Advances in Qucs-S Schematic Capture for SPICE and Verilog-A Device Modelling and Circuit Simulation

Mike Brinson
Centre for Communications Technology
London Metropolitan University
UK
email: mbrin72043@yahoo.co.uk

Daniel Tomazewski
Łukasiewicz - Institute of Microelectronics and Photonics
Warsaw, Poland
e-mail: daniel.tomaszewski@imif.lukasiewicz.gov.pl

***Abstract*—Schematic capture is an important and popular front-end for circuit simulation. It provides users with a flexible tool that allows circuit diagrams to be drawn and automatically converted into textual circuit netlists. Conventional SPICE simulators are essentially engines that input circuit data and simulation command netlists, undertake simulation, and output data for post-processing. This paper is concerned with an advance in circuit schematic capture functionality which allows both SPICE netlists and Verilog-A module code to be simultaneously generated from a device model or circuit schematic. This development, particularly when combined with SPICE behavioural device modelling, allows automatic generation of Verilog-A device modules rather than going through the manual conversion process from SPICE netlists to Verilog-A code modules. To demonstrate the validity of the reported advances in Qucs-S/Xyce schematic capture a behavioural model and a Verilog-A module for a GaAs MESFET are presented, and their performance described.**

***Index Terms*—Circuit simulation, Qucs-S, Ngspice, Verilog-A, compact device modelling.**

I. Introduction

This paper introduces a schematic capture technique for generating and extracting two or more hardware description language (HDL) netlists or code modules from a single circuit diagram or device model drawing. Conventional schematic capture acts as a circuit simulator front-end for drawing circuit or device model diagrams on a high resolution computer graphics workstation. Output is normally a text file listing component types, values, and connections plus commands to control the circuit simulation process. The de facto industrial netlist format for circuit simulation is based on Berkeley SPICE 2 [1] and SPICE 3 [2]. The current generation of FOSS (Free Open Source Software) circuit simulators, either derived from SPICE, for example Ngspice [3], or developed separately, for example Xyce [4], have adopted the de facto SPICE netlist format with additional simulation commands and device models. Due to the growing complexity of semiconductor compact device models there has in recent years been a move from compact model coding in C or C++ to Verilog-A HDL [5]. Verilog-A is an HDL specifically designed for modeling complex analogue devices. It provides automatic computation of the differential quantities required in non-linear device simulation. It is also ideal for device model interchange. In parallel to semiconductor device modeling with Verilog-A the long established techniques of model construction using behavioural modeling or macromodeling [6] are still popular and significant. The nature of SPICE behavioural modeling allows device models and circuits to be constructed in stages and tested using simulation ”on-the-fly”. However, on successful completion and testing of a model the translation of its netlist to a Verilog-A module code has normally to be done manually. This can be both complex and error prone. The simulation front-end technique reported here introduces a new approach to schematic capture where one, or more, HDL are combined with behavioural SPICE modeling. The schematic capture process is designed to allow easy extraction of both SPICE netlists and, for example, Verilog-A modules using simple Octave functions [7]. The text outlines the details of a new schematic capture process that has been implemented with the Qucs-S/Xyce simulator [8]. This is done with the aid of a series LCR circuit. To further demonstrate the application of the advances in schematic capture, particularly in compact device modeling, the construction of a GaAs MESFET TriQuint Own Model (TOM3) [9] [10] is documented and its simulation and performance described.

II. Qucs-S Schematic Symbols and Parameters

In general SPICE schematic drawing packages adopt a component format similar to that shown in Fig. 1(a), where a graphic symbol has one or more named connecting pins ($P1$ and $P2$), a component name (R_SPICE) and a parameter list with assigned numerical values. During the design of the Qucs-S/Xyce symbol specification a flexible format was adopted where a component is specified by one or more netlist lines. These are text strings that combine to form a SPICE component statement, consisting a one or more SPICE continuation lines after the first line, Fig. 1(b). Originally, this approach was chosen to allow devices with a large number of parameters to be easily specified with a SPICE .MODEL statement. It also allows alternative component formats, like the SPICE semiconductor resistor statement, to be attached to a symbol. Qucs-S also encourages attachment of SPICE documentation information to symbols by adding netlist lines starting with a *.

III. Generating SPICE Netlists and Verilog-A Modules from a Single Schematic

A * in the first column of a SPICE netlist line indicates that it is a comment. This has been standardised across modern FOSS circuit simulators, including Ngspice and Xyce. Fig. 2 shows a simple series LCR subcircuit with L, C and R each specified by two strings. The first string gives the Xyce SPICE netlist entry and the second introduces a comment where the first three columns contain text *va. These indicate that the line is a comment with an embedded Verilog-A HDL statement. Similarly, the Qucs-S INCLUDE SCRIPT attaches to subcircuit LCR the opening code of a Verilog-A module. These entries are passed as *va comments. Xyce treats the *va lines as conventional SPICE comments and as such they have no effect on Xyce SPICE simulation. A Xyce test bench for the LCR subcircuit is drawn in Fig. 3. Fig. 4 lists the Xyce LCR testbench netlist generated by Qucs-S, where Xyce SPICE statements are merged with *va comments. The order of the netlist statements are important. Qucs-S places subcircuits at the top of a Xyce netlist, with the .INCLUDE SCRIPT the first section within the subcircuit internal code, followed by component symbol entries. Finally, after the subcircuit end command .ENDS the LCR XYCE script statements are added to the Xyce netlist. These control simulation. Finally, the Xyce netlist is completed with SPICE .END command. Fig. 5 lists the code for a simple Octave function for extracting a Xyce SPICE netlist (file xxx.spi) and a Verilog-A module (file xxx.va) from a Qucs-S Xyce netlist. Extracted data for the LCR example are given in Fig. 6.

IV. A Combined GaAs MESFET SPICE Behavioural Model and Verilog-A Module

By attaching different HDL text strings to symbols that either specify device properties or model equation handling, schematic capture moves from simply a data entry process to a

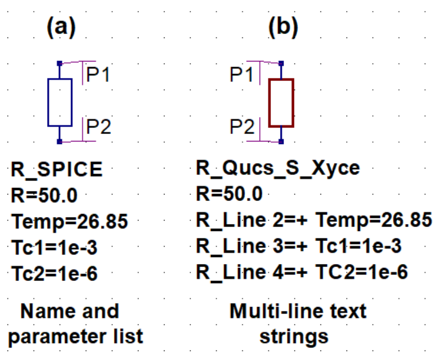


Fig. 1. Example resister drawing symbol and component parameters: (a) conventional SPICE drawing package, (b) Qucs-S/Xyce.

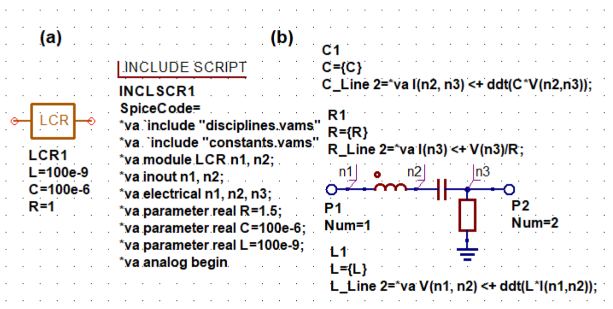

(a)
(b)
LCR
LCR1
L=100e-9
C=100e-6
R=1
INCLUDE SCRIPT
INCLSCR1
SpiceCode=
*va `include "disciplines.vams"
*va `include "constants.vams"
*va module LCR n1, n2;
*va inout n1, n2;
*va electrical n1, n2, n3;
*va parameter real R=1.5;
*va parameter real C=100e-6;
*va parameter real L=100e-9;
*va analog begin
C1
C={C}
C_Line 2=*va I(n2, n3) <+ ddt(C*V(n2,n3));
R1
R={R}
R_Line 2=*va I(n3) <+ V(n3)/R;
n1
n2
n3
P1
Num=1
P2
Num=2
L1
L={L}
L_Line 2=*va V(n1, n2) <+ ddt(L*I(n1,n2));

Fig. 2. An LCR Qucs-S Xyce subcircuit with embedded Verilog-A module code: (a) symbol and Xyce parameter list, (b) subcircuit body with SPICE and Verilog-A combined statements.

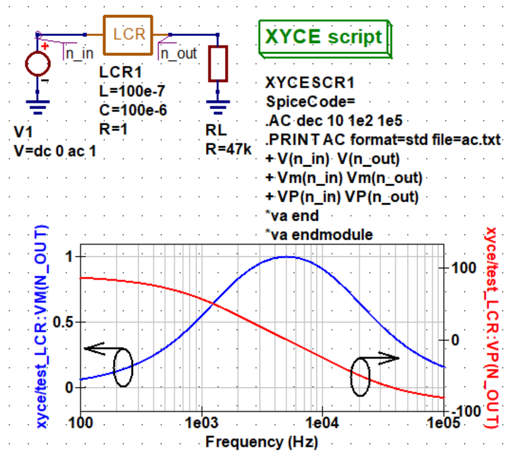
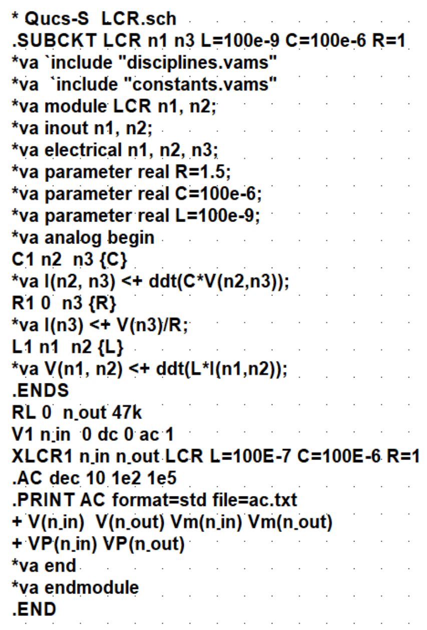


Fig. 3. An a.c. small signal Xyce LCR subcircuit test bench and output voltage magnitude (xyce/test_LCR:VM(N_OUT)) and phase (xyce/test_LCR:VP(N_OUT)) plots.

more flexible tool for generating two or more HDL models. In the previous LCR example the Qucs-S Xyce netlist supports interactive circuit simulation with post-debugging Verilog-A module extraction. Moreover, production level compact model testing then becomes possible using a circuit simulator that allows Verilog-A modules as part of its data input [11] [12]. To demonstrate the application of the proposed schematic capture advances the modeling and simulation of a n-channel GaAs MESFET TOM3 compact device model is introduced in the remaining sections of this paper. Fig. 7 to Fig. 12 illustrate different parts of a complete model schematic. When combined together these give details of a Qucs-S/Xyce SPICE/Verilog-A TOM3 MESFET model, including drain current modeling, gate current modeling, dynamic charge modeling, temperature effects and drain conductance dispersion feedback. Fig. 13 lists extracted TOM3 Verilog-A module code. No attempt is made to model device noise.

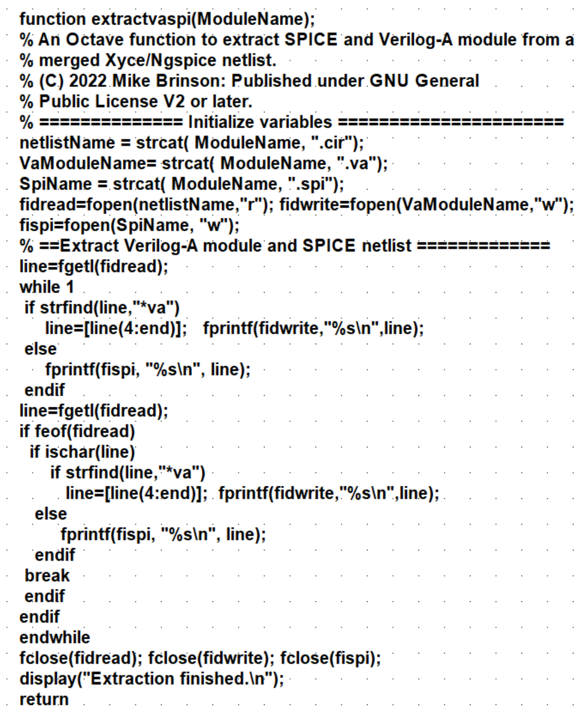
A. *Building Xyce Netlists with Qucs-S*

Qucs-S automatically constructs a Xyce subcircuit netlist file from a schematic drawing. Regardless of the complexity of the model the process follows the steps introduced in the LCR example. Qucs-S/Xyce assumes that a circuit schematic



```
* Qucs-S  LCR.sch
.SUBCKT LCR n1 n3 L=100e-9 C=100e-6 R=1
*va `include "disciplines.vams"
*va  `include "constants.vams"
*va module LCR n1, n2;
*va inout n1, n2;
*va electrical n1, n2, n3;
*va parameter real R=1.5;
*va parameter real C=100e-6;
*va parameter real L=100e-9;
*va analog begin
C1 n2  n3 {C}
*va I(n2, n3) <+ ddt(C*V(n2,n3));
R1 0  n3 {R}
*va I(n3) <+ V(n3)/R;
L1 n1  n2 {L}
*va V(n1, n2) <+ ddt(L*I(n1,n2));
.ENDS
RL 0  n.out 47k
V1 n.in  0 dc 0 ac 1
XLCR1 n.in n.out LCR L=100E-7 C=100E-6 R=1
.AC dec 10 1e2 1e5
.PRINT AC format=std file=ac.txt
+ V(n.in)  V(n.out) Vm(n.in) Vm(n.out)
+ VP(n.in) VP(n.out)
*va end
*va endmodule
.END
```

Fig. 4. A Xyce SPICE/Verilog-A merged netlist: file LCR.cir.



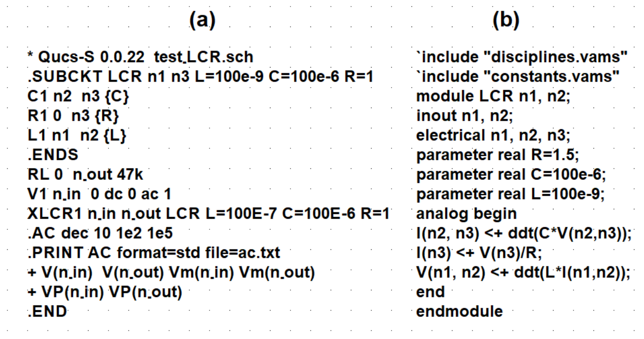
```
function extractvaspi(ModuleName);
% An Octave function to extract SPICE and Verilog-A module from a
% merged Xyce/Ngspice netlist.
% (C) 2022 Mike Brinson: Published under GNU General
% Public License V2 or later.
% ============= Initialize variables ======================
netlistName = strcat( ModuleName, ".cir");
VaModuleName= strcat( ModuleName, ".va");
SpiName = strcat( ModuleName, ".spi");
fidread=fopen(netlistName,"r"); fidwrite=fopen(VaModuleName,"w");
fispi=fopen(SpiName, "w");
% ==Extract Verilog-A module and SPICE netlist ============
line=fgetl(fidread);
while 1
 if strfind(line,"*va")
    line=[line(4:end)];   fprintf(fidwrite,"%s\n",line);
 else
    fprintf(fispi, "%s\n", line);
 endif
line=fgetl(fidread);
if feof(fidread)
  if ischar(line)
    if strfind(line,"*va")
       line=[line(4:end)];  fprintf(fidwrite,"%s\n",line);
   else
       fprintf(fispi, "%s\n", line);
   endif
 break
 endif
endif
endwhile
fclose(fidread); fclose(fidwrite); fclose(fispi);
display("Extraction finished.\n");
return
```

Fig. 5. An example Octave function for extracting SPICE netlist and Verilog-A files from Xyce netlists.

includes only one device model under development/simulation.

B. *Modeling non-linear device equations*

Fig. 9 and Fig. 10 illustrate how a non-linear device equation can be solved during simulation using a Xyce B independent



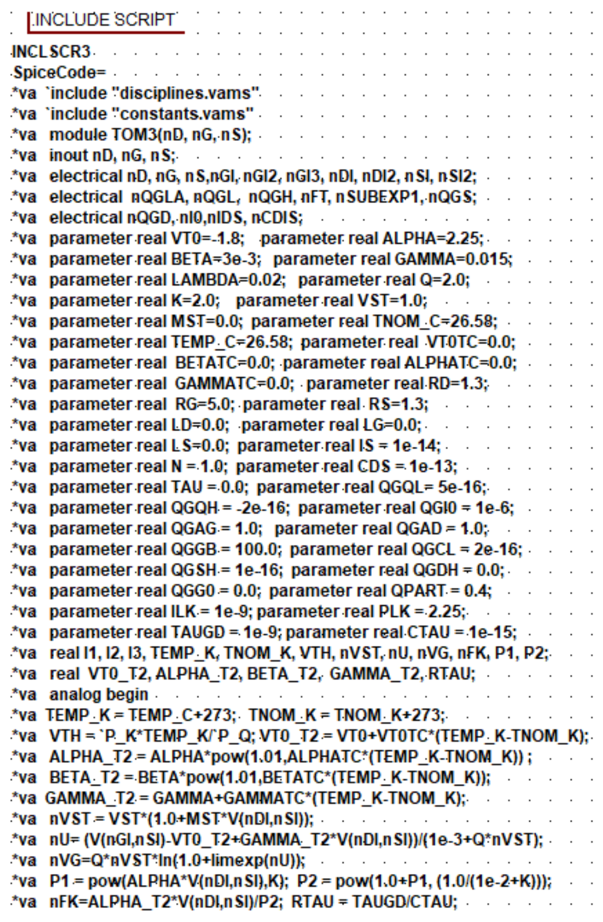
(a)

```
* Qucs-S 0.0.22  test.LCR.sch
.SUBCKT LCR n1 n3 L=100e-9 C=100e-6 R=1
C1 n2  n3 {C}
R1 0  n3 {R}
L1 n1  n2 {L}
.ENDS
RL 0  n.out 47k
V1 n.in  0 dc 0 ac 1
XLCR1 n.in n.out LCR L=100E-7 C=100E-6 R=1
.AC dec 10 1e2 1e5
.PRINT AC format=std file=ac.txt
+ V(n.in)  V(n.out) Vm(n.in) Vm(n.out)
+ VP(n.in) VP(n.out)
.END
```

(b)

```
`include "disciplines.vams"
`include "constants.vams"
module LCR n1, n2;
inout n1, n2;
electrical n1, n2, n3;
parameter real R=1.5;
parameter real C=100e-6;
parameter real L=100e-9;
analog begin
I(n2, n3) <+ ddt(C*V(n2,n3));
I(n3) <+ V(n3)/R;
V(n1, n2) <+ ddt(L*I(n1,n2));
end
endmodule
```

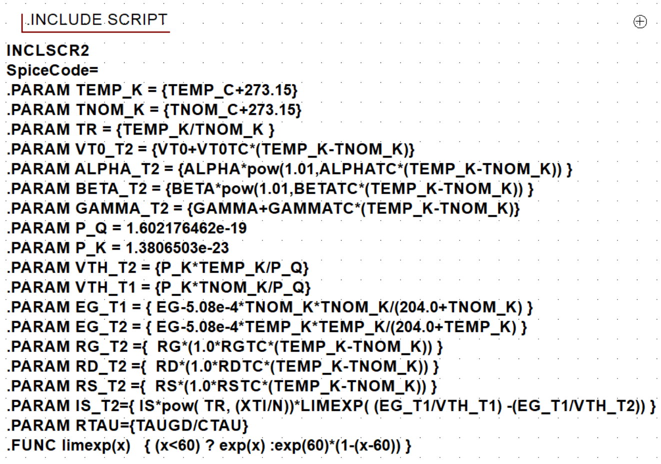
Fig. 6. Extracted SPICE and Verilog-A module files: (a) LCR.spi and (b) LCR.va.



```
.INCLUDE SCRIPT
INCLSCR3
SpiceCode=
*va  `include "disciplines.vams"
*va  `include "constants.vams"
*va   module TOM3(nD, nG, nS);
*va   inout nD, nG, nS;
*va   electrical nD, nG, nS,nGI, nGI2, nGI3, nDI, nDI2, nSI, nSI2;
*va   electrical  nQGLA, nQGL,  nQGH, nFT, nSUBEXP1, nQGS;
*va   electrical nQGD, nI0,nIDS, nCDIS;
*va   parameter real VT0=-1.8;   parameter real ALPHA=2.25;
*va   parameter real BETA=3e-3;  parameter real GAMMA=0.015;
*va   parameter real LAMBDA=0.02;  parameter real Q=2.0;
*va   parameter real K=2.0;   parameter real VST=1.0;
*va   parameter real MST=0.0;  parameter real TNOM_C=26.58;
*va   parameter real TEMP_C=26.58;  parameter real  VT0TC=0.0;
*va   parameter real  BETATC=0.0;  parameter real ALPHATC=0.0;
*va   parameter real  GAMMATC=0.0;   parameter real RD=1.3;
*va   parameter real  RG=5.0;  parameter real  RS=1.3;
*va   parameter real LD=0.0;   parameter real LG=0.0;
*va   parameter real LS=0.0;  parameter real IS = 1e-14;
*va   parameter real N = 1.0;  parameter real CDS = 1e-13;
*va   parameter real TAU = 0.0;  parameter real QGQL= 5e-16;
*va   parameter real QGQH = -2e-16;  parameter real QGI0 = 1e-6;
*va   parameter real QGAG = 1.0;   parameter real QGAD = 1.0;
*va   parameter real QGGB = 100.0;  parameter real QGCL = 2e-16;
*va   parameter real QGSH = 1e-16;  parameter real QGDH = 0.0;
*va   parameter real QGG0 = 0.0;  parameter real QPART = 0.4;
*va   parameter real ILK = 1e-9; parameter real PLK = 2.25;
*va   parameter real TAUGD = 1e-9; parameter real CTAU = 1e-15;
*va   real I1, I2, I3, TEMP_K, TNOM_K, VTH, nVST, nU, nVG, nFK, P1, P2;
*va   real  VT0_T2, ALPHA_T2, BETA_T2,  GAMMA_T2, RTAU;
*va   analog begin
*va  TEMP_K = TEMP_C+273;  TNOM_K = TNOM_K+273;
*va   VTH = `P_K*TEMP_K/`P_Q; VT0_T2 = VT0+VT0TC*(TEMP_K-TNOM_K);
*va   ALPHA_T2 = ALPHA*pow(1.01,ALPHATC*(TEMP_K-TNOM_K)) ;
*va   BETA_T2 = BETA*pow(1.01,BETATC*(TEMP_K-TNOM_K));
*va  GAMMA_T2 = GAMMA+GAMMATC*(TEMP_K-TNOM_K);
*va   nVST = VST*(1.0+MST*V(nDI,nSI));
*va   nU= (V(nGI,nSI)-VT0_T2+GAMMA_T2*V(nDI,nSI))/(1e-3+Q*nVST);
*va   nVG=Q*nVST*ln(1.0+limexp(nU));
*va   P1 = pow(ALPHA*V(nDI,nSI),K);  P2 = pow(1.0+P1, (1.0/(1e-2+K)));
*va   nFK=ALPHA_T2*V(nDI,nSI)/P2;  RTAU = TAUGD/CTAU;
```

Fig. 7. TOM3 first .INCLUDE SCRIPT (INCSCR3) defining Verilog-A module start code, parameter definitions, tha analog begin statement and non-solution dependent compact device model equations.

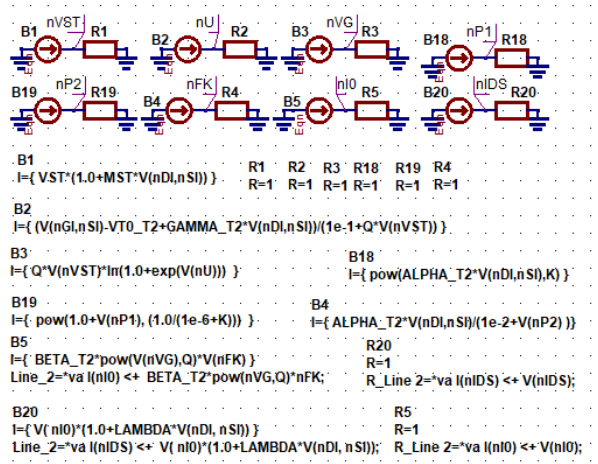
current source connected to a 1Ω resistor. The values of the solved equations are given by source/resistor node voltages. These nodes are labelled nxxx in Fig. 9 and Fig. 10 where xxx is an equation name. Each additional SPICE equation adds one extra internal subcircuit node to a compact device model. This is not necessarily the case with Verilog-A. In Fig. 9 current sources B1, B2, B3, B4, B18 and B19 are defined only for a Xyce netlist. In each case the Verilog-A *va string



```
.INCLUDE SCRIPT
INCLSCR2
SpiceCode=
.PARAM TEMP_K = {TEMP_C+273.15}
.PARAM TNOM_K = {TNOM_C+273.15}
.PARAM TR = {TEMP_K/TNOM_K }
.PARAM VT0_T2 = {VT0+VT0TC*(TEMP_K-TNOM_K)}
.PARAM ALPHA_T2 = {ALPHA*pow(1.01,ALPHATC*(TEMP_K-TNOM_K)) }
.PARAM BETA_T2 = {BETA*pow(1.01,BETATC*(TEMP_K-TNOM_K)) }
.PARAM GAMMA_T2 = {GAMMA+GAMMATC*(TEMP_K-TNOM_K)}
.PARAM P_Q = 1.602176462e-19
.PARAM P_K = 1.3806503e-23
.PARAM VTH_T2 = {P_K*TEMP_K/P_Q}
.PARAM VTH_T1 = {P_K*TNOM_K/P_Q}
.PARAM EG_T1 = { EG-5.08e-4*TNOM_K*TNOM_K/(204.0+TNOM_K) }
.PARAM EG_T2 = { EG-5.08e-4*TEMP_K*TEMP_K/(204.0+TEMP_K) }
.PARAM RG_T2 ={  RG*(1.0*RGTC*(TEMP_K-TNOM_K)) }
.PARAM RD_T2 ={  RD*(1.0*RDTC*(TEMP_K-TNOM_K)) }
.PARAM RS_T2 ={  RS*(1.0*RSTC*(TEMP_K-TNOM_K)) }
.PARAM IS_T2={ IS*pow( TR, (XTI/N))*LIMEXP( (EG_T1/VTH_T1) -(EG_T1/VTH_T2)) }
.PARAM RTAU={TAUGD/CTAU}
.FUNC limexp(x)   { (x<60) ? exp(x) :exp(60)*(1-(x-60)) }
```

Fig. 8. TOM3 second .INCLUDE SCRIPT (INSCR2) defining Xyce subcircuit .PARAM and .FUNC statements.

is not defined. Moreover, the missing equation strings are represented as Verilog-A assignments in the .INCLUDE SCRIPT INCSRC3, rather than current contribution statements. This has the effect of reducing the number of Verilog-A module internal nodes by six. An important advantage of the schematic capture advances reported here is their overall flexibility which encourages development of compact models with reduced simulation run times or smaller model code size.

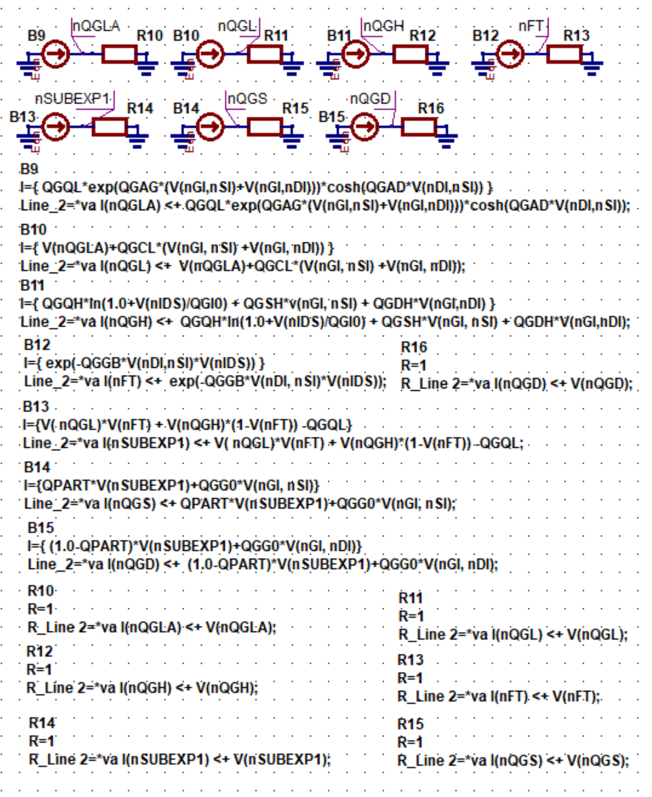


```
B1
I={ VST*(1.0+MST*V(nDI,nSI)) }
R1    R2    R3    R18   R19   R4
R=1   R=1   R=1   R=1   R=1   R=1

B2
I={ (V(nGI,nSI)-VT0_T2+GAMMA_T2*V(nDI,nSI))/(1e-1+Q*V(nVST)) }

B3
I={ Q*V(nVST)*ln(1.0+exp(V(nU)))  }
B18
I={ pow(ALPHA_T2*V(nDI,nSI),K) }

B19
I={ pow(1.0+V(nP1), (1.0/(1e-6+K))) }
B4
I={ ALPHA_T2*V(nDI,nSI)/(1e-2+V(nP2) )}

B5
I={ BETA_T2*pow(V(nVG),Q)*V(nFK) }
Line_2=*va I(nI0) <+  BETA_T2*pow(nVG,Q)*nFK;
R20
R=1
R_Line 2=*va I(nIDS) <+ V(nIDS);

B20
I={ V( nI0)*(1.0+LAMBDA*V(nDI, nSI)) }
Line_2=*va I(nIDS) <+  V( nI0)*(1.0+LAMBDA*V(nDI, nSI));
R5
R=1
R_Line 2=*va I(nI0) <+ V(nI0);
```

Fig. 9. Modeling n-channel TOM3 GaAs MESFET static drain current equations.

C. An n-channel TOM3 MESFET equivalent circuit

A non-linear equivalent circuit for the TOM3 n-channel MESFET is shown in Fig. 11, where drain current is determined by B6, gate current by B7, B16, B8 and B17, gate to drain capacitance by C2, gate to source capacitance by C3, gate drain to source capacitance by C1 and drain conductance dispersion feedback by R21, C4, R17 and B21. The remaining components model signal path series parasitic elements. Parameter QPART listed in the caption to Fig. 11 is 0.4, setting the charge partition scheme for the TOM3 MESFET model to 40/60. The Xyce and Verilog-A C2 and C3 charge equations illustrate the use of parameter QPART. Extracted Verilog-A module code for the GaAs MESFET example is listed in Fig. 13.



```
B9
I={ QGQL*exp(QGAG*(V(nGI,nSI)+V(nGI,nDI)))*cosh(QGAD*V(nDI,nSI)) }
Line_2=*va I(nQGLA) <+ QGQL*exp(QGAG*(V(nGI,nSI)+V(nGI,nDI)))*cosh(QGAD*V(nDI,nSI));
B10
I={ V(nQGLA)+QGCL*(V(nGI, nSI) +V(nGI, nDI)) }
Line_2=*va I(nQGL) <+  V(nQGLA)+QGCL*(V(nGI, nSI) +V(nGI, nDI));
B11
I={ QGQH*ln(1.0+V(nIDS)/QGI0) + QGSH*v(nGI, nSI) + QGDH*V(nGI,nDI) }
Line_2=*va I(nQGH) <+  QGQH*ln(1.0+V(nIDS)/QGI0) + QGSH*V(nGI, nSI) + QGDH*V(nGI,nDI);
B12
I={ exp(-QGGB*V(nDI,nSI)*V(nIDS)) }
Line_2=*va I(nFT) <+  exp(-QGGB*V(nDI, nSI)*V(nIDS));
R16
R=1
R_Line 2=*va I(nQGD) <+ V(nQGD);
B13
I={V( nQGL)*V(nFT) + V(nQGH)*(1-V(nFT)) -QGQL}
Line_2=*va I(nSUBEXP1) <+ V( nQGL)*V(nFT) + V(nQGH)*(1-V(nFT)) -QGQL;
B14
I={QPART*V(nSUBEXP1)+QGG0*V(nGI, nSI)}
Line_2=*va I(nQGS) <+ QPART*V(nSUBEXP1)+QGG0*V(nGI, nSI);
B15
I={ (1.0-QPART)*V(nSUBEXP1)+QGG0*V(nGI, nDI)}
Line_2=*va I(nQGD) <+  (1.0-QPART)*V(nSUBEXP1)+QGG0*V(nGI, nDI);
R10
R=1
R_Line 2=*va I(nQGLA) <+ V(nQGLA);
R11
R=1
R_Line 2=*va I(nQGL) <+ V(nQGL);
R12
R=1
R_Line 2=*va I(nQGH) <+ V(nQGH);
R13
R=1
R_Line 2=*va I(nFT) <+ V(nFT);
R14
R=1
R_Line 2=*va I(nSUBEXP1) <+ V(nSUBEXP1);
R15
R=1
R_Line 2=*va I(nQGS) <+ V(nQGS);
```

Fig. 10. Modeling n-channel TOM3 GaAs MESFET dynamic charge contributions.

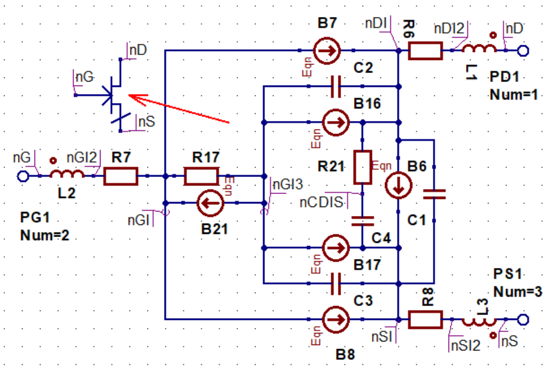
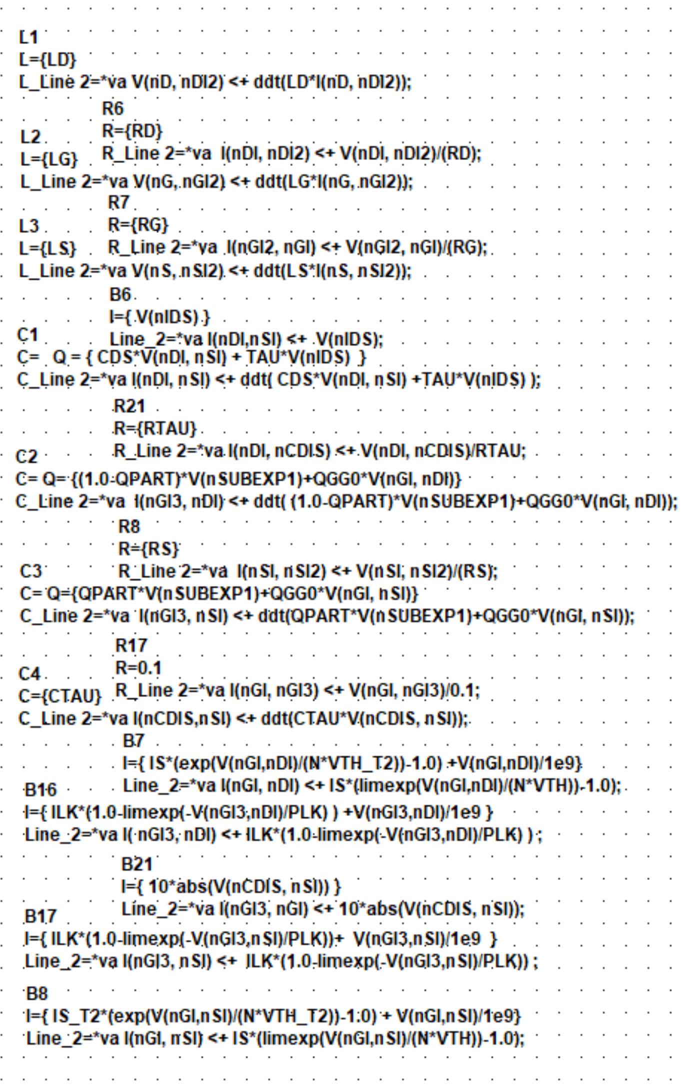


Fig. 11. The n-channel TOM3 GaAs MESFET equivalent circuit with parameters VTO=-1.8, BETA=3e-3, ALPHA=2.26, LAMBDA=0.02, RG=5.0, RD=1.3, RS=1.3, VT0TC=0.0, BETATC=0.0, Q=2.0, K=2.0, MST=0.0, GAMMA=0.015, TEMP_C=26.85, TNOM_C=26.85, GAMMATC=0.0, LD=0.0, LG=0.0, VST=1.0, N=1.0, IS=1e-14, CDS=1e-13, TAU=0.0, QGQL=1e-16, QGQH=-2e-16, QGI0=1e-16, QGAG=1.0, QGAD=1.0, QGGB=100.0, QGCL=2e-16, QGSH=1e-16, QGDH=0.0, QGDH=0.0, QGG0=0.0, QPART=0.0, EG=1.519, RGTC=0.0, RDTC=0.0044, RSTC=0.0016, XTI=2.0, ILK=1e-9, PLK=2.25, LS=0.0, TAUGD=1e-9, CTAU=1e-15.

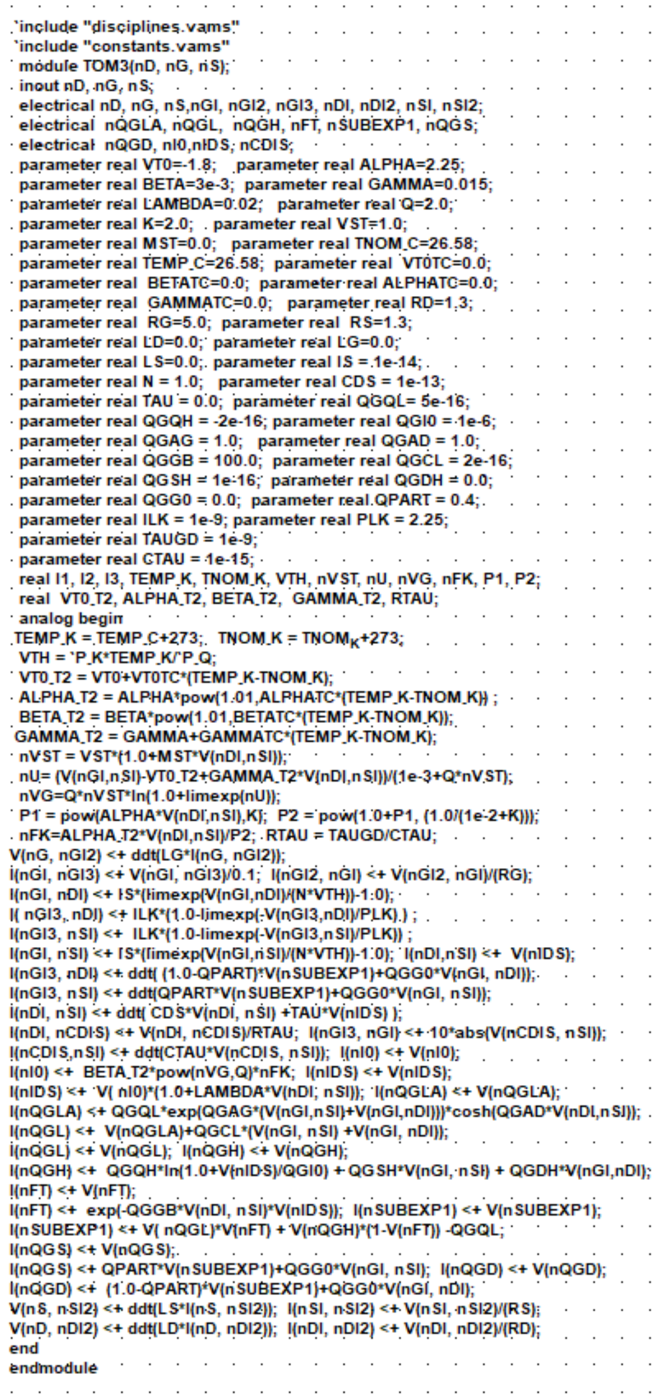


```
L1
L={LD}
L_Line 2=*va V(nD, nDI2) <+ ddt(LD*I(nD, nDI2));
            R6
L2          R={RD}
L={LG}      R_Line 2=*va  I(nDI, nDI2) <+ V(nDI, nDI2)/(RD);
L_Line 2=*va V(nG, nGI2) <+ ddt(LG*I(nG, nGI2));
            R7
L3          R={RG}
L={LS}      R_Line 2=*va  I(nGI2, nGI) <+ V(nGI2, nGI)/(RG);
L_Line 2=*va V(nS, nSI2) <+ ddt(LS*I(nS, nSI2));
            B6
            I={ V(nIDS) }
C1          Line_2=*va I(nDI,nSI) <+  V(nIDS);
C=  Q = { CDS*V(nDI, nSI) + TAU*V(nIDS)  }
C_Line 2=*va I(nDI, nSI) <+ ddt( CDS*V(nDI, nSI) +TAU*V(nIDS) );
            R21
            R={RTAU}
C2          R_Line 2=*va I(nDI, nCDIS) <+ V(nDI, nCDIS)/RTAU;
C= Q= {(1.0-QPART)*V(nSUBEXP1)+QGG0*V(nGI, nDI)}
C_Line 2=*va  I(nGI3, nDI) <+ ddt( (1.0-QPART)*V(nSUBEXP1)+QGG0*V(nGI, nDI));
            R8
            R={RS}
C3          R_Line 2=*va  I(nSI, nSI2) <+ V(nSI, nSI2)/(RS);
C= Q={QPART*V(nSUBEXP1)+QGG0*V(nGI, nSI)}
C_Line 2=*va  I(nGI3, nSI) <+ ddt(QPART*V(nSUBEXP1)+QGG0*V(nGI, nSI));
            R17
C4          R=0.1
C={CTAU}    R_Line 2=*va I(nGI, nGI3) <+ V(nGI, nGI3)/0.1;
C_Line 2=*va I(nCDIS,nSI) <+ ddt(CTAU*V(nCDIS, nSI));
            B7
            I={ IS*(exp(V(nGI,nDI)/(N*VTH_T2))-1.0) +V(nGI,nDI)/1e9}
B16         Line_2=*va I(nGI, nDI) <+ IS*(limexp(V(nGI,nDI)/(N*VTH))-1.0);
I={ ILK*(1.0-limexp(-V(nGI3,nDI)/PLK) ) +V(nGI3,nDI)/1e9 }
Line_2=*va I( nGI3, nDI) <+ ILK*(1.0-limexp(-V(nGI3,nDI)/PLK) ) ;
            B21
            I={ 10*abs(V(nCDIS, nSI)) }
B17         Line_2=*va I(nGI3, nGI) <+ 10*abs(V(nCDIS, nSI));
I={ ILK*(1.0-limexp(-V(nGI3,nSI)/PLK))+  V(nGI3,nSI)/1e9  }
Line_2=*va I(nGI3, nSI) <+  ILK*(1.0-limexp(-V(nGI3,nSI)/PLK)) ;

B8
I={ IS_T2*(exp(V(nGI,nSI)/(N*VTH_T2))-1.0) + V(nGI,nSI)/1e9}
Line_2=*va I(nGI, nSI) <+ IS*(limexp(V(nGI,nSI)/(N*VTH))-1.0);
```

Fig. 12. N-channel TOM3 GaAs MESFET subcircuit component equations.

V. Example Simulation Test Benches and Output Data

The n-channel MESFET simulations introduced in this section have been included to demonstrate the operation of the TOM3 model under d.c.and a.c. conditions and device temperature variations. In Fig. 14 .DC and .STEP commands control drain current as a function of the gate and drain voltages. The a.c. example drawn in Fig. 15 illustrates Xyce S-parameter simulation of the n-channel MESFET connected as a two port device with bias tee networks isolating d.c. bias voltages Vds (source V2) and Vgs (source V1). The third example, Fig. 16, illustrates how Xyce .PARAM statements can be effectively used as a variable. In this example device temperature (parameter TEMP_C is stepped from -100^{o}C to 100^{o}C and the gate current plotted against Vgs. In all three test examples the Xyce subcircuit model and the Verilog-A module gave similar simulation results.



```
`include "disciplines.vams"
`include "constants.vams"
module TOM3(nD, nG, nS);
inout nD, nG, nS;
electrical nD, nG, nS,nGI, nGI2, nGI3, nDI, nDI2, nSI, nSI2;
electrical  nQGLA, nQGL,  nQGH, nFT, nSUBEXP1, nQGS;
electrical  nQGD, nI0,nIDS, nCDIS;
parameter real VT0=-1.8;   parameter real ALPHA=2.25;
parameter real BETA=3e-3;  parameter real GAMMA=0.015;
parameter real LAMBDA=0.02;   parameter real Q=2.0;
parameter real K=2.0;   parameter real VST=1.0;
parameter real MST=0.0;   parameter real TNOM_C=26.58;
parameter real TEMP_C=26.58;  parameter real  VT0TC=0.0;
parameter real  BETATC=0.0;  parameter real ALPHATC=0.0;
parameter real  GAMMATC=0.0;   parameter real RD=1.3;
parameter real  RG=5.0;  parameter real  RS=1.3;
parameter real LD=0.0;  parameter real LG=0.0;
parameter real LS=0.0;  parameter real IS = 1e-14;
parameter real N = 1.0;   parameter real CDS = 1e-13;
parameter real TAU = 0.0;  parameter real QGQL= 5e-16;
parameter real QGQH = -2e-16; parameter real QGI0 = 1e-6;
parameter real QGAG = 1.0;   parameter real QGAD = 1.0;
parameter real QGGB = 100.0;  parameter real QGCL = 2e-16;
parameter real QGSH = 1e-16;  parameter real QGDH = 0.0;
parameter real QGG0 = 0.0;  parameter real QPART = 0.4;
parameter real ILK = 1e-9; parameter real PLK = 2.25;
parameter real TAUGD = 1e-9;
parameter real CTAU = 1e-15;
real I1, I2, I3, TEMP_K, TNOM_K, VTH, nVST, nU, nVG, nFK, P1, P2;
real  VT0_T2, ALPHA_T2, BETA_T2,  GAMMA_T2, RTAU;
analog begin
TEMP_K = TEMP_C+273;  TNOM_K = TNOM_K+273;
VTH = `P_K*TEMP_K/`P_Q;
VT0_T2 = VT0+VT0TC*(TEMP_K-TNOM_K);
ALPHA_T2 = ALPHA*pow(1.01,ALPHATC*(TEMP_K-TNOM_K)) ;
BETA_T2 = BETA*pow(1.01,BETATC*(TEMP_K-TNOM_K));
GAMMA_T2 = GAMMA+GAMMATC*(TEMP_K-TNOM_K);
nVST = VST*(1.0+MST*V(nDI,nSI));
nU= (V(nGI,nSI)-VT0_T2+GAMMA_T2*V(nDI,nSI))/(1e-3+Q*nVST);
nVG=Q*nVST*ln(1.0+limexp(nU));
P1 = pow(ALPHA*V(nDI,nSI),K);  P2 = pow(1.0+P1, (1.0/(1e-2+K)));
nFK=ALPHA_T2*V(nDI,nSI)/P2;  RTAU = TAUGD/CTAU;
V(nG, nGI2) <+ ddt(LG*I(nG, nGI2));
I(nGI, nGI3) <+ V(nGI, nGI3)/0.1;  I(nGI2, nGI) <+ V(nGI2, nGI)/(RG);
I(nGI, nDI) <+ IS*(limexp(V(nGI,nDI)/(N*VTH))-1.0);
I( nGI3, nDI) <+ ILK*(1.0-limexp(-V(nGI3,nDI)/PLK) ) ;
I(nGI3, nSI) <+  ILK*(1.0-limexp(-V(nGI3,nSI)/PLK)) ;
I(nGI, nSI) <+ IS*(limexp(V(nGI,nSI)/(N*VTH))-1.0);  I(nDI,nSI) <+  V(nIDS);
I(nGI3, nDI) <+ ddt( (1.0-QPART)*V(nSUBEXP1)+QGG0*V(nGI, nDI));
I(nGI3, nSI) <+ ddt(QPART*V(nSUBEXP1)+QGG0*V(nGI, nSI));
I(nDI, nSI) <+ ddt( CDS*V(nDI, nSI) +TAU*V(nIDS) );
I(nDI, nCDIS) <+ V(nDI, nCDIS)/RTAU;  I(nGI3, nGI) <+ 10*abs(V(nCDIS, nSI));
I(nCDIS,nSI) <+ ddt(CTAU*V(nCDIS, nSI));  I(nI0) <+ V(nI0);
I(nI0) <+  BETA_T2*pow(nVG,Q)*nFK;  I(nIDS) <+ V(nIDS);
I(nIDS) <+  V( nI0)*(1.0+LAMBDA*V(nDI, nSI));  I(nQGLA) <+ V(nQGLA);
I(nQGLA) <+ QGQL*exp(QGAG*(V(nGI,nSI)+V(nGI,nDI)))*cosh(QGAD*V(nDI,nSI));
I(nQGL) <+  V(nQGLA)+QGCL*(V(nGI, nSI) +V(nGI, nDI));
I(nQGL) <+ V(nQGL);  I(nQGH) <+ V(nQGH);
I(nQGH) <+  QGQH*ln(1.0+V(nIDS)/QGI0) + QGSH*V(nGI, nSI) + QGDH*V(nGI,nDI);
I(nFT) <+ V(nFT);
I(nFT) <+  exp(-QGGB*V(nDI, nSI)*V(nIDS));  I(nSUBEXP1) <+ V(nSUBEXP1);
I(nSUBEXP1) <+ V( nQGL)*V(nFT) + V(nQGH)*(1-V(nFT)) -QGQL;
I(nQGS) <+ V(nQGS);
I(nQGS) <+ QPART*V(nSUBEXP1)+QGG0*V(nGI, nSI);  I(nQGD) <+ V(nQGD);
I(nQGD) <+  (1.0-QPART)*V(nSUBEXP1)+QGG0*V(nGI, nDI);
V(nS, nSI2) <+ ddt(LS*I(nS, nSI2));  I(nSI, nSI2) <+ V(nSI, nSI2)/(RS);
V(nD, nDI2) <+ ddt(LD*I(nD, nDI2));  I(nDI, nDI2) <+ V(nDI, nDI2)/(RD);
end
endmodule
```

Fig. 13. Extracted n-channel GaAs MESFET Verilog-A module code.

VI. Conclusions

Building behavioural semiconductor compact device models with SPICE has become popular among the modeling community due the interactive nature of schematic capture linked to circuit simulation. The ability to design and test small sections of a model, as part of the overall model design process, has its obvious advantages. Conversely, model developers may prefer a more direct approach where compact models are constructed

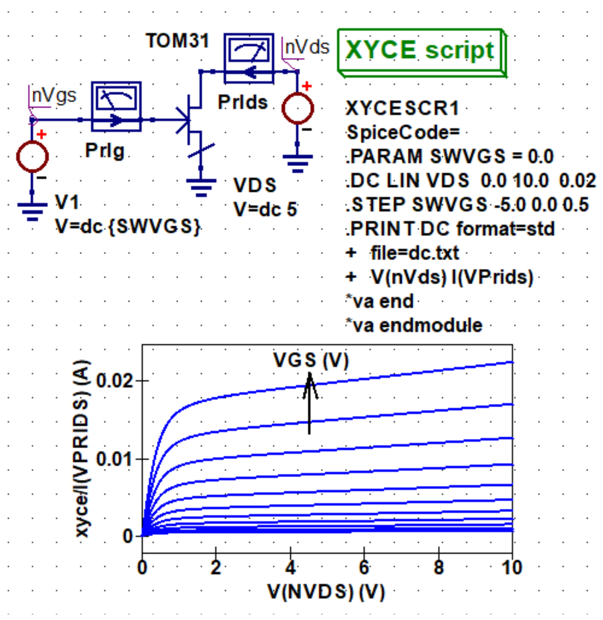

```
XYCESCR1
SpiceCode=
.PARAM SWVGS = 0.0
.DC LIN VDS  0.0 10.0  0.02
.STEP SWVGS -5.0 0.0 0.5
.PRINT DC format=std
+  file=dc.txt
+  V(nVds) I(VPrids)
*va end
*va endmodule
```

Fig. 14. N-channel TOM3 MESFET Ids/Vds d.c. test bench and simulation output characteristic plot.

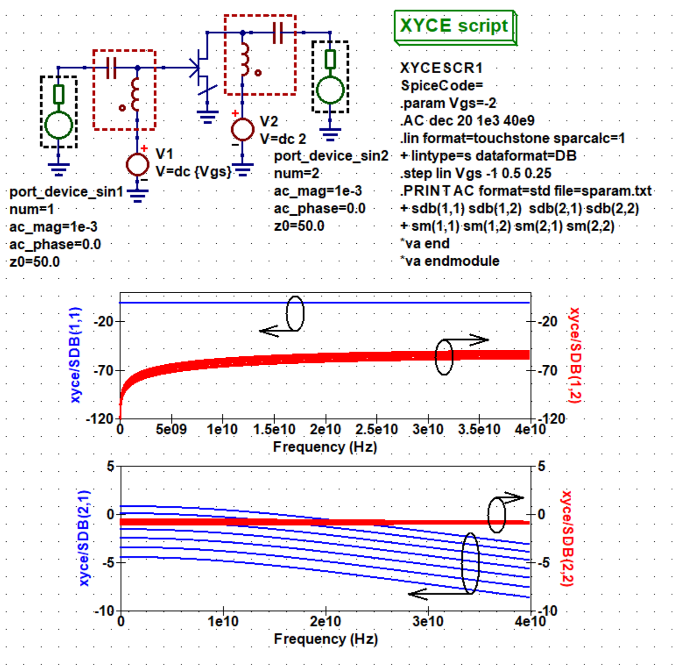

```
XYCESCR1
SpiceCode=
.param Vgs=-2
.AC dec 20 1e3 40e9
.lin format=touchstone sparcalc=1
+ lintype=s dataformat=DB
.step lin Vgs -1 0.5 0.25
.PRINT AC format=std file=sparam.txt
+ sdb(1,1) sdb(1,2)  sdb(2,1) sdb(2,2)
+ sm(1,1) sm(1,2) sm(2,1) sm(2,2)
*va end
*va endmodule
```

Fig. 15. N-channel TOM3 MESFET S-parameter two port test bench and simulation plots: SdB(1,1), SdB(1,2), SdB(2,1) and SdB(2,2) against frequency.

as Verilog-A module code and compiled as each stage in the overall design is completed. The advances in schematic capture reported here offer a ”half way house” between interactive behavioural modeling and direct coding of model with Verilog-A HDL. The extended schematic capture technique has the advantage that under development and testing it generates directly a Verilog-A module with a similar performance to

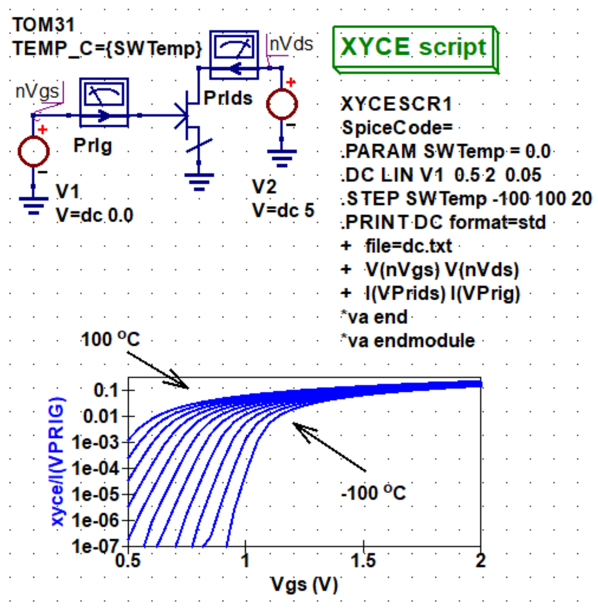

```
XYCESCR1
SpiceCode=
.PARAM SWTemp = 0.0
.DC LIN V1  0.5 2  0.05
.STEP SWTemp -100 100 20
.PRINT DC format=std
+  file=dc.txt
+  V(nVgs) V(nVds)
+  I(VPrids) I(VPrig)
*va end
*va endmodule
```

Fig. 16. N-channel TOM3 MESFET device temperature scan test bench and Igs/Vgs plots over the temperature range -100 ^{o}C to 100 ^{o}C.

a Xyce behvioural device model .

REFERENCES

[1] A.R. Newton, D. O. Pederson, A. Sangiovanni-Vincentelli, “SPICE Version 2g User’s Guide”, Department of Electrical Engineering and Computer Sciences, University of California: Berkeley, CA. 1981.

[2] B. Johnson, T. Quarles, A.R. Newton, D. O. Pederson, A. Sangiovanni-Vincentelli, “SPICE3 Version 3f User’s Manual”, Department of Electrical Engineering and Computer Sciences, University of California: Berkeley, CA. 1992.

[3] Ngspice mixed-level/mixed-signal circuit simulator based on Berkeley SPICE 3f5. Ngspice project team. Version ngspice-36. [Accessed March 2022] [Online] Available http://www.ngspice.sourceforge.net.

[4] Xyce parallel electronic circuit simulator. Sandia National Laboratories. Version 7.4. [Accessed March 2022] [Online] Available https://xyce.sandia.gov/.

[5] Accellera, Verilog-AMS Language Reference Manual. Version 2.2. [Accessed March 2022] [Online] Available http://www.accellera.org,

[6] J.A.Connelly and Pyung Choi, ”Macromodeling with SPICE”. Englewood Ciffs, New Jersey:Prentice Hall. 1992.

[7] GNU Octave Scientific Programing Language. Octave development team. Version 6.4.0. [Accessed March 2022]. [Online] Available https://www.gnu.org/software/octave/download.

[8] V. Kusnetsov and M. Brinson, ”Qucs-S: Qucs with SPICE”. Version 0.0.22. [Accessed March 2022] [Online] Available https://ra3xdh.github.io/.

[9] R.B Hallgren and D.S Smith, ”TOM3 Equations” Triquint Internal Report (unnumbered), 2 December 1999.

[10] R.B. Hallgren and P.H. litzenberg. ”TOM3 Capacitance Model Linking Large- and Small-signal MESFET Models on SPICE.” IEEE Trans. Microwave Theory Tech., vol. 47, no. 5. pp. 556-561, 1999.

[11] QucsStudio - A Free and Powerful Circuit Simulator. Michael Margraf. Version 4.2.2. [Accessed March 2022]. [Online] Available http://qucsstudio.de.

[12] P. Kuthe, M. Müller and M. Schröter, ”VerilogAE: An Open Source Verilog-A Compiler for Compact Model Parameter Extraction,” IEEE Journal of the Electron Devices Society, vol. 8, pp. 1416-1423, 2020.