Systematic Design of Analog CMOS Circuits with LUTs using open-source tools and PDKs

Part I (a)

* Most slides are borrowed or adapted from B. Murmann

Analog Flow

Front-end

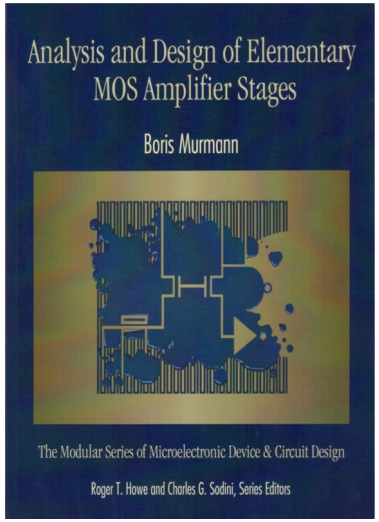
Back-end

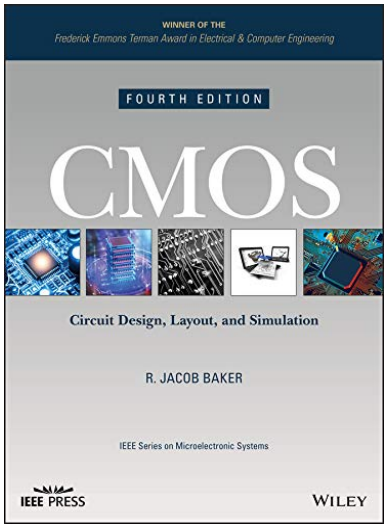
Design	Schematic	Simulation	Layout	DRC & LVS	PEX
 pygmid	XSCHEM	NGSPICE	Magic VLSI Current distribution versi	Magic VLSI Current distribution versi Netgen	Magic VLSI Current distribution versi

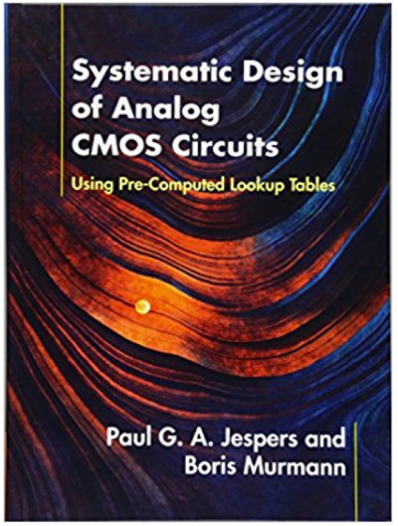
Tools ecosystem: IIC-OSIC-TOOLS JKU JOHANNES KEPLER UNIVERSITÄT LINZ

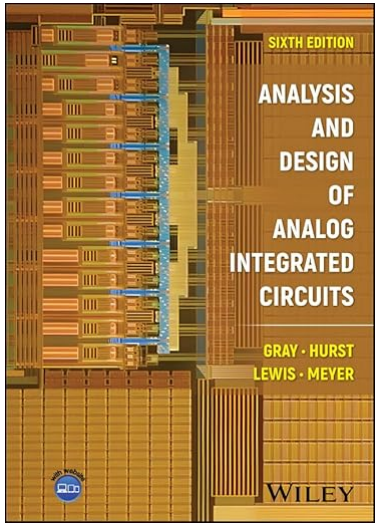
PDK: GLOBAL FOUNDRIES gf180mcuD

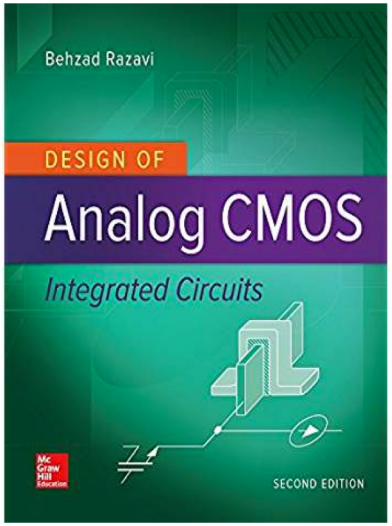
References

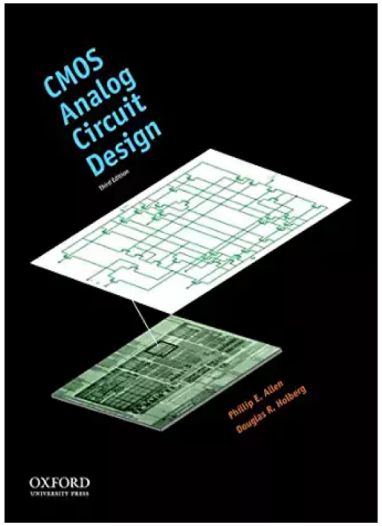

Analysis and Design of Elementary MOS Amplifier Stages
Boris Murmann
The Modular Series of Microelectronic Device & Circuit Design
Roger T. Howe and Charles G. Sodini, Series Editors


WINNER OF THE
Frederick Emmons Terman Award in Electrical & Computer Engineering
FOURTH EDITION
CMOS
Circuit Design, Layout, and Simulation
R. JACOB BAKER
IEEE Series on Microelectronic Systems
IEEE PRESS
WILEY


Systematic Design of Analog CMOS Circuits
Using Pre-Computed Lookup Tables
Paul G. A. Jespers and Boris Murmann


SIXTH EDITION
ANALYSIS AND DESIGN OF ANALOG INTEGRATED CIRCUITS
GRAY · HURST
LEWIS · MEYER
WILEY


Behzad Razavi
DESIGN OF
Analog CMOS
Integrated Circuits
SECOND EDITION


CMOS Analog Circuit Design
OXFORD

Some useful Links

- https://github.com/bmurmann/Book-on-MOS-stages OR https://bmurmann.github.io/COCOA/
- https://github.com/bmurmann/Book-on-gm-ID-design
- https://cmosedu.com/
- https://iic-jku.github.io/analog-circuit-design/

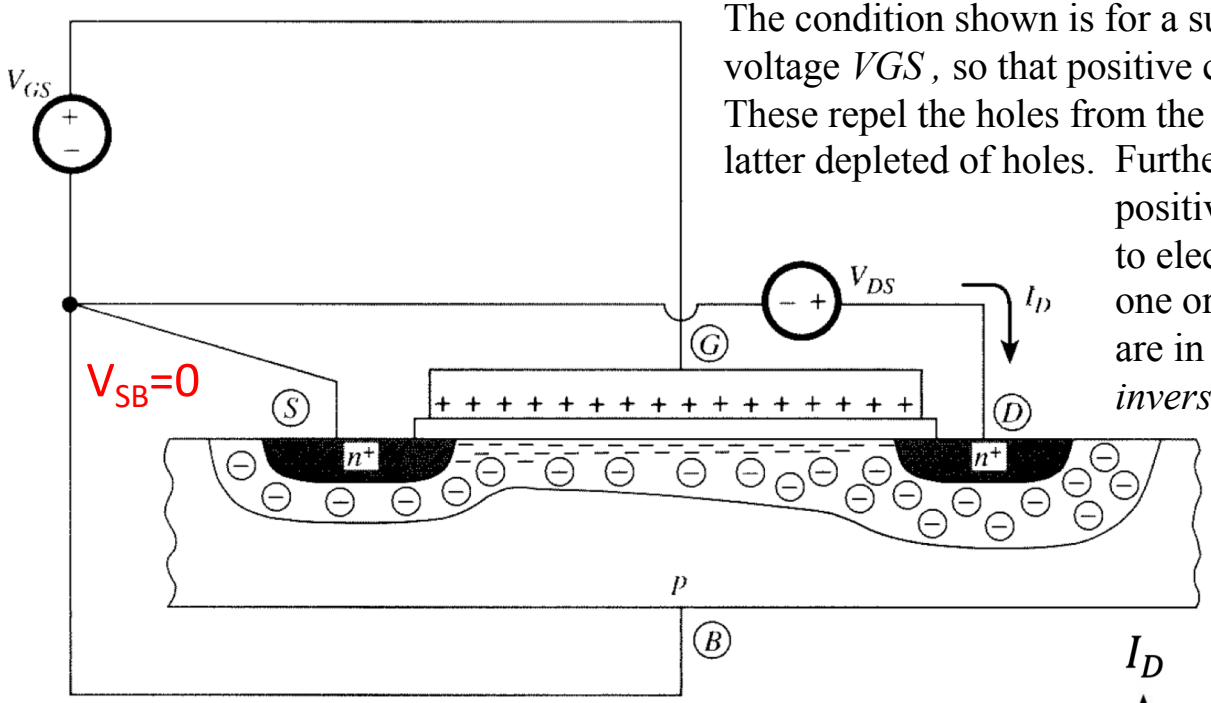
- https://github.com/bmurmann?tab=repositories
- https://github.com/bmurmann/Chipathon2024
- https://github.com/bmurmann/Chipathon2025

This presentation:
https://github.com/claudiotalarico/workshop-analog-2025
https://github.com/claudiotalarico/workshop-analog-2025/tree/main/gf180-2025

Outline

- MOSFET Modeling Challenges
- Pre-computed Lookup Tables:
  Technology Characterization and Data Generation
- Accessing the Data using pygmid
- Setting up the open-source tools ecosystem
- MOSFET FOMs
- g_m/I_D as a knob to control the design tradeoffs
- Differential Pair sizing example (basic)
- 5T-OTA Design Flow example

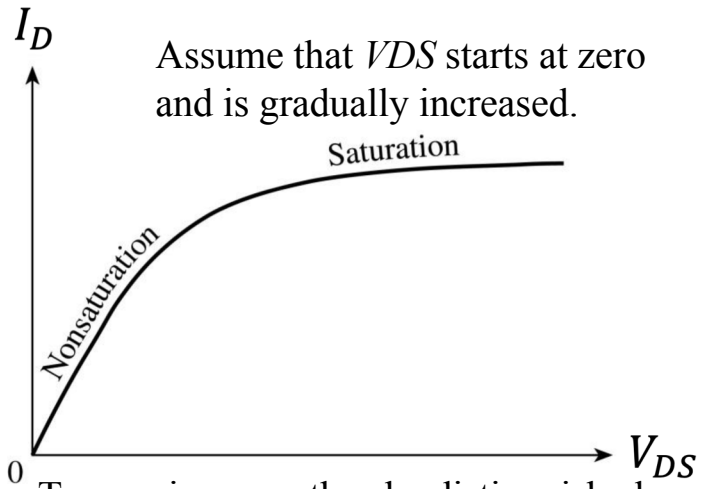
A qualitative description of the nMOST



The condition shown is for a sufficiently positive gate-source voltage *VGS* , so that positive charges are placed on the gate. These repel the holes from the silicon surface, leaving the latter depleted of holes. Further, *VGS* is assumed to be sufficiently positive to even make the surface attractive to electrons; the latter can easily enter through one or both of the two n+ regions, where they are in large supply. This situation is called *inversion*

The layer of electrons at the surface is called an *inversion layer.*

The potential difference *VDS* between the drain and the source is positive and appears across the inversion layer. It causes electron movement: electrons enter from the source, pass through the channel, and are drained by the drain.



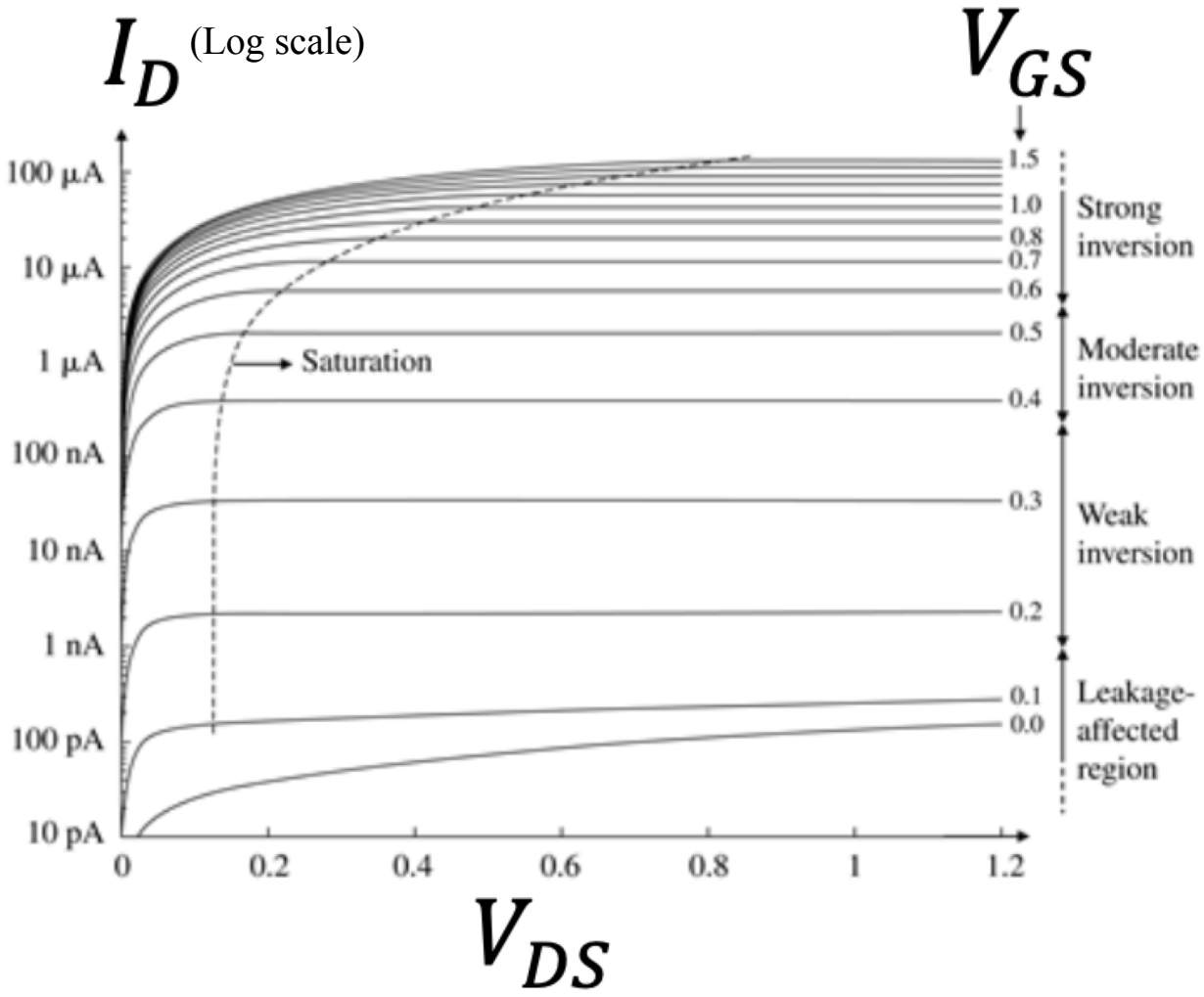
Assume that *VDS* starts at zero and is gradually increased.

Two regions can thus be distinguished, marked *non-saturation* and *saturation.*

source: Tsividis

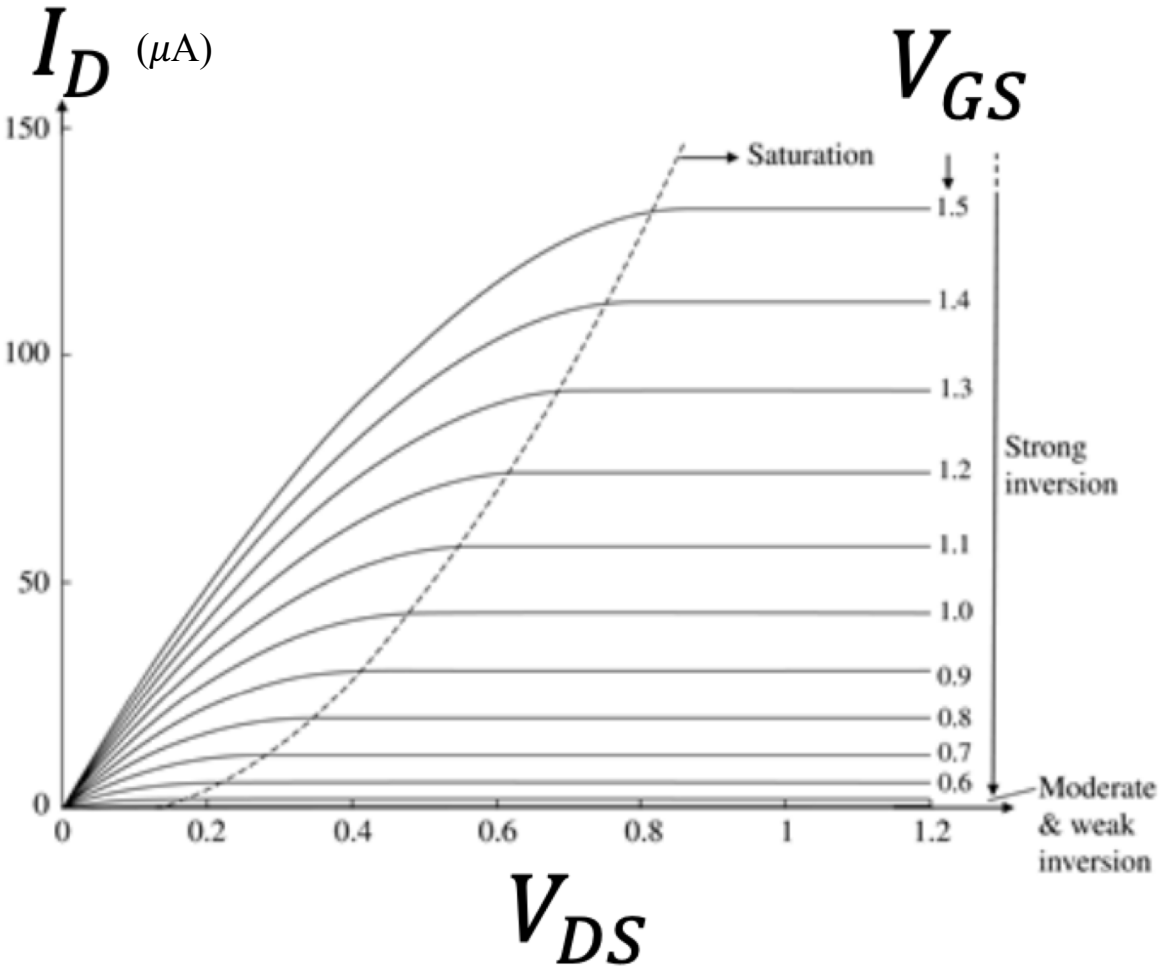
MOST characteristics

source: Tsividis



MOST characteristics

source: Tsividis

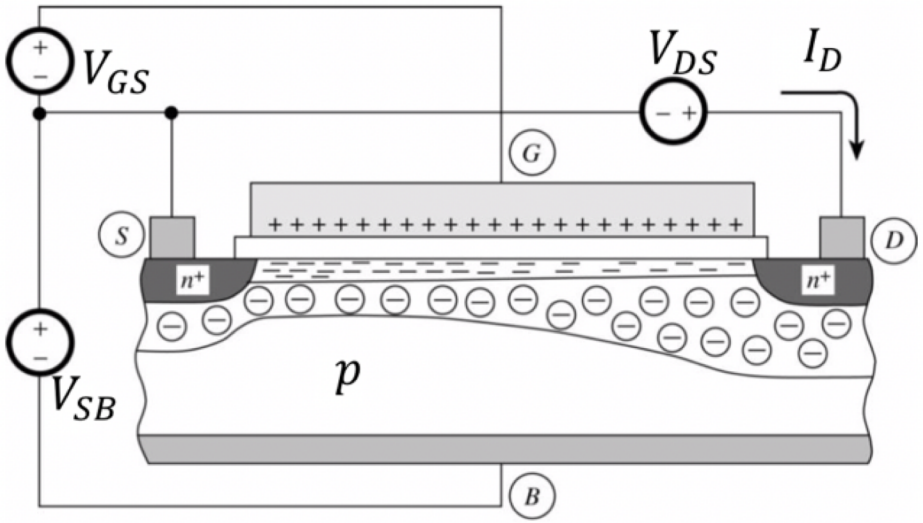


Typical textbook approach

source: Tsividis

Assumptions:

- Operation is DC.
- Horizontal electric field component much smaller than vertical one; ("Gradual channel approximation".)
- Channel length much larger than depletion region widths around source and drain; neglect situation around those regions for now.



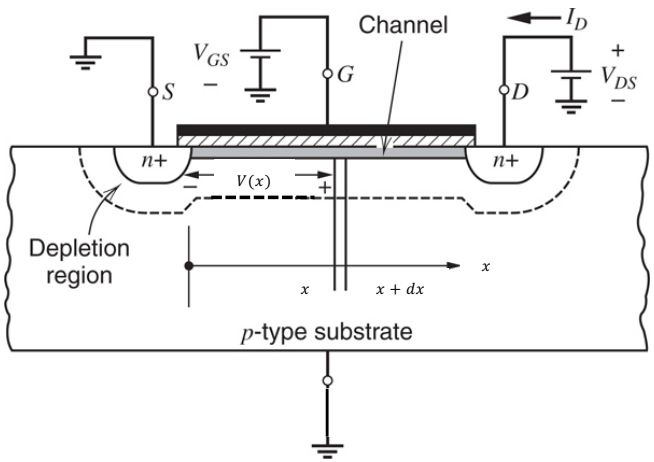
Objective:

Find I_D

(we need Q_I)

Textbook MOSFET operation

source: Pierret



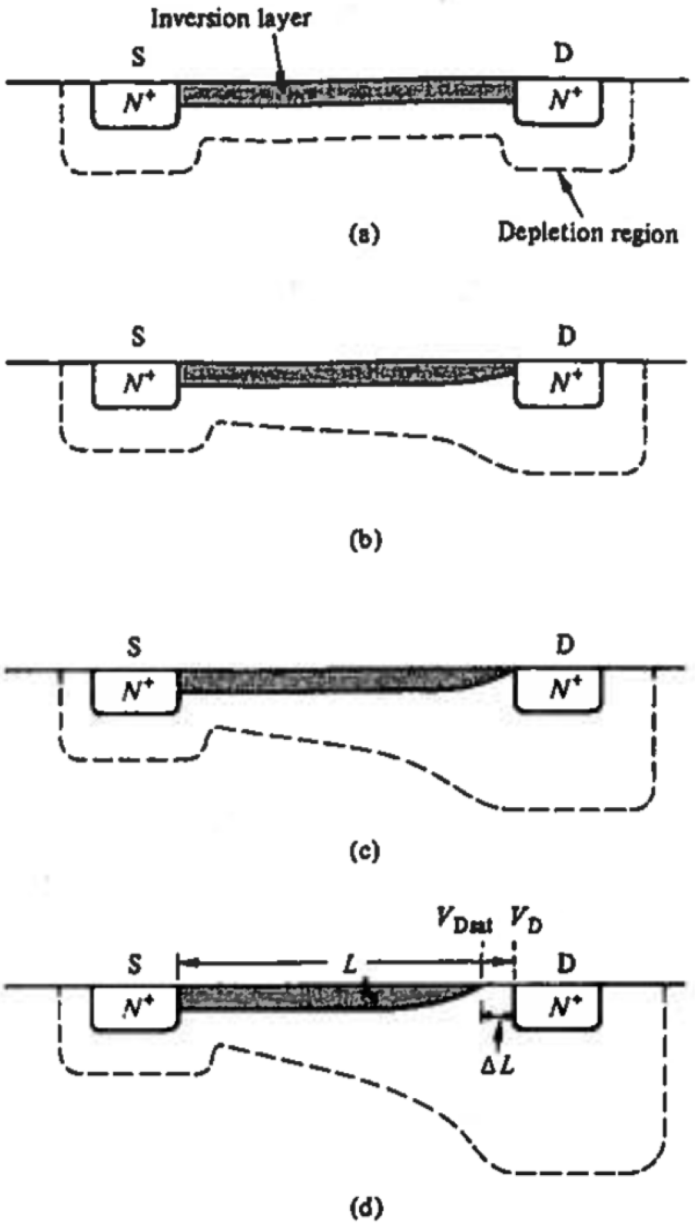
$$Q_I' = -C_{ox}' \cdot [V_{GS} - V(x) - V_t]$$

$$I_D = \frac{dQ_I}{dt} = Q_I' \cdot \frac{dx}{dt} \cdot W =$$

$$= Q_I' \cdot v_{drift}(x) \cdot W$$

$$v_{drift} = \mu \cdot \mathcal{E}_{horizontal}(x)$$

$$\mathcal{E}_{horizontal}(x) = -\frac{dV(x)}{dx}$$

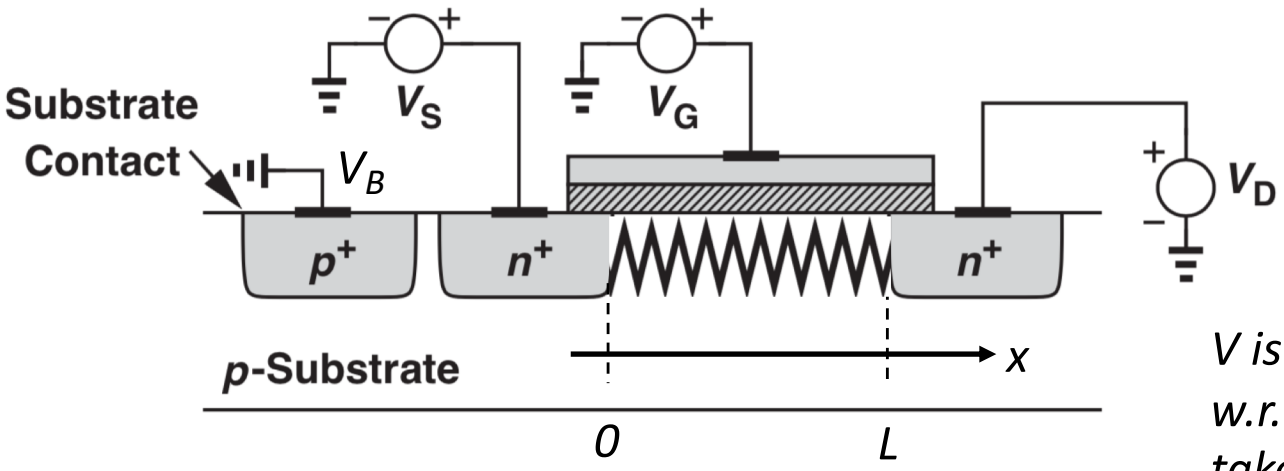


I_D doesn't depend on V_{DS} any longer. The charge is swept across by the strong field of the reverse p-n junction

Figure 17.2 Visualization of various phases of $V_G > V_T$ MOSFET operation. (a) $V_D = 0$; (b) channel (inversion layer) narrowing under moderate V_D biasing; (c) pinch-off; and (d) postpinch-off ($V_D > V_{Dsat}$) operation. (Note that the inversion layer widths, depletion widths, etc. are not drawn to scale.)

Typical Textbook Approach

source: Razavi



V is the voltage drop along x w.r.t. the voltage at the source taken as reference (V_S)

$$Q_I'(x) = -C_{ox}'(V_{GS} - V_t - V(x))$$

$$I_{DS} = \frac{dQ_I}{dt} \qquad dt = \frac{dx}{v_{drift}} \qquad v_{drift} = \mu\,\mathcal{E}_{horizontal} \qquad \mathcal{E}_{horizontal} = -\frac{dV}{dx}$$

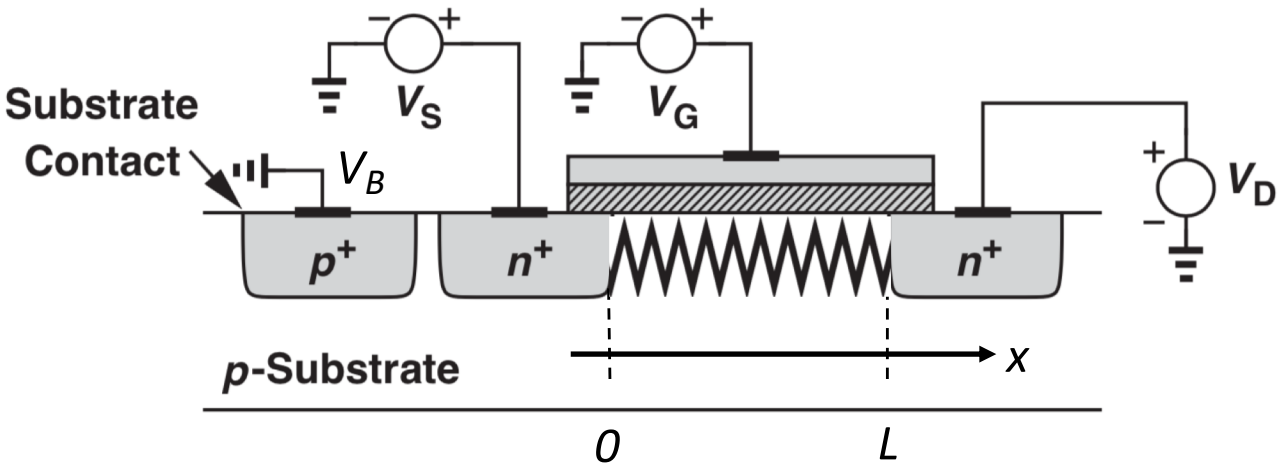
$$dQ_I = Q_I' W dx$$

$$\Rightarrow I_{DS} = \frac{dQ_I}{dx} v = \frac{C_{ox}'(V_{GS} - V_t - V)W\cancel{dx}}{\cancel{dx}} \mu \frac{dV}{dx}$$

$$\Rightarrow I_{DS}\,dx = \mu W C_{ox}'(V_{GS} - V_t - V)\mathrm{dV}$$

Typical Textbook Approach

source: Razavi



$$\int_0^L I_{DS}dx = \mu W C'_{ox} \int_{V(0)=0}^{V(L)=V_{DS}} (V_{GS} - V_t - V)\mathrm{dV}$$

$$\Longrightarrow I_{DS} = \mu \frac{W}{L} C'_{ox} \left[(V_{GS} - V_t)V_{DS} - \frac{V_{DS}^2}{2}\right]$$

… etc.

V is the voltage drop along x w.r.t. the voltage at the source taken as reference (V_S)
V(x=0) = $V_S - V_S = 0$
V(x=L) = $V_D - V_S = V_{DS}$

I_D vs. V_{DS}: Typical textbook approach

source: Carusone

$$I_D = \mu C'_{ox} \frac{W}{L}\left[(V_{GS} - V_t)V_{DS} - \frac{V_{DS}^2}{2}\right]$$

$$I_D = \frac{1}{2}\mu C'_{ox} \frac{W}{L}(V_{GS} - V_t)^2$$

Notation:

$V_{eff} = V_{OV} = V_{GS} - V_t$

Cut-off or subthreshold:

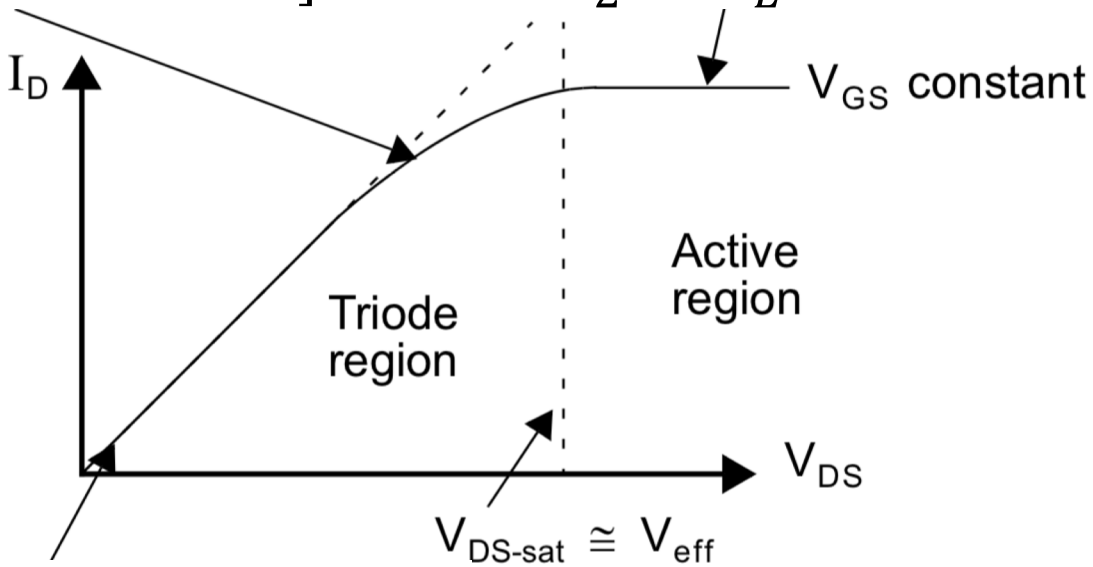
$V_{GS} \leq V_t$

Triode:

$V_{GS} > V_t$ and $V_{DS} < V_{GS} - V_t$

Saturation:

$V_{GS} > V_t$ and $V_{DS} \geq V_{GS} - V_t$



$$I_D \approx \mu C'_{ox} \frac{W}{L}[(V_{GS} - V_t)V_{DS}]$$

The true value of $V_{DS\text{-}sat}$ is slightly lower than V_{eff} (we have body effect also at the drain)

Typical Textbook approach

source: Razavi

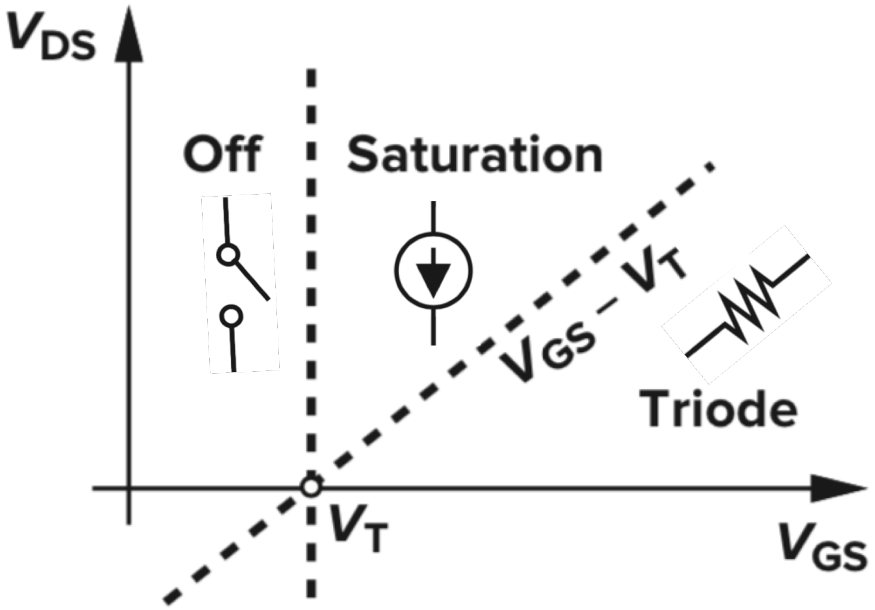


Figure 2.18 V_{DS}-V_{GS} plane showing regions of operation.

There is Body Effect both at source and drain !

source: Carusone

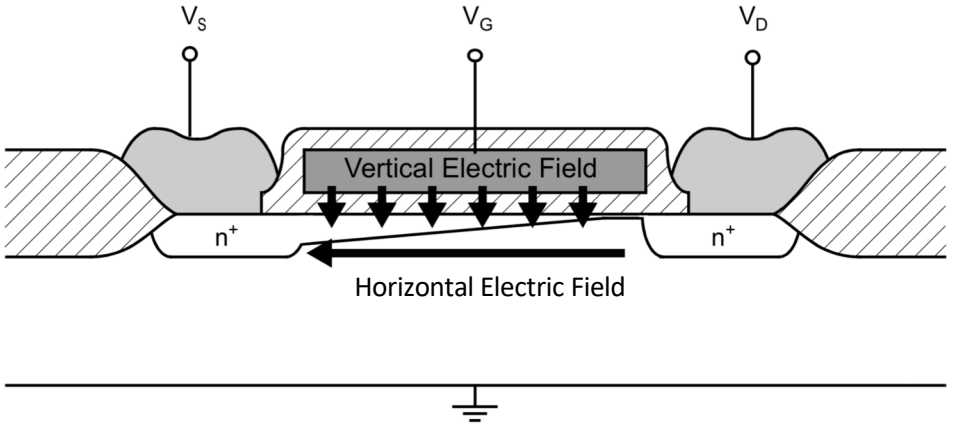
body effect only w.r.t. bulk and source

$$V_t = V_{t0} + \gamma\left(\sqrt{V_{SB} + |2\phi_F|} - \sqrt{2|\phi_F|}\right)$$

NOTE: it is more accurate to use PHI rather than $|2\phi_F|$ (see Tsividis)

$$\gamma = \frac{\sqrt{2qN_A\epsilon_s\epsilon_0}}{C'_{ox}}$$

$$\phi_F = \frac{kT}{q} ln\left(\frac{N_A}{n_i}\right)$$



V_{t0} voltage we need to apply to get a level of inversion for which $n_{surface}$ is about N_A

We are used to think of the inversion charge as being formed because of the vertical electric field resulting from applying $V_{GB} = V_{GS} - V_{SB}$ and tend to neglect the effect of the horizontal field (gradual channel approximation), hence we end up talking about body effect only w.r.t. bulk and source, **but there is also body effect w.r.t. bulk and drain**

What's wrong with the square law model ?

For $V_{DS} > V_{GS} - V_t$ we should use $L^* = L - \Delta L$ (CLM)

$$I_D = \frac{1}{2}\mu C'_{ox}\frac{W}{L^*}(V_{GS} - V_t)^2 \cong \frac{1}{2}\boldsymbol{\mu} C'_{ox}\frac{W}{L}(V_{GS} - \boldsymbol{V_t})^2(1 + \boldsymbol{\lambda} V_{DS})$$

$$g_m = \frac{dI_D}{dV_{GS}} = \mu C'_{ox}\frac{W}{L}(V_{GS} - V_t)(1 + \lambda V_{DS}) = \frac{2I_D}{V_{GS} - V_t}$$

$$g_{ds} = \frac{dI_D}{dV_{DS}} = \frac{\lambda I_D}{1 + \lambda V_{DS}} \cong \lambda I_D$$

- $\boldsymbol{\lambda}$ is not constant (it depends on L and the bias conditions)
- $\boldsymbol{\mu}$ and $\boldsymbol{V_t}$ are not constant. They depend on L and the bias conditions (besides temperature)
- (Beginner) Question: what are the values of $\boldsymbol{\mu C_{ox}}$, $\boldsymbol{\lambda}$ and $\boldsymbol{V_t}$ for our nanometer CMOS process ?
- Answer: **They don't exists !**

What are $\mu C'_{ox}$ and λ and V_t for our technology ?

source: Murmann

```
* 0.18um CMOS models (nominal process)
.MODEL nmos nmos (
+acm     = 3              hdif    = 0.32e-6        LEVEL   = 49
+ CAPOP = 39
+VERSION = 3.1            TNOM    = 27             TOX     = 4.1E-9
+XJ      = 1E-7           NCH     = 2.3549E17      VTH0    = 0.3618397
+K1      = 0.5916053      K2      = 3.225139E-3    K3      = 1E-3
+K3B     = 2.3938862      W0      = 1E-7           NLX     = 1.776268E-7
+DVT0W   = 0              DVT1W   = 0              DVT2W   = 0
+DVT0    = 1.3127368      DVT1    = 0.3876801      DVT2    = 0.0238708
+U0      = 256.74093      UA      = -1.585658E-9   UB      = 2.528203E-18
+UC      = 5.182125E-11   VSAT    = 1.003268E5     A0      = 1.981392
+AGS     = 0.4347252      B0      = 4.989266E-7    B1      = 5E-6
+KETA    = -9.888408E-3   A1      = 6.164533E-4    A2      = 0.9388917
+PRWG    = 0.5            PRWB    = -0.2
+WR      = 1              WINT    = 0              LINT    = 1.617316E-8
+XL      = 0              XW      = -1E-8          DWG     = -5.383413E-9
+DWB     = 9.111767E-9    VOFF    = -0.0854824     NFACTOR = 2.2420572
+CIT     = 0              CDSC    = 2.4E-4         CDSCD   = 0
+CDSCB   = 0              ETA0    = 2.981159E-3    ETAB    = 9.289544E-6
+DSUB    = 0.0159753      PCLM    = 0.7245546      PDIBLC1 = 0.1568183
+PDIBLC2 = 2.543351E-3    PDIBLCB = -0.1           DROUT   = 0.7445011
+PSCBE1  = 8E10           PSCBE2  = 1.876443E-9    PVAG    = 7.200284E-3
+DELTA   = 0.01           MOBMOD  = 1
+PRT     = 0              UTE     = -1.5           KT1     = -0.11
+KT1L    = 0              KT2     = 0.022          UA1     = 4.31E-9
+UB1     = -7.61E-18      UC1     = -5.6E-11       AT      = 3.3E4
+WL      = 0              WLN     = 1              WW      = 0
+WWN     = 1              WWL     = 0              LL      = 0
+LLN     = 1              LW      = 0              LWN     = 1
+LWL     = 0              CAPMOD  = 2              XPART   = 1
+CGDO    = 4.91E-10       CGSO    = 4.91E-10       CGBO    = 1E-12
+CJ      = 9.652028E-4    PB      = 0.8            MJ      = 0.3836899
+CJSW    = 2.326465E-10   PBSW    = 0.8            MJSW    = 0.1253131
+CF      = 0              PVTH0   = -7.714081E-4   PRDSW   = -2.5827257
+PK2     = 9.619963E-4    WKETA   = -1.060423E-4   LKETA   = -5.373522E-3
+PU0     = 4.5760891      PUA     = 1.469028E-14   PUB     = 1.783193E-23
+PVSAT   = 1.19774E3      PETA0   = 9.968409E-5    PKETA   = -2.51194E-3
+nlev    = 3              kf      = 0.5e-25)
```

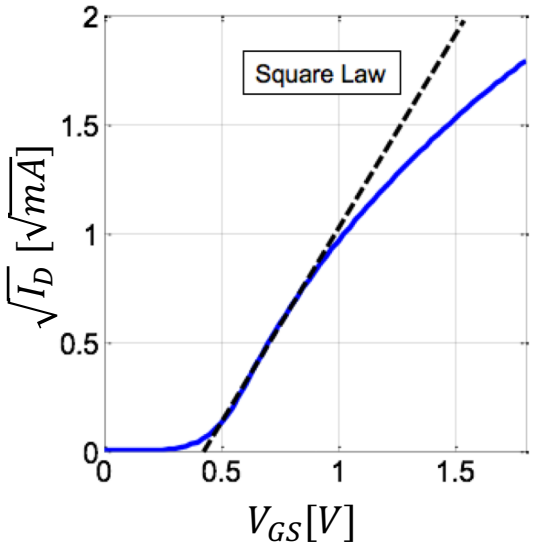
This is a 110-parameter BSIM3v3 SPICE model: $\mu C'_{ox} \triangleq$ KP and lambda are nowhere to be found

What's wrong with the square-law model ? Simulations for generic 0.18μm technology

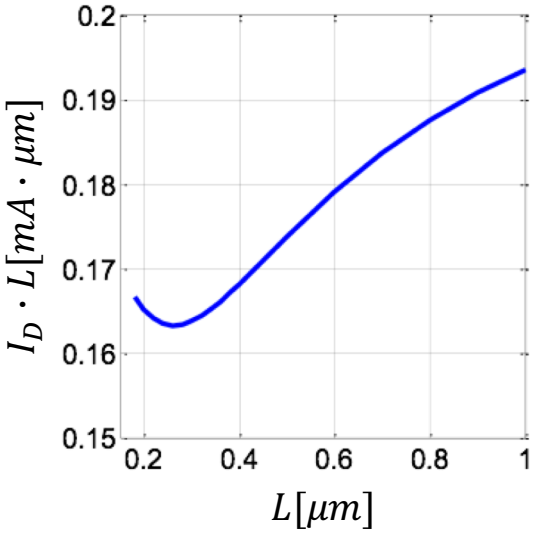
source: Murmann

- The transistor does not abruptly turn off at V_t
- The current is not perfectly quadratic with V_{OV} (= V_{GS}–V_t)
- The current does not scale perfectly with 1/L
- The threshold voltage V_t of the device changes with L

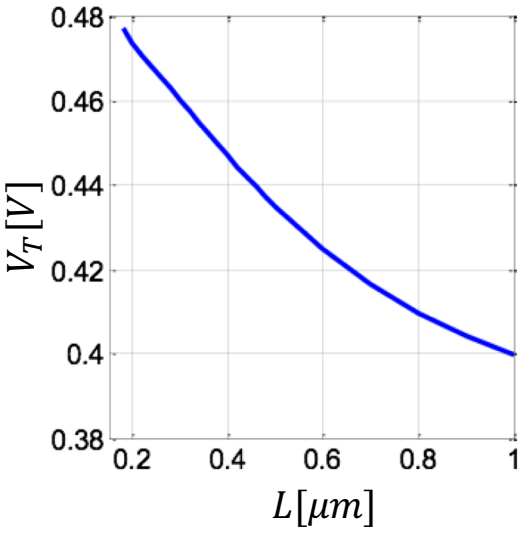
nMOST with W/L = 5μm/0.18μm



@V_{DS} = 1.8V



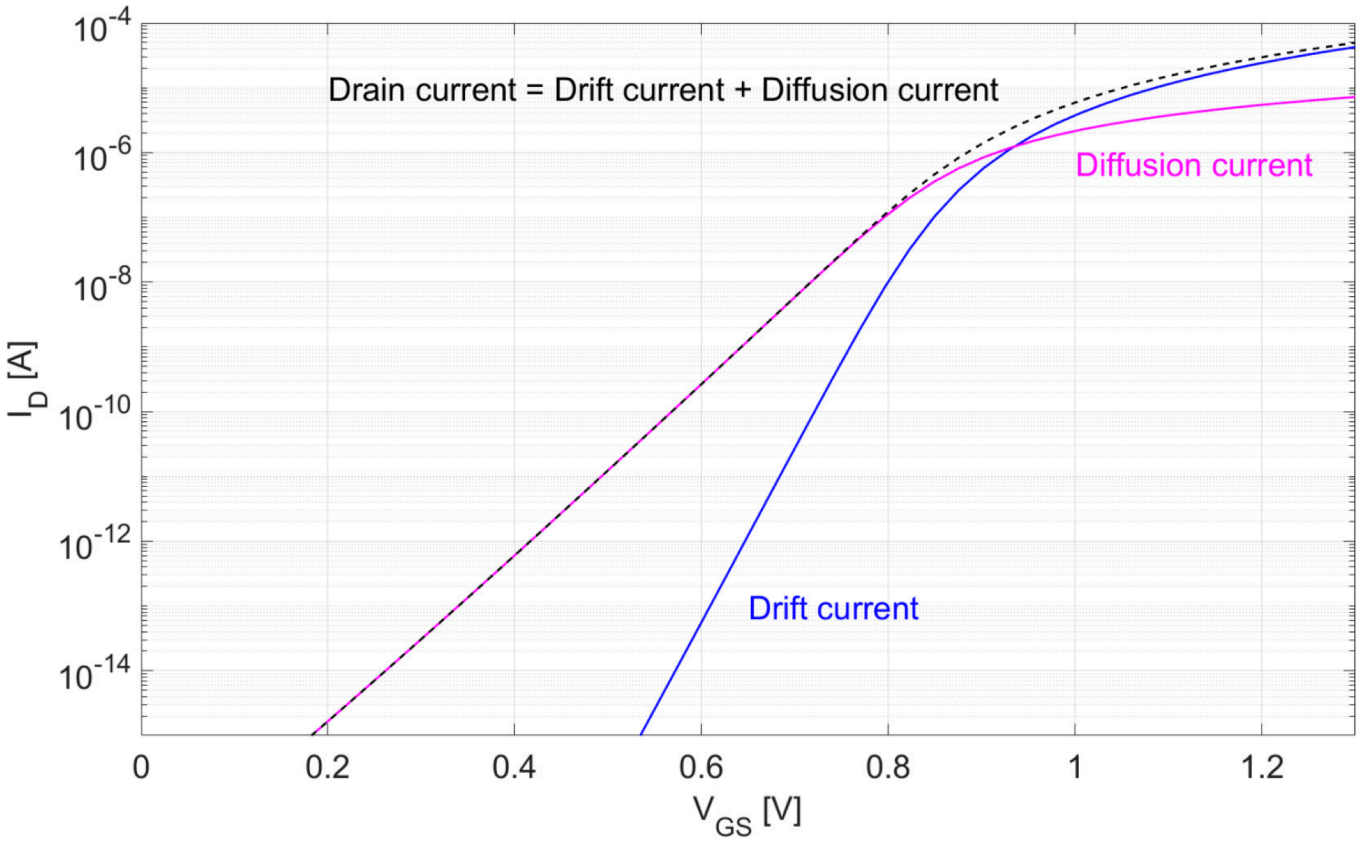
@V_{DS} = 1.8V, @V_{GS} = 0.9V



@V_{DS} = 1.8V, @V_{GS} = 0.9V

What's wrong with the square law model ?

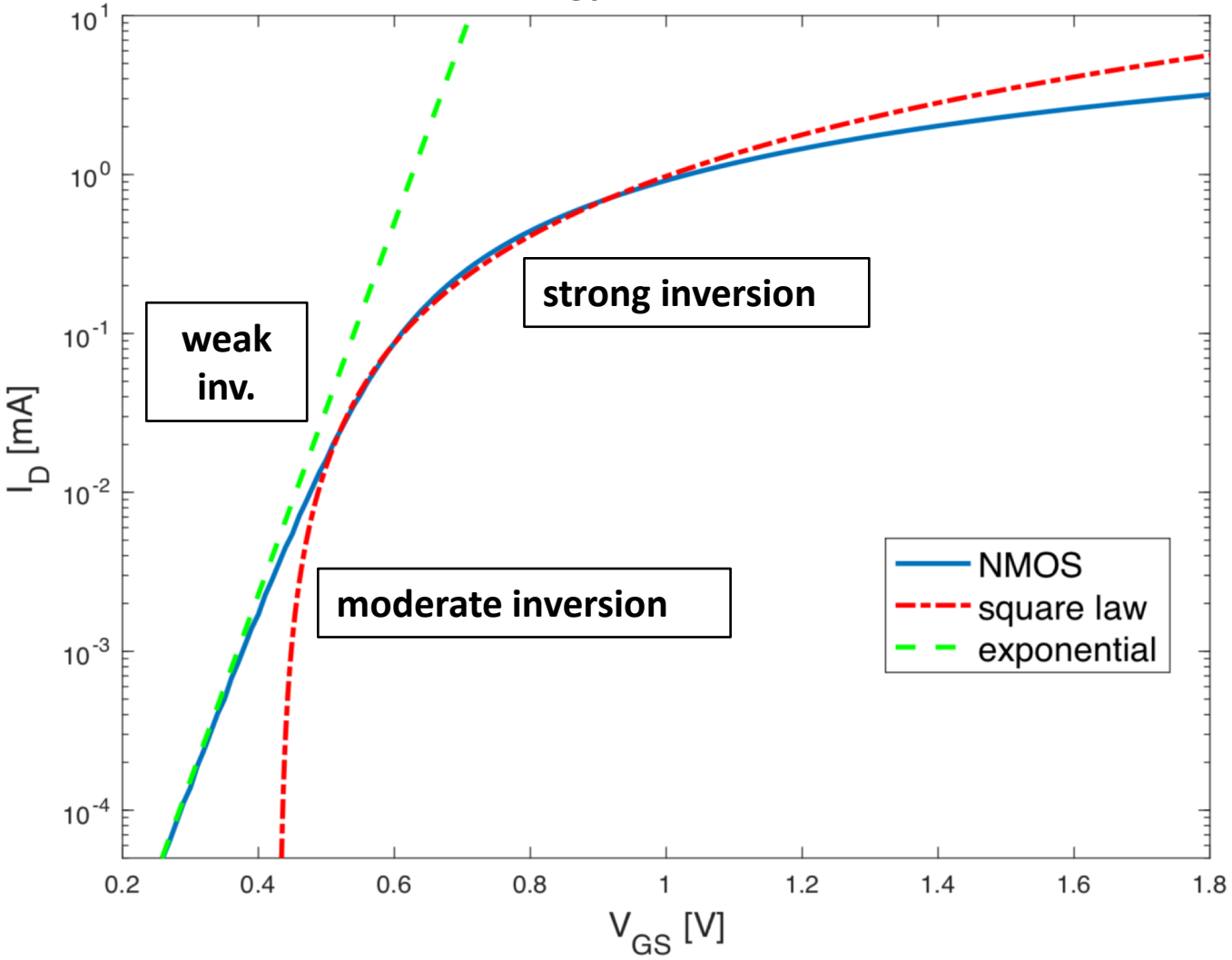
- In the derivation of the square-law model we considered only the drift current (strong inversion)
- Low-power design tends to happen in moderate and weak inversion
  - We must model both drift and diffusion currents



Drain current with the drift and diffusion contribution. The drift current dominates in strong inversion, the diffusion current dominates in weak inversion

Square-Law vs. Real Transistor

Generic 0.18um Technology – nMOST with W/L = 5um/0.18um, VDS= 1.8V



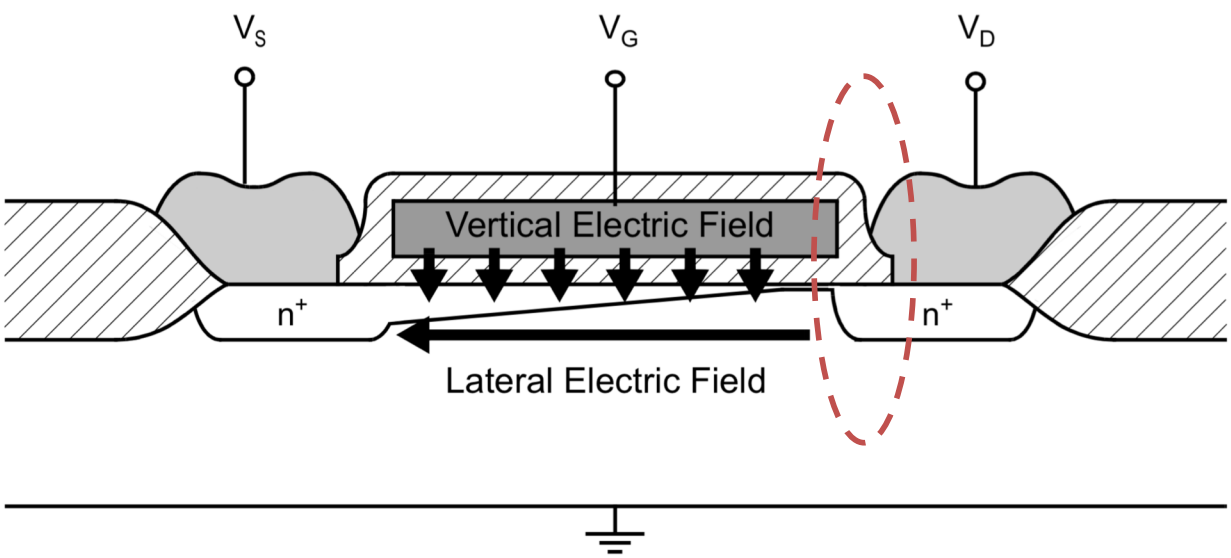
You can't even predict the current, forget about the derivatives

At large V_{GS}, the square law starts to deviate again. A more complex model is needed to iron out the discrepancy

Summary of Square Law Equation "Gotchas"

- The square law equation does not correctly handle subthreshold (cutoff) operation
  - I_D does not abruptly go to zero when V_{GS} is lowered to V_t
- The square law equation is derived for "strongly" inverted devices
  - i.e. relatively large V_{OV}
  - weak inversion and moderate inversion are also very useful operation options
- Modern MOSTs deviate from the square law equation significantly
  - There is a long list of second order effect usually as "categorized" as short channel effects
    - channel length modulation is more significant ($\lambda \propto 1/L$)
    - mobility issues (degradation due to high vertical field and saturation due to high lateral field)
    - DIBL, SCBE, SCE, RSCE, , …

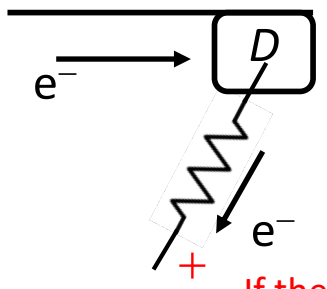
Drain Induced Barrier Lowering (DIBL)



With small L, the vertical and horizontal fields affect each other. VDS affect the amount of charge in the channel (the drain is just like another gate, just not as good at controlling the amount of charge)

- *In the square law model we attributed the I_D-V_{DS} dependence (and thus the finite intrinsic gain gm/gds) primarily to channel length modulation* ***CLM***
- *In a short channel device, it turns out that the intrinsic gain of the transistor is strongly affected by another effect called DIBL*
- *In essence, the drain can be viewed as an additional gate that modulates the inversion charge*

Substrate current-induced body effect (SCBE)



At very high V_{DS} the electrons in the channel reach ballistic speed, and they are swept against the drain so hard that they may bounce back in the substrate and cause an IR drop across the resistance of the substrate. This is equivalent to a V_{SB} drop and therefore a reduction in V_T (or in other words an increase in I_D)

If the electrons move this way it means that the positive of the voltage is on the bulk

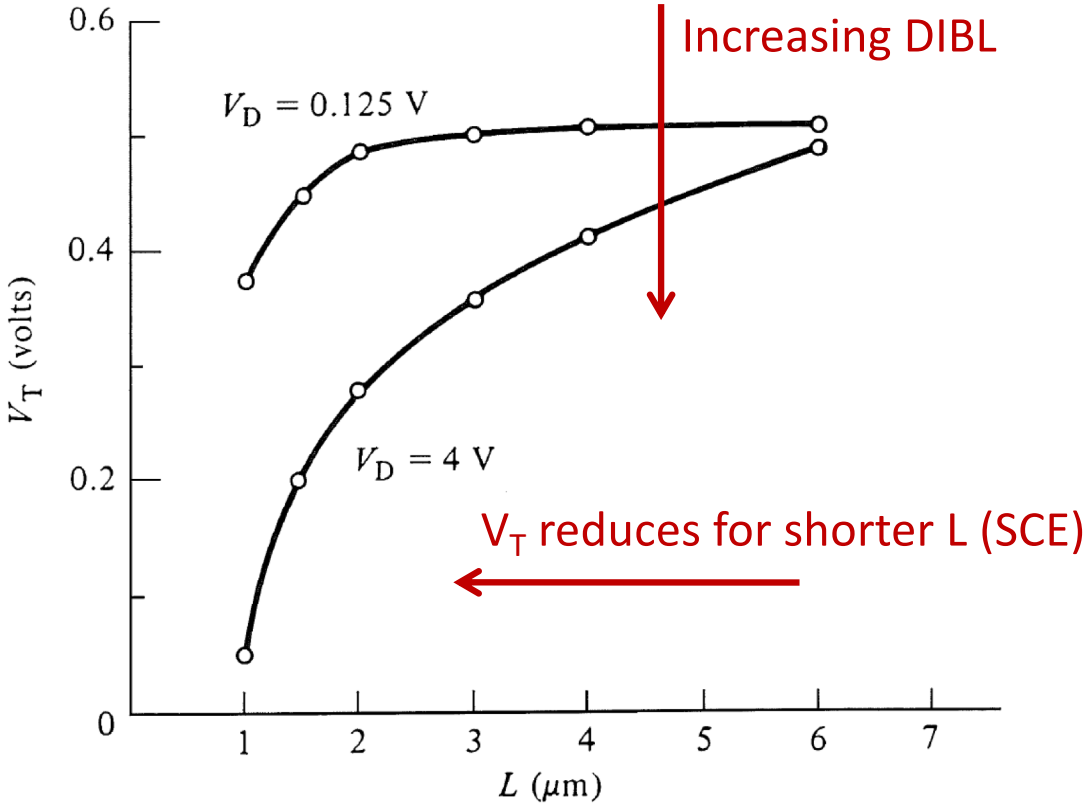
$$V_B \nearrow \leftrightarrow V_{SB} \searrow \leftrightarrow V_t \searrow \leftrightarrow I_D \nearrow$$

With today's low supply voltages SCBE is becoming less relevant

SCBE is a.k.a. hot-carrier effect

The “short channel effect” (SCE)

- *SCE manifests as a reduction of V_T as L gets smaller*
  ($L \searrow \leftrightarrow I_D \nearrow \leftrightarrow D$ sort of become another G)



[Pierret, Semiconductor Device Fundamentals, 1995 Prentice Hall]

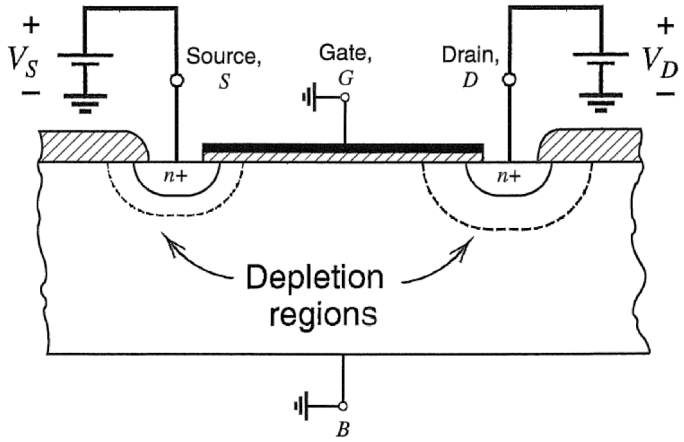
Before an inversion layer forms beneath the gate the sub-gate region must be first depleted.
In a short channel device the source and drain assist in depleting the region under the gate

NOTE:

- DIBL has to do with the drain voltage
- SCE with the depletion region overlap

Reverse Short Channel Effect (RSCE)

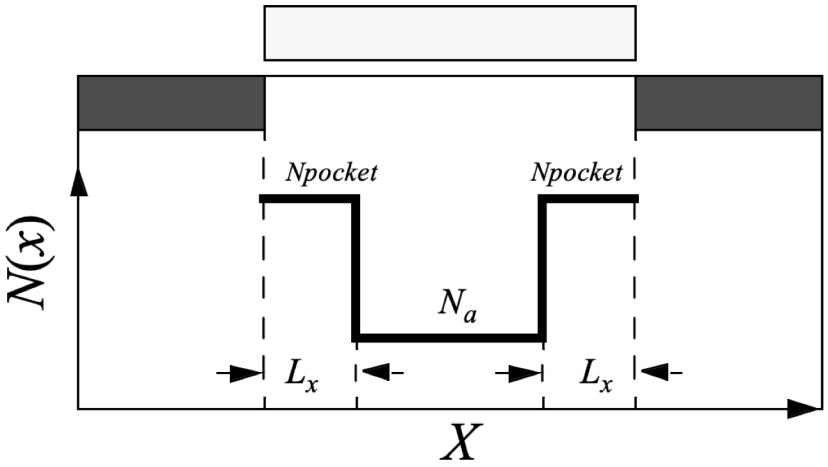
- *If we keep making the channel shorter and shorter at some point the depletion regions around D and S touch each other and we do not have a transistor anymore (punch-through). What can we do to avoid this effect? Build pocket implants. That is increase the doping of p close to S and D (since we do not want to reduce n+) . This way the depletion regions narrows and we can build devices with shorter L*



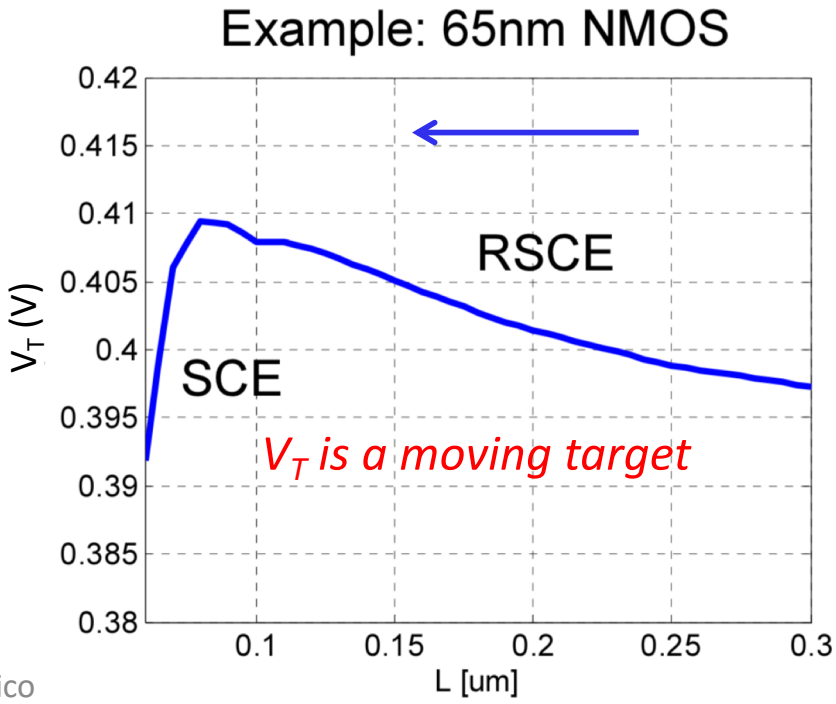
Reverse Short Channel Effect (RSCE)

Source: Murmann

- To reduce the width of the source/drain depletion regions, modern processes
- use pocket implants ("halos") → high doping
- It is difficult to predict what happen to V_t.
  It depends on how the device is engineered
- As L gets shorter the average doping in the channel increases and results in a higher V_t (RSCE). This is true only up to a point. After, SCE takes over.



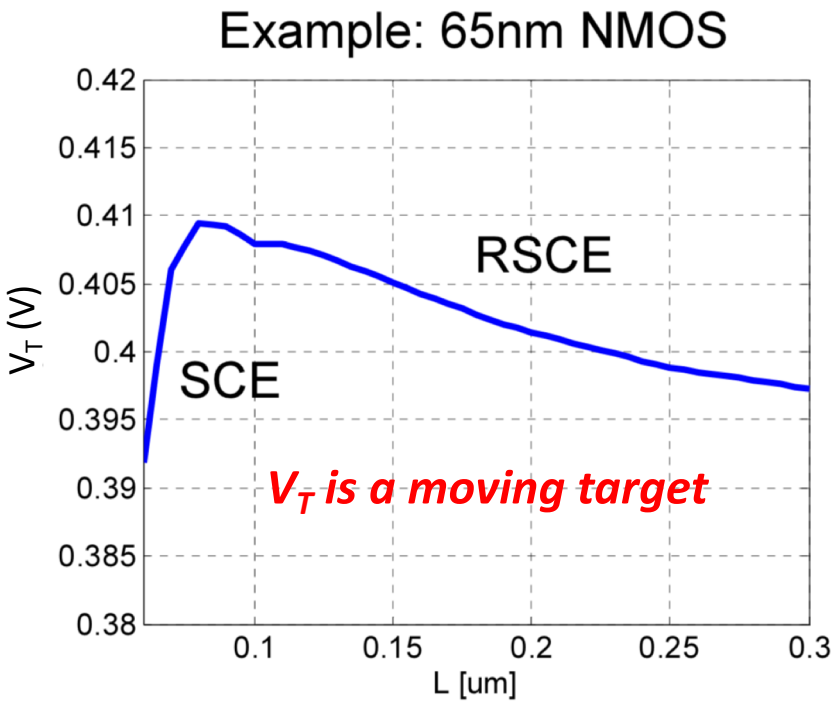
[BSIM3v3.3 Manual]



Reverse Short Channel Effect (RSCE)

Source: Murmann

- If we make L “very small” V_t goes down. But, also if we make L “very large” V_t goes down.



- The “concept” of **V_t** is quite **questionable** !

Bottom Line

- In modern technologies the square law is:
  - still useful for gaining intuition (**qualitative** understanding)
  - but not useful for systematic design (cannot be used for **quantitative** analysis)

Are hand calculation still an option ?

source: Carusone

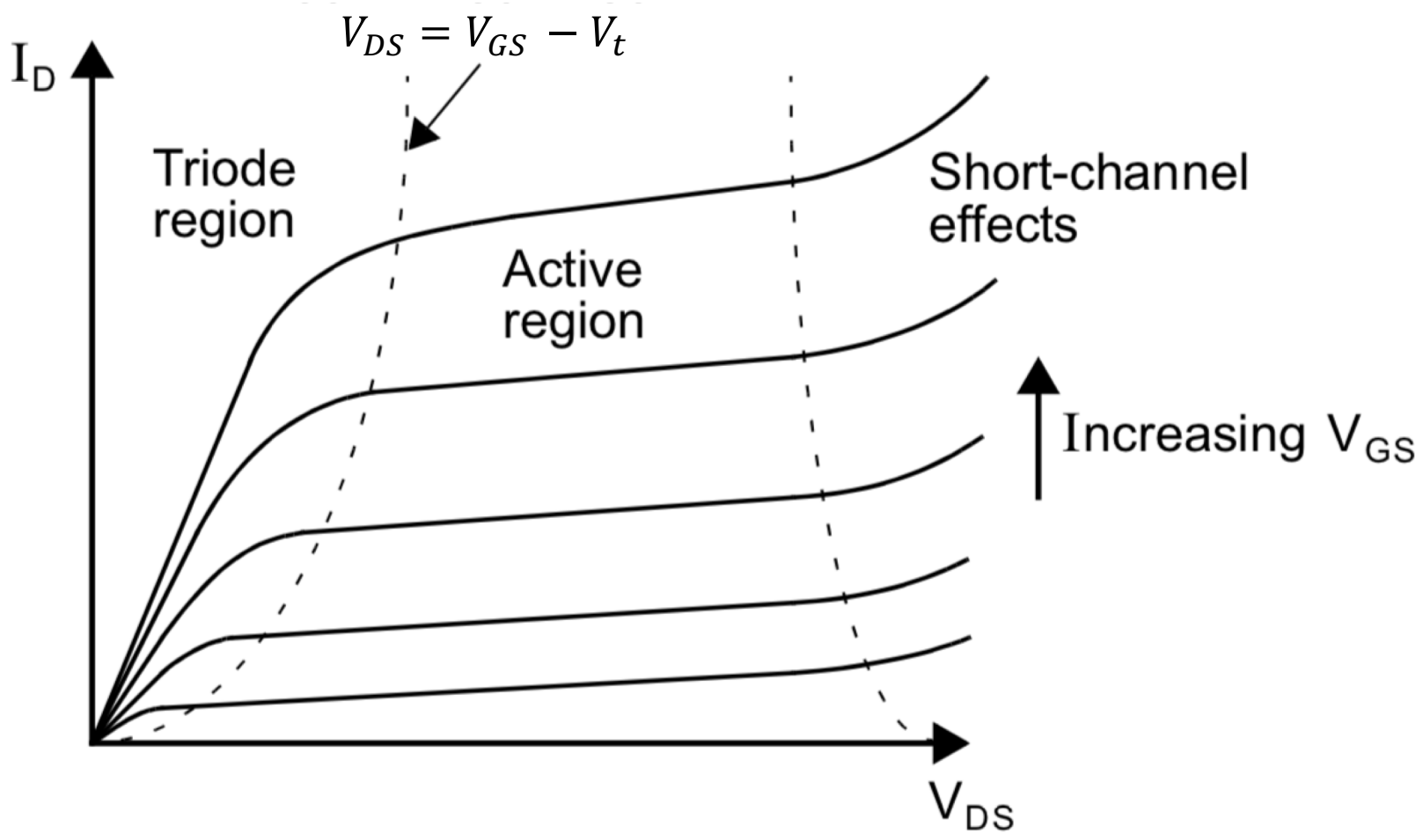
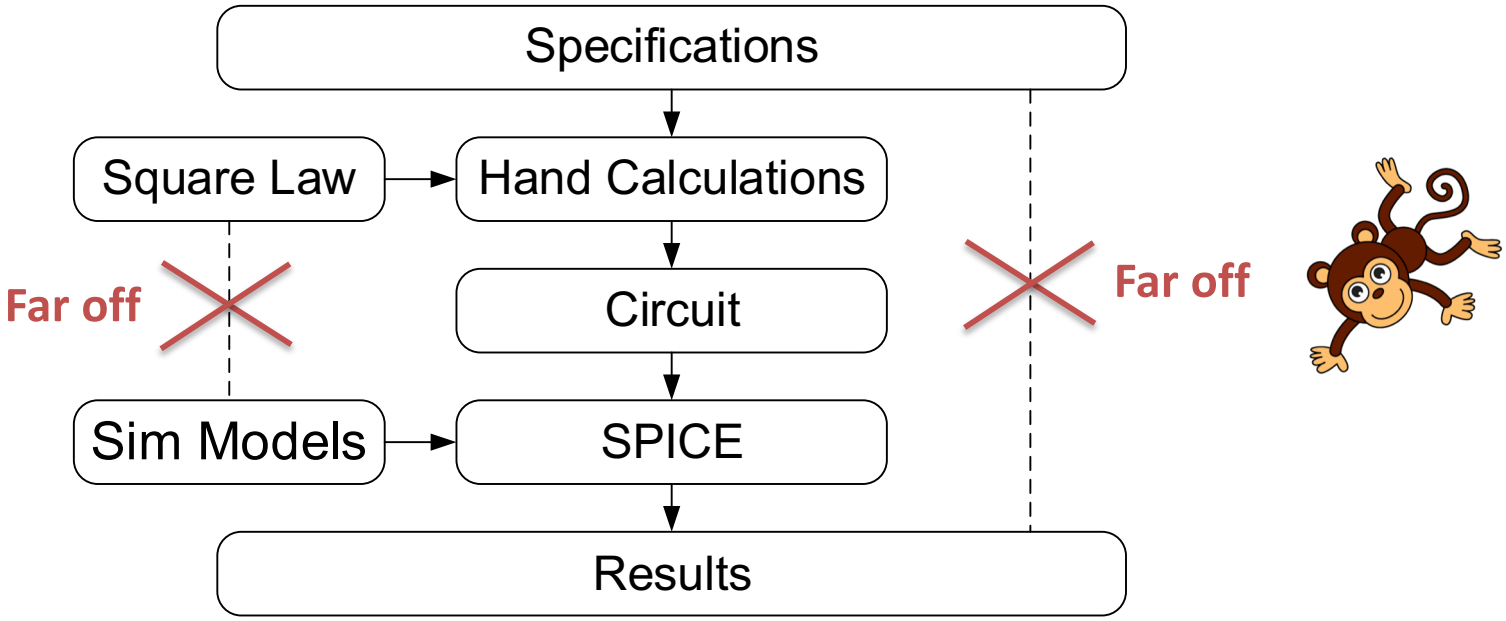


Fig. 1.16 I_D versus V_{DS} for different values of V_{GS}.

Typical Analog circuits design flow based on square law hand calculations and SPICE simulation



- The complexity of the transistor model preclude the derivation of simple closed form analytical expressions
- Design process takes multiple iterations and “hand” tweaking of the transistor sizing before converging toward a working circuit

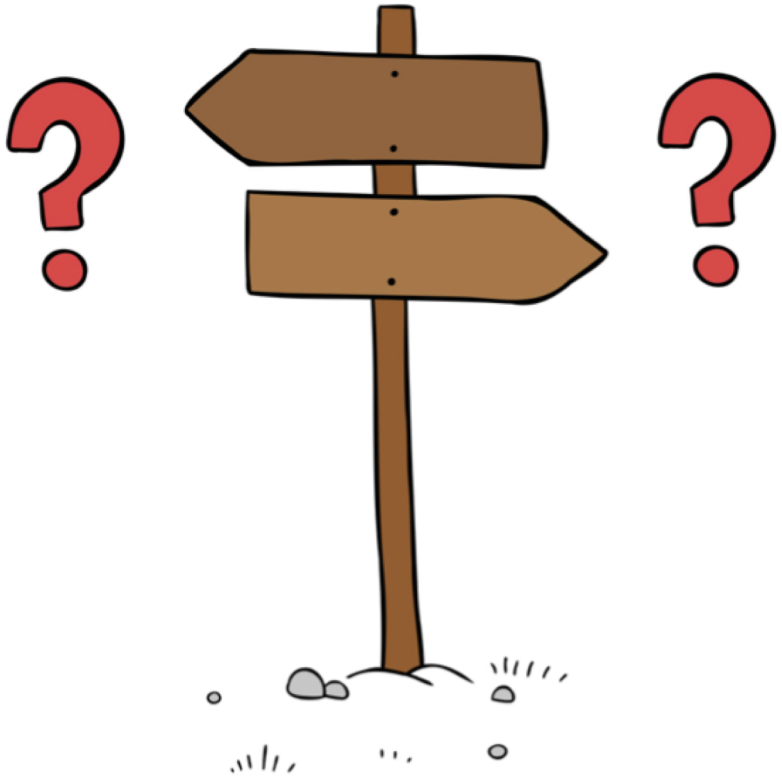
Unfortunate consequence

- In absence of a simple set of equations for hand analysis, many designers tend to converge toward a “SPICE monkey” design methodology
  - No hand calculation, iterate in SPICE until the circuit “somehow” meets specification
  - Typically results is sub-optimal design, uninformed design decisions, etc.

- SPICE is a validation tool, not a design tool !!!

What should we do ?

The land of better model equations (EKV model) & design using the inversion coefficient

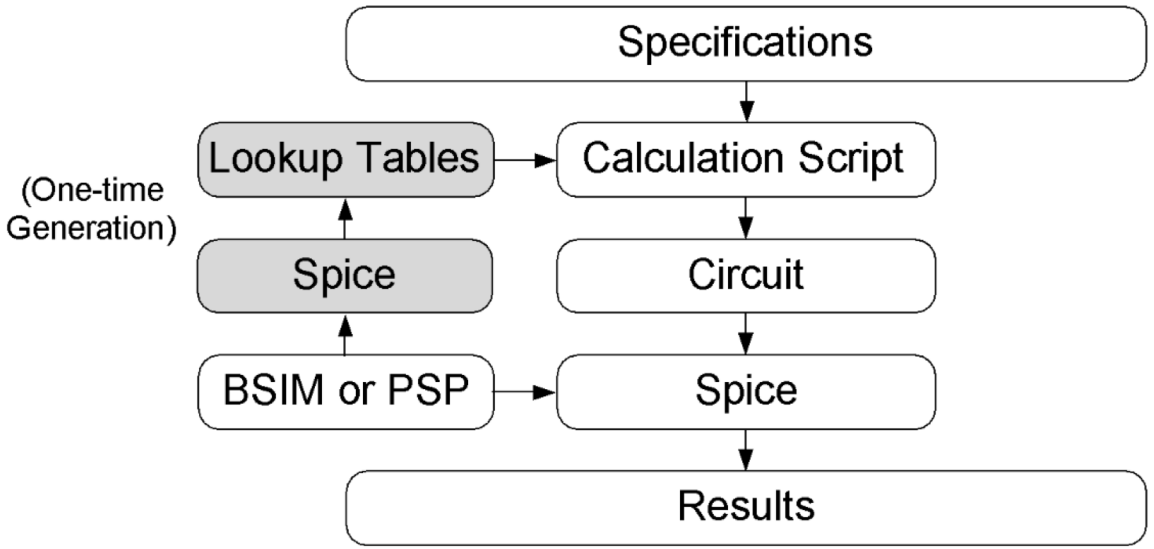


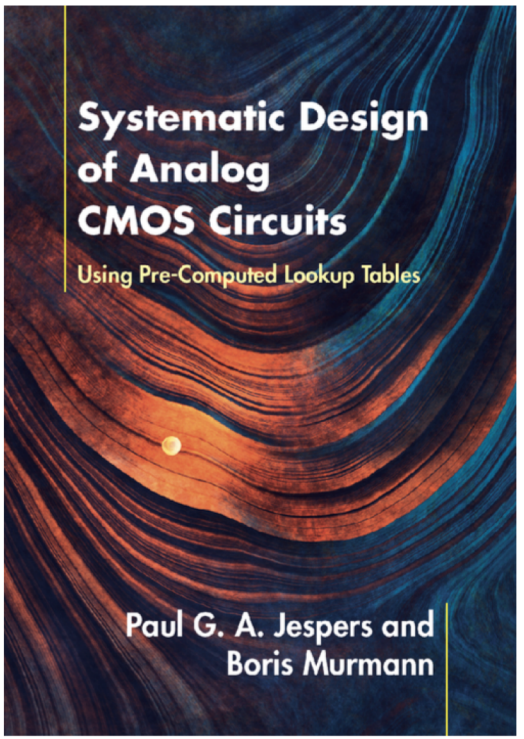
The land of no equations & design using the lookup tables (Jespers & Murmann)

combine drift and diffusion into a single equation (it is not a closed form equation)

EKV = Enz, Krummenacher, Vittoz

Sizing Using Precomputed LUTs





Numerical method rather than analytical method

LUTs are "physics" agnostic (they works for any device it may be invented)

Starting Point: Technology Characterization via DC Sweep

- The schematics and the scripts to generate the LUTs, as well as the LUTs data are available at: https://github.com/bmurmann/Book-on-gm-ID-design/tree/main/starter_files_open_source_tools

- In this tutorial we are going to use the PDK **`gf180mcuD`** https://github.com/bmurmann/Book-on-gm-ID-design/tree/main/starter_files_open_source_tools/gf180mcuD

4-D data table

$$I_D(L, V_{GS}, V_{DS}, V_{SB})$$
$$V_t(L, V_{GS}, V_{DS}, V_{SB})$$
$$g_m(L, V_{GS}, V_{DS}, V_{SB})$$
$$\dots$$

Data for gf180mcuD 3.3V devices

nfet_03v3.mat
pfet_03v3.mat

Technology Characterization via DC Sweep

COMMANDS1

```
.param wx=5.0u lx=0.28u
.noise v(n) vg lin 1 1 1 1

.control
option numdgt = 3
set wr_singlescale
set wr_vecnames

compose l_vec  values 0.28u 0.29u
+ 0.3u 0.4u 0.5u 0.6u 0.7u 0.8u 0.9u 1u 2u 3u
compose vg_vec start= 0 stop=3.301  step=25m
compose vd_vec start= 0 stop=3.301  step=25m
compose vb_vec start= 0 stop=-0.4 step=-0.2

foreach var1 $&l_vec
  alterparam lx=$var1
  reset
  foreach var2 $&vg_vec
    alter vg $var2
    foreach var3 $&vd_vec
      alter vd $var3
      foreach var4 $&vb_vec
        alter vb $var4
        run
        wrdata techsweep_nfet_03v3.txt noise1.all
        destroy all
        set appendwrite
        unset set wr_vecnames
      end
    end
  end
end
unset appendwrite

alterparam lx=0.28u
reset
op
show
write techsweep_nfet_03v3.raw
.endc
```

COMMANDS2

```
.save @m.xm1.m0[capbd]
.save @m.xm1.m0[capbs]
.save @m.xm1.m0[cdd]
.save @m.xm1.m0[cgb]
.save @m.xm1.m0[cgd]
.save @m.xm1.m0[cgdo]
.save @m.xm1.m0[cgg]
.save @m.xm1.m0[cgs]
.save @m.xm1.m0[cgso]
.save @m.xm1.m0[css]
.save @m.xm1.m0[gds]
.save @m.xm1.m0[gm]
.save @m.xm1.m0[gmbs]
.save @m.xm1.m0[id]
.save @m.xm1.m0[l]
.save @m.xm1.m0[vth]
.save @vb[dc]
.save @vd[dc]
.save @vg[dc]
.save onoise.m.xm1.m0.id
.save onoise.m.xm1.m0.1overf
.save g d b n
```

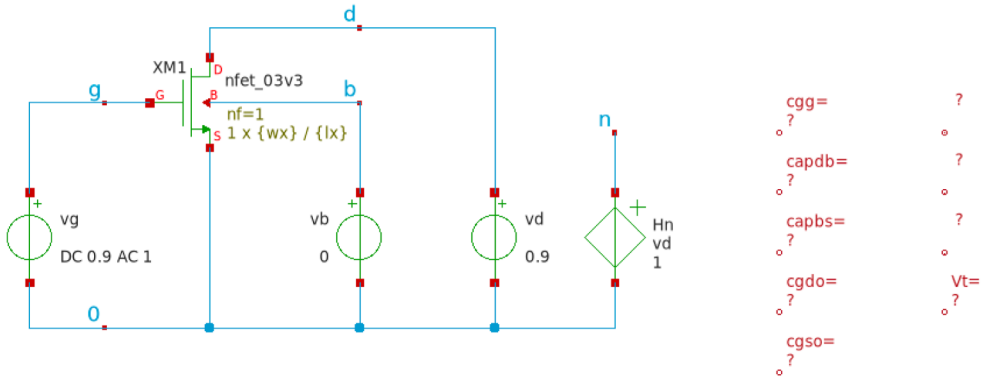
- sweep vg,vd,vb and L across reasonable ranges
- To first order all transistor's parameters scale linearly with W, so no need to sweep W

MODELS

```
.include $::180MCU_MODELS/design.ngspice
.lib $::180MCU_MODELS/sm141064.ngspice typical
```

save, netlist & simulate

load op & annotate



techsweep_nfet_03v3.sch

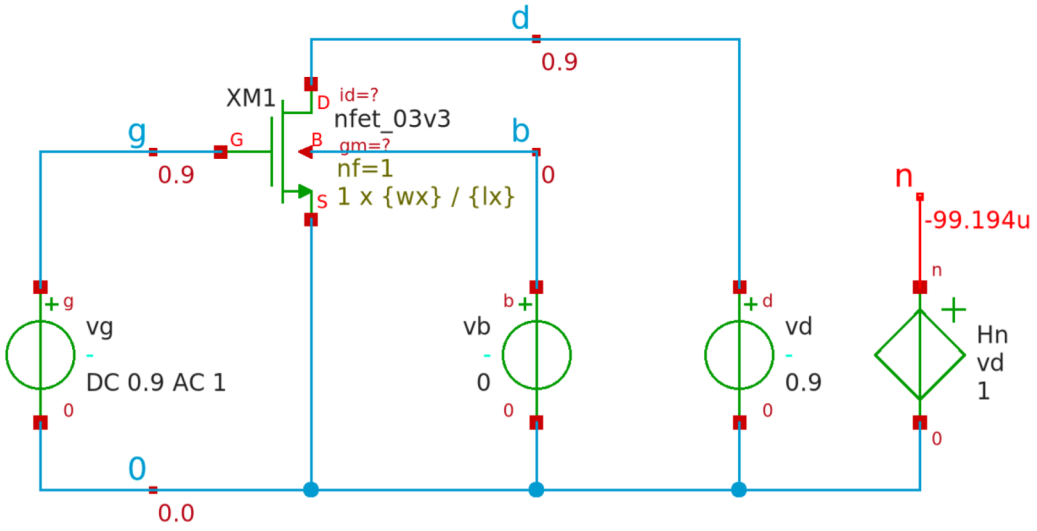
/foss/designs/gf180-2025 >

Run the characterization of the nfet_03v3

- Hit Ctrl + LMB on the launcher → save, netlist & simulate
- Hit Ctrl + LMB on the launcher → load op & annotate

→ save, netlist & simulate

→ load op & annotate



cgg= 4.12e-15

capdb= 3.777e-15

capbs= 4.714e-15

cgdo= 9.372e-16

cgso= 9.993e-16

gm/ID= 5.717

fT_intrinsic= 2.191e+10

gm/gds= 25.18

Vt= 0.6395

Where did the results go?

- The default settings will send the results to:

```
xschem [/foss/designs/gf180-2025] echo $netlist_dir
/headless/.xschem/simulations
```

- Results:

```
techsweep_nfet_03v3.raw
techsweep_nfet_03v3.spice
techsweep_nfet_03v3.txt
```

- The transistors spice models are at:

```
xschem [/foss/designs/gf180-2025] echo $180MCU_MODELS
/foss/pdks/gf180mcuD/libs.tech/ngspice
```

A quick aside …

- To see your Linux environment variables, use the following command:

```
/foss/designs/gf180-2025 > printenv
```

- The ngspice models for the gf180mcuD's transistors **nfet_03v3** and **pfet_03v3** are level 54 (BSIM 4) version 4.5
- The models are located at:

```
/foss/pdks/gf180mcuD/libs.tech/ngspice >
```

- The files to include in your netlists are:

```
design.ngspice
sm141064.ngspice
```

- Ngspice User's Manual

https://ngspice.sourceforge.io/docs/ngspice-html-manual/manual.xhtml

BSIM4 accessible instance parameters are covered in ch. 27

Setup your preferences

.spiceinit ← `/foss/designs/gf180-2025 >`

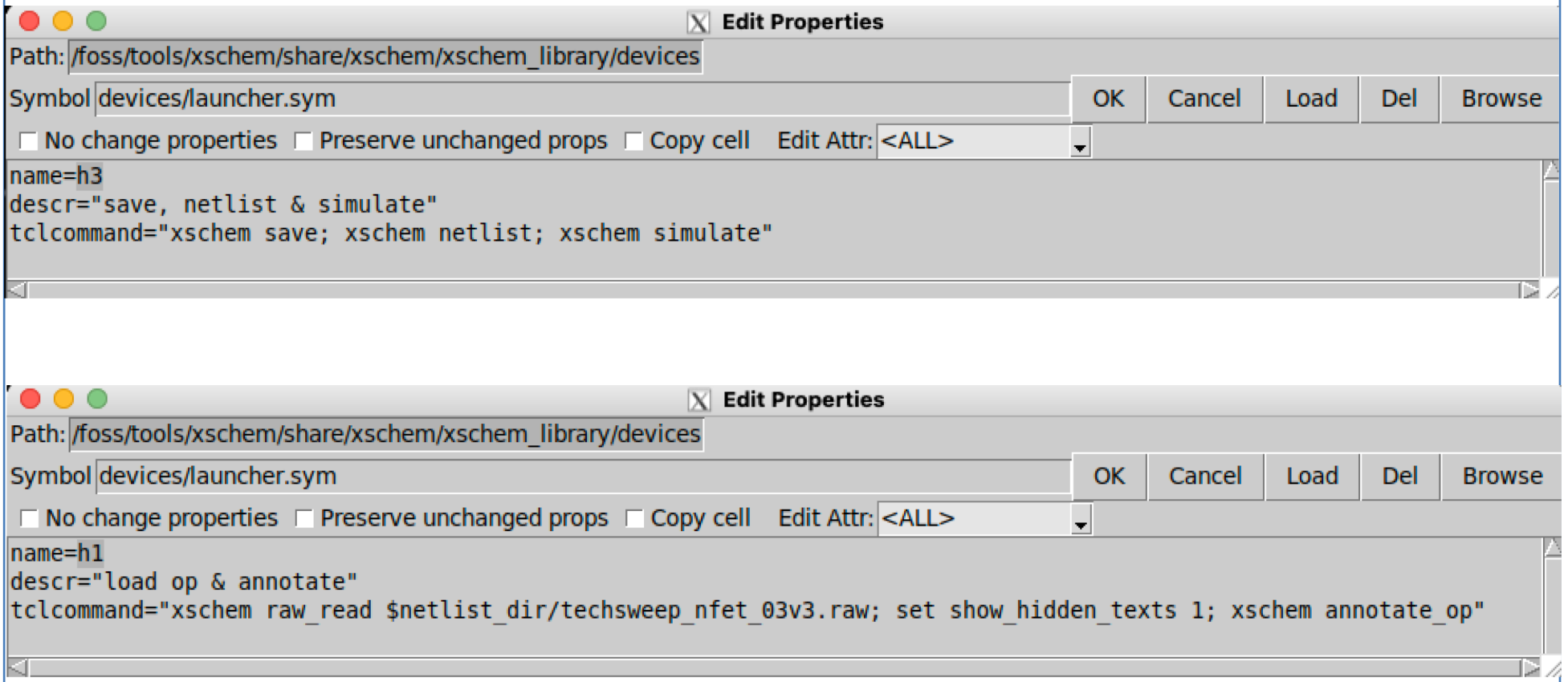
```
set ngbehavior=hsa
set ng_nomodcheck
set color0=white
set color1=black
set xbrushwidth=2
set altshow
```

xschemrc ← `/foss/designs/gf180-2025 >`

```
## Source the PDK xschemrc file
source $env(PDK_ROOT)/$env(PDK)/libs.tech/xschem/xschemrc
## Set the netlists and the simulations to be the current directory
set netlist_dir $env(PWD)
## Show netlist after netlist command
set netlist_show 1
## Add local directory to library
append XSCHEM_LIBRARY_PATH :[file dirname [info script]]
```

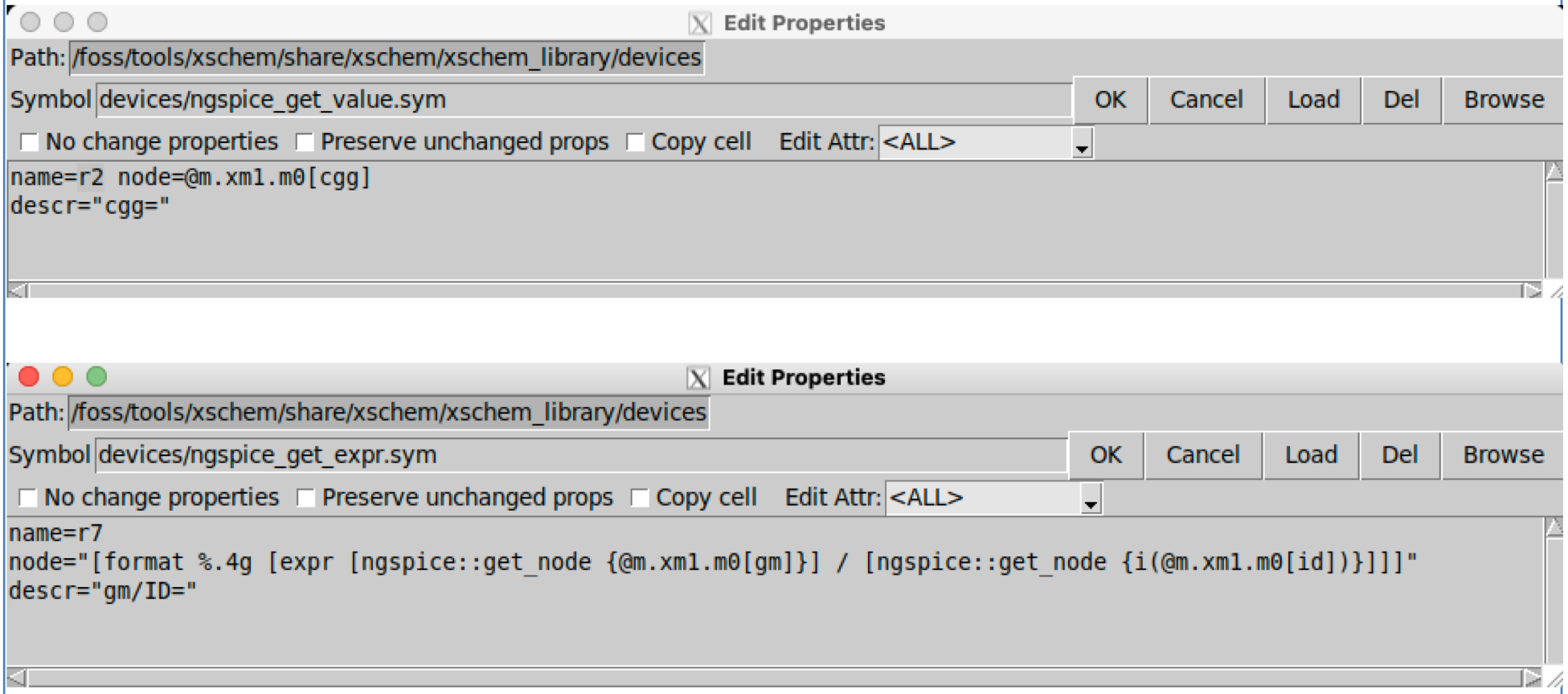
What's under the hood of the launchers

- RMB > Edit attributes on the launchers



… and the other back-annotations

- RMB > Edit attributes



Run the characterization of the pfet_03v3

COMMANDS1

```
.param wx=5.0u lx=0.28u
.noise v(n) vg lin 1 1 1 1

.control
option numdgt = 3
set wr_singlescale
set wr_vecnames

compose l_vec  values 0.28u 0.29u
+ 0.3u 0.4u 0.5u 0.6u 0.7u 0.8u 0.9u 1u 2u 3u
compose vg_vec start= 0 stop=3.301  step=25m
compose vd_vec start= 0 stop=3.301  step=25m
compose vb_vec start= 0 stop=-0.4 step=-0.2

foreach var1 $&l_vec
  alterparam lx=$var1
  reset
  foreach var2 $&vg_vec
    alter vg $var2
    foreach var3 $&vd_vec
      alter vd $var3
      foreach var4 $&vb_vec
        alter vb $var4
        run
        wrdata techsweep_pfet_03v3.txt noise1.all
        destroy all
        set appendwrite
        unset set wr_vecnames
      end
    end
  end
end
unset appendwrite

alterparam lx=0.28u
reset
op
show
write techsweep_pfet_03v3.raw
.endc
```

COMMANDS2

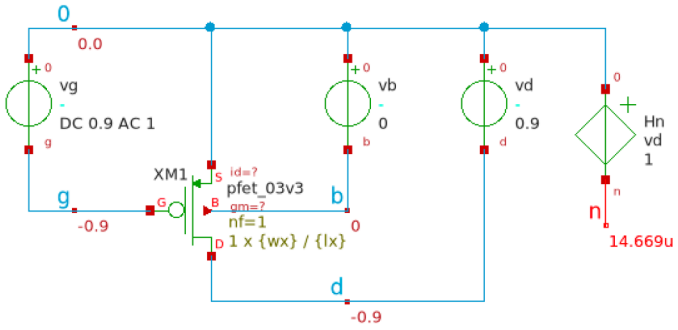
```
.save @m.xm1.m0[capbd]
.save @m.xm1.m0[capbs]
.save @m.xm1.m0[cdd]
.save @m.xm1.m0[cgb]
.save @m.xm1.m0[cgd]
.save @m.xm1.m0[cgdo]
.save @m.xm1.m0[cgg]
.save @m.xm1.m0[cgs]
.save @m.xm1.m0[cgso]
.save @m.xm1.m0[css]
.save @m.xm1.m0[gds]
.save @m.xm1.m0[gm]
.save @m.xm1.m0[gmbs]
.save @m.xm1.m0[id]
.save @m.xm1.m0[l]
.save @m.xm1.m0[vth]
.save @vb[dc]
.save @vd[dc]
.save @vg[dc]
.save onoise.m.xm1.m0.id
.save onoise.m.xm1.m0.1overf
.save g d b n
```

MODELS

```
.include $::180MCU_MODELS/design.ngspice
.lib $::180MCU_MODELS/sm141064.ngspice typical
```

save, netlist & simulate

load op & annotate



cgg= 4.576e-15
capdb= 3.415e-15
capbs= 4.061e-15
cgdo= 7.949e-16
cgso= 8.197e-16
gm/ID= 9.237
fT_intrinsic= 4.713e+09
gm/gds= 44.69
Vt= 0.731

techsweep_pfet_03v3.sch

```
/foss/designs/gf180-2025 >
```

Generating the Data

Data for gf180mcuD 3.3V devices

nfet_03v3.mat

pfet_03v3.mat

```
/foss/designs/gf180-2025 > jupyter notebook techsweep_txt_to_mat.ipynb
```

```
import pandas as pd
import numpy as np
from numpy._core.records import fromarrays
from scipy.io import savemat
choice = 0  # start from 0
devices = ['nfet_03v3', 'pfet_03v3']

# widths used for characterization
w = np.array([5.0, 5.0])
nfing = np.array([1, 1])
Lmin = np.array([0.28, 0.28])
# read ngspice data
df_raw = pd.read_csv('./techsweep_'+devices[choice]+'.txt', sep='\\s+')
par_names = df_raw.columns.to_list()
fet_name = par_names[1].split('[')[0]

# remove unwanted columns and rename for readability
df = df_raw.drop(['frequency', 'frequency.1'], axis=1)
df = df.apply(pd.to_numeric)
df.columns = df.columns.str.replace(fet_name, '')
df.columns = df.columns.str.replace(fet_name[1:], '')
df.columns = df.columns.str.replace('[dc]', '')
df.columns = df.columns.str.replace('onoise..', 'n')
df.columns = df.columns.str.removeprefix('@')
df.columns = df.columns.str.removeprefix('[')
df.columns = df.columns.str.removesuffix(']')
```

set choice = 0 to generate nfet_03v3.mat
set choice = 1 to generate pfet_03v3.mat

...

Generating the Data

Data for gf180mcuD 3.3V devices

nfet_03v3.mat

pfet_03v3.mat

```
# sweep variable vectors
l =   np.round(np.unique(df['l'])*1e6, 2)
vgs = np.unique(df['vg'])
vds = np.unique(df['vd'])
vsb = np.unique(-df['vb'])
# ngspice sweep order is l, vgs, vds, vsb
dims = [len(l), len(vgs), len(vds), len(vsb)]
id = np.reshape(df['id'].values, dims)
vt = np.reshape(df['vth'].values, dims)
gm = np.reshape(df['gm'].values, dims)
gmb = np.reshape(df['gmbs'].values, dims)
gds = np.reshape(df['gds'].values, dims)
cgg = np.reshape(df['cgg'].values, dims) \
      + np.reshape(df['cgdo'].values, dims) + np.reshape(df['cgso'].values, dims)
cgb = -np.reshape(df['cgb'].values, dims)
cgd = -np.reshape(df['cgd'].values, dims) \
      + np.reshape(df['cgdo'].values, dims)
cgs = -np.reshape(df['cgs'].values, dims) \
      + np.reshape(df['cgso'].values, dims)
cdd = np.reshape(df['cdd'].values, dims) \
      + np.reshape(df['capbd'].values, dims) \
      + np.reshape(df['cgdo'].values, dims)
css = np.reshape(df['css'].values, dims) \
      + np.reshape(df['capbs'].values, dims) \
      + np.reshape(df['cgso'].values, dims)
sth = np.reshape(df['nid'].values, dims)**2
sfl = np.reshape(df['n1overf'].values, dims)**2
```

...

Generating the Data

Data for gf180mcuD 3.3V devices

nfet_03v3.mat

pfet_03v3.mat

```
dic = {
  "INFO": "GlobalFoundries, 180nm MCU CMOS , BSIM4",
  "CORNER": "NOM",
  "TEMP": 300.0,
  "VGS": vgs,
  "VDS": vds,
  "VSB": vsb,
  "L": l,
  "W": w[choice],
  "NFING": nfing[choice],
  "ID": id,
  "VT": vt,
  "GM": gm,
  "GMB": gmb,
  "GDS": gds,
  "CGG": cgg,
  "CGB": cgb,
  "CGD": cgd,
  "CGS": cgs,
  "CDD": cdd,
  "CSS": css,
  "STH": sth,
  "SFL": sfl
}
savemat('./'+devices[choice]+'.mat', {devices[choice]: dic})
```

Accessing the Data Using pygmid

pygmid repo: https://github.com/dreoilin/pygmid

pygmid

A python3 implementation of the gm/ID starter kit

Report bug · Request feature

```
# test_lut.py

from pygmid import Lookup as lk

# read data for gf180mcuD NMOS and PMOS device
# the range of channel lengths is 0.28 to 3
# the range for VGS, VDS is 0 to 3.3
n = lk('./nfet_03v3.mat')
p = lk('./pfet_03v3.mat')
```

Accessing the Data Using pygmid

```
# basic example of usage mode 1
gm_n = n.look_up('GM', L=0.28, VGS=0.9, VDS=0.9, VSB=0.0)
ID_n = n.look_up('ID', L=0.28, VGS=0.9, VDS=0.9, VSB=0.0)
gm_p = p.look_up('GM', L=0.28, VGS=0.9, VDS=0.9, VSB=0.0)
ID_p = p.look_up('ID', L=0.28, VGS=0.9, VDS=0.9, VSB=0.0)
print(f'nch - (gm/ID) is: {gm_n/ID_n:0.2f} (S/A)')
print(f'pch - (gm/ID) is: {gm_p/ID_p:0.2f} (S/A)')
```

```
# check bias
gm_id_n = gm_n/ID_n
VGS = n.lookupVGS(GM_ID = gm_id_n, VDS = 0.9, VSB = 0.0, L = 0.28)
print(f'nch - VGS is: {VGS:0.3f} (V)')
gm_id_p = gm_p/ID_p
VGS = p.lookupVGS(GM_ID = gm_id_p, VDS = 0.9, VSB = 0.0, L = 0.28)
print(f'pch - VGS is: {VGS:0.3f} (V)')
```

```
# basic example of usage mode 2
GM_GDS_n= n.look_up('GM_GDS', GM_ID=gm_id_n, L=0.28, VSB=0, VDS = 0.9)
GM_GDS_p= p.look_up('GM_GDS', GM_ID=gm_id_p, L=0.28, VSB=0, VDS = 0.9)
print(f'nch - GM_GDS is: {GM_GDS_n:0.2f}')
print(f'pch - GM_GDS is: {GM_GDS_p:0.2f}')
```

Accessing the Data Using pygmid

```
/foss/designs/gf180-2025 > python test_lut.py
nch - (gm/ID) is: 5.717 (S/A)
pch - (gm/ID) is: 9.237 (S/A)
nch - VGS is: 0.90 (V)
pch - VGS is: 0.90 (V)
nch - GM_GDS is: 25.18
pch - GM_GDS is: 44.69
```