IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 11, Issue 1, Ser. I (Jan. - Feb. 2021), PP 22-28
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

MOSFET Compact Verilog-A Model Implementation in S-Edit

George P. Patsis[1,2]
[1]*Department of Electrical and Electronics Engineering, University of West Attica, Athens, Greece*
[2]*Nanometrisis private company, Scientific & Technological Park "Leukippos", NCSR Demokritos, Neapoleos 27 & Patr. Grigoriou Str., 15341, Agia Paraskevi, Attica, Greece*
Corresponding Author: George P. Patsis

Abstract: *A simple version of the EKV MOSFET model is implemented in Verilog-A and tested with Tanner EDA software suite using S-Edit and T-Spice. The process of model development in Verilog-A and its integration into the software's library is discussed in detail. The aim of this work is to present the advantages of analog modeling with hardware description languages, especially when developing nonlinear device models, and to present the details of implementing a Verilog-A EKV MOSFET model in the simulator.*

Keywords *– EKV MOSFET, Verilog-A, S-Edit, T-Spice, Tanner-EDA.*

Date of Submission: 12-02-2021 Date of Acceptance: 25-02-2021

I. INTRODUCTION

Analog device-modeling enables circuit-designers to capture high-level behavioral descriptions of components in a precise set of mathematical terms. An analog model should ideally model the characteristics of the behavior as accurately as possible, with the trade-off of model complexity, which is usually manifested by reduced execution speed. A compact-transistor-model is a model of a transistor's currents and voltages, built from physics-based equations, intended for use in an analog circuit simulator. In order for the model to be incorporated in the simulator, it has fist to be coded in a suitable hardware description language.

Hardware description languages (HDLs) were developed as a means to provide varying levels of abstraction to designers. HDLs allow a high level description of the model and simulation synthesis programs can then take the language and generate the gate level description. Verilog and VHDL are currently the two dominant languages. In this article the focus is on Verilog-A.

Verilog-A provides a high-level language to describe the analog behavior of conservative systems. The disciplines and natures of the Verilog-A language enable designers to reflect the potential and flow descriptions of electrical, mechanical, thermal, and other systems. It is a procedural language, with constructs similar to C and other languages. It provides simple constructs to describe the model behavior for the simulator program.

The advantages of Verilog-A is basically the faster implementation of the device model compared to implementation in C or Fortran language. It is a simple language and most of its concepts can be learned from the study of simple examples. Another important advantage is that quantity-derivatives are straight-forward coded. However there are also implementation problems such as slow performance, poor convergence and inconsistent results between simulations, if the developer is not careful.

In a former article [**1**], a simple version of the EKV MOSFET model was implemented in Verilog-A and tested in Keysight's Genesys software suite. In this work the same model is modified for incorporation in the Tanner EDA software suite. The purpose of the current work is mainly educational in order to help undergraduate students get a better understanding of the various tools used in VLSI design. This work follows a line of related articles [**2-6**].

In section II, an introduction to S-Edit is presented, along with the necessary steps for the model structuring in Verilog-A. Then in section III, the implementation of the model follows, along with a basic simulation for testing its validity.

II. PROJECT STRUCTURE IN S-EDIT

It is necessary to understand the basic project structure and terminology used in S-Edit [**7**]. This is a schematic capture tool within the Tanner EDA suite (today owned by Mentor Graphics) for VLSI design. Specifically, it provides the user with graphical tools to design circuits and then creates detailed SPICE netlists for simulation with T-Spice simulator. The highest level entity in the S-Edit schematic database hierarchy is the design. A design contains many cells, some of which may be referenced from a library. Most often a cell contains a single interface, which contains a single symbol view and a single schematic view.

A view is simply a component of a cell definition. Each view provides a different way of depicting a cell. S-Edit view types are symbols, schematic, and interface. Since T-Spice can run Verilog-A and Verilog-AMS, simulating a schematic with sub-cells that have either a Verilog-A or AMS description will simulate the Verilog code; so S-Edit has SPICE, Verilog-A, and Verilog-AMS views. Verilog-A is the analog-only subset to Verilog-AMS. It is intended to allow users of SPICE class simulators create device and system models for their simulations.

Verilog-A and Verilog-AMS files have different file extension when the cell is netlisted. Verilog-AMS views are written as ***.vams** files and Verilog-A views are written as ***.va** files. Verilog view types are written as ***.v** files. Verilog should be used for digital RTL and structural Verilog (netlists). Verilog-A should be used for Verilog-A code. Finally, Verilog-AMS should be used for code that mixes Verilog-A and Verilog constructs in the same cell.

One way to proceed with the creation of a design is to create a circuit schematic, then an instance of the circuit and finally a symbol for the circuit. A schematic for a component is made by placing a number of symbols on the schematic page and then connecting the symbols together with wires at specific connection points. A set of connection points connected together is called a net. A symbol of the new schematic can then be made and then be use in other schematics, thus allowing for the construction of hierarchical designs.

Instances in a cell are references to other cells. These cells might be common library cells such as basic circuit elements or larger, custom-designed cells. Instances are dynamically linked to their source cell so that any change you make to a source cell is reflected in each higher-level instance of that cell. Any option, set in the originating cell will be the default for all of its instances. However, you can override this setting in any of the instances, and the change will affect that instance only.

It is also possible to start the process of schematic capture with the creation of device symbols. These, are a pictorial representation of an electrical component together with a definition of the electrical connections that can be made to that component with other components from libraries or also other newly-created components. Symbols may also contain properties (parameters) that define the electrical characteristics of the component. In many design situations, a library of symbols already exists, usually of basic standardized components which a designer can use to create their schematic. In S-Edit this can be done through the **Cell ➔ New View ➔ Symbol View Type** as is seen in **Fig. 1**. Then a schematic window opens up where the user can draw the symbol for the new component, and place input-output ports on it in order to make connections with other circuit elements. (See **Fig. 2** in section III). If a symbol is created for a corresponding Verilog-A description (code within a file), then the ports are automatically placed on the symbol schematic along with the parameters of the code, as symbol-properties (see **Fig. 2** in section III).

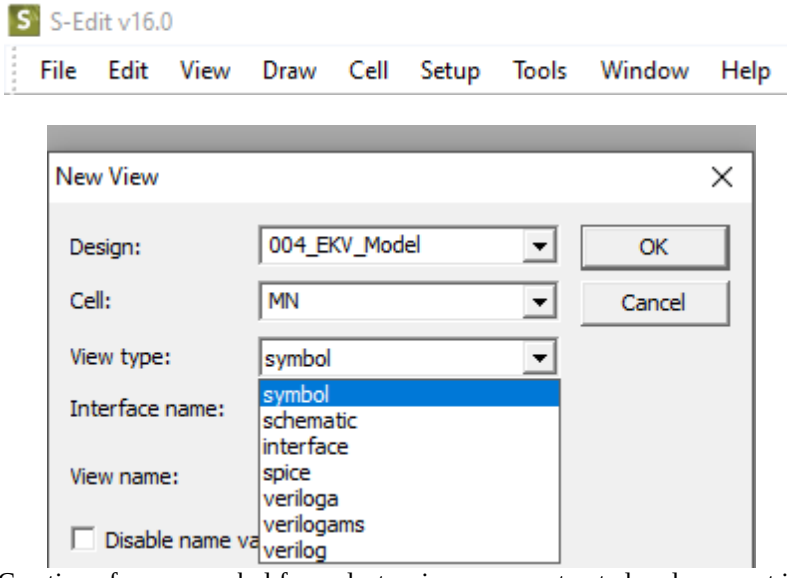


Figure 1. Creation of a new symbol for a electronic component not already present in a library.

III. VERILOG-A CODING OF THE MODEL

The simple version of the EKV model described in [**1, 8-12**] is implemented here in Verilog-A, and incorporated in S-Edit as a veriloga-view (saved in the project directory in a text-file named **MN.va**). The code begins with the calling of several header files. Specifically the "disciplines.vams", the "constants.vams", and the "compact.vams" files (**Listing 1**). These files contain definition of quantities that are useful in the following code. For example "constants.vams" contains Verilog-A definition of mathematical and physical constants. The

"disciplines.vams" contains Verilog-A definition of natures and disciplines. Finally, the "compact.vams" file contains various useful, common macro definition and utilities. Using these header files it is easier to develop the code for the MOSFET.

The body of the code should begin with a module definition with the name of the device and the terminal names, specifying the input-output nodes and the nature (electrical in this case) of these nodes (d, g, s, b). Any other intrinsic node not seen on the outside world is also defined here (di, gi, si, bi). Several parameters, characteristic of the model, should be defined with the "parameter" keyword, while any other variable used for holding intermediate calculations and data should be defined with an appropriate data type, e.g. as real variable. All the quantities used in the model equations are normalized depending on their nature. This way simpler equations occur, which are easier to handle computationally.

In S-Edit environment, select **Cell➔New View** and create a veriloga view, a symbol view and an interface view under the cell name: e.g., **MN**. The code of veriloga view is seen in **Listing 1**. The corresponding symbol and interface view are seen in **Fig. 2** and **Fig. 3** respectively. For the details of the model's physics refer to [**1**].

Listing 1. Verilog-A code of the veriloga-view of the cell MN (nMOS transistor).

```
// Module contents
`include "disciplines.vams"
`include "constants.vams"
`include "compact.vams"
`define    MY_P_ESI   11.7*`P_EPS0

module MN(d,g,s,b);
inout d,g,s,b;
electrical d ,g ,s ,b ; //input-output nodes
electrical di,gi,si,bi; //intrinsic nodes

parameter integer SIGN=1.0      from [-1:1] exclude 0;
parameter real    W=1e-5        from [0:inf];
parameter real    L=1e-5        from [0:inf];
parameter real    VTO=0.5       from [0:inf];
parameter real    PHI=0.9       from [0:inf];
parameter real    GAMMA=0.9     from [0:inf];
parameter real    KP=100e-6     from [0:inf];
parameter real    COX=11.50e-3  from [0:inf];
parameter real    UCRIT=3.8e6   from [0:inf];
parameter real    XJ=50.00e-9   from [0:inf];
parameter real    IBA=0.0       from [0:inf];
parameter real    IBB=300e6     from [0:inf];
parameter real    IBN=1.0       from [0:inf];
parameter real    RL=1.0e-6     from [0:inf];
parameter real    CGXO=1.0e-4   from [0:inf];

real VC,VDSS, VD,VG,VS,VG_PRIME,VP,n,BETA,Ispec,vps,vpd;
real qr,qf,z0,zk,yk,if_,ir,IDS,xf,xr;
//qs
real Q0,QS,QD,QI,QG,QB,nq,dqd,dqs,dqb,dqg;
//IDB
real LC,V_IB,IDB;

analog

begin

//extrinsic part
I(d,di) <+ (W/RL)*V(d,di);
I(s,si) <+ (W/RL)*V(s,si);
V(g,gi) <+ 0;   //zero resistance
V(b,bi) <+ 0;   //zero resistance
I(d,g) <+ (CGXO*W)*ddt(V(d,g));
I(s,g) <+ (CGXO*W)*ddt(V(s,g));

//intrinsic part
VD = SIGN*V(di,bi);
VG = SIGN*V(gi,bi);
VS = SIGN*V(si,bi);

//CALCULATE PINCH-OFF VOLTAGE VP
VG_PRIME = VG - VTO + PHI +GAMMA*sqrt(PHI);
VP = VG_PRIME - PHI - GAMMA*(sqrt(VG_PRIME+GAMMA*GAMMA/4)-GAMMA/2);
```

```
//CALCULATE SLOPE COEFFICIENT n
n=1+GAMMA/(2.0*sqrt(PHI+VP));

//CALCULATE NORMALIZING CURRENT COEFFICIENT
BETA = KP*W/L;
Ispec = 2*n*BETA*$vt*$vt;

//CALCULATION OF FORWARD CURRENT COMPONENT (SOURCE TERMINAL)
//INVERSION OF FUNCTION F, USING THE ITERATIVE NEWTON-RAPHSON METHOD
vps = (VP-VS)/$vt;
if (vps<-18) vps=-18; //PREVENT OPERATION WITH VERY SMALL ARGUMENTS
z0 = (vps>-0.35) ? 2/(1.3+vps-ln(vps+1.6)) : 1.55+exp(-vps);
zk = (2+z0)/(1+vps+ln(z0));
yk = (1+vps+ln(zk))/(2+zk);
qf = yk;
if_ = qf*qf+qf;

//CALCULATION OF REVERSE CURRENT COMPONENT (DRAIN TERMINAL)
//INVERSION OF FUNCTION F, USING THE ITERATIVE NEWTON-RAPHSON METHOD
vpd = (VP-VD)/$vt;
if (vpd<-20) vpd=-20; //PREVENT OPERATION WITH VERY SMALL ARGUMENTS
z0 = (vpd>-0.35) ? 2/(1.3+vpd-ln(vpd+1.6)) : 1.55+exp(-vpd);
zk = (2+z0)/(1+vpd+ln(z0));
yk = (1+vpd+ln(zk))/(2+zk);
qr = yk;
ir = qr*qr+qr;

//IDS – CHANNEL CURRENT CALCULATION
IDS = Ispec * (if_-ir);

// IDB-IMPACT IONIZATION CURRENT CALCULATION
//velocity saturation voltage
VC = UCRIT*L;
VDSS = VC*(sqrt(0.25+($vt*sqrt(if_)/VC))-0.5);
LC = sqrt(`MY_P_ESI*XJ/COX);
V_IB = VD - VS - IBN*2*VDSS;
if (V_IB>0) IDB=IDS*V_IB*exp(-IBB*LC/V_IB)*IBA/IBB;
else        IDB=0;

//QUASI-STATIC BEHAVIOR
xf= sqrt(0.25+if_);
xr= sqrt(0.25+ir);
nq = 1 + 0.5*GAMMA/sqrt(PHI+0.5*VP+1e-6);
Q0 = 2*nq*$vt*COX;
QD= -(0.5*Q0*W*L)*((4.0/15.0)*
((3*pow(xr,3)+6*pow(xr,2)*xf+4*pow(xf,2)*xr+2*pow(xf,3))/pow(xf+xr,2)-0.5));
QS= -(0.5*Q0*W*L)*((4.0/15.0)*
((3*pow(xf,3)+6*pow(xf,2)*xr+4*pow(xr,2)*xf+2*pow(xr,3))/pow(xf+xr,2)-0.5));
QI = QS+QD;
QB= - GAMMA*COX*W*L*sqrt(VP+PHI) - (nq-1)*QI/nq;
// QG= - QI - QB;
dqd=ddt(QB);
dqs=ddt(QS);
dqb=ddt(QB);

I(di,si) <+ IDS;
I(di,bi) <+ IDB;
I(di,gi) <+ dqd;
I(si,gi) <+ dqs;
I(bi,gi) <+ dqb;

end

endmodule // MN
```

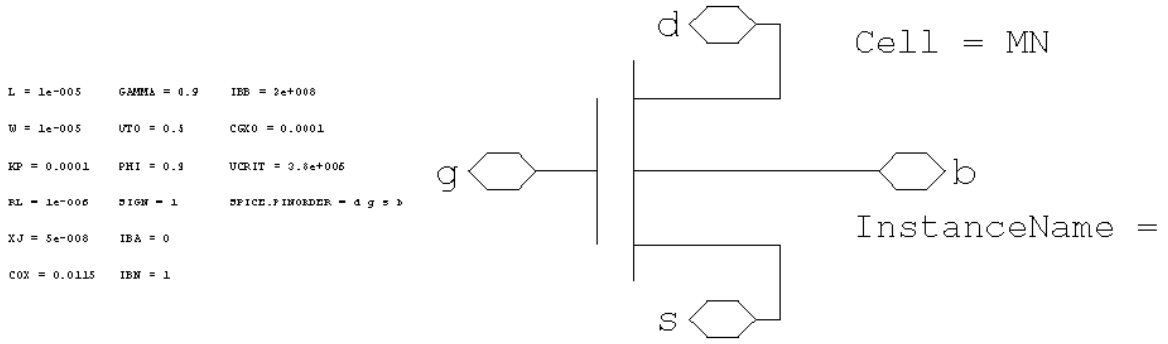


Figure 2. MN symbol view.

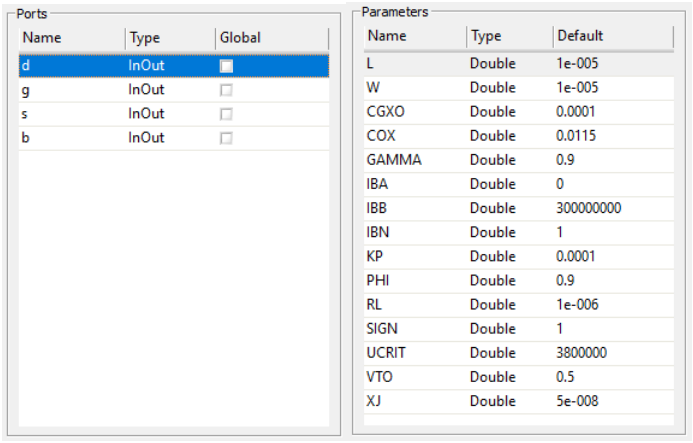
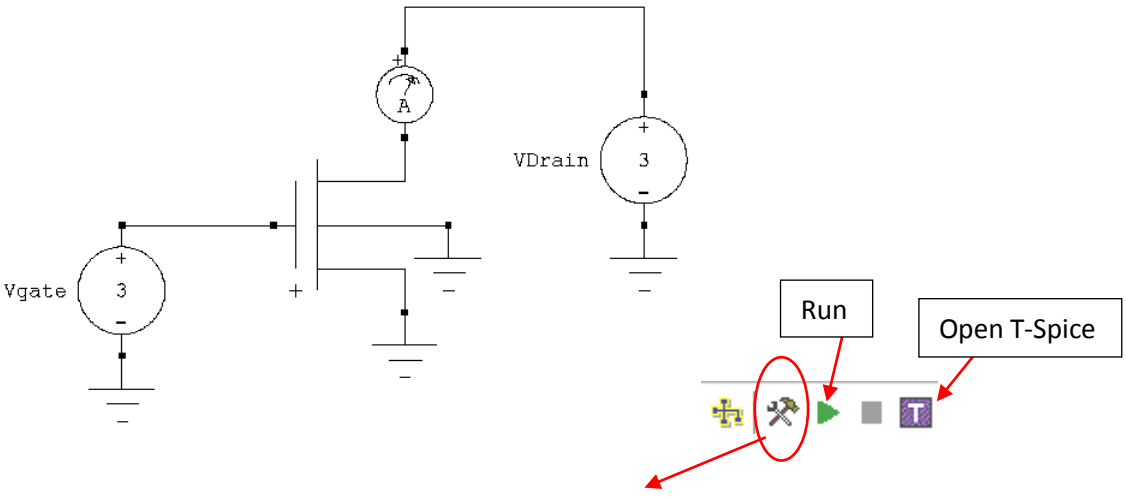


Figure 3. MN interface view.

Selecting **Cell ➔ New View ➔ Schematic** opens up a schematic window where the newly created symbol **MN** can be placed along with voltage sources and an ampere-meter as shown in **Fig. 4**. In order to prepare the schematic for simulation it is important to setup the Spice options as well with the type of analysis to be performed. Pressing the "**T**" symbol on the file menu opens T-Spice and brings up the netlist of the design with all the details for the analysis (**Listing 2**).



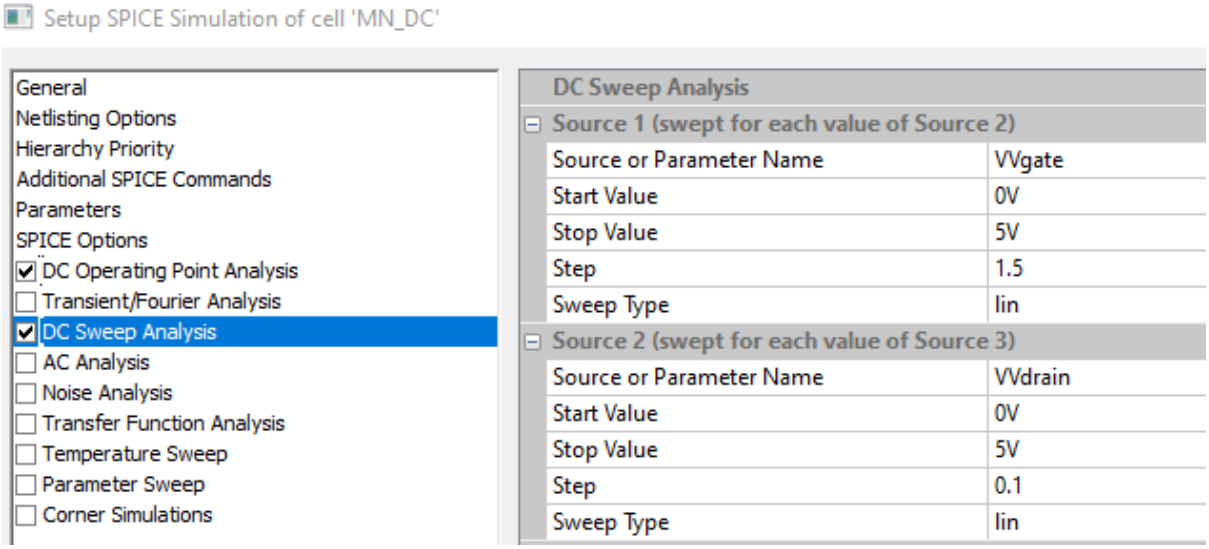


Figure 4. MN circuit schematic view and corresponding Spice setup options.

Command **.hdl** loads a Verilog-A module into the simulator (T-Spice), and optionally declares that the module will be used to simulate all devices of a specified type, level, and version. The command syntax [**13**] is: **.hdl** *filename [modulename [***type***=***type [***level***=***level [***version***=***version]]]]*. *filename* stands for the Verilog-A file to be loaded. It must exist in the current directory or in the T-Spice Verilog-A search path. Absolute or relative path names (according to the conventions of the operating system) can be used. If the referenced filename or path contains a space, enclose the entire path in single or double quotation marks. *modulename* is the name of a specific module within the file. Only this one module will be loaded into the simulator, and other modules within the file will be ignored. *type* is the type of devices that will use this module. *level* is the level number of the models that will use this module. *version* is the version number of the models that will use this module.

When the type, level, and version parameters are used in the **.hdl** command, T-Spice will use the Verilog-A module as the modeling code for all matching devices, i.e. devices of the specified type which reference a .model of the specified level and version. This capability may be used to either introduce new model levels and versions into T-Spice, or to replace the existing built-in T-Spice models with user defined.

The device type keyword is limited to the following: C or capacitor for capacitors, D or diode for diodes, J of jfet for JFETS, R or resistor for resistors, M or mosfet for MOSFETs, Q or bipolar for BJTs, and Z or mesfet for MESFETs. Example: .hdl bsim3v.34.va bsim3 type=mosfet level=49 version =3.4. This example demonstrates how a Verilog-A module can be used instead of the built-in device evaluation code. In this case, a Verilog-A representation of the BSIM3 model will be used instead of T-Spice's internal BSIM3 analysis code for those MOSFETs which reference a model that is level 49 and version 3.4. If the level or version is not specified or has a value of zero, then all levels and versions will be matched and simulated using the Verilog-A module.

Listing 2 shows the example of loading the above developed MOSFET model into a SPICE deck using the **.hdl** command and simulating the device in order to obtain its IV characteristic. Executing the Spice code, brings up the W-Edit window where the simulation results are presented as seen in **Fig. 5**. The familial IV curves of the MOSFET are produced.

Listing 2.

```
********* Simulation Settings - General Section *********
.option tnom=27

*************** Subcircuits *****************
.hdl MN.va

***** Top Level *****
XMN1 N_1 N_3 Gnd Gnd MN
+CGXO=100u COX=11.5m GAMMA=900m IBA=0 IBB=300X
+IBN=1 KP=100u L=10u PHI=900m RL=1u SIGN=1 UCRIT=3.8X VTO=500m
+W=10u XJ=50n
VVDrain N_2 Gnd  DC 3
VVgate N_3 Gnd  DC 3
VAm1 N_2 N_1 0v
.PRINT   I(VAm1)
```

```
*.MEASURE     avg I(VAm1)

********* Simulation Settings - Analysis Section *********
.op
.dc lin VVgate 0V 5V 1 SWEEP lin VVdrain 0V 5V 1

.end
```

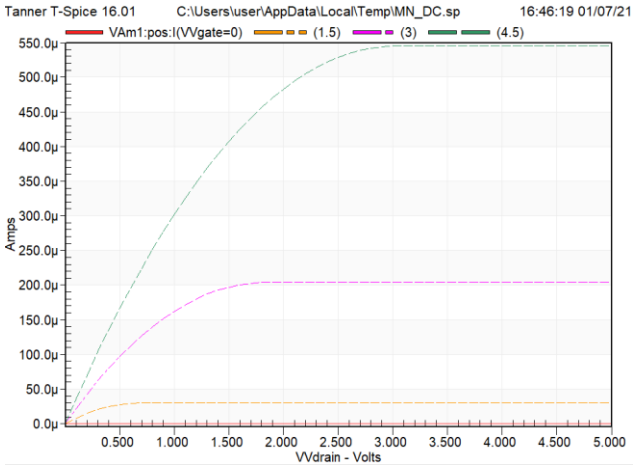


Figure 5. Simulation output. Verification of MOSFET model validity.

IV. CONCLUSION

In the current work, a Verilog-A model of a MOSFET was incorporated into S-Edit and T-Spice and a valid simulation-component (symbol) of the code was created that can be used along with other SPICE elements in a common circuit schematic. The whole process was described with only the absolutely necessary details for carrying the process from design to simulation and to final results. The interested reader should repeat the details of the article to reproduce the final results in order to get familiarity with the various stages of the design process in S-Edit and T-Spice.

REFERENCES

[1] G. P. Patsis, *MOSFET EKV Verilog-A Model Implementation in Genesys*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 1, 2018, pp. 73-86.
[2] G. P. Patsis, *Process-simulation-flow and metrology of VLSI layout fine-features*, IOSR Journal of VLSI and Signal Processing, vol. 7, no. 6, 2017, 23-28.
[3] G. P. Patsis, *VHDL-AMS macromodels of MOSFET. Consideration of gate length variability and single-electron-transistors*, IOSR Journal of VLSI and Signal Processing , vol. 7, no. 6, 2017, 29-33
[4] G. P. Patsis, *Basic topologies of MOS single-stage amplifiers. DC analysis for maximum input-voltage swing and amplification*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 1, 2018, 47-59.
[5] G. P. Patsis, *Educational Introduction to VLSI Layout Design with Microwind*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 5, 2018, 18-29.
[6] G. P. Patsis, *Educational Introduction to CMOS Circuit Design with Texas Instrument ANalyzer (TINA)*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 6, 2018, 39-48.
[7] Tanner EDA, S-Edit User Guide contained in software's help directories. (S-Edit 16 User Guide.pdf) .
[8] C. C. Enz, F. Krummenacher, E. A. Vittoz, *An Analytical MOS Transistor Model Valid in All Regions of Operation and Dedicated to Low-Voltage and Low-Current Applications*, Analog Integrated Circuits and Signal Processing, 8, 1994, 83-114.
[9] A. S. Bazigos, *Implementation of EKV MOSFET model in Verilog-A*, (degree thesis, National Technical University of Athens, 2003).
[10] C. C. Enz, E. A. Vittoz, *Charge-based MOS Transistor Modeling. The EKV model for low-power and RFIC design*, (Wiley, 2006).
[11] M. Bucher, C. Lallement, C. Enz, F. Theodoloz, F. Krummenancher, *The EPFL-EKV MOSFET Model Equations for Simulation*, (Technical Report, Model Revision 2.6, 1997).
[12] D. FitzPatrick, I. Miller, *Analog Behavioral Modeling with the Verilog-A Language* (Kluwer Academic Publishers, 2003).
[13] Tanner EDA, Applications and Examples User Guide contained in software's help directories. (Tanner Tools Tutorial.pdf, Tanner Tools Examples Guide.pdf).

George P. Patsis . "MOSFET Compact Verilog-A Model Implementation in S-Edit." *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, vol. 11, no. 1, 2021, pp. 22-28.