Advanced Design System 2011.01

Feburary 2011
Model Composer

Acknowledgments
Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Mentor products and processes are registered trademarks of Mentor Graphics Corporation. * Calibre is a trademark of Mentor Graphics Corporation in the US and other countries. "Microsoft®, Windows®, MS Windows®, Windows NT®, Windows 2000® and Windows Internet Explorer® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Oracle and Java and registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC. FLEXlm is a trademark of Globetrotter Software, Incorporated. Layout Boolean Engine by Klaas Holwerda, v1.7 http://www.xs4all.nl/~kholwerd/bool.html . FreeType Project, Copyright (c) 1996-1999 by David Turner, Robert Wilhelm, and Werner Lemberg. QuestAgent search engine (c) 2000-2002, JObjects. Motif is a trademark of the Open Software Foundation. Netscape is a trademark of Netscape Communications Corporation. Netscape Portable Runtime (NSPR), Copyright (c) 1998-2003 The Mozilla Organization. A copy of the Mozilla Public License is at http://www.mozilla.org/MPL/ . FFTW, The Fastest Fourier Transform in the West, Copyright (c) 1997-1999 Massachusetts Institute of Technology. All rights reserved.

The following third-party libraries are used by the NlogN Momentum solver:

"This program includes Metis 4.0, Copyright © 1998, Regents of the University of Minnesota", http://www.cs.umn.edu/~metis , METIS was written by George Karypis (karypis@cs.umn.edu).

Intel@ Math Kernel Library, http://www.intel.com/software/products/mkl

SuperLU_MT version 2.0 - Copyright © 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). All rights reserved. SuperLU Disclaimer: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

About Model Composer . . . 7
Requirements . . . 7
Model Generation Techniques . . . 7
Example . . . 9
Creating Models . . . 12
Model Composer Components . . . 22
B . . . 22
C . . . 22
G . . . 22
O . . . 22
R . . . 22
S . . . 22
T . . . 22
bend (Arbitrary Angle Bend) . . . 22
bend_m (Mitered Arbitrary Angle Bend) . . . 23
bend_r (Rounded Arbitrary Angle Bend) . . . 24
corner (90-degree Corner) . . . 24
corner_a (90-degree Asymmetric Corner) . . . 25
corner_am (90-degree, Asymmetric, 50%-miter Corner) . . . 25
corner_m (90-degree Mitered Corner) . . . 26
corner_r (90-degree Rounded Corner) . . . 27
cross (Cross Junction) . . . 27
cross_s (Symmetric Cross Junction) . . . 28
gap (Symmetric Gap) . . . 29
gap_a (Asymmetric Gap) . . . 29
open (Open-end Effect) . . . 30
rstub (Radial Stub) . . . 30
slit (Symmetric Slit) . . . 31
slit_a (Asymmetric, One-sided Slit) . . . 31
step (Step in Width) . . . 32
step_a (Asymmetric, One-sided Step in Width) . . . 33
taper (Tapered Step in Width) . . . 33
taper_a (Asymmetric, One-sided, Tapered Step in Width) . . . 34
tee (Tee Junction) . . . 34
tee_s (Symmetric Tee Junction) . . . 35
Troubleshooting a Library . . . 35
General Guidelines . . . 36
Using Models . . . 37

About Model Composer

Model Composer enables you to create multidimensional, parametrized, and passive planar components. The unique, patented modeling method is EM based, thus providing EM accuracy and generality at traditional circuit simulation speed.

Model Composer enables you to create many standard interconnect components such as opens, stubs, bends, and tees on custom substrates. These components/models are fully compatible within ADS and include both a schematic and layout representation, as well as the electrical model. The models are created in libraries, that are logical groupings or sets of components. You can use these models as you would other ADS microstrip component. They are compatible with all simulators, including optimization and tuning.

> **Note**
> *Advanced Model Composer* allows users to create multi-dimensional, parametrized, passive planar models for arbitrary shaped and user-defined parametrized layout components on standard and custom substrates. The Advanced Model Composer functionality is integrated into the ADS Layout under the Momentum menu: Momentum > Component > Advanced Model Composer. For more information on this functionality, refer to *Using Advanced Model Composer* (em).

The component definition and generation process is simple. The Model Composer Wizard leads you through each step, and you can save your work at any time, exit the tool, then return to complete your work later.

Requirements

Model Composer is a part of the ADS. No license is required to run Model Composer.

Model Generation Techniques

The Model Composer provides a method to build multidimensional parameterized analytical models for passive planar components. This method produces analytical models that can be used by all ADS circuit simulators, and the models are highly accurate. The model generation is based on EM simulation techniques, providing EM accuracy and generality at traditional circuit simulation speed.

The model generation technique is referred to as *Multidimensional Adaptive Parameter Sampling* (MAPS). It selects a minimum number of EM simulations, and builds a global analytical fitting model for the scattering parameters of general planar structures as a function of the geometrical parameters and of the frequency, with a predefined accuracy. Data points are selected efficiently and model complexity is automatically adapted. The algorithm consists of an adaptive modeling loop and an adaptive sample selection loop. Descriptions of each follow. An example is also presented to illustrate the technique.

Adaptive Model Building Algorithm

The scattering parameters *S* are represented by a weighted sum of multidimensional orthonormal polynomials (multinomials) P_m . The multinomials only depend on the multidimensional coordinates $\bar{x}$
in the parameter space *R* , while the weights C_m only depend on the frequency *f* :

$$S(f,\bar{x}) \approx M(f,\bar{x}) = \sum_{m=1}^{M} C_m(f) P_m(\bar{x})$$

The weights C_m are calculated by fitting equation (1) on a set of *D* data points $\{x_d, S(f,x_d)\}$
(with *d* = 1, ..., *D*). The number of multinomials in the sum is adaptively increased until the error function:

$$E(f,x) = |M(f,x) - S(f,x)|$$

is lower than a given threshold (which is function of the desired accuracy of the model) in all the data points. For numerical stability and efficiency reasons orthonormal multinomials are used, i.e. the multinomials $P_m(x)$
satisfy the condition:

$$\sum_{d=1}^{D} P_k(\bar{x}_d) P_l(\bar{x}_d) = \begin{cases} 1 & \text{for } (k = l) \\ 0 & \text{for } (k \neq l) \end{cases}$$

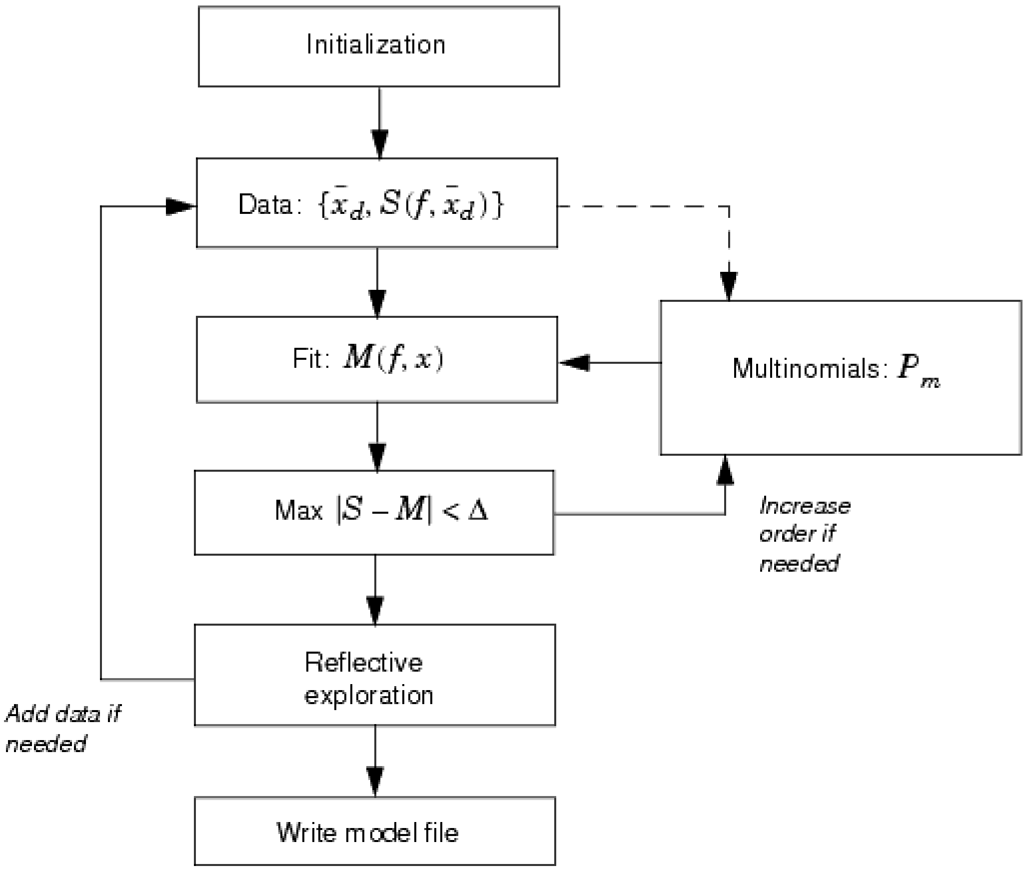
Adaptive Data Selecting Algorithm

The modeling process starts with an initial set of data points. New data points are selected adaptively in such a way that a predefined accuracy Δ for the models is guaranteed. The process of selecting data points and building models in an adaptive way is often called *reflective exploration* . Reflective exploration is useful when the process that provides the data is very costly, which is the case for full-wave electro-magnetic (EM) simulators. Reflective exploration requires *reflective functions* that are used to select a new data point. The reflective function used in the MAPS algorithm is the difference between two different models (different order M in equation *(1)*). A new data point is selected near the maximum of the reflective function. When the magnitude of the reflective function becomes smaller than Δ over the whole parameter space, no new data point is selected.

If one of the scattering parameters has a local minimum or maximum in the parameter space of interest, it is important to have at least one data point in the close vicinity of this extremum in order to get an accurate approximation. Therefore, if there is no data point close to a local maximum or minimum of $M(f,x)$, the local extremum is selected as a new data point. For resonant structures, the power loss has local maxima at the resonance frequencies. Again, to get an accurate approximation, a good knowledge of these local maxima is very important.

The scattering parameters of a linear, time-invariant, passive circuit satisfy certain physical conditions. If the model fails these physical conditions, it cannot accurately model the scattering parameters. The physical conditions act as additional reflective functions: if they are not satisfied, a new data point is chosen where the criteria are violated the most.

The complete flowchart of the algorithm is shown.



Example

The example presented here consists of two parts:

- The generation of a microstrip open stub and a symmetrical tee model
- Using these microstrip components to build a lowpass filter

Model Generation of Microstrip Components

The Model Composer was used to generate analytical models for an open stub (*open_1*) and for a symmetrical tee (*tee_s_1*), both on a 635 μm microstrip substrate, with ε_r =

10.0. Feedlines are connected to all ports (FEEDL=635 μm).

The open stub model (*open_1*) has one variable geometrical parameter, the width of the stub (*W*). There is only one relevant S-parameter, S_{11} .

The symmetrical tee model (*tee_s_1)* has two geometrical parameters: the width of the thru line (W_thru) and the width of the stub (W_stub). There are three relevant S-parameters, S_{11} , S_{12} and S_{13} . The ranges of the continuously varying geometrical parameters are in Parameter ranges for microstrips open stub and tee.

Parameter ranges for microstrips open stub and tee

component	variable	min	max
open_1	*W*	100 μm	1000 μm
	f	1 GHz	20 GHz
tee_s_1	*W_thru*	100 μm	1000 μm
	W_stub	300 μm	600 μm
	f	1 GHz	20 GHz

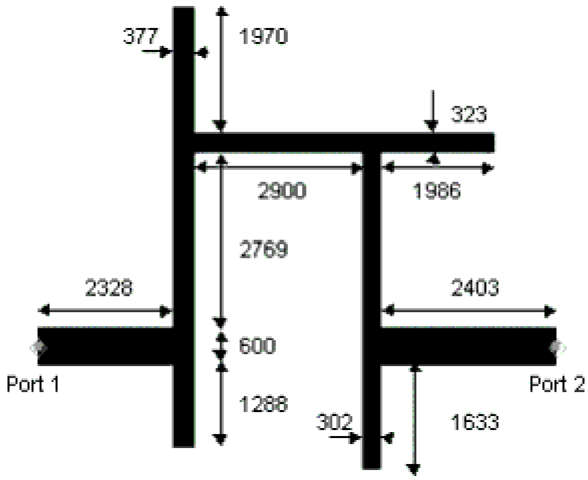
The Model Composer combines all S-parameters of multiple discrete parameter settings in one single global model file. The scattering parameters are generated using the Momentum simulator.

The desired accuracy for the open stub model was set to -55 dB (default accuracy). Building this model required 10 data points (adaptively selected). The accuracy of the model was checked in 71 points randomly chosen along the W-axis. The maximum deviation found between the Model Composer model and Momentum was -60.6 dB.

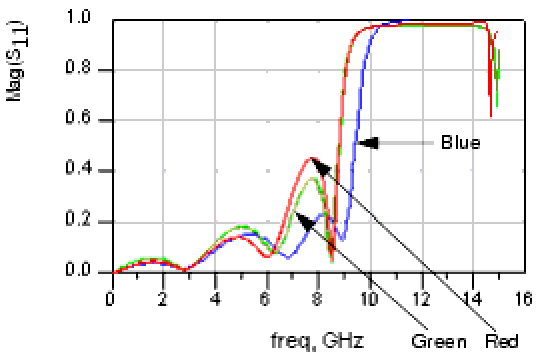
The desired accuracy for the tee model was set to -55 dB (default accuracy). Here there were 16 data points needed (adaptively selected) during model generation. The accuracy of the tee model was checked in 208 points randomly chosen in the parameter space. The maximum deviation found between the Model Composer model and Momentum was -54.3 dB.

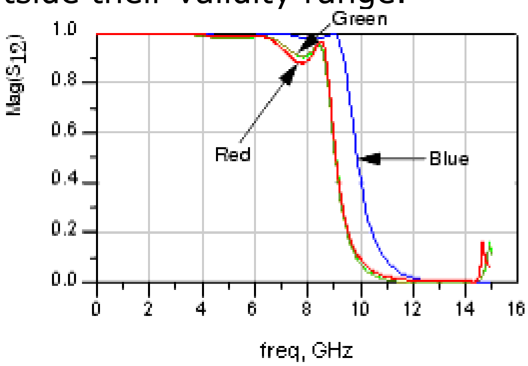
Library Usage to Design a Lowpass Filter

The adaptively generated models were then used to simulate a lowpass elliptic filter on a 635 μm microstrip substrate (ε_r = 10.0). The layout is shown here.



The graphs show the magnitude of S_{11} and S_{12} simulated with Momentum (red), with standard ADS analytical models (blue), and with the new EM-based Model Composer models for the open end (*open_1*) and the tee (*tee_s_1*) components (green).
The results using the multiple Model Composer models correspond very well to the global full-wave Momentum results, and yet the simulation using the Model Composer models took only a fraction of the time required for the full-wave Momentum simulation (due to the divide and conquer technique used).
On a 450 MHz PC, the full-wave simulation took 5037 seconds, while the simulation using the Model Composer models was virtually instantaneous. The results obtained with the "classic" analytical models of the circuit simulator differ significantly from the full-wave results because they were used outside their validity range.

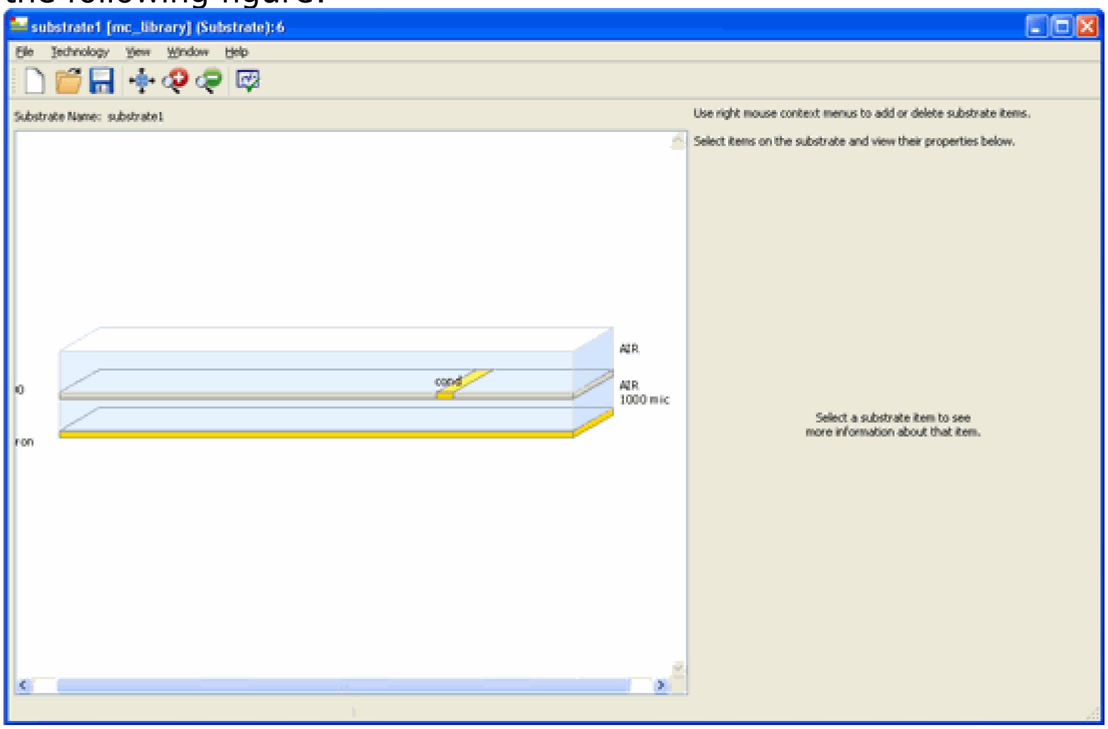




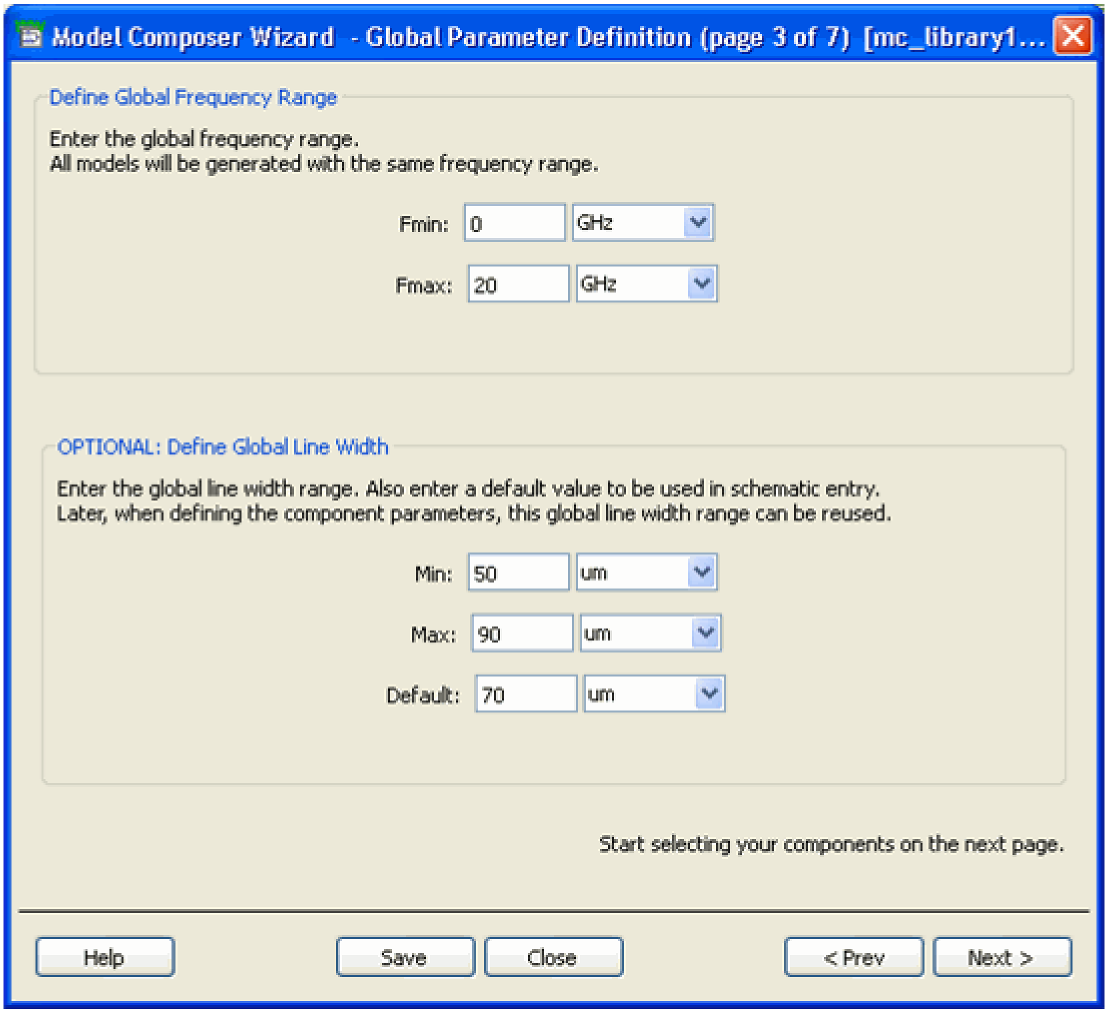
Creating Models

You can use the Model Composer wizard to create a library of passive components.

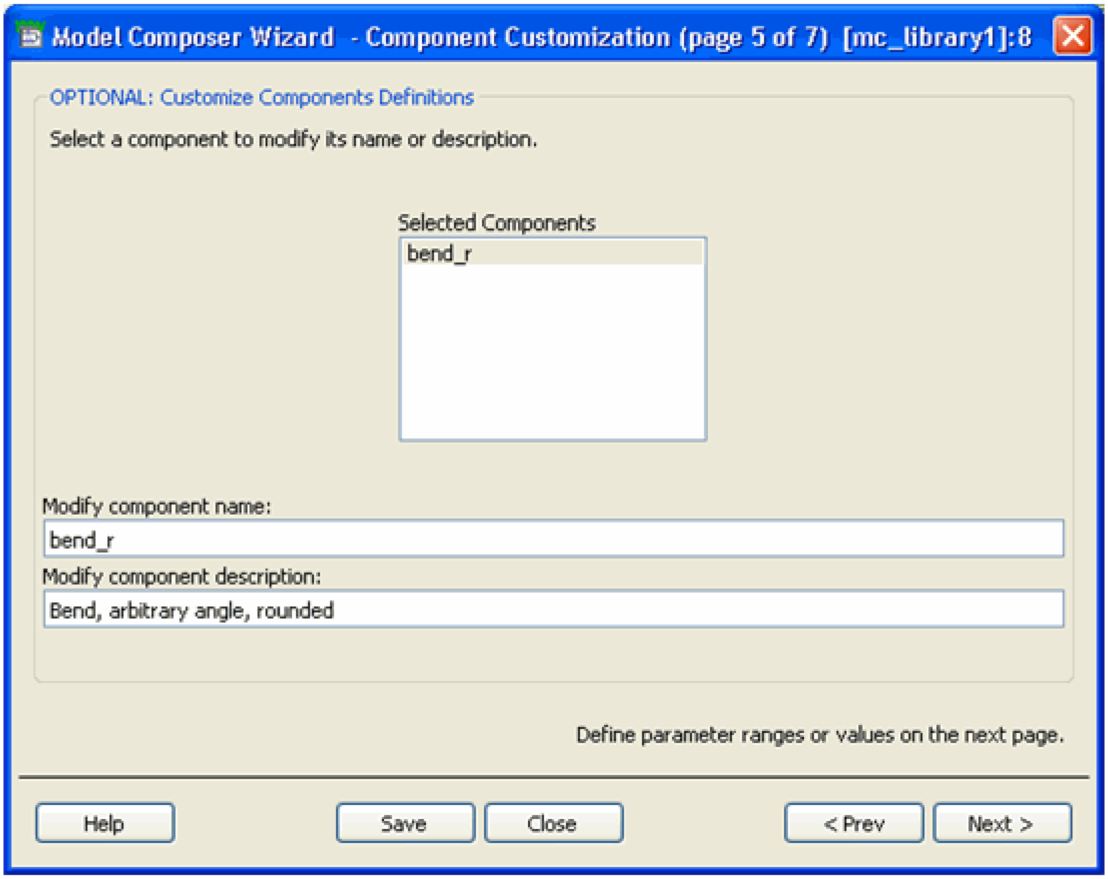
To define a collection of models:

1. Select **Tools** > **Model Composer** > **Create/Modify Library** in the Schematic window. The Model Composer Wizard is displayed.
2. Click **Next**.
3. Select the required option for creating a library. For more information, see Creating or Editing a Library.
4. Specify the library name.
5. Click **Next**. The **Substrate Definition** screen is displayed.
6. To define a substrate, click **Edit**. The **Substrate** window is displayed, as shown in the following figure:


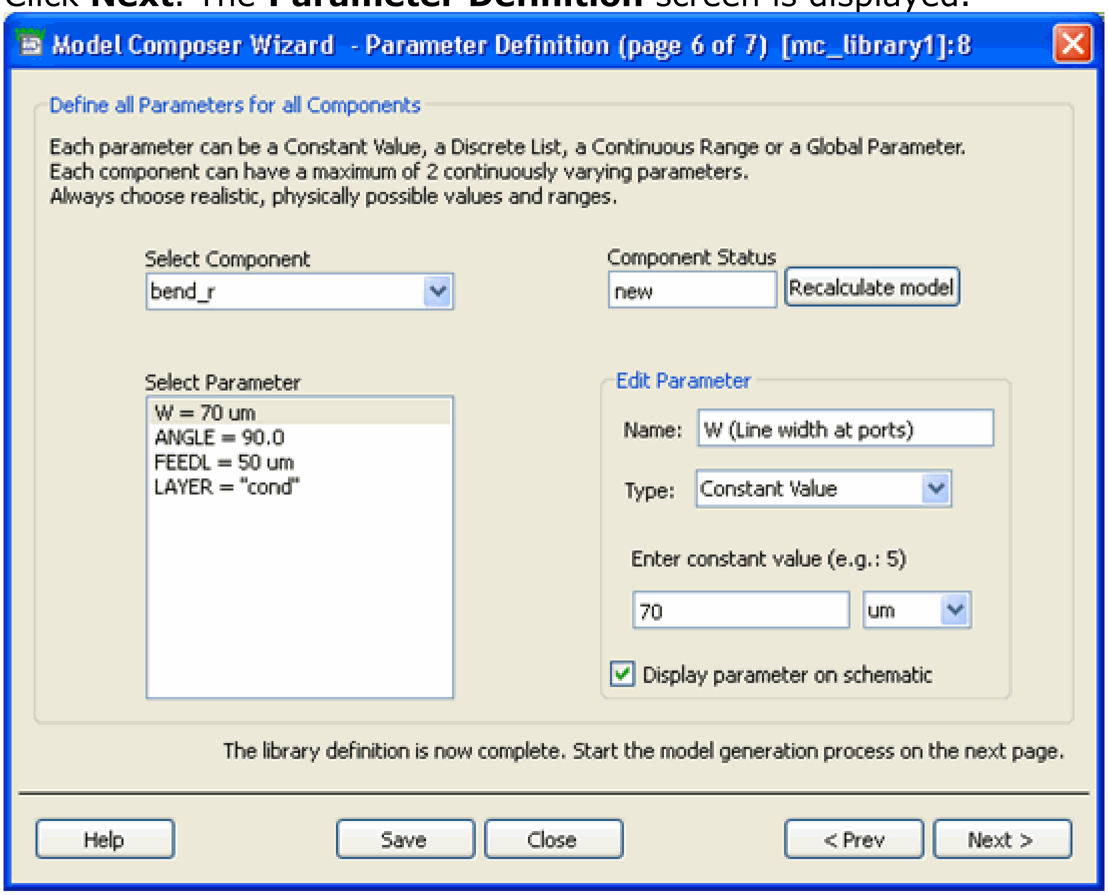
7. After specifying the substrate settings, close the Substrate dialog box.
8. Click **Refresh** in the Substrate Definition screen to update the selection list.
9. Click **Next**. The **Global Parameter Definition** screen is displayed, as shown in the following figure:



10. Specify the global parameters. For more information, see Setting the Frequency Range and Line Widths.
11. Click **Next**. The **Component Selection** screen is displayed.
12. Select the required components. For more information, see Selecting Components.
13. Click **Next**. The **Component Customization** screen is displayed.

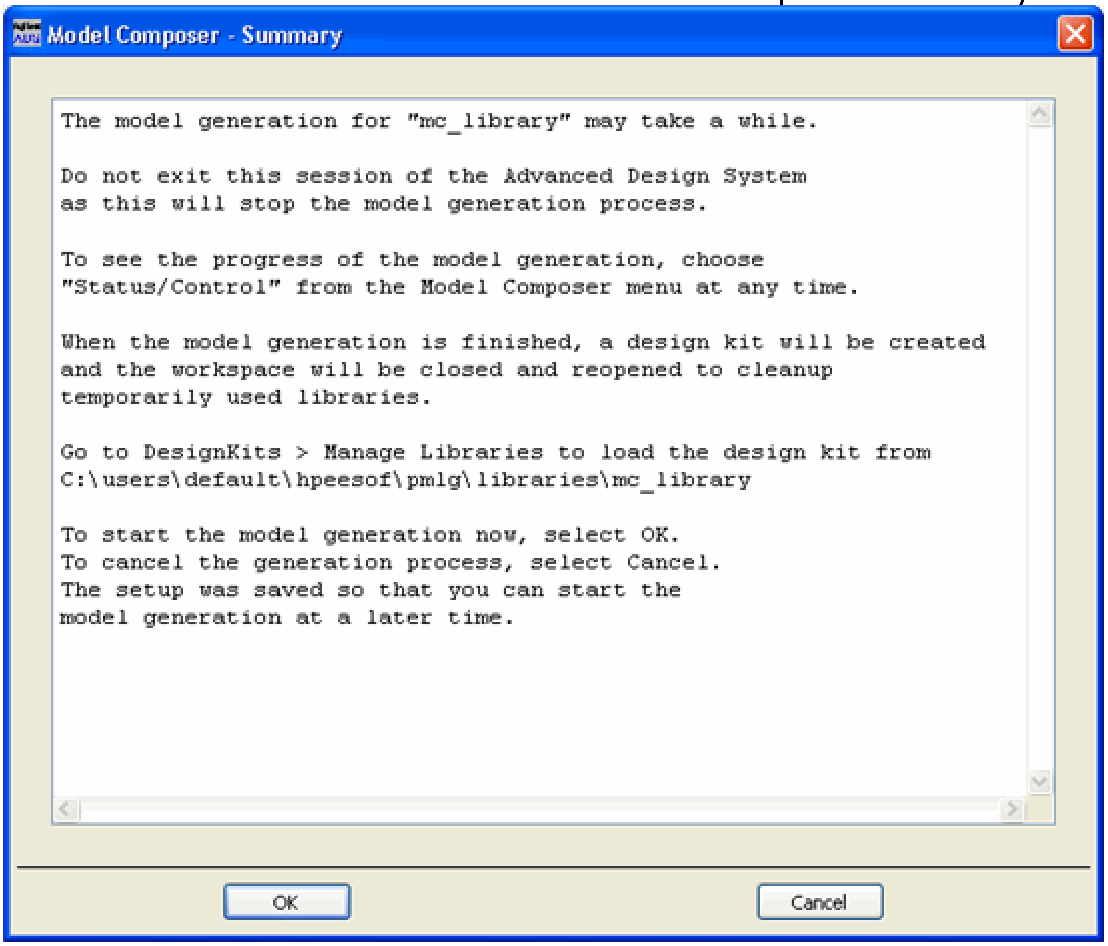


14. Modify the component name and description. For more information, see [Customizing Component Names and Descriptions](#).
15. Click **Next**. The **Parameter Definition** screen is displayed.



Select a component, parameters, component status. For more information, see

16. [Defining Component Parameters](#).
17. Click **Next**. The **Model Generation** screen is displayed.
18. Click **Save** to save your setup before starting the model generation process.
19. A message box is displayed that displays the path of the saved definition file. Click **OK**.
20. Click **Start Model Generation**. The Model Composer Summary screen is displayed.



21. Click **OK**.

> **Hint**
> While defining a library, you can save your work at any time and close the tool. Your partially-defined library will be available next time you start the Model Composer Wizard.

Creating or Editing a Library

You can create a new library of components or edit an existing library:

- If you are creating a new library, type the library name. All libraries are created under `$HOME/hpeesof/pmlg/libraries`, but they can be moved later.
- If you are editing a library, type the library name. If you have moved the library from `$HOME/hpeesof/pmlg/libraries` to another location, click **Browse** to specify the path.

Specifying a Substrate

A substrate defines the layers of material that surround a component, affects the characteristics of the model. All models in a library must be calculated by using the same substrate. If you want to edit a substrate, first select the substrate, then click **Edit**.

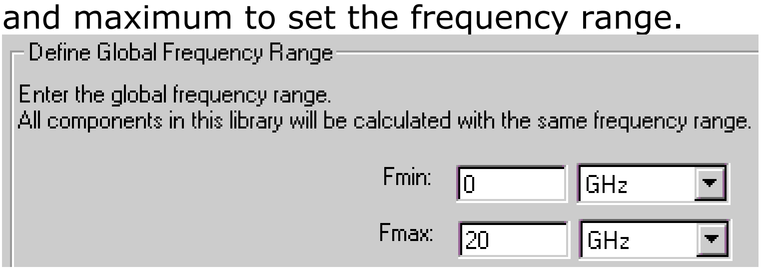
> **Note**
> For more information about substrates, refer *Substrates in EM Simulation* (adstour).

When working with substrates, consider the following:

- There are a variety of predefined substrates that are included with Momentum. They can be found under < installation_directory >*/momentum/lib*. They can be used as is or as the basis for creating a new substrate.
- A substrate can have multiple metallization layers. Strip layers can be used, Slot layers cannot. When defining the component parameters, make sure to specify a valid Layer parameter for each component.
- If you are editing a library and change the substrate, all models in the library must be recalculated.

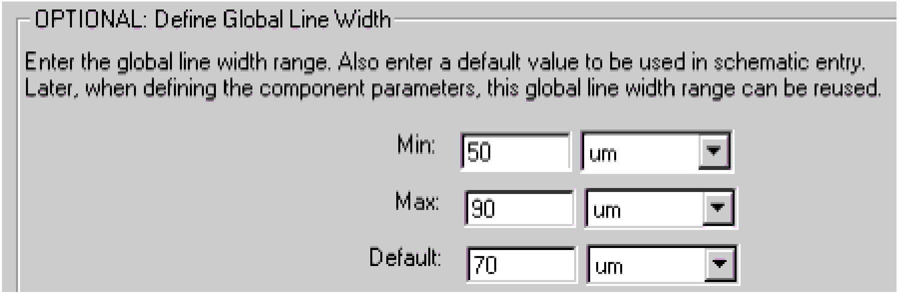
Setting the Frequency Range and Line Widths

The models in a library are calculated over the same frequency range. Specify a minimum and maximum to set the frequency range.



Setting a global line width is optional, but it offers a convenient way to specify a consistent line width for multiple components in a library:

- *Min* and *Max* enable you to specify a range of valid widths. Later, you can easily set component parameters to this variable. This is described in *Defining Component Parameters*.
- The Width parameter of a component is automatically set to the *default* value you enter here. You can see this when you edit a component that has been added to a schematic. You can change the width to any value within the *Min* and *Max* range.

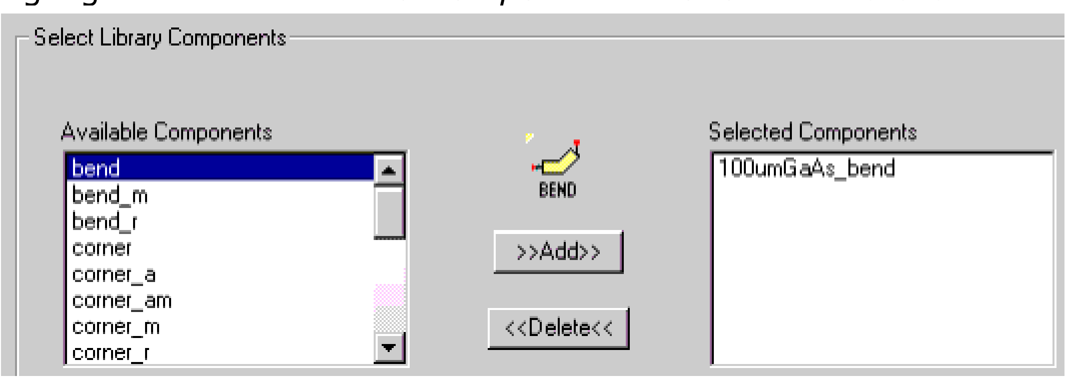


Note
Make sure that the specified ranges are physically meaningful, that is, make sure that no higher order modes are excited for structures with *Max* width at the *Fmax* frequency, on the specified substrate.

Also, be aware that the wider the frequency range and line width, the more simulations will be required to calculate a component and the longer the component generation process will become.

Selecting Components

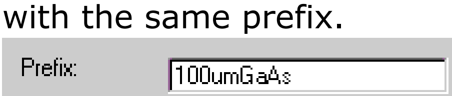
Select the components that you want to add to the library. Choose a component from the list of *Available Components* and click **Add**. To delete a component from the library, highlight it in the *Selected Component* list and click **Delete**.



As you select components, note that:

- To aid in choosing components, a description of the selected component appears at the bottom of the list.

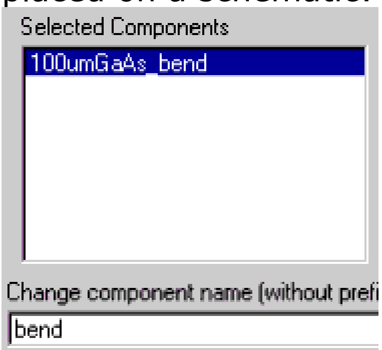
- A prefix is applied to the name of a selected component. The default prefix describes the substrate that was selected earlier in the setup. You can change the prefix as desired. A unique prefix is required, and all components in the library will be updated with the same prefix.



- You can select the same component more than once. This enables you to define the same type of component but with different parameters.

Customizing Component Names and Descriptions

In the Customizing Component screen, you can edit the names of the components in the library as desired. This name will appear as the *Instance Name* when the component is placed on a schematic:



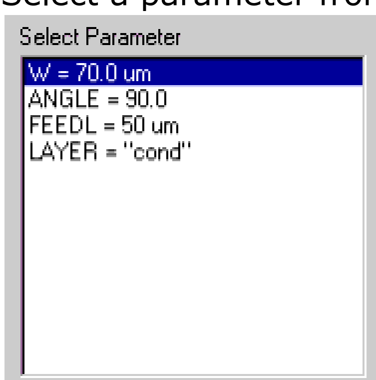
You can also use this screen to edit the component description. This is the information that will appear at the bottom of the Schematic window when the component is selected.

Defining Component Parameters

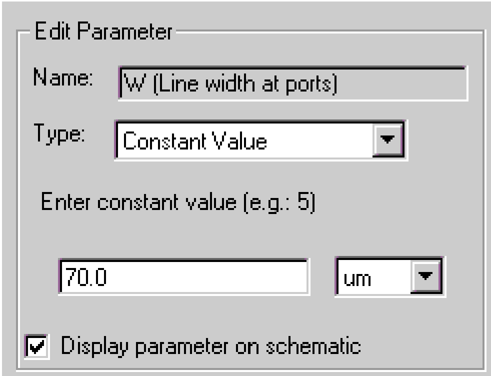
Before the components can be generated, the parameters for each component must be set. These parameters define the limits and ranges of a component, which is taken into account during the model generation process. All parameters have a default value, but you should review all parameters and change them as needed.

- Select a component from the *Select Component* list.

- Select a parameter from the *Select Parameter* list.



- Use the fields on the right side of the panel to edit the parameter.



You can change the parameter type to *Constant Value*, *Discrete List*, *Continuous Range*, or *Global Parameter*.
!modcomp-2-01-15.gif!** *Constant Value* is a single value and you will not be able to edit this value later on, such as after you add the component to a schematic.

- *Discrete List* is a set of discrete values. When you use the component on a schematic, you can edit the parameter to any value in this list.
- *Continuous Range* enables you to change the parameter to any value within the range after the component is added to a schematic.
  For practical reasons a maximum of 2 parameters in a component can be modeled in a continuous way.
- *Global Parameter* sets the parameter to the continuous global line width range that you defined earlier. When you add the component to a schematic, the parameter will be set to the default value you specified for the global line width. You can change the parameter to any value in the global line width range.

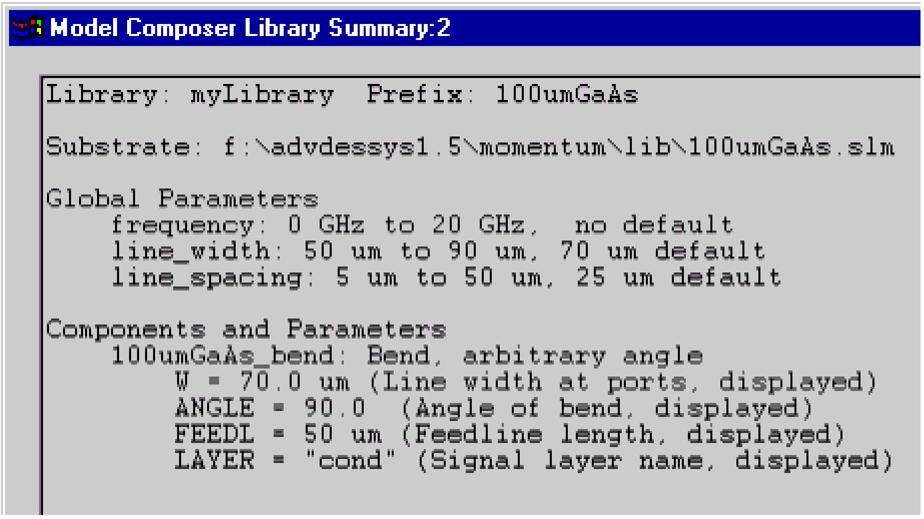
All components modeled with the Model Composer have a feedline attached to each port. As the modeling process is based on EM simulations, the components must have a non-zero length. Typically, the height of the substrate, or the width of line, can be used as feedline length (FEEDL). Make sure that the Layer parameter is set correctly.

Next, verify the Component Status of a component:

- *new* means the component has not been calculated yet or the parameter definition has changed so that a recalculation is required. You will see this status when a new component is added to the library, when the substrate is modified, or if component parameter settings are changed. If you want to force a component to be recalculated, click **Recalculate model**.
- *done* means that component modeling has been successfully calculated.
- *error* means that an attempt was made to calculate the component, but it was unsuccessful. The error is probably due to combinations of parameters and substrate settings that are physically impossible. For suggestions on how to correct this, refer to *Troubleshooting a Library*.
- *computing* means that the component modeling is in currently in process, the modeling process was stopped and is partially complete, or the component was skipped. Any earlier calculated data will be (re-) used if the simulation continues or restarts.

Starting the Generation Process

Click **View Summary** to review your settings; thoroughly check all parameter values and ranges to ensure the models are successfully generated.



To begin generating the components, click **Save**, then click **Start Library Generation**. The Model Composer Library Summary window will display. It explains that the library generation process will start as a separate process running in the background, and it tells you where the completed library will be stored. Click **OK** to start the modeling process now.

At this time, you can exit ADS, or continue with other tasks, depending on the available computer memory and processor power.

A Momentum status window will pop-up after some time. This window can be minimized, and it will disappear when the library generation is done.

A library can take some time to complete, so use the Status Control to view progress. Refer to Viewing and Controlling the Model Generation Process.

The components are generated based on the substrate, frequency range, and component parameter settings. Depending on the parameter definitions for a component, an appropriate number of Momentum simulations are run for each component, and the behavior of the component is saved in the form of S-parameters.

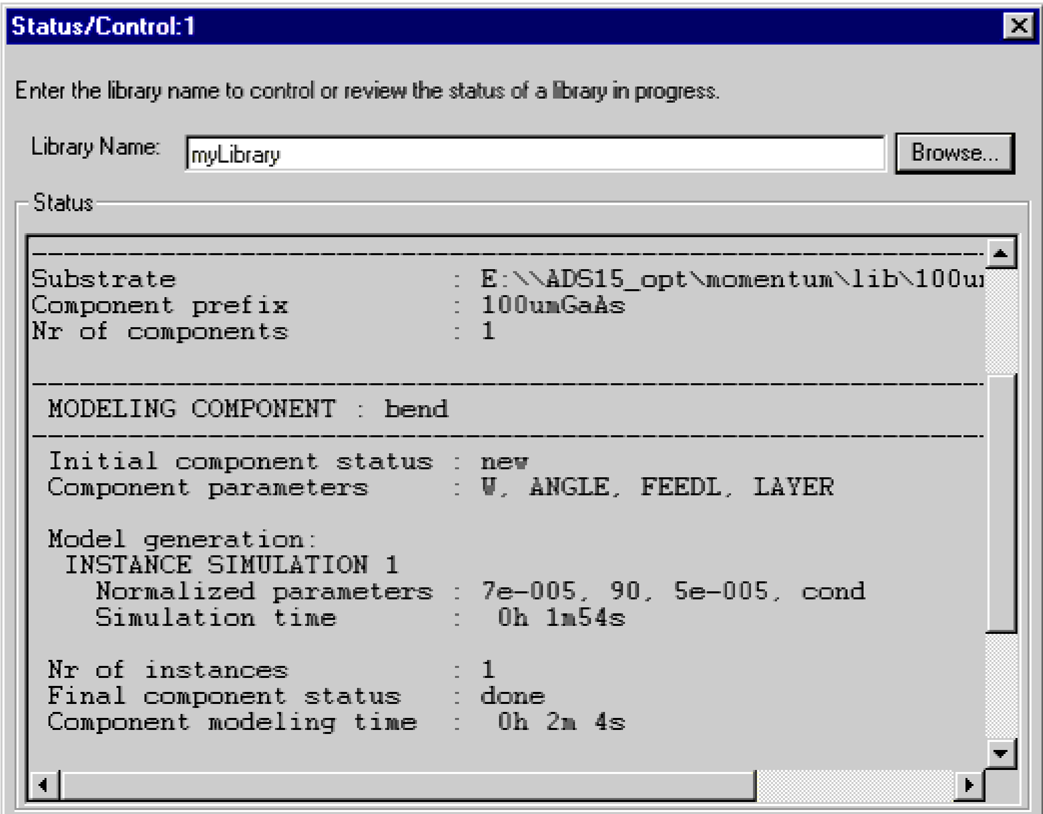
If the library is taking a very long time to generate or you are getting errors:

- Make sure that the frequency range, substrate definition, and component parameters make physical sense.
- Try reducing the frequency range and the number of continuously-varying parameters of a component. Continuously varying parameters can be replaced by discrete parameter lists or constant parameter values.
- For more suggestions, refer to *Troubleshooting a Library*

Viewing and Controlling the Model Generation Process

To check on the progress, from the *Tools* menu choose *Model Composer > Status/Control*. This can be done from any ADS session that is running on the same computer and started by the same user. Type in the *Library Name* or click **Browse** to locate the library of interest. This dialog box:

- Lists the status of the components in the library
- Displays any error messages or warnings
- Enables you to stop the process for a component or the entire library



Model Composer Components

This section lists the model composer components. For more information, click the required component.

B

- *bend (Arbitrary Angle Bend)* (modcomp)
- *bend m (Mitered Arbitrary Angle Bend)* (modcomp)
- *bend r (Rounded Arbitrary Angle Bend)* (modcomp)

C

- *corner (90-degree Corner)* (modcomp)
- *corner a (90-degree Asymmetric Corner)* (modcomp)
- *corner am (90-degree Asymmetric 50 percent miter Corner)* (modcomp)
- *corner m (90-degree Mitered Corner)* (modcomp)
- *corner r (90-degree Rounded Corner)* (modcomp)
- *cross (Cross Junction)* (modcomp)
- *cross s (Symmetric Cross Junction)* (modcomp)

G

- *gap (Symmetric Gap)* (modcomp)
- *gap a (Asymmetric Gap)* (modcomp)

O

- *open (Open-end Effect)* (modcomp)

R

- *rstub (Radial Stub)* (modcomp)

S

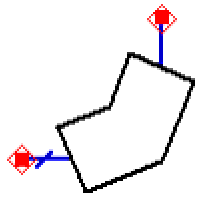
- *slit (Symmetric Slit)* (modcomp)
- *slit a (Asymmetric, One-sided Slit)* (modcomp)
- *step (Step in Width)* (modcomp)
- *step a (Asymmetric, One-sided Step in Width)* (modcomp)

T

- *taper (Tapered Step in Width)* (modcomp)
- *taper a (Asymmetric, One-sided, Tapered Step in Width)* (modcomp)
- *tee (Tee Junction)* (modcomp)
- *tee s (Symmetric Tee Junction)* (modcomp)

bend (Arbitrary Angle Bend)

Symbol



Parameters

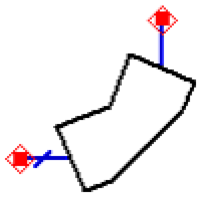
W = Line width at ports
ANGLE = Angle of bend, in degrees
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
-135 ≤ ANGLE ≤ 135
FEEDL > 0

bend_m (Mitered Arbitrary Angle Bend)

Symbol



Parameters

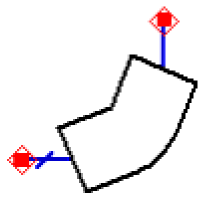
W = Line width at ports
ANGLE = Angle of bend, in degrees
M = Miter fraction
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
-135 ≤ ANGLE ≤ 135
0 ≤ M ≤ 1
FEEDL > 0

bend_r (Rounded Arbitrary Angle Bend)

Symbol



Parameters

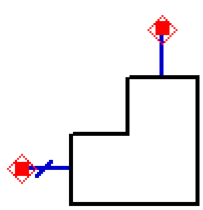
W = Line width at ports
ANGLE = Angle of bend, in degrees
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
-135 ≤ ANGLE ≤ 135
FEEDL > 0

corner (90-degree Corner)

Symbol



Parameters

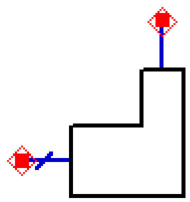
W = Line width at ports
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
FEEDL > 0

corner_a (90-degree Asymmetric Corner)

Symbol



Parameters

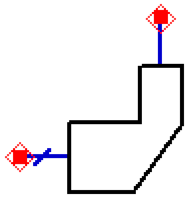
W1 = Line width at port 1
W2 = Line width at port 2
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
FEEDL > 0

corner_am (90-degree, Asymmetric, 50%-miter Corner)

Symbol



Parameters

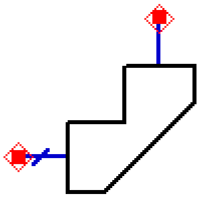
W1 = Line width at port 1
W2 = Line width at port 2
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
FEEDL > 0

corner_m (90-degree Mitered Corner)

Symbol



Parameters

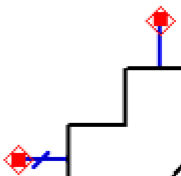
W = Line width at ports
M = Miter fraction
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
0 ≤ M ≤ 1
FEEDL > 0

corner_r (90-degree Rounded Corner)

Symbol



Parameters

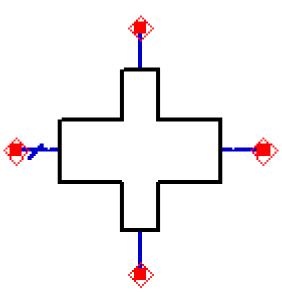
W = Line width at ports
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
FEEDL > 0

cross (Cross Junction)

Symbol



Parameters

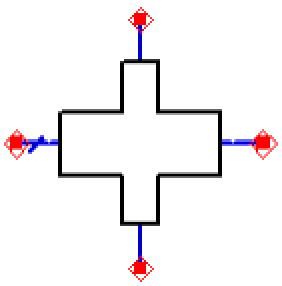
W1 = Line width at port 1
W2 = Line width at port 2
W3 = Line width at port 3
W4 = Line width at port 4
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
W3 ≥ 0
W4 ≥ 0
FEEDL > 0

cross_s (Symmetric Cross Junction)

Symbol



Parameters

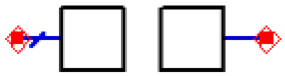
W_thru = Thru line width
W_cross = Cross line width
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W_thru ≥ 0
W_cross ≥ 0
FEEDL > 0

gap (Symmetric Gap)

Symbol



Parameters

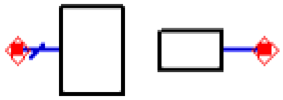
W = Line width
S = Spacing of gap
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
S ≥ 0
FEEDL > 0

gap_a (Asymmetric Gap)

Symbol



Parameters

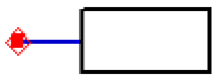
W1 = Line width at port 1
W2 = Line width at port 2
S = Spacing of gap
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
S ≥ 0
FEEDL > 0

open (Open-end Effect)

Symbol



Parameters

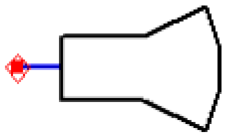
W = Line width
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
FEEDL > 0

rstub (Radial Stub)

Symbol



Parameters

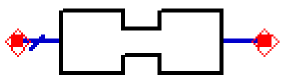
W = Width of input line
L = Length of stub
ANGLE = Angle of stub, in degrees
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
L ≥ 0
-270 ≤ ANGLE ≤ 270
FEEDL > 0

slit (Symmetric Slit)

Symbol



Parameters

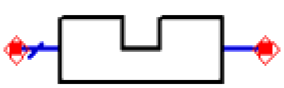
W = Line width at ports
W_slit = Line width of slit
L = Length of slit
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
W_slit ≥ 0
L ≥ 0
FEEDL > 0

slit_a (Asymmetric, One-sided Slit)

Symbol



Parameters

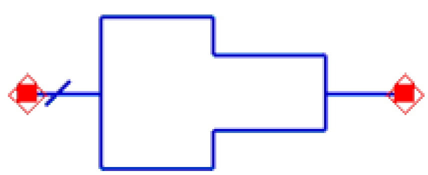
W = Line width at ports
W_slit = Line width of slit
L = Length of slit
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W ≥ 0
W_slit ≥ 0
L ≥ 0
FEEDL > 0

step (Step in Width)

Symbol



Parameters

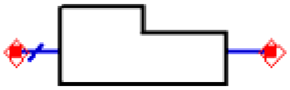
W1 = Line width at port 1
W2 = Line width at port 2
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
FEEDL > 0

step_a (Asymmetric, One-sided Step in Width)

Symbol



Parameters

W1 = Line width at port 1
W2 = Line width at port 2
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
FEEDL > 0

taper (Tapered Step in Width)

Symbol

Parameters

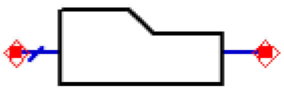
W1 = Line width at port 1
W2 = Line width at port 2
L = Length of taper
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
L ≥ 0
FEEDL > 0

taper_a (Asymmetric, One-sided, Tapered Step in Width)

Symbol



Parameters

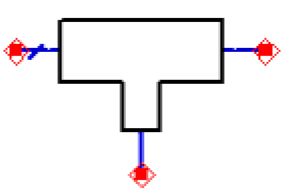
W1 = Line width at port 1
W2 = Line width at port 2
L = Length of taper
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
L ≥ 0
FEEDL > 0

tee (Tee Junction)

Symbol



Parameters

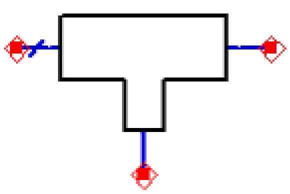
W1 = Line width at port 1
W2 = Line width at port 2
W3 = Line width at port 3
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W1 ≥ 0
W2 ≥ 0
W3 ≥ 0
FEEDL > 0

tee_s (Symmetric Tee Junction)

Symbol



Parameters

W_thru = Thru line width
W_stub = Stub line width
FEEDL = Feedline length
LAYER = Signal layer name

Range of Usage

W_thru ≥ 0
W_stub ≥ 0
FEEDL > 0

Troubleshooting a Library

This section identifies the problems that you may encounter while using with the Model Composer feature and offers suggestions to fix the problem.

Problem: *The library does not appear in the Component Palette or Component Library.*

To troubleshoot this problem:

- Ensure that you have installed the library (ADS Design Kit). Refer to *Preparing to Use Models.*

- Ensure that your design kit and its installation level are both enabled. For more information on installing and enabling design kits, refer to the ADS *"Design Kit Installation and Setup"* documentation.
- Ensure that there are no conflicts between the new ADS Design Kit software and an older version of the software.

Problem: *Errors are produced when attempting to generate a library of components.*

This can be due to a variety of problems. Usually, this will be caused by the unphysical combination of:

- Substrate
- Maximum frequency
- Layout parameters

The higher order modes are excited, the Momentum simulator produces unphysical results, and the Model Composer modeling process does not converge.

To solve this problem:

- Limit the frequency range.
- Limit parameter ranges (for example, limit width).
- Check units (for example, m instead of um).
- Limit the number of continuously varying parameters. The maximum is two per component, for practical reasons. Consider that the modeler uses about 5 to 10 Momentum simulations for each continuously-varying parameter. This means that one continuous parameter would require 5 to 10 Momentum simulations; two continuous parameters, 25 to 100 simulations; three would require 125 to 1000 simulations; 4 continuous parameters, 625 to 10,000 simulations.
- Use constant values and discrete lists. Quite often, the line width of the metallization is limited, and varies in a discrete way, based on a given technology or application. If so, use discrete parameter lists that correspond with the real world limitations.
- Replace a single component definition (with widely varying ranges) by multiple components of the same kind, each valid in a subset of the original parameter space.
- Perform a worst case analysis of the library specs (check the combination of max frequency, max width, and substrate) to make sure that no higher order modes occur. Momentum can be used to see if the simulation results (S-data) make sense.

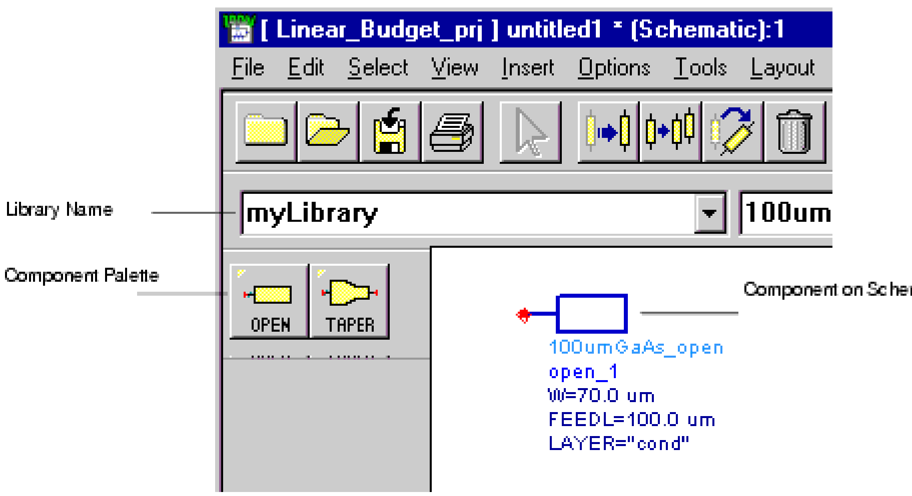
General Guidelines

Refer the following guidelines for troubleshooting problems:

- A general workaround in most cases is to use discrete lists and limit ranges. The wider the ranges, and the more continuously varying parameters, the more Momentum simulations are needed, and the longer the modeling time will be.
- Examine the parameter ranges and ensure that all values are physically possible.
- If higher order modes occur, the Momentum simulation will fail, and so, then, will the Model Composer process.

Using Models

Components created with the Model Composer can be added to a schematic like any other ADS component. You can also edit parameters.



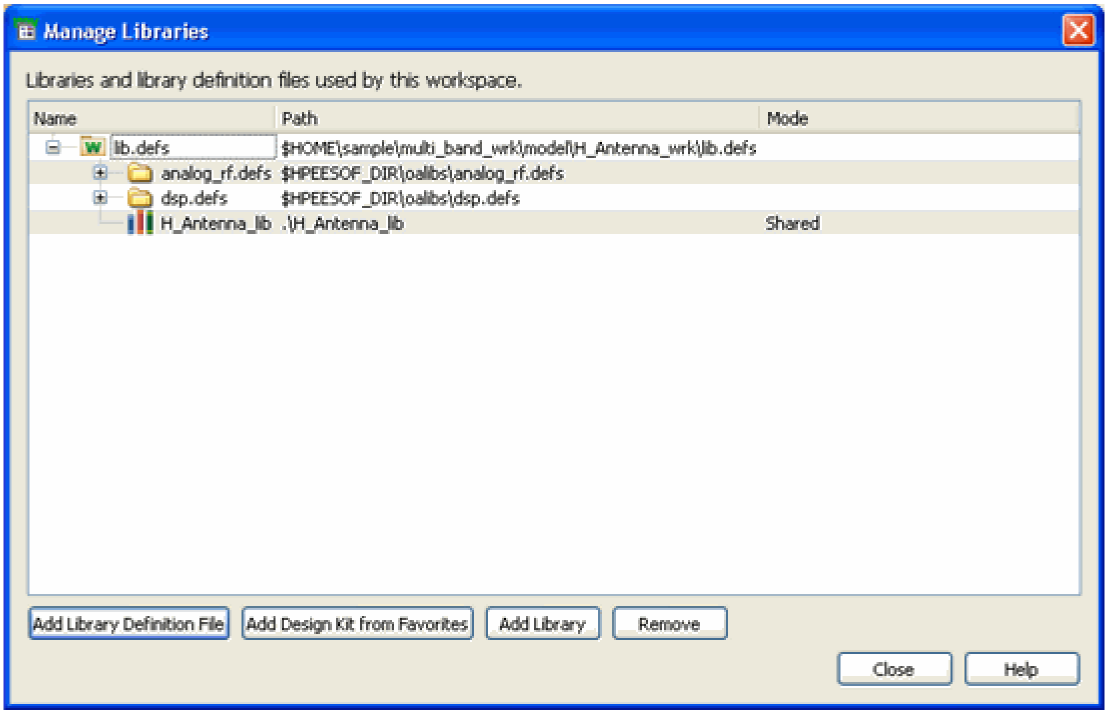
To use components created with the Model Composer, refer to these topics:

- Preparing to Use Models describes the steps you need to take before you can use a library. *You must complete this section in order to use a library*.
- Locating Models describes how to select a library and access components.
- Editing Component Values describes how to edit the parameters of a component.
- Moving and Copying Libraries describes how to copy or move a library to another location.

Preparing to Use Models

The library generated by the Model Composer is actually a standard ADS Design Kit. Before you can use a library, you must first install the design kit.
To add the library definition file:

1. From the ADS Main window, choose **DesignKits** > **Manage Libraries**. The **Manage Libraries** dialog box appears, as shown in the following figure:

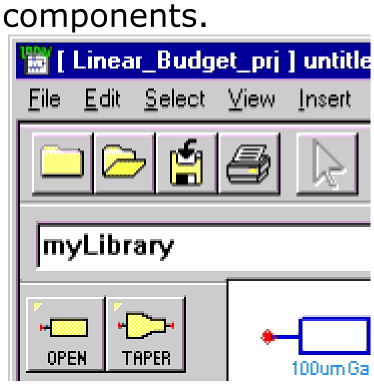


2. Click **Add Library Definition File** and browse to the following directory: *$HOME/hpeesof/pmlg/libraries/ <library_name>*
   where *<library_name>* is the name defined in Creating or Editing a Library.
3. After the path is entered, the name of the library definition file (lib.defs) is automatically updated.
4. Click **Open** in the **Select Library Definition File** dialog box to open the library definition file.
5. Click **Close** to close Manage Libraries dialog. Your Model Composer library is now ready for use.

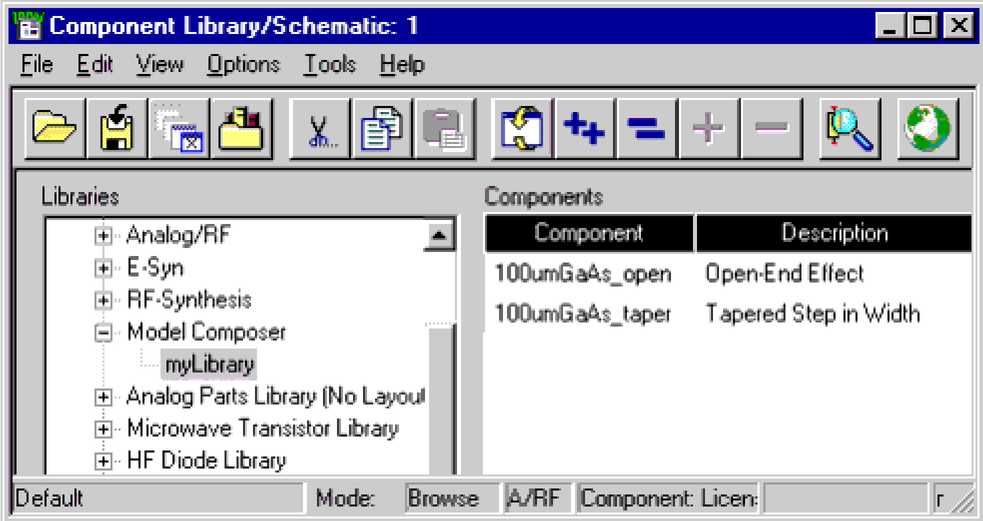
Locating Models

You access and use your models in the same way as any other components in ADS:

- **From the Component Palette**: The name of the library appears in the Component Palette list. When you select the library, a palette of the components you created is displayed. You select and place these components in the same way as other ADS components.



- **From the Component Library**: In the lists of libraries locate *Model Composer*, your libraries will appear below this.



> **Note**
> If a library does not appear in the Component Palette or Component Library, make sure you completed the steps in the section Preparing to Use Models and that you have selected Analog/RF Design as your design type.

Editing Component Values

You can edit component parameters, based on the following conditions:

- A component has *constant* parameters, these parameters cannot be changed.
- A component was created with *discrete* parameters, you can only select values that were specified when the model was generated.
- A component was created with *continuous* or *global* parameters, you can select any value within the range that was specified when the model was created.

A component can have a combination of constant, discrete, continuous, and global parameters. For more information on how parameters are defined, refer to the section *Defining Component Parameters*.

> **Note**
> A library can contain components located on different substrate layers. Make sure the *Layer* parameter is set correctly for any components you use from a library.
>
> A feedline of length FEEDL is attached to each port of the components. This length should be taken into account while using the components from a library.

Moving and Copying Libraries

All libraries are created under `$HOME/hpeesof/pmlg/libraries`, but they can be moved or copied to a new location. If you do this, be sure to complete the steps in the section Preparing to Use Models. Otherwise, you may use the incorrect copy or the library name may not appear in the Component Palette or Component Library list.

Note
You may want to zip the library prior to relocating it. To do this, open a term or MS-DOS window and type:
`zip -r <` *library_name* `> $HOME/hpeesof/pmlg/libraries/<` *library_name* `>`