

Tune and Verify Complex Model Library Using PathWave Model Builder and Model QA

Introduction

Advanced CMOS process technology is now driving IC designs to have smaller device scaling yet better performance. Smaller device size creates more unexpected effects and variation, which are not considered in older technologies. Consequently, simulation model libraries, required to describe all the effects and variations, are becoming huge, complicated, and difficult to read.

Not only the size and complexity of SPICE model increases, foundries also implement other types of codes to describe the special device electronic behaviors, such as encrypted SPICE model and Dynamic-link library (DLL). These types of model libraries work as a black box that users cannot view or modify the equations inside, and the model extraction tools also cannot read the contents. Consequently, this makes it even harder for modeling engineers to tune these model libraries parameters.

This application note provides information to tune complex model library using the latest PathWave Model Builder (MBP) 2020, and then verify it using PathWave Model QA (MQA).

Load and Tune Model Parameters Using PathWave Model Builder (MBP)

The modeling engineers in design houses need to tune the foundry's SPICE model library to get a better fitting accuracy in the extended geometries, temperatures, bias, or frequency range where their designs are applied. However, foundry models have a very complex hierarchical structure and some of them may also involve other script or even dynamic-link library (dll).

From the design houses users' perspective, they want to tune just a few model parameters without touching the existing structure of the model library.

In PathWave Model Builder (MBP) 2020, we have extended MBPs support to only aging TSMC Modeling Interface (TMI) model to general DC and CV models support on broader model structures. This is achieved by the introduction of the new Lib Tweaking feature. Using Lib Tweaking, users can easily select model parameters, perform tuning and optimization in GUI environment as conventional models. After the tweaking, the new library can be saved without damaging its original structure.

This application note provides information to tune complex model library using PathWave Model Builder (MBP), and then verify it using PathWave Model QA (MQA).

Prerequisites

Some of the SPICE model libraries are using the dynamic-link libraries (for example, TMI library), which could not be supported by the MBP internal engine. PathWave Model Builder (MBP) provides an option to tune model parameters using external SPICE simulators. Consequently, MBP will require some prerequisites to make sure the simulation works properly.

1. All compiled shared library files should be kept in the same structure as the foundry released. For example, in the TMI model, the parent folder contains the shared libraries of different platforms, such as RH, SUN, WIN, etc. Users need to keep the directory structure and place all configuration files for the TMI model at the same level of the parent folder.
2. PathWave Model Builder (MBP) invokes external simulators (Synopsys HSPICE or Cadence Spectre) for model simulation. Ensure HSPICE or Spectre has been installed and linked to MBP properly prior to the simulation.

Note: For Spectre, only Linux is supported for now.

3. Prepare the usage file or top file to set options, include libraries and declare all parameters you want to work on.

Load model, data and tune parameters in MBP

Choose model type

From the Main Menu, click **Model > Select Model** and select DC as the **Project Type**. Then, choose the core model (bsimcmg in this example) from the upper **Model Selection** section and hspice or spectre in the lower section.

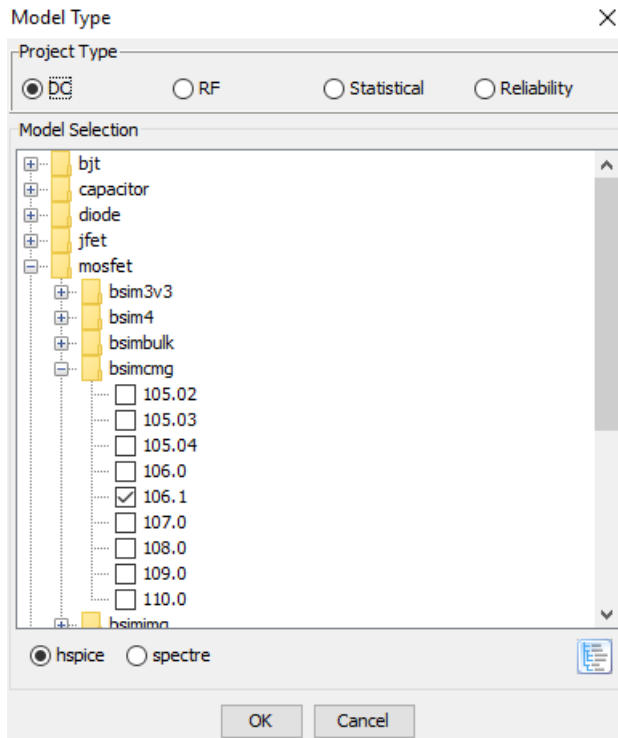


Figure 1. Select Model Type Window

Load model

From the Main menu, **select File > Model > Load** to load the usage file. If the usage file includes multiple model cards, MBP displays a window showing all the models inside the library including different device types, corners, polar types, subckt and compact, and so on.

Select the core model you want to work on.

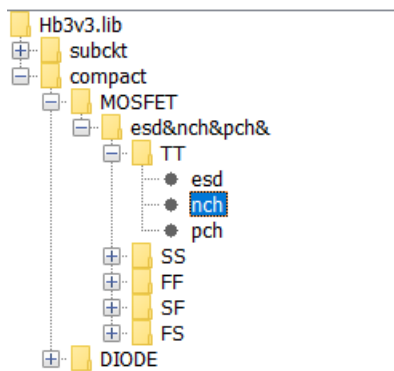


Figure 2. Select core model to work

Enable Lib tweaking

From the Main menu, click **Menu > Simulation > Lib Tweaking**. The Lib Tweaking function is enabled, and MBP will automatically switch to external simulator then.

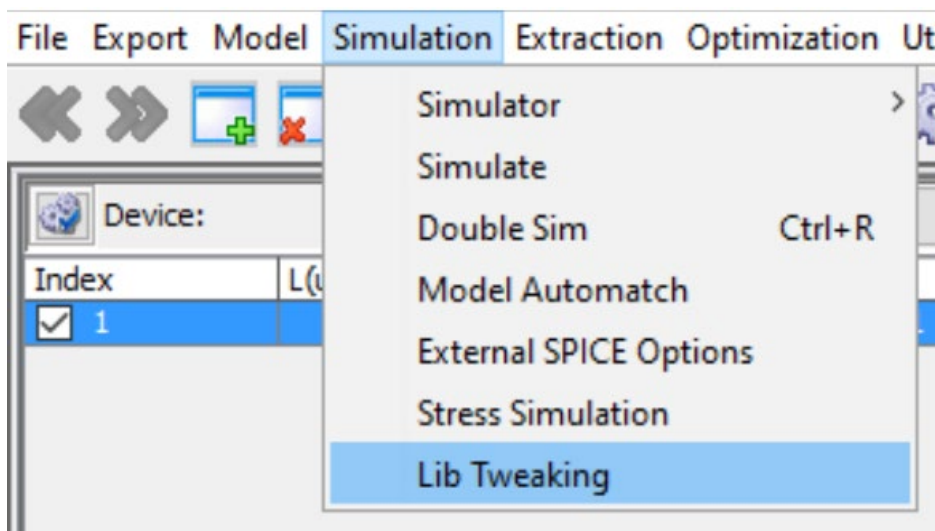


Figure 3. Lib Tweaking Option

Load data

To load the data file, click **File > Data > Load** from the Main menu. MBP simulates the first plot when the data is loaded.

Click on the devices or pages to select the target plot, and MBP invokes external SPICE each time when changing plot.

Select parameters to tweak

When the data and model is loaded, and Lib Tweaking option is enabled, user can select parameters and tune them.

All functions in MBP main GUI are also applicable if the Lib Tweaking feature is enabled.

Note that when using the external simulator, do not use the mouse wheel to tune parameters. It will trigger multiple simulation times and slow down the speed.

Save model

To save the model, click File > Model > Save. MBP will save the changed model parameters value back to the model card and does not change the original model hierarchy or format.

How Lib Tweaking Works with Complex Model Library

When the user loads a model into MBP, MBP will first parse the model structure, flatten all the nested structure, and send model parameters to MBP internal SPICE engine to simulate. However, with the increasing model library complexity, The MBP internal SPICE engine did not support some syntax in the library, such as encrypted model, TMI model, or some other type of compiled libraries. These models require to call external simulators.

By enabling the **Lib tweaking** option, MBP will not parse and flatten the entire model lib when the user loads the library, but simply generate netlist using all the original structure in model library. Then, call external simulator and return the result to display the plot in MBP GUI.

When the user changes the parameter value, MBP will find and change the selected parameter value in the original library but does not change the model structure or format.

This can be demonstrated if we compare the netlist generated by MBP before and after enabling the **Lib tweaking** feature. We take a simple demo model lib Hb3v3.lib as an example.

```
e36 42 43 vcvs 20 0 1.0
v37 44 43 dc=-1.5
v38 45 43 dc=0.05
v39 43 0 dc=0.0
v41 45 41 DC=0.0
m40 41 42 43 44 nmos w=1.0E-6 l=9.999999999999999E-6 dtemp=0.0
.dc v10 poi 34 0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5
.probe i(v17)
.probe i(v23)
.probe i(v29)
.probe i(v35)
.probe i(v41)

.model nmos nmos
+level = 54                version = 4.5                binunit = 1
+paramchk = 1              mobmod = 0                    capmod = 2
+rdsmode = 0               igcmode = 0                   igbmod = 0
+rbodysmod = 0             trngsmode = 0                 acngsmode = 0
+fnoimode = 1              tnoimode = 0                   diomode = 1
+tempmode = 0              permode = 1                    geomode = 1
```

Figure 4. Netlist without enabling 'Lib tweaking'

```

e36 42 43 vcvs 20 0 1.0
v37 44 43 dc=-1.5
v38 45 43 dc=0.05
v39 43 0 dc=0.0
v41 45 41 DC=0.0
x40 41 42 43 44 mqa_macro W=1.0E-6 L=9.999999999999999E-6 T=25.0
.dc v10 poi 34 0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 (
.probe i(v17)
.probe i(v23)
.probe i(v29)
.probe i(v35)
.probe i(v41)
.lib 'C:\Keysight\MBP_2020\demo\Library\Hb3v3.lib' TT

```

Figure 5. Netlist with enabled 'Lib tweaking'

Comparing the highlighted part in Figure 4 and Figure 5, the netlist in Figure 4 inserts all parameters in the model card, and in Figure 5 it uses the statement .lib to include related model card.

For the simple model library as the example Hb3v3, the simulation result will be the same when you enable or disable the **Lib tweaking** option'. However, for the advanced model library with the contents that MBP does not recognized, the netlist with 'Lib tweaking' checked will always be the correct one.

Verify Complex Model Library Using PathWave Model QA (MQA)

PathWave Model QA (MQA) is designed to verify the foundry model library. The objective of MQA is to automatically generate netlist, call external simulator, and return the results.

From the Main menu, **click Options > General > Preferences tab > Flatten model library** to **No**. When MQA generates the netlist, MQA maintains the hierarchical structure and original format of the entire model library. If we review the MQA generated netlist, it looks very similar to Figure 5 above. The top library is included using the statement `.lib` and other supply voltage, circuit, user options are auto generated by MQA. User can then use the complex model library the same with conventional model libraries and go through the verification flow with user-defined rule file.

```
.option brief=1 post=2
.param
+qa_swp_1 = '0'

.lib 'Hb3v3.lib' TT
.TEMP 25

.options DCCAP post_version=2001 post=2 probe brief=1
v20 53 54 dc=1
v21 52 54 dc=1
v22 55 54 dc=qa_swp_1
v23 52 56 dc=0
m51 56 53 54 55 nch l=1e-06 w=1e-06
```

Figure 6. Netlist generated by MQA

Conclusion

The foundry SPICE model library is becoming complicated to better describe all the complex device behaviors and variations in the advanced technology. On the other hand, design houses want to easily add their own sub-circuits or tune parameters on top of the foundry model without touching the inside structure.

With the new capability of dealing with different types of model structure, PathWave Model Builder (MBP) and PathWave Model QA (MQA) offers the total solution to address complex model library tweaking and verification flow. It could greatly improve the usability and productivity while working with complex model library.