Verilog-A compact modelling of SiC devices with Qucs-S, QucsStudio and MAPP/Octave FOSS tools

Mike Brinson [1], mbrin72043@yahoo.co.uk.

[1]Centre for Communications Technology, London Metropolitan University, UK

Presented at IEEE EDS DL Mini-Colloquium, Gdynia, Poland, 20 June 2018

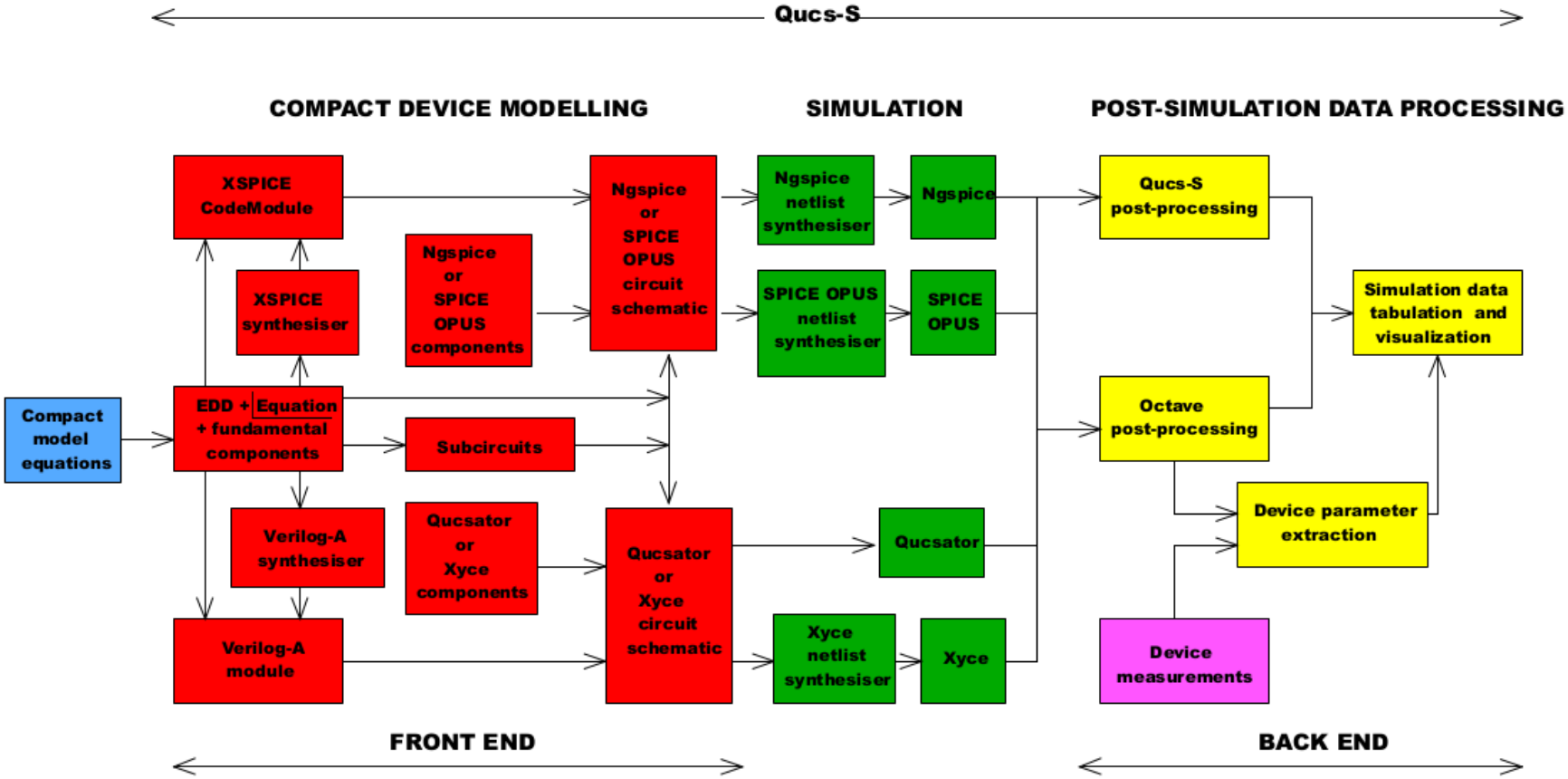
An outline of compact device modelling tools implemented by FOSS tools:1

Simulator	Simulation and output commands	Components/models	Modelling features
Berkeley SPICE	.OP, .DC, .TF, .AC, .TRAN, .NOISE., .DISTO, .PZ, .SENS, .SAVE, .PRINT, .PLOT, .FOUR, .SUBCKT .... .ENDS Berkeley Nutmeg simulation data post processor.	R, C, L, M, S, W, V, I, T, LTRA, U, URC, D, BJT, J, M, Z, B	1. Fully expanded circuits, 2. Subcircuits[A], 3. Macromodels[B], 4. C compiled code models[C].
Qucs	DC, AC, transient, harmonic balance[D], Parameter sweep, equations, S-parameter analysis, Qucs and Octave post simulation data processors. Noise analysis. Optimization.	Similar to Berkely SPICE plus RF component and device models, VHDL and Verilog Digital models.	1., 2. and 3. the same as SPICE, 4. EDD and RFEDD behavioural modelling 5. Verilog-A code models[E].
QucsStudio	DC, AC, transient, harmonic balance (including large signal AC and noise), equations, S parameter analysis, Qucs and Octave post simulation data processors, Parameter sweep, transient shooting method periodic steady-state simulation, Monte Carlo analysis	Similar to Berkely SPICE plus RF component and device models, VHDL and Verilog Digital models. Communication system models	1., 2. and 3. the same as SPICE, 4. EDD and RFEDD behavioural modelling 5. Verilog-A and C++ code models[E].

NOTES

A : Linear and non-linear device models based on subcircuits.
B: Linear and non-linear device models constructed from existing component models.
C: SPICE C code model API .
D: Incomplete single tone Harmonic Balance simulation feature.
E : ADMS Verilog-A compact modelling feature with "Turn-key" capabilities.

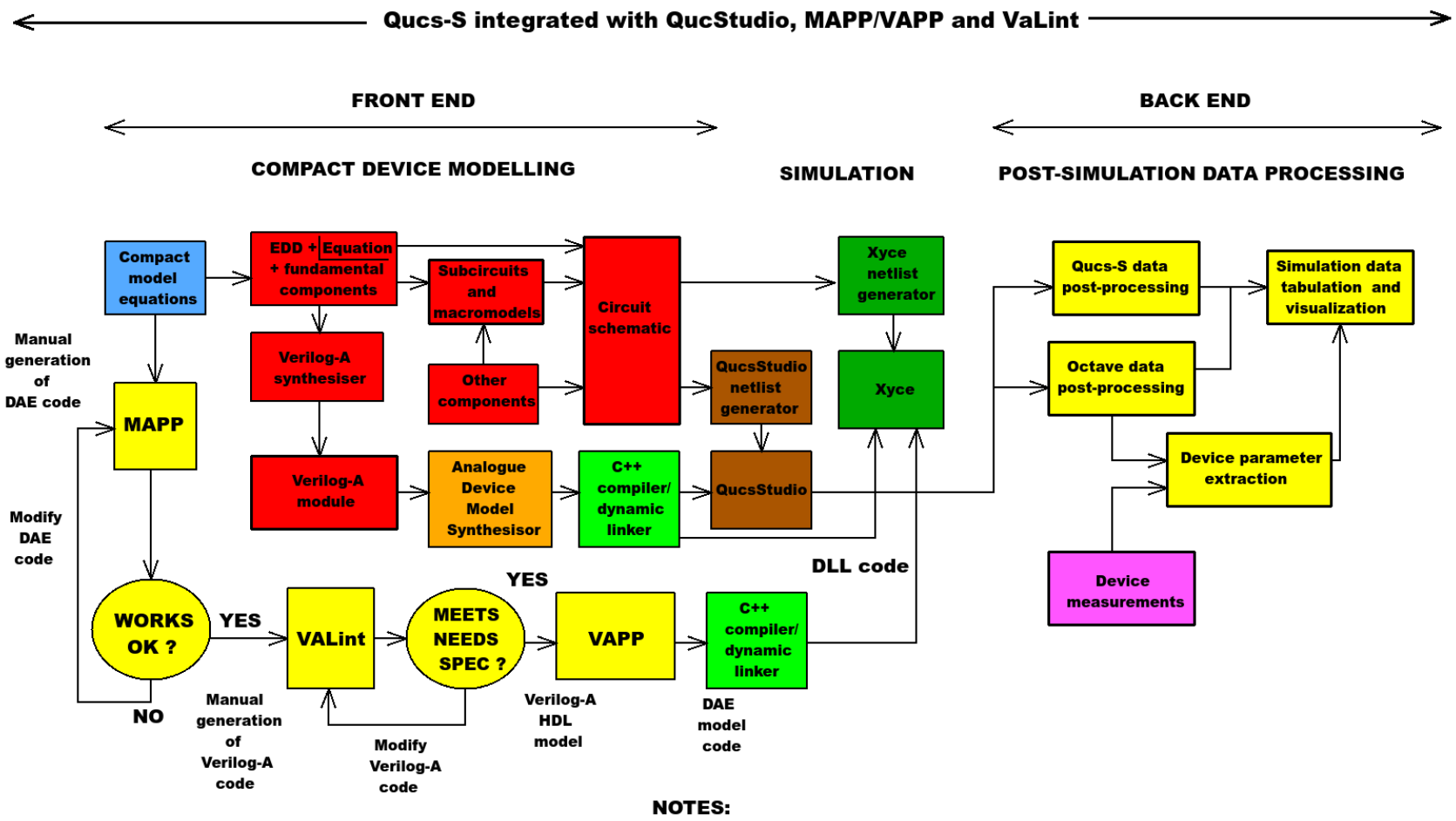
An outline of compact device modelling tools implemented by FOSS tools:2



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.

Qucs-S, QucsStudio and Xyce/MAPP/VAPP/VAlint Verilog-A Compact modelling tools



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.
4. Berkeley MAPP/VAPP and VaLint requires that Octave 4.0 or greater is installed.

More detailed instructions for Verilog-A and DAE model generation and a number of examples are presented in later slides.

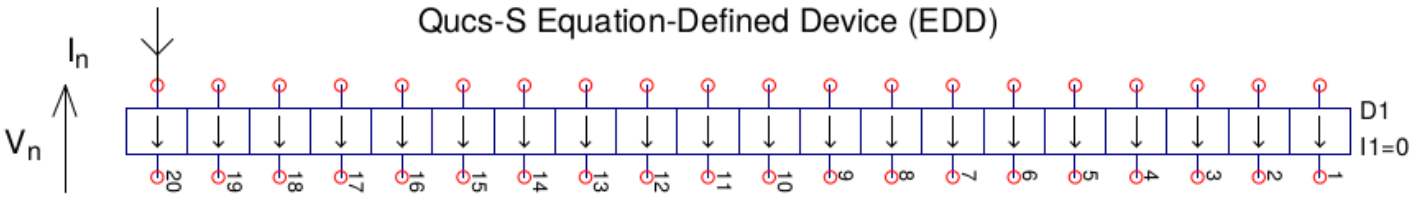
Building compact device models: specify the physical properties of a device as a set of equations and parameters: introductory example - the static and dynamic device properties of a semiconductor diode

- This choice of device is deliberate because its properties are well known, making the operation of the modelling tools easier to follow and understand.
- (1) Non-linear static Id-Vd characteristics :
  $Id = IST2 \cdot (exp(Vd/(N \cdot Vt(T2)))) - 1.0) + GMIN \cdot Vd$, where
  $Vd = V(Anode, Cathode)$,
  $T1 = TNOM + 273.15$,
  $T2 = TEMP + 273.15$,
  $Vt(T2) = (k \cdot T2)/q$,
  $IST2 = IS \cdot AREA \cdot (T2/T1)^{XTN/N} \cdot exp(-Eg(300)/Vt(T2))$,
  $Eg(T) = Eg - (7.02e-4 \cdot T \cdot T)/(1108.0 + T)$, here
  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: continued

- (2) Reverse breakdown voltage : device model with Qucs-S Equation-Defined Devices (EDD)
  $K2 = 1.0/(N \cdot Vt(T2)), K5 = N \cdot Vt(T2), IBVEFF = IBV \cdot AREA$
  $IDBV = -IST2 \cdot (limexp(-BV \cdot K2) - 1.0)$,
  $BVEFF = (IBVEFF > IDBV)?BV - K5 \cdot ln(IBVEFF/IDBV) : BV$,
  $Id = -IST2 \cdot (limexp(-(BVEFF - Vd) \cdot K2) - 1.0 + BVEFF \cdot K2)$, where the breakdown physical parameters have their usual meaning: $BV = 4.5$, and $IBV = 1e - 3$.
- (3) Semiconductor diode depletion charge :
  $Qdep = (Vd >= 0.0)?CJ0T2 \cdot (Vd + P11 \cdot Vd \cdot Vd) : P6 \cdot (1 - (1 - Vd/JT2)^{P7})$, where
  $CJ0T2 = CJ0 \cdot AREA, P11 = M/(2 \cdot VJ), P6 = (CJ0T2 \cdot VJT2)/P7$,
  $P7 = 1 - M$, 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:
  $CJ0 = 1e - 12, VJ = 1.0$ and $M = 0.5$.

Building compact device models with Qucs-S: compact device modelling with Equation-Defined Devices (EDD)



$I = I(V)$, $g = dI/dV$

$Q = Q(V,I)$, $C = dQ/dV = \partial Q(V)/\partial V + \partial Q(I)/\partial I \cdot g$, where

the current flowing in branch n is $I_n = I(V_n) + d/dt(Q_n)$, and $1 <= n <= 20$.

- 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 d/dt is undertaken internally by Qucs-S
- Qucs-S EDD can have a maximum of 20 two terminal branches

Building compact device models with Qucs-S: compact device modelling with Equation-Defined Devices (EDD). Diode static and dynamic models plus noise

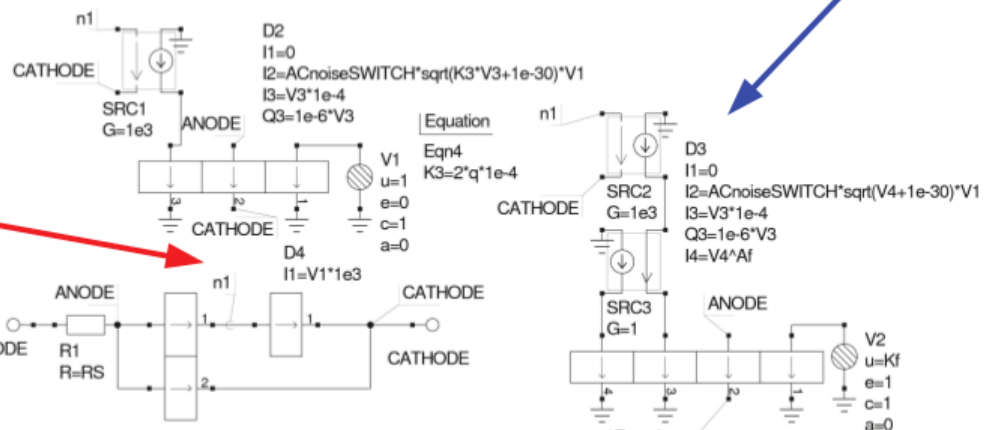
Model
Function
Control
Parameters

BVSWITCH
CdepSWITCH
CdiffSWITCH
ACnoiseSWITCH

Shot noise

1/f noise

Noise free resistor



$Q1 = Qdep$

$Q2 = Qdiff$

I_d

Breakdown I_d

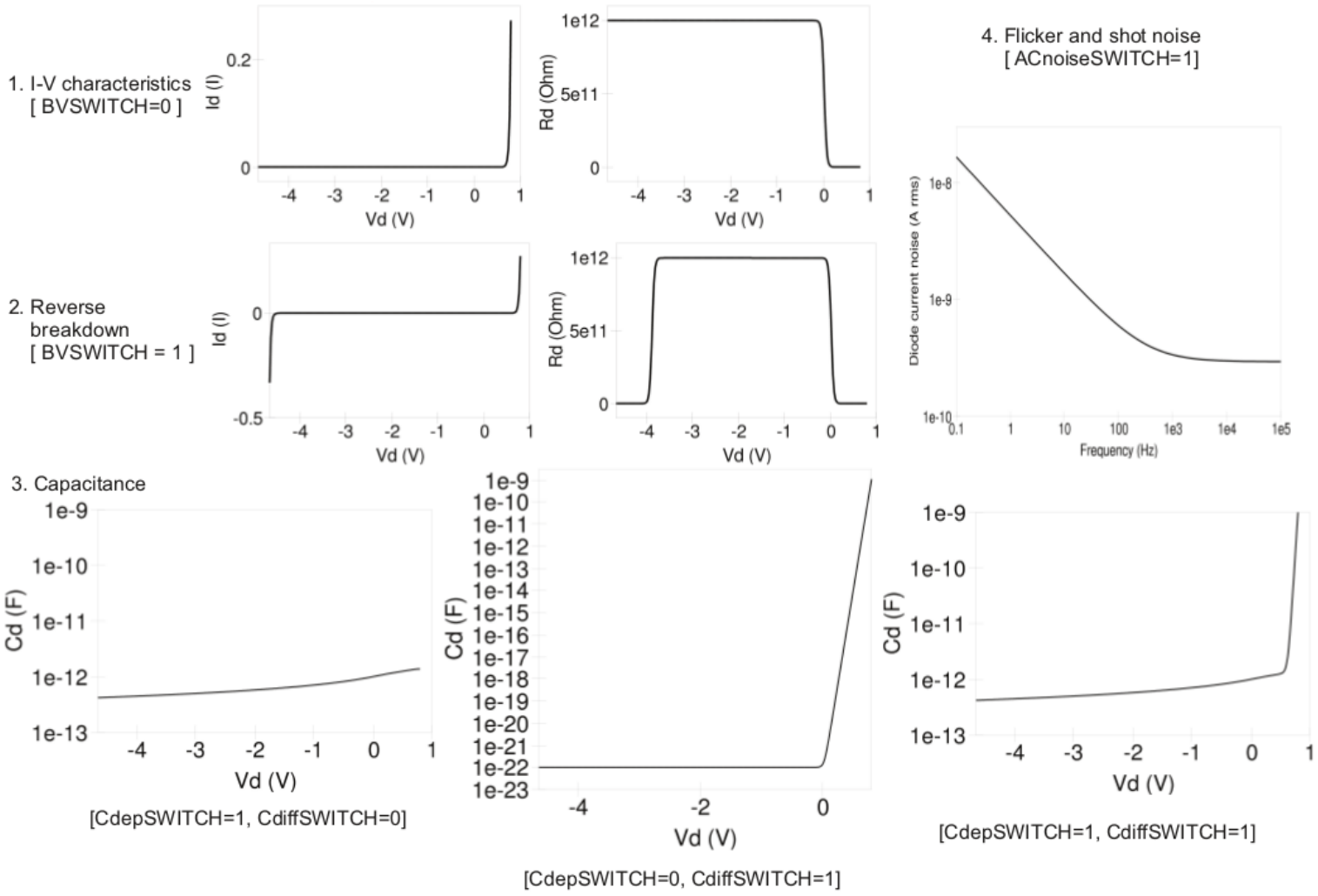
D1
I1= IST2*(limexp (V1*K2)-1.0)+GMIN*V1
Q1=(CdepSWITCH==1) ? ((V1 >=0.0) ? CJ0T2*(V1+P11*V1*V1) : P6*(1-(1-V1/VJT2)^P7)) : 0
I2=(BVSWITCH==1) ? (-IST2*(exp(-(BVEFF+V1)*K2)-1.0 +BVEFF*K2)) : 0
Q2=(CdiffSWITCH==1) ? TT*I1 : 0

Equation
Eqn3
P6=CJ0T2*VJT2/P7
P7=1-M
K5=N*VTT2
IBVEFF=IBV*AREA
IDBV=-IST2*(exp(-BV*K2)-1.0)
BVEFF=(IBVEFF>IDBV) ? BV-K5*ln(IBVEFF/IDBV) : BV
P11=M/(2*VJ)

Equation
Eqn1
GMIN=1e-12
ISEFF=IS*AREA
K2=1/(N*VTT2)
CJ0T2=CJ0*AREA

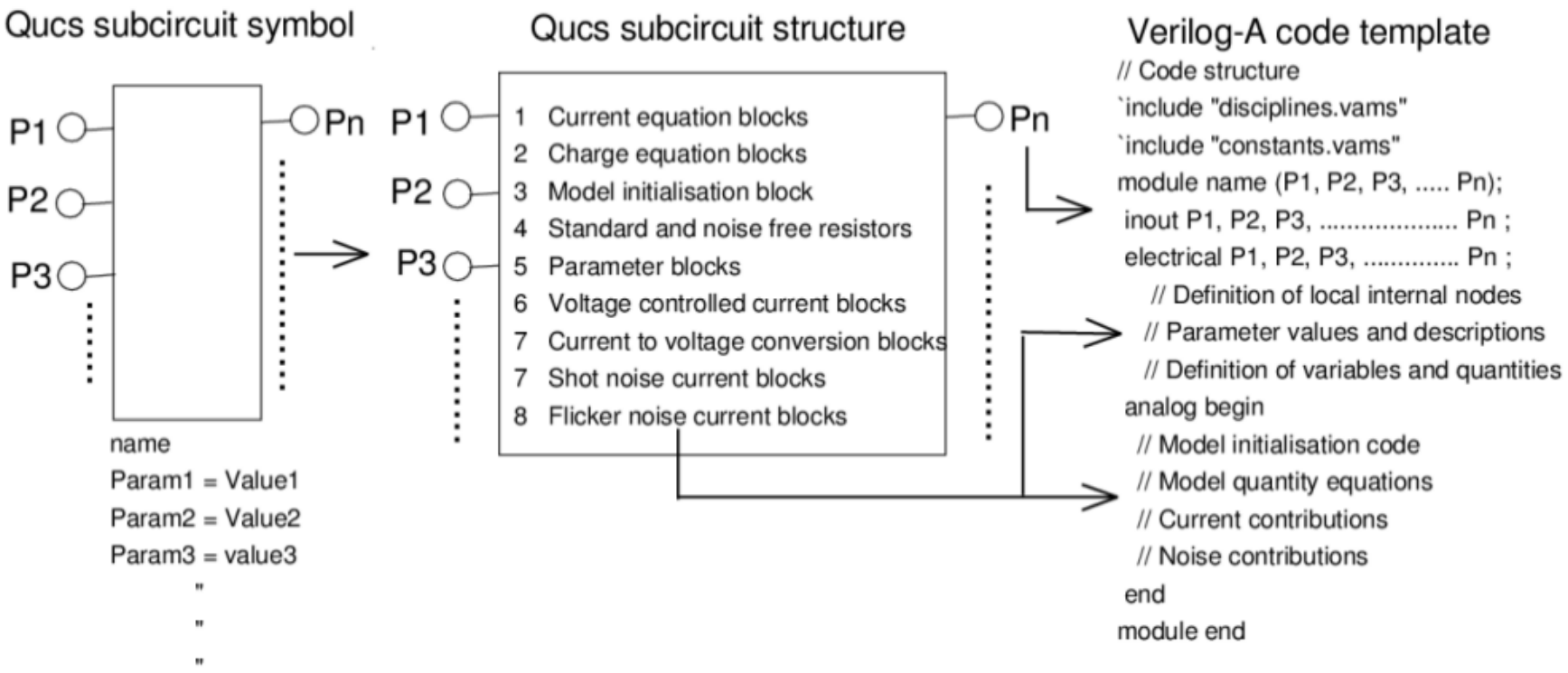
Equation
Eqn2
T1=TNOM+273.15
VJT2= (T2*VJ/T1)-2*VTT2*ln(T2/T1)^1.5-((T2*EgT1/T1)-EgT2)
VTT2=PK*T2/PQ
EgT1=Eg-7.02e-4*T1*T1/(1108+T1)
EgT2=Eg-7.02e-4*T2*T2/(1108+T2)
T2=TEMP+273.15
IST2= ISEFF*(T2/T1)^(XTI/N)*exp((-EgT1/VTT2)*(1-T2/T1))
PQ=1.602176462e-19
PK=1.3806503e-23

Building compact device models with Qucs-S: compact device modelling with Equation-Defined Devices (EDD). Typical test data



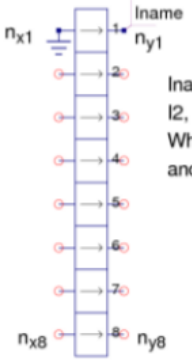
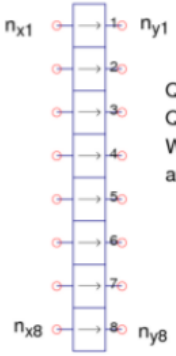
Building compact device models with Qucs-S: generating Verilog-A compact device models

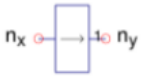
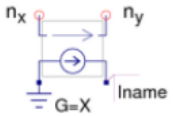
- The following diagram illustrates the initial stage in the construction of a Qucs Verilog-A compact device model.



Building compact device models with Qucs-S: relationships between schematic symbols and Verilog-A code fragments

Fundamental EDD blocks

Qucs symbol	Quantity equations	Verilog-A code fragment
	Iname = I1 = f(V2, V3, ... V8) I2, I3, ... I8 = 0 and Q1, Q2, ... Q8 = 0. Where Vm = V(n_{xm}, n_{ym}) or Vm = V(n_{xm}), and 2 ≤ m ≤ 8.	Iname = f(V2, V3, ... V8); Or Iname <+ f(V2, V3, ... V8);
	Q1 = f(V1, V2, ... V8, I1, I2, ... I8) Q2, ... Q8 = 0. Where Vm = V(n_{xm}, n_{ym}) or Vm = V(n_{xm}), and 1 ≤ m ≤ 8.	I(n_{x1}, n_{y1}) <+ ddt(Q1); Or I(n_{x1}, n_{y1}) = ddt(Q1);

	Quantity equations	Verlog-A code fragment
(a) Model initialisation block	Equation Eqn1 con1= ..... con2= ..... con3=	@(initial_model) begin con1 = ...; con2 = ...; con3 = ...; end
(b) Standard resistors		I(n_x, n_y) <+ V(n_x, n_y)/R; I(n_x, n_y) <+ white_noise((FourKT/R, "thermal"); Where FourKT = 4.0 · `P_K · $temperature, and `P_K = 1.3806505e-23 K^{-1}, $temperature is the resistor temperature in Kelvin.
(c) Noise free resistors	 I1 = V1/R = V(n_x, n_y)/R	I(n_x, n_y) <+ V(n_x, n_y)/R;
(d) Voltage controlled current block	 Iname = X · V(n_x, n_y)	Iname = X*V(n_x, n_y);
(e) current to voltage conversion block	 Iname = I1 = V1	Iname = V(n_x);

Building compact device models with Qucs-S: MOT-ADMS - introduction to the basic available Verilog-A subset

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: **integer**s, **real**s and **string**s
- 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.

Building compact device models with Qucs-S: MOT-ADMS - introduction to the basic available Verilog-A subset; 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 ae listed in file “disciplines.vams”
- Port, net and node examples in Verilog-A:

```
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) <+ Is*(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)**

Building compact device models with Qucs-S: MOT-ADMS - introduction to the basic available Verilog-A subset; continued

- Analogue behavioural statements:

1. Analog process/procedural block;

```
analog begin
   I(diode) <+ Is*(limexp(V(diode)/$vt)-1);
   qd       =  tf*I(diode) -2*cj0*phi*sqrt(1-V(diode)/phi);
   I(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)
    I(p,n) <+ 1;
else
    I(p,n) <+ 0;
```

TRUE = non-zero value

5. Case statement:

```
case (select)
  0 : out = I(in0);
  1 : out = I(in1);
  2 : out = I(in2);
  default : out = 0;
endcase
```

6. While statement:

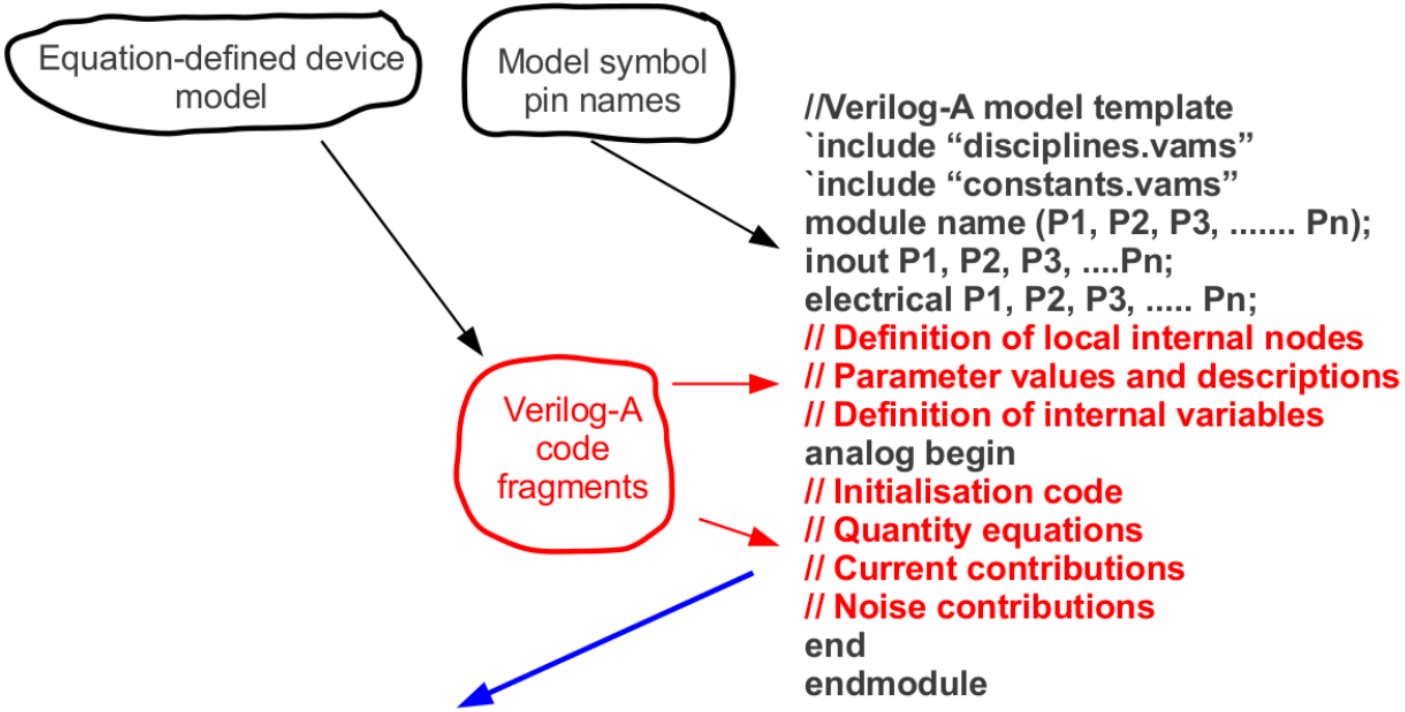
```
test = 4;
While( test) begin
  A = A+1;
  B = B-1;
  test = test-1;
end
```

7. For loops:

```
for (i=0; I <10; i=i+1) begin
                        ..
                        ..
end
```

8. User defined functions and function calls.

Building compact device models with Qucs-S: generating Verilog-A compact device models; the original Qucs 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

Building compact device models with Qucs: implementing Qucs Verilog-A compact device models with C++ patches; the model REGISTRATION process

Compiling XXX.va with ADMS-2.30 results in files →

XXX.core.cpp, XXX.core.h
XXX.defs.h
XXX.gui.cpp, XXX.gui.h
XXX.analogfunction.cpp

Qucs uses static C++ libraries

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);**

REGISTER NEW MODEL

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= ..."

5. Qucs

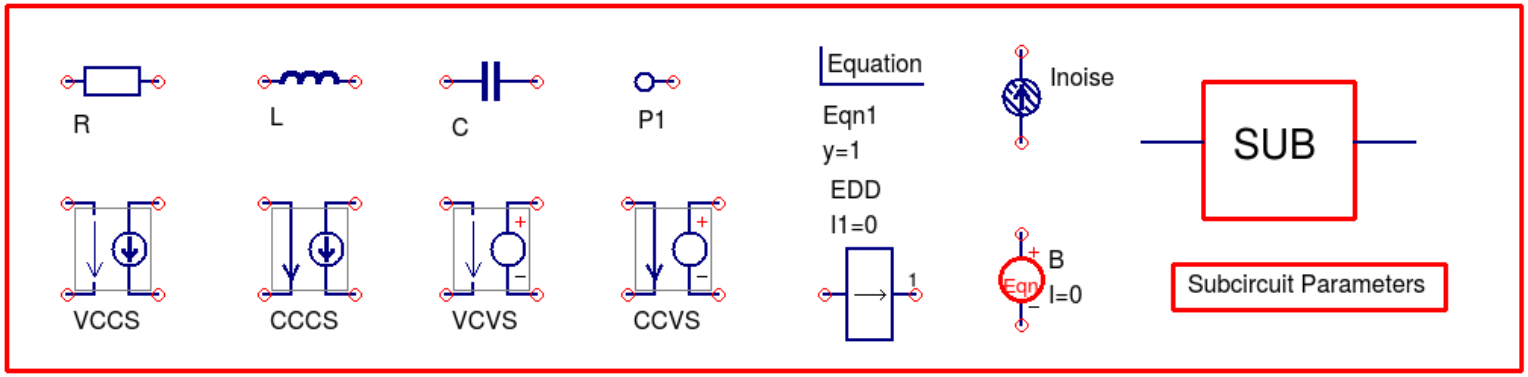
5.1 Directory ../qucs/qucs/components
Modify file Makefile.am
ADD to libcomponents_a_SOURCES = ...
XXX.cpp
ADD to noinst_HEADERS = ...
XXX.h

5.2 Directory ../qucs/qucs/components
Modify file components.h
ADD **#include "XXX.h"**

5.3 Directory ../qucs/qucs
Modify file module.cpp
ADD **REGISTER_VERILOGA_1(XXX);**

Introduction to the Qucs GPL Verilog-A module synthesizer: part I

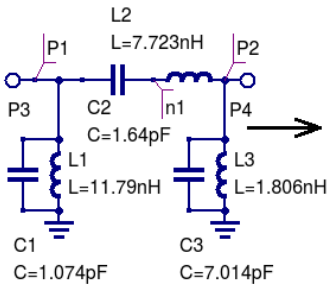
- The Qucs-S-0.0.20 Verilog-A synthesizer is the latest version of this new open source ECAD tool.
- Generated synthesized Verilog-A code is basic and has to be optimized manually for speed, if required. However, it is expected that in the future its operation will improve as development of the synthesizer progresses.
- The synthesized Verilog-A code can be interchanged by Qucs, QucsStudio, Xyce and Berkeley MAPP/VAPP.
- Synthesized circuits and models can be constructed from the following Qucs-S/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



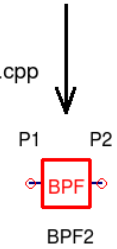
VERILOG-A MODEL SYNTHESIS

```
`include "disciplines.vams"
`include "constants.vams"
module BPF2(P1, P2);
inout P1, P2;
electrical P1, _net0L1, n1, P2, _net0L2, _net0L3;
analog begin
@(initial_model)
begin
end
I(_net0L1) <+ ddt(V(_net0L1));
I(_net0L1) <+ (-V(P1));
I(P1) <+ V(_net0L1)/(11.79n+1e-20);
I(P1)  <+ ddt( (-V(P1)) *  1.074p  );
I(_net0L2) <+ ddt(V(_net0L2));
I(_net0L2) <+ V(n1,P2);
I(n1,P2) <+ V(_net0L2)/(7.723n+1e-20);
I(P1,n1)  <+ ddt( V(P1,n1) *  1.64p  );
I(_net0L3) <+ ddt(V(_net0L3));
I(_net0L3) <+ (-V(P2));
I(P2) <+ V(_net0L3)/(1.806n+1e-20);
I(P2)  <+ ddt( (-V(P2)) *  7.014p  );
end
endmodule
```

Build Verilog-A module from subcircuit

QUCS/ADMS VERILOG-A "TURN KEY" COMPILER

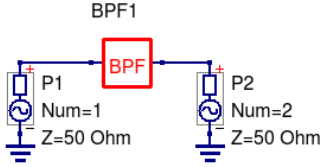
xxxx.va → ADMS → xxxx.va.cpp



Edit text symbol

DEVELOP TEST CIRCUIT, SIMULATE, AND EVALUATE OUTPUT DATA

Create circuit schematic and simulate



dc simulation

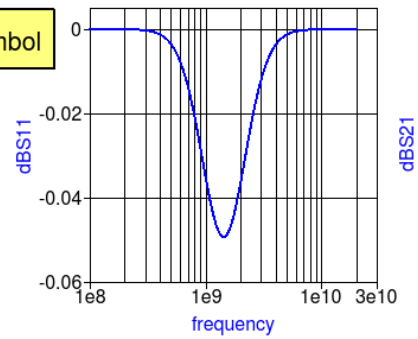
DC1

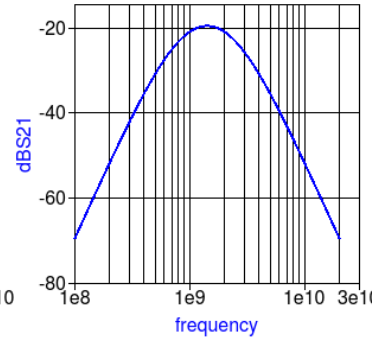
Equation

Eqn1
dBS21=dB(S[2,1])
dBS11=dB(S[1,1])

S parameter simulation

SP1
Type=log
Start=100MHz
Stop=20GHz
Points=201

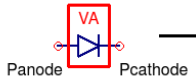




Plotted and tabulated simulation data

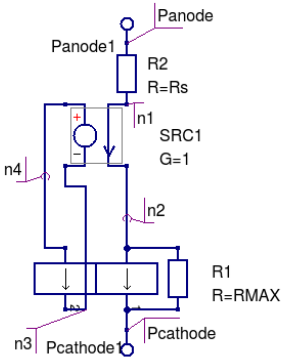
Synthesis of a SPICE like compact semiconductor diode model: static I_d and dynamic capacitance model plus synthesized Verilog-A module code.

EDDdiode21
Area=1
Is=1e-14
Rs=0.1
N=1
Temp=26.85
Vj=1.0
Fc=0.5
M=0.5
Cj0=1e-12
Tt=1e-12



Click on File tab

Build Verilog-A module from subcircuit



Equation

Eqn1
RMAX=1e15
Vt=(kB*(Temp+273.15))/q
Con1=5*N*Vt
Con2=Fc*Vj
F1=(Vj/(1-M))*(1-exp((1-M)*ln(1-Fc)))
F2=exp((1+M)*ln(1-Fc))
F3=1-Fc*(1+M)

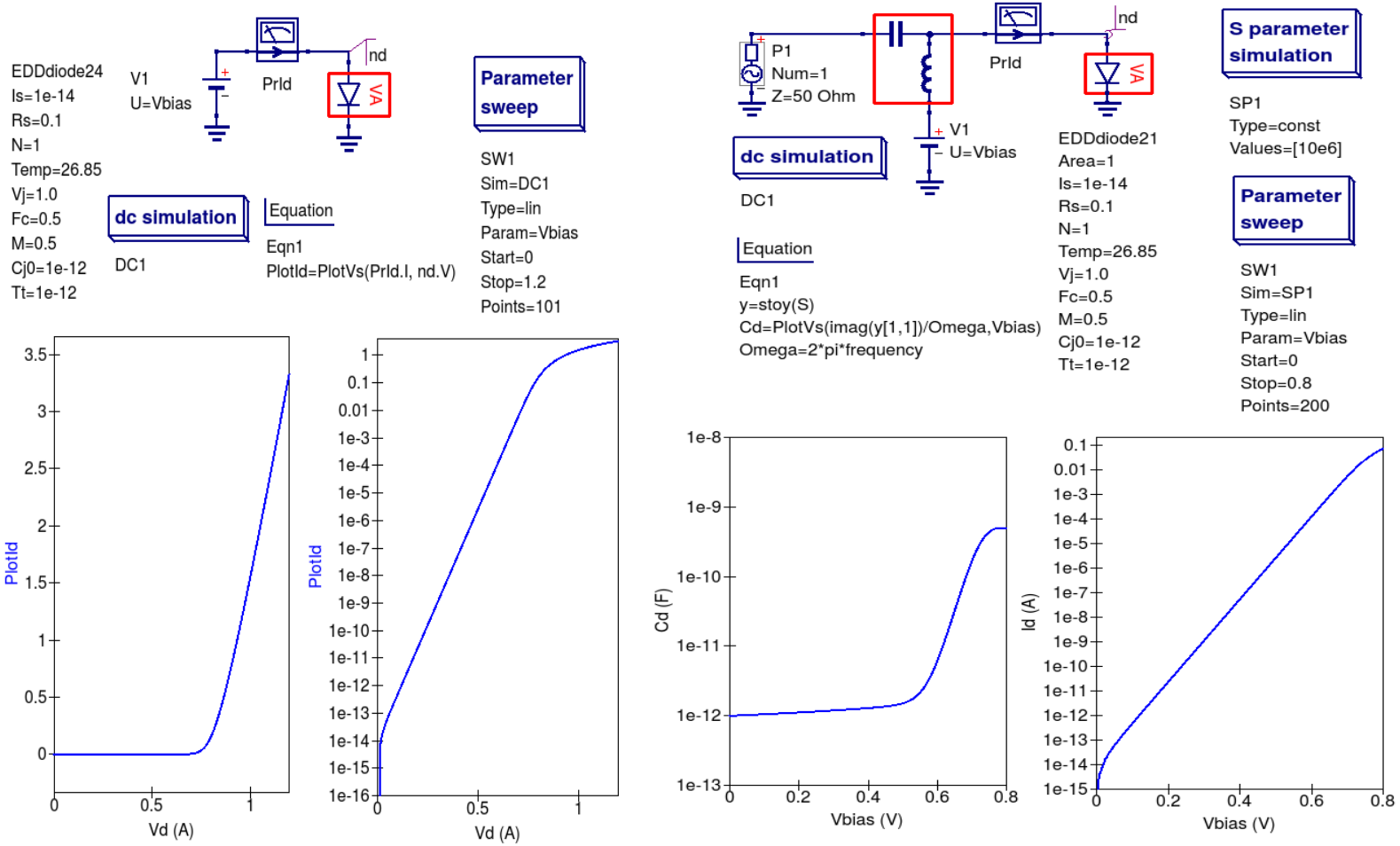
D1
I1= Area*Is*(limexp(V1/(N*Vt))-1)
Q1= (V1 <= Con2) ? Tt*V2+Area*(Cj0*Vj/(1-M))*(1-exp((1-M)*ln(1-V1/Vj))) : Tt*V2+Area*Cj0*(F1+(1/F2)*(F3*(V1-Fc*Vj)+(M/(2*Vj))*(V1*V1-Fc*Fc*Vj*Vj)))
I2=0
Q2=0

Diode subcircuit EDDdiode2.sch

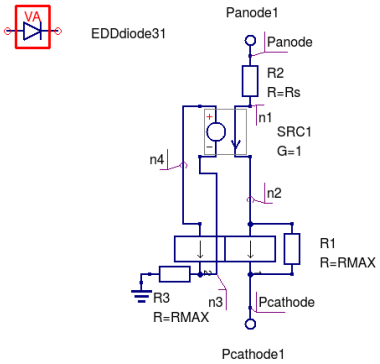
```
`include "disciplines.vams"
`include "constants.vams"
module EDDdiode2(Panode, Pcathode);
inout Panode, Pcathode;
electrical Pcathode, n2, n1, Panode, n4, n3;
parameter real Area=1;          parameter real Is=1e-14;
parameter real Rs=0.1;          parameter real N=1;
parameter real Temp=26.85;      parameter real Vj=1.0;
parameter real Fc=0.5;          parameter real M=0.5;
parameter real Cj0=1e-12;       parameter real Tt=1e-12;
real RMAX, Vt, Con1, Con2, F1, F2, F3;
analog begin
@(initial_model)
begin
RMAX=1e15;     Vt=(`P_K*(Temp+273.15))/`P_Q;
Con1=5*N*Vt;   Con2=Fc*Vj;
F1=(Vj/(1-M))*(1-exp((1-M)*ln(1-Fc)));    F2=exp((1+M)*ln(1-Fc));
F3=1-Fc*(1+M);
end
I(Pcathode,n2) <+ V(Pcathode,n2)/( RMAX );
I(n1,Panode) <+ V(n1,Panode)/( Rs );
I(n1,Panode) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( Rs ), "thermal" );
I(n1,n2) <+ V(n1,n2) * 1e3;  I(n3,n4) <+ I(n3,n4) *1e-3;   I(n3,n4) <+ V(n1,n2) * 1e6 * 1 ;
I(n2,Pcathode) <+ Area*Is*(limexp(V(n2,Pcathode)/(N*Vt))-1);
I(n2,Pcathode) <+ ddt( (V(n2,Pcathode)<=Con2) ? Tt*V(n4,n3)+Area*(Cj0*Vj/(1-M))*
                (1-exp((1-M)*ln(1-V(n2,Pcathode)/Vj)))
               : Tt*V(n4,n3)+Area*Cj0*(F1+(1/F2)*(F3*
                V(n2,Pcathode)-Fc*Vj)+(M/(2*Vj))*
                V(n2,Pcathode)*V(n2,Pcathode)-Fc*Fc*Vj*Vj))) );
end
endmodule
```

Introduction to the Qucs GPL Verilog-A module synthesizer: part V

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



Equation

```
Eqn1
RMAX=1e12
T1=Tnom+273.15
T2=Temp+273.15
Con1=5*N*Vt
Con2=Fc*Vj
Vt=kB*T2/q
F1=(Vj/(1-M))*(1-exp( ( 1-M)*ln(1-Fc)))
F2=exp( (1+M)*ln(1-Fc))
F3=1-Fc*(1+M)
A=7.02e-4
B=1108
Eg_T1=Eg-A*T1*T1/(B+T1)
Eg_T2=Eg-A*T2*T2/(B+T2)
Vj_T2=(T2/T1)*Vj-(2*Vt)*ln(exp(1.5*ln(T2/T1)))-((T2/T1)*Eg_T1-Eg_T2)
Cj0_T2=Cj0*(1+M*(400e-6*(T2/T1)-(Vj_T2-Vj)/Vj))
Is_T2=Is*exp((XTI/N)*ln(T2/T1))*limexp((-Eg_T1/Vt)*(1-T2/T1))
```

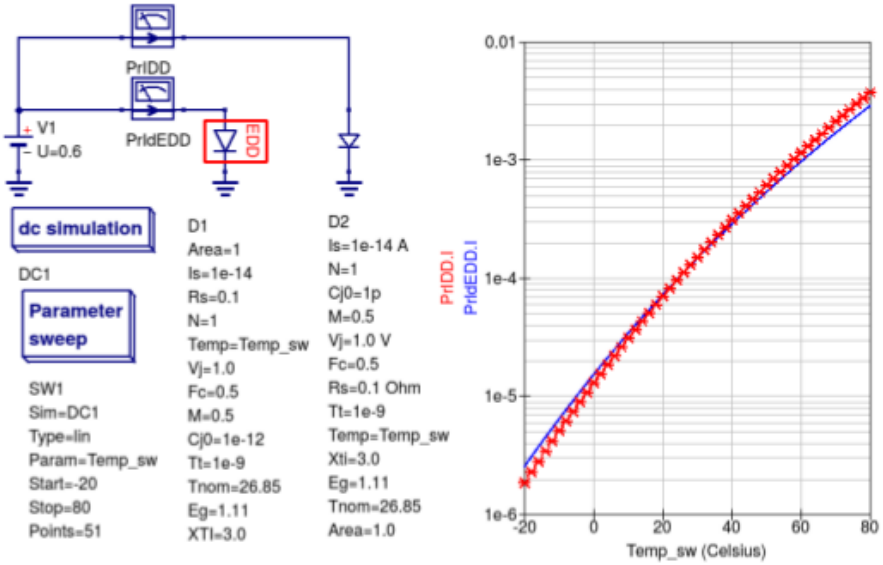
```
D1
I1=(V1 > -Con1) ?  Area*Is_T2*(limexp(V1/(N*Vt))-1) : -Is_T2
Q1= (V1 <= Con2) ? Tt*V2+Area*(Cj0_T2*Vj_T2/(1-M))*(1-exp( (1-M)*ln( 1-V1/Vj_T2))) :  Tt*V2+Area*Cj0_T2*(F1+(1/F2)*(F3*(V1-Fc*Vj_T2)+(M/(2*Vj_T2))*(V1*V1-Fc*Fc*Vj_T2*Vj_T2)))
I2=0
Q2=0
```

Qucs EDD diode model with temperature effects

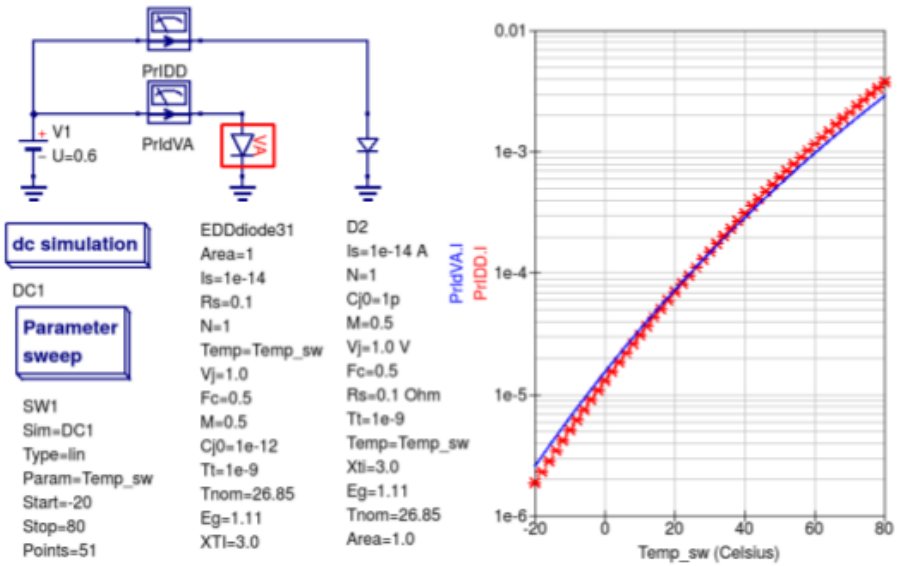
Synthesized Verilog-A code

```
`include "disciplines.vams"
`include "constants.vams"
module EDDdiode3(Pcathode, Panode);
inout Pcathode, Panode;
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 Vj=1.0;  parameter real Fc=0.5;  parameter real M=0.5;
parameter real Cj0=1e-12;    parameter real Tt=1e-9; parameter real Tnom=26.85; parameter real Eg=1.11;
parameter real XTI=3.0;
real RMAX, T1, T2, Con1, Con2, Vt, F1, F2, F3, A, B, Eg_T1, Eg_T2, Vj_T2, Cj0_T2, Is_T2;
analog begin
@(initial_model)
begin
RMAX=1e12;  T1=Tnom+273.15;  T2=Temp+273.15;  Con1=5*N*Vt;  Con2=Fc*Vj; Vt=`P_K*T2/`P_Q;
F1=(Vj/(1-M))*(1-exp((1-M)*ln(1-Fc))); F2=exp((1+M)*ln(1-Fc)); F3=1-Fc*(1+M); A=7.02e-4; B=1108;
Eg_T1=Eg-A*T1*T1/(B+T1); Eg_T2=Eg-A*T2*T2/(B+T2);
Vj_T2=(T2/T1)*Vj-(2*Vt)*ln(exp(1.5*ln(T2/T1)))-((T2/T1)*Eg_T1-Eg_T2);
Cj0_T2=Cj0*(1+M*(400e-6*(T2/T1)-(Vj_T2-Vj)/Vj));
Is_T2=Is*exp((XTI/N)*ln(T2/T1))*limexp((-Eg_T1/Vt)*(1-T2/T1));
end
I(Pcathode,n2) <+ V(Pcathode,n2)/( RMAX );
I(Pcathode,n2) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( RMAX ), "thermal" );
I(n1,Panode) <+ V(n1,Panode)/( Rs );
I(n1,Panode) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( Rs ), "thermal" );
I(n1,n2)  <+  V(n1,n2) * 1e3; I(n3,n4)  <+  I(n3,n4) *1e-3; I(n3,n4)  <+  V(n1,n2) * 1e6 * 1 ;
I(n2,Pcathode) <+ (V(n2,Pcathode)>-Con1)?Area*Is_T2*(limexp(V(n2,Pcathode)/(N*Vt))-1):-Is_T2;
I(n2,Pcathode) <+ ddt( (V(n2,Pcathode)<=Con2)?Tt*V(n4,n3)+Area*(Cj0_T2*Vj_T2/(1-M))*(1-exp((1-M)*
                    ln(1-V(n2,Pcathode)/Vj_T2))):
                    Tt*V(n4,n3)+Area*Cj0_T2*(F1+(1/F2)*(F3*(V(n2,Pcathode)-Fc*Vj_T2)+(M/(2*Vj_T2))*
                    (V(n2,Pcathode)*V(n2,Pcathode)-Fc*Fc*Vj_T2*Vj_T2))) );
I(n3) <+ (-V(n3))/( RMAX );
I(n3) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( RMAX ), "thermal" );
end
endmodule
```

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

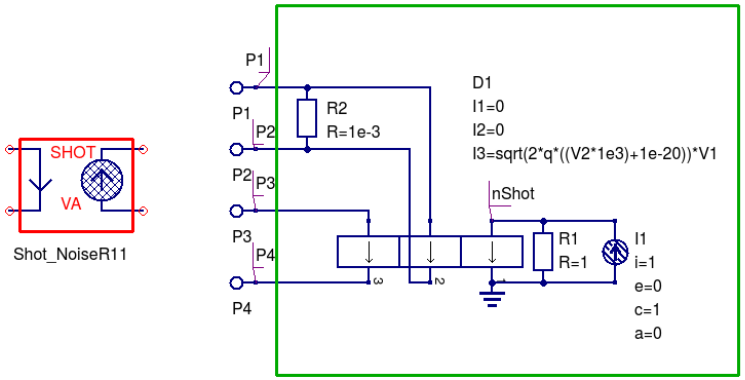


Simulation data for
Qucs EDD model and built-in diode model



Simulation data for
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.

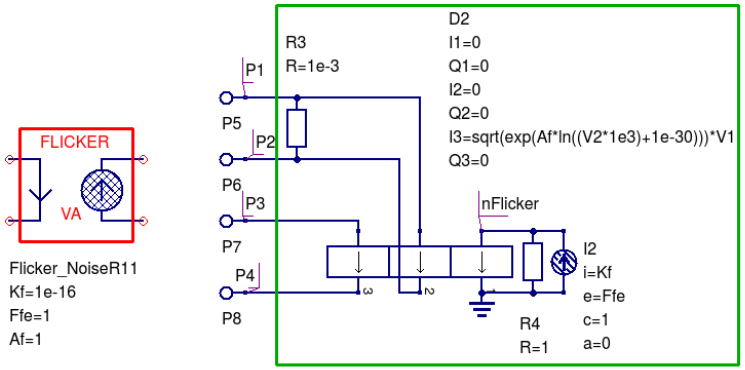


Noise model symbols

Compact modelling TEMPLATE

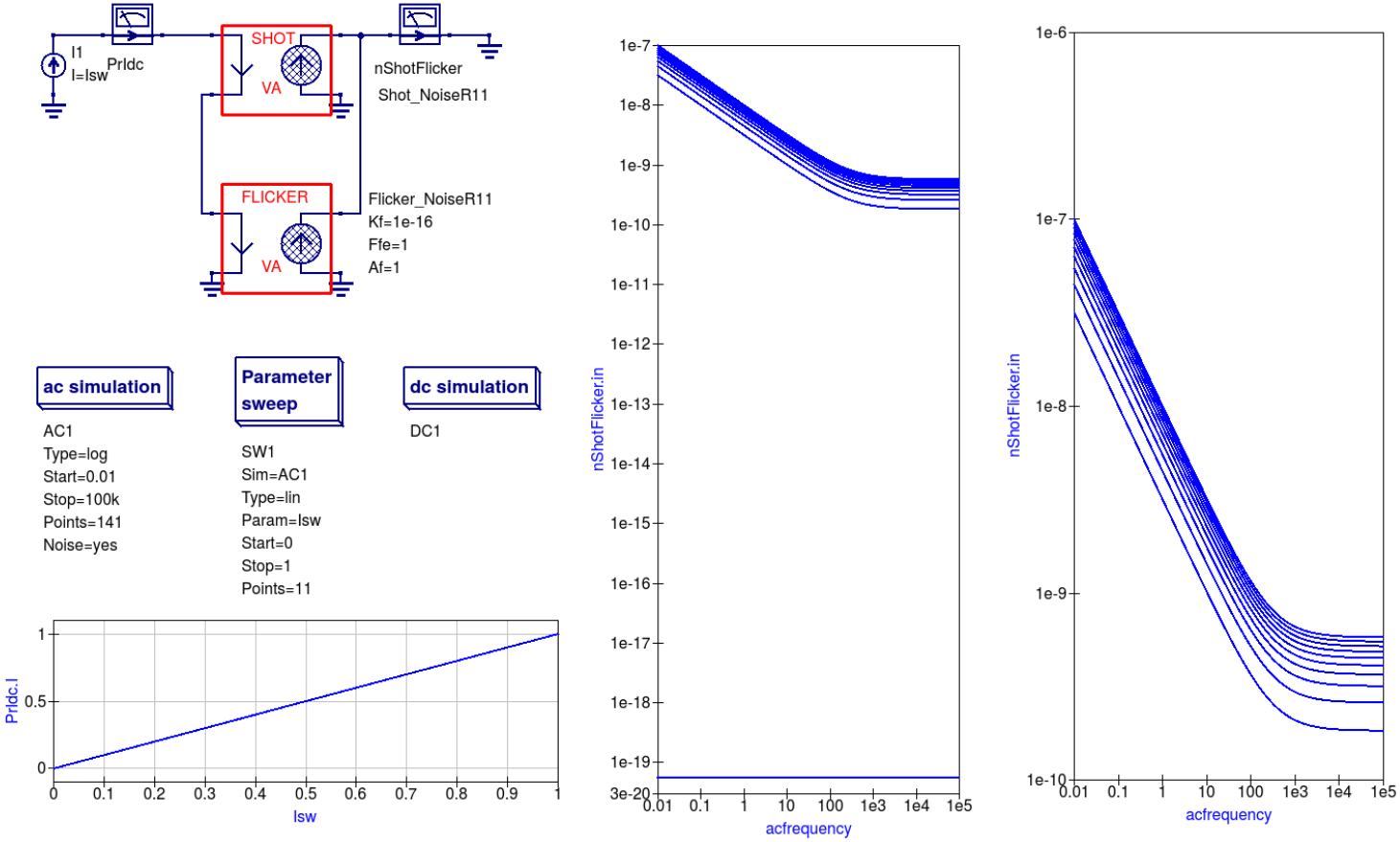
```
`include "disciplines.vams"
`include "constants.vams"
module Shot_NoiseR1(P1, P2, P3, P4);
inout P1, P2, P3, P4;
electrical nShot, P2, P1, P3, P4;
analog begin
@(initial_model)
begin
end
I(nShot) <+ (-V(nShot))/( 1 );
I(nShot) <+ white_noise(1,"shot" );
I(P2,P1) <+ V(P2,P1)/( 1e-3 );
I(P3,P4) <+ sqrt(2*`P_Q*((V(P1,P2)*1e3)+1e-20))*V(nShot);
end
endmodule
```

Synthesized Verilog-A module code

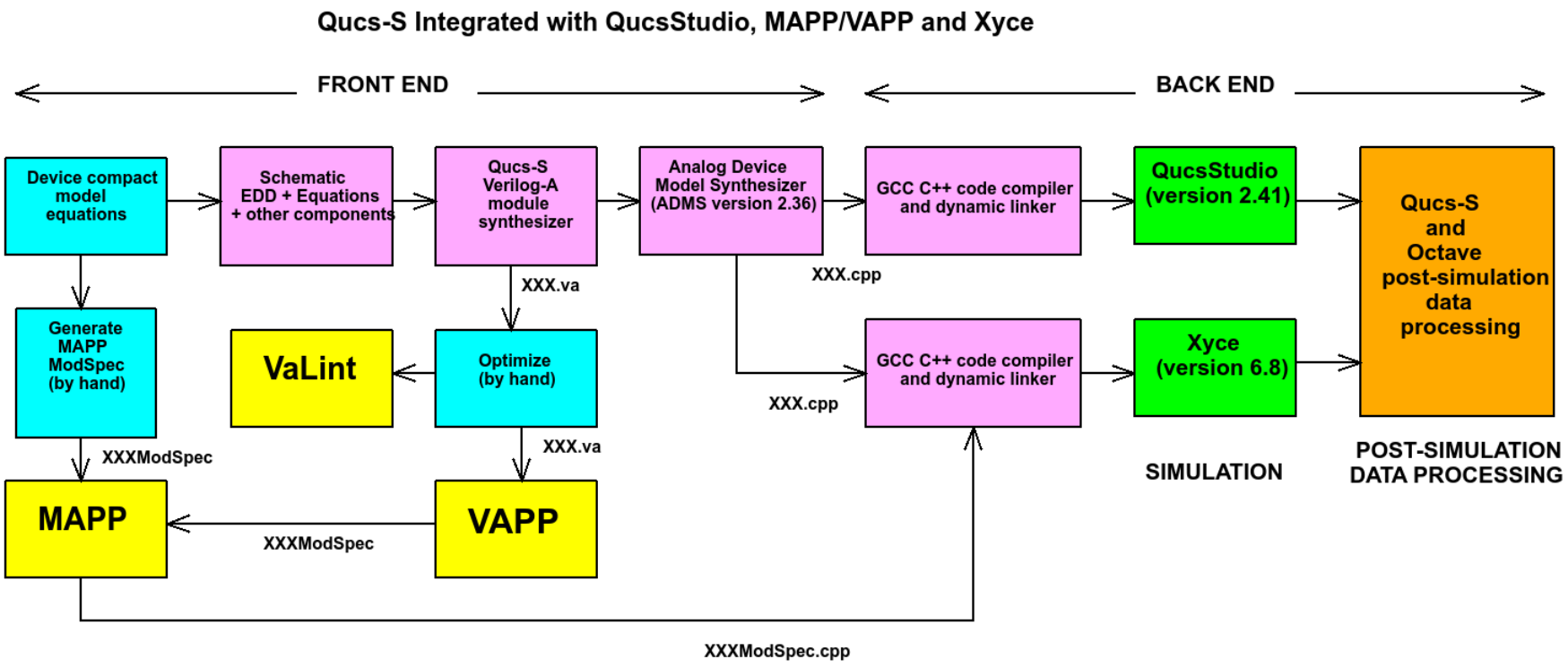



```
`include "disciplines.vams"
`include "constants.vams"
module Flicker_NoiseR1(P1, P2, P3, P4);
inout P1, P2, P3, P4;
electrical P2, P1, nFlicker, P3, P4;
parameter real Kf=1e-12;
parameter real Ffe=1;
parameter real Af=1;
analog begin
@(initial_model)
begin
end
I(P2,P1) <+ V(P2,P1)/( 1e-3 );
I(nFlicker) <+ flicker_noise(Kf, Ffe, "flicker" );
I(nFlicker) <+ (-V(nFlicker))/( 1 );
(P3,P4) <+ sqrt(exp(Af*ln((V(P1,P2)*1e3)+1e-30)))*V(nFlicker);
end
endmodule
```

Verilog-A synthesis of semiconductor device shot and flicker noise: small signal AC domain simulation data.



Building SiC compact device models with Qucs-S, QucsStudio, MAPP/VAPP and Xyce: the model development tool kit

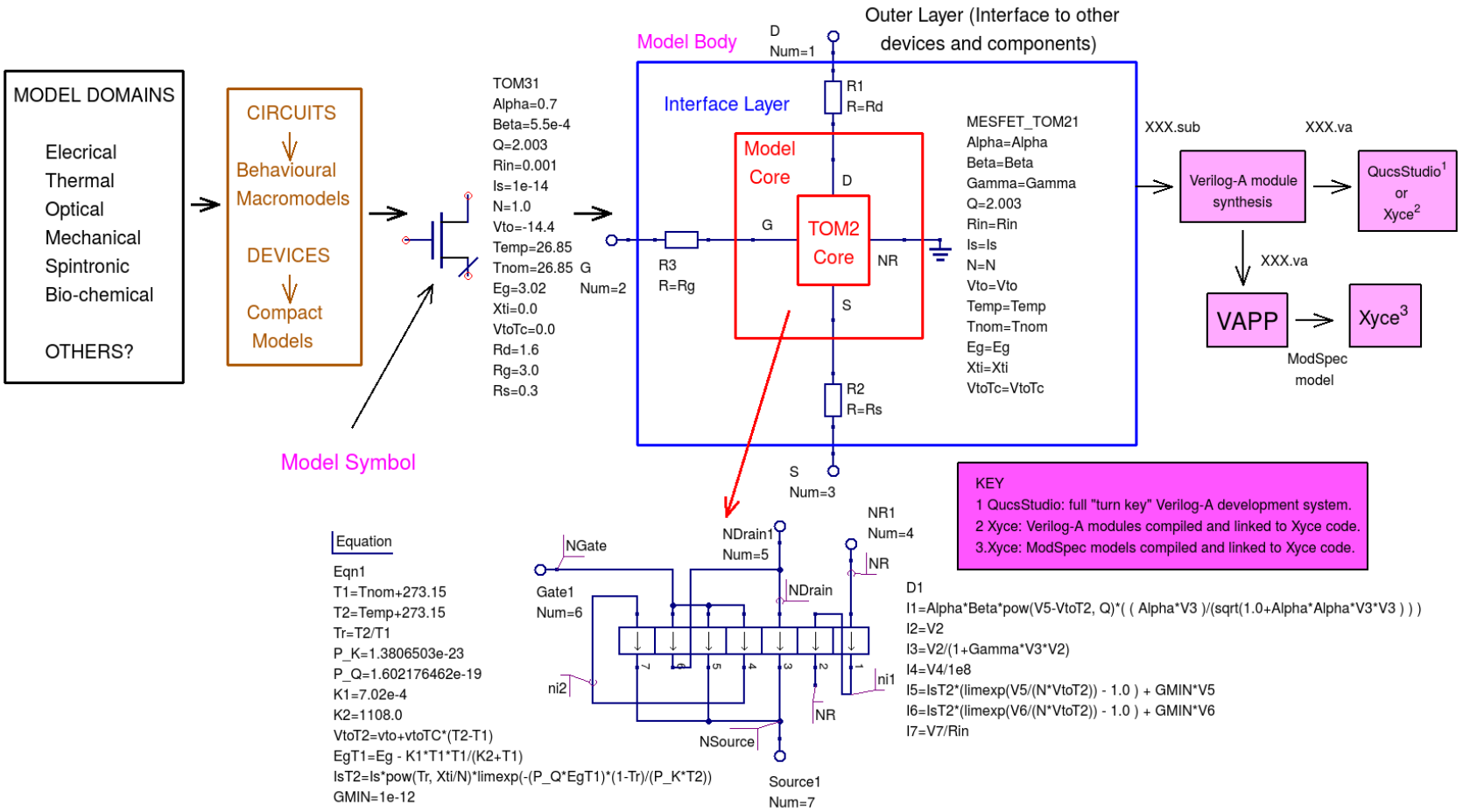


MAPP : The Berkeley Model and Algorithm Prototyping Platform

VAPP : Verilog-A Parser and Processor

VaLint : NEEDS Verilog-A checker

Building SiC compact device models with Qucs-S, QucsStudio, MAPP/VAPP and Xyce: model structure



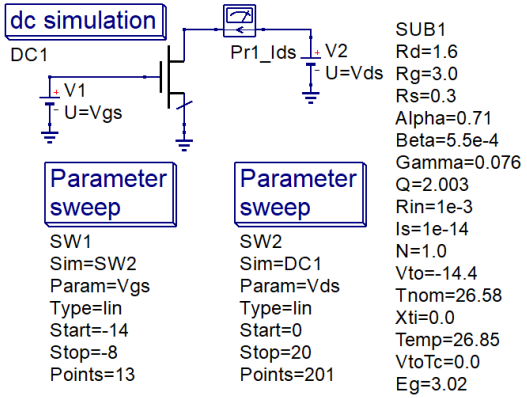
Building SiC compact device models with Qucs-S, QucsStudio, MAPP/VAPP and Xyce: the development of a fundamental 4H-SiC MESFET "Triquint level 2 (TOM2)" model

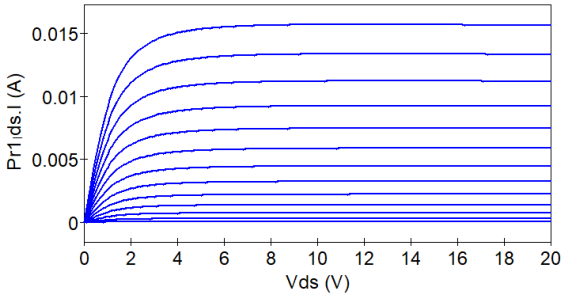
Tools ➤ Compact modeling ➤ Build Verilog-A module from subcircuit

↓

TOM2 core

```
// Basic static I/V characteristic Verilog-A code for the TOM2 model.
// Reverse breakdown characteristics NOT modelled.
`include "disciplines.vams"
`include "constants.vams"
module MESFET_TOM2(NR, NDrain, NGate, NSource);
inout NR, NDrain, NGate, NSource;
electrical NR, ni1, NDrain, NSource, NGate, ni2;
parameter real Alpha=Alpha; parameter real Beta=Beta; parameter real Gamma=Gamma;
parameter real Q=2.003;
parameter real Rin=Rin; parameter real Is=Is; parameter real N=N; parameter real Vto=Vto;
parameter real Temp=Temp; parameter real Tnom=Tnom; parameter real Eg=Eg;
parameter real Xti=Xti; parameter real VtoTc=VtoTc;
real T1, T2, Tr, P_K, P_Q, K1, K2, VtoT2, EgT1, IsT2, GMIN, VthT2;
analog begin
@(initial_model)
begin
T1=Tnom+273.15; T2=Temp+273.15;  Tr=T2/T1; P_K=1.3806503e-23;  P_Q=1.602176462e-19;
K1=7.02e-4;  K2=1108.0; VtoT2=vto+vtoTC*(T2-T1); EgT1=Eg-K1*T1*T1/(K2+T1);
IsT2=Is*pow(Tr,Xti/N)*limexp(-(P_Q*EgT1)*(1-Tr)/(P_K*T2)); GMIN=1e-12;  VthT2=P_K*T2/P_Q;
end
I(NR,ni1) <+ Alpha*Beta*pow(V(NGate,NSource)-Vto,Q)*((Alpha*V(NDrain,NSource))/
  (sqrt(1.0+Alpha*Alpha*V(NDrain,NSource)*V(NDrain,NSource)))); I(ni1,NR) <+ V(ni1,NR);
I(NDrain,NSource) <+ V(ni1,NR)/(1+Gamma*V(NDrain,NSource)*V(ni1,NR));
I(NGate,ni2) <+ V(NGate,ni2)/1e8;
I(NGate,NSource) <+ IsT2*(limexp(V(NGate,NSource)/(N*VthT2))-1.0)+GMIN*V(NGate,NSource);
I(NGate,NDrain) <+ IsT2*(limexp(V(NGate,NDrain)/(N*VthT2))-1.0)+GMIN*V(NGate,NDrain);
I(ni2,NSource) <+ V(ni2,NSource)/Rin;
end
endmodule
```





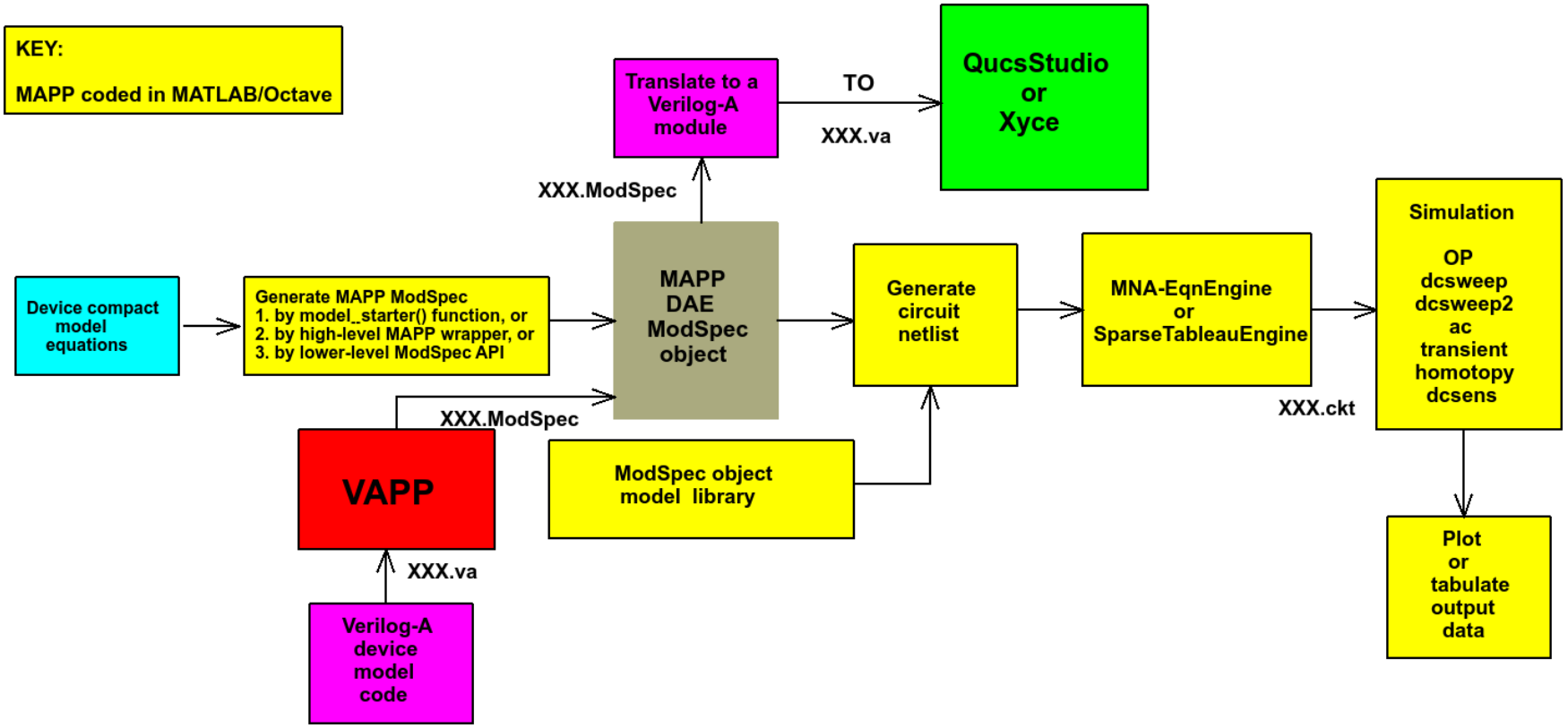
Building SiC compact device models with Qucs-S, QucsStudio, MAPP/VAPP and Xyce: the development of a fundamental 4H-SiC MESFET "Triquint level 2 (TOM2)" model; improvements and limitations

- Improvements: extending, for example, the TOM2 model to improve I/V characteristics, include dynamic model charge properties, add thermal effects and electrical noise is straight forward provided model Verilog-A module code is written in the ADMS Verilog-A subset.
- Limitations: ADMS performance is limited by a number of features; 1. the generated backend differential code is very large leading to poor simulation times, 2. ADMS has node collapsing problems, 3. ADMS is written in XSLT making its code difficult to understand and maintain, 4. ADMS has no support for current branches and 5. very poor documentation.
- Industrial level compact Verilog-A device models are normally written in a much larger subset of Verilog-A, often resulting in model failure when compiled with ADMS.
- Future model development: For industrial grade compact modelling of electrical and multi-physics systems requires a reliable an much improved software tool - hence move Qucs-S compact model development to the Berkeley MAPP and VAPP tools. These have been under continuous development over the last six years as part of the NEEDS NonoHUB project and have been released under the open source GPL licence.

Building SiC compact device models with Qucs-S, QucsStudio, MAPP/VAPP and Xyce: Moving forward - merging MAPP/VAPP and XYCE within the Qucs-S framework

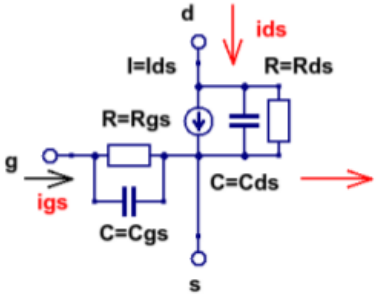
- The release of the advanced Xyce parallel circuit simulator under the GPL licence has made available to the wider compact modelling community a package with significant new simulation facilities.
- By combining the Xyce circuit simulator with the new Berkeley "Modelling and algorithm prototyping platform, under the GPL licence, the available compact modelling tool set has been "future proofed" for the foreseeable future.
- Release of the latest stable version of Qucs-S in October 2017 has triggered work on merging MAPP/VAPP (coupled with Xyce) within the Qucs-S framework.
- The remaining slides in this presentation report on the work done so far to achieve the merger of MAPP/VAPP and Xyce with Qucs-S. An indication of future plans are also introduced and a number of SiC modelling examples are outlined.

Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: compact device model development with MAPP and VAPP; part 1



Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: compact device model development with MAPP and VAPP;part 2

A simplified circuit model for a 4H-SiC MESFET*



Here, Rds = Rgs = 1/GMIN

Differential/Algebraic Equations (DAE)

ids=Ids+d/dt(Cds · Vds) +Vds · GMIN

igs=d/dt(Cgs ·Vgs) + Vgs · GMIN,

where

$$Ids = (\beta \cdot (Vgs\text{-}Vto)^{Q})/(1+\delta \cdot Vds \cdot \backslash be\ Vdsta \cdot (Vgs\text{-}Vto)^{Q}) \cdot \tanh((\alpha \cdot Vds)/(Vds\text{-}Vto)^{m}),$$

and $\alpha = 0.71$, $\beta = 5.5e\text{-}4$, $Q = 2.003$, $Vto=-14.4$, $Cds=2.67e\text{-}12$, $Cgs=0.499e\text{-}12$, $m=1.6$, $GMIN=1e\text{-}9$.

In general MAPP model branches, electrical systems, can be represented by

d/dt(q(v)) + f(v) + b(t) =0,

where, nonlinear functions q(v) and f(v) represent the charge andresistive parts of a branch, respectively, and b(t) is an external voltage or current input which is a function of time.

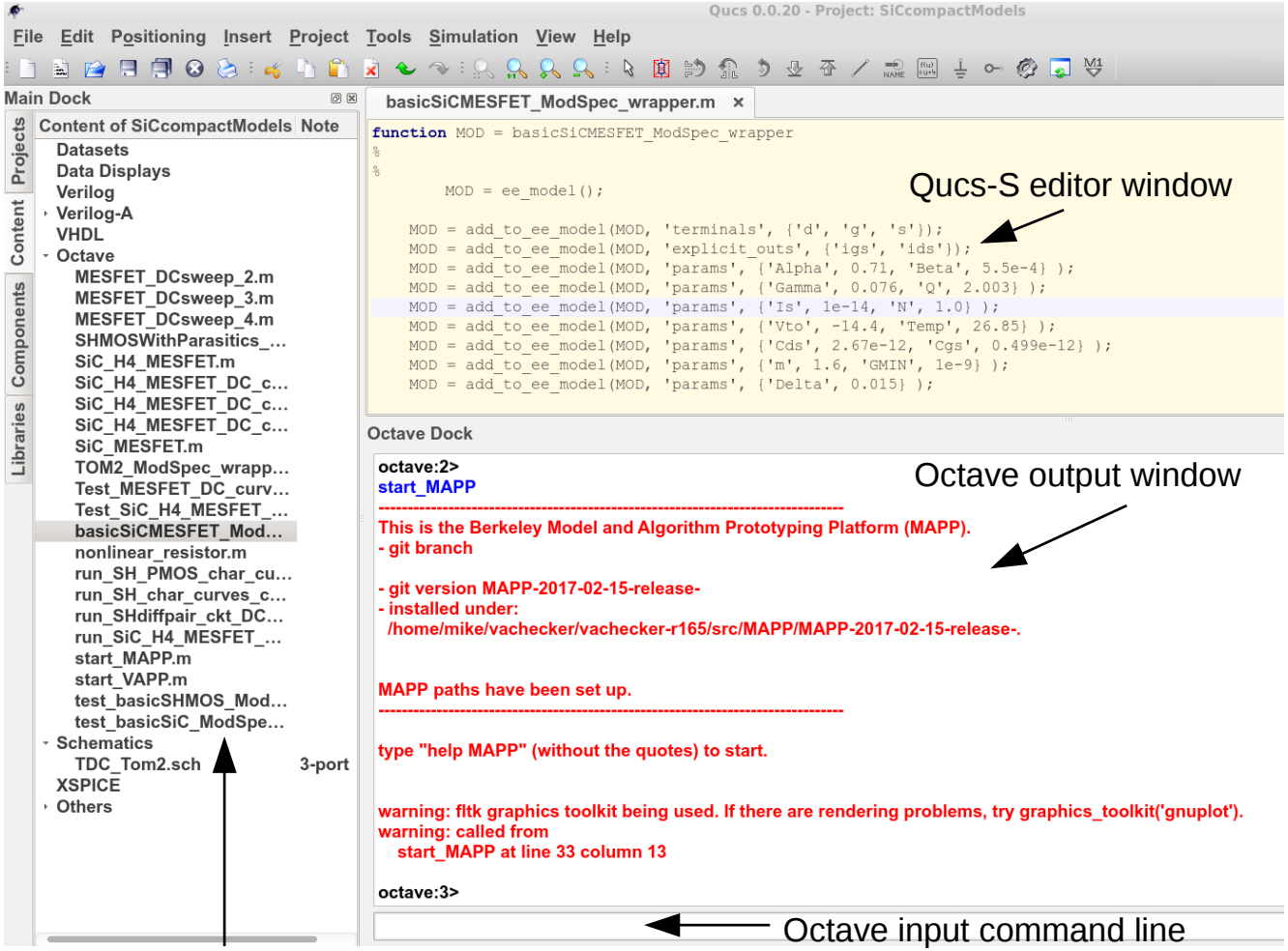
ModSpec files represent objects being modelled in a very general way which allows explicit and implicit formulation with specified inputs and outputs, supports internal unknown quantities and time-varying behaviour.

MAPP ModSpec function

```
function MOD = basicSiCMESFET_ModSpec_wrapper()
%
  MOD = ee_model();

  MOD = add_to_ee_model(MOD, 'terminals', {'d', 'g', 's'});
  MOD = add_to_ee_model(MOD, 'explicit_outs', {'igs', 'ids'});
  MOD = add_to_ee_model(MOD, 'params', {'Alpha', 0.71, 'Beta', 5.5e-4} );
  MOD = add_to_ee_model(MOD, 'params', {'Gamma', 0.076, 'Q', 2.003} );
  MOD = add_to_ee_model(MOD, 'params', {'Is', 1e-14, 'N', 1.0} );
  MOD = add_to_ee_model(MOD, 'params', {'Vto', -14.4, 'Temp', 26.85} );
  MOD = add_to_ee_model(MOD, 'params', {'Cds', 2.67e-12, 'Cgs', 0.499e-12} );
  MOD = add_to_ee_model(MOD, 'params', {'m', 1.6, 'GMIN', 1e-9} );

  MOD = add_to_ee_model(MOD, 'f', @f);
  MOD = add_to_ee_model(MOD, 'q', @q );

  MOD = finish_ee_model(MOD);
end
%
function fout = f(S)
 v2struct(S);
 Aval = (vgs-Vto)^Q;
 Bval = (vgs-Vto)^m;
 Ids = ( Beta*Aval/(1.0+Gamma*vds*Beta*Aval) )*tanh(Alpha*vds/Bval) + vds*GMIN;
 %
 igs = vgs*GMIN;
 fout = [Ids; igs];
%
function qout = q(S)
 v2struct(S);
 qds = Cds*vds;
 qgs = Cgs*vgs;
 %
 qout = [qds; qgs];
```

*XU Yue-hang et al., Advanced SPICE- modelling 0f 4H-SiC MESFET's, Journal of Electronic Science and Technology of China, March 2007, Vol. 5 No. 1, pp. 62-65.

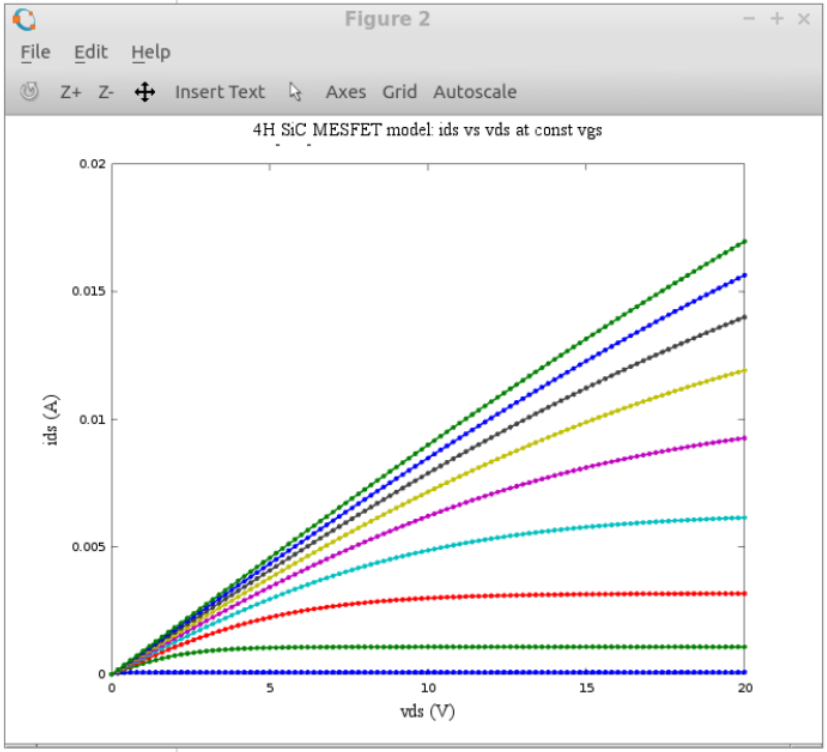
Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: compact device model development with MAPP and VAPP;part 3



MAPP and VAPP Octave functions and scripts

Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: compact device model development with MAPP and VAPP;part 4

```
function out = test_basicSiC_ModSpec_wrapper()

    S.Alpha = 0.71;     S.Beta = 5.5e-4;    S.Delta = 0.015;
    S.Q = 2.003;        S.Gamma = 0.076;    S.Cgs = 0.499e-12;
    S.Cds = 2.67e-12;   S.Vto = -14.4;      S.m   = 1.6;
    S.GMIN = 1e-9;
   MOD = basicSiCMESFET_ModSpec_wrapper;

   vgs_min = -14; vgs_max = -6; vgs_step = 1;
   vds_min =  0;  vds_max = 20.0; vds_step = 0.2;

   vgss = vgs_min : vgs_step : vgs_max;
   vdss = vds_min : vds_step : vds_max;

   nvgss = size(vgss,2); nvdss = size(vdss,2);

   % produce characteristic curves: id vs vds at constant vgs

   figure();
   set(gcf,'color','white');
   set(gca, 'FontName', 'Times New Roman', 'FontSize', 15);
   plots = [];
   legend_strs = {};
   for row_idx = 1 : 1 : nvgss
       S.vgs = vgss(1, row_idx);
       idss = zeros(1, nvdss);
       for col_idx = 1:1:nvdss
           S.vds = vdss(1, col_idx);
           currents = MOD.fe_of_S (S);
           idss(1, col_idx) = currents(2,1);
       end
       h = plot(vdss, idss, '.-', 'LineWidth', 1.5);
       hold all;
       plots = [plots, h];
       xlim ([0, vds_max]);
       ylim ([0, 2]);
       xlabel ('vds (V)', 'FontName', 'Times New Roman', 'FontSize', 15);
       ylabel ('ids (A)', 'FontName', 'Times New Roman', 'FontSize', 15);
       title ('4H_{-}SiC_{-}MESFET model: ids vs vds at const vgs',
              'FontName', 'Times New Roman', 'FontSize', 15);
   end
end
```



Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: compact device model development with MAPP and VAPP;part 5

Notes on MAPP and VAPP Verilog-A limitations:

1. The voltage statement style V(drain) is not allowed - recomend the use of the Verilog-A branch statement; branch (a, b) bab;

2. Verilog-A module terminals may need to be rearranged because the last terminal is assumed to be the circuit reference node.

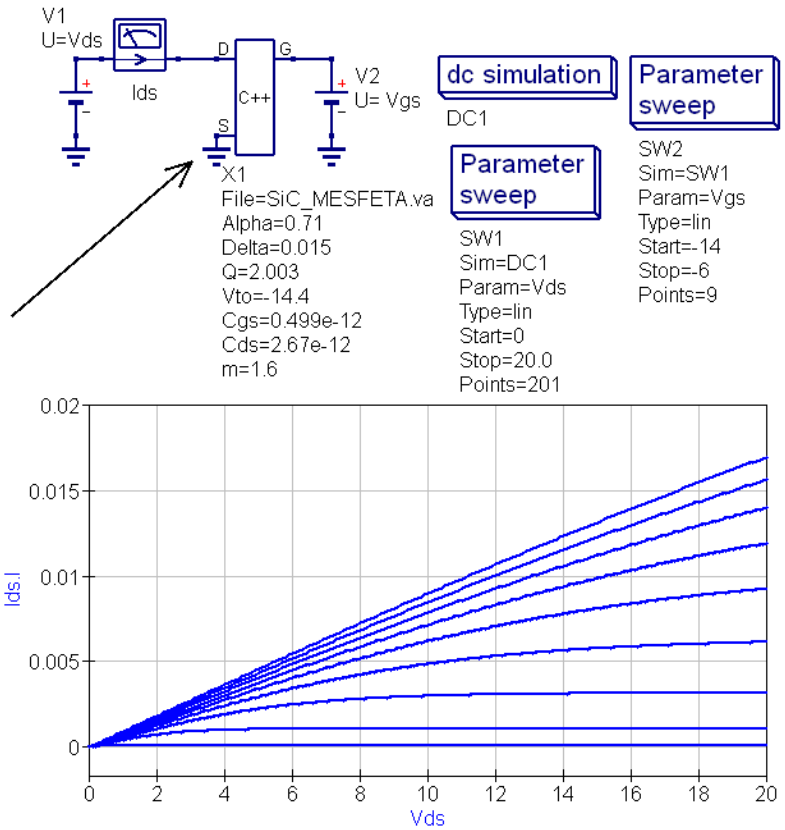
3. Not all Verilog-A functions are implemented yet, for example: the ddx function and multiple output functions.

4. Needs compatibility does not allow some Verilog-A language features: the most important being simulator directives beginning with @.

NOTE MAPP and VAPP allow a high percentage of the Veriog-A statements in the language standardization manual. However, the ADMS software only allows a smaller subset of the language statements.

```
`include "disciplines.vams"
`include "constants.vams"
module SiC_MESFET(D, G, S);
inout D, G, S;
electrical D, G, S;
parameter real Alpha=0.71;
parameter real Beta=5.5e-4;
parameter real Delta=0.015;
parameter real Q=2.003;
parameter real N=1.0;
parameter real Vto=-14.4;
parameter real Temp=26.85;
parameter real Cgs=0.499e-12;
parameter real Cds=2.67e-12;
parameter real m = 1.6;

branch (D, S) bDS1;
branch (D, S) bDS2;
branch (D, S) bDS3;
branch (G, S) bGS1;
branch (G, S) bGS2;

real TempK, Vth, Aval, Bval, Cval;

analog begin
TempK = Temp+273.15;
Vth = `P_K*TempK/`P_Q;
Cval=1.0/(N*Vth);
//
I(bDS1) <+ (Beta*pow(V(bGS1)-Vto,Q)/
  (1.0+Delta*V(bDS1)*Beta*pow(V(bGS1)-Vto,Q) ))
  *tanh(Alpha*V(bDS1)/pow(V(bGS1)-Vto,m));
I(bDS2) <+  ddt(Cds*V(bDS2));
I(bDS3) <+ V(bDS3)*1e-9;
I(bGS1) <+ ddt(Cgs*V(bGS1));
I(bGS2) <+ V(bGS2)*1e-9;
end
endmodule
```

Verilog-A module code derived from MAPP DAE model.

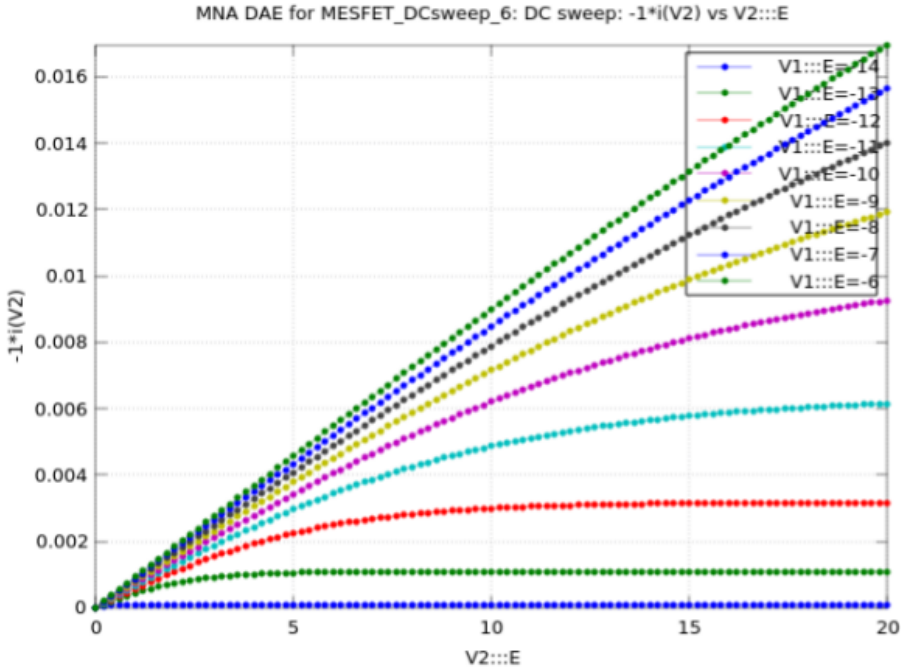


Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: compact device model development with MAPP and VAPP;part 6

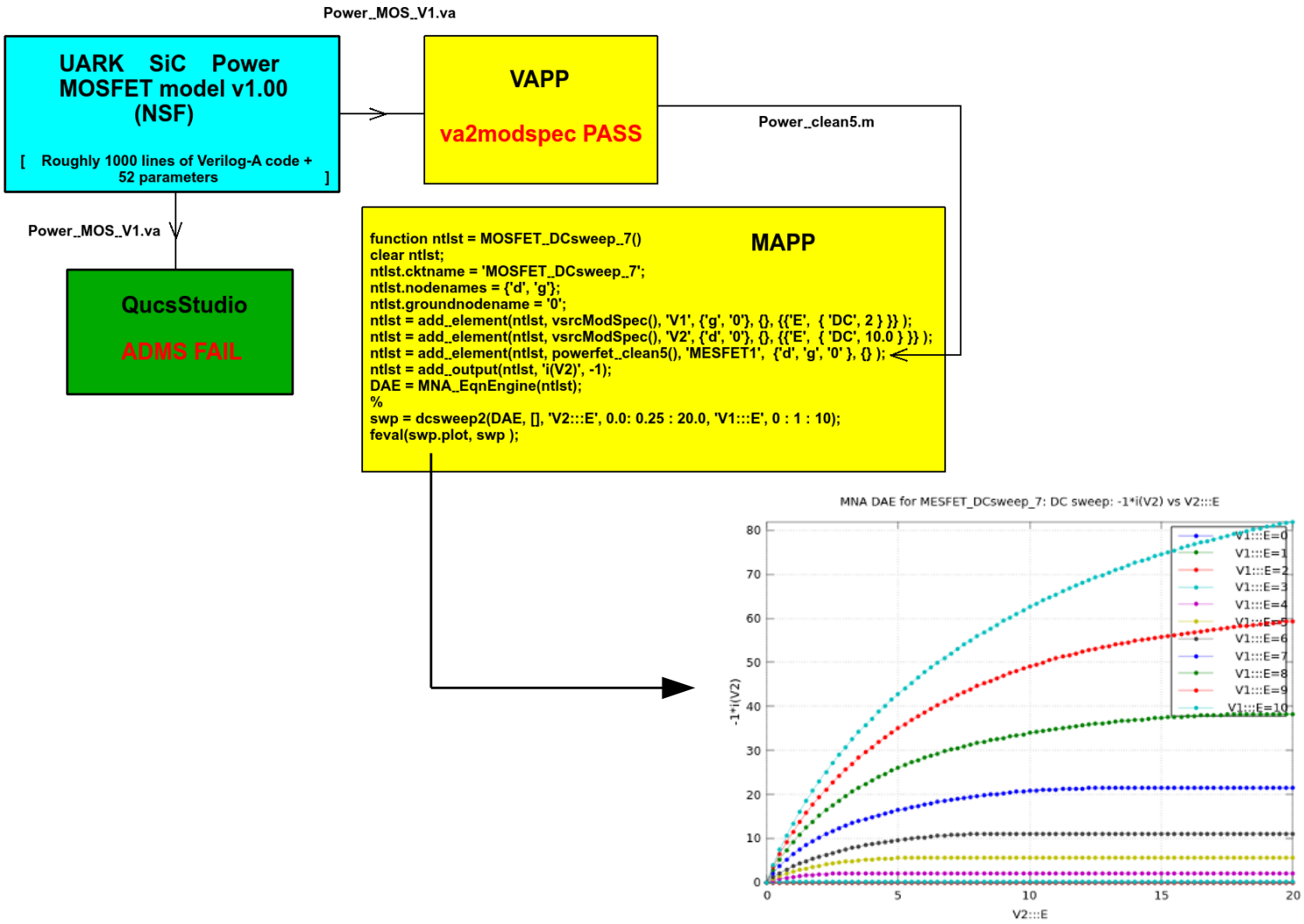
```
function ntlst = MESFET_DCsweep_6()
clear ntlst;
ntlst.cktname = 'MESFET_DCsweep_6';
ntlst.nodenames = {'d', 'g'};
ntlst.groundnodename = '0';
ntlst = add_element(ntlst, vsrcModSpec(), 'V1', {'g', '0'}, {}, {{'E',  { 'DC', -12.0 } }} );
ntlst = add_element(ntlst, vsrcModSpec(), 'V2', {'d', '0'}, {}, {{'E',  { 'DC', 10.0 } }} );
ntlst = add_element(ntlst, SiC_H4_MESFET, 'MESFET1',  {'d', 'g', '0' }, {} );
ntlst = add_output(ntlst, 'i(V2)', -1);
DAE = MNA_EqnEngine(ntlst);
%
swp = dcsweep2(DAE, [], 'V2:::E', 0.0: 0.2 : 20.0, 'V1:::E', -14 : 1 : -6.0);
feval(swp.plot, swp );
```

Test circuit netlist

New MESFET model



Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: compact device model development with MAPP and VAPP; high power SiC MOS model



Using the Berkeley model and algorithm prototyping platform within the Qucs-S framework: Plans for the future

- Increased integration of Qucs-S with MAPP, VAPP and Xyce.
- Improvements in MAPP support: new component models, better tabular output data and Octave/MATLAB visualization.
- Implementation of Xyce compiled MNA DAE models: Integration of Xyce with MAPP, VAPP and Qucs-S via ModSpec-C++API.

Verilog-A compact modelling of SiC devices with Qucs-S, QucsStudio and MAPP/Octave FOSS tools: download links

- Qucs-S: Download links —The latest stable release is Qucs-0.0.20. It is based on stable release Qucs-0.0.19. Documentation can be found at readthedocs.io. Source tarball qucs-s-0.0.20.tar.gz at Github repository. Debian repository (32 and 64 bit), built with openSUSE OBS: Debian 9 "Stretch", Debian 8 "Jessy", Debian 7 "Wheezy", Ubuntu 14.04 and 16.04. RPM Packages (32 and 64 bit) for CentOS and Fedora-24, 25, and 26. Windows installer (Zipped EXE): qucs-s-0.0.20-setup.zip.
- Xyce: Download and documentation from https://xyce.sandia.gov/.
- MAPP/VAPP: Download from https://github.com/jaijeet/MAPP, see also http://draco.eecs.berkeley.edu/mapptiki/tiki-index.php.
- Octave: https://www.gnu.org/software/octave/.

Verilog-A compact modelling of SiC devices with Qucs-S, QucsStudio and MAPP/Octave FOSS tools: references; part1 Qucs-S

- Jahn S., and Brinson M., Interactive compact device modelling using Qucs equation defined devices, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, September/October 2008, 21(5), pp 335-349, DOI:10.1002/jnm.676.
- Brinson M., and Jahn, Qucs: A GPL software package for circuit simulation, compact device modeling and circuit macromodeling from DC to RF and beyond, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, July/August 2009, 22(4), pp 207-319, DOI:10.1002/jnm.702.
- Mike Brinson and Michael Margraf, Verilog-A compact semiconductor device modelling and circuit macromodelling with the QucsStudio-ADMS Turn-Key modelling system, International journal of Microelectronics and Computer Science, Vol. 3, No. 1, pp. 32-40, Jan. 2012. ISSN 2080-8755
- Wladek Grabinski, Mike Brinson, Paolo Nenzi, Francesco Lannutti, Nikolaos Makris, Angelos Antonopoulos and Matthias Bucher, Open-source circuit simulation tools for RF compact semiconductor device modelling, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, Volume 27, Issue 5-6, September- December 2014, Pages: 761779, DOI:10.1002/jnm.1973.
- Mike Brinson and Vadim Kuznetsov, A new approach to compact semiconductor device modelling with Qucs Verilog-A analogue module synthesis, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, Volume 29, Issue 6 November-December 2016, Pages 10701088, DOI: 10.1002/jnm.2166.
- Mike Brinson and Vadim Kuznetsov, Extended behavioural device modelling and circuitsimulation with Qucs-S, International Journal of Electronics, Published online on 29 July 2017. http://dx.doi.org/10.1080/00207217.2017.1357764.

Verilog-A compact modelling of SiC devices with Qucs-S, QucsStudio and MAPP/Octave FOSS tools: references; part2 Berkeley MAPP/VAPP

- MAPP: A Platform for Prototyping Algorithms and Models Quickly and Easily Tianshi Wang, Aadithya V. Karthik, Bichen Wu, and Jaijeet Roychowdhury (invited paper) in Proceedings of IEEE International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO), Aug 2015.
- Multiphysics Modelling and Simulation in Berkeley MAPP Tianshi Wang and Jaijeet Roychowdhury in Proceedings of IEEE International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO), July 2016.
- MAPP: The Berkeley Model and Algorithm Prototyping Platform Tianshi Wang, Aadithya V. Karthik, Bichen Wu, Jian Yao, and Jaijeet Roychowdhury (invited paper) in Proceedings of IEEE Custom Integrated Circuits Conference, Sept 2015.
- Modelling Optical Devices and Systems in MAPP Tianshi Wang and Jaijeet Roychowdhury EECS Department, University of California, Berkeley, Tech. Rep., UCB/EECS-2017-160, Oct 2017.
- Well-Posed Device Models for Electrical Circuit Simulation A. Gokcen Mahmutoglu, Tianshi Wang, Archit Gupta, and Jaijeet Roychowdhury NEEDS: Nano-Engineered Electronic Device Simulation Node Tech. Rep., Mar 2017.
- Further Qucs-S and MAPP/VAPP publications can be found at: 1. MOS-AK Association : http://www.mos-ak.org/, and 2. Nano-Engineered Electronic Device Simulation Node, https://nanohub.org/groups/needs/.