
Mike Brinson ¹, mbrin72043@yahoo.co.uk.
Vadim Kuznetsov ², ra3xdh@gmail.com

¹Centre for Communications Technology, London Metropolitan University, UK

²Bauman Moscow Technical University, Russia

Presented at IEEE ED Mini-Colloquium, Bygoszcz, Poland, 21 June 2017
Qucs-S: Qucs with SPICE

Download links

The latest stable release is Qucs-0.0.19S. It is based on stable Qucs-0.0.19:

- Source tarball: qucs-0.0.19S.tar.gz
- Debian repository (32 and 64 bit packages), built with openSUSE OBS:
  - Debian 8 “Jessy” at: download.opensuse.org
  - Debian 7 “Wheezy” at: download.opensuse.org
- Windows installer (zipped EXE): qucs-0.0.19S-setup.zip

News

- January 26, 2017: Qucs-S 0.0.19 is released!
- November 6, 2016: Qucs-S RC8 released.
- September 3, 2016: Qucs-S RC7 released.
- August 20, 2015: Qucs-S RC3 released.

Simulation example with Qucs-S and Ngspice

What is Qucs-S?

Qucs-S is a spin-off of the Qucs cross-platform circuit simulator. “S” letter indicates SPICE. The purpose of the Qucs-S subproject is to use free SPICE circuit simulation kernels with the Qucs GUI. It merges the power of SPICE and the simplicity of the Qucs GUI. Qucs intentionally uses its own SPICE incompatible simulation kernel Qucsat: It has advanced RF and AC domain simulation features, but most of existing industrial SPICE models are incompatible with it. Qucs-S is not a simulator by itself, but it requires to use a simulation backend with it. The schematic document format of Qucs and Qucs-S are fully compatible. Qucs-S allows to use the same simulation kernels with it:

- Ngspice is recommended to use. Ngspice is powerful mixed-level/mixed-signal circuit simulator. The most of industrial SPICE models are compatible with Ngspice. It has an excellent performance for time-domain simulation of switching circuits and powerful postprocessor.
- XYCE is a new SPICE-compatible circuit simulator written by Sandia from the scratch. It supports basic SPICE simulation types and has an advances RF simulation features such as Harmonic balance simulation.
- SpiceOpus is developed by the Faculty of Electrical Engineering of the Ljubljana University. It is based on the SPICE-3f5 code.
- Qucsat as backward compatible
History of Qucs-S compact device modelling

- 21 Feb 2007: Implementation of subcircuit parameters. Allow equation variables and sweep parameters in component properties and subcircuit parameters. Equations can be placed in subcircuits.
- 24 Feb 2007: Input parameters of components can be used in Qucs-S equations.
- 2 Sep 2007: Allow number engineering notation in equations (pre- and post-processing as well as EDD).
- 2008 to 2011: Generation of Verilog-A compact semiconductor device models.
- 3 Mar 2011: Implementation of interactive GNU Octave connection to Qucs-S.
- 31 Aug 2014: Dynamic compilation and loading of Verilog-A modules, Addition of a full ADMS/Qucs "turn key" Verilog-A compact device modelling system to Qucs. Users are no longer required to manually edit C++ code and build system to be able to run Verilog-A models. Uses ADMS 2.3.4.
- 2014 to 2015: Full set of SPICE commands added, for example .PARAM, and .OPTION.
- 2014 to 2015: Full set of SPICE components added.
- 28 Sept 2015: Qucs PlotVs() function added to Qucs-S.
- 9 March 2016: XSPICE distributed analogue device models and user defined Code Models added.
- 4 Nov 2016: Start developing XSPICE "Turn-Key" Code Model synthesiser.
- 1 Feb 2017: EDD maximum number of branches increased from 8 to 20.
- 5 March 2017: Added .FUNC and .include-script components.
- 2017 ........: Continuing Qucs-S development in preparation for release 0.0.20.
A flow chart showing Qucs-S compact modelling facilities and data movement

NOTES:
1. Qucs-S allows the selection of the simulation engine to use.
2. Available simulation components depends on the simulation engine chosen.
3. Users may select either Qucs-S or Octave post-processing of simulator data.
In this presentation a compact model for a semiconductor step diode has been chosen to illustrate the different model building tools implemented by Qucs-S. This choice of model is deliberate because it’s properties are well known, making the operation of the modelling tools easier to follow and understand.

Non-linear static $I_d-V_d$ characteristics:

\[
I_d = IST2 \cdot (\exp(V_d/(N \cdot Vt(T2)))) - 1.0) + GMIN \cdot V_d,
\]

where

\[
V_d = V(Anode, Cathode),
\]

\[
T1 = TNOM + 273.15,
\]

\[
T2 = TEMP + 273.15,
\]

\[
Vt(T2) = (k \cdot T2)/q,
\]

\[
IST2 = IS \cdot AREA \cdot (T2/T1)^{XTN/N} \cdot \exp(-Eg(300)/Vt(T2)),
\]

\[
Eg(T) = Eg - (7.02e-4 \cdot T \cdot T)/(1108.0 + T),
\]

$k$ is the Boltzmann constant and $q$ the elementary charge. Other physical parameters have their usual meaning: $AREA = 1, N = 1, IS = 1e-14, XTI = 3.0, Eg = 1.16, TNOM = 26.85, TEMP = 26.85$ and $GMIN = 1e-9.$
Building compact device models with Qucs-S: 1 Specification of the static and dynamic device properties of a semiconductor step recovery diode example

- Reverse breakdown voltage:

  \[ K2 = \frac{1.0}{N \cdot Vt(T2)}, \quad K5 = N \cdot Vt(T2), \quad IBVEFF = IBV \cdot AREA \]

  \[ IDBV = -IST2 \cdot (\text{limexp}(-BV \cdot K2) - 1.0), \]

  \[ BVEFF = (IBVEFF > IDBV)?BV - K5 \cdot \ln(\frac{IBVEFF}{IDBV}) : BV, \]

  \[ Id = -IST2 \cdot (\text{limexp}(-(BVEFF - Vd) \cdot K2) - 1.0 + BVEFF \cdot K2), \]

  where the breakdown physical parameters have their usual meaning: \( BV = 4.5 \), and \( IBV = 1e - 3 \).

- Basic semiconductor diode depletion charge:

  \[ Qdep = (Vd >= 0.0)?CJ0T2 \cdot (Vd + P11 \cdot Vd \cdot Vd) : P6 \cdot (1 - (1 - Vd/\mu T)^P7), \]

  where

  \( CJ0T2 = CJ0 \cdot AREA, \quad P11 = M/(2 \cdot VJ), \quad P6 = (CJ0T2 \cdot VJT2)/P7, \)

  \( P7 = 1 - M, \) and

  \( VJT2 = (T2 \cdot VJ)/T1 - 2 \cdot Vt(T2) \cdot \ln(T2/T1)^{1.5} - ((T2 \cdot Eg(T1)/T1) - Eg(T2), \)

  where the depletion capacitor physical parameters have their usual meaning:

  \( CJ0 = 1e - 12, \quad VJ = 1.0 \) and \( M = 0.5 \).
Building compact device models with Qucs-S: 1 Specification of the static and dynamic device properties of a semiconductor step recovery diode example

- **Noise current:**
  \[
i^2 = 2 \cdot q \cdot I_d \cdot \Delta f + \frac{K_f \cdot I_d^A f}{f} \cdot \Delta f + \frac{4 \cdot K \cdot T}{R_s} \cdot \Delta f,
\]
  where the noise physical parameters have their usual meaning: 
  \(K_f = 0.0, A_f = 1.0\).

- **Basic semiconductor diode diffusion charge:**
  \[Q_{diff} = TT \cdot I_d\]

- **Step recovery diode charge:**
  \(Q_d = (V_d \leq 0.0)?CJ0 \ast V_d : 0.0,\)
  \(Q_d = (V_d > 0.0)\&\&(V_d < FCP)?C1 \ast (V_d + C2)^2 - C3 : 0.0,\)
  \(Q_d = (V_d > 0.0)\&\&(V_d > FCP)?C_f \ast V_d - C_f : 0.0,\) where
  \(FCP = FC \ast V_J, C_f = TAU/R_s, C_m = C_f - CJ0, C_1 = C_f - CJ)/2 \ast FCP,\)
  \(C_2 = (CJ0 \ast FCP)/C_n, C_3 = (CJ0 \ast CJ0 \ast FCP)/(2 \ast C_m), C_4 = C_m \ast FCP/2,\)
  where the capacitor physical parameters have their usual meaning:
  \(TAU = 2e - 9, R_s = 0.1, FC = 0.5.\)
Qucs-S equation blocks can be used as a design aid to calculate component values at the start of a simulation sequence. Unlike Qucs, the order of the equations in a Qucs-S block is important. Right hand variables must be calculated before use. Multiple blocks are combined by Qucs.

Adds a design element to subcircuits.

Boyle OP AMP macromodel
Photodiode subcircuit body

Test circuit and simulation data

Photodiode symbol

Green denotes light source and light bus
I = I(V), \( g = \frac{dI}{dV} \)

\[ Q = Q(V,I), \quad C = \frac{dQ}{dV} = \frac{\partial Q(V)}{\partial V} + \frac{\partial Q(I)}{\partial I} \cdot g, \] where

the current flowing in branch \( n \) is \( I_n = I(V_n) + \frac{d}{dt}Q_n \), and \( 1 \leq n \leq 20 \).

### Qucs-S: Nonlinear equation defined devices (EDD)

- EDD is a multiterminal nonlinear component with branch currents that can be functions of EDD branch voltage, and stored charge that can be a function of both EDD branch voltages and currents
- EDD is similar, but more advanced to the SPICE 3f5 B type I or V controlled sources
- EDD can be combined with conventional circuit components and Qucs-S equation blocks when constructing compact device models and subcircuit macromodels
- EDD is an advanced component, allowing users to construct prototype experimental models from a set of equations derived from physical device properties
- EDD operator \( \frac{d}{dt} \) is undertaken internally by Qucs-S
- Qucs-S EDD can have a maximum of 20 two terminal branches

---


Qucs-S: An EDD compact model of a semiconductor diode, including noise

Model
Function
Control
Parameters

BVSWITCH
CdepSWITCH
CdiffSWITCH
ACnoiseSWITCH

\begin{align*}
Q1 &= Q_{\text{dep}} \\
Q2 &= Q_{\text{diff}} \\
I_d &= I_d \quad \text{Breakdown}
\end{align*}

Shot noise

1/f noise

Noise free resistor
Qucs-S: An EDD compact model of a semiconductor diode, typical simulation data

1. I-V characteristics  
[BVSWITCH=0]

2. Reverse breakdown  
[BVSWITCH = 1]

3. Capacitance  
[CdepSWITCH=1, CdiffSWITCH=0]

4. Flicker and shot noise  
[ACnoiseSWITCH=1]
Qucs-S: An EDD compact model of a step recovery semiconductor diode

Q3 represents Cdep

Q4 and Q5 represent Cdiff

Small signal AC shot and 1/f noise model not included

If IrecSWITCH == 1 then CdepSWITCH and CdiffSWITCH over-riden
Qucs-S: Test circuit and simulation data for the step recovery semiconductor diode model

**DC Simulation**
- V1
  - U1 = 0.9
  - U2 = -5
  - T1 = 0.2 ns
  - T2 = 400 ns
  - Tr = 0.001 ns
  - Tf = 0.001 ns

**Transient Simulation**
- TR1
  - Type = lin
  - Start = 0
  - Stop = 5 ns
  - Points = 5000
  - Integration Method = Gear
  - Order = 6

**Simulation Results**
- Vd (V)
  - 0 to -5
  - Time (s) from 0 to 5e-9

- Id (A)
  - 0 to -5
  - Time (s) from 0 to 5e-9

- Parameters:
  - AREA = 1
  - IS = 1e-14
  - N = 1
  - XTI = 3.0
  - TEMP = 26.85
  - Eg = 1.16
  - TNOM = 26.85
  - TT = 1 n
  - CJ0 = 1e-12
  - VJ = 1.0
  - M = 0.5
  - BVSWITCH = 0
  - CdepSWITCH = 1
  - BV = 100
  - CdiffSWITCH = 1
  - IBV = 1e-3
  - IreccSWITCH = 1
  - TAU = 10 n
  - RS = 1
The following diagram illustrates the initial stage in the construction of a Qucs Verilog-A compact device model.
A maximum of 20 two port branches are now allowed per EDD.
The MOT-ADMS software is supplied with little documentation! These brief notes provide a basic introduction to the MOT-ADMS Verilog-A subset.

- Verilog-A is a case sensitive language
- Comments: single line comments start with //, block comments begin with /* and end with */
- Identifiers are sequences of letters, digits, dollar signs '$' and the underscore '_', the first letter of an identifier must not be a digit
- MOT-ADMS version 2.30 keywords: `parameter`, `aliasparameter`, `aliasparam`, `module`, `endmodule`, `function`, `endfunction`, `discipline`, `potential`, `flow`, `domain`, `ground`, `enddiscipline`, `nature`, `endnature`, `input`, `output`, `inout`, `branch`, `analog`, `begin`, `end`, `if`, `while`, `case`, `endcase`, `default`, `for`, `else`, `integer`, `real`, `string`, `from`, `exclude`, `inf`, `INF`
- Compiler directives: `define`, `undef`, `ifdef`, `else`, `endif`, `include`
- Data types: `integers`, `reals` and `strings`
- Predefined constants in “constants.vams”: `M_PI`, `M_TWO_PI`, `M_PI_2`, `M_PI_4`, `M_1_PI`, `M_2_PI`, `M_2_SQRTPI`, `M_E`, `M_LOG2E`, `M_LOG10E`, `M_LN2`, `M_LN10`, `M_SQRT2`, `M_SQRT1_2`, `P_Q`, `P_C`, `P_K`, `P_H`, `P_EPS0`, `P_U0`, `P_CELSIUS0`
- Variables are named objects that contain a value of a particular type. They are initialised to zero or unknown. They retain their value until changed by an assignment statement.
MOT-ADMS: Introduction to the basic Verilog-A subset available with ADMS; continued

- Parameters are declared using statements of the form:
  parameter integer size=16; parameter real period = 1.0 from (0:inf);
  parameter integer dir = 1 from [-1:1] exclude 0;
- Verilog-A natures and disciplines are listed in file “disciplines.vams”
- Port, net and node examples in Verilog-A:
  
  module amp(out1, in1);
  input in1;
  output out1;
  electrical out1, in1;
  
  module amp(out1, in1);
  input in1;
  output out1;
  electrical out1, in1;
- Branches declared with statement branch (n1,n2) b1;
- Signal access function examples: V(n2), I(n), V(b1), I(b1), V(n,m), I(n,m)
- Current contribution examples:
  
  I(diode) <- ls*(limexp(V(diode)/$vt)-1);
  I(diode) <- ddt(-2*cj0*phi*sqrt(1-V(diode)/phi));
- MOT-ADMS allows an extensive range of Verilog-AMS operators and mathematical functions
- Environmental Functions: $temperature, $vt, $strobe, $finish, $given, $parameter_given
- Analogue operators: @(initial_step), @(final_step), @(initial_model), @(initial_instance)
• Analogue behavioural statements:
  1. Analog process/procedural block;
        analog begin
           l(diode) <= ls*(limexp(V(diode)/$vt)-1);
           qd      = tf*l(diode) -2*cj0*phi*sqrt(1-V(diode)/phi);
           l(diode) <= ddt(qd);
        end
  2. Assignment statements consist of a variable followed by = and an expression
  3. Conditional operator cond ? Val1 : Val2, for example
       State = (V(d) > 0.0) ? 1 : -1;
  4. if-else statement:
       If (V(ps,ns) > thresh)
           l(p,n) <= 1;
       else
           l(p,n) <= 0;

  5. Case statement:
       case (select)
          0 : out = l(in0);
          1 : out = l(in1);
          2 : out = l(in2);
          default : out = 0;
       endcase

  6. While statement:
       test = 4;
       While( test ) begin
          test = test-1;
       end

  7. For loops:
       for (i=0; i<10; i=i+1) begin
          ..
          A = A+1;
          ..
          B = B-1;
       end

  8. User defined functions and function calls.
Generating Qucs-S Verilog-A compact device models: original user controlled construction of Verilog-A models using static C libraries

1. Compile Verilog-A template code with ADMS
2. Add new model to Qucs by patching C++ code
3. Add new model symbol to Qucs C++ code
4. Compile and link Qucs **static** C++ code to generate new version of Qucs
Generating Qucs Verilog-A compact device models: C++ code patches; model REGISTRATION process

1. Qucs-core
1.1 Directory ../src/components/verilog
   Modify file Makefile.am
   ADD to libverilog_SOURCES = ...
   XXX.analogfunction.cpp XXX.core.cpp
   ADD to noinst_HEADERS = ...
   XXX.analogfunction.h XXX.defs.h XXX.core.h
   ADD to VERILOG_FILES = ...
   XXX.va
   ADD to "if MAINTAINER_MODE"
   An entry for XXX (use existing code as a template)
1.2 Directory ../src/components
   Modify file components.h
   ADD #include "verilog/XXX.core.h"
1.3 Directory ../src
   Modify file module.cpp
   ADD REGISTER_CIRCUIT(XXX);

2. model symbol
After entering the Verilog-a code for a new model, pressing key F9 will automatically generate a Qucs schematic symbol for the new model. This may be edited using the Qucs drawing tools. On saving the symbol Qucs writes the C++ drawing code for the symbol to file XXX.dat in the working project directory.

3. Qucs model graphics
Copy files XXX.gui.cpp and XXX.gui.h to directory ../qucs/qucs/components as files XXX.cpp and XXX.h respectively.
3.1 File XXX.cpp
(a) Change the XXX.cpp statement
   #include XXX.gui.h to
   #include XXX.h
(b) Change code line Name ="T"; to a more appropriate name, like Name = "BJT";
(c) Replace the symbol drawing statement, at the bottom of the file, with the C++ code held in file XXX.dat.

4. Model bitmap
4.1 Generate a 30x30 pixel bitmap (png format) using Gimp.
4.2 Save XXX.png in Qucs directory ../qucs/qucs/bitmaps.
4.3 Modify file Makefile.am in directory ../qucs/qucs/bitmaps to include Model name XXX.png in "XPMS="..."
Qucs-0.0.19S includes the first release of a GPL Verilog-A synthesis tool for compact device modelling.

- The Qucs-0.0.19-S Verilog-A synthesizer is a basic working version of this new open source ECAD tool.

- Generated synthesized Verilog-A code is relatively basic and has to be optimized manually for speed. However, it is expected that in the future its operation will improve as development of the Qucs synthesizer progresses.

- Circuits and Verilog-A synthesized models can be constructed from the following Qucs/SPICE built in components:
Data flow through the Qucs GPL compact device modelling tool set.

**QUCS FILTER SYNTHESIS**
- Realization: LC ladder (pi-type)
  - Type: Bessel
  - Class: Bandpass
  - Order: 3
  - Fstart: 1 GHz
  - Fstop: 2 GHz
  - Impedance: 50 Ohm

**VERILOG-A MODEL SYNTHESIS**
- Include "disciplines.vams"
- Include "constants.vams"
- module BPF2(P1, P2);
  - inout P1, P2;
  - electrical P1._net0L1, n1, P2._net0L2, _net0L3;
  - analog begin
  - @t(initial_model)
  - begin
  - (l._net0L1) <= ddt(V(_net0L1));
  - (l._net0L1) <= -V(P1);
  - (P1) <= V(_net0L1)(1.793n+1e-20);
  - (P1) <= ddt(V(P1)) * 1.674p;
  - (l._net0L2) <= ddt(V(_net0L2));
  - (l._net0L2) <= V(n1,P2);
  - (n1,P2) <= V(_net0L2)(7.733n+1e-20);
  - (P1,n1) <= ddt(V(P1,n1) * 1.64p);
  - (l._net0L3) <= ddt(V(_net0L3));
  - (l._net0L3) <= -V(P2);
  - (P2) <= V(_net0L3)(1.86n+1e-20);
  - (P2) <= ddt(V(P2)) * 7.014p;
  - end
  - endmodule

**QUCS/ADMS VERILOG-A "TURN KEY" COMPILER**
- xxxx.va
- xxxx.va.cpp

**DEVELOP TEST CIRCUIT, SIMULATE, AND EVALUATE OUTPUT DATA**
- Create circuit schematic and simulate
- dc simulation
- S parameter simulation
- Edit text symbol

**Plotted and tabulated simulation data**
Synthesis of a SPICE like compact semiconductor diode model: static $I_d$ and dynamic capacitance model plus synthesized Verilog-A module code.

Equation

\[
\begin{align*}
E1 & = Vt \ln(\frac{V1}{Vt}) \cdot (1-N) \\
E2 & = Vt \ln(\frac{V1}{Vt}) \\
F1 & = (N-M)^2 \ln(1-Fc) \\
F2 & = (N-M)^2 \ln(1-Fc) \\
F3 & = 1-Fc \\
D1 & = \text{Area}^2 \ln(\frac{V1}{Vt}) \cdot (1-N) \\
Q1 & = (V1 < Con2) \cdot (V1^2 + \text{Area} \cdot (N-M)^2 \ln(1-Fc) + (1-N-M)^2 \ln(1-Fc) + (1-M)^2 \ln(1-Vt)) \\
Q2 & = 0
\end{align*}
\]

Diode subcircuit EDDdiode2.sch

```verilog
module EDDdiode2 (Panode, Pcathode);
    input Panode, Pcathode;
    electrical Pcathode, n2, n1, Panode, n4, n3;
    parameter real Area=1;  // parameter real Is=1e-14;
    parameter real Rs=0.1;  // parameter real N=1;
    parameter real Temp=26.85;  // parameter real Vt=1;
    parameter real Fc=0.5;  // parameter real M=0.5;
    parameter real Cj=1e-12;  // parameter real Ti=1e-12;
    real RMAX, Vt, Con1, Con2, F1, F2, F3;
    analog begin
        @ (initial_model) begin
            RMAX = 1e15;
            Vt = (K * (Temp + 273.15)) / P_Q;
            Con1 = 1e5 * Vt;
            Con2 = Fc * Vt;
            F1 = (V1 - (1 - M) * ln(1 - Fc));
            F2 = (1 - M) * ln(1 - Fc);
            F3 = 1 - Fc;
            n4 = Pcathode;
            n2 = Panode;
            n3 = Panode;
            n1 = Pcathode;
            end
        endmodule
```

Synthesis of a SPICE like semiconductor diode model: simulated static and dynamic characteristics.
Verilog-A synthesis of a SPICE like semiconductor diode model: temperature effects

Qucs EDD diode model with temperature effects

Synthesized Verilog-A code

```
include "disciplines.vms"
include "constants.vms"

module EDDdiode31(Poathode, Panode);
    inout Poathode, Panode;
    electrical Poathode, n2, n1, Panode, n1, n3;
    parameter real Area=1;  parameter real Ia=1e-14;  parameter real Rs=0.1;  parameter real N=1;
    parameter real Temp=26.85;  parameter real Vf=1.0;  parameter real Fc=0.5;  parameter real M=0.5;
    parameter real Cj1=1e-12;  parameter real Th=1e-9;  parameter real Trnom=26.85;  parameter real Eg=1.1;  parameter real RT=1.0;
    parameter real VT=1.5;  parameter real Rj=1.0;
    real RMSAX,1e12;
    T1=Trnom/273.15;
    T2=Temp/273.15;
    Con=1.0*ln(Vf);
    Con=1.0*ln(Vf);
    F2=exp(1.0*M*(n1+1.0));
    F3=1.0+Fc(1.0+M);
    A=7.02e-4;
    B=1.108;
    Eqg,T1=Eg-A*1.0*(1.0+1.0)/B
    Eqg,T2=Eg-A*1.0*(1.0+1.0)/B
    Vj,T2=Trnom/273.15*(1.0-4000.0-6.0)*(1.0+1.0)/B
    Vj,T2=Trnom/273.15*(1.0-4000.0-6.0)*(1.0+1.0)/B
    Is,T2=4.0*(X1)/((X1-X2))
    Is,T2=4.0*(X1)/((X1-X2))
end module
```

26 / 33
Verilog-A synthesis of a SPICE like semiconductor diode model: simulated $I_d - V_d$ temperature effects.

Simulation data for
- Qucs EDD model and built-in diode model
- Verilog-A model and built-in diode model
Verilog-A synthesis of semiconductor device shot and flicker noise: EDD models and Verilog-A module code.
Verilog-A synthesis of semiconductor device shot and flicker noise: small signal AC domain simulation data.

![Diagram of Verilog-A circuit with simulation parameters and graphs showing shot and flicker noise over frequency and current]

**Simulation Parameters**
- **AC1**
  - Type: log
  - Start: 0.001
  - Stop: 100k
  - Points: 141
  - Noise: yes

- **DC1**
  - SW1
  - Sim: AC1
  - Type: lin
  - Start: 0
  - Stop: 1
  - Points: 11
Verilog-A synthesis of multi-EDD models: EKV2p6 nMOS

\[ I_{ds} = f(V_d, V_g, V_s, V_b) \] model for a transistor operating in long channel mode.
Verilog-A synthesis of multi-EDD models: EKV2p6 nMOS

\[ I_{ds} = f(V_d, V_g, V_s, V_b) \] swept DC simulation data.
Introduction to the Qucs GPL Verilog-A module synthesizer: Part XII


```
#include "idsrnmos.vane"
#include "constants.vane"
module EKV_VA_OPT(PB, PD, P0, P5);
output PB, PD, P0, P5;
input PB, PD, P0, P5;
electrical PB, PD, P0, P5, P11;
parameter real L=10u; parameter real W=20u;
parameter real VTO=0.5; parameter real GAMMA=0.7;
parameter real PHI=3.5; parameter real KP=20e-6;
parameter real THETA=50e-3; parameter real Temp=26.85;
real TempK, VTO, PHI, GAMMA, K1, K2, K3, K4, K5, K7, K8;
real Vg, Vs, Vd, nVP, nETA, nif, nfr;

// Analog model

module Initial_model

begin
  TempK=Temp/273.15; KS=K*TempK; K1=GAMMA/2;
  K2=VTO+PHI*GAMMA*sqrt(1+PHI); K3=K1*K1;
  K4=PHI*4*VT; K5=1/(2*VT); K7=2*VT*VT;
  Vg = V(PD,P0); Vs = V(PS,P0); Vd = V(PD,PB);
nVP = (Vg+K2)/2*(Vg+K2-2*GAMMA*(sqrt(Vg+K2+K3)*K1)); PHI:
nETA = K5/(1+THETA)*VT;
nif = (1+linexp(VP-Va)*K1)/(1+linexp(VP-Va)*K1);
nfr = (1+linexp(VP-Va)*K1)/(1+linexp(VP-Va)*K1);
  (PD,P0) = K7*n1*YBETA*Vd*nfr;
endmodule
```

**Notes:**

1. At this stage in the development of the Qucs synthesizer optimized Verilog-A module code is done manually.
2. General procedure:
   2.1 Reduce current contribution statements to a minimum. This can be done by representing model equation quantities by real variables rather than internal node voltages.
   | one I(a) <= ... in the EKV nMOS example |
   2.2 Eliminate as many as possible internal model nodes and remove current to voltage one Ohm conversion resistors.
   | zero left in EKV nMOS example |

**Test Module**

**DC Simulation**

**Parameter sweep**

**Simulation run time tests** indicate that the optimized EKV2p6 Verilog-A model simulation speed is at least 30X faster than the Interactive EDD model.
An outline of Qucs-S compact device modelling: History and capabilities: Part 1– From Equation-Defined Device (EDD) modelling to Verilog-A module synthesis

End of Part 1