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, gate current, fringing
        - Functionality: model variables & parameters, debug info, operating points info

A. Bazigos, PhD dissertation, 2008
## Phenomena covered by EKV3 (1/2)

<table>
<thead>
<tr>
<th>MODELED EFFECT</th>
<th>PARAMETERS / MODELING PRACTICE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physical Modelling of Charges</strong> (Accumulation Region Polysilicon Depletion, Quantum Mechanical Effects)</td>
<td>COX(TOX), PHIF, GAMMA(NSUB), VTO(VFB), GAMMAG(NGATE)</td>
</tr>
<tr>
<td><strong>Bias-Dependent Overlap Capacitances</strong></td>
<td>LOV, GAMMAOV(NOV), VFBOV</td>
</tr>
<tr>
<td><strong>NQS</strong></td>
<td>Channel segmentation</td>
</tr>
<tr>
<td><strong>RF</strong></td>
<td>External subcircuit (RG,RSUB w/ scaling)</td>
</tr>
<tr>
<td><strong>Mobility</strong> (Reduction due to Vertical Field Effect) Surface Roughness-, Phonon-, Coulomb Scattering</td>
<td>KP(U0), E0, E1, ETA, ZC, THC</td>
</tr>
<tr>
<td><strong>Impact Ionization Current</strong></td>
<td>IBA, IBB, IBN</td>
</tr>
<tr>
<td><strong>Gate Current</strong></td>
<td>(IGS, IGD, IGB) KG, XB, UB</td>
</tr>
<tr>
<td><strong>Lateral Field Effect</strong> (VS,CLM)</td>
<td>UCRIT(VSAT), LAMBDA, DELTA</td>
</tr>
<tr>
<td><strong>Reverse Short Channel Effect</strong></td>
<td>LR, QLR, NLR</td>
</tr>
</tbody>
</table>
## Phenomena covered by EKV3 (2/2)

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

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 Spectre, 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

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 VLADIR
      • Add #VLAMKF
        src/spicelib/devices/adms/ekv3/Mak file
      • Add /ekv3/ libekv3.la w/ VLAVELV
      • 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

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

- Simulate for different $V_{DS}$
- Plot model versus data
EKV3 simulations in JupyterLab

```python
1 Vgs=data[3][0][3]
2 Vds=[0.025,0.05,0.1,0.15,0.2]
3 Ids=[]
4 for cnt in range(len(psp_sim(Vds))):
5     Ids.append(data[1][cnt][1])
6     print(os.getcwd())

9     def idvg_sim(Vd,Vb,W,L):
10         sile_idvd_text='''
11             .include model_mos_nsp110v2.sp
12             .option temp=25C
13             " START SOURCES
14             vd 1 0 0 %str(Vd)%
15             vg 1 0 1.2
16             vb 0 0 dc "%str(Vb)%"
17             w=1 l=%str(L)"
18             .control
19             dc vg -0.3 1.2 0.4
20             plot -i(vd)
21             write idvg_dc.raw all
22             .endc
23             .RD
24             ...'''
25             with open(os.getcwd()+"/idvg_dc.sp",'w') as f:
26                 f.write(sile_idvd_text)
27             exec_file ="mpspice.exe -b idvg_dc.sp > output.log"
28             os.system(exec_file)
29             data = spc1.idvd_read("idvg_dc.raw")
30             df = pd.DataFrame(data['idvg'])
31             return df
32     ...
33 plt.figure()
34 for cnt in range(len(psp_sim(Vds))):
35     plt.plot(Vgs,Id[s][cnt],'r-')
36     for Vd in Vds:
37         df=idvg_sim(Vd,0.5,'0.1','0.2')
38     plt.plot(df['Vd'], df['Id'], 'b-')
39 plt.xlabel('$V_D$ (V)')
40 plt.ylabel('$I_D$ (A)')
41 plt.tight_layout()
```
EKV3 simulations in JupyterLab

```python
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=ivg_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_{GS} (A)')
plt.tight_layout()
```
EKV3 simulations in JupyterLab

```python
[10]:
1  Vbs=[0.2,0.0,-0.2,-0.4,-0.6,-0.8]
2  Vgs=data[1][0][0]
3  ids=[]
4  for cnt in range(0,np.size(Vbs)):
5      ids.append(data[1][cnt][1])
6  plt.figure()
7  for cnt in range(0,np.size(Vbs)):
8      plt.plot(Vgs-Vbs[cnt],ids[cnt],'rs')
9  for Vb in Vbs:
10     df=idvg_sim(1.2,Vb,'3u','0.1u')
11     plt.semilogy(df['v(2)']::Vb,-df['i(2d)'],'b')
12
13  plt.xlabel('$V_{GB}$ (V)')
14  plt.ylabel('$I_{DS}$ (A)')
15  plt.tight_layout()
```

Edge Effect
EKV3 simulations in JupyterLab

```python
Vbs=[0.2,0.6,0.2,0.4,0.6,0.8]
Vgs=data[1][0][0]
ids=[]
for cnt in range(len(Vbs)):
    ids.append(data[1][cnt][1])
plt.figure()
for cnt in range(len(Vbs)):
    [gm,vgsm]=difflog(Vgs,ids[cnt])
    plt.plot(vgsm,0.0258*gm,'g')
for Vb in Vbs:
    df=idsim(1.2,Vb,0.1,'0.1u')
    [gs,m2]=difflog(np.array(df['v(2)']),np.array(df['i(vd)']))
    plt.semilogx(vgsm,0.0258*gm2,'b')
plt.xlabel('U_{T(SM)} (A)')
plt.ylabel('U_{T(SM)} (A)')
plt.tight_layout()
```
EKV3 simulations in JupyterLab

- List of measured data

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

- Simulate for different $V_{GS}$

- Plot model versus data

```python
Vgs=[0.2, 0.4, 0.6, 0.8, 1.0, 1.2]
Vds=data[2][0][0]
Ids=[]
for vgs in range(0,mp.size(Vgs)):
    Ids.append(data[1][contx])

def idvd_sim(vgs,Vd,Ids):
    sim_idvd_text=''
    .include model_mos_iidv2.sp
    .option temp=25C
    .start SOURCES
    vd 1 0 dc 1.0
    vg 1 0 dc 0
    .control
    dc vd 0.0 1.2 0.02
    plot -syn
    write idvd_dc.raw all
    .end
    ...
    with(open(os.getcwd(),:'/idvd_dc.sp', 'w')) as fi:
        fi.write(sim_idvd_text)
    exec_file="ngspice.exe -b idvd_dc.sp > output.log"
    os.system(exec_file)
    data = mpg.load_raw("idvd_dc.raw")
    df = pd.DataFrame(data['values'])
    return df

plt.figure()
for vgs in range(0,mp.size(Vgs)):
    plt.plot(vgs,Ids[contx], 'm')
for vgs in Vgs:
    df=idvd_sim(vgs,Vd,Ids)
    plt.plot(df['vgs'], df['(vds)'], 'b')
plt.xlabel('Vd (GS) $\text{V}$')
plt.ylabel('Id (GS) $\text{mA}$')
plt.tight_layout()
```
EKV3 simulations in JupyterLab

```python
Vgs=[0,1,0,4,0,5,6,8,1,2,1,2]
Vds=data[2][0][0]
Ids=[]
for vgs in range(0,mp.size(Vgs)):
    Ids.append(data[3][cnt][1])

def idvds(x,y,vb,xj):
    sm_idvd_text=''
    sm_idvd_text+='
    include model_mos_ii8v2.sp
    .option temp=25C
    .begin SOURCES
    vd 0 0 1.0
    vg 2 0 DC 0
    g 0 0 DC 0
    m 2 0 3 nch V="" x="" L="" L=""
    .end
    .control
dc vd 0.0 1.2 0.02
plot -i(vds)
write idvd_dc.raw all
.end
.with open(open('idvd_dc.dat', 'w')) as fi:
    f.write('Vgs ^Vds ^Idvds
    exec_file="mspice.exe -b idvd_dc.sp > output.log"
    os.system('exec_file')
data = pd.DataFrame(data['values'])
df = pd.DataFrame(data['values'])
return df
plt.figure()
for vgs in range(0,mp.size(Vgs)):
    plt.plot(Vgs,Ids[cnt][1])
for vg in Vgs:
    df-idvds(vg,0.0,'0.0','0.1')
plt.plot(df['(vB)'][1], df['(vD)'][1])
plt.xlabel('Vg (OS) (V)')
plt.ylabel('Ids (OS) (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 open source
Thank you for your attention