Transistor Modeling with MODELICA - Experiences

Dresden, April, 2012

Christoph Clauß, Kristin Majetta, Sandra Böhme

Fraunhofer-Institut für Integrierte Schaltungen IIS, Institutsteil Entwurfsautomatisierung EAS, Dresden
Overview

- Modelica
- Simple Models in Modelica.Electrical.Analog
- SPICE3 transistor models
- Performance
- Further development

M. Otter: Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter. Diss., Bochum, 1994

-> DSblock Model


1997 Feb. Technical Commitee of the Federation of European Societies (EUROSIM)

1997 Sep. Modelica Version 1.0

2000 Oct. First Modelica Conference („Workshop“), Lund

Establishing Modelica Association

2007 Sep. Modelica Version 3.0

2010 Mar. Modelica Version 3.2

2011 Mar. 8th Modelica Conference, Dresden

2012 May 75th Modelica Design Meeting
Modelica

At Fraunhofer IIS EAS Dresden

- Membership Modelica Association (A. Schneider, C. Clauß, K. Majetta)

- Library Development:
  - Modelica Standard Library
  - Electrical Analog
  - Electrical Digital
  - Electrical Spice3
  - Spice3 (full, netlist translator)
  - Statistics (statistical parameter variation)
  - Complex

- Modelica related research projects

- Hosting of the 8th Modelica Conference, Dresden

- Tutorials
Introduction

Modelica Tools

Equations

Algorithms and Functions

Instantiation and Connectors

Inheritance

Annotations and Graphics

Overview Standard Library

Concluding Example
This document defines the Modelica language, version 3.2, which is developed by the Modelica Association, a non-profit organization with seat in Linköping, Sweden. Modelica is a freely available, object-oriented language for modeling of large, complex, and heterogeneous physical systems. It is suited for multi-domain modeling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic and control subsystems, process oriented applications and generation and distribution of electric power. Models in Modelica are mathematically described by differential, algebraic and discrete equations. No particular variable needs to be solved for manually. A Modelica tool will have enough information to decide that automatically. Modelica is designed such that available, specialized algorithms can be utilized to enable efficient handling of large models having more than one hundred thousand equations. Modelica is suited and used for hardware-in-the-loop simulations and for embedded control systems. More information is available at http://www.Modelica.org/
Modelica

Specialized Algorithms

Modelica Model

Flat Model

Set of Equations

Symbolically optimized Set of Equations

C Code

Solver

Executable

Results

Postprocessed Results
Modelica

- SimulationX (ITI)
- Dymola (Dassault Systemes)
- AMESim (LMS)
- MapleSim (MapleSoft, Canada, www.maplesoft.com)
- OpenModelica (Open Source Modelica Consortium, OSMC www.openmodelica.org)
- JModelica
Example:

Mathematical description

\[ a(x - b) + b(x + a) = -x \]

Solution

\[ ax - ab + bx + ab = -x \]

\[ x = 0 \]
Example: Rössler Differential Equation

### Mathematical description

\[
\begin{align*}
\dot{x} &= -y - z \\
\dot{y} &= x + ay \\
\dot{z} &= xz - cz + b
\end{align*}
\]

Parameters:  
- \(a = 0.2\)  
- \(b = 0.2\)  
- \(c = 4.1\)

Start values:
- \(x = -0.03\)
- \(y = -0.03\)
- \(z = +0.03\)

### Modelica model

```modelica
model RoesslerAttractor
  parameter Real a = 0.2;
  parameter Real b = 0.2;
  parameter Real c = 4.1;
  Real x (start = -0.03, fixed = true);
  Real y (start = -0.03, fixed = true);
  Real z (start = 0.03, fixed = true);
  equation
    der(x) = -y - z;
    der(y) = x + ay;
    der(z) = xz - cz + b;
end RoesslerAttractor;
```

Derivative operator: `der`
Modelica

Modelica model

model InstantiationFailed
model Instant
    constant Real a = 5;
    parameter Real b = 3;
    Real x;
    equation
        x = b*sin(time) + a;
end Instant;
end InstantiationFailed;

model InstantiationSuccessful
model Instant
    constant Real a = 5.5;
    parameter Real b = 3;
    Real x;
    equation
        x = b*sin(time) + a;
end Instant;
end InstantiationSuccessful;

<table>
<thead>
<tr>
<th>i_x</th>
<th>i_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Modelica model

```modelica
model OscillatorParameter
  parameter Real xs = 1;
  parameter Real a = 2;
  Real x(start = xs);
  Real xp( start = 0);
end OscillatorParameter;

model OscillatorParameter1
  extends OscillatorParameter(a = 10);
end OscillatorParameter1;

model OscillatorParameter2
  extends OscillatorParameter1(xs = 2);
end OscillatorParameter2;
```

Inheritance
Modelica

\[
v := [\dot{x}, x, y, t, m, \text{pre}(m), p]
\]

\[
c := F_c(\text{relation}(v)) \quad (1a)
\]

\[
m := F_m(v, c) \quad (1b)
\]

\[
0 = F_x(v, c) \quad (1c)
\]

\[
p \quad \text{parameters or constants}
\]

\[
t \quad \text{variable time}
\]

\[
x(t) \quad \text{variables type Real, appearing differentiated}
\]

\[
m(t_e) \quad \text{unknown variables, change value only at time events } t_e
\]

\[
y(t) \quad \text{variables type Real}
\]

\[
c(t_e) \quad \text{Conditions of if-expressions}
\]

\[
\text{relation}(v) \quad \text{Relation containing variables } v_i
\]

Hybrid DAE

**Simulation Algorithm**

1. DAE(1c) solved, c and m constant

2. During integration, all relations from (1a) are monitored, if one value changes → event is triggered (special handling of relations that are only time dependent)

3. At event, (1) is a mixed set of algebraic equations which is solved for all unknowns

4. After event is processed → 1
Modelica Libraries

- collections of well tested and often very complex components written in Modelica
- Modelica models independent from simulation environment
- can be shared with other developers even if they use another simulation tool

**Free libraries**

- Modelica Standard Library
- ATplus Library
- BondLib
- Buildings Library
- ExtendedPetriNets Library
- FuelCellLib
- FuzzyControl Library
- MultiBondLib
- SPICELib
- SystemDynamics Library
- ...
- more at https://www.modelica.org/libraries

**Commercial libraries**

- AirConditioning Library
- BG_RT Library (for realtime simulation)
- Belts Library
- CombiPlant Library
- FlexibleBodies Library
- FlexBody Library
- HumanComfort Library
- Hydraulics Library
- Pneumatics Library
- Power Train Library
- SmartElectricDrives
- VehicleDynamics Library
- ...
Overview

- Modelica
- **Simple Models in Modelica.Electrical.Analog**
- SPICE3 transistor models
- Performance
- Further development
Simple Models in Modelica.Electrical.Analog

- Simple MOS
- Bipolar Ebers-Moll

Temperature dependent calculation
Calculation of loss power
model PMOS "Simple MOS Transistor"
  Interfaces.Pin D, G, S, B;
  parameter SIunits.Length W=20.0e-6 "Width";
  parameter SIunits.Length L=6.0e-6 "Length";
  parameter SIunits.Transconductance Beta=0.0105e-3;
  parameter SIunits.Voltage Vt=-1.0 "Zero bias threshold voltage";
  parameter Real K2=0.41 "Bulk threshold parameter";
  parameter Real K5=0.839 "Reduction of pinch-off region";
  parameter SIunits.Length dW=-2.5e-6 "Narrowing of channel";
  parameter SIunits.Length dL=-2.1e-6 "Shortening of channel";
  parameter SIunits.Resistance RDS=1.e+7 "Drain-Source-Resistance";
extends Interfaces.ConditionalHeatPort(T=293.15);

Real v, uds, ubs, ud, us, id, gds = ... 1/RDS;

equation
  v = Beta*(W + dW)/(L + dL);
  ud = if (D.v > S.v) then S.v else D.v;
  us = if (D.v > S.v) then D.v else S.v;
  uds = ud - us;
  ubs = if (B.v < us) then 0 else B.v - us;
  ugst = (G.v - us - Vt + K2*ubs)*K5;
  id = if (ugst >= 0) then uds*gds else if (ugst < uds) then -v*uds*(ugst - uds/2) + uds*gds else -v*ugst*ugst/2 + uds*gds;
  G.i = 0; B.i = 0;
  D.i = if (D.v > S.v) then -id else id;
  S.i = if (D.v > S.v) then id else -id;
  LossPower = D.i * (D.v - S.v);  //T_heatPort is temperature
end PMOS;
Simple Models in Modelica.Electrical.Analog

Example
Overview

- Modelica
- Simple Models in Modelica.Electrical.Analog
- **SPICE3 transistor models**
- Performance
- Further development
SPICE3 Transistor Models

SPICE-netlist

**inverter**

Mp1 12 1 13 12 MPmos L=5U W=2U
Mp2 22 13 23 22 MPmos L=5U W=2U
Mn1 13 1 0 0 MNmos L=5U W=2U
Mn2 23 13 0 0 MNmos L=5U W=2U

Vgate 1 0 PULSE(0 5 2s 1s)
Vdrain 11 0 PULSE(0 5 0s 1s)
V1 11 12 0
V2 11 22 0

.model MPmos PMOS (Id=0.8u vt0=1 )
.model MNmos NMOS (lambda=0.02 kp=3.1e-5)

.tran 0.01 5
.control
run
set options no break
plot i(v1) i(v2)

.endc
.end

**Simulation Program with integrated Circuit Emphasis**

**simulator SPICE3**

**types of analysis:**
- transient
- DC
- AC
SPICE3 Transistor Models

Task of Transformation

System of differential equations

\[ F(\dot{x}(t), x(t), p, t) = 0 \]

Discretization of time

Nonlinear systems of equations

\[ \tilde{F}(\frac{1}{h}(x_i - x_{i-1}), x_i, p, t_i) = 0 \]

Iteration method

Linear systems of equations

\[ Ax_i^{j+1} = b(x_i^j) \]
SPICE3 Transistor Models

Task of Transformation

- System of differential equations
  \[ F(\dot{x}(t), x(t), p, t) = 0 \]

  - Discretization of time

- Nonlinear systems of equations
  \[ \tilde{F}(\frac{1}{h}(x_i - x_{i-1}), x_i, p, t_i) = 0 \]

  - Iteration method

- Linear systems of equations
  \[ A x_i^{j+1} = b(x_i^j) \]

- SPICE3 Transistor Models
  - Modelica text
    - Equations (declarative)
    - Symbolic preprocessing

- SPICE netlist
  - Assignments (procedural)

- SPICE3/C
SPICE3 Transistor Models

Task of Transformation

Converting the semiconductor models

SPICE netlist

Modelica text

equations (declarative)

symbolic preprocessing

SPICE3/C (C++)

assignment (procedural)

System of differential equations

\[ F(\dot{x}(t), x(t), p, t) = 0 \]

discretization of time

nonlinear systems of equations

\[ \tilde{F}(\frac{1}{h}(x_i - x_{i-1}), x_i, p, t_i) = 0 \]

iteration method

linear systems of equations

\[ Ax_{i+1} = b(x_i^j) \]
SPICE3 Transistor Models

Task of Transformation

1. creating Top-level model
2. Transforming of parameters
3. Transforming of data structure
4. Transforming C++ methods to Modelica functions

transforming steps

precondition

splice3 library for modelica

result

C++ class
Modelica record

C++ method
Modelica function

channel charge
bulk source junction
bulk drain junction
channel current
parameter
technology parameter
device parameter
data

1. creating Top-level model

Interfaces.POSITIVEPIN NG "gate node";
Interfaces.POSITIVEPIN ND "drain node";
Interfaces.NEGATIVEPIN NS "source node";
Interfaces.POSITIVEPIN NB "bulk node";

//inner nodes
Real Din, Sin;
Real ird, irs, icgd, icgs,...;

// resistance at drain and source
ird * cl.m_draingResistance = (ND.v - Din);
irs * p.m_sourceResistance = (NS.v - Sin);

//sum of currents at outer nodes
NG.i = icGB + icGD + icGS;
NB.i = cc.iBD + cc.iBS;
ND.i = ird;
NS.i = irs;

//currentsum at inner nodes
0 = -ird + cc.idrain - cc.iBD - ibdmin - icGD -icBD;
0 = -irs - cc.idrain - cc.iBS - ibsgmin - icGS - icBS;
SPICE3 Transistor Models

2. Transforming of parameters

```
parameter Real mtype(start = 0); // PMOS
parameter Real L(start = 1.e-4);
parameter Real W(start = 1.e-4);
parameter Real AD( start = 0);
parameter Real AS( start = 0);
parameter Real PD( start = 0);
parameter Real PS( start = 0);
parameter Real NRD( start = 1);
parameter Real NRS( start = 1);
parameter Real OFF( start = 0);
parameter Real IC( start = -1e40); // def 0
parameter Real TEMP( start = 300.15);
parameter Repository.modelcardMOS modelcardMOS

record modelcardMOS
  "record with technology parameters (.model)"
  parameter Real VTO=-1e40
    "V zero-bias threshold voltage, default 0;"
  parameter Real PHI=-1e40
    "V surface potential, default 0.6;"
  parameter Real LAMBDA=0
    "1/V channel-length modulation, default 0;"
  parameter Real RD=-1e40
    "Ohm drain ohmic resistance, default 0;"
  parameter Real CBD=-1e40
    "F zero-bias B-D junction cap., default 0;"
  parameter Real IS=1.e-14
    "A bulk junction saturation current;"
  parameter Integer LEVEL=1;
... end modelcardMOS;
```
3. Transforming of data structure

In C++:
- C++ class
- data

In Modelica:
- Record
- data

- Collection of data
- Administration of data
- Initiate data at the same time
- Inheritable

Inheriting:
- MosfetCalc
- MosCalc
- Mos1Calc
### 4. Transforming C++ of methods to Modelica functions

**C++ method** → **Modelica function**

#### function DrainCur

```cpp
//C++ double vb, double vg, double vds,
//C++ double &cdrain, double &gm, ...)
input Real vb, vg, vds;
input Mos1Calc in_c;
input Mos1ModelLineParams in_p;
...
output Mos1Calc out_c;
//C++ double arg, betap, sarg, vgst;
protected
  Real arg, betap, sarg, vgst;
algorithm
  out_c := in_c;
  //C++ if (vb <= 0)
  //C++      sarg = sqrt( m_tPhi - vb);
  //C++ else { sarg = sqrt( m_tPhi);... }
if (vb <= 0) then
  sarg := sqrt( out_c.m_tPhi - vb);
else
  sarg := sqrt( out_c.m_tPhi);
end if; ...
end Draincur;
```

- **name from C++**
- **needed input- and output records**
- **internal variables**
- **input record is written to output record**
- **original C++ code as comment**
- **transformed Modelica source code**
## SPICE3 Transistor Models

### Basics
- R, C, L, K
- linear controlled sources

### Semiconductors
- **Resistor**
- **Capacity**
- **Diode**
- **Junction field effect transistor (JFET)**
- **Metal semiconductor field effect transistor (MESFET)**
- **Metal oxide semiconductor field effect transistor (MOSFET)**

### Lines
- U-Line, O-Line, T-Line

### Sources
- **(Current, Voltage)**
- **Constant**
- **Pulse**
- **Damped Sine**
- **Exponential**
- **Piecewise linear**

### Source Models

<table>
<thead>
<tr>
<th>Level</th>
<th>Model</th>
<th>Channel Length/µm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MOS1 (Shichman-Hodges)</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>MOS2 (more realistic)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>MOS3 (semi-empirical)</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>BSIM1 (Berkely Short Channel IGFET M.)</td>
<td>0.8</td>
</tr>
<tr>
<td>5</td>
<td>BSIM2</td>
<td>0.35</td>
</tr>
<tr>
<td>6</td>
<td>MOS6</td>
<td></td>
</tr>
<tr>
<td>7/8</td>
<td>BSIM3 (scalable)</td>
<td>0.25</td>
</tr>
</tbody>
</table>

### Bipolar Junction Transistor (BJT)
- **Ebers-Moll, Gummel-Poon**

### Switch
- **part of Standard Library**

**All: part of the EAS Spice3 Library**
SPICE3 Transistor Models  Example Comparison MOS1, MOS2

- MOS1
- MOS2

48 parameter
about 1500 lines of code
about 170 Variables
SPICE3 Transistor Models

SPICE3

Rectifier
vsin  1  0  dc 1 sin(0  15  50) AC 1
ri    1  2  0.1
d1    2  3  diode
c     3  0  0.0022
rl    3  0  100
.model diode d  is=1.e-12 rs=5
.tran 0.002 0.3 0 0.001
.options nopage opts  acct list
.control
.run
.set options no break
.print v(3) v(1)
.print v(2)
.print i(vsin)
.endc
.end

Spice3 Netlist Translator SpiToMo

model "Rectifier"
.import Modelica.Electrical.Spice3.*;
.parameter Semiconductors.ModelcardDIODE DIODE(IS=1e-012, RS=5);
.Sources.V_sin VSIN( VO=0, VA=15, FREQ=50);
.Basic.R_Resistor RI(R=0.1);
.Semiconductors.D_DIODE D1(modelcarddiode=DIODE);
.Basic.C_Capacitor C(C=0.0022);
.Basic.R_Resistor RL(R=100);
.Basic.Ground g;
.protected
.equation
.connect (g.p,n0);
.connect (VSIN.p, n1);
.connect (VSIN.n, n0);
.connect (RI.p, n1);
.connect (RI.n, n2);
.connect (D1.p, n2);
.connect (D1.n, n3);
.connect (C.p, n3);
.connect (C.n, n0);
.connect (RL.p, n3);
.connect (RL.n, n0);
.annotation (uses(Modelica(version="3.2")),
.experiment(StopTime=0.3, Interval=0.002));
.end Rectifier;

Written in Modelica (String-functions)
Overview

- Modelica
- Simple Models in Modelica.Electrical.Analog
- SPICE3 transistor models
- Performance
- Further development
Performance

Modelica Model

Flat Model

Set of Equations

Symbolically optimized Set of Equations

C Code

Solver

Executable

Results

Postprocessed Results
Performance

RC chain – one C

Symbolical Simplification before Simulation

Dymola

Original Model
- Number of components: 304
- Variables: 4217
- Parameters: 1503 (1503 scalars)
- Unknowns: 2714 (2714 scalars)
- Differentiated variables: 1 scalars
- Equations: 2416
- Nontrivial: 2112

Translated Model
- Constants: 799 scalars
- Free parameters: 902 scalars
- Parameter depending: 600 scalars
- Continuous time states: 1 scalars
- Time-varying variables: 706 scalars
- Alias variables: 1210 scalars
- Assumed default initial conditions: 1
- Number of mixed real/discrete systems of equations: 0
- Sizes of linear systems of equations: {400}
- Sizes after manipulation of the linear systems: {100}
- Sizes of nonlinear systems of equations: {}
- Sizes after manipulation of the nonlinear systems: {}
- Number of numerical Jacobians: 0
### Performance

**RC chain – one C**

- **Spice**
  - Number of equations: 201 nodes
  - 0 internal nodes
  - 201 total

### Symbolical Simplification before Simulation

#### Dymola

- Number of components: 303
- Variables: 3510
- Parameters: 1102 (1102 scalars)
- Unknowns: 2408 (2408 scalars)
- Differentiated variables: 100 scalars
- Equations: 2010
  - Nontrivial: 1707

#### Translated Model

- Constants: 699 scalars
- Free parameters: 701 scalars
- Parameter depending: 400 scalars
- Continuous time states: 100 scalars
- Time-varying variables: 603 scalars
- Alias variables: 1107 scalars
- Assumed default initial conditions: 100
- Number of mixed real/discrete systems of equations: 0
- Sizes of linear systems of equations: {4}
- Sizes after manipulation of the linear systems: {0}
- Sizes of nonlinear systems of equations: {}
- Sizes after manipulation of the nonlinear systems: {}
- Number of numerical Jacobians: 0

---

**Evaluate**

- true
- 2003
- 0
- 0
- 100
- 503
- 1004
- 100
- 3
- 0
**Performance**

Inverter using Spice3 MOS

**Symbolical Simplification before Simulation**

<table>
<thead>
<tr>
<th>Spice</th>
<th>Number of equations:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 nodes</td>
</tr>
<tr>
<td></td>
<td>4 internal nodes</td>
</tr>
<tr>
<td></td>
<td>7 total</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dymola</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Model</strong></td>
<td></td>
</tr>
<tr>
<td>Number of components: 22</td>
<td></td>
</tr>
<tr>
<td>Variables: 696</td>
<td></td>
</tr>
<tr>
<td>Constants: 2 (2 scalars)</td>
<td></td>
</tr>
<tr>
<td>Parameters: 608 (608 scalars)</td>
<td></td>
</tr>
<tr>
<td>Unknowns: 86 (86 scalars)</td>
<td></td>
</tr>
<tr>
<td>Differentiated variables: 2 scalars</td>
<td></td>
</tr>
<tr>
<td>Equations: 66</td>
<td></td>
</tr>
<tr>
<td>Nontrivial: 59</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Translated Model</strong></th>
<th>Evaluate true</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants: 615 scalars</td>
<td>635</td>
</tr>
<tr>
<td>Free parameters: 13 scalars</td>
<td>0</td>
</tr>
<tr>
<td>Parameter depending: 7 scalars</td>
<td>0</td>
</tr>
<tr>
<td>Time-varying variables: 41 scalars</td>
<td>0</td>
</tr>
<tr>
<td>Alias variables: 20 scalars</td>
<td></td>
</tr>
<tr>
<td>Assumed default initial conditions: 4</td>
<td></td>
</tr>
<tr>
<td>Number of mixed real/discrete systems of equations: 0</td>
<td></td>
</tr>
<tr>
<td>Sizes of linear systems of equations: {}</td>
<td></td>
</tr>
<tr>
<td>Sizes after manipulation of the linear systems: {}</td>
<td></td>
</tr>
<tr>
<td>Sizes of nonlinear systems of equations: {39}</td>
<td></td>
</tr>
<tr>
<td>Sizes after manipulation of the nonlinear systems: {1}</td>
<td></td>
</tr>
<tr>
<td>Number of numerical Jacobians: 1</td>
<td></td>
</tr>
</tbody>
</table>
Performance

smooth, noEvent

\[
\text{id} = \text{smooth}(0, \text{if } (\text{ugst} \geq 0) \text{ then } \text{uds} \times \text{gds} \\
\text{else if } (\text{ugst} < \text{uds}) \text{ then } -v \times \text{uds} \times (\text{ugst} - \text{uds}/2) + \text{uds} \times \text{gds} \\
\text{else } -v \times \text{ugst} \times \text{ugst}/2 + \text{uds} \times \text{gds})
\]

\[
\text{id} = \text{smooth}(p, \text{expression})
\]

Expression is p times continuously differentiable

\[
\text{id} = \text{noEvent}(\text{expression})
\]

No event is generated

Adder chain 10 full adders

<table>
<thead>
<tr>
<th>No-Event used</th>
<th>smooth used</th>
<th>F-Evaluations</th>
<th>steps</th>
<th>CPU/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1994791</td>
<td>8575</td>
<td>1230</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>1487274</td>
<td>6767</td>
<td>962</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>28284</td>
<td>1541</td>
<td>4.8</td>
</tr>
</tbody>
</table>

Dymola
Performance

- At initialization a simplified expression is step by step changed into the actual expression:

  \[ \lambda \times \text{actual} + (1 - \lambda) \times \text{simplified} \]

- Example Voltage Source

```model ConstantVoltageSource
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  parameter Modelica.SIunits.Voltage V;
  equation
    v = homotopy(actual=V, simplified=0.0);
end ConstantVoltageSource;
```

- To be checked for transistor modeling
Overview

- Modelica
- Simple Models in Modelica.Electrical.Analog
- SPICE3 transistor models
- Performance
- Further development
Further Development

**Ekv 2.6**
- Verilog-A code

**EKV_26.mo**
- preliminary Modelica code

**EKV_26.mo**
- final Modelica code

got from Wladek Grabinski December 2011

Verilog reference simulations prepared

- Syntactical correct code ready
- to be tested - no results yet
- Dymola: convergency problems in solving nonlinear equations
- planned
  - Numerical reference tests
  - Code optimizations
  - Usage of more equations instead of assignments (:=)
Further Development

```model EKV_26
    constant Real C_EPSSIL = 1.03594314e-10;
    constant Real C_EPSOX = 34.5e-12;
    constant Real C_QE = 1.602e-19;
    constant Real C_K = 1.3807e-23;
    constant Real P_K = Modelica.Constants.k;
    constant Real P_EPS0 = Modelica.Constants.epsilon_0;
    constant Real P_CELSIUS0= - Modelica.Constants.T_zero;
    constant Real POS_MIN = 1.0E-6;
    constant Real SQRT2= sqrt(2);
    constant Real ONE3RD= 1/3;
    constant Real ONESQRT2= 1/sqrt(2);
    constant Integer FWD= 1;
    constant Integer REV= -1;
    // AB 040902
    constant Real NOT_GIVEN = -1.0e21;
    constant Real DEFAULT_TNOM = 25;

    // parameter definitions
    parameter Integer TYPE(min=-1, max=1) = 1; // NMOS 1, PMOS -1
    parameter Integer Noise(min=0, max=1) = 1; // Set to zero to
        // prevent noise calculation
    parameter Real Trise = 0.0; // Difference sim. temp
        // and device temp [C deg]
```
Further Development

Modelica.Electrical.Analog.Interfaces.Pin D;


...  
**algorithm**

\[
\text{EPSOX} := 3.9 \times P_{\text{EPS0}};
\]

\[
\text{epssil} := 11.7 \times P_{\text{EPS0}};
\]

**...**

\[
\text{if } (\text{TYPE} > 0) \text{ then}
\]

\[
\text{VTO_S} := \text{if } (\text{AVTO} \neq 1e-6) \text{ then } (\text{AWL} \times (\text{AVTO} - 1e-6) + \text{VTO_T}) \text{ else } (\text{VTO_T});
\]

**else**

\[
\text{VTO_S} := \text{if } (\text{AVTO} \neq 1e-6) \text{ then } (\text{AWL} \times (1e-6 - \text{AVTO}) - \text{VTO_T}) \text{ else } (-\text{VTO_T});
\]

**end if**;

**...**

**equation**

\[
\text{B}.i = - \text{iSB} - \text{iDB} - \text{iGB};
\]

\[
\text{S}.i = - \text{iDS} + \text{iSB};
\]

\[
\text{D}.i = \text{iDB} + \text{iDS};
\]

\[
\text{G}.i = \text{iGB};
\]

\[
\text{temperature}.Q\text{\_flow} = 0.0;
\]

**end EKV_26;**
Further Development

Modelica Model Generation Tool

Automatic transformation of VerilogA models into other languages using XSLT-descriptions

At present:
Transformation into SystemC-AMS

Planned:
Transformation into Modelica

SystemC-AMS description - XML
Further Development

- Usage of Modelica for systems design in electronics and mechatronics

- Enlargement of electronic Modelica libraries (both ideal, and detailed models)

- Test and comparison to electronic circuit simulators (Saber, Spice), regression tests

- Algorithmic improvements (DC-transfer, AC, sensitivity analysis, )

- (Automatical) model generation (from VerilogA, VHDL-AMS)

Planned research project Modelectrica
Conclusion

- Complex transistor modeling (many equations) with Modelica is possible.

- Transistor models of different complexity are provided for everyone’s usage.

- The presented approach (a few equations - many functions) comes from the transformation of existing procedural coded models. The Modelica typical usage of many equations has to be checked.

- The performance of simulating transistor circuits is much worse using Modelica simulators than using electrical simulators (SPICE).

- Modelica transistor circuits can be connected to other Modelica models – multidomain simulation with one simulation tool becomes possible.