Automatic Generation of Compact Semiconductor Device Models using Paragon and ADMS

*Vivek Chaudhary, Matt Francis, Wei Zheng, Alan Mantooth, Laurent Lemaitre**

Mixed Signal CAD Laboratory
Department of Electrical Engineering
University of Arkansas

* Freescale Semiconductor
Geneva Switzerland

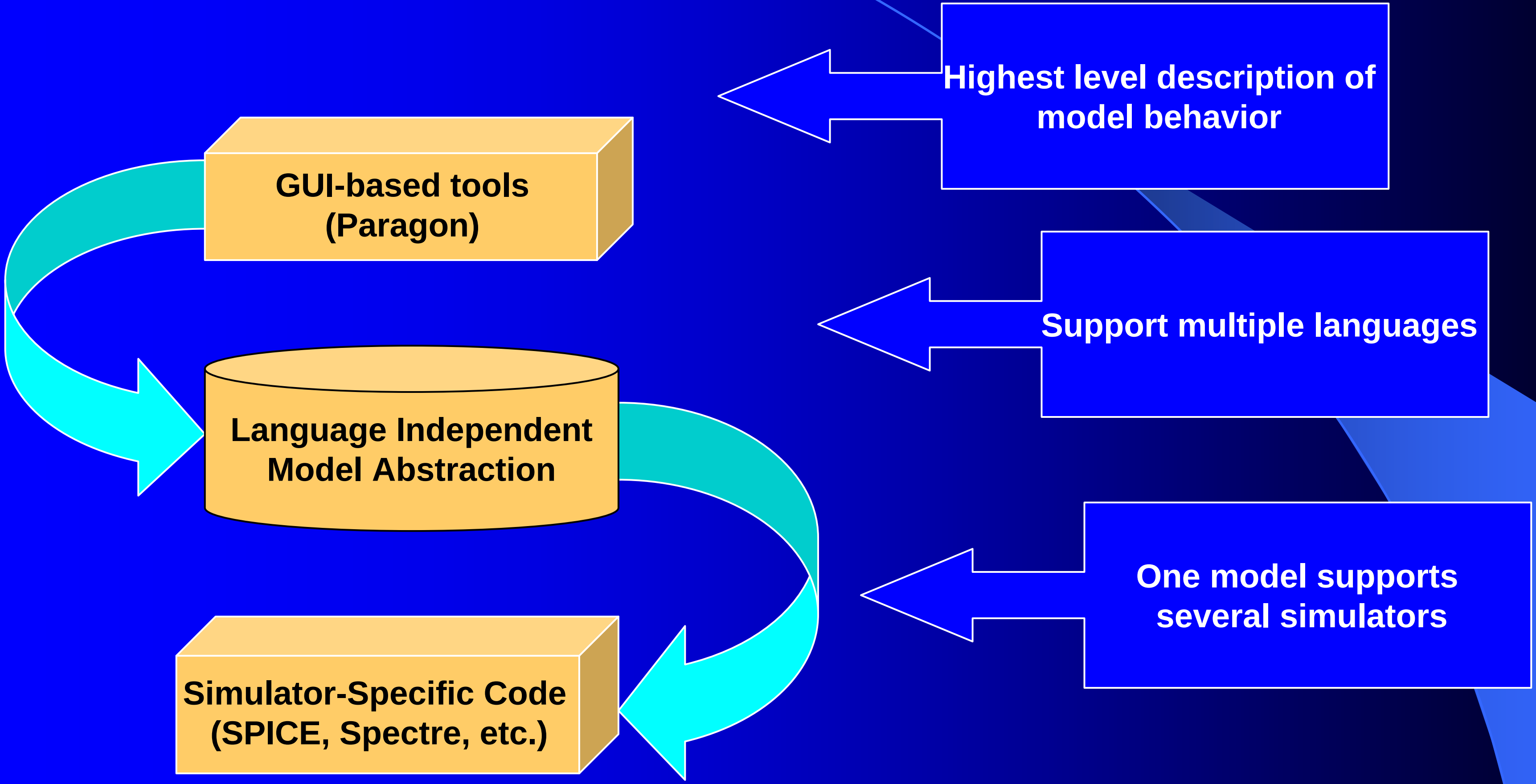
Outline

- Introduction
- Abstract Model Representation
- XML Schema
- Modeling Methodology
- Model Compilation
- BSIMSOI Model
- Conclusion

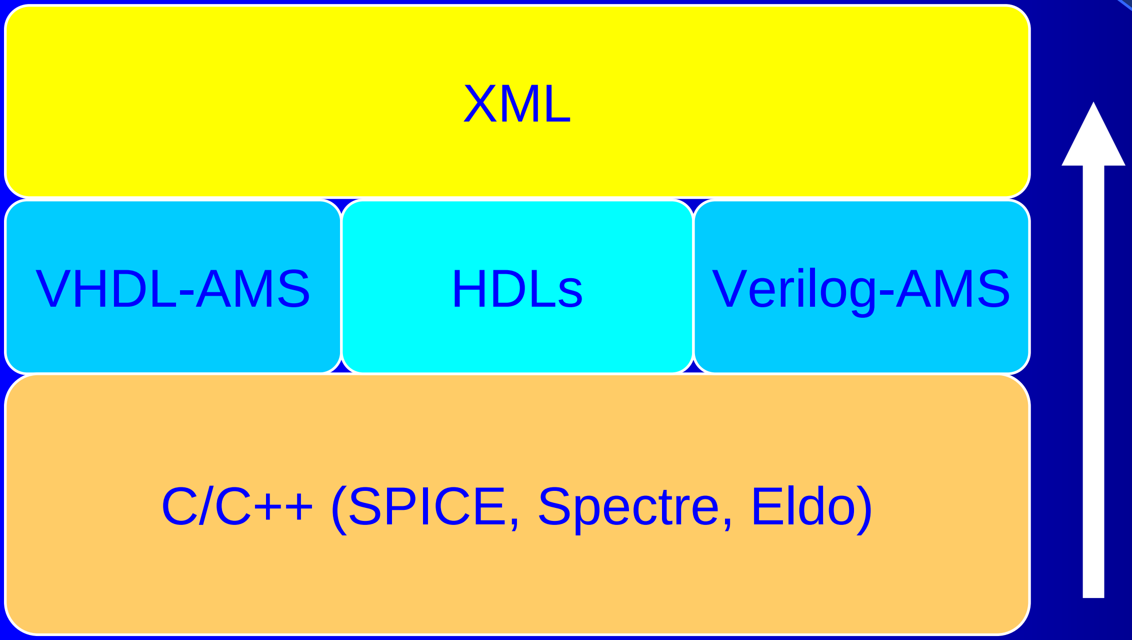
Introduction

- Semiconductor device modeling is extremely time consuming
- Model implementation in SPICE-like simulators is cumbersome
- Designers from different scientific backgrounds do not use the same analysis tools or same modeling languages

Various Levels of Abstraction


GUI-based tools (Paragon)
Language Independent Model Abstraction
Simulator-Specific Code (SPICE, Spectre, etc.)
Highest level description of model behavior
Support multiple languages
One model supports several simulators

Model Representation



Goals for XML format:

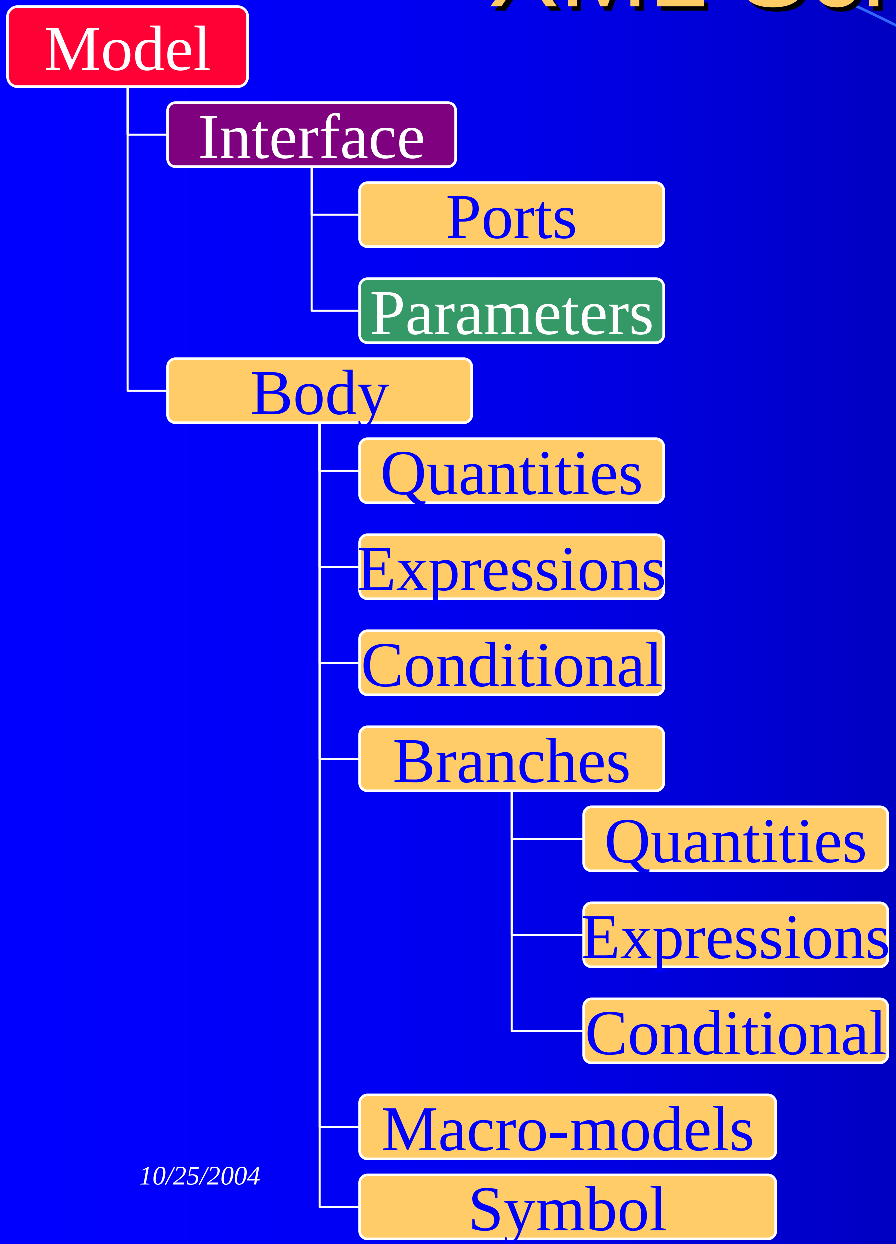
- Simulator independent
- Higher than HDLs
- Captures semiconductor device physics and multi-physics (electro-thermal, etc.)

Why XML?

XML:

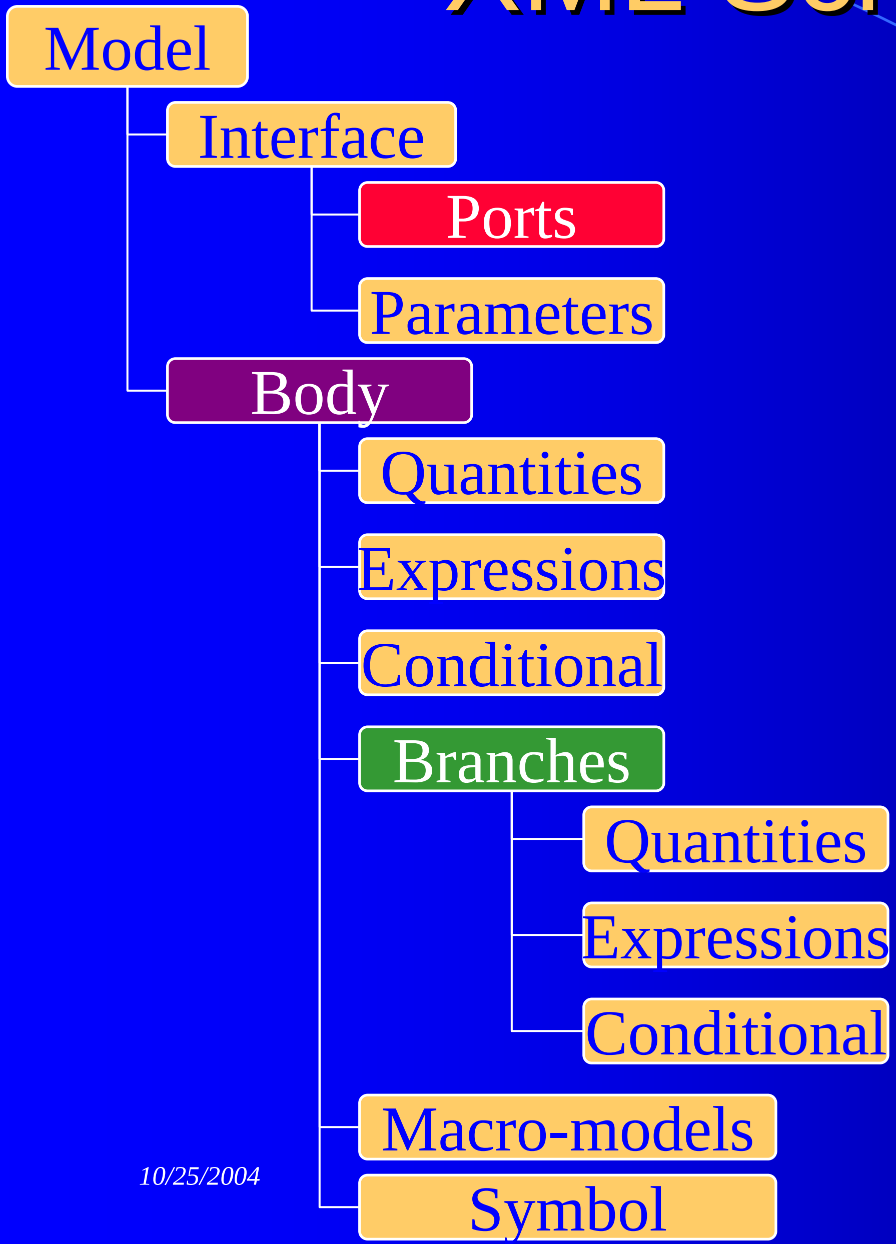
- Superset of HTML, simple/structured format
- Lends itself to open source and standardization
- Much XML-based technology already exists
  - Example: MathML, used for model expressions, is becoming *de facto* standard for math on the web
- Extensible
- Self-documenting

XML Schema-DTD



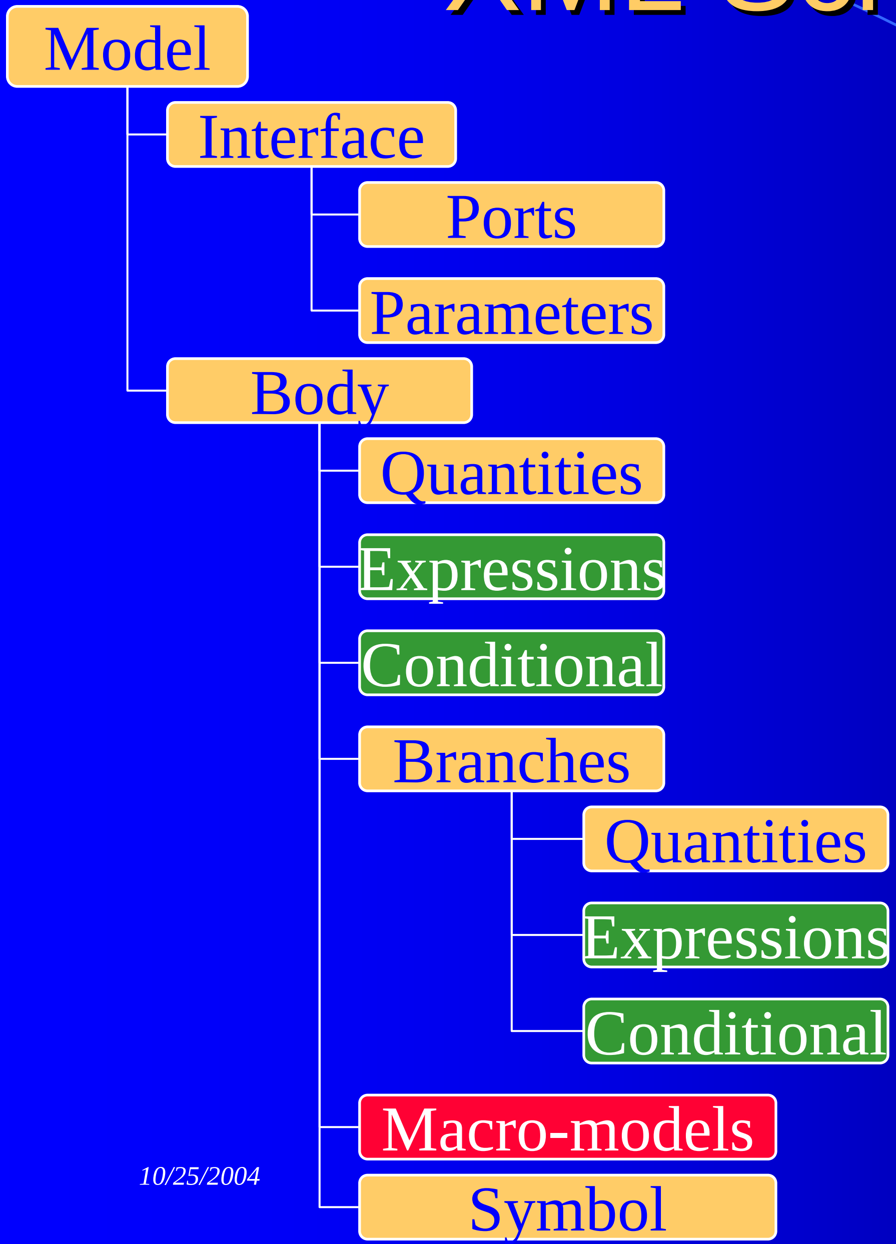
```
<?xml version="1.0"?>
<!-- DTD -->
<!-- model declarations -->
<!ELEMENT model (comment?, interface, body+)>
<!ATTLIST model
            name CDATA #REQUIRED
            version CDATA #REQUIRED
>
<!ELEMENT comment (#PCDATA)>
<!-- model interface -->
<!ELEMENT interface (comment?, parameter*, port*)>
<!-- model parameters -->
<!ELEMENT parameter (comment?, validity*)>
<!ATTLIST parameter
            default CDATA #IMPLIED
            name CDATA #REQUIRED
            nature (real | integer | time) #REQUIRED
            unit CDATA #REQUIRED
            type (instance | process) #REQUIRED
>
<!-- parameter validity -->
<!ELEMENT validity (range+)>
<!ATTLIST validity
            message CDATA #REQUIRED
            type (note | error | warning) #REQUIRED
>
<!ELEMENT range EMPTY>
<!-- range of validity -->
<!ATTLIST range
            min CDATA #REQUIRED
            max CDATA #REQUIRED
            exclude_max (yes | no) #IMPLIED
            exclude_min (yes | no) #IMPLIED
>
```

XML Schema-DTD



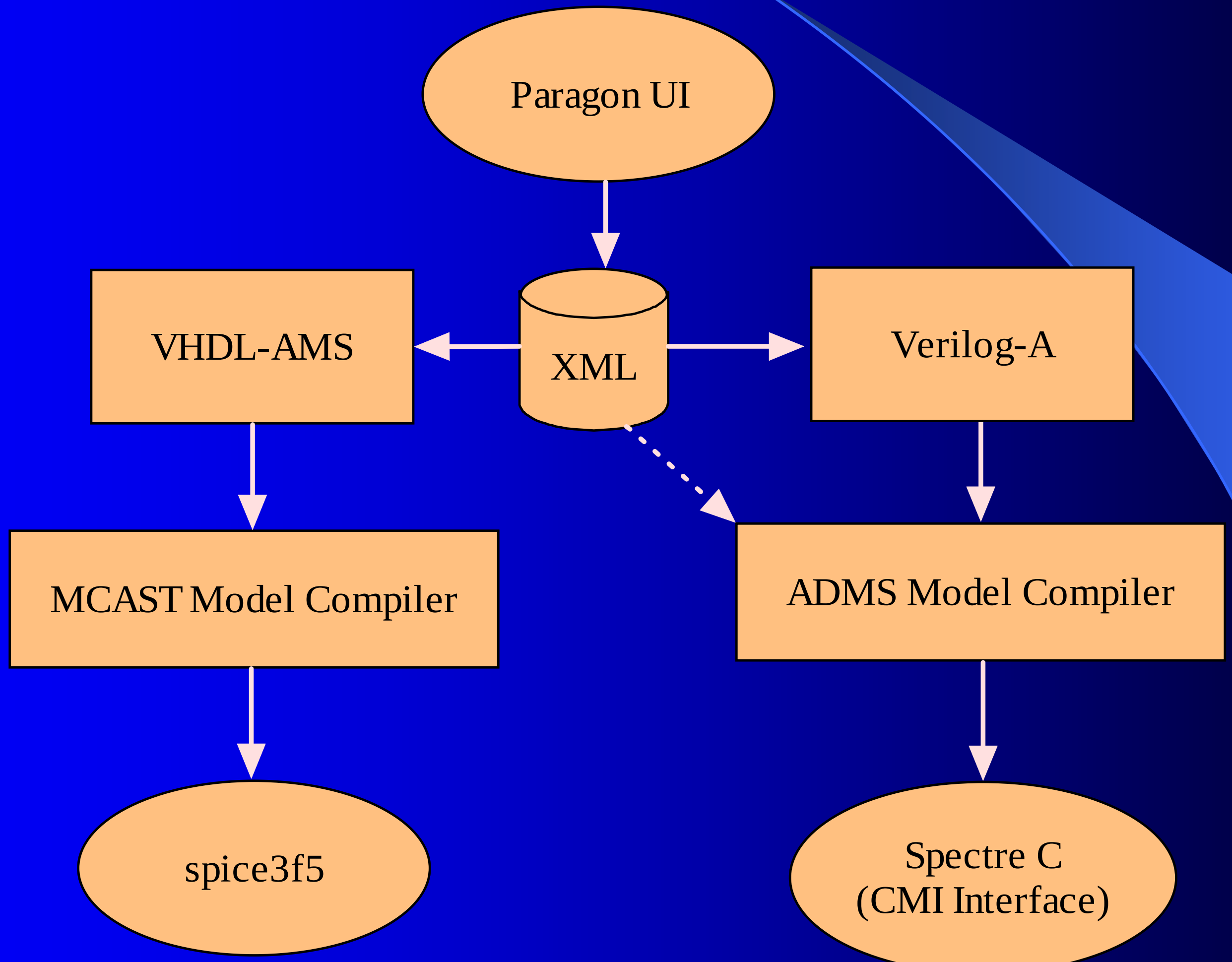
```
<!-- model interface ports -->
<!ELEMENT port (comment?)>
<!ATTLIST port
                name CDATA #REQUIRED
                mode (in | out | inout) #REQUIRED
                nature (electrical | mechanical_angular_speed |
mechanical_angular_displacement |
mechanical_translational_speed |
mechanical_translational_displacement | thermal | optical |
magnetic) #REQUIRED
                type (terminal | quantity | signal | logic)
#REQUIRED
>
<!-- end of model interface declaration -->
<!-- model body -->
<!ELEMENT body (branch*, equation*, piecewise*, eqblock*,
macromodel*)>
<!ATTLIST body
                name CDATA #REQUIRED
>
<!-- model body symbol -->
<!ELEMENT symbol (vector_graphics)>
<!-- model body branch -->
<!ELEMENT branch (connection+, quantity+, equation*,
piecewise*, eqblock*, comment?)>
<!ATTLIST branch
                name CDATA #IMPLIED
>
<!-- model body branch quantity -->
<!ELEMENT quantity EMPTY>
<!ATTLIST quantity
                name CDATA #REQUIRED
                nature (through | across) #REQUIRED
                type CDATA #IMPLIED
                unit CDATA #IMPLIED
>
```

XML Schema-DTD



```
<!-- model body macromodel -->
<!ELEMENT macromodel (connection*, comment?, macromodel.parameter,
macromodel.architecture)>
<!ATTLIST macromodel
                name CDATA #REQUIRED
                filename CDATA #REQUIRED
>
<!ELEMENT macromodel.parameter EMPTY>
<!ATTLIST macromodel.parameter
                name CDATA #REQUIRED
                value CDATA #REQUIRED
>
<!ELEMENT macromodel.architecture EMPTY>
<!ATTLIST macromodel.architecture
                name CDATA #REQUIRED
>
<!-- model body branch/macromodel connection -->
<!ELEMENT connection EMPTY>
<!ATTLIST connection
                name (pos | neg | CDATA) #REQUIRED
                mappedto CDATA #REQUIRED
                type (port | param) #REQUIRED
>
<!-- model/branch equation blocks -->
<!ELEMENT eqblock (equation*, piecewise*, comment*)>
<!ATTLIST eqblock
                nature (simultaneous | sequential) #REQUIRED
>
<!ELEMENT piecewise (if, elseif*, else, comment*)>
<!ELEMENT if (condition, equation+, piecewise*, comment*)>
<!ELEMENT elseif (condition, equation+, piecewise*, comment*)>
<!ELEMENT else (condition, equation+, piecewise*, comment*)>
<!ELEMENT condition (math)>
<!-- model/branch equations -->
<!ELEMENT equation (math)>
<!ATTLIST equation
                type (sequential | simulataneous | conditional) #REQUIRED
```

Modeling Methodology



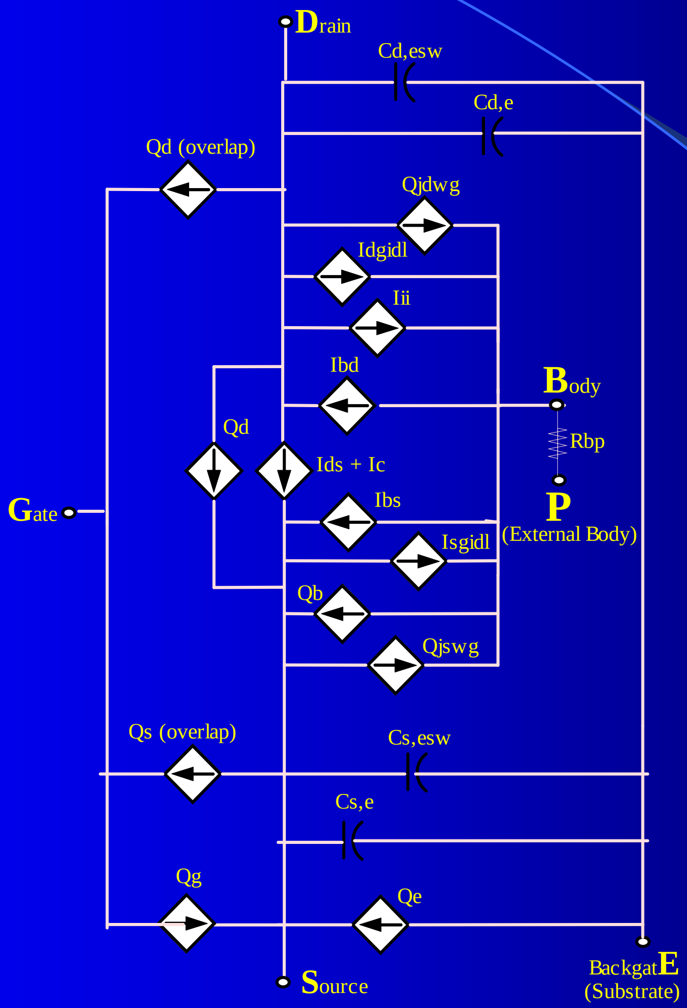
Model Compilation

- Model compilation is automatic generation of compact semiconductor device models for various SPICE like simulators
- ADMS is a freely available model compiler for SPICE-like simulators

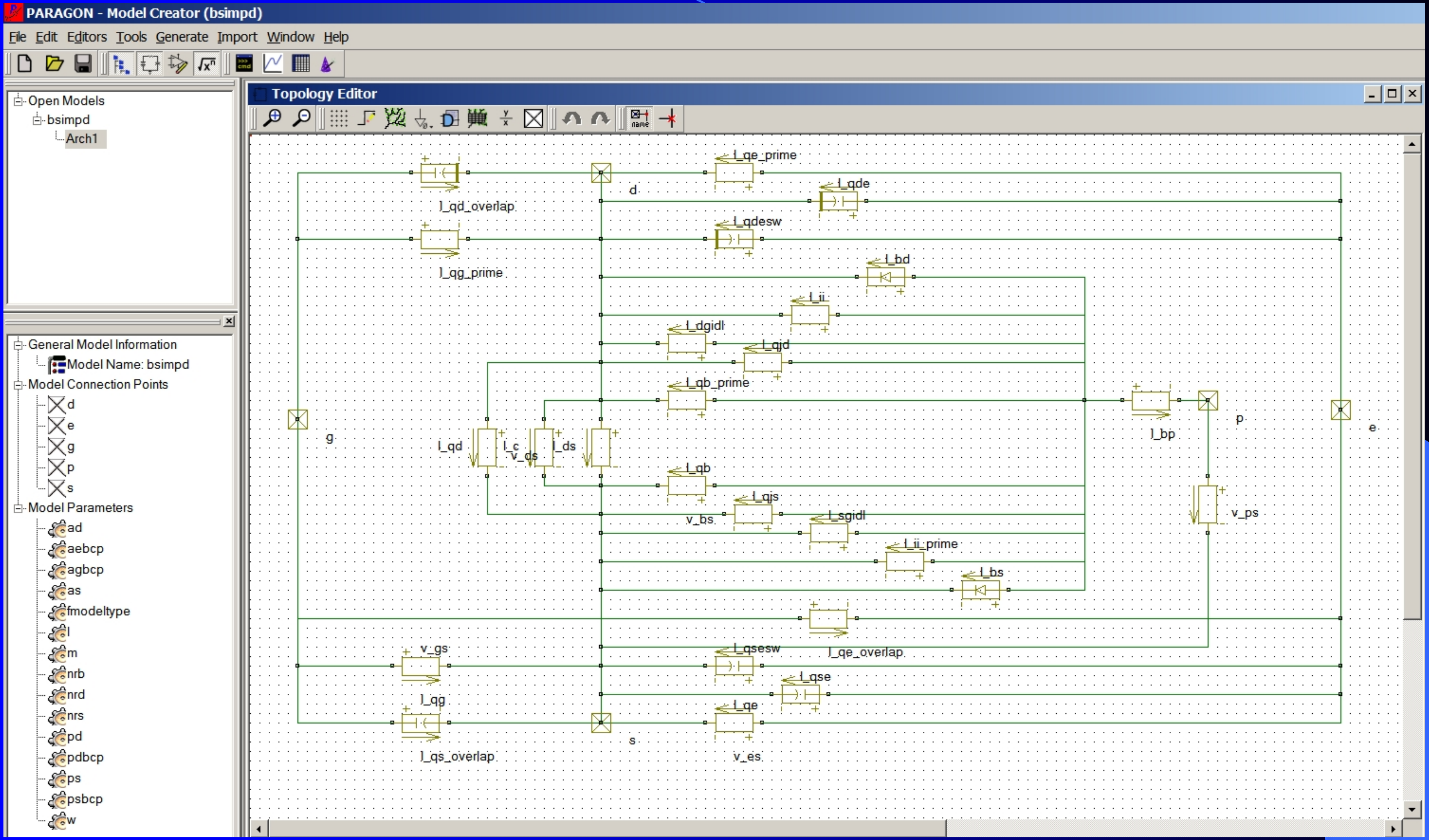
Advantages

- Model development time is dramatically reduced
- Generated model is easier to maintain and reuse
- Same abstract representation of the model can be used by a model compiler to generate low level C/C++ code for different simulators

Large-Signal Model of BSIMSOI



Paragon Implementation



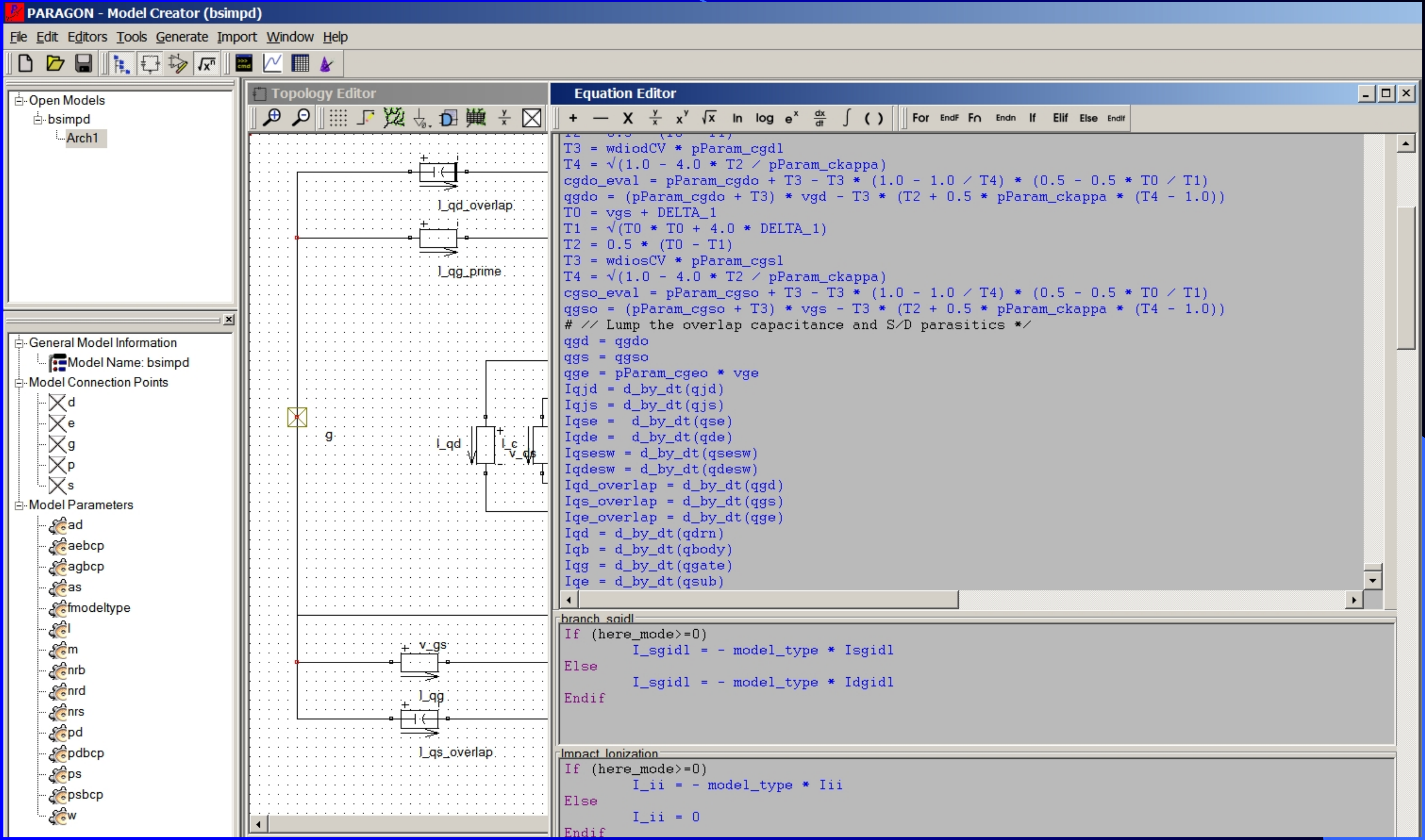
Open Models
- bsimpd
  - Arch1

General Model Information
- Model Name: bsimpd

Model Connection Points
- d
- e
- g
- p
- s

Model Parameters
- ad
- aebcp
- agbcp
- as
- fmodeltype
- l
- m
- nrb
- nrd
- nrs
- pd
- pdbcp
- ps
- psbcp
- w

Paragon Implementation



```
T3 = wdiodCV * pParam_cgdl
T4 = √(1.0 - 4.0 * T2 / pParam_ckappa)
cgdo_eval = pParam_cgdo + T3 - T3 * (1.0 - 1.0 / T4) * (0.5 - 0.5 * T0 / T1)
qgdo = (pParam_cgdo + T3) * vgd - T3 * (T2 + 0.5 * pParam_ckappa * (T4 - 1.0))
T0 = vgs + DELTA_1
T1 = √(T0 * T0 + 4.0 * DELTA_1)
T2 = 0.5 * (T0 - T1)
T3 = wdiosCV * pParam_cgsl
T4 = √(1.0 - 4.0 * T2 / pParam_ckappa)
cgso_eval = pParam_cgso + T3 - T3 * (1.0 - 1.0 / T4) * (0.5 - 0.5 * T0 / T1)
qgso = (pParam_cgso + T3) * vgs - T3 * (T2 + 0.5 * pParam_ckappa * (T4 - 1.0))
# // Lump the overlap capacitance and S/D parasitics */
qgd = qgdo
qgs = qgso
qge = pParam_cgeo * vge
Iqjd = d_by_dt(qjd)
Iqjs = d_by_dt(qjs)
Iqse =  d_by_dt(qse)
Iqde =  d_by_dt(qde)
Iqsesw = d_by_dt(qsesw)
Iqdesw = d_by_dt(qdesw)
Iqd_overlap = d_by_dt(qgd)
Iqs_overlap = d_by_dt(qgs)
Iqe_overlap = d_by_dt(qge)
Iqd = d_by_dt(qdrn)
Iqb = d_by_dt(qbody)
Iqg = d_by_dt(qgate)
Iqe = d_by_dt(qsub)
```

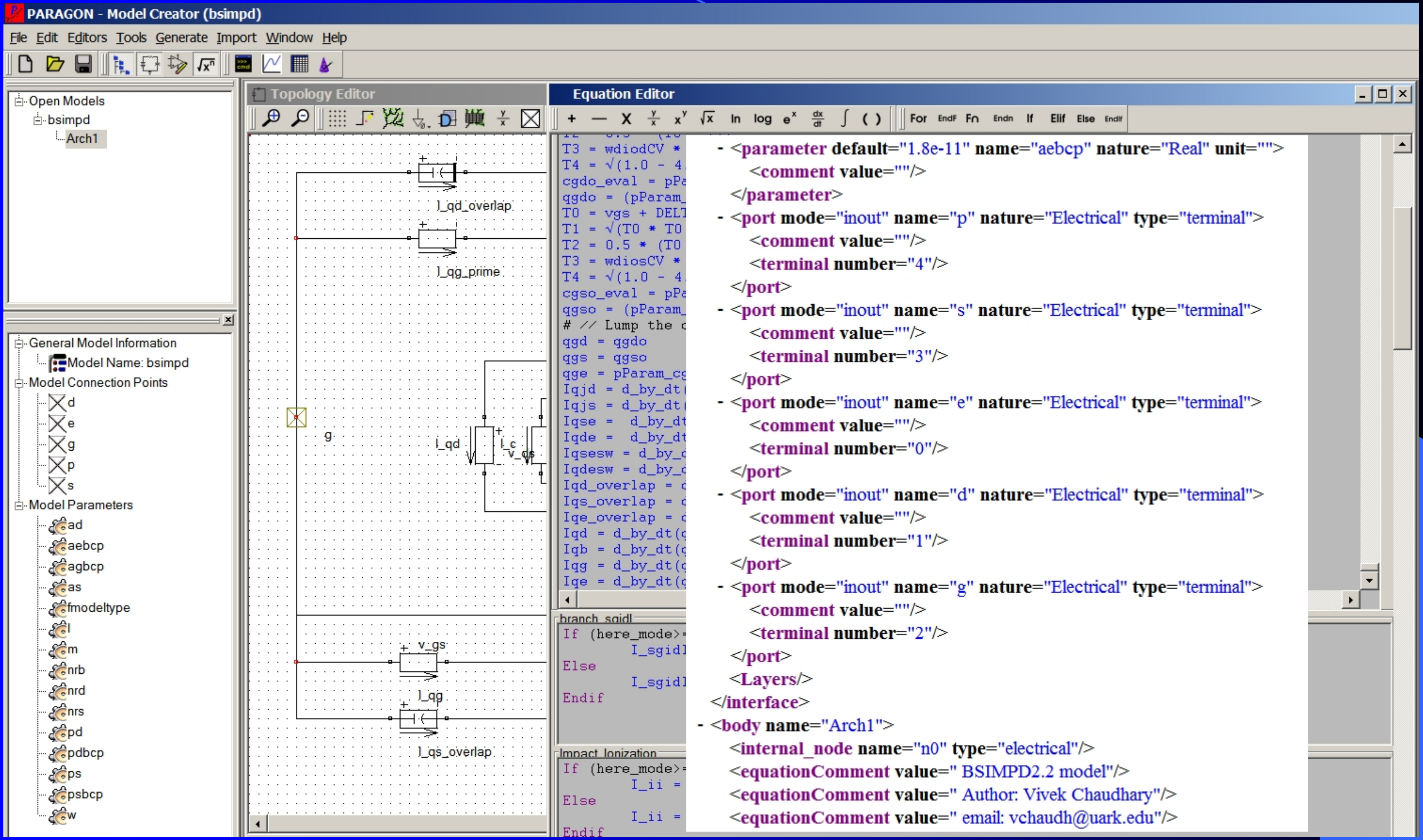
branch_sgidl

```
If (here_mode>=0)
        I_sgidl = - model_type * Isgidl
Else
        I_sgidl = - model_type * Idgidl
Endif
```

Impact Ionization

```
If (here_mode>=0)
        I_ii = - model_type * Iii
Else
        I_ii = 0
Endif
```

Paragon Implementation



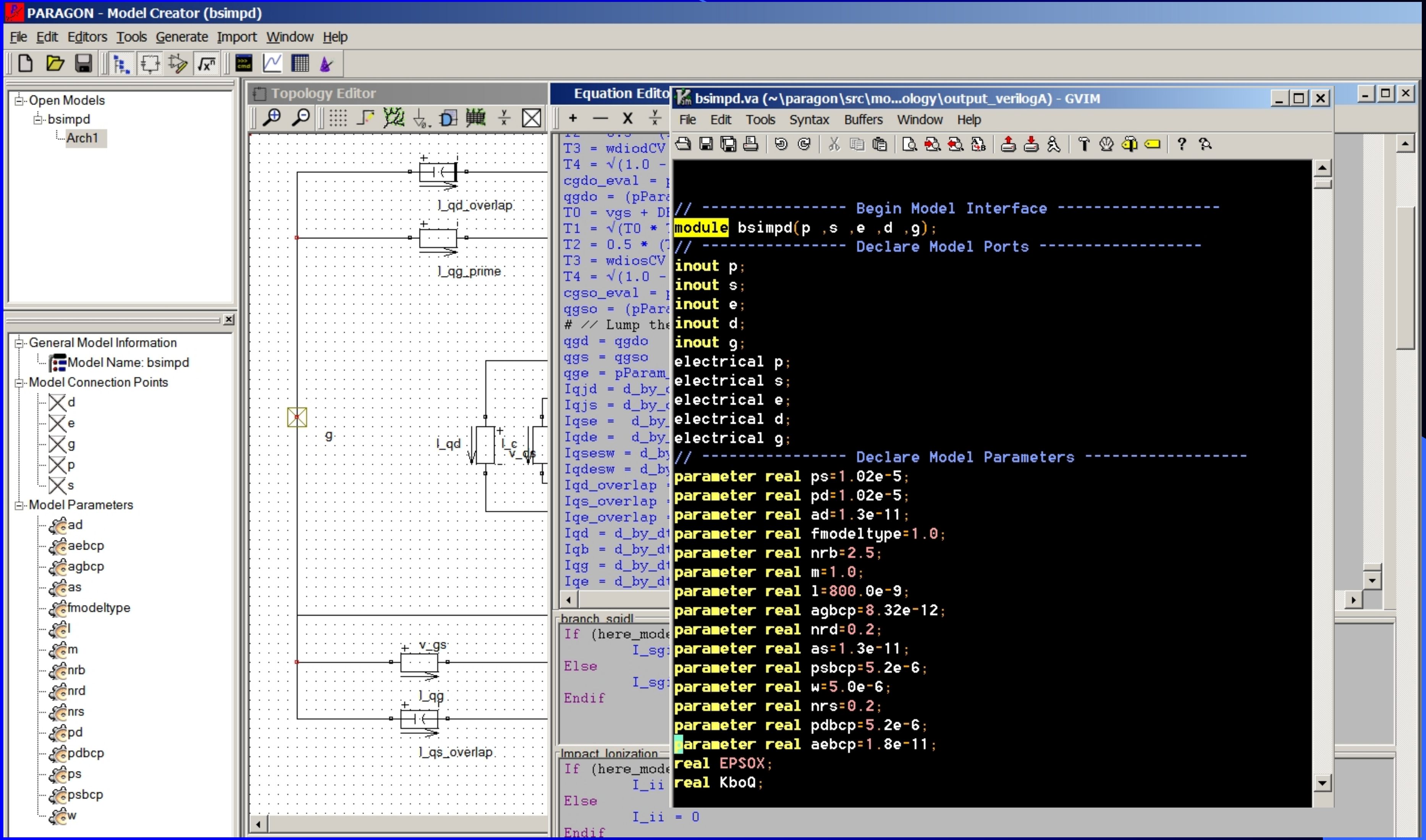
```
T3 = wdiodCV *
T4 = √(1.0 - 4.
cgdo_eval = pPa
qgdo = (pParam_
T0 = vgs + DELT
T1 = √(T0 * T0
T2 = 0.5 * (T0
T3 = wdiosCV *
T4 = √(1.0 - 4.
cgso_eval = pPa
qgso = (pParam_
# // Lump the c
qgd = qgdo
qgs = qgso
qge = pParam_cg
Iqjd = d_by_dt(
Iqjs = d_by_dt(
Iqse =  d_by_dt
Iqde =  d_by_dt
Iqsesw = d_by_d
Iqdesw = d_by_d
Iqd_overlap = d
Iqs_overlap = d
Iqe_overlap = d
Iqd = d_by_dt(q
Iqb = d_by_dt(q
Iqg = d_by_dt(q
Iqe = d_by_dt(q
```

```
If (here_mode>=
        I_sgidl
Else
        I_sgidl
Endif
```

```
If (here_mode>=
        I_ii =
Else
        I_ii =
Endif
```

```
 - <parameter default="1.8e-11" name="aebcp" nature="Real" unit="">
     <comment value=""/>
   </parameter>
 - <port mode="inout" name="p" nature="Electrical" type="terminal">
     <comment value=""/>
     <terminal number="4"/>
   </port>
 - <port mode="inout" name="s" nature="Electrical" type="terminal">
     <comment value=""/>
     <terminal number="3"/>
   </port>
 - <port mode="inout" name="e" nature="Electrical" type="terminal">
     <comment value=""/>
     <terminal number="0"/>
   </port>
 - <port mode="inout" name="d" nature="Electrical" type="terminal">
     <comment value=""/>
     <terminal number="1"/>
   </port>
 - <port mode="inout" name="g" nature="Electrical" type="terminal">
     <comment value=""/>
     <terminal number="2"/>
   </port>
   <Layers/>
 </interface>
- <body name="Arch1">
   <internal_node name="n0" type="electrical"/>
   <equationComment value=" BSIMPD2.2 model"/>
   <equationComment value=" Author: Vivek Chaudhary"/>
   <equationComment value=" email: vchaudh@uark.edu"/>
```

Paragon Implementation



PARAGON - Model Creator (bsimpd)
File Edit Editors Tools Generate Import Window Help
Topology Editor
Equation Editor
bsimpd.va (~\paragon\src\mo...ology\output_verilogA) - GVIM
File Edit Tools Syntax Buffers Window Help
I_qd_overlap
I_qg_prime
g
I_qd
I_c
v_qs
v_gs
I_qg
I_qs_overlap

```
Open Models
  bsimpd
    Arch1
```

```
General Model Information
  Model Name: bsimpd
Model Connection Points
  d
  e
  g
  p
  s
Model Parameters
  ad
  aebcp
  agbcp
  as
  fmodeltype
  l
  m
  nrb
  nrd
  nrs
  pd
  pdbcp
  ps
  psbcp
  w
```

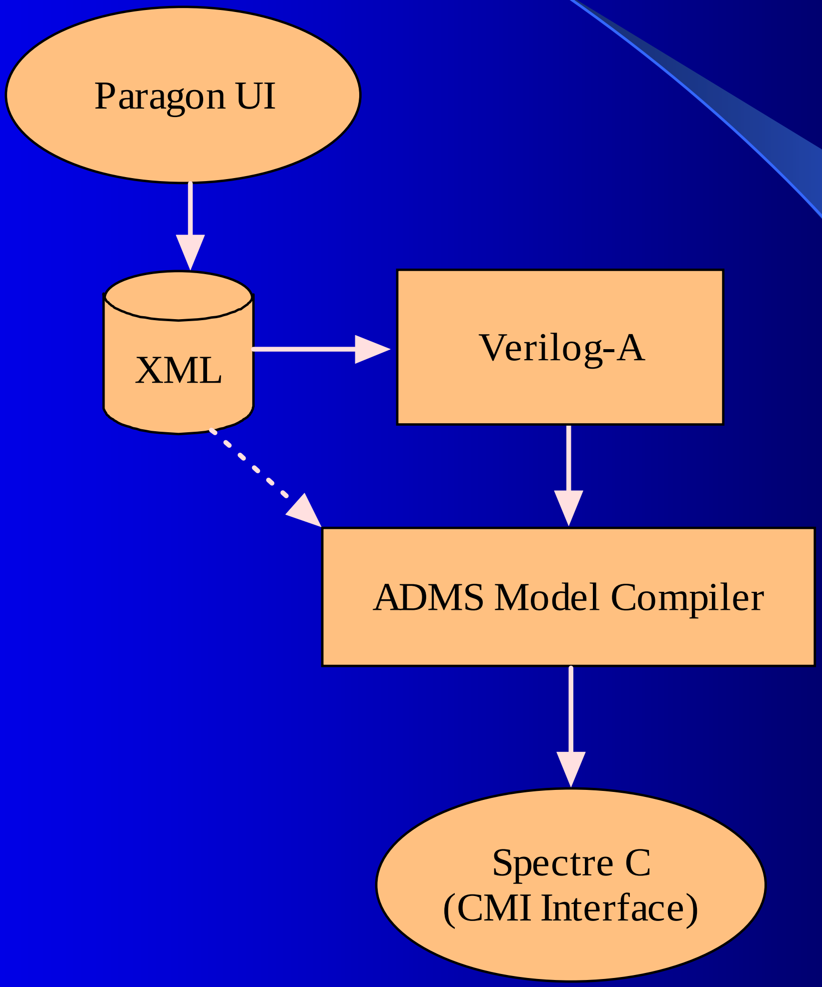
```
T3 = wdiodCV
T4 = √(1.0 -
cgdo_eval = 
qgdo = (pPara
T0 = vgs + DE
T1 = √(T0 * 
T2 = 0.5 * (T
T3 = wdiosCV
T4 = √(1.0 -
cgso_eval = 
qgso = (pPara
# // Lump the
qgd = qgdo
qgs = qgso
qge = pParam_
Iqjd = d_by_d
Iqjs = d_by_d
Iqse =  d_by
Iqde =  d_by
Iqsesw = d_by
Iqdesw = d_by
Iqd_overlap =
Iqs_overlap =
Iqe_overlap =
Iqd = d_by_dt
Iqb = d_by_dt
Iqg = d_by_dt
Iqe = d_by_dt
```

```
branch_sgidl
If (here_mode
        I_sgi
Else
        I_sgi
Endif
```

```
Impact_Ionization
If (here_mode
        I_ii
Else
        I_ii = 0
Endif
```

```
// ---------------- Begin Model Interface -----------------
module bsimpd(p ,s ,e ,d ,g);
// ---------------- Declare Model Ports -----------------
inout p;
inout s;
inout e;
inout d;
inout g;
electrical p;
electrical s;
electrical e;
electrical d;
electrical g;
// ---------------- Declare Model Parameters -----------------
parameter real ps=1.02e-5;
parameter real pd=1.02e-5;
parameter real ad=1.3e-11;
parameter real fmodeltype=1.0;
parameter real nrb=2.5;
parameter real m=1.0;
parameter real l=800.0e-9;
parameter real agbcp=8.32e-12;
parameter real nrd=0.2;
parameter real as=1.3e-11;
parameter real psbcp=5.2e-6;
parameter real w=5.0e-6;
parameter real nrs=0.2;
parameter real pdbcp=5.2e-6;
parameter real aebcp=1.8e-11;
real EPSOX;
real KboQ;
```

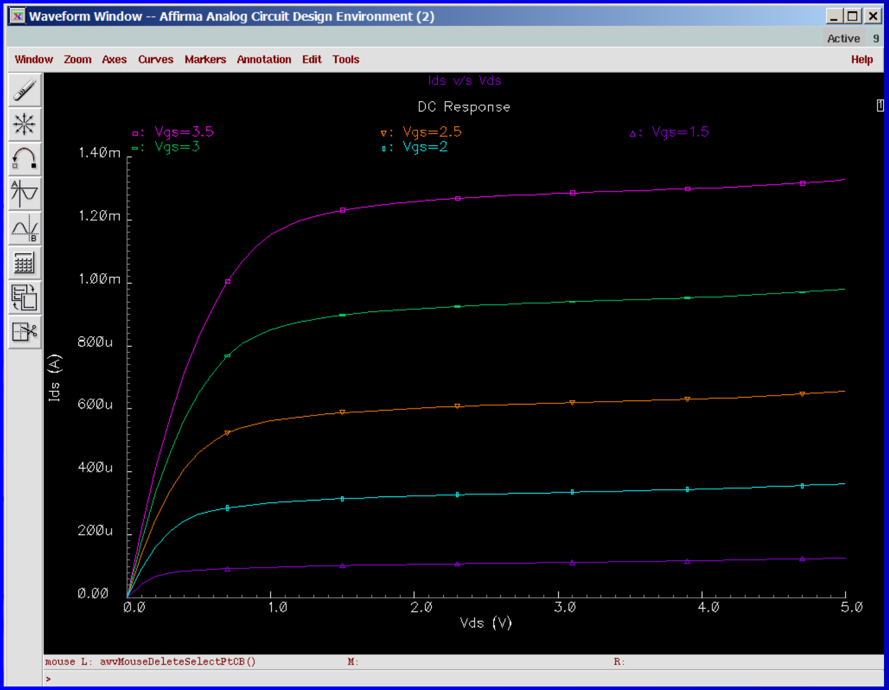
Model Compilation using ADMS

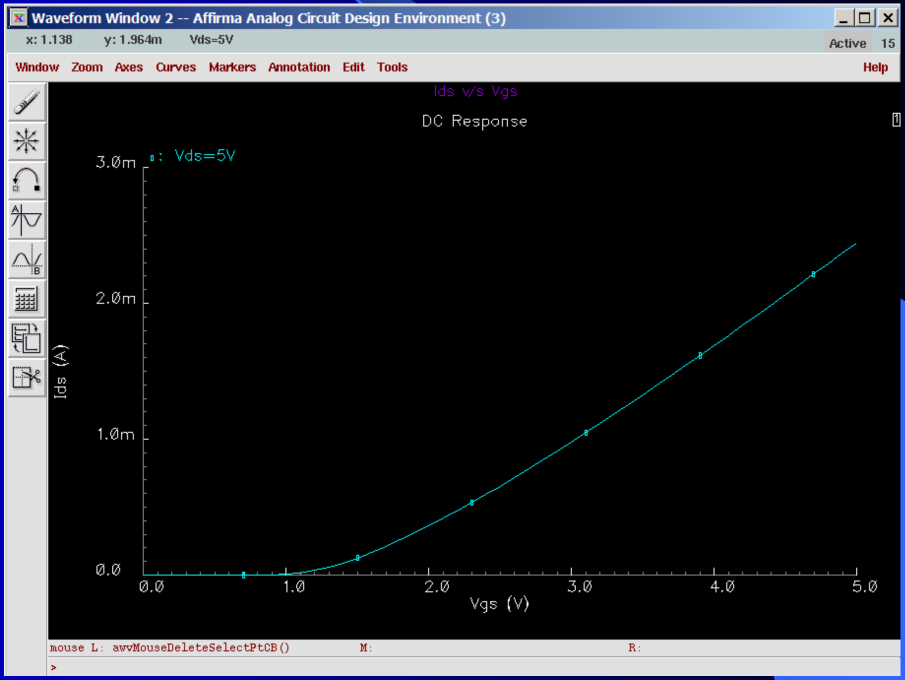


Results

- Duration: about two week to implement and validate the BSIMSOI model in Paragon model and generate C model for Spectre using ADMS.
- Results for all analyses match exactly with native Spectre model

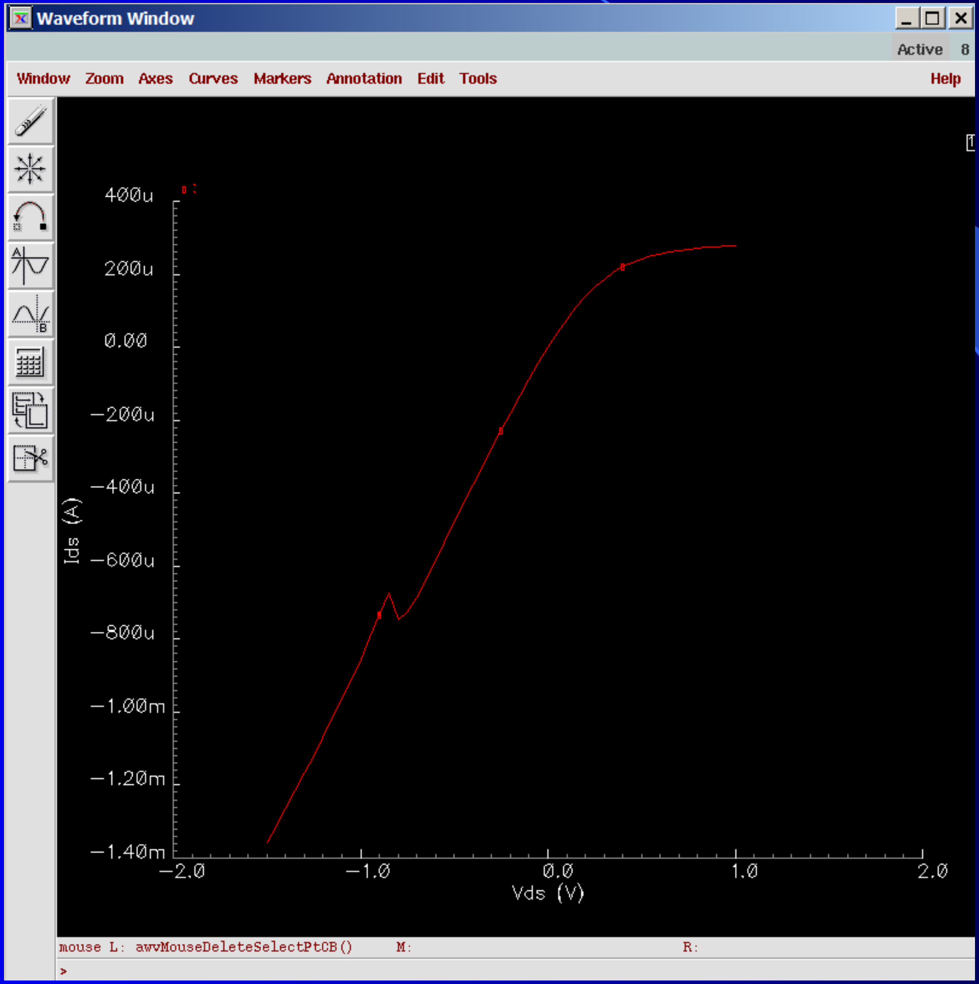
DC Characteristics





10/25/2004

Interesting Behavior

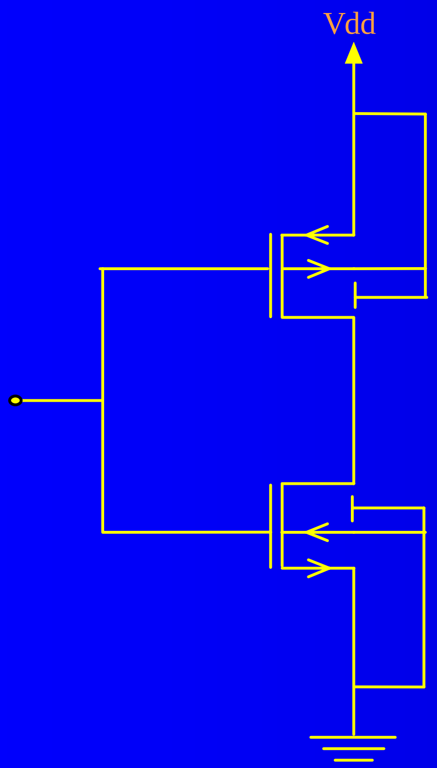


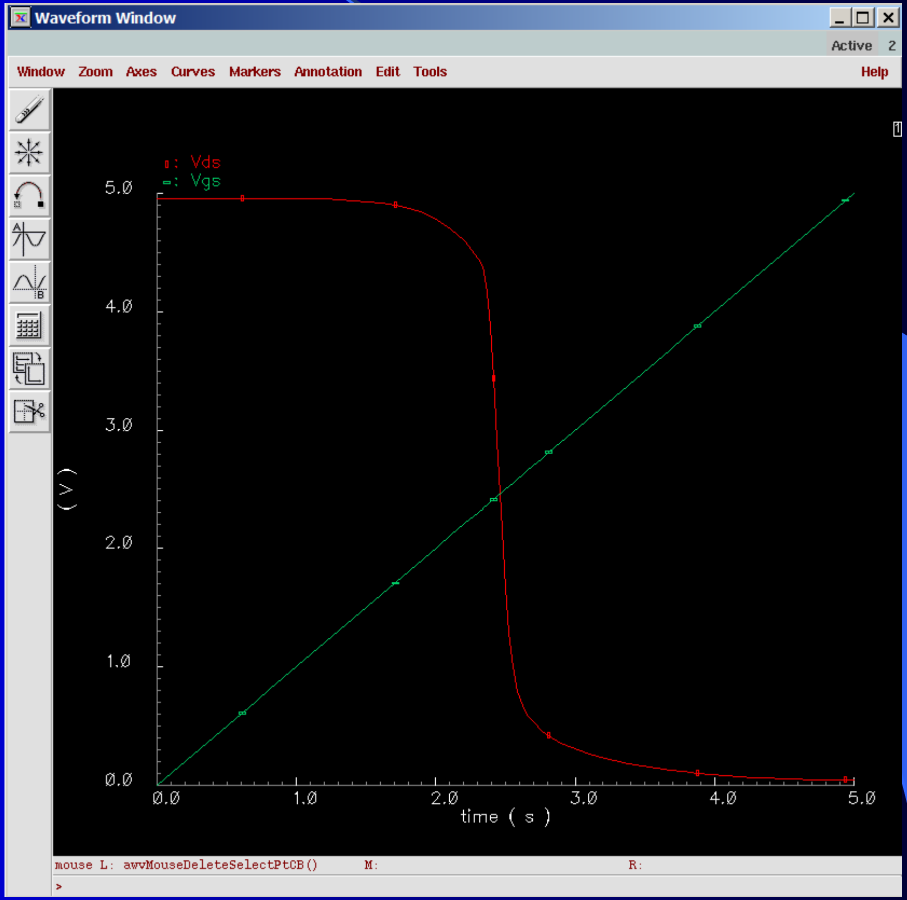
```
mouse L: awvMouseDeleteSelectPtCB()       M:                  R:
>
```

10/25/2004

Relative Performance

Transient analysis of a 2 transistor circuit

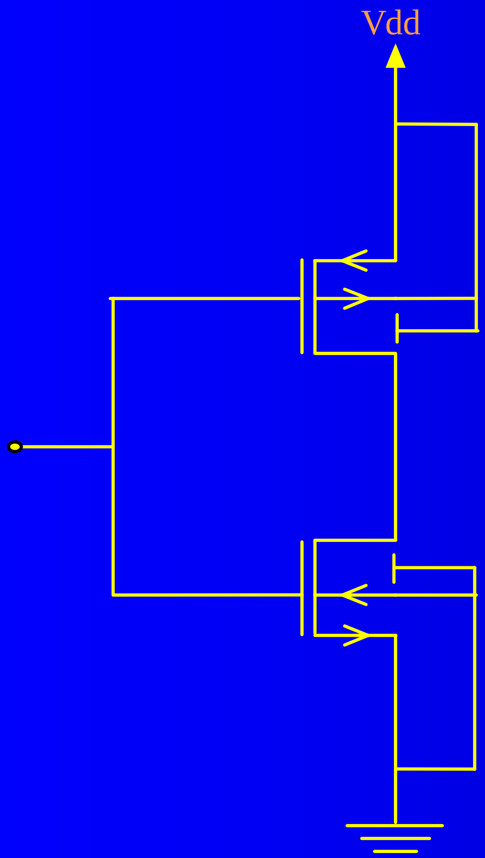




```
mouse L: awvMouseDeleteSelectPtCB()        M:                         R:
>
```

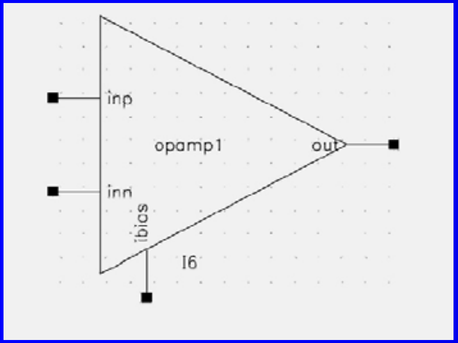
10/25/2004

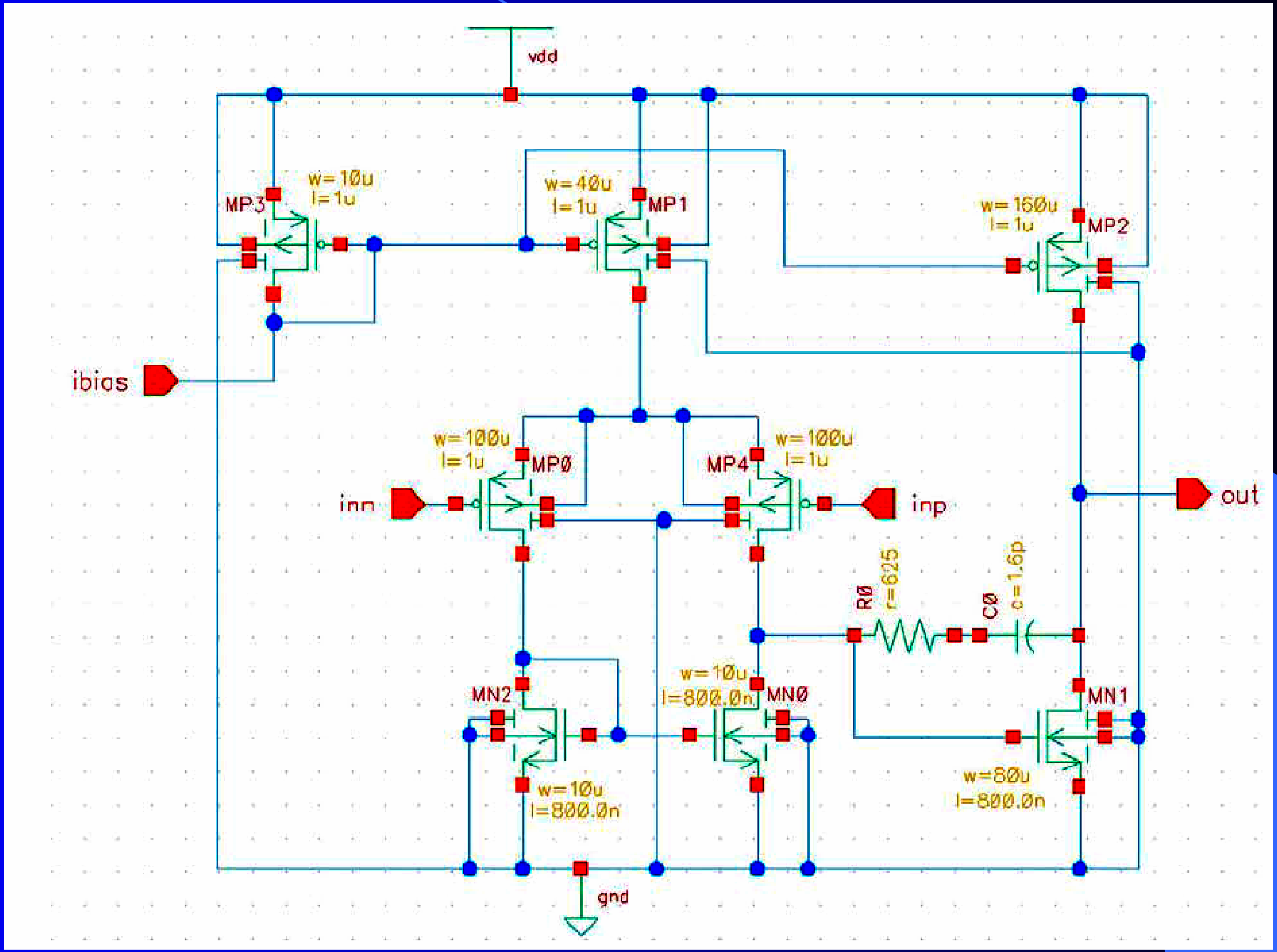
Relative Performance



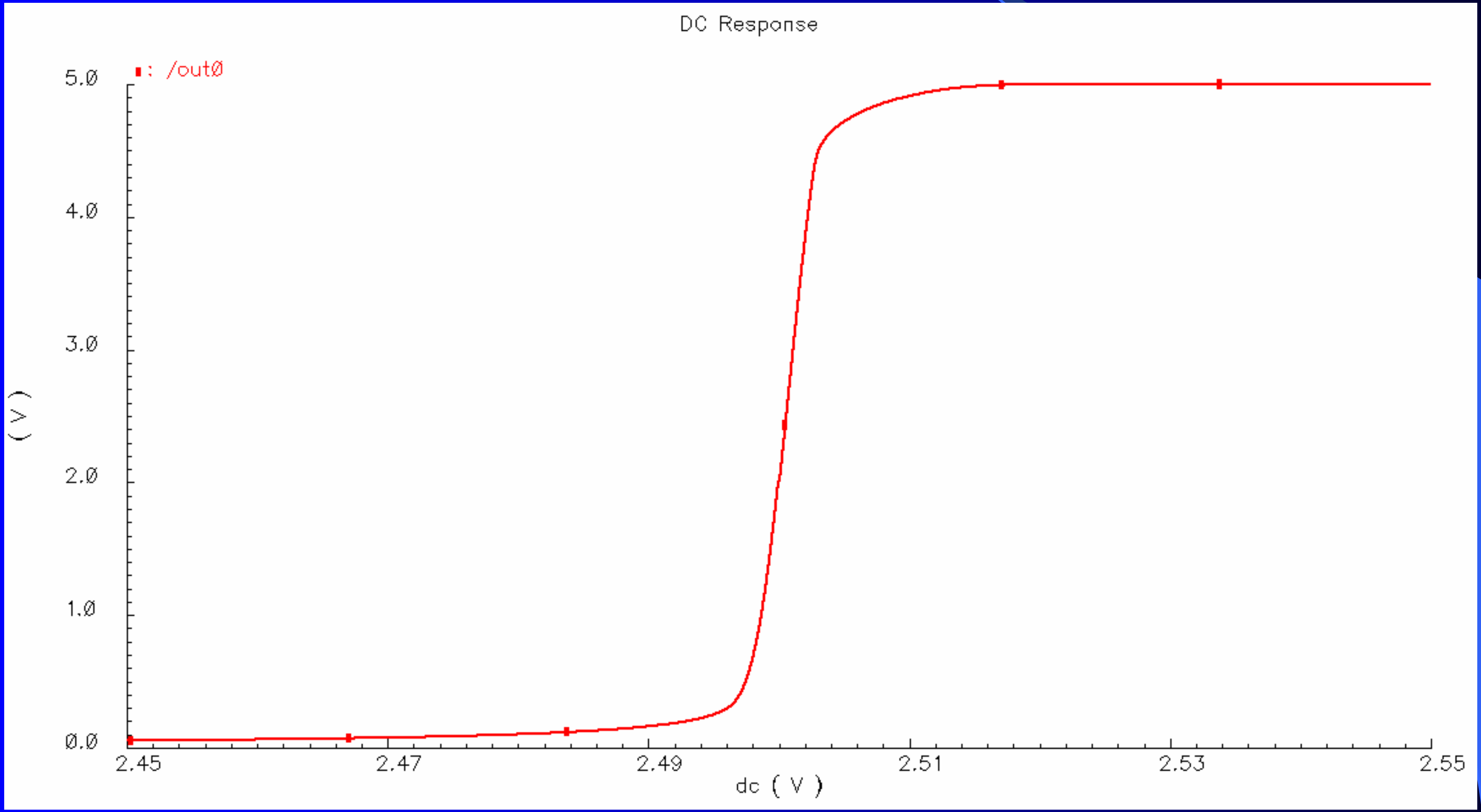
Model	CPU time	Time factor
Native Spectre	140ms	1X
Verilog-AMS	33.2s	237X
ADMS	200ms	1.4X

Radhard Opamp

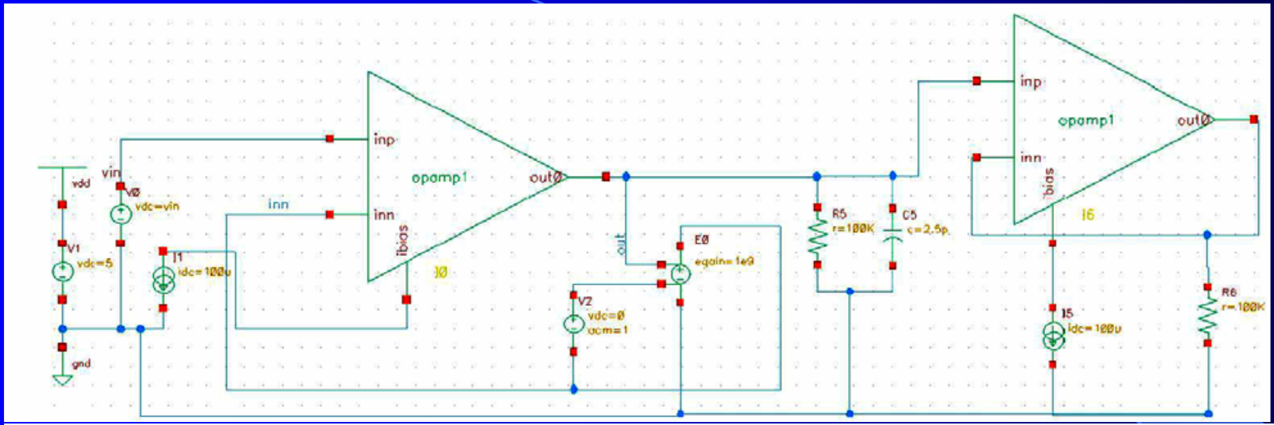


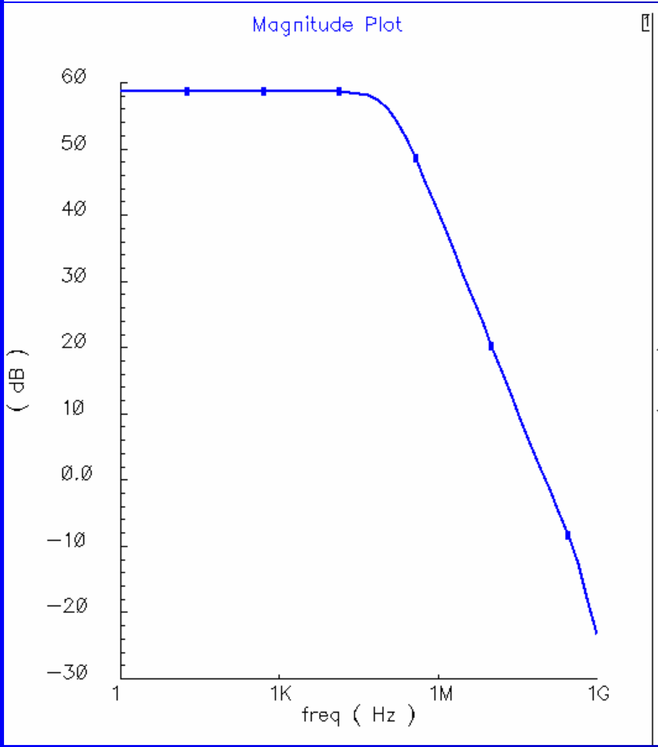


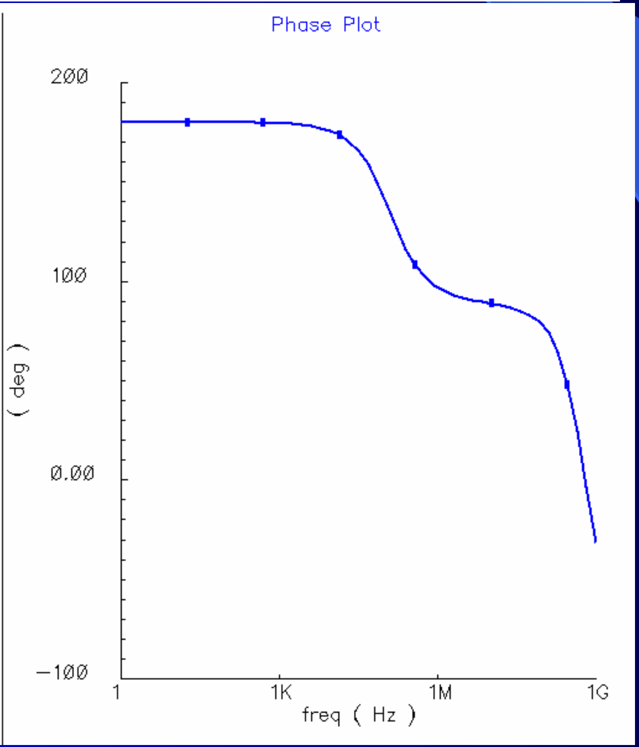
DC Curve



Open Loop Frequency Response







Conclusion

- Modeling methodology described enables the user to quickly and correctly create new compact semiconductor device models for various SPICE and SPICE-like simulators
- Model development time is significantly reduced

Conclusion

- Time required to validate and test a new model is also significantly reduced
- Higher-level XML description of the model is easy to maintain, reuse and update and this leads to an increased efficiency in the overall model creation process.