# BSIM3_Matlab: Reorganizing BSIM3 Model to Explore Circuit Simulation Techniques

Shih-Hung Weng, Hao Zhuang and Chung-Kuan Cheng
Computer Science and Engineering Department
University of California, San Diego
Email: hao.zhuang@cs.ucsd.edu, ckcheng@ucsd.edu
August 20, 2013

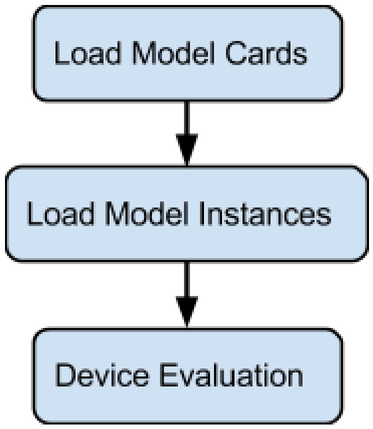
## 1. Introduction

BSIM3 model was originally developed by UC Berkeley, and later adopted widely into different SPICE simulators. e.g., spice3, hspice, ngspice, etc. The tailored BSIM3 device model for the SPICE simulation limits the adoption of different advanced numerical integration approaches. Current BSIM3 model is specific for Modified Nodal Analysis while more generalized state equations is lack of support. Without properly support, circuit simulation community is unable to develop more advanced numerical engines under a practical compact model.

In order to improve the support of BSIM3 model, we develop Matlab BSIM3 model, which is modified from the original BSIM3 model from ngspice package. Matlab BSIM3 model is able to construct general state equations and has interface between Matlab. Therefore, other different numerical approaches can adopt more realistic model, and developers can prototype quickly under the Matlab environment.

## 2. Basic Working Principle in BSIM3 Model

The working flow of BSIM3 model during the transient analysis is shown as the following figure. There three major stages, Model Cards Loading, Model Instances Loading and Device Evaluation.



- **Model Cards Loading**: This stage will load different model cards. Each model card includes different settings of each BSIM3 model, such as low Vth or high Vth process. Each parameter in the model card will be set in the structure ***ModelCard*** by function ***BSIM3mParam***. The rest of unassigned required parameters in structure ***ModelCard*** will be set as default values by function ***BSIM3ModelSetup***. Detailed parameters in BSIM3 model are listed in ***param_init.c***. Users can call Matlab function ***LoadBSIM3Model*** to load a specific model file.

- **Model Instances Loading**: This stage will load and create all MOSFET instance using BSIM3 model. The device related parameters, such as channel length, width, will be set by function ***BSIM3parm***. Then, the required sparse matrices for MOSFEST are also created. Unassigned device related parameters are set by function ***BSIM3InstanceSetup***. The device operation temperature is set by function ***BSIM3temp***. User can call Matlab function ***LoadBSIM3Instance*** to load MOSFET instances.

- **Device Evaluation**: This stage will evaluate the charges and currents of MOSFET device instances according to the previous states (for charge conservation) and terminal voltages. The device instances will be evaluated by function ***BSIM3load***. Since device evaluation needs to consider either for DC analysis or for transient analysis, there are different options in ***BSIM3load***. The mode **MODETRANOP** is for DC analysis and mode **MODETRAN** is for transient analysis. Besides these two options, different options are used when performing during Newton’s iteration. In DC analysis, the first iteration of Newton’s iteration needs another option **MODEINITJCT**, and the rest of iterations needs option **MODEINITFIX**, and the last iteration needs option **MODEINITFLOAT** for the following transient analysis. In transient analysis, the very first iteration (not of Newton’s iteration, but first iteration of current time stepping) requires option **MODEINITTRAN**, and the first iterations of each Newton’s iteration of current time stepping needs option **MODEINITFLOAT** and **MODEFIRSTITER**, and the rest of iterations needs option **MODEINITFLOAT**. The returned parameters from function ***BSIM3load*** include right hand-side vector (Ax=b) for backward Euler method and right hand-side vector (equivalent nonlinear currents) for MEXP method. Note that right hand-side vector for MEXP could be modified according to the nonlinear formulation. Please modify ***BSIM3load*** function directly. Equivalent capacitance associated with matrix C can be determined by if dividing by time step ***ag0***. The following table summarizes the options for different iterations and analyses.

| | DC Analysis | Transient Analysis |
|---|---|---|
| 1st iter. of current step | n/a | MODEINITTRAN\|MODETRAN |
| 1st iter. of Newton method | MODEINITJCT\|MODETRANOP | MODEINITFLOAT\|MODETRAN\|MODEFISTITER |
| rest iter. of Newton method | MODEINITFIX\|MODETRANOP | MODEINITFLOAT\|MODETRAN |
| last iter. of Newton method | MODEINITFLOAT\|MODETRANOP | MODEINITFLOAT\|MODETRAN |

## 3. BSIM3 Structures

- **BSIM3model**: This structure stores BSIM3 model dependent parameters, such as threshold voltage. Those parameters will be given in user provided model card.

- **BSIM3instance**: This structure stores BSIM3 instance parameters, such as length, width of a transistor. This structure also holds pointers of sparse matrices as well as the pointers of corresponding elements in the sparse matrices. Those elements are for efficient stamping after device linearization.

- **ModelInstance**: Wrap structure for **BSIM3instance**.

- **ModelCard**: Wrap structure for **BSIM3model**.

## 4. BSIM3 Functions

- **BSIM3load**: Load and evaluate all BSIM3 instance. After evaluation (linearization), those equivalent capacitance and resistance will be filled into the sparse matrices.

- **BSIM3ModelSetup**: Set unassigned parameters in default values for device model.

- **BSIM3InstanceSetup**: Set unassigned parameters in default values for device instances, and also allocate memory and create pointers for sparse matrices elements.

- **BSIM3mParam**: Set value to a specific parameter of device model.

- **BSIM3Param**: Set value to a specific parameter of device instance.

- **BSIM3temp**: Set and calculate temperature-related parameters.

- **BSIM3checkModel**: Check validness of parameters.

- **NIintegrate**: Calculate charges of devices over a time step.

## 5. Usage of Matlab MEX Functions

- **LoadBSIM3Model**: Load BSIM3 model from a given path.
    - Input
        - String of file path
    - Output
        - pointer to BSIM3model structure (in scalar variable of Matlab)
    - Format
        - [*ptr_model_card*] = ***LoadBSIM3Model***(*str_path*)

- **LoadBSIM3Instance**: Load BSIM3 instances in SMORES format.
    - Inputs:
        - List of pMOSFET instances in SMORES format
        - List of nMOSFET instances in SMORES format
        - Pointer to BSIM3 model structure
        - Name of model for pMOSFET
        - Name of model for nMOSFET
        - Number of nodes
        - Temperature
    - Output:
        - Pointer to BSIM3 instance structure (0 means no BSIM3 devices)
    - Format:
        - [*ptr_to_instance*] = ***LoadBSIM3Instance***(*nMos, pMos, model, 'pModel', 'nModel', num_node, temp*)

- **ListModelName**: List names of models in the model file
    - Input:
        - Pointer to BSIM3 model structure
    - Output:
        - List of model names
    - Format:
        - [*list_of_names*] = ***ListModelName***(*ptr_to_model*)

- **BSIM3Eval**: Evaluate BSIM3 instances
    - Inputs:
        - Pointer to BSIM3 instance structure
        - Voltages of nodes (should include GND voltage)
        - Step size
        - Type of analysis (either in 'dc' or 'tran')
        - First time run (either 0 or 1). Both Type of analysis and First time run determine the evaluation options (See the table in Section 2).
        - Current iteration number. This is for moving between states. (See the table in Section 2).
    - Outputs:
        - Sparse matrix C
        - Sparse matrix G
        - Current vector b
        - (optional) Current vector b for MEXP
        - (optional) Extrinsic sparse matrix eC
    - Format:
        - [*C G f f_mexp eC*] = ***BSIM3Eval***(*ptr_to_instance,* [0; *x*], *step_size*, *mode*, *first*, *iter*);
        - Note: [0; *x*] is for GND voltage.

## 6. **Example**

- DC Analysis

```
for it = 1:max_iter
  % 0 is initial value for gnd; should always be 0
    % f includes Gnl*x_k + i(x_k), should calculate x_{k+1} = x_{k} - f(x_k)/f'(x_k)
    % instead of dx = J\F
    [spm_c spm_g f] = BSIM3Eval(instance, [0; voltage], step_size, 'dc', it, it);

    %Gl is matrix G of linear components
    T = Gl + spm_g;
    b = B*input;
    b(1:size(f,1)) = b(1:size(f,1)) + f;
    x = T\b;
    dx = x - voltage;

    %update with current solution
    voltage = x;
    if (norm(dx) < iter_err_tol)
        %last evaluation for starting transient analysis, set iter_num as -1
        [spm_c spm_g f f_mexp] = BSIM3Eval( instance, [0; voltage], step_size, 'dc', it, -1);
        T = Gl + spm_g;
        b =B*input;
        b(1:size(f,1)) = b(1:size(f,1)) + f;
        x = T\b;
        break;
    end
end
result = x;
```