Joint AWG MOS-AK Workshop, Dec.7, 2022

EKV3 in NGSPICE using ADMSXL

Nikolaos Makris(TUC, IESL/FORTH)
Matthias Bucher (TUC)
Loukas Chevas (TUC)

nmakris@tuc.gr mbucher@tuc.gr lchevas@tuc.gr

Outline

- ❑ EKV3 Model
- ❑ EKV3 Verilog-A structure
- ❑ Phenomena covered by EKV3
- ❑ EKV3 in Simulators
- ❑ EKV3 in ngspice
- ❑ EKV3 simulations in JupyterLab
- ❑ Conclusions

EKV3 Model

- Compact MOS transistor model
  - Analog/RF IC design
  - Special focus in RF and Noise performance
  - Further development and maintenance from TUC
- Charge based approach
  - Smooth transition from weak – moderate – strong inversion
  - Charge based expressions -> avoidance of convergence problems
- Successor of EKV2.6 (since 1997)
- Utilized in various CMOS technologies
- Used in commercial PDKs
- Several model variations to cover different R&D needs
- Distributed to Keysight, Cadence, Synopsys.

EKV3 Model

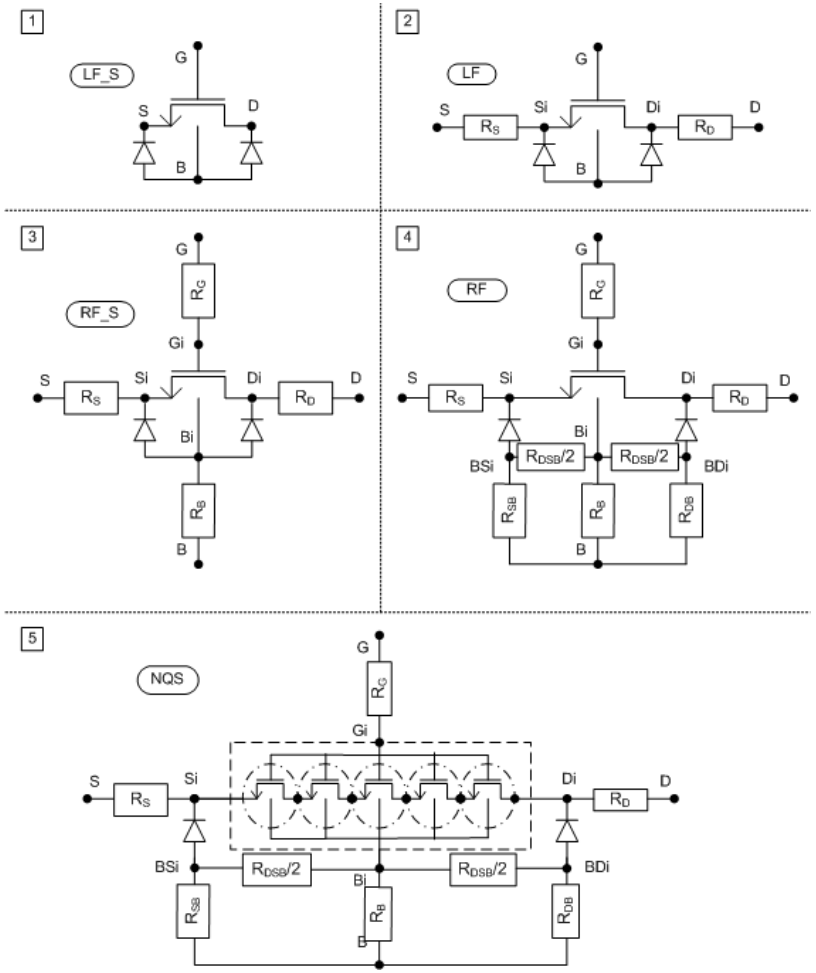
- EKV3 model contributors
  - **Coordination:**
    - ✓Matthias Bucher
  - **Coding, implementation and testing:**
    - ✓ Antonios Bazigos, Matthias Bucher, Marianna Chalkiadaki, Nikolaos Makris, Nikolaos Mavredakis
  - **Contributions to model formulation and coding:**
    - ✓Francois Krummenacher, Michel Sallese, Christian Enz, Eric A. Vittoz
  - **Contributions to noise modelling:**
    - ✓Ananda Roy, Christian Enz, Nikolaos Mavredakis
  - **Contributions to code standardization:**
    - ✓Wladek Grabinski

EKV3 Model

- Code versions
  - 30x.xx TBA **(Under development)**
  - 302.00 April 2, 2015 **(Currently distributed)**
  - 301.04 July 31, 2014 **(revised version)**
  - 301.04 September 4, 2009
  - 301.02 June 28, 2008 **(Previously distributed)**
  - 301.01 November 22, 2007
  - 301.00 September 14, 2007
  - 300.03 August 1, 2007
  - 300.02 April 23, 2007
  - 300.01 October 28, 2006
  - 300.00 March 23, 2005
- Several sub and special versions

EKV3 Verilog-A structure

- EKV3 Verilog-A Code
  - version 302.00
    - ✓ Operating point info addition
    - ✓ Enhanced flicker noise model
    - ✓ Several bug fixes
  - Different model instances
    - ✓ DC_S,DC: Mainly DC and analog operation
    - ✓ RF_S, RF: Additional nodes to cover RF behavior
    - ✓ NQS: FET Channel segmentation
  - EKV3 file hierarchy
    - ✓ ekv3.va: Core model code
      - Externally included files
        - Phenomena: noise, overlap capacitances, edge effect, GID(S)L, series resistances, extrinsic diodes, bulk current, gat current, fringing
        - Functionality: model variables & parameters, debug info, operating points info



A. Bazigos, PhD dissertation, 2008

Phenomena covered by EKV3 (1/2)

MODELED EFFECT	PARAMETERS / MODELING PRACTICE
Physical Modelling of Charges (Accumulation Region Polysilicon Depletion, Quantum Mechanical Effects**)**	COX(TOX), PHIF, GAMMA(NSUB), VTO(VFB), GAMMAG(NGATE)
Bias-Dependent Overlap Capacitances	LOV, GAMMAOV(NOV), VFBOV
NQS	Channel segmentation
RF	External subcircuit (RG,RSUB w/ scaling)
Mobility (Reduction due to Vertical Field Effect) Surface Roughness-, Phonon-, Coulomb Scattering	KP(U0), E0, E1, ETA, ZC, THC
Impact Ionization Current	IBA, IBB, IBN
Gate Current	(IGS, IGD, IGB) KG, XB, UB
Lateral Field Effect (VS,CLM)	UCRIT(VSAT), LAMBDA, DELTA
Reverse Short Channel Effect	LR, QLR, NLR

Phenomena covered by EKV3 (2/2)

MODELED EFFECT	MODELING PRACTICE
Inverse Narrow Width Effect	WR, QWR, NWR
Drain Induced Barrier Lowering	ETAD, SIGMAD
Source and Drain Charge Sharing	LETA, {LETA2}, WETA
Halo/Pocket implant effects	LETA0
Edge Conduction	WEDGE, DGAMMAEDGE, DPHIEDGE
Geometrical Effects, Width scaling	Various Parameters (DL, WQLR, …)
Noise Flicker Noise, Short-Channel Thermal Noise, Induced Gate and Substrate Noise	LFNOI = 0 : KF, EF, AF (std model) LFNOI = 1: NT, ALPHAC, ALPHAH, KGFN, ECN, SDR (MC WORTHER – HOOGE – D/S ACCESS noise model) [1]
Temperature Effects	Various Parameters
TOTAL	226

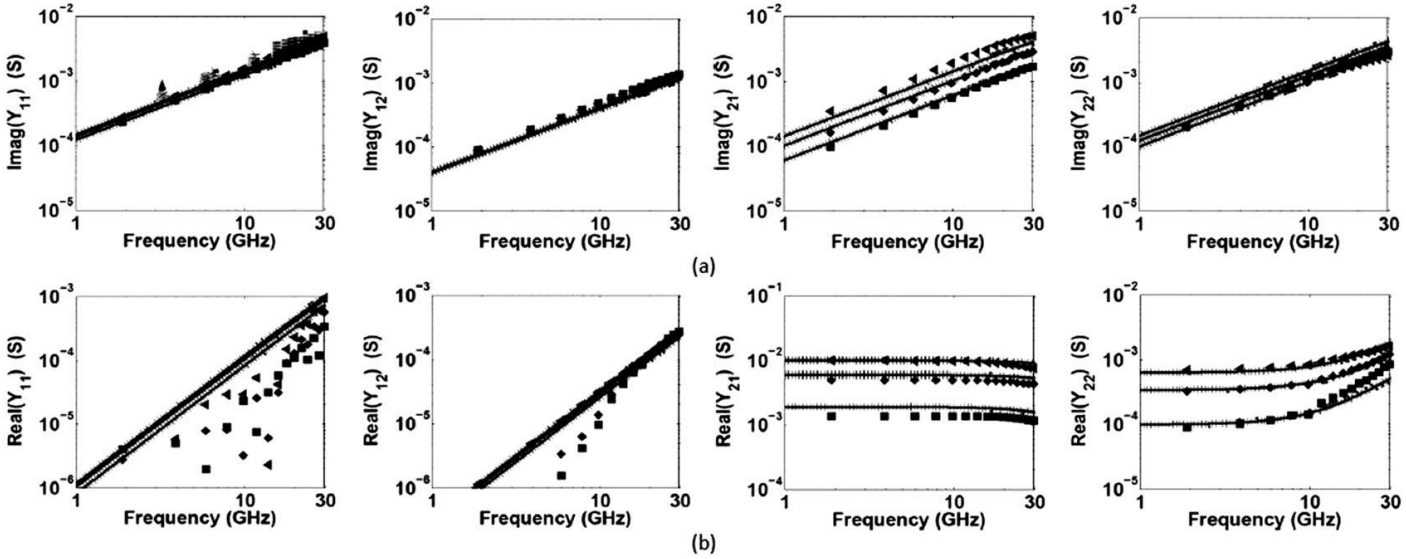
[1] N. Mavredakis, A. Antonopoulos and M. Bucher, "Measurement and modelling of 1/f noise in 180 nm NMOS and PMOS devices, " *Proceedings of Papers 5th European Conference on Circuits and Systems for Communications (ECCSC'10)*, 2010, pp. 86-89.

EKV3 in Simulators

- Using EKV3 model as:
  - embedded model in the simulator (Spectre, hpeesofsim, xyce,ngspice etc).
    - ✓ NGSPICE: Verilog-A -> C++ using ADMSXML & XML files
  - externally loaded Verilog-A code (include statement), compiled into a shared library during simulation time using ADMSXML. Used in Specte, hpeesofsim and QUCS (subversions).
  - precompiled external library (Spectre's CMI)
- Possible limitations and issues using ADMSXML:
  - Not all features supported (ex. NGSPICE does not support noise extraction)
  - Different simulators -> different XML files -> different Code interpretation -> Different problems

EKV3 in ngspice

- First time EKV3 (v. 301.02) in ngspice [1]
  - RF measurements vs model simulations



- Usage: Experimental and teaching

[1] Grabinski, W., Brinson, M., Nenzi, P., Lannutti, F., Makris, N., Antonopoulos, A. and Bucher, M. (2014), "Open-source circuit simulation tools for RF compact semiconductor device modelling. Int. J. Numer. Model., 27: 761-779. https://doi.org/10.1002/jnm.1973

EKV3 in ngspice

- EKV3 (v. 302.00) in ngspice v. 3.8
  - Compilation using ADMSXML
    - ✓ Create folder src/spicelib/devices/adms/ekv3/admsva
    - ✓ Copy Verilog –A code to admsva
    - ✓ Editing configure.ac :
      - Add **adms/ekv3** in VLADEVDIR
      - Add **#VLAMKF src/spicelib/devices/adms/ekv3/Makefile**
      - Add **/ekv3/ libekv3.la** w/ VLADEV
      - Remove entries of Verilog-A models that are not included
    - ✓ Editing XML files :
      - produce ekv3 C++ files (ngspiceVersion.xml)
      - ekv3noise.c in the compilation (ngspiceMakefile.am.xml , MODULEinit.c.xml)
    - ✓ Edit dev.c (get model info) , inpdomd.c (ekv3 as level=46 device) & inp2m.c (add ekv3 as type “m” device) files
- Noise experimental integration
  - Altering produced code using BSIM1 C++ code as sample
  - Recompile

EKV3 in ngspice

- Compilation in Linux Mint
  - Linux standalone executable
  - windows executable without GUI (Cross compilation)

- Python(3.10) and NGSPICE
  - JupyterLab
    - ✓ Quick and easy visualization of results
    - ✓ Easy data manipulation
  - Libraries
    - ✓ matplotlib (plotting)
    - ✓ pandas (dataset manipulation)
    - ✓ os (NGSPICE execution)
    - ✓ spyci (raw data import)

EKV3 simulations in JupyterLab

[6]:
```
Vgs=data[1][0][0]
Vds=[0.025,0.05,0.1,0.3,0.6,1.20]
ids=[]
for cnt in range(0,np.size(Vds)):
    ids.append(data[1][cnt][1])
```

- List of measured data

```
print(os.getcwd())

def idvg_sim(Vd,Vb,W,L):
    sim_idvd_text='''
    .include model_nmos_110v2.sp
    .option temp=25C
    * START SOURCES
    vd 1 0 DC '''+str(Vd)+'''
    vg 2 0 DC 1.2
    vb 3 0 DC '''+str(Vb)+'''
    mn0 1 2 0 3 nch W='''+W+''' L='''+L+'''
    .control
    dc vg -0.3 1.2 0.01
    plot -i(vd)
    write idvg_dc.raw all
    .endc
    .END
        '''
    with open(os.getcwd()+'/idvg_dc.sp', 'w') as f:
        f.write(sim_idvd_text)
    exec_file="ngspice.exe -b  idvg_dc.sp > output.log"
    os.system(exec_file)
    data = spyci.load_raw("idvg_dc.raw")
    df = pd.DataFrame(data['values'])
    return df
```

- Idvg_sim function:
  - Modify the simulation file
  - Write it to disk
  - Execute ngspice
  - Import data from output file

```
plt.figure()
for cnt in range(0,np.size(Vds)):
    plt.plot(Vgs,ids[cnt],'rs')
for Vd in Vds:
    df=idvg_sim(Vd,0.0,'3u','0.1u')
    plt.plot(df["v(2)"],-df["i(vd)"],'b')

plt.xlabel('$V_{GS}$ (V)')
plt.ylabel('$I_{DS}$ (A)')
plt.tight_layout()
```

- Simulate for different V_{DS}
- Plot model versus data

EKV3 simulations in JupyterLab

[6]:

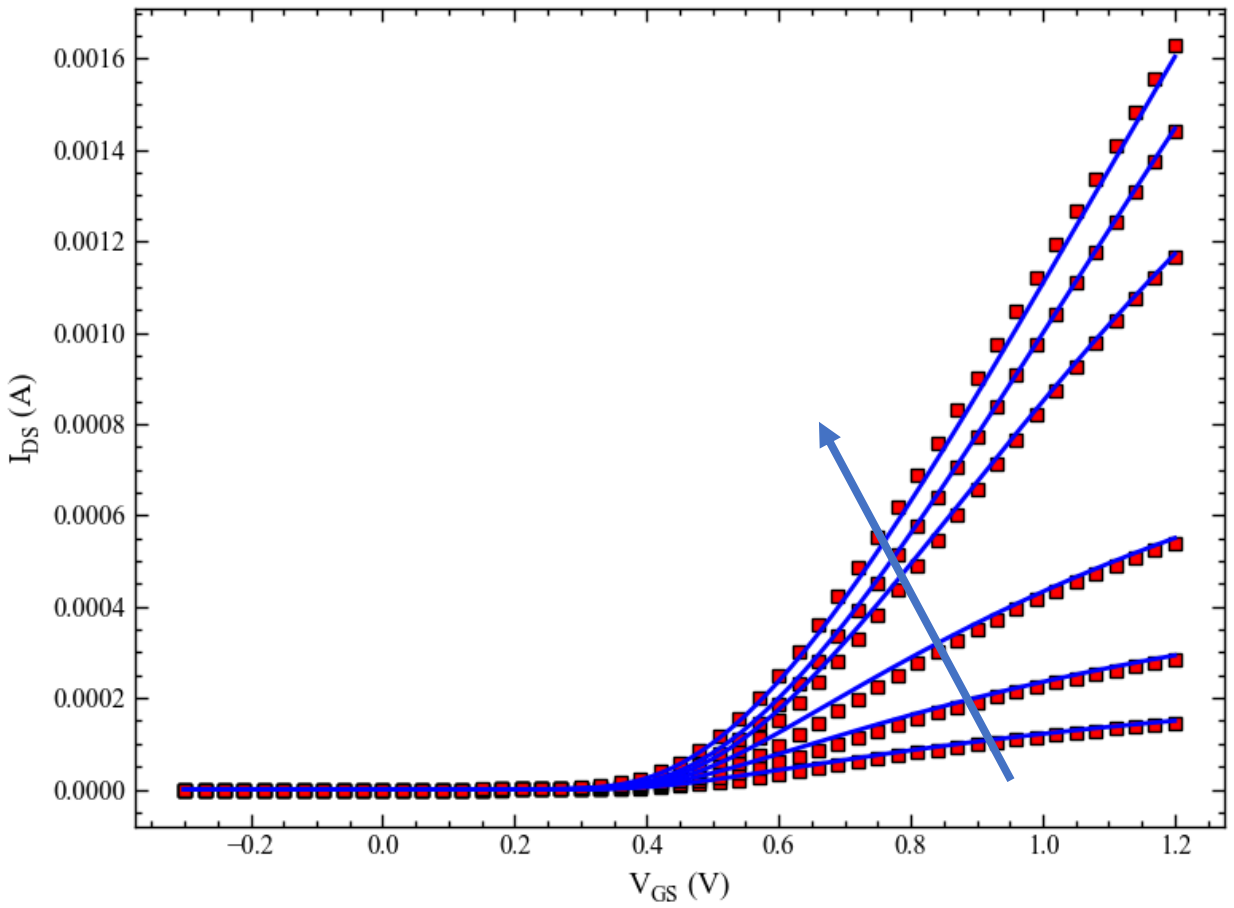
```
Vgs=data[1][0][0]
Vds=[0.025,0.05,0.1,0.3,0.6,1.20]
ids=[]
for cnt in range(0,np.size(Vds)):
    ids.append(data[1][cnt][1])

print(os.getcwd())

def idvg_sim(Vd,Vb,W,L):
    sim_idvd_text='''
    .include model_nmos_110v2.sp
    .option temp=25C
    * START SOURCES
    vd 1 0 DC '''+str(Vd)+'''
    vg 2 0 DC 1.2
    vb 3 0 DC '''+str(Vb)+'''
    mn0 1 2 0 3 nch W='''+W+''' L='''+L+'''
    .control
    dc vg -0.3 1.2 0.01
    plot -i(vd)
    write idvg_dc.raw all
    .endc
    .END
        '''
    with open(os.getcwd()+'/idvg_dc.sp', 'w') as f:
        f.write(sim_idvd_text)
    exec_file="ngspice.exe -b  idvg_dc.sp > output.log"
    os.system(exec_file)
    data = spyci.load_raw("idvg_dc.raw")
    df = pd.DataFrame(data['values'])
    return df


plt.figure()
for cnt in range(0,np.size(Vds)):
    plt.plot(Vgs,ids[cnt],'rs')
for Vd in Vds:
    df=idvg_sim(Vd,0.0,'3u','0.1u')
    plt.plot(df["v(2)"],-df["i(vd)"],'b')

plt.xlabel('$V_{GS}$ (V)')
plt.ylabel('$I_{DS}$ (A)')
plt.tight_layout()
```



EKV3 simulations in JupyterLab

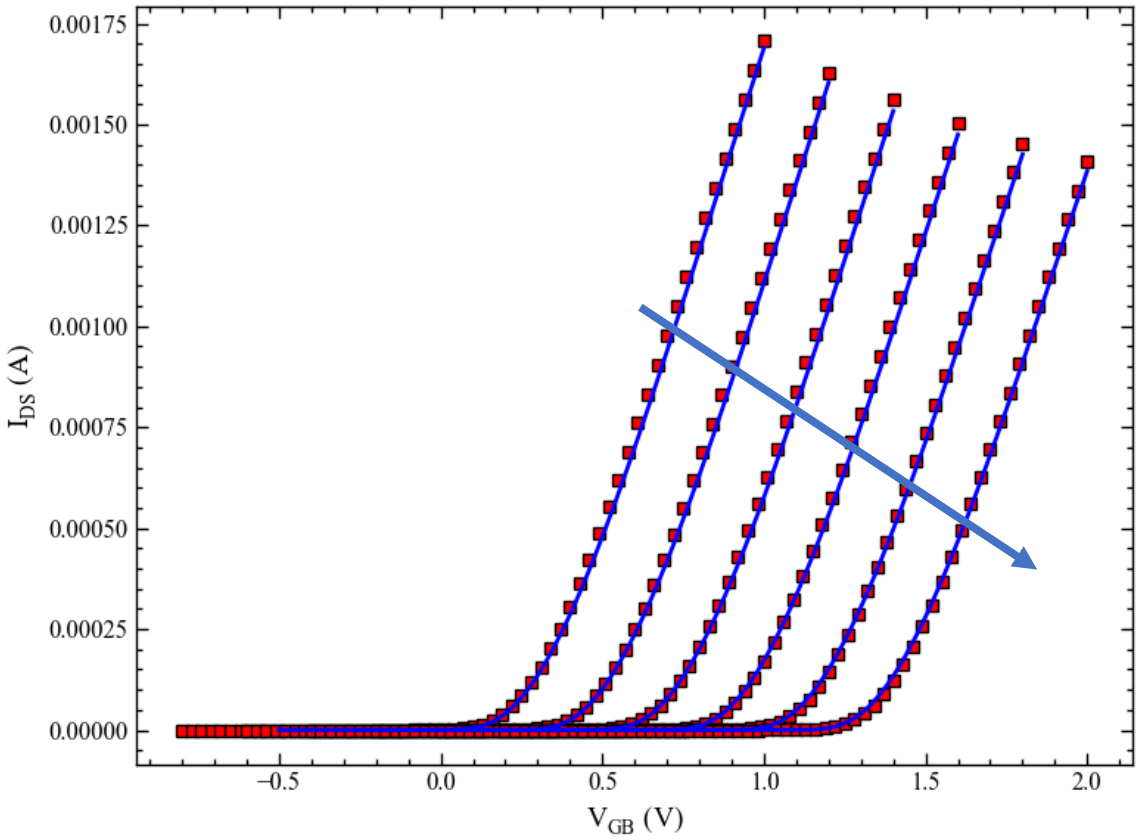
[12]:

```
Vbs=[0.2,0.0,-0.2,-0.4,-0.6,-0.8]
Vgs=data[1][0][0]
ids=[]
for cnt in range(0,np.size(Vbs)):
    ids.append(data[1][cnt][1])

plt.figure()
for cnt in range(0,np.size(Vbs)):
    plt.plot(Vgs-Vbs[cnt],ids[cnt],'rs')

for Vb in Vbs:
    df=idvg_sim(1.2,Vb,'3u','0.1u')
    plt.plot(df["v(2)"]-Vb,-df["i(vd)"],'b')


plt.xlabel('$V_{GB}$ (V)')
plt.ylabel('$I_{DS}$ (A)')
plt.tight_layout()
```



EKV3 simulations in JupyterLab

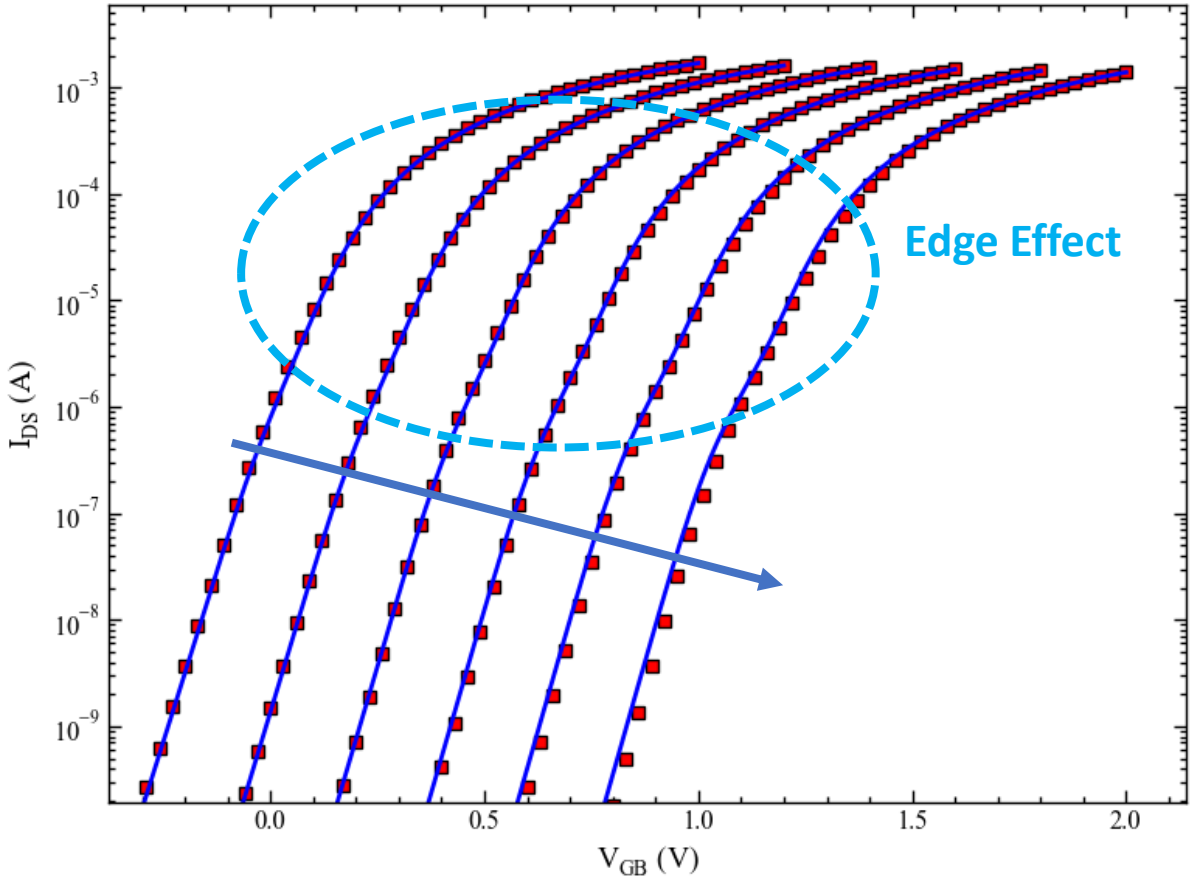
[10]:

```
Vbs=[0.2,0.0,-0.2,-0.4,-0.6,-0.8]
Vgs=data[1][0][0]
ids=[]
for cnt in range(0,np.size(Vbs)):
    ids.append(data[1][cnt][1])

plt.figure()
for cnt in range(0,np.size(Vbs)):
    plt.plot(Vgs-Vbs[cnt],ids[cnt],'rs')

for Vb in Vbs:
    df=idvg_sim(1.2,Vb,'3u','0.1u')
    plt.semilogy(df["v(2)"]-Vb,-df["i(vd)"],'b')


plt.xlabel('$V_{GB}$ (V)')
plt.ylabel('$I_{DS}$ (A)')
plt.tight_layout()
```



EKV3 simulations in JupyterLab

[16]:

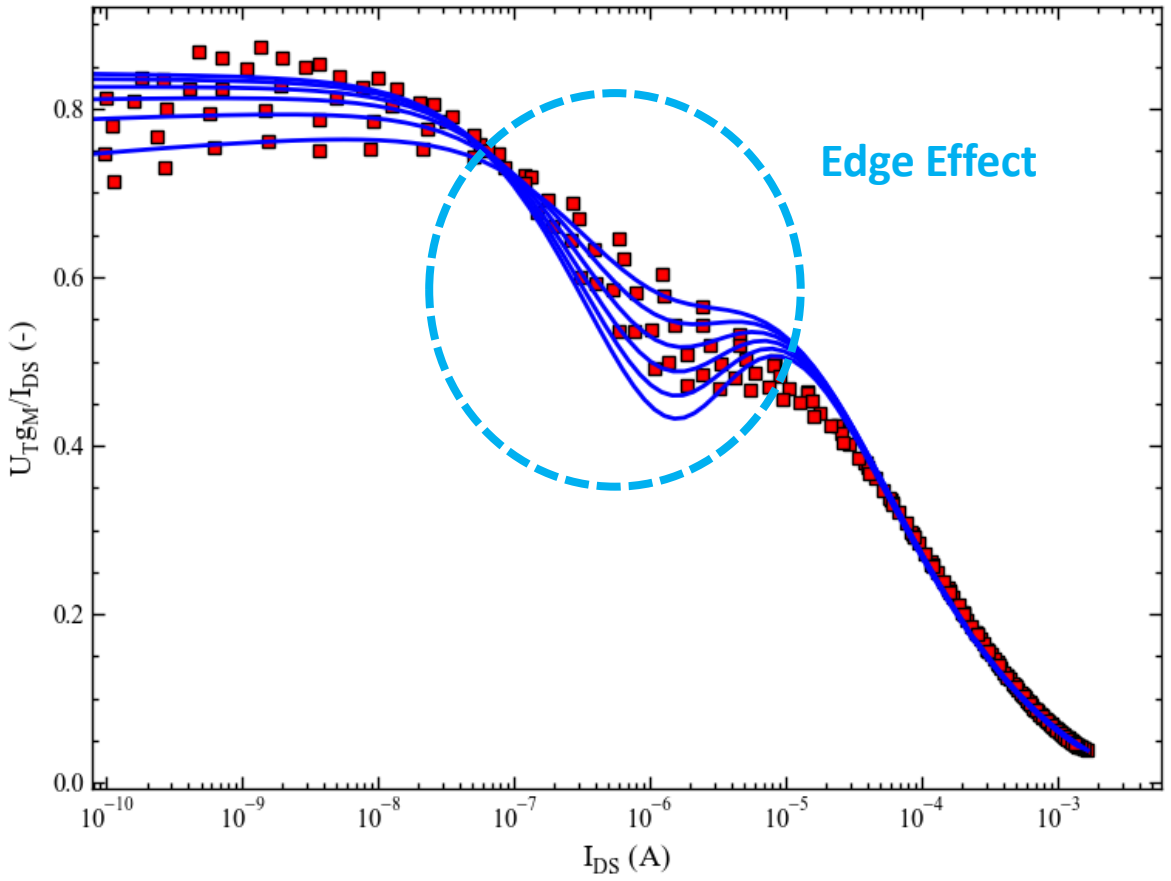
```
Vbs=[0.2,0.0,-0.2,-0.4,-0.6,-0.8]
Vgs=data[1][0][0]
ids=[]
for cnt in range(0,np.size(Vbs)):
    ids.append(data[1][cnt][1])

plt.figure()
for cnt in range(0,np.size(Vbs)):
    [gmidm,vgsm]=difflog_(Vgs,ids[cnt])
    plt.plot(vgsm,0.0258*gmidm,'rs')

for Vb in Vbs:
    df=idvg_sim(1.2,Vb,'3u','0.1u')
    [gmidm2,vgsm2]=difflog_(np.array(df["v(2)"]),np.array(-df["i(vd)"]))
    plt.semilogx(vgsm2,0.0258*gmidm2,'b')


plt.xlabel('$I_{DS}$ (A)')
plt.ylabel('$U_Tg_M/I_{DS}$ (-)')
plt.tight_layout()

```



EKV3 simulations in JupyterLab

```
Vgs=[0.2,0.4,0.6,0.8,1.0,1.2]
Vds=data[1][0][0]
ids=[]
for cnt in range(0,np.size(Vgs)):
    ids.append(data[1][cnt][1])
```

- List of measured data

```
def idvd_sim(Vg,Vb,W,L):
    sim_idvd_text='''
    .include model_nmos_110v2.sp
    .option temp=25C
    * START SOURCES
    vd 1 0 DC 1.0
    vg 2 0 DC '''+str(Vg)+'''
    vb 3 0 DC '''+str(Vb)+'''
    mn0 1 2 0 3 nch W='''+W+''' L='''+L+'''
    .control
    dc vd 0.0 1.2 0.01
    plot -i(vd)
    write idvd_dc.raw all
    .endc
    .END
        '''
    with open(os.getcwd()+'/idvd_dc.sp', 'w') as f:
        f.write(sim_idvd_text)
    exec_file="ngspice.exe -b  idvd_dc.sp > output.log"
    os.system(exec_file)
    data = spyci.load_raw("idvd_dc.raw")
    df = pd.DataFrame(data['values'])
    return df
```

- Idvd_sim function:
  - Modify the simulation file
  - Write it to disk
  - Execute ngspice
  - Import data from output file

```
plt.figure()
for cnt in range(0,np.size(Vgs)):
    plt.plot(Vds,ids[cnt],'rs')

for Vg in Vgs:
    df=idvd_sim(Vg,0.0,'3u','0.1u')
    plt.plot(df["v(1)"],-df["i(vd)"],'b')

plt.xlabel('$V_{DS}$ (V)')
plt.ylabel('$I_{DS}$ (A)')
plt.tight_layout()
```

- Simulate for different V_{GS}
- Plot model versus data

EKV3 simulations in JupyterLab

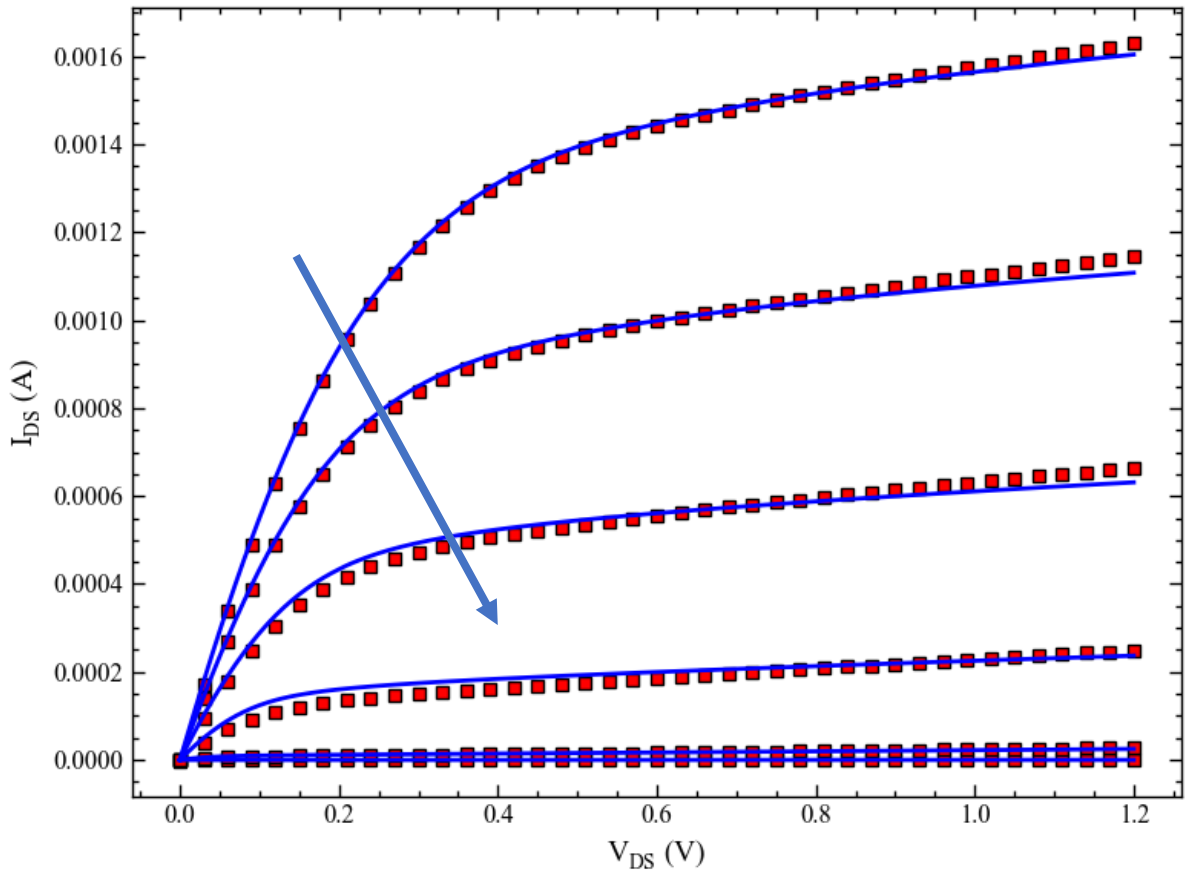
```
Vgs=[0.2,0.4,0.6,0.8,1.0,1.2]
Vds=data[1][0][0]
ids=[]
for cnt in range(0,np.size(Vgs)):
    ids.append(data[1][cnt][1])

def idvd_sim(Vg,Vb,W,L):
    sim_idvd_text='''
    .include model_nmos_110v2.sp
    .option temp=25C
    * START SOURCES
    vd 1 0 DC 1.0
    vg 2 0 DC '''+str(Vg)+'''
    vb 3 0 DC '''+str(Vb)+'''
    mn0 1 2 0 3 nch W='''+W+''' L='''+L+'''
    .control
    dc vd 0.0 1.2 0.01
    plot -i(vd)
    write idvd_dc.raw all
    .endc
    .END
        '''
    with open(os.getcwd()+'/idvd_dc.sp', 'w') as f:
        f.write(sim_idvd_text)
    exec_file="ngspice.exe -b  idvd_dc.sp > output.log"
    os.system(exec_file)
    data = spyci.load_raw("idvd_dc.raw")
    df = pd.DataFrame(data['values'])
    return df

plt.figure()
for cnt in range(0,np.size(Vgs)):
    plt.plot(Vds,ids[cnt],'rs')

for Vg in Vgs:
    df=idvd_sim(Vg,0.0,'3u','0.1u')
    plt.plot(df["v(1)"],-df["i(vd)"],'b')

plt.xlabel('$V_{DS}$ (V)')
plt.ylabel('$I_{DS}$ (A)')
plt.tight_layout()
```



Conclusions

- EKV3 is a robust charge based compact model used for many years in commercial PDKs covering a large range of phenomena.
- EKV3 has be embedded and used in commercial and open-source simulators
- EKV3 can be embedded in latest NGSPICE(3.8) using ADMSXML
- Special effort is needed for noise model integration
- The combination of NGSPICE and PYTHON in JupyterLab reveals an easy way to visualize and manipulate simulations
- EKV3 eventually becoming **opensource**

Thank you for your attention