Component, Model and Library Concepts

Summary

This article defines components, models and libraries, and their relationships. The search sequence for locating models is explained, as well as options that make this search more restrictive for specific models.

Components are the basic building blocks of an electronic product. During the design capture and implementation processes each component needs to be represented in different ways: as a logical symbol on the schematic, as a footprint/pattern on the PCB, as a SPICE definition for simulation, as a suitable signal integrity description for signal integrity analysis, or as a three-dimensional description for a 3D representation of the finished PCB.

Not all of these representations are necessary for every component, but the logical symbol is the starting point. Every component must be defined, at the very least, with its own name in a schematic library. It may contain pins and graphic symbols in single or multi-part fashion, and even have alternative display options. As such, it may be placed in any schematic design. However, until models have been added to the component, it cannot be implemented in any practical sense.

Definitions

Component: an object which can be placed into a design.

Symbol: graphical representation of the component, that is placed on the schematic. The symbol can include drawing objects that define the shape, and pins that define the electrical connection points.

Part: certain components, such as a resistor network or a relay, can be drawn as a series of separate parts, which can be placed independently on the schematic (referred to as a multi-part component).

Model: a representation of the component which is useful in some practical domain.

Footprint/pattern: names used for the model that represents the component on the PCB layout. A footprint/pattern is a grouped set of PCB pads and component overlay shapes that define the space required to mount and connect the physical component on the board layout.

Domain: type, group, or area of representation. In the DXP 2004 environment the valid domains include PCB layout, SPICE simulation, Signal Integrity analysis, and 3D.

Library: a file containing a collection of components, or a collection of models, or both.

Model Library: a file containing a collection of component models.

Component Library: a file containing a collection of schematic components.

Integrated Library: a file containing a collection of schematic components, and their associated models.

Fundamentals

At the schematic stage, the design is a collection of components which have been connected logically. To test or implement the design it needs to be transferred to another modeling domain, such as simulation, PCB layout, signal integrity analysis, and so on.

Each domain needs some information about the component, and also some way to map that information to the pins of the schematic symbol. Some of this domain information resides in model files, the format of which is typically pre-defined. Examples of these include IBIS, MDL and CKT. Some of the information does not reside in the model files, for example the SPICE pin-mapping and netlisting data must be stored and managed by the system.
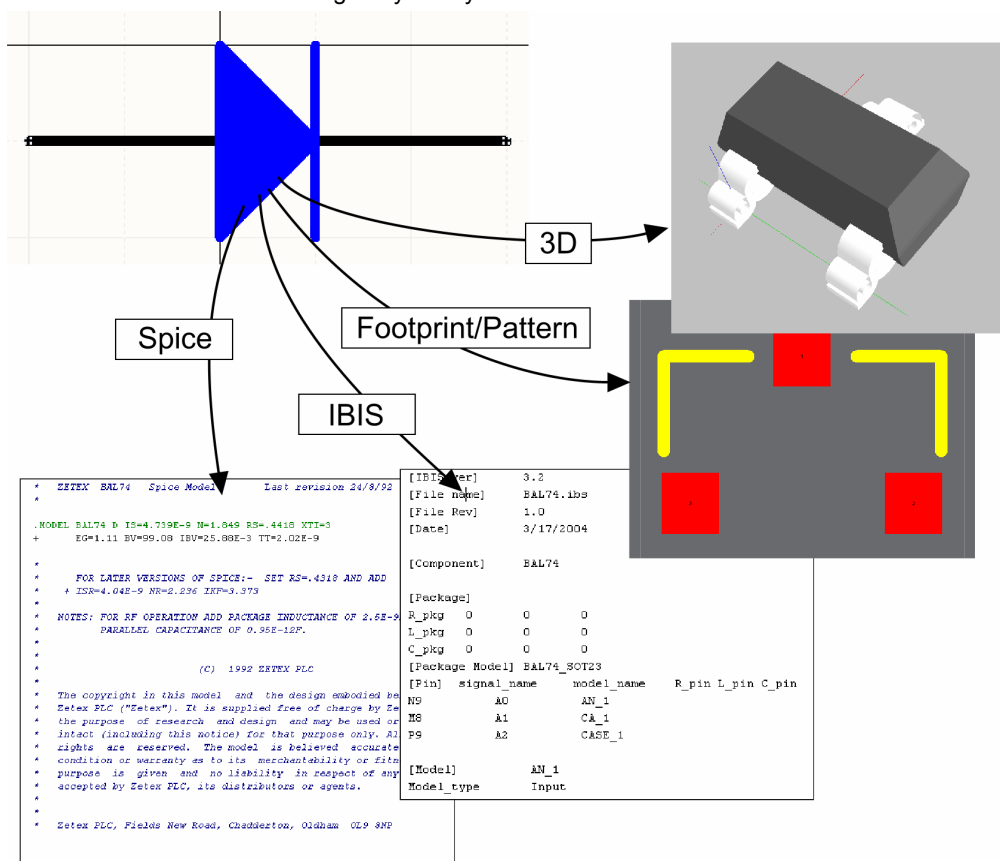

Figure 1. Information on how to model the component in each domain is stored in the model files.

Note that IBIS signal integrity models and VRML or IGES 3D models must be imported into DXP 2004 format models before they can be used. IBIS models are imported directly in the *Signal Integrity Model* dialog, which opens when you add an SI model to a component. VRML and IGES models must be imported into a Pcb3DLib before they can be added to a schematic component.

All of the necessary domain information is contained within the schematic component, which stores a separate interface to each model that has been added to it. In effect, the complete model is the combination of the model mapping information stored in the component, and the domain modeling information stored in the model library.
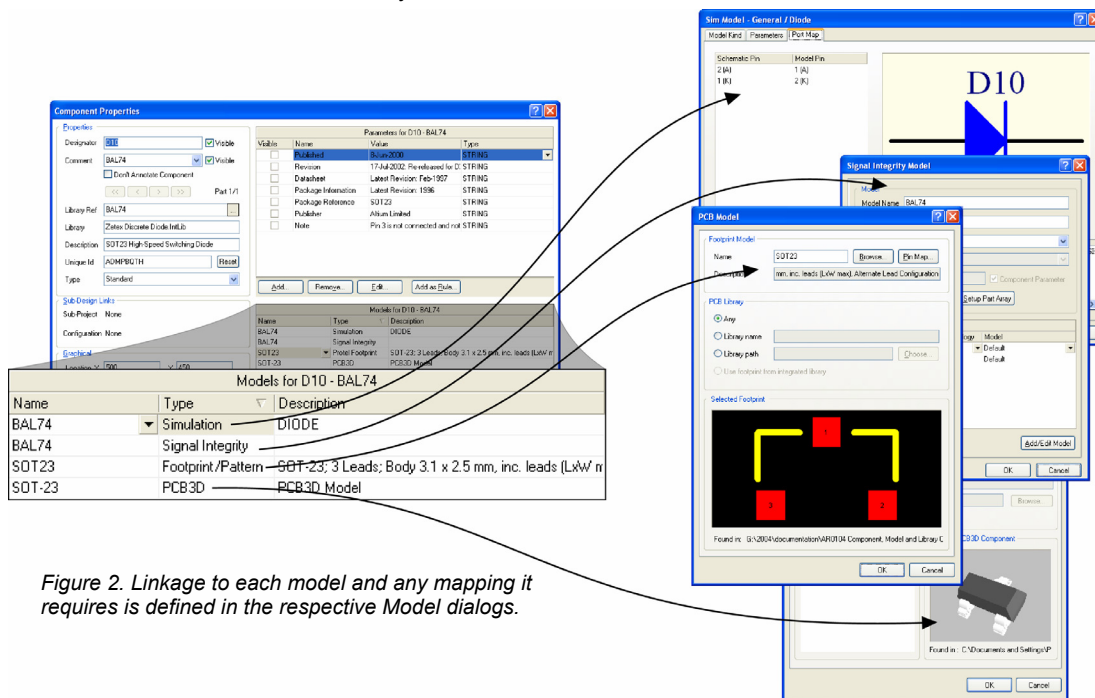

Figure 2. Linkage to each model and any mapping it requires is defined in the respective Model dialogs.

Components may have models for multiple domains, and can also have multiple models per domain, one of which will be the set as the current model.

Library Types

There are three types of libraries that can be used in the DXP 2004 environment.

Model libraries – the models for each domain are stored in "model containers", typically called model libraries. In some domains, such as SPICE, where the storage is typically one model per file, they are also referred to as model files (*.MDL, *.CKT). In other domains, the models are normally grouped into library files according to a user-defined categorization, such as PCB footprint/patterns grouped into package-type libraries (*.PcbLib, *.PcadLib).

Schematic libraries – these contain schematic components and their model interface definitions (*.SchLib). Each model interface definition contains a link to its respective model library.

Integrated libraries – these are a set of schematic libraries, which, together with their linked model libraries, are 'compiled' into one file – the integrated library (*.IntLib). The advantage of compiling into an integrated library is that all component information is available in a single portable file. Integrated libraries cannot be edited without extracting the sources, and recompiling.

Referencing a Model

Whenever you add a model to a component you have the option of defining how tightly you want to control where the model is searched for. Although they vary slightly from one model type to another, the model editor dialogs generally include these options:

Any – searches all valid libraries for a matching model.

Library name – only searches valid libraries of this name for a matching model.

Library path – only searches a valid library of this name in this location for a matching model.

Integrated library – draws the model directly from the integrated library used to place this component. The integrated library must be available in a valid location.

Locating the Model – Valid Search Locations

Each time you perform an operation that requires a model, the system will search for it in the valid locations, according to the reference criteria defined in the previous section. For example, when you perform a circuit simulation, the SPICE model linked to each component is retrieved and used by the XSPICE simulation engine. Another example would be transferring the design from schematic capture to PCB layout. During this process, the footprint/pattern for each component must be retrieved and placed on the PCB.

The valid locations of models that can be searched include:

Project Model/Library – models and model libraries can be added into the project. Like all project files, model/library files are only linked to the project, so you can link a model/library to many projects. The advantage of this approach is that whenever the project is opened the models will be available. The disadvantage is that models/libraries that are not stored in the project folder structure can be forgotten if the project files are moved from one PC to another. Project models/libraries are only available when you are editing a document that belongs to the current project.

Installed Library – these libraries are associated with the DXP 2004 environment. Components in installed libraries are available to all open projects.

Project Search Path – models/libraries can be made available to a project by defining a search path in the *Options for Project* dialog. Each search path defines a folder, and can include sub-folders if the Recursive option is enabled. All model and library files found down the search path will be valid. Note that retrieving models using search paths can be slow if there is a large number of files in the search path folder(s).

Checking the Available Models/Libraries

You can review the models/libraries available to the current project in the *Available Libraries* dialog. Click the **Libraries** button in the **Libraries** panel, or select **Design » Add/Remove Library** to open the dialog.
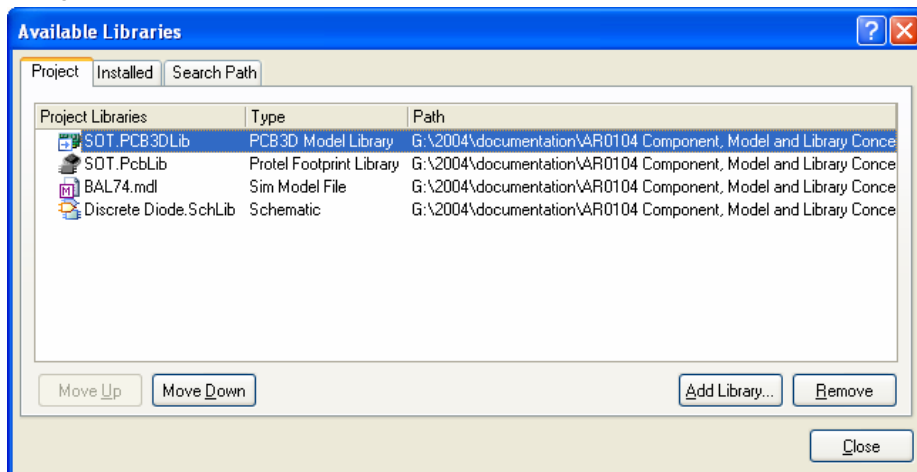

Figure 3. The Available Libraries dialog shows all models/libraries that are available to the active project.

General Search Order

The rule is to stop searching for a model as soon as a match is found. For all models not tied to an integrated library, the search will proceed in this order:

- Project Libraries
- Installed libraries
- Search Paths

You will notice that this order is followed from left to right through the *Available Libraries* dialog. In fact, since the available libraries can be ordered within this dialog from top to bottom, the entire search sequence is intuitive and easy to set up.

While the DXP 2004 environment offers flexibility and control over your model locations, it does require you to use the correct file extension for each model type. For example, a footprint cannot be found unless it is in a file with a `*.lib` or `*.pcblib` extension. Similarly, a SPICE `.SUBCKT` will not be found unless it is in a `*.ckt` file, nor will a SPICE `.MODEL` if it is not in a `*.mdl` file. Whenever a model search does not yield a match, an error will appear in the *Messages* panel.

Integrated Libraries

Integrated libraries are a special class of library, where the schematic symbol library and all referenced models are compiled into a single file. The advantage of an integrated library is portability and security. Since all models are packaged into the integrated library only one file needs to be available to the project, or moved when the project is relocated. Components and models in an integrated library are not available for editing, unless the library is decompiled (open the IntLib to extract the sources).

An Integrated library is built by:

- Creating an integrated library package
- Adding schematic symbol libraries
- Referencing the required models from each component symbol
- Setting the integrated library project options, including the location of the compiled output
- Compiling the integrated library to produce the integrated library (`Name.IntLib`)

If a component symbol is placed from an integrated library then the system will only attempt to retrieve the referenced models from that integrated library. Note that the integrated library must be available in the valid search locations when the system is attempting to retrieve a model.

For more information on integrated libraries refer to the article *Enhanced Library Management Using Integrated Libraries*.

For step by step instructions on creating a component library, creating component symbols, and referencing models, refer to the tutorial *Creating Library Components*.

For instructions on creating an integrated library, refer to the tutorial *Building an Integrated Library*.

Properties of the Component

The basis of a component is the symbol. Rather than requiring each component symbol represent a unique component, the system supports different approaches to building components.

Common graphic, different component

One component symbol for each physical component – ideal for components such as integrated circuits, where each component symbol represents an actual physical component. The component would include suitable models, such as the PCB footprint/pattern, the signal integrity model and the 3D model.

One symbol for graphically equivalent components – useful when components are logically equivalent, but have a different component specification. An example would be a logic gate that is available in a variety of logic families, for example a 74ACT32 and a 74HC32. In this case the component symbol is drawn once, and a *component alias* is defined for each equivalent component required. Component aliases are added in the Schematic Library editor panel. Component aliases can be thought of as one component, with multiple names.

One symbol for a type of component – useful for discrete components, such as resistors. The component could link to multiple PCB footprints/patterns. The component value is defined when the component is placed on the schematic, rather than in the library.

Common component, different graphics

The system also supports multiple graphics for the same component. For example, you may have one client that requires their component symbols drawn using component traditional drawing shapes and an other that requires the component symbols to be drawn in accordance with a specific standard. To define additional graphical representations for the one component, add a new component Mode, either in the library editor **Tools** menu, or using the **Mode** toolbar.
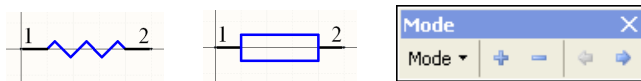

Figure 4. Use the Mode feature to define multiple graphical representations of the one component.

Non-standard component types

Not all components are destined to be mounted on the assembled PCB, not all components are required in the bill of materials, and not all items that are mounted on the PCB need to be represented on the schematic. The system supports this through the **Component Type** property, set in the *Component Properties* dialog (in the library or on the schematic).

For example, the presentation and readability of your schematic might be enhanced by including a chassis mounted component that is wired to the PCB. If this component was not required in the PCB BOM, then the component type can be set to **Graphical**. A graphical component is not included during
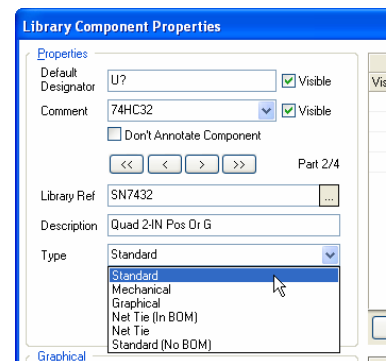

Figure 5. Set the component type for special component requirements.

schematic electrical verification, it is not included in the BOM, nor is it checked during schematic to PCB synchronization. In this case the **Component Type** is set to **Graphical**.
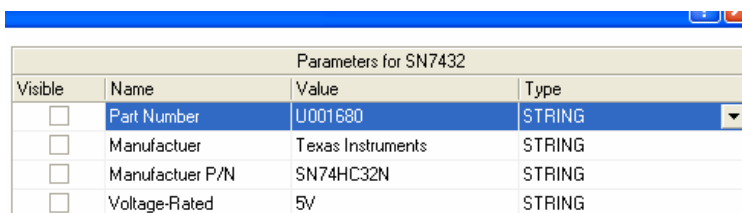
Another special class of component would be a test point – this component is required on both the schematic and the PCB, it should be checked during design synchronization, but is not required in the BOM. In this case the **Component Type** is set to **Standard (no BOM)**.

Another example of a special component kind would be a heatsink – typically it is not shown on the schematic (but could be), is not required to be checked during schematic electrical verification, but must be included in the BOM. In this case the component type is set to **Mechanical**.

Use the What's This help ❓ for details on the various component types, or press F1 when the *Component Properties* dialog is open.

Component parameters

Typically there is a variety of text information that must be included with a component. This could include such things as component electrical specifications (eg wattage or tolerance), component purchasing or stock details, designer notes, or

Figure 6. Add component parameters to include required component detail.

references to component datasheets. This information is included by adding parameters to the component, either during component creation in the library editor, or once the component has been placed on the schematic.

📖 For more information on linking from components to a company database, refer to the tutorial *Linking from a Company Database to Components in Your Design*.

📖 To reference a document such as a datasheet from a component, include the parameter *HelpURL*. The parameter value points to the document, and can include a page number for a PDF. For example, the HelpURL value of *CR0118 FPGA Generic Library Guide.pdf#page=93* would result in the referenced PDF file being opened at page 93, when the F1 button is pressed when the cursor is over the component. The reference can be to a local document (include the path if the path is not defined by the operating system), or the reference can be to an internet URL. If the component does not include a HelpURL parameter the default component help topic will appear.

Multi-part components

In some instances it is more appropriate to represent the one physical component using multiple symbols, for example each resistor in a resistor network, or the coil and contacts of a relay.

Additional parts are added or removed using the commands in the library editor **Tools** menu. Each part is drawn individually, and pins are added.
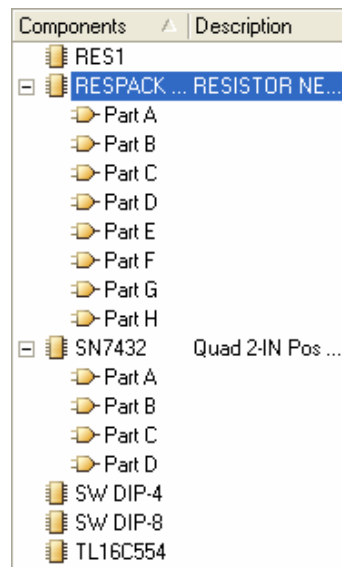
Figure 7. A component can have one or more component parts.

Power pins

Component power pins can be shown visibly, like any other component pin, or they can be hidden. Hidden pins, that have their electrical type set to Power, are automatically connected to the net defined by the **Connect To** property in the *Pin Properties* dialog.

To display hidden power pins during component design, enable the **Show Hidden Pins** option in the library editor **View** menu.

Figure 8. For hidden power pins the Connect To property defines the net that this pin must connect to.

Revision History

Date	Version No.	Revision
9-Dec-2003	1.0	New product release
28-Jul-04	1.1	Added images and extra detail to text

Software, documentation and related materials:

Copyright © 2004 Altium Limited.