Agilent Technologies

Advanced Design System 2002

MDS Design Translation

February 2002

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms that apply to this software product is available upon request from your Agilent Technologies representative.

Restricted Rights Legend

Use, duplication or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

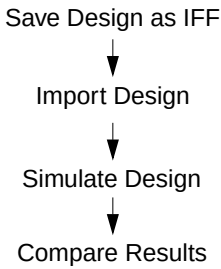
Agilent Technologies
395 Page Mill Road
Palo Alto, CA 94304 U.S.A.

Contents

Chapter 1: Design Translation Overview

The MDS-to-ADS translator allows you to import MDS designs into a new or existing Advanced Design System project. This is done by saving MDS designs in IFF format and passing them through the translator. Each MDS component is replaced by its MDS equivalent or closest ADS equivalent, and the layout geometry is preserved.

The following steps are normally followed when translating MDS designs for use in the Advanced Design System:



- Save Design as IFF: The MDS design is saved as an IFF file using the migration script *migrate.ddl*.
- Import Design: The design is imported into an Advanced Design System project. For information on importing designs, refer to "Importing and Simulating" on page 2-1.
- Simulate Design: The design is simulated. The Simulation/Synthesis window appears and displays the simulation status including any problems that were encountered.
- Compare Results: The ADS and MDS simulation results are compared to verify that the translation occured normally. The results may differ because MDS and ADS use different components and simulators. Significantly different results could be an indication of a translation or model mapping problem.

Translated Items

The translator allows you to migrate the following libraries from MDS to ADS:

- Microwave (MWLib)
- System Model (SysLib)
- Multilayer (MultiLayerLib)

A portion of the MDS components are not supported. These are identified in "Translation Issues" on page E-1. Note that DDL scripts, presentations, workbenches, and documents are not translated. Datasets can be imported into ADS using CITI files (refer to "Importing MDS Datasets" on page 2-5).

Chapter 2: Importing and Simulating

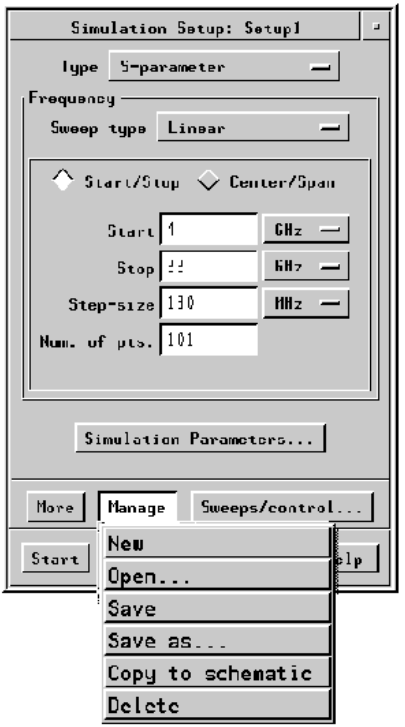
To use MDS designs in ADS, the designs must be exported from MDS, translated to ADS, and the MDS datasets imported. The procedures for accomplishing these tasks are presented in this chapter.

Exporting Designs from MDS

The translator comes with the file *migrate.ddl* which allows you to export hierarchical designs from MDS. This file should be copied from the ADS installation directory *$HPEESOF_DIR/config* to the directory that you use to start MDS.

Note Do not use the command *Export > IFF* to generate the IFF file.

In MDS there are two ways to specify the simulation setup: using the Simulation Setup dialog box (shown in the following illustration) or by placing the simulation components on the schematic.



If you use the Simulation Setup dialog box, the simulation controls might not be correctly set in ADS. To prevent this problem, convert the dialog box settings to components on the schematic before exporting the designs. To do this, click the **Manage** button and select **Copy to schematic**. This will place simulation components on the schematic.

To export the design from MDS:

1. Start MDS.
2. Open the Workbench window containing the designs or the top design of a design hierarchy.
3. Choose **PERFORM** > **DDL** in the top menu bar.
4. The DLL Command File dialog box appears. It prompts you for the name of a DDL file. Enter **migrate.ddl** and click **OK**.

5. The Migration dialog box appears and prompts you for the name of the Migration Output File. Enter a filename that ends with an *.iff* extension (i.e., *amp900mhz.iff)*. If you invoked the DDL script from a workbench window, you also have the option to include or exclude layout pages (by default, layouts are not translated). Click **OK**.
6. The status of the export will appear in the Messages dialog box. The message *Output Complete* should appear. It indicates that the export was successful.

   If the dialog box indicates that a component was not found during the export, you may not have the necessary library files mounted on the MDS screen.
7. Repeat the previous steps for each design that you need to migrate.

Translating Designs into ADS

An MDS design can be imported into an ADS project using the steps given below.

Note Prior to translating designs/files that reference or consist of MDS user-defined models, you need to update those models with their ADS equivalents.

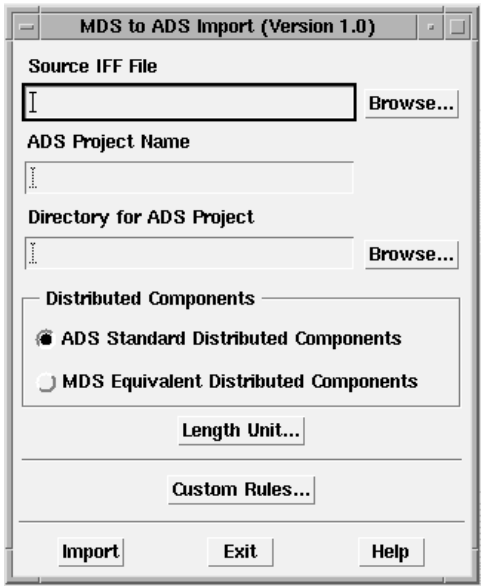
When exporting a design from MDS to ADS, use the script *migrate.dll*.

To import an MDS design into an ADS project, do the following:

1. Start the translator.

   On UNIX systems, type **mds2ads** at the prompt.

   On a PC, from the Start menu choose **Programs** > **Advanced Design System 2001** > **ADS Tools** > **Microwave Design System Import**.
2. The MDS to ADS Import dialog box appears.



Identify the MDS IFF filename by entering the path and filename name into the Source IFF File field. You can also click the **Browse** button and use the Import File Selection dialog box to locate and select the file.

3. In the ADS Project Name field, accept the existing ADS project name or specify a different one (if the project does not exist, it will be created).

Note Spaces are not allowed in project paths or project names.

---

4. In the Directory for ADS Project field type the complete path. You can also click the **Browse** button and use the Destination Project Selection dialog box to set the path.
5. Select the Distributed Components that you want to use.

Note The ADS standard distributed components and MDS equivalent distributed components may produce slightly different simulation results. It is recommended that you use the ADS distributed components.

---

6. Click **Length Units** and select the appropriate setting. This establishes a default for components placed in the new project. It also serves as a default for all designs in the project and is both:
   - The unit of measure for parameters with physical length (in both Schematic and Layout windows)
   - The design unit (grid display and cursor snapping) in the Layout window

   Click **OK** to establish the Length Units.
7. If the design(s) you are translating references user-defined models for which you have created one or more customized rules files, click **Custom Rules**. Select the **Use Custom Rules** check box and click **OK**. For details on creating and using customized rules files, refer to "Creating Customized Rules Files" on page C-9.
8. Click **Import** to begin the import process. The Status window displays feedback during the import process and will list any problems that occur. This information is also written to the file *me_proj.log* in the ADS project directory. The file *me_err.log* in the same directory contains detailed information about the component translation.

Importing MDS Datasets

To import an MDS dataset into ADS, do the following:

1. From the MDS main window, double-click the design icon of interest.
2. The File window appears. Choose **Window** > **Change Page** > **Index Page**.
3. Right-click and choose **Perform** > **Write** > **Citifile**.
4. The Citifile dialog box appears. Enter the dataset name and click **OK**.
5. From the ADS Main window, choose **File** > **Open Project** to open the translated MDS project.
6. From a Schematic or Layout window, start the Instrument Server (*Window* > *File/Instrument Server*, or click the toolbar icon).
7. The Instrument Server window appears. In the Read From area, select **File**.
8. Select **Citifile** from the File Format to Read from list.
9. Enter the name of the Citifile in the File Name field or click the **Browse** button to locate and select it.

10. Type the name of the dataset in Dataset Name field.
11. Click the **Read File** button. The Status Window appears. It should say that the file was read successfully.
12. Click the **Update Dataset List** button.

Specifying the Metal Layer for Circuit Components in Layout Pages

MDS circuit elements in layout drawing pages are laid out using a table that maps symbolic layer names (metal, assembly,...) to physical layer names. For example, the default metal layer for mwlib elements is FMetal. In ADS, the metal layer for microstrip elements is always layer 1. You can't change this assignment. A layer mapping rule is used to swap the metal layer with the layer that is assigned the ID = 1.

A layer mapping rule uses the following syntax:

```
layerIcon| ;metalLayer; ; ;MDSlayer1|
```

where

layerIcon is the name of the MDS layout icon that contains the layer definition being used.

metalLayer is the name of the metal layer.

MDSlayer1 is the name of the MDS layer with ID = 1. This entry must be specified in the field n, where n is the ID of the metal layer in MDS.

For example, the following rule is used to make sure that mwlib and MMICA circuit elements are correctly drawed in ADS:

```
mwlib_layerDef| ;FMetal; ; ;FDieDef|
MMICA_layers| ;SecndMet; ; ; ; ; ; ;Ohmic|
```

Refer to Appendix A, Model and Component Differences, for more information on writing custom translation rules.

Chapter 3: Translation Example

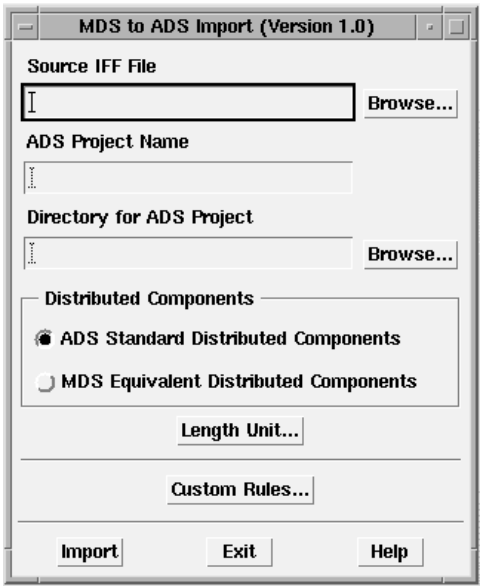
This chapter provides a detailed example that illustrates how an MDS design is imported into ADS. In the example, an IFF file is created for an MDS FET amplifier and the *mds2ads* utility is used to translate the design from MDS to ADS. After translation, the design is simulated in ADS and the S-parameter data is plotted.

Translating the Design

To import an MDS design into an ADS project:

1. Launch MDS and double-click the **Passive** library icon.
2. The File window appears. Double-click on **FET_matching_network**.
3. In the Workbench window, double-click the **Schematic** icon.
4. The schematic appears. From the menu bar, choose **Perform** > **DDL**.
5. In the Command File dialog box enter **migrate.ddl** and click **OK**.
6. The Design Migration dialog box appears. Enter **fet_matching.iff** and click **OK**. The Message window appears and provides the status of the IFF file creation process.
7. Double-click the **Dataset** icon.
8. Right-click and choose **Perform** > **Write** > **Citifile**.
9. Enter the name of the CITI file (e.g., *FET_matching_network_mds.cti*) and click **OK**.
10. Start the translator.
    - UNIX—Type **mds2ads** at the prompt.
    - PC—From the Start menu, choose **Programs** > **Advanced Design System 2001** > **ADS Tools** > **Microwave Design System Import**.

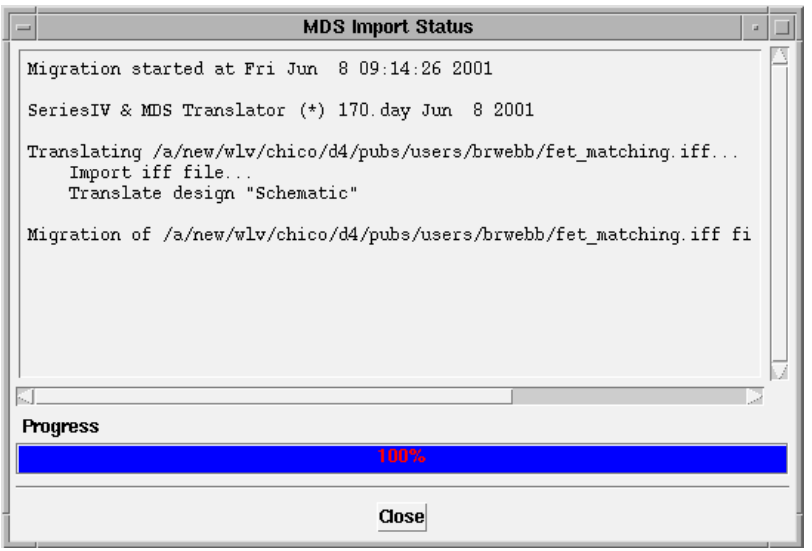
11. The MDS to ADS Import dialog box appears.



Click the Source IFF File **Browse** button, locate the IFF file that you have just created, select it, and click **OK**.

12. Return to the MDS to ADS Import dialog box and accept the default ADS project name or enter a new one.
13. Click the Directory for ADS Project **Browse** button and select a directory.
14. Click the **Import** button in the MDS to ADS Import dialog box.

15. The MDS Import Status window appears. When the translation is complete, click **Close**.



16. Click **Exit** in the MDS to ADS Import dialog box.

Importing the MDS Dataset

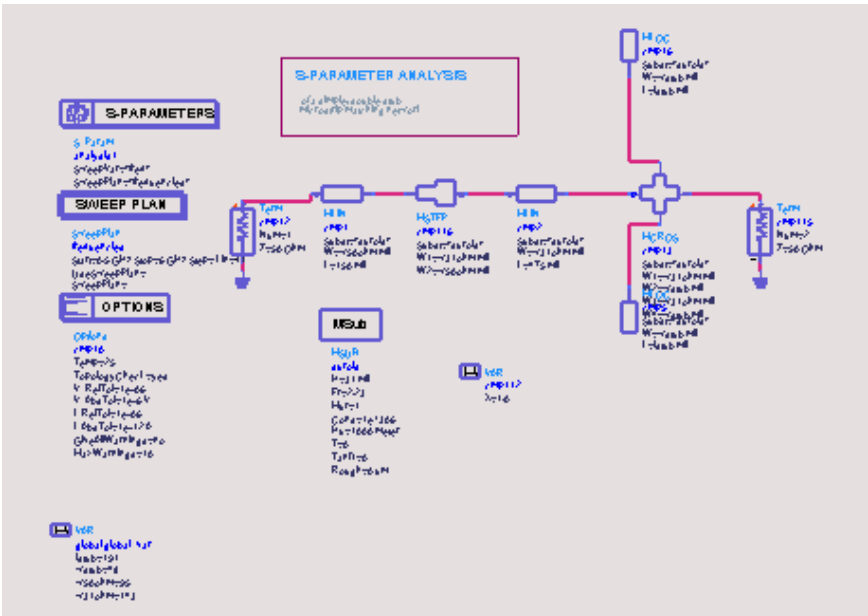
To import the MDS dataset containing the MDS simulation results:

1. Launch ADS.
2. From the ADS Main window, choose **File** > **Open Project**.
3. In the Open Project dialog box locate *fet_matching_prj*, select it, and click **OK**.
4. Start the instrument server (*Window > File/Instrument Server*).
5. Click **Read** followed by **File**.
6. Select **Citifile** from the file format list.
7. Click the File Name **Browse** button to select the Citifile. Specify the dataset name.
8. Click the **Read File** button.

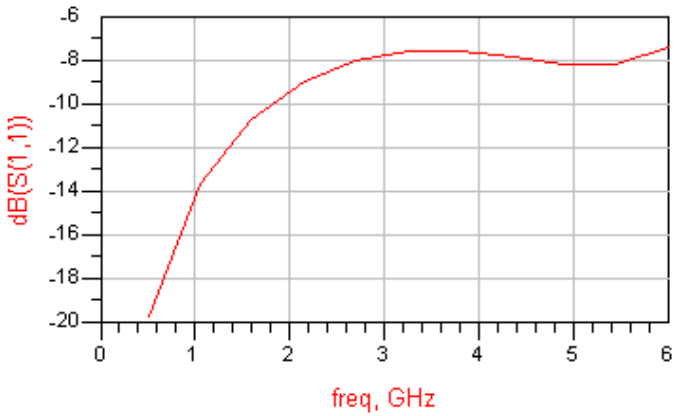
Simulating the Design in ADS

To simulate the design in ADS:

1. Open the design if it is not yet open. Choose **Simulate** > **Simulate** from the Schematic window, or click the Simulate icon.



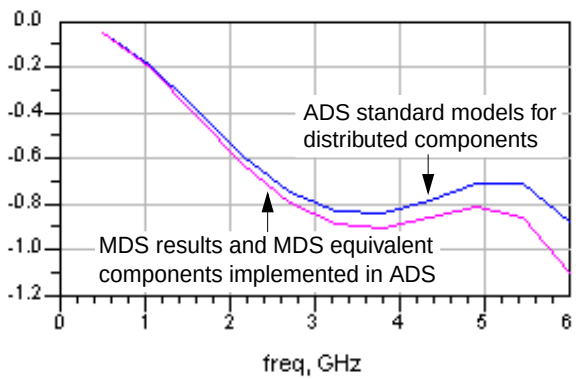
2. When the Data Display window opens, click the **Rectangular Plot** icon, move the pointer to the display area, and click to place the plot.
3. The Plot Traces & Attributes dialog box appears. Select **S(1,1)**, click **Add**, and click **OK**. The S(1,1) data is plotted in the Data Display window (shown next).



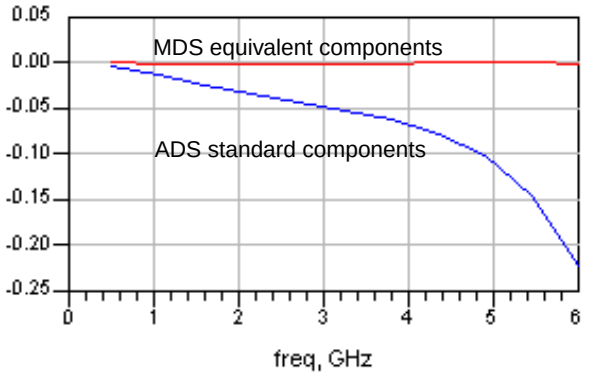
Comparing MDS and ADS Results

MDS and ADS simulations can provide different results because of the different simulation models used for the distributed components.

The first plot below shows the simulation results for the transmission parameter S12. The blue trace represents the ADS results obtained with the ADS standard models for distributed components. The magenta trace shows the results obtained with MDS. The trace of the ADS results obtained with MDS equivalent components implemented in ADS is directly under the magenta line and is not visible.



The difference between the MDS and ADS results is presented on the second plot (shown next). The blue trace shows the results using ADS standard models. The red trace shows the results when MDS equivalent components are used.



Appendix A: Model and Component Differences

Introduction

A large number of MDS distributed components have been added to ADS 2001 to allow for the translation of MDS designs to ADS. They are named using an" _MDS" suffix appended to the end of the MDS component name. These components do not appear on the ADS component palettes, but they can be added by typing the component name in the ADS Component History field.

Similar Components

Of the MDS distributed components added to ADS 2001, there are several similar ADS components. These similar components originally came from Series IV and are treated as ADS standard distributed components.

The similarities and differences of MDS and ADS components are summarized in the following tables.

Table A-1. Similar MDS and ADS Components.

Library	Components
Coplanar	CPWCTL, CPWTL, GCPWTL
Interconnects	WIRES
Microstrip	MS3CTL, MSABND, MSACTL, MSAGAP, MSBEND, MSCTL, MSCRNR, MSCROSS, MSGAP, MSICDF, MSIDC, MSLANGE, MSOBND, MSOC MSRND, MSRTK2, MSRTL, MSRTL2, MSSLIT, MSSPLC, MSSPLR, MSSPLS, MSSTEP, MSTAPER, MSTEE, MSTL, TFC, TFR
Nonlinear Devices	BJT (Gummel-Poon, VBIC), MEXTRAM), Diode (Diode, HP Diode), MESFET (Curtice, Materka, TOM1, HP FET), MOSFET (Level 1 and 3, HP MOS, MOS Model 9, BSIM1, BSIM2, BSIM3), JFET
Stripline	SL3CTL, SL4CTL, SL5CTL, SLABND, SLCRNR, SLCTL, SLGAP, SLOBND, SLOC, SLRBND, SLTEE, SLTL, SLUCTL, SLUTL
Suspended Substrate	SSCTL, SSTL
Transmission Lines	COAX, CTL, DRC, ETAPER, FINLINE, RWGTL, TLE, TLOC

When MDS designs are translated to ADS 2001, you can use either the ADS standard distributed components or the MDS equivalent distributed components. You may want to perform separate translations using each set of components and compare the results to verify that the design was correctly translated.

It is recommend that the ADS standard distributed components be used for your final design translation because these components provide more accurate results. Also, the ADS components will be improved in future releases, but the MDS equivalent components may not.

Components Unique to MDS

Thirty-four MDS components do not have equivalent components in ADS. Since these were also added to ADS 2001, the translation of MDS designs will always use these MDS components. They will be used regardless of the distributed components selected in the MDS to ADS Import dialog box.

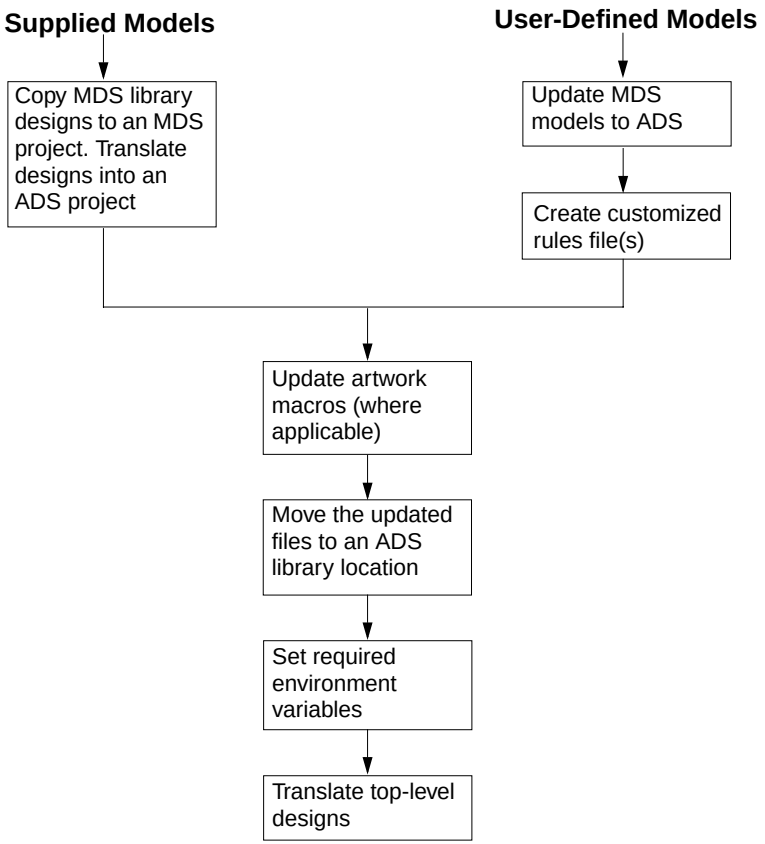
The components unique to MDS are listed in the following table.

Table A-2. Components Unique to MDS.

Library	Components
Coplanar	ACPWDS, ACPWTL, CPWDS, CPWTLFG,
Interconnects	RIBBONG, RIBBONS, WIREG
Microstrip	MS4CTL, MS5CTL, MSAGAP, MSIDC, MSRND, MSSVIA, MSVIA, MSWRAP
Stripline	SL3CTL, SL4CTL, SL5CTL, SLGAP, SLUCTL
Suspended Substrate	BR3CTL, BR4CTL, BRCTL, BROCTL, SS3CTL, SS4CTL, SS5CTL, SSTFR, SSLANGE, SSSPLC, SSSPLR, SWSSPLS
Transmission Lines	SLOTTL, TL

Appendix B: Translating Design Libraries

Translating design libraries consists of several steps. If any of the designs are user-defined (as opposed to designs built from supplied models), additional steps are required. For details on that process, refer to "Creating Customized Rules Files" on page C-9.



Copying Designs to an MDS Project

Several designs can be translated simultaneously. But they must first be part of an MDS project. If you have several designs to translate, and they are not currently part of a project directory (for example, in a directory outside a project that serves as a library), copy them to a project before translation. If you plan on translating designs individually, you can leave them in their current location.

Relocating Designs for Site-Wide Use

Once all designs involved have been translated, you can move the designs into ADS library locations shown below:

Design (*.dsn*) Files: *$HPEESOF_DIR/custom/circuit/symbols*

AEL (*.ael*) Files: *$HPEESOF_DIR/custom/circuit/ael*

Set the SITE_AEL variable to include the search path for the *.ael* files (the */symbols* directory will be searched automatically for *.dsn* files). Example:

```
SITE_AEL = $HPEESOF_DIR/custom/circuit/ael/
```

Set this variable in the file *$HPEESOF_DIR/custom/config/de_sim.cfg*.

Appendix C: Translator Customization

MDS to ADS component translation is accomplished using a database of rules that map the components on a parameter-to-parameter basis. Standard components are mapped with *MDS*.rul*, the supplied rules file. To map custom parts, you need to create a custom rules file.

When designs or files are translated, the translator first looks for a custom rules database and then for the supplied rules database. Custom translation rules take precedence over supplied rules.

Writing Custom Translation Rules

Examples of rules used by the translator can be found on UNIX platforms at

$HPEESOF_DIR/config/MDS.rul*

and on the PC at

%HPEESOF_DIR%\config\MDS.rul*

Rules files can be created using any text editor. Blank lines or lines that begin with a pound (#) symbol are ignored. Translation rules use the following syntax and should be on a single line of the rules file:

```
OldName|ADSName|ParmRule|DefRule|PinRule|NameRule|DelFlag|MappedParmRule|
```

The rule file fields are described in the following table.

Field	Description
OldName	MDS component name
ADSName	ADS component name
ParmRule	Parameter mapping AEL function name (optional)
DefRule	Component definition mapping rule (obsolete)
PinRule SrcPinNum SrcPinName DestPinNum DestPinName	Pin mapping rule list (optional) Old pin number Old pin name (obsolete) Destination pin number Destination pin name (obsolete)
NameRule	Name mapping AEL function name (optional)
DelFlag	Delete flag (TRUE or FALSE)

Note that when any of the above fields is not used (obsolete or optional) the field separator "|" must still be included.	
MappedParmRule ADSParmName ParmName DataItemName DataItemParmName DataItemParmDefForm	Mapped parameter rule list (optional) ADS parameter name MDS parameter name MDS Data Item Name MDS Data Item parameter This parameter is no longer used

Example:

```
XFER|Transformer| | | | |FALSE| |
```

In this example, an MDS component named XFER is translated to an ADS component named Transformer. All parameters that have the same name in MDS as they do in ADS are copied to the new component. A simple rule of this form will translate most custom parts.

Example:

```
XFER| | | | | |TRUE| |
```

In this example (setting the Delete Flag to TRUE), the MDS component will not be translated. The status messages produced during translation will indicate that the component is not translatable and has been removed from the design.

If the names of component parameters need to be changed during the translation, you need to use the MappedParmRule field. The syntax for this field is:

```
ADSParmName,[@|&|%]ParmName,DataItemName,DataItemParmName,DataItemParmDefFor
m,
```

The MappedParmRule field syntax is described in the following table.

Field, MappedParmRule	Description
ADSParmName	ADS Component Parameter Name
ParmName @ - Followed by a default value of the parameter. This allows a new ADS parameter to have a default value rather than copying a value from the MDS component & - The parameter value of MDS component will be quoted as a string. This is normally needed in ADS for items that are not numeric such as the name of a substrate or a filename. % - The case of the specified MDS parameter name is not significant (e.g Cond and COND refer to the same parameter)	MDS Component Parameter Name
DataItemName	MDS Data Item from which to retrieve the named parameter
DataItemParmName	MDS Data Item Parameter from which to retrieve the value
DataItemParmDefForm	No longer used

Example:

```
GYR|Gyrator|  |  |  |False|Ratio,R,  ,  ,  ,;|
```

In this example, the rule copies the value from the old name (R) to the new name (Ratio). The third through fifth parameters are not used.

Example:

```
MONOPOLE|AntLoad| | | | |FALSE|AntType,@MONOPOLE, , , ,;RatioLR,LR, , ,
,;Length,L, , , ,;|
```

In this example, an MDS component called MONOPOLE is translated to a more general ADS component named AntLoad. The parameter AntType is given a default value of "MONOPOLE" by using the "@" character.

Retrieving Parameter Values from MDS Data Items

Because ADS does not support the MDS Data Item (as such), you may want to copy the parameter value of an MDS Data Item to an appropriate ADS component parameter. You can do this using the DataItemName and DataItemParmName fields. For example, you can retrieve the value of the TEMP parameter of the MDS TEMP Data Item—using these fields—and copy it to the ADS parameter Temp, for a given component.

Example:

```
RES|R|  |   |  |FALSE|Temp,TEMP,TEMP,TEMP, ;|
```

In this example, the MDS component RES is translated to the ADS component R and all parameters that have the same name are copied. The MappedParamRule says to copy the value of the TEMP parameter of the MDS TEMP Data Item to the ADS parameter Temp (for this component).

Setting Parameter Values Using AEL Functions

The translator can invoke an AEL function during the translation. To use this feature, specify the name of an AEL function in the ParmRule field. This enables you to extend the translation capabilities, such as computing numerical values from other parameters.

Four pieces of information are combined as a list and sent to the AEL function, which will return an updated parameter list to the translator. The parameter list sent to the AEL function will already have parameter values for all parameters that have the same name in the MDS component that they have in the ADS component. It will also have any parameters copied that are specified by a MappedParmRule. The translator will use the updated parameter list to modify the replaced instance.

Input Parameter:

```
list(list(mdsParmList,    /*MDS component parameter list*/
    adsParmList),   /*ADS component parameter list*/
    instH,          /*New ADS instance handle*/
    mdsItemName)    /*MDS component Name*/
```

Output Parameter:

```
adsParmList - Updated ADS component parameter list
```

The syntax for mdsParmList and adsParmList is shown next.

```
mdsParmList =
```

or

```
adsParmList =

list(list(Name1, Form1, Value1, UnitString1, UnitCode1),
    list(Name2, Form2, Value2, UnitString2, UnitCode2),
    ...
    list(NameN, FormN, ValueN, UnitStringN, UnitCodeN))
```

where

Name	Name of parameter
Form	Form of the current parameter
Value	Value of the parameter (with unit string)
UnitString	Unit string of the current parameter
UnitCode	Unit code of the parameter

Consider the following AEL function:

```
defun merullib_SUBSTRATE ()
{
    decl newParmH, tmpStr, newParmName;
    decl oldParmList = car(car(arg_list()));
    decl newParmList = car(cdr(car(arg_list())));
    decl j, i = listlen(newParmList);
    decl rhsH;

    if (oldParmList != NULL)
    {
      for(j=0; j<i; j++)
      {
          tmpStr = "";
          newParmH = nth(j, newParmList);
          newParmName = car(newParmH);

          if (!strcmp("Cond", newParmName))
          {
              decl rhoH;
              //
              //  get parameter RHO
              //
              tmpStr = merul_extract_parm(oldParmList, "RHO", NULL, NULL, NULL);
                                  if( tmpStr )
              {
                  rhoH = car(merul_plib_parse(tmpStr, nth(MERUL_PARM_UNIT_CODE,newParmH)));
                   //
                  //  construct parameter
                  //
                  if (rhoH || strlen (rhoH) !=0)
                  {
                                          if (val (rhoH) != 0)
```

```
                        tmpStr = identify_value(1.0E+50/val (rhoH));
                    else
                        tmpStr = identify_value(1.0E+50);
                }
                else
                    tmpStr = identify_value(1.0E+50);
            }
        }
        else
        if (!strcmp("Mur", newParmName))
        {
            decl tmpParmList = meutil_get_dataitem_parmlist( TRUE, "PERM", FALSE);
            tmpStr = merul_extract_parm(tmpParmList, "MUR", NULL, NULL, NULL);
        }
        else
        if (!strcmp("TanD", newParmName))
        {
            decl tmpParmList = meutil_get_dataitem_parmlist( TRUE, "TAND", FALSE);
            tmpStr = merul_extract_parm(tmpParmList, "TAND", NULL, NULL, NULL);
        }

if(tmpStr)
          {
              newParmH    = repla(newParmH, tmpStr, MERUL_PARM_VALUE);
              newParmList = repla(newParmList, newParmH, j);
          }
      }
  }
  return newParmList;
}
*/
```

This example illustrates most features of a typical ParmRule AEL function.

When this AEL function runs, all parameters with the same name in both the MDS and the ADS components have already been copied to the ADS component. This AEL function only needs to modify parameters that need special handling.

The following events occur when the function is run:

- On the fourth and fifth lines, it gets the oldParmList and newParmList for the MDS and ADS components.
- The next line sets "i" to the length of the newParmList.
- On line 11 a "for" loop is started that loops through the parameters from the newParmList.
- The variable "newParmName" is assigned the name of the next parameter of the new component.
- A series of conditional "if" statements looks for specific parameter names and does special handling for these parameters. Within the "if" statement that

selects "Cond" the function merul_extract_parm (line 23) is used to get the value of the "RHO" parameter from the old component.

- The third and forth parameters of merul_extract_parm are DataItemName and DataItemParmName, as described previously.
- The term "tmpStr" is a string containing the old parameter.
- The term rhoH is the numeric value of the parameter including applying the unit associated with this parameter.
- The lines starting at line 30 compute a string value for tmpStr based on the numeric value of the RHO parameter. Near the end of the function, if tmpStr had a value, a newParmH is created for tmpStr and it is used to update the parameter list for the new component.

Mapping Component Pin Changes

If the pin connections of the old component and the new component are not the same, you can create a PinRule to interchange the pin numbers using the following sub-fields:

SrcPinNum,SrcPinName,DestPinNum,DestPinName,;

where

SrcPinNum	Old pin number (optional)
SrcPinName	Old pin name (not used)
DestPinNum	Destination pin number
DestPinName	Destination pin name (not used)

Note that when any of the above fields is not used or is optional, the comma (,) field separator must still be used. A semicolon (;) must be used at the end, and multiple PinRules can be appended together to map several pins, with each PinRule being separated by a semicolon (;).

Note If DestPinNum is specified but SrcPinNum is not, the DestPinNum of the new instance will be grounded.

Example:

```
Neg1|Deembed1|  |  | , ,2, ,;|  |FALSE| |
```

In this example, pin 2 of the ADS Deembed1 component would be grounded.

Example:

```
PCCROS|PCCROS|  |  |2, ,4, ,;3, ,2, ,;4, ,3, ,;|  |FALSE| |
```

This example contain 3 PinRules. MDS pin 2 is mapped to ADS pin 4, MDS pin 3 is mapped to ADS pin 2, and MDS pin 4 maps to ADS pin 3.

Mapping to a Component Based on Parameter Specifics

If you want to map an MDS component to an ADS component based on an MDS component parameter, you can specify an AEL function in the NameRule field.

Example:

```
defun merullib_name_P2D()
{
    decl oldParmList = car(car(arg_list()));
    if (listlen(oldParmList) == 1)
    {
        return "DataAccessComponent";
    }
    else
    {
        return "AmplifierP2D";
    }
}
```

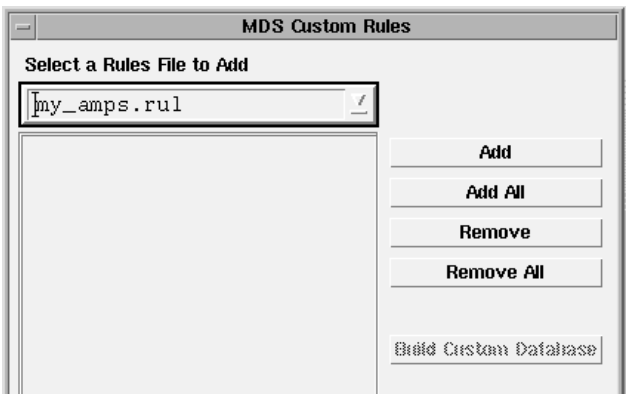
This example looks at the number of parameters of the MDS component. If there is one parameter, then the "DataAccessComponent" name is returned, otherwise the name "AmplifierP2D" is returned.

Creating Customized Rules Files

To use custom rules, you need to create one or more rules files and place them in your *$HOME/hpeesof/config* (UNIX) or *%HOME%\hpeesof\config* (PC) directory. These files must end with the *.rul* suffix and be compiled into a rules database file (*meruls_mds_custom.db*).

To compile your custom rules file(s)g:

1. Launch the translator as described in "Translating Designs into ADS" on page 2-3.
2. Click **Custom Rules**. The MDS Custom Rules dialog box appears. If any *.rul* files are found in the local *config* directory, they will appear on the drop-down list.



3. Select the desired rules file from the drop-down list and click **Add**. Alternatively, click **Add All** to merge all rules files into one custom database.

Hint To modify the entries in the list box, highlight a single entry and click *Remove* or *Remove All* to clear the box and start over.

---

4. Once the list box contains the desired filename(s), click **Build Custom Database**. The status panel at the bottom of the dialog box displays messages reporting the progress. When the process is complete, a message to the effect of *The rules database has been successfully built* appears.
5. Select the option **Use Custom Database** to use it for the upcoming translation and click **OK** to proceed with the translation.

Appendix D: Batch Mode Translation

This appendix describes how to set up and perform MDS batch translations from the command line.

Note Translating MDS designs from a command line requires knowledge of how to set environment variables and work in a DOS window or UNIX shell. Therefore, this task should only be attempted by advanced users.

---

hpeesofme Setup

Prior to use, the *hpeesofme* program must be properly configured.

Windows

To setup the *hpeesofme* program on a PC, follow the steps listed below:

1. Open an MS DOS shell.
2. Set the $HPEESOF_DIR environment variable to your ADS installation directory. For example

   set HPEESOF_DIR=<ADS_install_dir>

3. Set your PATH environment variable to include the *$HPEESOF_DIR\bin* directory. For example

   set path=C:\ADS2001\bin;%path%

4. Set the appropriate library path for your operating system. For example

   set shlib_path=$shlib_path:$HPEESOF_DIR\lib\win

UNIX

The *hpeesofme* program is setup on UNIX platforms as follows:

1. Set your $HPEESOF_DIR environment variable to your ADS installation directory. For example, if you are using the Korn shell enter

   export HPEESOF_DIR=<ADS_install_dir>

2. Set your PATH environment variable to include the *$HPEESOF_DIR/bin* directory. For example, if you are using the Korn shell enter

   export PATH=$HPEESOF_DIR/bin:$PATH

3. Set the appropriate library path for your operating system. For HP-UX operating systems (i.e. hpux10 or hpux11), enter the following:

   SHLIB_PATH=$SHLIB_PATH:$HPEESOF_DIR/lib/hpux10

   or

   SHLIB_PATH=$SHLIB_PATH:$HPEESOF_DIR/lib/hpux11

   For SUN operating systems (i.e. sun4 or sun55), enter

   LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HPEESOF_DIR/lib/sun4

   or

   LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HPEESOF_DIR/lib/sun55

   For AIX operating systems (i.e., aix4), enter

   LIBPATH=$LIBPATH:$HPEESOF_DIR/lib/aix4

Executing hpeesofme

The MDS translator is started using the following syntax:

```
hpeesofemx hpeesofme -me -nw -mep <projectDef> -mem <modelType> -mec
<customRules> -env <envFileName>
```

where:

-me	Forces the translation to start after boot-up (required)
-nw	Invokes the program in the non-visual mode (required)
-mep <projectDef>	Specifies the project to migrate (required)
-mem <modelType>	Specifies the distributed models type (optional)

-mec <customRules>	Specifies the custom rules database with path and extension (optional)
-env <envFileName>	Specifies the environment file to be used (usually *de_sim*) (required)

Additional Option Information

-mep

This option has the following syntax:

`"SrcPath|DefaultDsnName|Option|DestPath|UnitPref"`

where:

SrcPath	IFF file path
DefaultDsnName	Not applicable for MDS translator ("NA")
Option	Remove existing ADS project flag (0: remove, 1: keep existing project)
DestPath	Destination project path
UnitPref	Unit preference (0: mil, 1: millimeter, 2: micron)

-mem

This option allows you to specify the distributed components used during MDS translation. The acceptable values are:

"0" ADS standard distributed components

"1" MDS equivalent distributed components

Example

The following is an example of a Perl script that is capable running multiple translations in batch mode:

```
# a script to run the mds translator from the command line
#
@designs = (“file1”, “file2”);
$iffdir = “c:/users/myname/iffFiles”;
$adsdir = “c:/users/myname/adsmigration/mdsmigrated”;
$customRules = “c:/users/myname/hpeesof/config/meruls_mds_custom.db”;
foreach (@designs)
{
print “\n\n********* translate $_ **********\n\n”;
$mdsfile = “$iffdir/$_”.’.iff’;
$adsproj = “$adsdir/$_”.’.prj’;
system(“hpeesofemx hpeesofme -nw -me -mep \”$mdsfile|defaults|0|$adsproj\”
-mec \”$customRules\” -mem \”1\” -env de_sim”);
system(“egrep -e ERROR -e WARNING -e Error -e Warning
$adsproj/$_/me_err.log”);
}
```

Appendix E: Translation Issues

This appendix describes the MDS to ADS translation issues that have been identified to date and provides workarounds for several of them.

Known Issues

The various MDS to ADS translation issues are summarized in Table E-1.

Table E-1. MDS to ADS translation issues

Item	Notes
AC Control ACFORM	Not translated
Datasets	Not translated. Refer to "Importing MDS Datasets" on page 2-5.
Density & Distribution Controls	Not translated. Refer to "Density and Distribution Controls" on page E-4.
Design of Experiments Control (DOE)	Not translated
DATASETVARIABLE, POLARDATASETVARIABLE	Translated to a DataAccessComponent(DAC). Refer to "Importing MDS Datasets" on page 2-5.
IBIS Models	Not supported in ADS
MDS components using dBm scale factor	Many MDS components use dBm as a scale factor, which ADS does not recognize. Refer to "Component Power Parameters" on page E-2.
MDS designs using Symbolically Defined Devices (SDD) with complex variables	MDS and ADS use different SDDs. Refer to "Symbolically Defined Devices" on page E-5.
MDS Momentum	Frequency dependent material parameter specification using lists, coaxial ports, and partial edge ports no longer supported in ADS Momentum. MDS Momentum ports, substrates, mesh and simulation must be setup within the ADS use model following migration. Refer to "Momentum" on page E-4.

Table E-1. MDS to ADS translation issues (continued)

Item	Notes
Monte Carlo Controls	Not translated. Refer to "Monte Carlo Controls" on page E-4.
MSTEE models	Microstrip tee models are implemented differently in MDS and ADS. Refer to "Microstrip Tee Models" on page E-4.
step( ) function	MDS and ADS handle the step( ) function differently when the argument is an integer equal to 0. Refer to "step( ) Function" on page E-5.
System Waveform Sources	Not translated and not supported in ADS
Layout pages (1)	ADS does not support multiple units within the hierarchy of a design. All translated designs will be in the same unit.
Layout pages (2)	ADS metal layer cannot be assigned. Refer to the section """Specifying the Metal Layer for Circuit Components in Layout Pages" on page 2-6.

Component Power Parameters

Many MDS power parameter values were specified using dBm as a scale factor. However, nearly all ADS component power parameters are defined in Watts and ADS does not recognize dBm as a scale factor. In most cases, when an MDS value is specified in dBm, it needs to be mapped to ADS in Watts using the dbmtow(x) function, where x is the value in dBm. In the rare cases in which the ADS power parameter is defined in dBm, the dBm scale factor should be stripped from the MDS value when it is used in ADS. You can also change dBm to "_dBm", which is interpreted by the simulator as a scale factor of 1.

Datasets

The MDS translator does not translate datasets. To work around this problem, do the following:

1. Translate the MDS designs.
2. Use MDS to write the dataset as a CITI file.

3. Open the translated project in ADS and import the CITI file using the Instrument Server. A new dataset is created under <*project*>/*data*.

Density and Distribution Controls

The procedure to model parameters using statistical distributions is as follows:

1. Select the component to model. Double-click on it to access its associated dialog box.
2. From the dialog box, highlight the parameter that you want to vary in the Select Parameters dialog box.
3. Click the **Optimization/Statistic Setup** button.

For more information, refer to the Tuning, Optimization, and Statistical Design manual.

Microstrip Tee Models

The MDS and ADS microstrip tee models are implemented differently. The reference plane is in the center for MDS (MSTEE) and on the edges for ADS (MSTEE). This can lead to significantly different simulation results if you use ADS microstrip models in a converted ADS design.

Momentum

The following MDS Momentum features are no longer supported by ADS Momentum:

- Frequency dependent material parameter specification using lists
- Coaxial ports
- Partial edge ports

The migration tool allows the migration of layouts. However ports, substrates, mesh and simulation have to be setup within the ADS use model.

Monte Carlo Controls

The Monte Carlo elements (MonteCarlo and MonteCarloForm) are not translated into ADS. Use the component in the Optim/Stat/Yield palette to specify a yield analysis based on the Monte Carlo method.

step( ) Function

MDS and ADS handle the step( ) function differently when the argument is an integer equal to 0. In MDS, step( ) returns different results for integer and real arguments. In ADS these arguments return the same value. The differences in how MDS and ADS handle this function are summarized below:

MDS	ADS
step(int (0)) = 0	step(int (0)) = 0.5
step(real (0)) = 0.5	step(real (0)) = 0.5

Symbolically Defined Devices

MDS designs that use SDD with complex variables can be translated, but will fail to simulate because the SDD model in ADS 2001 does not support complex variables. The suggested workaround is to translate the design, open it in ADS 2001, and reformulate the SDD equations using real variables prior to simulation.

Unsupported Components

If an MDS component is no longer used by the Advanced Design System, it is deactivated in the imported copy of the design. After translation, replace the unsupported component with an ADS component that has similar functionality. If it is a user-defined component, and the same component is available in the Advanced Design System, it can be activated.

Index

U

V