NEEDS compact model release – lessons learned from MVS 1.0.0

Shaloo Rakheja and Dimitri Antoniadis

November 22, 2013

Thanks to:
Dr. Geoffrey Coram, Ujwal Radhakrishnan, Xingshu Sun

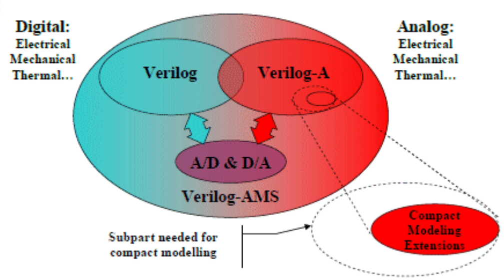
Massachusetts Institute of Technology

shaloo@mit.edu

This presentation focuses on:

I. MVS 1.0.1 package

II. Checklist & guidelines for model release



III. Good practices for writing Verilog-A models

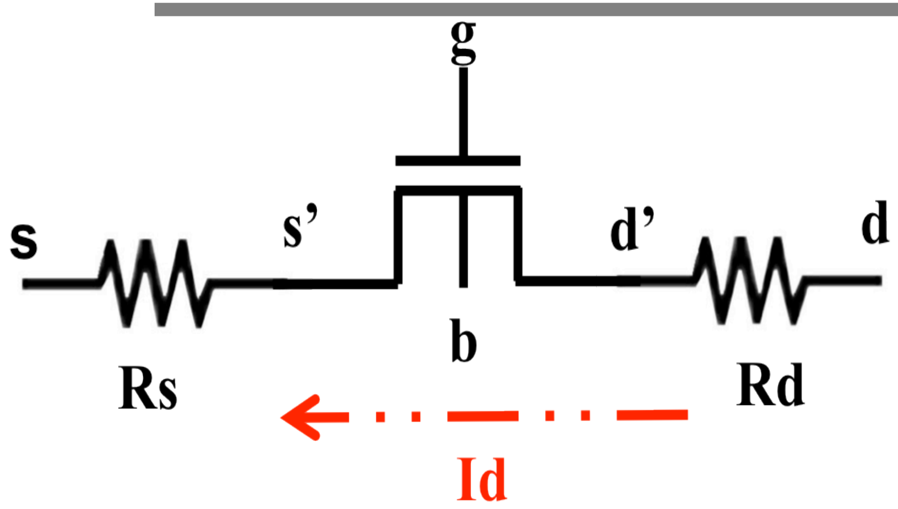
This presentation also briefly talks about:

IV. Updates (MVS 1.0.1) and open issues

Discussions

I. MVS PACKAGE ON NANOHUB

What is the MVS model ?



MIT **V**irtual **S**ource (MVS) Transistor model gives *currents* and *charges* as functions of terminal voltages.

Currents

Id = f(Vg,Vd,Vs,Vb)

Ig = Ib = 0

Charges

Qs = f1(Vg,Vd,Vs,Vb)

Qd = f2(Vg,Vd,Vs,Vb)

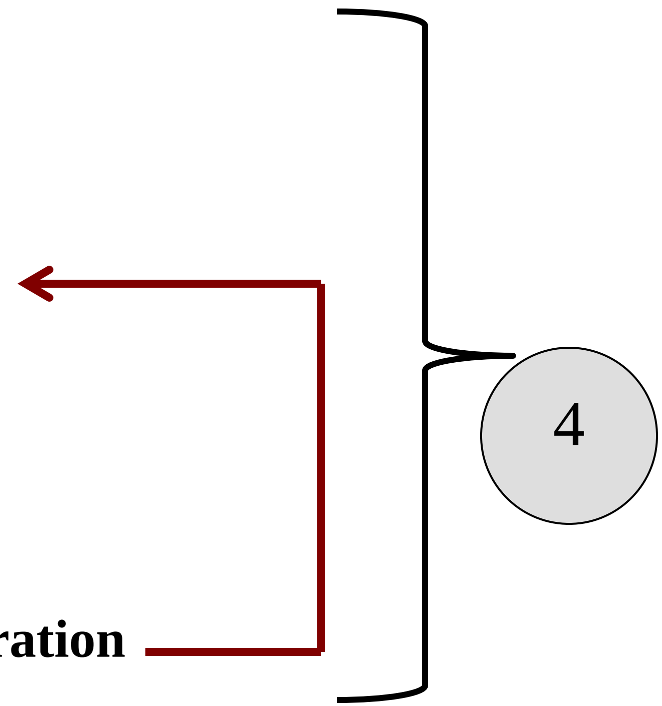
Qb = f3(Vg,Vd,Vs,Vb)

Qg = -(Qs+Qd+Qb)

NEEDS MTL

Release package components- 1/2

1. **MATLAB-related**
   i. Model implementation
   ii. Model exerciser
   iii. Numerical parameter extractor
2. **Verilog-related**
   i. Model implementation
   ii. Test-benches for simple circuits
3. **Experimental data for model calibration**
4. **Model manual**



Release package components- 2/2

5. **Update log (when a new version is released)**
6. **License agreement**

Link to the model on nanohub:

https://nanohub.org/resources/19684

Wiki for model-release checklist:

https://nanohub.org/groups/needs/wiki/SpecificInstructionsforNEEDSCompatibleCompactModels

II. CHECKLIST/GUIDELINES FOR MODEL RELEASE

Quick checklist for model release

Component	Associated files and/or requirements
MATLAB	• Model file Model exerciser • Parameter extraction (analytical/non-linear) • Readme file
Verilog-A	• Model file • SPECTRE/HSPICE netlists for simple circuits Readme file
Experimental data	• Readme file for data format and references
Model manual	• Explaining all of the model equations • Simulation results • Extraction methodology • Proper references

\+

CMC license agreement

Update log (if needed)

MATLAB

What must the model.m file contain?

1. Equations that describe the physics of the model.
2. A model header stating
    a. What the model returns (currents, charges etc …)
    b. Range of the model validity (limited bias etc …)
    c. The date last updated and by whom
3. Adequate comments to help understand the code and make debugging easier.

shaloo@mit.edu

MATLAB

What must the model.m file contain?

4. A parameter for model version.
   a. For MVS 1.0.0, ***version = 1.00***
   b. For MVS 1.0.1, ***version = 1.01***
5. Follow the ***CMC convention*** for assigning version to a model and its subsequent updates.
6. Model file name must match the module name.

NEEDS

MTL

Example from MVS 1.0.1

```
function [Idlog,Id,Qs,Qd,Qg,Qb,Vdsi_out]=mvs_si_1_0_1(coeff,bias_data)
% Symmetrical Short-Channel MOSFET model (VERSION=1.0.1)

% Returns the log of drain current, Id [A] and partitioned charges
% This model is only valid for Vg >~ Vg(psis=phif) where psis is the
surface
% potential.  I.e range of validity is from onset of weak inversion trhough
% strong inversion.

% Original Dimitri Antoniadis, MIT, 09/17/10
% Modified, DAA 10/20/12
% Modified, DAA 07/01/13
% Modified SR 07/24/13
% Modified SR 09/19/13
```

Model file is named as *mvs_si_1_0_1.m* to match with the module name.

Model header with appropriate information.

NEEDS

CMC convention for model versioning

<version#>.<subversion#>.<revision#>, where the numbers in angle brackets (< >) are integers.

a. **Model version number**: major model formulation change (i.e. not backward compatible with the previous release.)
b. **Subversion number**: minor model formulation change (i.e. no new parameters introduced, improve run-time efficiency, reset when model version is update.)
c. **Revision number**: identify different implementations of the same set of equations (numerical measures to improve convergence, restructuring code, smoothing functions, bug fixes that do not change model formulation.)

MATLAB

What is the purpose of model exerciser?

1. Plots currents, charges, and capacitances as functions of terminal voltages.

2. Also computes and plots 1st and 2nd derivatives of currents.

3. Values of various parameters in the model are either (i) obtained through parameter extractor or (ii) reasonable values must be used.

4. File must be properly commented.

Model exerciser can be tweaked to plot various other quantities as well (for example, derivatives of capacitances)

MATLAB

What does the parameter extractor file do?

1. Extracts ***selected parameters*** in the model upon calibration with experimental data (MVS 1.0.0 has 32 nm and 45 nm data on NFETs from Intel).
2. Uses MATLAB's built-in least square curve fit (**lsqcurvefit**) routine to extract parameters.
3. Specify an '***educated***' initial guess (mostly from some analytical method).
4. Important to specify ***lower and upper constraints*** on parameters to achieve "physically realistic" results.

NEEDS

MTL

This is how the crux of the parameter extraction script in MVS looks like:

```
%% now we run optimization. Make a matrix of initial guess
options = optimset('Display','iter','TolFun',1e-11);

lb=[1;1;0;1;0;0.1;50;0.2]; % lower bound constraints
ub=[500;500;0.5;2;0.5;10;1000;0.8]; % upper bound constraints
```

Bounds for params

```
% Optimization routine
[coeff_op_tran,resnorm,residual,exitflag] = lsqcurvefit(@mvs_si_1_0_1,↙
coeff_init,bias_data,log10(Id_data),lb,ub,options);
```

x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub)

starts at x0 and finds coefficients x to best fit the nonlinear function fun(x,xdata) to the data ydata (in the least-squares sense). ydata must be the same size as the vector (or matrix) F returned by fun. **lb** stands for the lower bound, while **ub** stands for the upper bound on the parameter set.

Verilog-A

What must the model.va file contain?

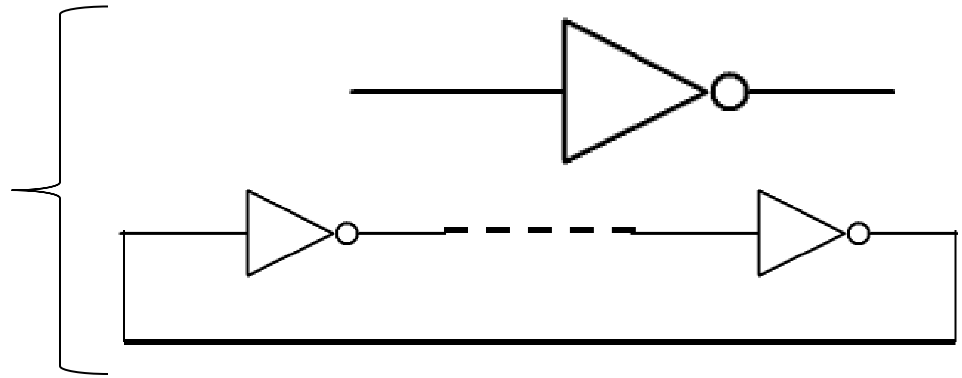
1. Equations describing model physics
2. Real parameter for model version
3. Header (changes/license agreement)
4. Variable names must be consistent with what is implemented in MATLAB.
5. Adequate comments to make understanding and debugging easier.

NEEDS

Verilog-A

Simulation Decks

HSPICE/SPECTRE netlist for simulating simple circuits (inverter, RO etc.)



1. Circuit netlist must note (i) loading scenario, (ii) type of run (DC/AC/TRAN….), (iii) choice of input parameters, (iv) choice of simulator options (noting simulator version).
2. ***Readme file*** must list simulator version for queer simulator version dependencies.
3. Simulation results must be properly ***documented*** (*.pdf).

NEEDS

MTL

Verilog-A

Simulation deck for inverter transient simulation example from MVS 1.0.1

Header

```
// Test-bench for generting transient response of an inverter with two loading scenarios:
// scenario 1: You can choose either c1=1pF (constant loading) by uncommenting line 25
// scenario 2: You can have lines 26-29 uncommented for simulating a fan-out of FOUR.

// TESTED ON SPECTRE Version 10.1.1.374.isr21
simulator lang=spectre
ahdl_include "mvs_si_1_0_1.va"
format options rawfmt=psfascii
```

SPECTRE version

{More code ….}

Simulator dependencies

```
// add cmin=1e-18 in simulatorOptions line to achieve better convergence.
// check for cmin compatability with older Spectre versions.

// following lines can be uncommented to store results in rawfile.
```

Other info.

```
//modelParameter info what=captab where=rawfile
//element info what=inst where=rawfile
//outputParameter info what=output where=rawfile
//designParamVals info what=parameters where=rawfile
//primitives info what=primitives where=rawfile
//subckts info what=subckts  where=rawfile
//saveOptions options save=allpub
```

cmin adds a small cap from a node to ground (avoids infinitely fast transitions)

Experimental data

Some guidelines for presenting the data:

1. Experimental data sets are used for ***parameter extraction***.
2. ***Format*** experimental data in a manner that is easily read by the software.
3. Include a ***readme file*** that explains that format of the experimental data (what different columns represent …)
4. Properly ***cite*** the source of the experimental data both in the readme file and the model manual.

NEEDS

Experimental Data

Example from parameter extractor in MVS

Transfer curve data

VGS (V)	ID (A) (VDS = 50 mV)	ID (A) (VDS = 1V)
0.0	--	--
0.1	--	--
--	--	--
--	--	--
--	--	--
--	--	--

Output curve data

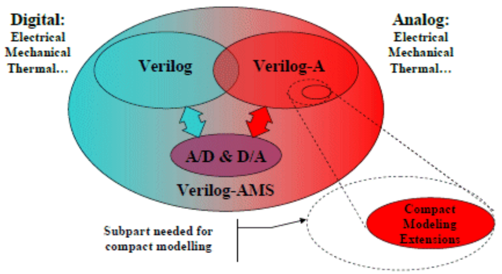
VDS (V)	VGS (V)	ID (A)
0.0	--	--
0.05	--	--
--	--	--
--	--	--
--	--	--
--	--	--

The model manual must contain all of the following:

1. Model physics
2. All equations used in the model with an explanation of all variables and their units
   i. Variable names must be consistent throughout.
3. Functionality of extraction routine
4. Sample experimental data sets and their formatting
5. Simulation results
6. Proper references (model physics & experiments)

License agreement

1. NEEDS-modified Compact Model Council (CMC) standard license for compact models
2. https://nanohub.org/groups/needs/users_developers
3. For MVS model, the copyright is owned by MIT.
4. All package components are under the license agreement.
5. For your model, you may have to check with your institute if the copyright can/must be owned by model developers, i.e. you.



III. GOOD PRACTICES FOR WRITING VERILOG-A MODELS

Good practices for writing models in Verilog-A

Good Verilog-A practices

- I — Look and feel
- II — Variables
- III — Built-in functions
- IV — Performance limits & debugging

Look and feel; debugging easier

- **Write code legibly**
  - Indentation
  - Align code at (<+ or =)
  - Meaningful variable names

- **Partition code logically**
  - Initialization
  - Static quantities
  - Dynamic quantities
  - Noise

Makes debugging the code a lot easier

NEEDS

MTL

Example from MVS 1.0.0

```
analog begin

    //Voltage definitions
    Vgsraw                  =       type * ( V(g) - V(si) );
    Vgdraw                  =       type * ( V(g) - V(di) );
    if (Vgsraw >= Vgdraw) begin
            Vds             =       type * ( V(d) - V(s) );
            Vgs             =       type * ( V(g) - V(s) );
            Vbs             =       type * ( V(b) - V(s) );
            Vdsi            =       type * ( V(di) - V(si) );
            Vgsi            =       Vgsraw;

            Vbsi            =       type * ( V(b) - V(si) );
            dir             =       1;
    end
```

1. If-else statements aligned
2. Voltages are named appropriately/meaningful
3. Code aligned at =
4. Block comment added

Example from MVS 1.0.0

```
`include "constants.vams"
`include "disciplines.vams"

module mvs(d, g, s, b);
inout d, g, s, b;
electrical d, g, s, b;
electrical di, si;

// Original VS parameters
parameter real          version          = 1.00;
                //       MVS model version = 1.0.0
- - - - - - - - - - - - - - - - - - - - - - - - -

 //Total drain current

 Id                    =       Qinv_corr * vx0 * Fsat * W;

- - - - - - - - - - - - - - - - - - - - - - - - -
//Partitioned charge
Qs                     =       -W * ( Qinvs + Qsov + Qsif );
            //      s-terminal charge
Qd                     =       -W * ( Qinvd + Qdov + Qdif );
            //      d-terminal charge
```

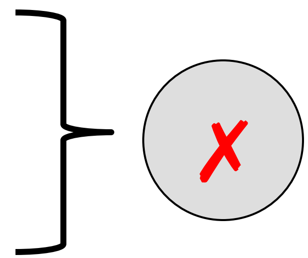
1. Initialization

2. Static quantities

3. Dynamic quantities

Variables in Verilog-A

1. Avoid ***unused variables*** in the code.
2. Avoid ***superfluous*** assignments.
3. Be careful of "***memory states***".



4. Assign ***parameter range***.
5. Verilog-A is ***case-sensitive***. Be aware whether or not your simulator is case sensitive (provide variable aliases).
6. Make sure variable names are ***identical*** in MATLAB script and Verilog-A code.

Superfluous assignments

Consider:

```
(1)   x = V(a,b)/R;
(2)   if (type == 1)
(3)         x = V(a,b)/R1;
(4)   else
(5)         x = V(b,a)/R2;
```

→ Superfluous

Diagnostic message from compiler:

```
Warning: Assignment to 'x' may be superfluous.
      [ filename.va, line 1 ]
```

Parameter range – example from MVS 1.0.0

```
parameter real          Rs0              = 100                    from (0:inf);
                    //      Access resistance on s-terminal [Ohms-micron]
parameter real          Rd0              = 100                    from (0:inf);
                    //      Access resistance on d-terminal [Ohms-micron]

                    //      Generally, Rs0 = Rd0 for symmetric source and drain
parameter real          Cif              = 1e-12                  from [0:inf);
                    //      Inner fringing S or D capacitance [F/cm]
parameter real          Cof              = 2e-13                  from [0:inf);
                    //      Outer fringing S or D capacitance [F/cm]
```

Avoids ***garbage in garbage out***.

Memory states

1. Also known as ***hidden states***.
2. Variables are initialized to zero on first call to module.
3. Simulator will retain the value of the previous iteration if the variable is not assigned before it is used.
4. Memory states cause ***unexpected behavior***.
5. These states are not typically identified in DC/TRAN simulations.

Example of a memory state in MVS 1.0.0 – 1/2

```
        if (eta0 <= `LARGE_VALUE) begin
                psis    =       phib + phit * ( 1.0 + ln( ln( 1.0 + `SMALL_V
ALUE + exp( eta0 ))));
        end
        else begin
                psis    =       phib + phit * ( 1.0 + ln( eta0 ));
        end
```

The variable **psis** must always be assigned a value.

Example of a memory state in MVS 1.0.0 – 2/2

Simulation error due to hidden state in MVS 1.0.0 (fixed in 1.0.1)
Discovered through periodic steady state (PSS) analysis

```
Error found by spectre during periodic steady state analysis
`pss1'.
ERROR (SPCRTRF-15177): PSS analysis doesn't support
behavioral module components with hidden states found in
component
'daa_mosfet'. Skipped.

mvs_si.va, declared in line 64: Hidden state variable: psis

Analysis `pss1' was terminated prematurely due to an error.
```

NEEDS

MTL

Built-in functions

1. Check ***compatibility*** of built-in functions in Verilog-A with various versions of simulators.
2. Be careful of ***derivatives*** (`$abs(x)`, `$ddx(sqrt(x))`) around `x=0`.
3. Watch for ***expensive*** functions (`$exp(), $pow()`)
4. Avoid language constructs not required (or desired) for compact modeling (Harmonic balance, Shooting, Envelope).
5. Avoid Verilog-A block level modeling features (transition, slew, last_crossing, absdelay)

Example from MVS 1.0.0

- Function `$exp()` versus `$limexp()`
  - `$limexp()` provides better convergence than `$exp()` to model semiconductor junctions although at the cost of extra memory.
  - Compatibility with various versions of SPECTRE must be tested.
  - `$limexp()` worked with SPECTRE version **10.1.1.374.isr21** but failed to run with SPECTRE version **5.10.41.121508**.
  - Current implementation of MVS 1.0.0 uses `$exp()` everywhere.

Example from MVS 1.0.0

Explicitly linearize $exp() above a break-point

```
//Charge at VS in saturation (Qinv)
if (eta  <= `LARGE_VALUE) begin
        Qinv_corr        =        Qref * ln( 1.0 + exp(eta) );
end
else begin
        Qinv_corr        =        Qref * eta;
end
```

Recommended practice

Example from MVS 1.0.0

Evaluating function **$ln()**

psis = (1.0 + ln(ln(1.0 +exp(eta0))));

eta0 → large negative, exp(eta0) = 0 → ln(0) can't be evaluated

Adding a small correction `SMALL_VALUE fixed the problem

psis = (1.0 + ln(ln(1.0 +**`SMALL_VALUE**+ exp(eta0))));

Defined as 1e-10

Performance limits – 1/3

Extra state variables impact model efficiency

1. Be mindful of performance limiting assignments

   – Voltage contributions on LHS; voltage sources add an extra state variable for the branch current

     • `V(a,b) <+ ….`

   – Implementing a non-linear capacitance [f(I), g(V)] as

     • **I(a,b) <+ g(V(a,b));✔**

     • **V(a,b) <+ f(I(a,b)); ✗**

   – Implementing an inductor

     • **V(a,b) <+ L*ddt(I(a,b))**

Performance limits – 2/3

Extra state variables impact model efficiency

2. Branch ddt(.) lead to extra state variables
   - Do not place the function **ddt(.)** within conditionals
   - Place the arguments to **ddt(.)** within conditionals
3. Formulate contributions as currents
   - **I(a,b) <+ ….**
4. Consider alternate, simplified expressions
5. Only truly voltage-controlled elements must be implemented with voltage contributions.

NEEDS

MTL

Performance limits – 3/3

Collapse nodes to improve efficiency

- Collapse nodes when possible

```
if (Ra > 0.0)
          I(bi,si) <+ V(bi,si)/Ra;
 else
          V(bi,si) <+ 0;
```

- What happens when **Ra** is too small ?

```
if (Ra >`SMALL_Ra)
   I(p,m)  <+ V(p,m)/Ra;
 else
   V(p,m)  <+ I(p,m)*Ra;
```

Does not work when:

a. Ra varies dynamically with bias
b. Some model interfaces for some simulators
c. Other instances ?

Massachusetts Institute of Technology

shaloo@mit.edu

Debugging

- Watch for **compile time warnings**.
- Developer must provide floating point exceptions, overflows, underflows etc.
- Get your code checked by peers and experts.
- Use source-code control and set up regression tests to test out each new model feature.

NEEDS

MTL

Following Verilog-A resources are extremely helpful:

[1] http://www.mos-ak.org/baltimore/talks/11_Mierzwinski_MOS-AK_Baltimore.pdf

[2] www.**mos-ak**.org/sanfrancisco/.../01_McAndrew_**MOS-AK**_SF08.ppt

[3] www.mos-ak.org/montreux/papers/06_Coram_MOS-AK06.ppt

[4] G. Coram, “How to (and how not not) write a compact model in Verilog-A”, BMAS 2004.

[5] Tianshi Wang; Jaijeet Roychowdhury (2013), "Guidelines for Writing NEEDS-certified Verilog-A Compact Models," https://nanohub.org/resources/18621

IV. UPDATES AND OPEN ISSUES

Updates in MVS 1.0.1

- **Verilog-A related**
  - Indentation and alignment fixed
  - Surface potential ***psis*** initialized properly
  - Unused parameters ***Qx, Qy, S, Vsatq, dibl, Vgd, Vgdi, qe*** are removed
- **MATLAB-related**
  - Model file name changed to match with the module name
  - **UNCOMMENT** flag added in files: model_exercise.m, extract_main.m, optimize_transfer.m, optimize_output.m
- **General**
  - Parameter ***tipe*** changed to ***type***
  - External parameter ***phit*** has been eliminated. Instead junction temperature ***Tjun*** has been added.
  - **LICENSE.txt** file added.

NEEDS

Open issues

This is what we hope to address in a future release

32 nm

```
%% UNCOMMENT following 5 lines for 32-nm node
IdVd=abs(dlmread('idvd_Intel_32_nFET_09.txt')); % first column is Vd,
second col. is Vg and 3rd col is Id.
Vymin = 0.05;
Vymax=1;
Vystep=0.05;
Vypre=Vymin:0.05:Vymax;
```

45 nm

```
%% UNCOMMENT following 5 lines lines for 45-nm node
% IdVd=abs(dlmread('idvd_Intel_45_nFET_12.txt')); % first column is Vd,
second col. is Vg and 3rd col is Id.
% Vymin = 0.1;
% Vymax=1;
% Vystep=0.1;
% Vypre=Vymin:0.1:Vymax;
```

1. Automation in parameter extractor and model exerciser implemented in MATLAB needs to be improved.

NEEDS

Open issues

This is what we hope to address in a future release

2. Couple MATLAB optimization with Verilog-A model. Write a script that calls Verilog-A routine and uses that instead of MATLAB model file.

3. Charge/dynamic model may be improved to yield continuous derivatives of certain capacitances with voltage **(update of version #).**

Summary

- Checklist/Guidelines wiki

https://nanohub.org/groups/needs/wiki/SpecificInstructionsforNEEDSCompatibleCompactModels

&

This presentation

- Verilog-A resources available online (some through NEEDS)