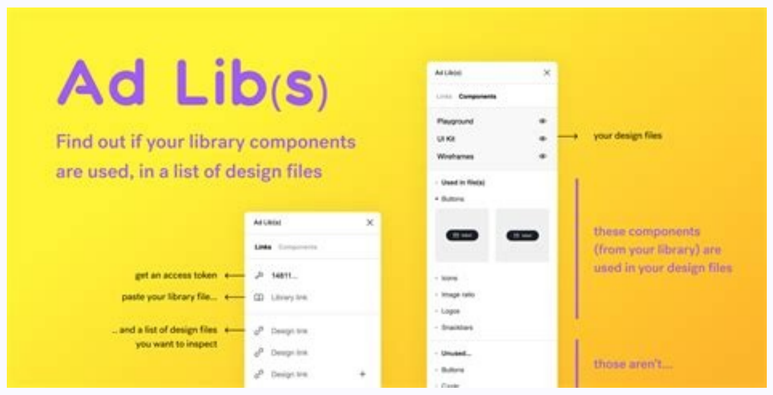
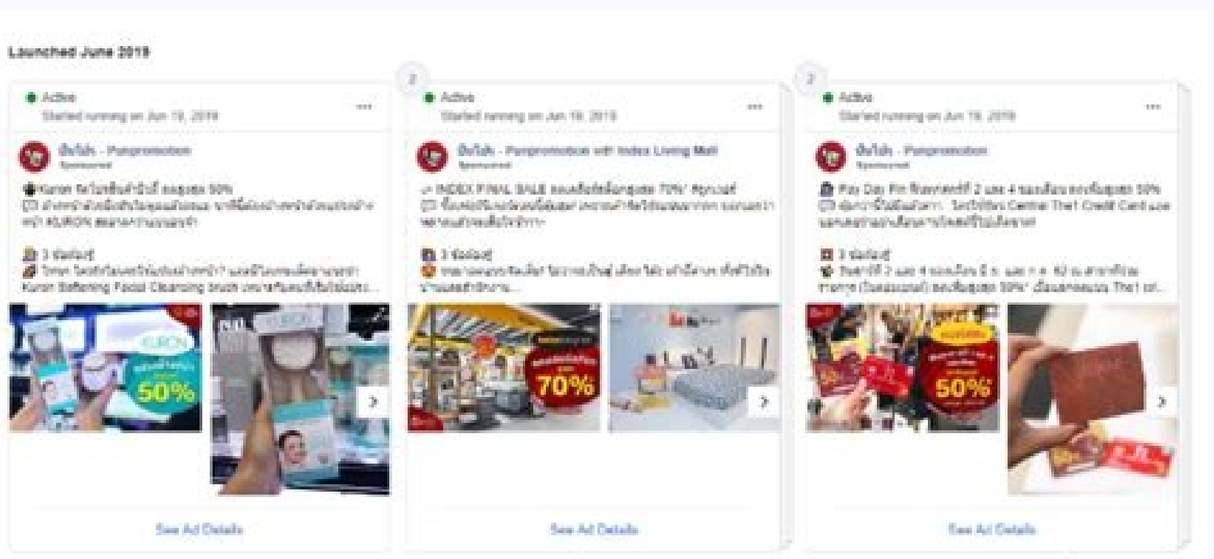
Ads component library

I'm not robot!





Ads component library download. Murata ads component library. Ads add component library. Whats ad lib.

This section offers a detailed understanding of all the built-in and custom model components available in Library Builder. Built-in Components and Models Single and Sequence Components Library Builder presents component types two ways. Either as a a single component, having a single set of parameter values or a sequence of components which enable you change the given values for a component sequentially. To create either type of component you simply specify the Component type using the Component Parameters tab of the Component dialog. Single Components As illustrated above, a single component has a single set of parameters R , Temp , Trise, etc. This component type provides a fixed value for the component when no variation in values is required. For more information on setting up single component parameters, refer to Setting Component Parameters. Sequence of Components There are situations when different sets of parameters are all associated with the same type of component. For example, a set of resistors may have five different sets of R values. You may wish to group the resistors into a single component, so you can select a particular set of parameters. This is achieved by creating a Sequence of Components. In the figure below, a sequence of components has been created using a single resistor. In the sequence of component view, the parameters are displayed in a spreadsheet, where each row is a specific set of parameters denoted by a descriptive name. For instance, the first set (Row 1) has the name as 50 Ohm @ 21C to denote that it is a 50 Ohm resistor at 21C temperature. In this view, a new set of parameters can be created by clicking New Row . This will create a new row with the default values in it. Similarly, a row can be deleted by selecting a row and selecting Delete Row. To add a new row containing a set of values other than default, select a row by clicking its number on the left column. Click Copy Row and then Paste Row. This adds a new row having the values as the row just copied. To add new parameters, click New Column . This will

open a form to add information about the new parameter. Once the form is filled, click OK and the new column will be added to the end of the spreadsheet. Built-in Components Library Builder comes with a number of built-in components. These include resistors (R), capacitors (C), inductors (L), and 1-Port to 99-Port S-parameter File (SNP). The following tables list parameters and default values for these components. For more information, refer to Introduction to Circuit Components. R (Resistor) Parameters Name Description Unit Default R resistance mOhm, Ohm, kOhm, MOhm, GOhm 50 Ohm Temp temperature °C Trise temperature rise above ambient °C 0 Tnom nominal temperature °C 25 TC1 linear temperature coefficient 1/°C TC2 quadratic temperature coefficient 1/°C2 Noise resistor thermal noise option: yes=enable; no=disable yes wPmax maximum power dissipation (warning) pW, nW, uW, mW, W, kW, dbm wImax maximum current (warning) fA, pA, nA, uA, mA, A Model name of a resistor model to use Width physical width for use with a model um, mm, cm, meter, mil, in model Length physical length for use with a model um, mm, cm, meter, mil, in model _M number of resistors in parallel 1 Notes/Equations The resistor value can be made a function of temperature by setting Tnom and TC1 or TC2 or both. Tnom specifies the nominal temperature at which R is given. Tnom defaults to 25°C. If Temp≠Tnom, then the simulated resistance value is given by: R′ = R × [1 + TC1 (Temp − Tnom) + TC2 (Temp − Tnom)2] If Temp is not explicitly specified, it defaults to the global temperature specified in the options item. The resistor generates thermal noise: = 4kT/R Noise generation can be disabled by setting Noise=no. wPmax and wImax are used by the overload alert feature. They set limits on the maximum instantaneous power dissipated by the resistor and maximum current through the resistor. If these limits are specified, the simulator will issue a warning the first time they are exceeded during a dc, harmonic balance or transient simulation. Simulation results are not affected by this parameter. For a transient simulation, the resistance can vary with time. The resistance value should be assigned an expression that is a function of the reserved variable time, which is the simulation time in seconds. If a model name is given, then values that are not specified on the resistor instance are taken from the model values. Typical values that can be defaulted are resistance, length and width, nominal temperature, temperature coefficients, and overload alert parameters. If a model is used, the resistance value to be simulated (before temperature scaling is applied) is calculated as: _M is used to represent the number of resistors in parallel and defaults to 1. M cannot be zero. If a resistor model is used, an optional scaling parameter Scale can also be defined on the model; it defaults to 1. The effective resistance that will be simulated is R × Scale/M. The following table lists the DC operating point parameters that can be sent to the dataset. Name Description Units I Current A Power DC power dissipated W R Resistance Ohms V Resistor voltage V C (Capacitor) Parameters Name Description Unit Default C capacitance fF, pF, nF, uF, mF 1.0 pF Temp temperature °C Trise temperature rise over ambient °C 0 Tnom nominal temperature °C TC1 linear temperature coefficient 1/°C TC2 quadratic temperature coefficient 1/°C2 wBV breakdown voltage warning fV, pV, nV, uV, mV, V InitCond initial condition voltages for transient analysis Model name of a capacitor model to use Width physical width for use with a model um, mm, cm, meter, mil, in Length physical length for use with a model um, mm, cm, meter, mil, in _M number of capacitors in parallel 1 Notes/Equations The capacitor value can be made a function of temperature by setting Tnom and either TC1 or TC2 or both. Tnom specifies the nominal temperature at which C is given. Tnom defaults to 25°C. If Temp≠Tnom, then the simulated capacitance value is given by: If Temp is not explicitly specified, it defaults to the global temperature specified in the options item. wBV is used by the overload alert feature. It sets a limit on the maximum voltage across the capacitor. If this limit is specified, the simulator will issue a warning the first time it is exceeded during a dc, harmonic balance or transient simulation. Simulation results are not affected by this parameter. If a model name is given, then values that are not specified on the capacitor instance are taken from the model values. Typical values that can be defaulted are capacitance, length and width, nominal temperature, temperature coefficients, and overload alert parameters. If a model is used, the capacitance value to be simulated (before temperature scaling is applied) is calculated as: _M is used to represent the number of capacitors in parallel and defaults to 1. If a capacitor model is used, an optional scaling parameter Scale can also be defined on the model; it defaults to 1. The effective capacitance that will be simulated is C × Scale × M. When InitCond is explicitly specified, the check-box Use user-specified initial conditions must be turned on in the Convergence tab of the Tran transient simulation controller for the parameter setting to take effect. The following table lists the DC operating point parameters that can be sent to the dataset. Name Description Units C Capacitance F L (Inductor) Parameters Name Description Unit Default L inductance fH, pH, nH, uH, mH 1.0 nH R series resistance mOhm, Ohm, kOhm, MOhm, GOhm Temp nominal temperature °C Trise temperature rise over ambient °C 0 Tnom nominal temperature °C TC1 linear temperature coefficient 1/°C TC2 quadratic temperature coefficient 1/°C2 InitCond transient analysis initial condition current Noise noise generation option: yes=1, no=0 yes Model model instance name _M number of inductors in parallel 1 Notes/Equations The inductor value can be made a function of temperature by setting Tnom and either TC1 or TC2 or both. Tnom specifies the nominal temperature at which L is given. Tnom defaults to 25°C. If Temp ≠ Tnom , then the simulated inductance value is given by: L′ = L × [1 + TC1 (Temp - Tnom) + TC2 (Temp - Tnom)2] The resistance, if specified, is not temperature scaled. If Temp is not explicitly specified, it defaults to the global temperature specified in the options item. If the series resistance is specified, it always generates thermal noise: = 4kT/R If a model name is given, then values that are not specified on the inductor instance are taken from the model values. Typical values that can be defaulted are the inductance, series resistance, nominal temperature and temperature coefficients. When InitCond is explicitly specified, the check-box Use user-specified initial conditions must be turned on in the Convergence tab of the Tran transient simulation controller for the parameter setting to take effect. _M is used to represent the number of inductors in parallel and defaults to 1. M cannot be zero. If an inductor model is used, an optional scaling parameter Scale can also be defined on the model; it defaults to 1. The effective inductance that will be simulated is L × Scale/M; the effective resistance is R × Scale/M. The following table lists the DC operating point parameters that can be sent to the dataset. Name Description Units I Current A L Inductance H SNP (1-Port to 99-Port S-parameter File) This component can be used to describe an N-port component using its N-port S-, Y-, Z-parameter data files. Parameters File = name of data file containing N-port S-, Y-, or Z-parameters for this component. The file extension and directory path are optional. Default extension is .s1p and the default directory is /data where is your current project directory. Type = file type: Touchstone Notes/Equations For information on data file formats, refer to Working with Data Files in Using Circuit Simulators. The 2-port S-parameters are assumed to be measured with pin 1 as the input, pin 2 as the output. The number of terminals increases sequentially from 1 to 99, and is equal to the number of ports of the component. The S, Y, Z, and N matrix measurements are allowed for up to 99-port networks. In addition, single measurements are applicable: IJ, for example (S(29,28)) VSWR, for example (S(29,29)) For N 20, these components primarily support electromagnetic simulation results of circuits with a large number of ports, such as antenna feed networks. If the component temperature Temp is less than -273oC, then the component does not generate any noise. If the S-parameters describe a passive device, then Temp and Twiss's theorem are used to calculate its noise performance. If the S-parameters describe an active device, no noise is generated. For time-domain analysis, the frequency-domain S-parameters are used. This component has no default artwork associated with it. SnP Component (n>99) Use ADS Design Kit components to create this component. Parameters File = name of data file containing #-port S-, Y-, or Z-parameters for this component. The file extension and directory path are optional. Default extension is .s#p and the default directory is /data where is your current project directory. Type = file type: Touchstone, Dataset, CITIfile Block = (for Type=Dataset) name of S-parameter data block InterpMode = interpolation mode: Linear, Cubic Spline, Cubic, Value Lookup InterpDom = interpolation domain: Data Based (polar for S and rectangular for Y and Z), Rectangular, Polar, DB ExtrapMode = extrapolation mode: Interpolation Mode, Constant Extrapolation Temp = physical temperature, in °C ImpNoncausalLength = non-causal function impulse response order (value type: integer) ImpMode = convolution mode (value type: integer) ImpMaxFreq = maximum frequency to which device is evaluated, in hertz ImpDeltaFreq = sample spacing in frequency, in hertz ImpMaxOrder = maximum impulse response order (value type: integer) ImpWindow = smoothing window (value type: integer) ImpRelTol = relative impulse response truncation factor ImpAbsTol = absolute impulse response truncation factor Notes/Equations SnP component (n>99) is used to create SnP components with a port number larger than 99. They cannot be selected from the component palette; they are accessed by typing the appropriate name (such as S123P) into the field above the viewing area and press Enter , then moving the cursor to the viewing area to place the item SnP(n>99) component is not a built-in component. You must create this component by following the procedure given in the Example section below. Required Files: The following files are required when creating an SnP component: /S*P (netlist definition) /networks/SYM_S*P.dsn (symbol definition) /networks/S*p.ael (parameter definition) Tools used to create an SnP component makeSnP A Ksh script that creates the netlist definition S*P and the parameter definition S*P.ael. Usage: makeSnP num_of_port generate_snp_symbol This is an AEL function that generates the symbol definition SYM_S*P.dsn. This must be invoked from ADS. Usage: generate_snp_symbol(numPort) SnP (n>99) Component Example The following steps are used to create an SnP (n>99) component: Edit the cygwin.bat file in $HPEESOF_DIR\tools\bin and add the following to SET PATH=: c:/ADS2008/tools/bin;%PATH% Change the first line in the makeSnP file located in $HPEESOF_DIR\bin from: #! /bin/ksh to #! /ADS2008/tools/bin/ksh Double click cygwin.bat to open a bash command shell. Type ksh. This opens a kshell. Type export HPEESOF_DIR=/cygwindrive/c/ADS2008. Type ./makeSnP . This creates the netlist definition (e.g., S123P) and the parameter definition (e.g., S123P.ael) in the current directory. Copy the netlist definition to directory. Copy the parameter definition to /networks directory. Open the project in ADS. From the Main window select Tools > Command Line and type: generate_snp_symbol(123) where: 123 is the number of the Port. This generates the SYM_S123P.dsn file in /network and loads the device into ADS. Type the name of the component (e.g., S123P) in the schematic component history list to place it into the Schematic window. If the component symbol (e.g., SYM_123.dsn) already exists, or if it was created before performing steps 1 through 3, you must the load the SnP component definition into ADS manually. Once all the required files are in place, do one of the following: Reopen the project (ADS will load all designs that are defined in the /network directory. Load the device definition manually: From the Main window select Tools > Command Line and type: load(strcat(getcwd(),"/networks/S123P")) where: S123P is the name of the SnP component. Creating ADS Design Kit Components Using ADS Design Kit components enables you to include any type of component in the library. This is accomplished by creating the required models using the ADS Schematic Design window. There are two main steps required for the creation of libraries with such components: Creating the required files for the component. Adding the component in the library In the following sections we will explain how to create and edit the ADS design file, AEL file and netlist file. Creating the Required Files for a Component The files required for creating a component with ADS custom model depends on whether or not you want to encode the model. Creating a component without encoding of model netlist You can create this component using one of two file sets. For the first case the required files are: Component definition AEL file Design file Data file, mainly with an S-parameter component Using the preceding files does not require significant editing and is a simpler method for creating a component. Once these files have been created, you can create a component in Library Builder and browse for these files to associate them with the component. For the second case the files required are: Component definition AEL file Netlist file for defining the component model Data file, mainly with an S-parameter component Symbol It is possible that you will decide to use unmodified built-in ADS symbols in your design kit. In this case, the symbols do not need to be copied to your design kit circuit/symbols directory. You can simply reference the symbol by name in your create_item() command. No definition is needed. Using this method, you will need to edit the AEL and netlist files. Creating a Design File The following steps are needed to create the component definition AEL file: Start ADS. Create a new project. Select File > New Project from the ADS Main menu. Create a new design. Select File > New Design from the ADS Main menu. Open a new Schematic window. Be sure you give a unique name to your design. Create your custom component and terminate it with ports. Select File > Save Design. This design file will be saved in the current working project's network directory by the name of your design with extension .dsn. Create the AEL file. From the Main menu of the ADS Schematic window, select File > Design Parameters ... A new dialog box for the Design Parameters opens. Use the General and Parameter tabs to enter specific design information per your design requirements. Click Save AEL file and OK to create the .ael file and dismiss the dialog box. The AEL file is saved in the current working project's network directory by the name of your design with extension .ael. For more information about Design Parameters, refer to, Creating Elements in Schematic Capture & Layout. Adding the Component to a Library After preparing all the requisite files, you can package them into to a design kit using Library Builder. Start Library Builder. Select Create New Library from the Welcome dialog. Enter the required information, giving the new library a unique name and select Apply. Create a new component using the same component name used for the design you created in ADS. Select the Model type, ADS Custom. Browse the Design field for the design (.dsn) file you created in ADS. Browse the AEL Files field for the .ael file you created the ADS. Select Add. Creating a Component Without Model Netlist Encoding Once you have created an AEL file, as shown in the second case, you must edit it in order to attach the associate the ADS netlist model to the component. Following is a modified AEL file (save as " custom_component_res2_netlist.ael ") used with the ADS netlist model: decl fileName_custom_component_res2_netlist=""; fileName_custom_component_res2_netlist = strcat\{}} MyCustomLibrary_withNetcomp _CIRCUIT_MODEL_DIR, "}} mycustomNetlist.net"); decl MYCmpNetlistFmt_custom_component_res2_netlist = strcat\("#include \"", fileName_custom_component_res2_netlist, "\" %d:%t %# %44?0%:%31?%C%:_net%c%;%;%e%r%b%8?%29?%:%30?%p %:%k%?[%1i]%;=%p %;%;%e%e"\); set_simulator_type\(1\); create_item\("custom_component_res2_netlist",//Component name "custom_component_res2", //Description "res", //Instance Prefix NULL, //attribute -1, //priority NULL, //iconName "Component Parameters", //dialogCode "", //dialogData MYCmpNetlistFmt_custom_component_res2_netlist, //netlistFormat "custom_component_res2", //netlistData ComponentAnnotFmt, //displayFormat "SYM_2Port", //symbolName 0, //artworkType "", //artworkData 0 //extraAttribute \); set_design_type\(1\); Where: MyCustomLibrary_withNetcomp is the Library name. mycustomNetlist.net is the netlist name. "//" is the commenting of the remaining line. Creating the Netlist File Open a new Schematic window from the ADS Main menu and click the Library Browser icon. Select the project you created above and double click the custom component to select it and place it on the schematic. In the Main menu of the ADS Schematic window, click DynamicLink > Top-level Design Netlist. This opens the netlist text file. Delete the first and the last line in the netlist text file: Options ResourceUsage=yes UseNutmegFormat=no topDesignname="untitled" and custom_component_res2:X1 net7 net8 L=1.0 H R=50 Ohm Save this file in the appropriate location with extension .net. If you don't select the location, it will be saved in the project directory (i.e., save the file as mycustomNetlist.net). Enclose the file name in double quotes or a .txt will be appended to the name. Creating a Component with Model Netlist Encoding You need to have the following files for the ADS design kit component: AEL file ADS netlist and/or S-parameter data file Symbol You have to create an AEL file as explained previously and edit it in order to attach the encoded ADS netlist model to the component. Following is a modified AEL file used with the encoded netlist: set_simulator_type\(1\); create_item\("custom_component_res2", //Component name "custom_component_res2", //Description "res", //Instance Prefix NULL, //attribute -1, //priority NULL, //iconName "Component Parameters", //dialogCode "", //dialogData "#uselib \"MyCustomLibrary\", \ "mycustomNetlist\" {{%d:%t %# %44?0%:%31?%C%:_net%c%;%;%e %b%r%8?%29?%:%30?%p %:%k%?\[%1i\]%;=%p %;%;%e%e", //netlistFormat "custom_component_res2", //netlistData ComponentAnnotFmt, //displayFormat "SYM_2Port", //symbolName 0, //artworkType "", //artworkData 0 //extraAttribute \); set_design_type\(1\); Following are the changes in this AEL file: Attribute Changed from: " 16 " to "NULL". Netlist format Changed the netlist format from: "%43?global %;%d:%t %# %44?0%:%31?%C%:_net%c%;%;%e %b%r%8?%29?%:%30?%p %:%k%?[%1i]%;=%p %;%;%;%e%e" to "#uselib \"MyCustomLibrary\", \"mycustomNetlist\"%d:%t %# %44?0%:%31?%C%:_net%c%;%;%e %b%r%8?%29?%:%30?%p %:%k%?[%1i]%;=%p %;%;%;%e%e;" to ComponentAnnotFmt. ADS netlist: Create the ADS netlist as explained previously and delete the first and last line of the netlist file: Options ResourceUsage=yes UseNutmegFormat=no TopDesignName="untitled1" and custom_component_res2:X1 _net7 _net8 L=1.0 H R=50 Ohm. Creating a Netlist Based Component The Netlist Based component is a more automated form of the ADS Design Kit component. For this type of component, the editing the AEL and netlist files is done by the Library Builder. However, you do need to generate these files from the ADS design file. This is in the similar to the method used for the ADS Design Kit component. The following sections explain how to create a Netlist Based component. To create a netlist based component in ADS, you must create creating an AEL file and a netlist file from the ADS design file. To Create the AEL file: In the ADS schematic, create a design terminated with ports. Save the design file with a suitable name. Note that an AEL is also created in the project's networks sub-directory with the same name as design file. To Create the Netlist file. Open a new ADS schematic window and place an instance of the created design into the schematic. You can place the design instance in the schematic either by typing the design name in the Component History or select it from the Library Browser. After placing the design instance in the ADS schematic, click DynamicLink in the main menu and then click Top-level Design Netlist. You will see a text editor opened with the netlist of the design file. Save this netlist with a suitable name and a suitable file extension(e.g., .net, .ads, .ckt, etc). Tasks in Library Builder These tasks involve importing the AEL file and netlist. In the Library Builder, open an existing library, or create a new library. To create a netlist based component perform the following tasks: Select the Model Type as Netlist Based. Click on the Import ADS AEL... button and select the AEL file you created in Tasks in the ADS Your should note the following: The component name and description is filled automatically. If your design contained Synchronized type of artwork then it will be converted to the fixed artwork type and a message pops-up, informing you about the same. In this case, you need to write the artwork name and attach the artwork design file. If you don't want to attach any artwork, then make the artwork type as No Artwork. Symbol field is filled up automatically. If the symbol is pre-defined in ADS then you don't need to do anything, otherwise you can browse the ADS design file to act as a symbol. Also, you may use any suitable custom symbol with appropriate port mapping. If the design has the parameters defined with the VAR component, then those parameters will be displayed in the Component Parameters page. Browse the netlist file which contains the model information of the component. After browsing the netlist file, notice that all the models defined in the netlist are displayed in the Model drop down list. If there was any data file in the design then data file name will also be displayed. Port Mapping Symbols for SNP Components When using SNP components you must provide proper port mapping for the symbols to generate correct simulation results. Port mapping solves two potential problems. The first is, for an active device where gain is directional, such as a BJT or FET, you need to make sure the S21 of the data agrees with the pin numbers that are assigned to the symbol. If you do not, you may have to drive the collector in the schematic when the measurement itself was made by driving the base. The second problem is that the SNP component for an N-port actually has N+1 ports, with the additional pin being the ground reference. When assigning a symbol to the SNP component, you need to determine whether or not your going to provide access to the ground reference, or assign it to ground yourself. SYM_S2P component SNP components can use Library Builder custom symbols or a pre-defined ADS symbol, so long as appropriate port mapping is used. This adds flexibility when defining symbols for use with SNP components. For example, an S2P component has 3 pins, in which pin1 is the input, pin2 the output and the pin3 is the reference (ground). You can use any of the following symbols for the S2P component if you apply the appropriate port mapping: Symbol Port Mapping Pin Number Port Typ e SYM_S2P 1 2 3 1 2 3 SYM_2Port, SYM_C, SYM_L, SYM_R and SYM_DIODE 1 2 3 1 2 0 SYM_BJT_NPN, SYM_BJT_PNP, SYM_JFET_PFET, SYM_JFET_NFET etc... 1 2 3 2 1 3 SYM_LB_BJT_NPN_CE, SYM_LB_BJT_NPN_CC etc... 1 2 3 1 2 0 For more details on the port mapping, refer to Port Mapping Custom Symbols for SNP Components and Port Mapping of Pre-defined ADS Symbols for SNP Components. Port Mapping Symbols for Netlist Based Components Port mapping for Netlist based components is similar to port mapping for SNP components. However, it should be noted that an N-port netlist based component has N+1. Port Mapping Custom Symbols for SNP Components In order to use a custom symbol for a component with S-parameter data file, you must map the ports using Library Builder. Failure to do this will result in inaccurate simulation results. Custom symbols are provided with the Library Builder. You can select the appropriate symbol from the custom symbol directory located at: $HPEESOF_DIR/circuit/LBCustomSymbols Port mapping for these symbols is described in the following section. Pin Diode Diode with grounded N-terminal SYM_LB_DIODE_NG.dsn Port Mapping Pin Number Port Type 1 1 2 Ground Diode with grounded P-terminal SYM_LB_DIODE_PG.dsn Port Mapping Pin Number Port Type 1 Ground 2 1 2-Pin BJTs BJT NPN with common (grounded) emitter SYM_LB_BJT_NPN_CE.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground BJT NPN with common (grounded) Collector SYM_LB_BJT_NPN_CC.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground BJT NPN with common (grounded) Base SYM_LB_BJT_NPN_CB.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground BJT PNP with common (grounded) Emitter SYS_LB_BJT_PNP_CE.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground BJT PNP with common (grounded) Collector SYM_LB_BJT_PNP_CC.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground BJT PNP with common (grounded) Base SYM_LB_BJT_PNP_CB.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground 3-Pin BJTs BJT NPN SYM_LB_BJT_NPN_CE.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground 2-Pin FETs N-Channel FET with common (grounded) Source or Drain SYM_LB_NFET_CS.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground N-Channel FET with common (grounded) Gate SYM_LB_NFET_CG.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground P-Channel FET with common (grounded) Source or Drain SYM_LB_PFET_CS.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground P-Channel FET with common (grounded) Gate SYM_LB_PFET_CG.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 Ground Using S2P Symbols for 2 Port Devices SYM_S2P When using 3 pin devices to represent a device that has 2 port measured data, such as a capacitor or inductor, you must map the ports using Library Builder. Failure to do this will result in inaccurate simulation results. SYM_S2P.dsn Port Mapping Pin Number Port Type 1 1 2 2 3 3 Port Mapping of Pre-defined ADS Symbols for SNP Components The pre-defined ADS symbols can be found at: $HPEESOF_DIR/circuit/symbols or $HPEESOF_DIR/de/symbols 3-Pin BJTs BJT NPN In order to use a 3-pin pre-defined ADS BJT symbol and FET symbol for a component with 2-port S-parameter data file, you must map the ports using Library Builder. Failure to do this will result in inaccurate simulation results. Pre-defined BJT NPN Symbol in ADS (SYM_BJT_NPN.dsn) Port Mapping Pin Number Port Type 1 2 2 1 3 3 BJT PNP Pre-defined BJT PNP Symbol in ADS (SYM_BJT_NPN.dsn) Port Mapping Pin Number Port Type 1 2 2 1 3 3 3-Pin FETs N-channel FET Pre-defined BJT NPN Symbol in ADS (SYM_JFET_NFET.dsn) Port Mapping Pin Number Port Type 1 2 2 1 3 3 P-channel FET Pre-defined BJT NPN Symbol in ADS (SYM_JFET_PFET.dsn) Port Mapping Pin Number Port Type 1 2 2 1 3 3 Creating Components with PSpice Model The process of creating a component with PSpice model is similar to the creation of ADS Design Kit component. Therefore, you need to translate the pspice netlist model to ADS schematic design or/and ADS netlist. This translation can be done with the help of Netlist Translator. For more information on the Netlist Translator, refer to Netlist Translator for SPICE and Spectre. The following example demonstrates the creation of the ADS design and ADS netlist files from the PSpice netlist. Suppose we have the PSpice netlist file "netlist_dio_spice.psp": *PSPICE model .SUBCKT ad508 1 2 DCD1 1 2 DMODEL .MODEL DMODEL D\(IS=1.2E-9, CJO=1.06E-12, VJ=.3, BV=70, IBV=1E-7 + EG=0.67, N=1.05, RS=27, XTI=3, M=0.3\) .ENDS To create an ADS design for the above netlist you have to follow the following steps: Start ADS. Create a new project. Import the PSpice netlist into ADS, select File > Import... In the Import dialog box select File Type as Netlist File and select More Options.... In the Import Netlist Options dialog box, select PSPICE and Translated Output Format as ADS Schematic (with name connections). Click OK. Select Browse... and select the PSpice netlist file. Click OK . Check the Translation log for any warnings/errors in the translation. The following ADS schematic window opens: Save this design file. Creation of ADS Netlist There are two ways to generate the ADS netlist: Using the GUI using the command line Creating ADS netlist using GUI: The process of creating the netlist file is similar to the creating of ADS design file except for the following: In the "Import Netlist Options" dialog box, select Translated Output Format as "ADS Netlist". Browse to the appropriate directory and the store the ADS netlist. Select the desired subcircuit from Netlist Subcircuit List (In this case, ad508 with 2 pins). Ignore the ADS design file. The generated ADS netlist should look like the following: ; Translated with ADS Netlist Translator \(*\) 312.400 January 30 2008 ;SPICE model define ad508 \(_node1 _node2\) dmodel:dcd1 _node1 _node2 Mode=1 model dmodel Diode Is=1.2e-9 Cjo=1.06e-12 Vj=.3 Bv=70 Ibv=1e-7 Eg=0.67 N=1.05 Rs=27 Xti=3 M=0.3 \ Nbv=1.05 Tnom=27 IkModel=1 end ad508 Creating ADS Netlist Using the Command Line Before you generate the ADS netlist through command line you need to set up the ADS environment variables. for more information, refer to Packaging Libraries from the Command Line, for the Cygwin, bash shell commands and Netlist Translator for SPICE and Spectre for importing a non-ADS netlist file through the command line. The nettrans command uses the following general syntax: nettrans input filename output filename [-s2|-s3|-p|-h\] -g Where: -s2 = Berkeley (2g6) -s3 = Berkeley (3c, 3e, 3f) -p = PSpice -h = HSpice In this case the command is: nettrans.exe netlist_dio_spice.psp netlist_dio_spice.net -p -g The following two files are created: netlist_dio_spice.net: ADS netlist file. This will be similar to the one generated above through GUI import. nettrans.log: Log file containing the error and warning messages.