeta}{2}(q_S - q_D)^2\right] \frac{q_S - q_{sat}}{1 + q_S - q_{sat}} \\ & - \left[1 + q_D - \frac{\zeta}{2}(q_S - q_D)^2\right] \frac{q_D - q_{sat}}{1 + q_D - q_{sat}} \Bigg\} \end{aligned} \tag{12}$$

$$\begin{aligned} g_{ms} = \frac{2I_S}{\phi_t\left[1+\zeta(q_S-q_D)\right]^2} \Bigg\{ & \left[1 + q_S + \frac{\zeta}{2}(q_S - q_D)^2\right] \left(1 - \frac{\sigma}{n}\right) \frac{q_S - q_{sat}}{1 + q_S - q_{sat}} \\ & + \left[1 + q_D - \frac{\zeta}{2}(q_S - q_D)^2\right] \frac{\sigma}{n} \frac{q_D - q_{sat}}{1 + q_D - q_{sat}} \Bigg\} \end{aligned} \tag{13}$$

$$\begin{aligned} g_{md} = \frac{2I_S}{\phi_t\left[1+\zeta(q_S-q_D)\right]^2} \Bigg\{ & \left[1 + q_S + \frac{\zeta}{2}(q_S - q_D)^2\right] \frac{\sigma}{n} \frac{q_S - q_{sat}}{1 + q_S - q_{sat}} \\ & + \left[1 + q_D - \frac{\zeta}{2}(q_S - q_D)^2\right] \left(1 - \frac{\sigma}{n}\right) \frac{q_D - q_{sat}}{1 + q_D - q_{sat}} \Bigg\} \end{aligned} \tag{14}$$

3.3 DYNAMIC MODEL

By providing explicit expressions for intrinsic charges and transcapacitances, transient simulations can be performed more efficiently, enabling faster and more accurate analysis of the device under consideration.

3.3.1 Channel linearity factor

$$i_{dsat} = \frac{2}{\zeta} q_{sat} \tag{15}$$

$$\alpha = \frac{q_D + 1 - q_{sat}\frac{id}{i_{dsat}}}{q_S + 1 - q_{sat}\frac{id}{i_{dsat}}} \tag{16}$$

3.3.2 Total charges

$$Q_I = -n\phi_t C_{ox} W L \left[\frac{2}{3} \frac{1+\alpha+\alpha^2}{1+\alpha}(q_S + 1 - q_{sat}) - 1 \right] - \frac{L I_D}{v_{sat}} \tag{17}$$

$$Q_B = -\frac{n-1}{n} Q_I \tag{18}$$

$$Q_G = -Q_B - Q_I \tag{19}$$

$$Q_D = -n\phi_t C_{ox} W L \left[\frac{2}{15} \frac{2+4\alpha+6\alpha^2+3\alpha^3}{(1+\alpha)^2}(q_S + 1 - q_{sat}) - \frac{1}{2} \right] - \frac{L I_D}{2 v_{sat}} \tag{20}$$

$$Q_S = Q_I - Q_D \tag{21}$$

3.3.3 Intrinsic capacitances

$$C_{gso} = \frac{2}{3} W L C_{ox} \frac{1+2\alpha}{(1+\alpha)^2} \frac{q_S - q_{sat}}{1 + q_S - q_{sat}} \tag{22}$$

$$C_{gdo} = \frac{2}{3} W L C_{ox} \frac{\alpha^2+2\alpha}{(1+\alpha)^2} \frac{q_D - q_{sat}}{1 + q_D - q_{sat}} \tag{23}$$

$$C_{sdo} = -\frac{4}{15} n W L C_{ox} \frac{\alpha+3\alpha^2+\alpha^3}{(1+\alpha)^3} \frac{q_D - q_{sat}}{1 + q_D - q_{sat}} \tag{24}$$

$$C_{dso} = -\frac{4}{15} n W L C_{ox} \frac{1+3\alpha+\alpha^2}{(1+\alpha)^3} \frac{q_S - q_{sat}}{1 + q_S - q_{sat}} \tag{25}$$

$$C_{gs} = C_{gso}\left(1 - \frac{\sigma}{n}\right) - C_{gdo}\frac{\sigma}{n} + \frac{Lg_{ms}}{3nv_{sat}}\frac{(1-\alpha)^2}{(1+\alpha)^2} \tag{26}$$

$$C_{gd} = C_{gdo}\left(1 - \frac{\sigma}{n}\right) - C_{gso}\frac{\sigma}{n} - \frac{Lg_{md}}{3nv_{sat}}\frac{(1-\alpha)^2}{(1+\alpha)^2} \tag{27}$$

$$C_{gb} = \frac{n-1}{n}\left[WLC_{ox} - C_{gso} - C_{gdo} - \frac{Lg_m}{3v_{sat}}\frac{(1-\alpha)^2}{(1+\alpha)^2}\right] + \frac{2\sigma}{n}\left[(n-1)WLC_{ox} + C_{gso} + C_{gdo}\right] \tag{28}$$

$$C_{bs} = (n-1)C_{gs} \tag{29}$$

$$C_{bd} = (n-1)C_{gd} \tag{30}$$

$$C_{sd} = C_{sdo}\left(1 - \frac{\sigma}{n}\right) + \frac{Lg_{md}}{30v_{sat}}\frac{(3+7\alpha)(1-\alpha)^2}{(1+\alpha)^3} \tag{31}$$

$$C_{ds} = C_{dso}\left(1 - \frac{\sigma}{n}\right) - \frac{Lg_{ms}}{30v_{sat}}\frac{(3\alpha+7)(1-\alpha)^2}{(1+\alpha)^3} \tag{32}$$

$$C_m = \frac{C_{sd} - C_{ds}}{n} \tag{33}$$

3.3.4 Extrinsic capacitances

3.3.4.1 Overlap capacitances

$$Cgsov = C_{ox}WL_D \tag{34}$$

$$Cgdov = C_{ox}WL_D \tag{35}$$

3.3.4.2 Junction capacitances

Built-in potential:

$$\phi_0 = \phi_t \ln\left(\frac{N_A N_D}{n_i^2}\right) \tag{36}$$

Sidewall built-in potential:

$$\phi_{0sw} = \phi_t \ln\left(\frac{N_{Asw} N_D}{n_i^2}\right) \tag{37}$$

Zero bias junction capacitance:

$$C_{j0} = \sqrt{\left(\frac{e_{si}q}{2}\frac{N_A N_D}{N_A + N_D}\frac{1}{\phi_0}\right)} \tag{38}$$

Sidewall zero bias junction capacitance:

$$C_{j0sw} = \sqrt{\left(\frac{e_{si}q}{2}\frac{N_{Asw} N_D}{N_{Asw} + N_D}\frac{1}{\phi_{0sw}}\right)} \tag{39}$$

Drain-Bulk:
if $V_{DB} > 0$

$$C_{jdb} = C_{j0}ADe^{-m_j \ln\left(\frac{1+V_{DB}}{\phi_0}\right)}; \tag{40}$$

$$C_{jdbsw} = C_{j0sw}x_j PDe^{-m_{jsw} \ln\left(\frac{1+V_{DB}}{\phi_{0sw}}\right)}; \tag{41}$$

else

$$C_{jdb} = C_{j0}AD * (1.0 - m_j \frac{1+V_{DB}}{\phi_0}); \tag{42}$$

$$C_{jdbsw} = Cj0sw * xj * PD * (1.0 - m_{jsw}\frac{V_{DB}}{\phi_{0sw}}); \tag{43}$$

Source-Bulk:
if $V_{SB} > 0$

$$C_{jsb} = C_{j0}ASe^{-m_j \ln\left(\frac{1+V_{SB}}{\phi_0}\right)}; \tag{44}$$

$$C_{jsbsw} = C_{j0sw}x_j PSe^{-m_{jsw} \ln\left(\frac{1+V_{SB}}{\phi_{0sw}}\right)}; \tag{45}$$

else

$$C_{jsb} = C_{j0}AS\left(1 - m_j \frac{V_{SB}}{\phi_0}\right); \tag{46}$$

$$C_{jsbsw} = C_{j0sw}x_j PS\left(1 - m_{jsw}\frac{V_{SB}}{\phi_{0sw}}\right); \tag{47}$$

3.4 MISMATCH

$$V_{T0}(aV_{T0}) = \frac{aV_{T0}}{\sqrt{W * L}} \tag{48}$$

$$I_S(a_K) = \frac{I_S a_K}{\sqrt{W * L}} \tag{49}$$

3.5 TEMPERATURE

Each of the five parameters was extracted for a temperature range of -25°C to 125°C. Fig. 2 shows the temperature dependence of each parameter. Fitting curves were found for each parameter to incorporate the temperature variation in the simulation model.

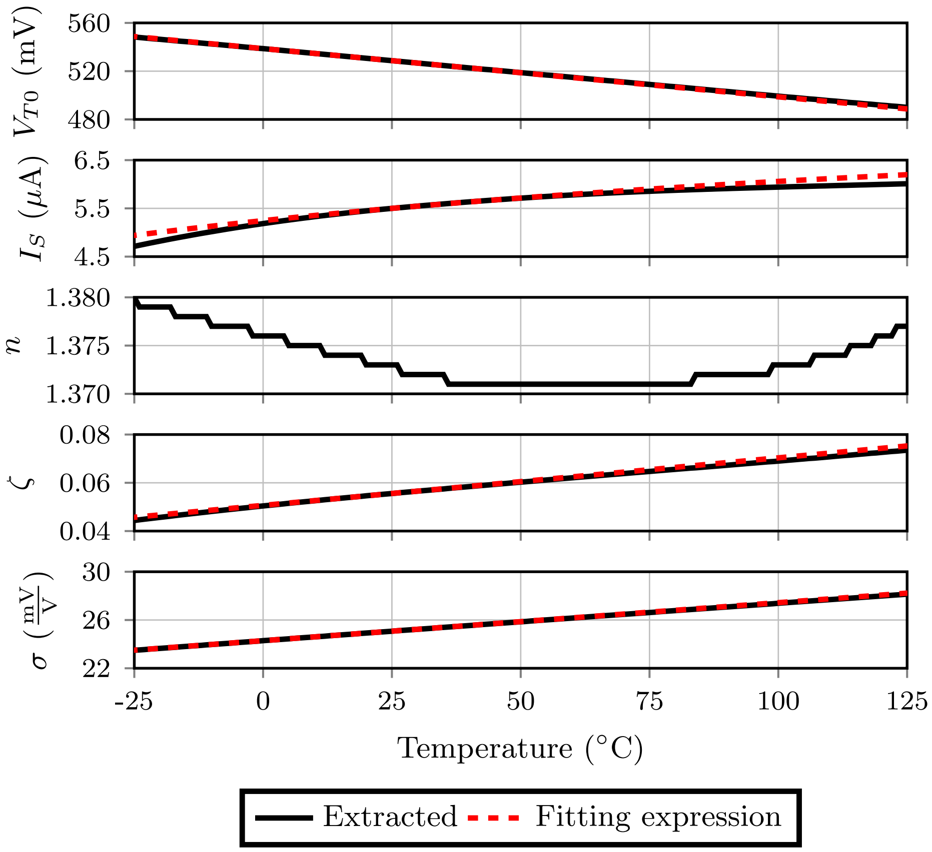

Figure 2 – Temperature variation of the five parameters (from top to bottom): $V_{T0}(T)$, $I_S(T)$, n, $\zeta(T)$ and $\sigma(T)$) and respective fitting curves for the NMOS transistor from the 0.18 μm CMOS process in Table 9.

The threshold voltage dependence on the temperature, which is linear as expected from its physical definition [4] - [5], is given by (50).

$$V_{T0}(T) = V_{T0}(T_{ref}) + \alpha_{V_{T0}}(T - T_{ref}) \tag{50}$$

The specific current varies with temperature partially due to the mobility variation with temperature. The fitting expression (51) was derived based on the assumption that the mobility is proportional to $T^{-\alpha}$ [4], in which α is a fitting coefficient.

$$I_S(T) = I_S(T_{ref})\,[1 + \frac{(2-\alpha)}{T}(T - T_{ref})] \tag{51}$$

The slope factor n is almost constant over the whole temperature range. The short-channel parameters ζ and σ also present a linear dependency on the temperature.

The fitting expressions have been determined for a reference temperature $T_{ref} = 300$ K. The temperature coefficients employed are $\alpha_{V_{T0}} = -0.4$ mV/K, $\alpha =$1.5, $\alpha_{\zeta} = 0.2 \times 10^{-3}$ K^{-1}, and $\alpha_{\sigma} = 0.32 \times 10^{-6}$ K^{-1}. With these fitting parameters, the designer extracts the five parameters at T_{ref} to simulate MOS circuits over a wide range of temperatures.

3.6 NOISE

Thermal and flicker noise valid for all regions of operation were included in the model [2], as shown in the equations below

$$\frac{\overline{i_d^2}}{\Delta f} = \frac{4kT\mu Q_I}{L^2} \tag{52}$$

$$\frac{\overline{i_d^2}}{\Delta f} = \frac{q_e^2 N_{ot}\mu}{L^2 n C_{ox} I_D} \ln\left(\frac{nC_{ox}\phi_t - Q_{IS}}{nC_{ox}\phi_t - Q_{ID}}\right)\frac{I_D^2}{f} \tag{53}$$

Fig. 3 shows the simulated PSDs of the channel noise for various operating points, which are consistent with the expected MOSFET noise models, despite some differences between the 5PM and BSIM.

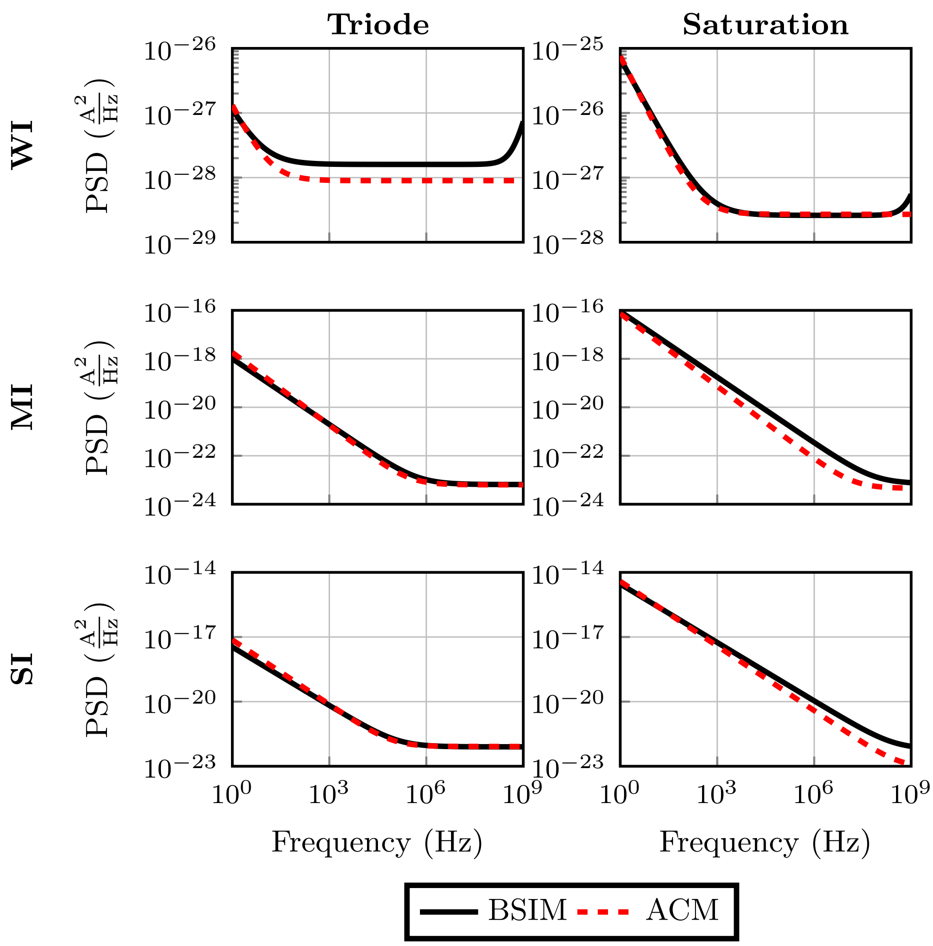

Figure 3 – Noise simulation results of the power spectral density (PSD) for SVT n-channel transistor with W/L = 5 μm/0.18 μm from 0.18 μm CMOS process. Operating points: in triode $V_D = 13$ mV and saturation $V_D = 1.8$ V; in weak inversion (WI) $V_G = 100$ mV, moderate inversion (MI) $V_G = V_{T0} = 528$ mV, strong inversion (SI) $V_G = 1$ V.

3.7 SYMMETRY AND CONTINUITY TESTS OF DC CHARACTERISTICS

MOS transistors are symmetrical devices, so their models must also be symmetrical, i.e., the drain and source terminals can be chosen arbitrarily, and the transistor characteristics must remain the same regardless of choice. Furthermore, the transition between forward $(V_{DS} > 0)$ and reverse $(V_{DS} < 0)$ operations must be continuous. The term , in the denominator of (1) causes a discontinuity in the derivative of the current, with respect to V_{DS}, at $V_{DS} = 0$. To avoid this problem, we approximate the absolute value function by

$$1 + \zeta\,|q_S - q_D| \cong 1 + \sqrt{\zeta^2 \left(q_S - q_D\right)^2 + \epsilon^2} \tag{54}$$

where where $\epsilon = 0.1$. The Gummel symmetry test is used to demonstrate the quality of the model [6]. The transistor is symmetrically biased in this test with a voltage $V_{DS} = 2V_X$ and a common model voltage V_{CM}, as shown in figure 4. Changing the voltage V_X between negative and positive values, the transistor will pass symmetrically from the reverse region to the direct region. The model must present a continuous transition for the drain current and its derivatives with respect to Vx.

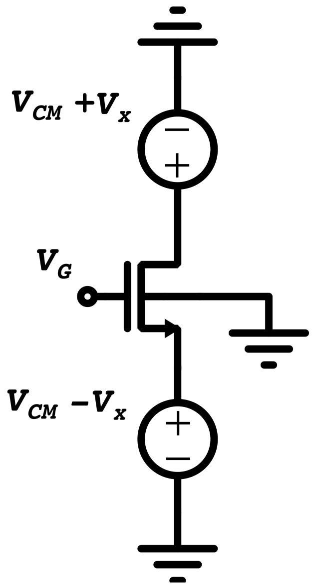

Figure 4 – Circuit for the Gummel symmetry test.

Figures 5 shows that the ACM derivatives are continuous, presenting smooth transitions from the first to the fifth derivative.

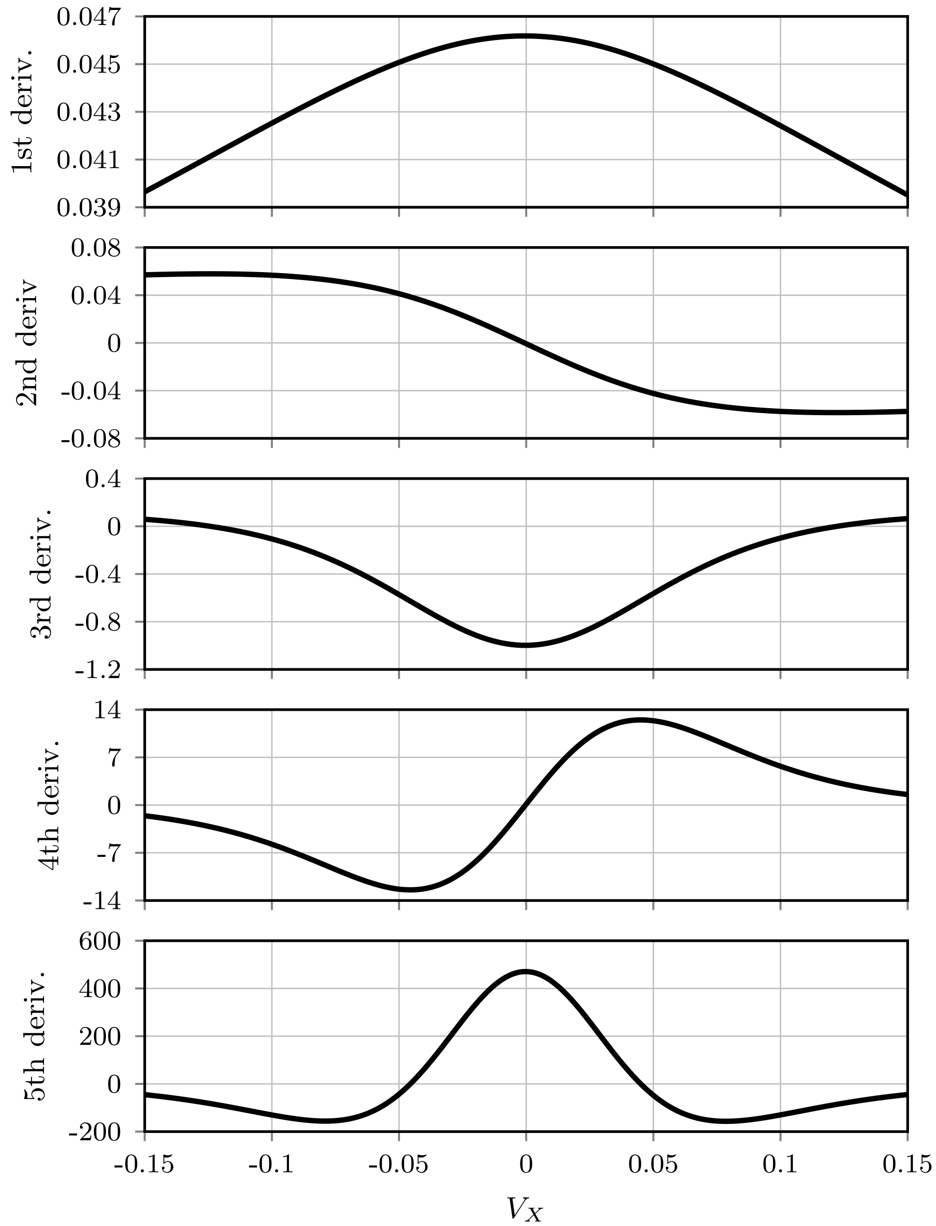

Figure 5 – Derivatives of I_D with respect to V_X, first to the fifth order derivatives, for $V_G =$ 1.8 V and $V_{CM} =$ 100 mV.

REFERENCES

[1] M. C. Schneider and C. Galup-Montoro, *CMOS analog design using all-region MOSFET modeling*, 1st. USA: Cambridge University Press, 2010, ISBN: 052111036X. DOI: 10.1017/CBO9780511803840.

[2] C. Galup-Montoro and M. C. Schneider, *Mosfet modeling for circuit analysis and design*. USA: World Scientific Publishing Co., Inc., 2007, ISBN: 9789812568106.

[3] X. Xi, M. Dunga, J. He, *et al.*, *BSIM4.5.0 MOSFET Model - User's Manual*. Jan. 2003.

[4] E. Vittoz, "MOS transistor: model and modes of operation", in *MEAD course on "Advanced Analog CMOS IC Design"*, Aug. 2019.

[5] J. Raposo, F. Alves, and P. Ferreira, "A general gm/Id temperature-aware design methodology using 180 nm CMOS up to 250 °C", *Journal of Integrated Circuits and Systems*, vol. 17, pp. 1–9, Apr. 2022. DOI: 10.29292/jics.v17i1.552.

[6] O. da Costa Gouveia-Filho, A. Cunha, M. Schneider, and C. Galup-Montoro, "Advanced compact model for short-channel mos transistors", in *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No.00CH37044)*, 2000, pp. 209–212. DOI: 10.1109/CICC.2000.852650.

[7] B. Ankele, W. Holzl, and P. O'Leary, "Enhanced mos parameter extraction and spice modelling for mixed analogue and digital circuit simulation", in *Proceedings of the 1989 International Conference on Microelectronic Test Structures*, 1989, pp. 73–78. DOI: 10.1109/ICMTS.1989.39285.

APPENDIX A – VERILOG-A CODE

```
//*****************************************************************************
// *  ACM NMOS model (Verilog-A)                                             *
// *  07/2023     V2.0.0                                                     *
//*****************************************************************************

// ****************************************************************************
// * Copyright under the ECL-2.0 license                                     *
// * Universidade Federal de Santa Catarina                                  *
// *                                                                         *
// * Current developers:  Deni Germano Alves Neto (Doctoral student, UFSC)   *
// *                      Cristina Missel Adornes (Doctoral student, UFSC)   *
// *                      Gabriel Maranhao        (Doctoral student, UFSC)   *
// *                                                                         *
// * Project Supervisors:  Prof. Carlos Galup-Montoro                        *
// *                       Prof. Marcio Cherem Schneider                     *
// ****************************************************************************

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

// function of the algorithm 443 to calculate de normalize charge densities
`define algo_443(Z,qn) \
 if(Z < 0.7385) begin \
  numeratorD = Z + (4.0/3.0)*Z*Z; \
  denominatorD = 1.0 + (7.0/3.0)*Z+(5.0/6.0)*Z*Z; \
  WnD = numeratorD/denominatorD; \
 end else begin \
  numeratorD = ln(Z)*ln(Z)+2.0*ln(Z)-3.0; \
  denominatorD = 7.0*ln(Z)*ln(Z) + 58.0*ln(Z) +127.0; \
  WnD = ln(Z) - 24.0*(numeratorD/denominatorD); \
 end \
ZnD = ln(Z) -WnD - ln(WnD); \
TermD = ZnD/(1.0 + WnD); \
numeratorED = 2.0*(1.0+WnD)*(1.0+WnD+(2.0/3.0)*ZnD)-ZnD; \
denominatorED = 2.0*(1.0+WnD)*(1.0+WnD+(2.0/3.0)*ZnD)-2*ZnD; \
EnD = TermD*(numeratorED/denominatorED); \
qn = (WnD*(1+EnD)); \

module nmos_ACM(D, G, S, B);

// pinout definition
inout D,G,S,B;

// electrical nodes definition
electrical D,G,S,B;

//Branches
branch(G,B) BGB;
branch(D,B) BDB;
branch(S,B) BSB;
branch(D,S) BDS;
branch(G,D) BGD;
branch(G,S) BGS;

// Device input variables
(*desc= "Channel width", units = "m", type = "instance"*) parameter real W = 1e-6 from
    [0:inf];
```

```
(*desc= "Channel length", units = "m", type = "instance"*) parameter real L = 1e-6 from [0:inf];
(*desc= "Drain diffusion area", units = "m^2", type = "instance"*)  parameter real AD = 2e-12 from [0:inf];
(*desc= "Source diffusion area", units = "m^2", type = "instance"*) parameter real AS = 2e-12 from [0:inf];
(*desc= "Drain diffusion perimeter", units = "m", type = "instance"*)  parameter real PD = 10e-6 from [0:inf];
(*desc= "Source diffusion perimeter", units = "m", type = "instance"*) parameter real PS = 10e-6 from [0:inf];
(*desc= "Number of transistor in parallel", units = "", type = "instance"*) parameter real m = 1 from [1:inf];

// DC related parameters
(*desc= "Threshold voltage", units = "V", type = "instance"*) parameter real VT0 = 0.5 from [0:inf];
(*desc= "Specific current", units = "A", type = "instance"*) parameter real IS = 5e-6 from [0:inf];
(*desc= "Slope factor", type = "instance"*) parameter real n = 1.3 from [1:3];
(*desc= "Sigma", type = "instance"*) parameter real sigma = 0.03 from [0:inf];
(*desc= "Zeta", type = "instance"*) parameter real zeta = 0.05 from [0:inf];

// Junction capacitance parameters
(*desc= "Zero bias junction capacitance", units = "F/m^2", type = "instance"*) parameter real Cj0 = 4e-3 from [0:inf];
(*desc= "Side-wall zero bias junction capacitance", units = "F/m^2", type = "instance"*)  parameter real Cj0sw= 4e-3 from [0:inf];
(*desc= "Drain(Source) bottom junction capacitance gradient coeficient", units = "", type = "instance"*)  parameter real xd_mj = 0.5 from [0:inf];
(*desc= "Side-wall drain(Source) bottom junction capacitance gradient coeficient", units = "", type = "instance"*)  parameter real xd_mjsw= 0.5 from [0:inf];
(*desc= "Bulk-junction built-in potential", units = "V", type = "instance"*)  parameter real phi_zero = 0.6 from [0:inf];
(*desc= "Side-wall bulk-junction built-in potential", units = "V", type = "instance"*) parameter real phi_zerosw= 0.6 from [0:inf];

// Process related parameters
(*desc= "Junction depth", units = "m", type = "instance"*)  parameter real xj = 150e-9 from [0:inf];
(*desc= "Gate overlap", units = "m", type = "instance"*)  parameter real LD = 30e-9 from [0:inf];
(*desc= "Thickness of oxide", units = "m", type = "instance"*)  parameter real tox = 4e-9 from [0:inf];

//Mismatch related parameters
(*desc= "Area related threshold voltage mismatch", units = "V*m", type = "instance"*) parameter real aVT0 = 0 from [-inf:inf];
(*desc= "Area related gain mismatch", units = "m", type = "instance"*)  parameter real aK = 0 from [0:inf];

//Temperature related parameters
(*desc= "VT0 fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphaVT0 = -0.4e-3 from [-1:1];
(*desc= "IS fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphaIS = 1.5 from [0:inf];
(*desc= "Sigma fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphasigma = 0.3e-6 from [0:inf];
(*desc= "Zeta fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphazeta = 0.2e-3 from [0:inf];

```

```
//Noise related parameter
(*desc= "Equivalent density of oxide traps", units = "m^-2", type = "instance"*) parameter real N_ot = 1e12 from [0:inf];

// Intern variables
real VP, PhiT, VDB, VSB, VGB, VGD, VGS, VDS, VSD, VXB, VYB;
real qsat, qidsat, q1, q2,qS, qD;
real X, Y, numeratorD, denominatorD, TermD, numeratorED, denominatorED, ZnD, EnD, WnD;
real id, ID, e0, eox, esi, epsi, We;
real umob,vsat, Cox, kboltz, q_e;
real gm_frac1, gm_frac2, gm_frac3, gms,gmd, gm;
real alpha, alpha_frac, alpha_frac2, alpha_frac3;
real QI, QD, QS, QG, QB, QID, QIS;
real Cgso, Cgdo, Cgbo, Cbso, Cbdo, Cdso, Csdo;
real Cgd, Cgs, Cgb, Cbs, Cbd, Cds, Csd, Cm;
real Cgs_ov, Cgd_ov, Cjdb, Cjsb, Cjdb_sw, Cjsb_sw, Cjsw;
real qovd, qovs, qjd, qjs;
real VT0_T, IS_T, sigma_T, zeta_T, Tin, Tref;
real func_thermal, func_flicker, ND_thermal, ND_flicker;
real gmin;

analog begin

//Constants
e0 = `P_EPS0; // Vacuum permittivity
eox = 3.97*e0; // Silicon dioxide permittivity
esi = 11.9*e0; // Silicon permittivity
Tref = 300.15; // Reference temperature
q_e = `P_Q; // Elementary charge
kboltz = `P_K; // Boltzmann constant
epsi = 0.1; // Constant used for device symmetry
gmin = $simparam("gmin",0); // gmin is placed in parallel with drain-bulk and source-bulk current sources to increase the numerical convergence
We = m*W; // Equivalent W of the parallel devices

//Temperature variation
PhiT = $vt($temperature); // Thermal voltage
Tin = $temperature; //Simulator temperature
VT0_T = VT0 + alphaVT0*(Tin-Tref) + aVT0/sqrt(We*L); // Temperature variation and mismatch on VT0
IS_T = IS*(1+ ((2-alphaIS)/Tin)*(Tin-Tref)) + (IS*aK)/sqrt(We*L); // Temperature variation and mismatch on IS
sigma_T = sigma+ alphasigma*(Tin-Tref); // Temperature variation on sigma
zeta_T = zeta + alphazeta*(Tin-Tref); // Temperature variation on zeta

//Control voltages from branches
VGB = V(BGB);
VDB = V(BDB);
VSB = V(BSB);
VDS = V(BDS);
VGD = V(BGD);
VGS = V(BGS);
VSD = -VDS;
VXB = min(VSB,VDB);
VYB = max(VDS,VSD);

// Intern process parameters
Cox = eox/tox; // [F/m^2]
umob = (IS_T*L)/(Cox*n*PhiT*PhiT*We*0.5); //mobility [m^2/V.s]
vsat = (umob*PhiT)/(L*zeta_T); //[m^2/s]
```

```
//Pinch-off voltage
VP = (VGB - VT0_T + sigma_T*(VDB +VSB) )/n;

////q1 from algorithm 443 and UCCM_LC
X = exp(((VP - VXB)/PhiT)+1.0);
`algo_443(X,q1)

//Calculating qdsat
qsat = q1 + 1.0 + (1.0/zeta_T) - sqrt(((1+(1/zeta_T))*(1+(1/zeta_T)))+((2*q1)/zeta_T));

////q2 from algorithm 443 and UCCM_SC
Y = (((q1-qsat))*exp(-VYB/PhiT)*exp(q1-qsat));
`algo_443(Y,q2)
q2 = q2 + qsat;

//Normalized densities charges for a symmetric model
 if (VDS >= 0) begin

  qS = q1;
  qD = q2;
 end
 else begin
  qD = q1;
  qS = q2;
 end

///////////// Calculating Drain Current ID //////////////////////
id = ((qD+qS+2.0)/(1.0+ sqrt((zeta_T*(qS-qD))*zeta*((qS-qD))+(epsi*epsi))))*(qS - qD);
ID = m*IS_T*id;
I(D,S) <+ ID;

///////////// Transconductances ///////////////////
qidsat =(zeta_T/2.0)*id;

//gm_frac1 to gm_frac3 are intermediate fractions of the transconductances
gm_frac1 = (2.0*IS_T)/(PhiT*(1.0+zeta_T*(q1-q2))*(1.0+zeta_T*(q1-q2)));
gm_frac2 = (1.0+q1)+(zeta_T/2.0)*(((q1-q2)*(q1-q2)));
gm_frac3 = (1.0+q2)-(zeta_T/2.0)*(((q1-q2)*(q1-q2)));
gms =
     gm_frac1*((gm_frac2)*(1.0-(sigma_T/n))*((q1-qidsat)/(1+q1-qidsat))+((gm_frac3)*(sigma_T*n)*((q2-
     //Source transconductance
gmd =
     gm_frac1*((gm_frac2)*(sigma_T/n)*((q1-qidsat)/(1+q1-qidsat))+(gm_frac3)*(1.0-(sigma_T/n))*((q2-
     //Drain transconductance
gm = (gm_frac1/(n))*((gm_frac2)*((q1-qidsat)/(1+q1-qidsat)) -
     (gm_frac3)*((q2-qidsat)/(1+q2-qidsat))); //Gate transconductance

//////// Dynamic model ////////////
alpha = (1.0 + qD-qidsat)/(1 + qS-qidsat); //linearity factor

////////////// Total Charges ////////////
QI = -n*PhiT*Cox*We*L*((2.0/3.0)*(((1+alpha+alpha*alpha)/(1+alpha))*(qS+1.0-qidsat)) -
     1.0 ) - (L*ID)/(vsat);
QB = -((n-1)/(n))*QI;
QG = -QB-QI;
QD =
    -n*PhiT*Cox*We*L*((2.0/15.0)*(qS+1.0-qidsat)*((2.0+4.0*alpha+6.0*alpha*alpha+3.0*alpha*alpha*alph
     (1.0/2.0)) -(L*ID)/(2.0*vsat);
QS = QI - QD;
```

```
//////////////// Charges densities ////////////////
QID = qD*(-n*Cox*PhiT);
QIS = qS*(-n*Cox*PhiT);

//////////////// Intrinsic Capacitances /////////////////////
//////// Capacitances coefficients - long channel ////////////
Cgso = (2.0/3.0) * We*L*Cox * ( (1.0 + 2.0*alpha) * (qS-qidsat) )/( ((1.0+alpha)*(1.0+alpha))*(1.0+(qS-qidsat)));
Cgdo = (2.0/3.0)* We*L*Cox * ( ((alpha*alpha)+2.0*alpha)*(qD-qidsat) )/( (1.0+alpha)*(1.0+alpha)*(1+(qD-qidsat) ));
Cgbo = ( (n-1.0)/n )*(We*L*Cox - Cgso - Cgdo);
Cbso = (n-1.0)*Cgso;
Cbdo = (n-1.0)*Cgdo;
Cdso = (-4.0/15.0)*n*We*L*Cox *(((1.0 + 3.0*alpha + alpha*alpha)*(qS-qidsat) )/((1.0+alpha)*(1.0+alpha)*(1.0+alpha)*(1+(qS-qidsat))));
Csdo = (-4.0/15.0)*n*We*L*Cox *(((alpha + 3.0*alpha*alpha + alpha*alpha*alpha)*(qD-qidsat) )/((1.0+alpha)*(1.0+alpha)*(1.0+alpha)*(1+(qD-qidsat))));

/// Capacitances coefficients with DIBL and velocity saturation effects ////
alpha_frac = ((1.0-alpha)*(1.0-alpha))/((1.0+alpha)*(1.0+alpha));
alpha_frac2 = (((3.0*alpha)+7.0)*(1.0-alpha)*(1.0-alpha))/((1.0+alpha)*(1.0+alpha)*(1.0+alpha));
alpha_frac3 = ((3.0+(7.0*alpha))*(1.0-alpha)*(1.0-alpha))/((1.0+alpha)*(1.0+alpha)*(1.0+alpha));
Cgs = Cgso*(1.0-(sigma_T/n)) - Cgdo*(sigma_T/n) +((L*gms)/(3.0*n*vsat))*alpha_frac;
Cgd = Cgdo*(1.0-(sigma_T/n)) - Cgso*(sigma_T/n) -((L*gmd)/(3.0*n*vsat))*alpha_frac;
Cgb = ((n-1.0)/n)*(We*L*Cox-Cgso-Cgdo-((L*gm)/(3.0*vsat))*alpha_frac)+((2.0*sigma_T)/n)*((n-1.0)*We*L*Cox+Cgso+Cgdo);
Cbs = (n-1.0)*Cgs;
Cbd = (n-1.0)*Cgd;
Cds = Cdso*(1.0-(sigma_T/n)) - ((L*gms)/(30.0*vsat))*alpha_frac2;
Csd = Csdo*(1.0-(sigma_T/n)) + ((L*gmd)/(30.0*vsat))*alpha_frac3;
Cm = (Csd - Cds)/n;

//////////////// Extrinsic Capacitances /////////////////////
/////// Overlap Capacitances /////////
Cgs_ov = Cox*We*LD;
Cgd_ov = Cox*We*LD;

///// Junction Capacitances /////////
//DRAIN - BULK
if (VDB>0.0)
begin
 Cjdb   = Cj0*AD*exp(-xd_mj*ln(1.0+VDB/phi_zero));
 Cjdb_sw= Cj0sw*xj*PD*exp(-xd_mjsw*ln(1.0+VDB/phi_zerosw));
end
else
begin
 Cjdb   = Cj0*AD*(1.0 - xd_mj*VDB/phi_zero);
 Cjdb_sw= Cj0sw*xj*PD*(1.0 - xd_mjsw*VDB/phi_zerosw);
end

//SOURCE - BULK
if (VSB>0.0)
begin
 Cjsb   = Cj0*AS*exp(-xd_mj*ln(1.0+VSB/phi_zero));
 Cjsb_sw= Cj0sw*xj*PS*exp(-xd_mjsw*ln(1.0+VSB/phi_zerosw));
end
```

```
else
begin
 Cjsb   = Cj0*AS*(1.0 - xd_mj*VSB/phi_zero);
 Cjsb_sw= Cj0sw*xj*PS*(1.0 - xd_mjsw*VSB/phi_zerosw);
end

qovd = Cgd_ov * VGD;
qovs = Cgs_ov * VGS;

qjd = (Cjdb+Cjdb_sw)*VDB;
qjs = (Cjsb+Cjsb_sw)*VSB;

//////////// Transient currents with charges ///////////////
I(G,B) <+  ddt(QG);
I(D,B) <+  ddt(QD)+VDB*gmin;
I(S,B) <+  ddt(QS)+VSB*gmin;

////// Transient currents with extrinsic capacitances //////
I(G,D) <+ ddt(qovd);
I(G,S) <+ ddt(qovs);
I(D,B) <+ ddt(qjd);
I(S,B) <+ ddt(qjs);

//////////////////// Noise //////////////////////////////
func_thermal = (4 * kboltz * Tin * umob * QI)/(L*L);
ND_thermal = white_noise(func_thermal,"thermal");

func_flicker =
    (q_e*q_e*N_ot*umob/(L*L*n*Cox*ID))*ln((n*Cox*PhiT-QIS)/(n*Cox*PhiT-QID))*ID*ID;

 if (ID < 0) begin
  func_flicker = -func_flicker;
 end

ND_flicker = flicker_noise(func_flicker,1.0,"flicker");
I(D,S) <+ ND_thermal + ND_flicker;

end
endmodule
```

```
//************************************************************************
// *  ACM PMOS model (Verilog-A) - 5PM                                    *
// *   07/2023     V2.0.0                                                 *
//************************************************************************

// **********************************************************************
// * Copyright under the ECL-2.0 license                                  *
// * Universidade Federal de Santa Catarina                               *
// *                                                                      *
// * Current developers:  Deni Germano Alves Neto (Doctoral student, UFSC)  *
// *                      Cristina Missel Adornes (Doctoral student, UFSC)  *
// *                      Gabriel Maranhao        (Doctoral student, UFSC)  *
// *                                                                      *
// * Project Supervisors:  Prof. Carlos Galup-Montoro                     *
// *                       Prof. Marcio Cherem Schneider                  *
// **********************************************************************

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

```
// function of the algorithm 443 to calculate de normalize charge densities
`define algo_443(Z,qn) \
 if(Z < 0.7385) begin \
  numeratorD = Z + (4.0/3.0)*Z*Z; \
  denominatorD = 1.0 + (7.0/3.0)*Z+(5.0/6.0)*Z*Z; \
  WnD = numeratorD/denominatorD; \
 end else begin \
  numeratorD = ln(Z)*ln(Z)+2.0*ln(Z)-3.0; \
  denominatorD = 7.0*ln(Z)*ln(Z) + 58.0*ln(Z) +127.0; \
  WnD = ln(Z) - 24.0*(numeratorD/denominatorD); \
 end \
ZnD = ln(Z) -WnD - ln(WnD); \
TermD = ZnD/(1.0 + WnD); \
numeratorED = 2.0*(1.0+WnD)*(1.0+WnD+(2.0/3.0)*ZnD)-ZnD; \
denominatorED = 2.0*(1.0+WnD)*(1.0+WnD+(2.0/3.0)*ZnD)-2*ZnD; \
EnD = TermD*(numeratorED/denominatorED); \
qn = (WnD*(1+EnD)); \

module pmos_ACM(D, G, S, B);

// pinout definition
inout D,G,S,B;

// electrical nodes definition
electrical D,G,S,B;

//Branches
branch(B,G) BBG;
branch(B,D) BBD;
branch(B,S) BBS;
branch(S,D) BSD;
branch(D,G) BDG;
branch(S,G) BSG;

// Device input variables
(*desc= "Channel width", units = "m", type = "instance"*) parameter real W = 1e-6 from [0:inf];
(*desc= "Channel length", units = "m", type = "instance"*) parameter real L = 1e-6 from [0:inf];
(*desc= "Drain diffusion area", units = "m^2", type = "instance"*)  parameter real AD = 2e-12 from [0:inf];
(*desc= "Source diffusion area", units = "m^2", type = "instance"*) parameter real AS = 2e-12 from [0:inf];
(*desc= "Drain diffusion perimeter", units = "m", type = "instance"*)  parameter real PD = 10e-6 from [0:inf];
(*desc= "Source diffusion perimeter", units = "m", type = "instance"*) parameter real PS = 10e-6 from [0:inf];
(*desc= "Number of transistor in parallel", units = "", type = "instance"*) parameter real m = 1 from [1:inf];

// DC related parameters
(*desc= "Threshold voltage", units = "V", type = "instance"*) parameter real VT0 = 0.5 from [0:inf];
(*desc= "Specific current", units = "A", type = "instance"*) parameter real IS = 5e-6 from [0:inf];
(*desc= "Slope factor", type = "instance"*) parameter real n = 1.3 from [1:3];
(*desc= "Sigma", type = "instance"*) parameter real sigma = 0.03 from [0:inf];
(*desc= "Zeta", type = "instance"*) parameter real zeta = 0.05 from [0:inf];

// Junction capacitance parameters
```

```
(*desc= "Zero bias junction capacitance", units = "F/m^2", type = "instance"*) parameter real Cj0 = 4e-3 from [0:inf];
(*desc= "Side-wall zero bias junction capacitance", units = "F/m^2", type = "instance"*) parameter real Cj0sw = 4e-3 from [0:inf];
(*desc= "Drain(Source) bottom junction capacitance gradient coeficient", units = "", type = "instance"*) parameter real xd_mj = 0.5 from [0:inf];
(*desc= "Side-wall drain(Source) bottom junction capacitance gradient coeficient", units = "", type = "instance"*) parameter real xd_mjsw = 0.5 from [0:inf];
(*desc= "Bulk-junction built-in potential", units = "V", type = "instance"*) parameter real phi_zero = 0.6 from [0:inf];
(*desc= "Side-wall bulk-junction built-in potential", units = "V", type = "instance"*) parameter real phi_zerosw = 0.6 from [0:inf];

// Process related parameters
(*desc= "Junction depth", units = "m", type = "instance"*) parameter real xj = 150e-9 from [0:inf];
(*desc= "Gate overlap", units = "m", type = "instance"*) parameter real LD = 30e-9 from [0:inf];
(*desc= "Thickness of oxide", units = "m", type = "instance"*) parameter real tox = 4e-9 from [0:inf];

//Mismatch related parameters
(*desc= "Area related threshold voltage mismatch", units = "V*m", type = "instance"*) parameter real aVT0 = 0 from [-inf:inf];
(*desc= "Area related gain mismatch", units = "m", type = "instance"*) parameter real aK = 0 from [0:inf];

//Temperature related parameters
(*desc= "VT0 fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphaVT0 = -0.4e-3 from [-1:1];
(*desc= "IS fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphaIS = 1.5 from [0:inf];
(*desc= "Sigma fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphasigma = 0.3e-6 from [0:inf];
(*desc= "Zeta fitting temperature coefficient", units = " ", type = "instance"*) parameter real alphazeta = 0.2e-3 from [0:inf];

//Noise related parameter
(*desc= "Equivalent density of oxide traps", units = "m^-2", type = "instance"*) parameter real N_ot = 1e12 from [0:inf];

// Intern variables
real VP, PhiT, VBD, VBS, VBG, VDG,VSG, VSD, VDS, VBX, VBY;
real qsat, qidsat, q1, q2,qS, qD;
real X, Y, numeratorD, denominatorD, TermD, numeratorED, denominatorED, ZnD, EnD, WnD;
real id, ID, e0, eox, esi, epsi, We;
real umob,vsat, Cox, kboltz, q_e, ni;
real gm_frac1, gm_frac2, gm_frac3, gms,gmd, gm;
real alpha, alpha_frac, alpha_frac2, alpha_frac3;
real QI, QD, QS, QG, QB, QID, QIS;
real Cgso, Cgdo, Cgbo, Cbso, Cbdo, Cdso, Csdo;
real Cgd, Cgs, Cgb, Cbs, Cbd, Cds, Csd, Cm;
real Cgs_ov, Cgd_ov, Cjdb, Cjsb, Cjdb_sw, Cjsb_sw, Cjsw;
real qovd, qovs, qjd, qjs;
real VT0_T, IS_T, sigma_T, zeta_T, Tin, Tref;
real func_thermal, func_flicker, PSD_thermal, PSD_flicker;
real gmin;

analog begin
```

```
//Constants
e0 = `P_EPS0; // Vacuum permittivity
eox = 3.97*e0; // Silicon dioxide permittivity
esi = 11.9*e0; // Silicon permittivity
Tref = 300.15; // Reference temperature
q_e = `P_Q; // Elementary charge
kboltz = `P_K; // Boltzmann constant
epsi = 0.1; // Constant used for device symmetry
gmin = $simparam("gmin",0);// gmin is placed in parallel with drain-bulk and source-bulk
    current sources to increase the numerical convergence
We = m*W; // Equivalent W of the parallel devices

//Temperature variation
PhiT = $vt($temperature); // Thermal voltage
Tin = $temperature; //Simulator temperature
VT0_T = VT0 + alphaVT0*(Tin-Tref) + aVT0/sqrt(We*L); // Temperature variation and
    mismatch on VT0
IS_T = IS*(1+ ((2-alphaIS)/Tin)*(Tin-Tref)) + (IS*aK)/sqrt(We*L); // Temperature
    variation and mismatch on IS
sigma_T = sigma+ alphasigma*(Tin-Tref); // Temperature variation on sigma
zeta_T = zeta + alphazeta*(Tin-Tref); // Temperature variation on zeta

//Control voltages from branches
VBG = V(BBG);
VBD = V(BBD);
VBS = V(BBS);
VSD = V(BSD);
VDG = V(BDG);
VSG = V(BSG);
VDS = -VSD;
VBX = min(VBS,VBD);
VBY = max(VSD,VDS);

// Intern process parameters
Cox = eox/tox; // [F/m^2]
umob = (IS_T*L)/(Cox*n*PhiT*PhiT*We*0.5); //mobility [m^2/V.s]
vsat = (umob*PhiT)/(L*zeta_T); //[m^2/s]

//Pinch-off voltage
VP = (VBG - abs(VT0_T) + sigma_T*(VBD+VBS) )/n;

////q1 from algorithm 443 and UCCM_LC
X = exp(((VP - VBX)/PhiT)+1.0);
`algo_443(X,q1)

//Calculating qdsat
qsat = q1 + 1.0 + (1.0/zeta_T) - sqrt(((1+(1/zeta_T))*(1+(1/zeta_T)))+((2*q1)/zeta_T));

////q2 from algorithm 443 and UCCM
Y = (((q1-qsat))*exp(-VBY/PhiT)*exp(q1-qsat));
`algo_443(Y,q2);
q2 = q2 + qsat;

//Normalized densities charges for a symmetric model
 if (V(S,D) >= 0) begin

  qS = q1;
  qD = q2;
 end
 else begin
```

```
  qD = q1;
  qS = q2;
 end

///////////// Calculating Drain Current ID //////////////////////
id = ((qD+qS+2.0)/(1.0+ sqrt((zeta_T*(qS-qD))*zeta*((qS-qD))+(epsi*epsi))))*(qS - qD);
ID = m*IS_T*id;
I(S,D) <+ ID;

///////////// Transconductances ////////////////////
qidsat =(zeta_T/2.0)*id;

//gm_frac1 to gm_frac3 are intermediate fractions of the transconductances
gm_frac1 = (2.0*IS_T)/(PhiT*(1.0+zeta_T*(q1-q2))*(1.0+zeta_T*(q1-q2)));
gm_frac2 = (1.0+q1)+(zeta_T/2.0)*(((q1-q2)*(q1-q2)));
gm_frac3 = (1.0+q2)-(zeta_T/2.0)*(((q1-q2)*(q1-q2)));
gms =
    gm_frac1*((gm_frac2)*(1.0-(sigma_T/n))*((q1-qidsat)/(1+q1-qidsat))+((gm_frac3)*(sigma_T*n)*((q2-q
    //Source transconductance
gmd =
    gm_frac1*((gm_frac2)*(sigma_T/n)*((q1-qidsat)/(1+q1-qidsat))+(gm_frac3)*(1.0-(sigma_T/n))*((q2-q
    //Drain transconductance
gm = (gm_frac1/(n))*((gm_frac2)*((q1-qidsat)/(1+q1-qidsat)) -
    (gm_frac3)*((q2-qidsat)/(1+q2-qidsat))); //Gate transconductance

//////// Dynamic model ///////////
alpha = (1.0 + q2-qidsat)/(1 + q1-qidsat); //linearity factor

//////////////Charges///////////
QI = -n*PhiT*Cox*We*L*((2.0/3.0)*(((1+alpha+alpha*alpha)/(1+alpha))*(qS+1.0-qidsat)) -
    1.0 ) - (L*ID)/(vsat);
QB = -((n-1)/(n))*QI;
QG = -QB-QI;
QD =
    -n*PhiT*Cox*We*L*((2.0/15.0)*(qS+1.0-qidsat)*((2.0+4.0*alpha+6.0*alpha*alpha+3.0*alpha*alpha*alph
    (1.0/2.0)) -(L*ID)/(2.0*vsat);
QS = QI - QD;

////////////// Charges densities ////////////////
QID = qD*(-n*Cox*PhiT);
QIS = qS*(-n*Cox*PhiT);

//////////////// Intrinsic Capacitances //////////////////////
//////// Capacitances coefficients - long channel ///////////
Cgso = (2.0/3.0) * We*L*Cox * ( (1.0 + 2.0*alpha) * (qS-qidsat) )/(
    ((1.0+alpha)*(1.0+alpha))*(1.0+(qS-qidsat)));
Cgdo = (2.0/3.0)* We*L*Cox * ( ((alpha*alpha)+2.0*alpha)*(qD-qidsat) )/(
    (1.0+alpha)*(1.0+alpha)*(1+(qD-qidsat) ));
Cgbo = ( (n-1.0)/n )*(We*L*Cox - Cgso - Cgdo);
Cbso = (n-1.0)*Cgso;
Cbdo = (n-1.0)*Cgdo;
Cdso = (-4.0/15.0)*n*We*L*Cox *(((1.0 + 3.0*alpha + alpha*alpha)*(qS-qidsat)
    )/((1.0+alpha)*(1.0+alpha)*(1.0+alpha)*(1+(qS-qidsat))));
Csdo = (-4.0/15.0)*n*We*L*Cox *(((alpha + 3.0*alpha*alpha +
    alpha*alpha*alpha)*(qD-qidsat)
    )/((1.0+alpha)*(1.0+alpha)*(1.0+alpha)*(1+(qD-qidsat))));

/// Capacitances coefficients with DIBL and velocity saturation effects ////
alpha_frac = ((1.0-alpha)*(1.0-alpha))/((1.0+alpha)*(1.0+alpha));
alpha_frac2 =
```

```
(((3.0*alpha)+7.0)*(1.0-alpha)*(1.0-alpha))/((1.0+alpha)*(1.0+alpha)*(1.0+alpha));
alpha_frac3 = ((3.0+(7.0*alpha))*(1.0-alpha)*(1.0-alpha))/((1.0+alpha)*(1.0+alpha)*(1.0+alpha));
Cgs = Cgso*(1.0-(sigma_T/n)) - Cgdo*(sigma_T/n) +((L*gms)/(3.0*n*vsat))*alpha_frac + Cgdo*W;
Cgd = Cgdo*(1.0-(sigma_T/n)) - Cgso*(sigma_T/n) -((L*gmd)/(3.0*n*vsat))*alpha_frac + Cgdo*W;
Cgb = ((n-1.0)/n)*(We*L*Cox-Cgso-Cgdo-((L*gm)/(3.0*vsat))*alpha_frac)+((2.0*sigma_T)/n)*((n-1.0)*We*L*Cox+Cgso+Cgdo);
Cbs = (n-1.0)*Cgs;
Cbd = (n-1.0)*Cgd;
Cds = Cdso*(1.0-(sigma_T/n)) - ((L*gms)/(30.0*vsat))*alpha_frac2;
Csd = Csdo*(1.0-(sigma_T/n)) + ((L*gmd)/(30.0*vsat))*alpha_frac3;
Cm = (Csd - Cds)/n;

//////////////// Extrinsic Capacitances /////////////////////
/////// Overlap Capacitances /////////
Cgs_ov = Cox*We*LD;
Cgd_ov = Cox*We*LD;

///// Junction Capacitances /////////
//DRAIN - BULK
if (VBD>0.0)
begin
 Cjdb   = Cj0*AD*exp(-xd_mj*ln(1.0+VBD/phi_zero));
 Cjdb_sw = Cj0sw*xj*PD*exp(-xd_mjsw*ln(1.0+VBD/phi_zerosw));
end
else
begin
 Cjdb   = Cj0*AD*(1.0 - xd_mj*VBD/phi_zero);
 Cjdb_sw = Cj0sw*xj*PD*(1.0 - xd_mjsw*VBD/phi_zerosw);
end

//SOURCE - BULK
if (VBS>0.0)
begin
 Cjsb   = Cj0*AS*exp(-xd_mj*ln(1.0+VBS/phi_zero));
 Cjsb_sw = Cj0sw*xj*PS*exp(-xd_mjsw*ln(1.0+VBS/phi_zerosw));
end
else
begin
 Cjsb   = Cj0*AS*(1.0 - xd_mj*VBS/phi_zero);
 Cjsb_sw = Cj0sw*xj*PS*(1.0 - xd_mjsw*VBS/phi_zerosw);
end

qovd = Cgd_ov * VDG;
qovs = Cgs_ov * VSG;

qjd = (Cjdb+Cjdb_sw)*VBD;
qjs = (Cjsb+Cjsb_sw)*VBS;

////// Transient currents with intrinsic capacitances //////
I(B,G) <+  ddt(QG);
I(B,D) <+  ddt(QD)+VBD*gmin;
I(B,S) <+  ddt(QS)+VBS*gmin;

////// Transient currents with extrinsic capacitances //////
I(D,G) <+ ddt(qovd);
I(S,G) <+ ddt(qovs);
I(B,D) <+ ddt(qjd);
```

```
I(B,S) <+ ddt(qjs);

//////////////////// Noise /////////////////////////////
func_thermal = (4 * kboltz * Tin * umob * QI)/(L*L);
PSD_thermal = white_noise(func_thermal,"thermal");

func_flicker =
    (q_e*q_e*N_ot*umob/(L*L*n*Cox*ID))*ln((n*Cox*PhiT-QIS)/(n*Cox*PhiT-QID))*ID*ID;

 if (ID < 0) begin
  func_flicker = -func_flicker;
 end

PSD_flicker = flicker_noise(func_flicker,1.0,"flicker");
//I(S,D) <+ PSD_thermal + PSD_flicker;

end
endmodule
```

APPENDIX B – CONTINUITY BETWEEN THE TRIODE AND THE SATURATION REGIONS

A most traditional method to obtain the continuity between the triode and the saturation regions is introduced in [7], which consists in defining a drain-to-source saturation voltage V_{DSsat} (55) and an effective drain-to-source voltage $V_{DS}^{'}$ using a interpolation function (56).

$$\frac{V_{DSsat}}{\phi_t} = q_S - q_{sat} - 1 + \ln\left(\frac{q_S}{q_{sat}}\right) \tag{55}$$

$$V_{DS}^{'} = \frac{V_{DS}}{\sqrt[4]{1 + (\frac{V_{DS}}{V_{DSsat}})^4}} \tag{56}$$

With the drain effective voltage calculated (57), one can determine the effective normalized drain charge density $q_D^{'}$ (58) and finally the drain current in (59).

$$V_D^{'} = V_{DS}^{'} + V_S \tag{57}$$

$$\frac{V_P - V_D^{'}}{\phi_t} = q_D^{'} - 1 + \ln(q_D^{'}) \tag{58}$$

$$I_D = I_S \frac{(q_S + q_D^{'} + 2)}{1 + \zeta\,|q_S - q_D^{'}|}(q_S - q_D^{'}) \tag{59}$$

Fig. 6 compares the single-piece model of saturation proposed herein on the 5PM model with the interpolation function from [7]. Note that both methods have very similar results. However, the single-piece model of saturation from (1) and (**??**) does not rely on a non-physical interpolation function and provide a faster simulation due to the lesser number of equations.

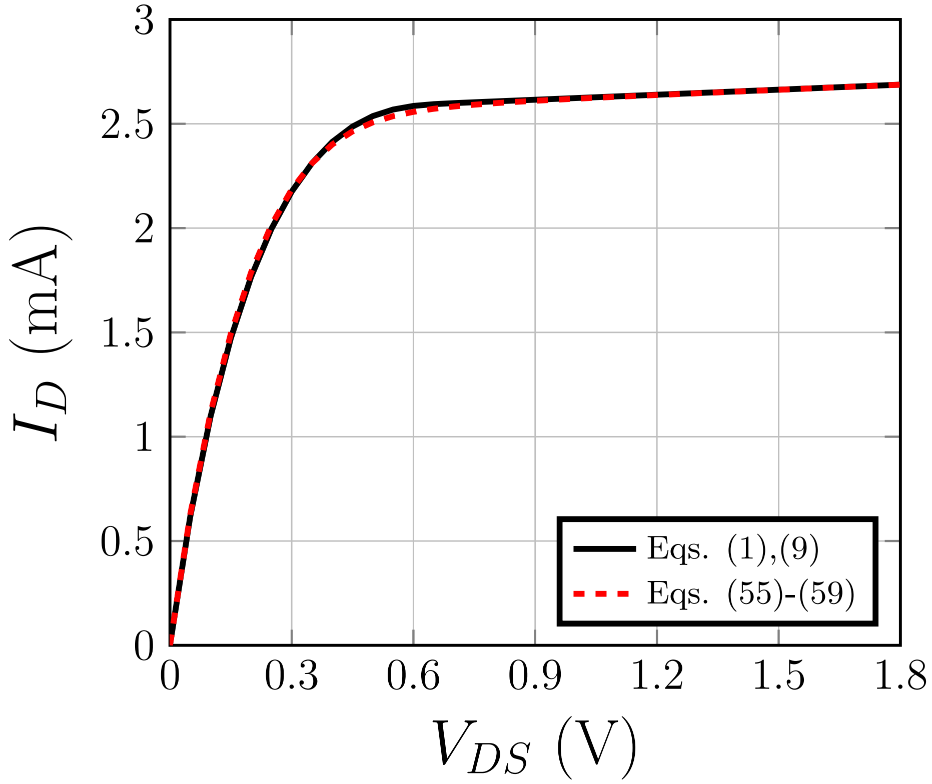

Figure 6 – DC characteristics $I_D \times V_{DS}$ with $V_{GB} = 1.8$ V.