# Jazz Design Kit Overview

This document contains an overview of the Jazz Semiconductor analog and mixed signal design system, based primarily on Cadence dfII design framework with integrated support for Agilent ADS simulation and Mentor Calibre verification. It assumes that the design kit is properly installed and supported tool versions are being used, as described in the design kit installation instructions. If you are unfamiliar with Cadence IC design software, Cadence training is highly recommended. The following topics are covered in this document:





## 1. Revision History

| Revision | Revision Description | Date |
|---|---|---|
| 01 | Initial release. Copied for rdsoverview.pdf | 06/18/03 |
| 02 | Section 2 - Updated "user_list" setup instructions.<br>- Added -tech_ver option in the makeProjectTree example.<br>Section 4 - Removed reference to Cadence Design Management system and managed libraries.<br>Section 5.1 - Updated Schematic creation instructions<br>Section 7.6 - Deleted "Rose" instructions. Rose no longer supported. | 07/22/03 |
| 03 | Added Revision Description table. Made no additional content changes. | 07/31/03 |
| 04 | Restructured document to include typical design kit environment and tool flow. Updated sections on library manager and added Jazz menus. Added reference to verification using Calibre Interactive and Assura. | 09/29/05 |

# 2. Design Kit Environment

The color coding in this section is used to highlight a particular environment variable, file or directory or set of directories that is referred to multiple times. Red denotes an environment variable, blue denotes a user setting for a variable, orange denotes a system file and green is used to highlight a particular directory path that will be referred to in greater detail in a the next section. Angle brackets indicate that a user value must be filled in for the variable.

## 2.1. Typical Installation

In the typical Jazz design kit setup, work is done from a project tree which is initially created as part of the design kit installation. This section describes this work environment and assumes typical installation as per the "Jazz Cadence Design Kit Installation Instructions," document NPB PS-0419.

The project tree contains a directory structure and hierarchical set of initialization files. This sets a common methodology, set of tools and directory structure geared towards quickly and efficiently sharing design information among different groups and/or sites. If a different setup is preferred, then the Jazz kit can be installed using the custom procedures detailed in the installation document.

The project tree contains a work directory for each user. An alias is used to go to this directory and source the user's initialization file. Additional files used by Cadence are contained in the user's work directory. These files have a hierarchy and set the process variant and allow customization for project user, all users of a project, or all users that set a particular environment variable.



## 2.2. Adding the kit to your environment

The first step to load the Jazz design kit is to set several environment variables that define the location of the kit. This done by sourcing a file created during kit installation, typically called **JAZZ.cshrc**. For convenience, an alias can be added to the ~/.cshrc file called "jazz" to source this file automatically. For example:

**alias jazz 'source ~/JAZZ.cshrc'**

With this alias, typing "jazz" will setup the shell to load the Jazz design kit.

## 2.3. Project Creation / Adding Users

A "Jazz" project is created during design kit installation for a set of users. The project contains a directory and file structure for all users of the project, including both shared directories and user-specific directories. Additional projects can be created as needed, and the existing projects can also be updated to add new users. Adding new users does not change anything for existing users. The project is created or updated using a "makeProjectTree" perl script supplied in the Jazz kit. Once the Jazz kit environment variables have been sourced as in the previous section, a summary of all of the command line arguments can be obtained by typing:

**makeProjectTree -help**

Please refer to NPB PS-0419 "Jazz Cadence Design Kit Installation Instructions" for a detailed description of the makeProjectTree utility. Below is the most commonly used method for creating a new Jazz Cadence project, or adding users to an existing project:

- Determine the root project directory. This is the root location for all of the design projects using the Jazz kit, and is typically set by the kit administrator at the time of the installation. All members-to-be of the project need access and write privileges to this directory.

  **echo $PROJ_ROOT**

- CD into the project directory. If it does not already exist, then create it:

  **mkdir $PROJ_ROOT /design_project_sbc18hx**

  **cd $PROJ_ROOT /design_project_sbc18hx**

- Create a UNIX file called "user_list" containing a list of UNIX login account names (1 per line) that defines all the new members of the project to be created. Existing user accounts will not be effected:

  **echo user1 > user_list**

  **echo user2 >> user_list**

- Run the "makeProjectTree" script as shown below. Substitute <PROJ_ID> with the name of a project, <tech> with the name of the Jazz technology to be used, and <tech_ver> with the name of the technology variant.

  **makeProjectTree <PROJ_ID> -users user_list -cds -tech <tech> -tech_ver <tech_ver> -ver**

- For example, to create the project "design_project_sbc18hx" using the "sbc18hx" variant of "sbc18":

  **makeProjectTree design_project_sbc18hx -users user_list -cds -tech sbc18 –tech_ver sbc18hx –ver**



## 2.4. Technology Setting

After setting the kit location and identifying or creating a project tree, the Jazz alias "cdsprj" completes the final setup step. There may be multiple Jazz technology nodes in use, and each technology node supports a number of process variants. This final step combines the process-specific kit information with the kit framework from the JAZZ.cshrc sourced in section 2.2. Note that this "cdsprj" alias uses the environment variable $PROJ_ROOT as the base location for Jazz projects. Assuming the project directory is $PROJ_ROOT/ **design_project_sbc18hx**, type the following to complete the project-specific setup:

**cdsprj design_project_sbc18hx**

- If the project is not in the default site-wide location under $PROJ_ROOT, it can be reset before using the "cdsprj" command. For example, the project nextGreatChip has been created in a top secret location:

  **setenv PROJ_ROOT ~/my_very_own_secret_projects**

**cdsprj nextGreatChip**

- The “cdsprj” command sources a hierarchical set of files that refine the environment variables to make them specific to the process used. It also places you in a directory with a predefined .cdsinit file that loads the design kit skill code through a sequence of *.cdsinit files. Together, these two sets of files allow the kit to be customized on a combination of user, project and site level. These files are introduced in subsequent sections, and the order in which they take effect is also shown.
- Once you have setup your shell for a project, you are ready to start the Cadence executable, for example by typing “icfb &”. If the “cdsprj” has not been used, then the environment definitions and skill code must be added independently as part of a “custom” installation.

## 2.5. Launching the design kit

Once the alias and project are set up, only three steps are needed to start Cadence each session. For example, using the **design_project_sbc18hx:**

**jazz**

**cdsprj design_project_sbc18hx**

**icfb &**



After a successful setup, you should be presented with the CIW (Command Interface Window), from which you can launch all Cadence tools. The Jazz menu will appear on the rightmost spot in the menu bar, and includes release notes and other items documented in section 3. Additional Jazz menus are included in the schematic, analog artist and layout windows. As usual, clicking on the Help button in the top right corner of the CIW brings up Cadence’s extensive on-line documentation.

The Jazz kit version is displayed in the CIW and cds.log next to the string “Jazz kit version.” It can also be generated by calling the procedure jazzGetKitVersion().

## 2.6. Base directory structure and initialization files

The following acronyms will be used throughout this section:

**“RDS_ROOT”** RDS_ROOT represents the complete path to the Jazz design kit installation (all the way to the HOTCODE directory). This location can be shown by typing “echo $RDS_ROOT” from within the Jazz kit environment (IE after using the jazz alias). The installation procedure may define this as a soft-link pointing to a particular kit revision in order to make it easier to install a design kit upgrade. RDS_ROOT can be customized on a user and/or project specific basis as described in section 2.8, allowing a one project to maintain an old kit version and another project to use a newer kit version.

**“RDS_TECH”** RDS_TECH is a sequence of letters and numbers that uniquely identify a process technology throughout Jazz Semiconductor (e.g. sbc18). A process

technology may contain multiple process variants differing in front or backend features such as triple well, number of metal layers, etc.

**"RDS_CUSTOM"** This variable can be set to a location outside of the Jazz kit to allow customization to be put in place that is unaffected by Jazz kit upgrades. For useage instructions, see section 2.8.

The basic directory structure and support files are described below. The support files can be viewed in the design kit installation.

**$RDS_ROOT /**

| | |
|---|---|
| bin/ | (directory containing makeProjectTree) |
| **etc/** | (directory containing initialization and configuration files) |
| amslibs/cds_default/ | |
| **etc/** | (directory containing Cadence initialization files) |
| **cdsware/** | (directory containing general purpose files) |
| **cdslibs/** | (directory containing technology specific files) |

**$RDS_ROOT /etc/** (directory containing initialization and configuration files)



| | |
|---|---|
| **RDS.cshrc** | Sets design kit variables and default locations for RDS_CUSTOM and PERL. |
| **cdsDesKit.cshrc** | Adds kit bin directories to path using prepend command contained in jazz environment. Defines cdsprj alias. |
| **cdsSystem.cshrc** | This file contains Unix environment variables used by the design kit as default. Project and User settings overwrite anything specified here. |

**$RDS_ROOT/amslibs/cds_default/etc** (directory containing Cadence initialization files)

| | |
|---|---|
| **system.cdsinit** | Loads the skill code for the Jazz design kit. It is loaded by the ".cdsinit" file in the user's project-specific work directory, and it also loads site, project and user specific .init files to support custom skill code. |
| **system.lib** | Top-level file mapping the library names for both Cadence and Jazz libraries. Utilizes include to add technology specific library. |
| **display.drf** | Color mapping file. Project specific color mapping files can be loaded by adding the file $PROJ_ROOT/$PROJ_ID/cds_master/display.drf. |
| **/templates/** | (Includes template files that are customized by makeProjectTree to create the project specific initialization files. If the standard project setup is not used, then the template files can be copied into the work area and customized.) |

**$RDS_ROOT/amslibs/cds_default/cdsware/**

($RDS_CDSWARE: Contains general skill code and utilities that are used across multiple process technologies. This is also the location for the application notes and documentation files contained within the Jazz menus.)

**$RDS_ROOT/amslibs/cds_default/cdslibs/**

**cds_generic/** (directory location for cds_generic library of symbols. This library contains all the components that can be found in analogLib plus components added by Jazz.)

**$RDS_ROOT/amslibs/cds_default/cdslibs/$RDS_TECH/**

($RDS_CDSLIBS/$RDS_TECH: directory containing technology specific files)

| | |
|---|---|
| **bin/** | (directory containing technology specific skill code) |
| **calibre/** | (directory containing runsets for calibre interactive) |
| **devVariantsList** | Defines process variants. |
| **devices/** | (actual devices used within Jazz kit) |
| **models/** | (models – typically ads, spectre and hspice models.) |
| **pex/** | (directory containing definitions for Jazz custom parasitic extraction) |
| **processData** | Specifies current density for resistors |
| **rcxlib/** | (directory containing reference devices for internal use by RCX only) |
| **technology.lib** | This is the technology library for the particular process technology. |
| **techfiles/** | (directory containing technology files for each process variant) |



## 2.7. Project directory structure and initialization files

The directory structure and locations of all initialization files used in the standard Jazz project structure are described below. They are automatically created by running the perl script makeProjectTree as described in section 2.3. The makeProjectTree command is added to the path when the jazz alias is typed, as described in section 2.2, or the JAZZ.cshrc is otherwise sourced.

The following kit variables will be referenced throughout this section:

| | |
|---|---|
| **"PROJ_ROOT"** | root location for all of the design projects using the Jazz kit |
| **"PROJ_ID"** | the name of a particular project |

The project directory contains both shared directories and user work directories. Design management tools can be used to provide revision control of the process of copying data between the shared and individual work directories.

The typical design kit setup provides customization to both the environment variables and the skill code at the user, project and site levels. An example of this customization is that a different version of the design kit can be specified in these files, leaving freedom to leave a design on an older version of the kit or switch just one project user to a new kit for evaluation purposes. The underlines files below can be edited to provide this customization. The next section gives more details on customization.

**Top-Level Project Directory Structure**

**$PROJ_ROOT/$PROJ_ID**

| | |
|---|---|
| doc/ | (project documentation) |
| **cds_master/** | (common project definition) |
| gds_dir/ | (repository for project layout data in GDSII format) |
| **verification/** | (project verification directory) |
| **$PROJ_ID_creation.log** | stores project creation and revision history |
| **work_libs/** | (user working directories) |
| bond_info_dir/ | (project workarea for bonding information) |
| dxf_dir/ | (directory for storing project bonding diagrams in dxf format) |

**Project Shared Directories**

**$PROJ_ROOT/$PROJ_ID/cds_master/**

| | |
|---|---|
| design/ | shared project library defined as **<$PROJ_ID>**_master |
| project.lib | Defines libraries which are visible to all of the users of the project. The file automatically includes the system.lib file, which includes the project.lib file containing all of the devices in the process. Users can add additional libraries to this file to share them with all of the project users. |
| cdsPrj.cshrc | This project initialization file defines the technology and the process variant. Environment variables can be set and/or changed on a project specific basis by defining them at the end of this file. |
| proj.cdsinit | Cadence initialization file for the project. If this file exists it is loaded by system.cdsinit. Project specific skill code can be loaded here and all of the project users can run it. |
| softcells/ | (project repository of generated cells for Cadence) |



**$PROJ_ROOT/$PROJ_ID/verification/**

Directory structure for storing verification results in Jazz calibre flow

| | |
|---|---|
| cbr_dir/ | (netlists of design data for verification with Calibre) |
| drc/ CellName/ | (DRC results) |
| ant/ | (antenna results) |
| density/ | (DRC density results) |

| | | |
|---|---|---|
| esd_lup/ | | (LUP and ESD results) |
| lup/ | | (LUP only) |
| softerc/ | | (soft erc results) |
| stress/ | | (DRC stress results) |
| street/ | | (DRC street results) |
| lvs/ | CellName/ | (LVS results) |
| pex/ | | (parasitics extraction results) |

**User Working Directories**

**$PROJ_ROOT/$PROJ_ID /work_libs/**

username/cds/ (user's schematic and simulation directories)

| | |
|---|---|
| **.cdsinit** | This file loads the system.cdsinit, which loads the Jazz design kit skill source code. |
| **assura_tech.lib** | defines location of assura verification deck(s) |
| **cds.lib** | defines the user's cds.lib, including their design library |
| **cdsUsr.cshrc** | user initialization file for environment variables – sources cdsPrj.cshrc. Customization can be added at the end of the file. |
| design/ | (user's project design library **<$PROJ_ID>**_username) |
| doc/ | (user's documentation directory) |
| simulation/ | (default location to store simulation data) |
| hsp_dir/ | (default location to store hspice netlists) |
| vhdl_dir/ | (default location to store vhdl netlists) |
| iccraft/ | (location where IC-Craftsman stores temp data) |



## 2.8. Customization

**A. Site customization files**

The Jazz kit uses the "**RDS_CUSTOM**" environment variable source optional global or site customization files. By default, **$RDS_CUSTOM** is set to point to a directory called "**/rds/prod/custom**" by the **RDS.cshrc** file, which is sourced by the **JAZZ.cshrc**. The value can be changed by adding the following at the **top** of the JAZZ.cshrc:

**setenv RDS_CUSTOM <your customization directory>**

The following files will be used if found in the **RDS_CUSTOM** location:

**$RDS_CUSTOM/**

| | |
|---|---|
| **cdsSite.cshrc** | (This file adds to and/or overrides the environment variables set in the **cdsSystem.cshrc** file. Project and User settings override anything specified here.) |
| **cadence/site.cdsinit** | (This file adds to and/or redefines skill code loaded by the **system.cdsinit** file. Project and User settings are loaded after this file, and so take precedence.) |

These files do not need to be defined. They should be created only if a site or global customization is needed.

**B. Project customization files**

The project specific environment variables are set **at the end** of the **cdsPrj.cshrc**. The variable PROJ_RDS_ROOT can be set in this file to point to a particular design kit version, causing all users of a project to override the site wide default design kit version. The project specific skill code is set in the **proj.cdsinit**, which can be created if it does not exist. When project specific skill code is loaded here, all of the project users can run it. For example, to make the library manager appear automatically when Cadence is launched, add the following to the **proj.cdsinit**:

```
ddsOpenLibManager()
```



**C. User customization files**

The user specific environment variables are set in the project work directory, **at the end** of the **cdsUsr.cshrc**. The variable "**USER_RDS_ROOT**" can be set in this file to point to a particular design kit version. User specific skill code can be added **at the end** of the **.cdsinit** file in the same location. This **.cdsinit** loads the **system.cdsinit** file, which loads the skill code for the design kit and then loads (1) **site.cdsinit** (2) **proj.cdsinit** and (3) **~/.cdsinit**.

A final environment variable, "**DEF_RDS_ROOT**," can be set in ~/.cshrc to define a user specific, project independent design kit location which will be overridden by **$USER_RDS_ROOT** and **$PROJ_RDS_ROOT**. The RDS_ROOT order of precedence is (1) **$USER_RDS_ROOT**, (2) **$PROJ_RDS_ROOT** and last (3) **$DEF_RDS_ROOT**.

## 3. Library Manager and CIW

The Cadence Library Manager is the main tool for managing Cadence libraries and is usually one of the first tools you will need to launch after starting icfb. With the Library Manager you can perform many library management tasks such as:

- Create new libraries, cells and cell views.
- Open existing cell views for editing or viewing
- Check out, check in and version control of cell views

- Change library, cell, and view properties
- Copy cells from one library to another
- Rename libraries, cells, views, files, or reference libraries
- Delete libraries, cells, views and files
- Organize cells into categories to help you quickly locate them

**Recommended Naming Convention for cell, pin and net names:**
Use alphanumeric characters and "_". Do not use other characters like "+", "-", ".", "#". Not all EDA tools are case sensitive, so it is suggested to use always lower cases. Upper cases may be used but do not rely on the case difference to distinguish between names: myNet and mynet may turn out to be the same for some tools.

The Library Manager can be started from the Tools pull down menu of the main icfb window (CIW). Consult the on-line documentation on Library Manager for details on using this tool.
Note that typically there are many libraries available for use for any given project. Libraries that may be defined by the Jazz AMS design kit include:

- **<projName>_<userName>** - User's personal library, not typically visible by others in the project. Can be used as individual staging and development areas. To share some cells with others in the project, they can be moved or copied to the main shared library described below.

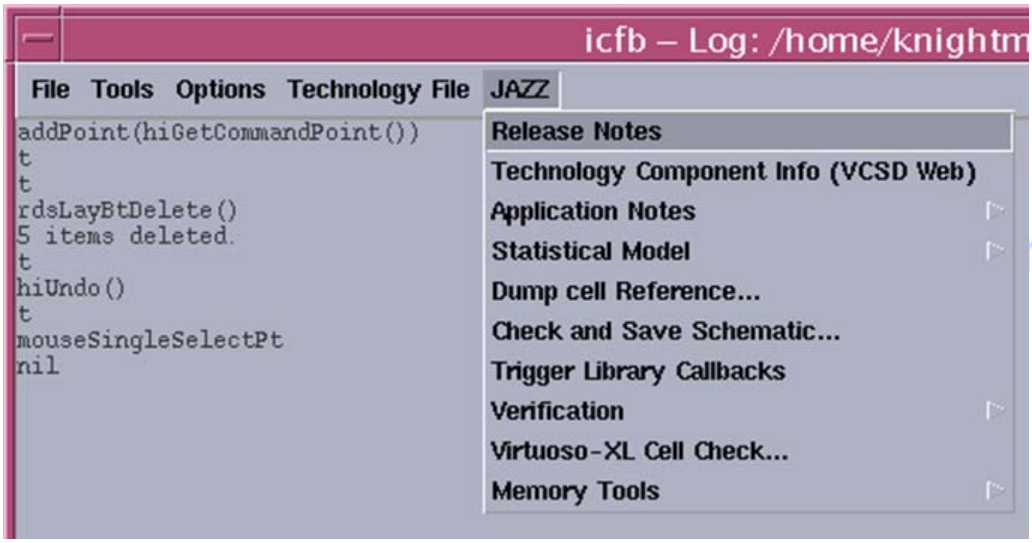


- **<projName>_master** - Shared library that is visible by all others in the project. Depending on work style, this library could contain "golden copies" of cell views that have been generated and verified in the project. These cell views could originate from individual user libraries described above.
- **<tech>** - This library contains the primary Jazz design kit components supported by the specific process technology.

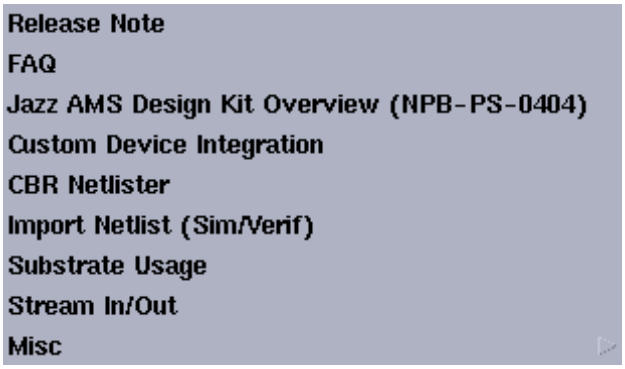
- **<tech>_techlib** - Technology file and library that contains information about the mask layers and design rules specific to the process variant. All design libraries should be attached to this library.
- **<tech>_esdlib** – Contains ESD diodes and clamp elements, plus Jazz bond pad(s) with proven reliability. The *fet*_esd devices, when present, are in the main <tech> library.
- **<tech>_rcx** - This is a reference library that contains cells and cell views used internally by RCX. The components should not be used directly.
- **Discrete_Components** - Obsolete

- **scgp and scgp_analog** - These are general purpose standard cell (ASIC) libraries that are distributed separately. Once installed, they can be added to the project.lib or added to the site customization.
- **cds_generic** - Ideal components included in the Jazz kit. Includes parasitic devices used in the Jazz design flow
- **sheetBorder** - Sample sheet border included in the Jazz kit.
- **softcells** - Required scratch directory used to construct layouts for some devices.
- **analogLib, ahdlLib, basic, sbaLib** - The Jazz kit system.lib defines these Cadence libraries pointing to the Cadence installation directory. See the Cadence documentation for more information about these components.

## 3.1. Jazz menu in CIW (Command Interface Window / Initial Screen):




- **Release Notes:** Describes changes made in each release of the design kit.
- **Technology Component Into (VCSD Web):** Detailed descriptions of components in the Jazz design kit libraries and their CDF parameters.
- **Application Notes Submenu Contents:**




- **Release Note:** Same as Release Notes, above.
- **FAQ:** Frequently asked questions about design kit usage.
- **Jazz AMS Design Kit Overview (NPB-PS-0404):** This document.
- **Custom Device Integration:** Application note describing the proper way to integrate a new device into the Jazz design kit when using Jazz Calibre verification.
- **CBR Netlister:** Basic information on the Jazz custom CBR netlister used in the Jazz Calibre verification flow.
- **Import Netlist (Sim/Verif):** Information on importing spice netlist as a block-box for simulation and verification.
- **Substrate Usage:** Describes the programmable nodes used to represent the substrate in the design kits for most processes.
- **Stream In/Out:** Basic instructions for how to stream in/out dfII data to gds.



➢ **Misc Submenu Contents under Release Notes Submenu Contents:**


- **Default Plotter:** Describes how to create ~/.defplot.tem and ~/.defLeplot.tem files to define user-specific plotters for composer and layout, respectively.
- **Using Sheet Border:** Brief description of the sheet border library included in the Jazz design kit as a utility.
- **Jazz Public Skill Functions:** Describes a few skill functions available to users to customize netlisting behavior and load user triggers.

➢ **Statistical Model Submenu Contents:**

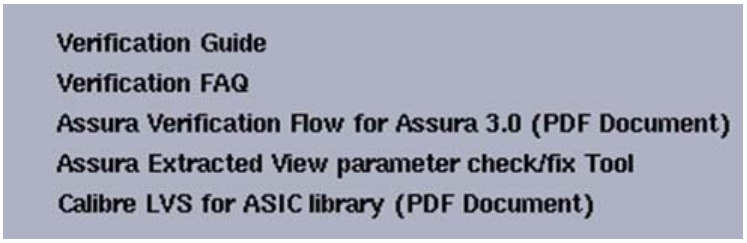
- **Mismatch User Guide and Overview:** Application Note describing Jazz statistical and mismatch model usage.

➢ **Dump Cell Reference:** Creates a file containing a list of cells and which library they are from.

➢ **Check and Save Schematic:** Checks and saves all the schematics in the selected library.

➢ **Trigger Library Callbacks:** Trigger all CDF callbacks in a library. Please see FAQ section entitled "What steps should I perform when I migrate to a new design kit?" for usage.

➢ **Verification Submenu Contents:**




- **Verification Guide:** Table describing checks are required at each level.
- **Verification FAQ:** Note on how to pass LVS for MIM cap stacked over MOS cap.
- **Assura Verification Flow for 3.1 (or 3.0):** Essential document describing setup and usage of Assura for Jazz kit.
- **Assura Extracted View parameter check/fix Tool:** A tool available to fix cases where the extracted view parameters do not match the schematic parameters.
- **Calibre LVS for ASIC library:** Documents procedures required for LVS when using ASIC libraries.



➢ **Virtuoso-XL Cell Check:** Utility to check to be sure cells match between layout and schematic.

➢ **Memory Tools Submenu Contents:**


- Cell DFM Tool: Obsolete.
- Bit Locator: Obsolete.
- Memory Stream-In: Obsolete.

➢ **Additional commands that can by typed in CIW:**

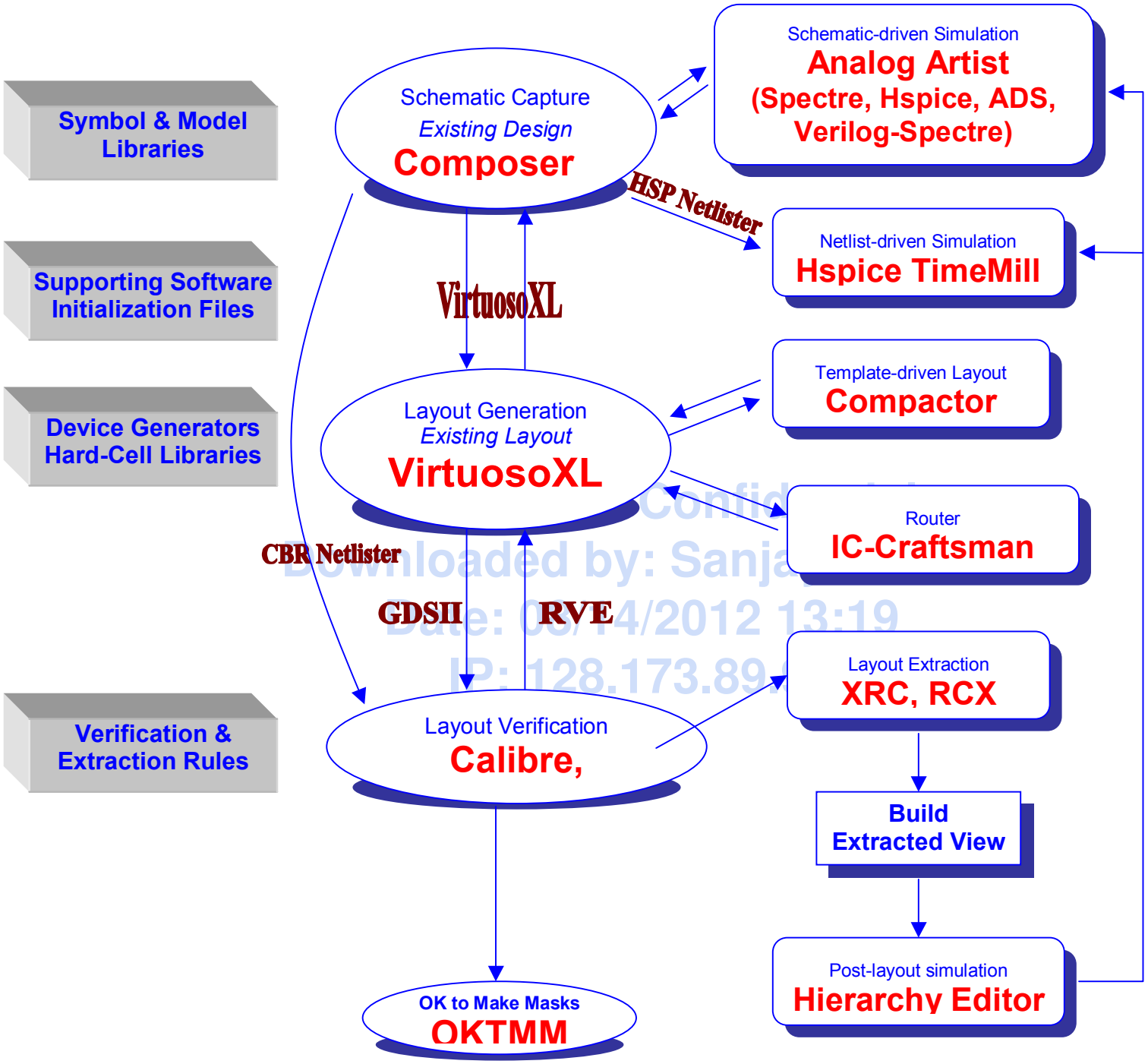
- **von & voff:** These two CIW commands turn on or off layer's visibility.
  von -> turn all layers on, voff -> turn all layers off

von(8 18 28 ...) -> turn on layer 8, 18, 28 ...
voff(8 18 28 ...) -> turn off layer 8, 18, 28 ...

- **son & soff:** These CIW commands to turn on or off layer's selectability.
  son -> turn all layers selectable, soff -> turn all layers unselectable
  son(8 18 28 ...) -> turn layer 8, 18, 28 ... selectable
  soff(8 18 28 ...) -> turn layer 8, 18, 28 ... unselectable
- **gdsNum & layerName** These CIW commands display the mapping between GDS number and Virtuoso layer names.
  gdsNum -> list the mapping of all layers.
  gdsNum("met1 met2 met3 ...") -> list the gds number for met1, met2, met3
  layerName -> list the mapping of all layers.
  layerName(8 18 28 ...) -> list the Vituoso layer name for layer 8, 18, 28.



# 4. Tool Flow



# 5. Schematic Capture - Composer

Note that the Composer Schematic Capture tool has many features, many of which will not be covered in this overview. You should consult the Cadence on-line manuals and/or sign up for Cadence training to learn more details about the many features. A few essential capabilities are covered here to get you started.

## 5.1. Creating a schematic view

You can create a new schematic view from the Library Manager as follows:

- Select the library in which you want to create the schematic view
- Select **File -> New -> Cell View...** from the Library Manager to bring up the **Create New File** form.
- Give the name of the cell in the **Cell Name** field
- You must select **Composer -> Schematic** in the **Tool** cyclic menu. This will change the **View Name** to 'schematic'. A new empty schematic view will open in edit mode after you **OK** the form.

## 5.2. Adding components to a schematic view

- In the schematic window, select **Add -> Instance** (bindkey **i**), or click on the fixed menu "chip" symbol to bring up the **Add Instance** form.
- If you know the exact library and cell name of the cell you want to place then enter them in the appropriate fields of the form. Optionally, you can click on the **Browse** button to bring up the Library Browser to aid in locating the cell you want to instantiate. Note that you should always place the **symbol** views in schematics even though there are other views available for the cell.



You can optionally specify the instance name to use for this instance (must be unique). If you don't specify an instance name, a unique name is automatically assigned to the instance when you place it in the schematic.

- Once you have chosen a valid cell symbol to place, the **Add Instance** form expands to show any optional properties you may want to modify for this instance. Once you are satisfied with the properties, move the mouse pointer into the schematic window and you will see a ghost image of the instance. Click anywhere in the schematic window to place an instance of the chosen cell. Note that you can keep placing multiple instances of this cell until you press the **Esc** key or cancel the **Add Instance** form.

## 5.3. Editing properties of schematic instances

You can edit properties of placed instances in the schematic as follows:

- Select **Edit -> Properties -> Objects...** (bindkey **q**) or select the "Property" icon from the fixed menu. Then click the mouse pointer on the schematic instance for which you want to edit properties for.

## 5.4. Connecting components in a schematic view

You can place wires between component terminals to connect them together or you can use wire labels and pins to create logical connections between component terminals.

- To add wires to your schematic, choose **Add -> Wire (narrow)** (bindkey **w**).
- To add wire labels to wires in your schematic, choose **Add -> Wire Name...** (bindkey **l**).
- To add pins to your schematic, choose **Add -> Pin...** (bindkey **p**).

## 5.5. Checking and saving captured schematics

Changes you make to the schematic are not automatically checked or saved. You should save your changes often so that you will not lose your work in case of a system crash. Also, before closing a modified schematic, you must check (extract) the schematic before it can be used for simulation purposes.

- To check (extract) a schematic, select **Check -> Current Cellview** menu pull down. Note that you can optionally modify what is checked for by changing options in the **Schematic Check Options** form.
- To save a schematic to disk, select **Design -> Save**.
- Most often, check and save are both desired after modifying a schematic, so you can do both at once by selecting **Design -> Check and Save** (bindkey **X**) or by selecting the "Check" icon in the fixed menu.

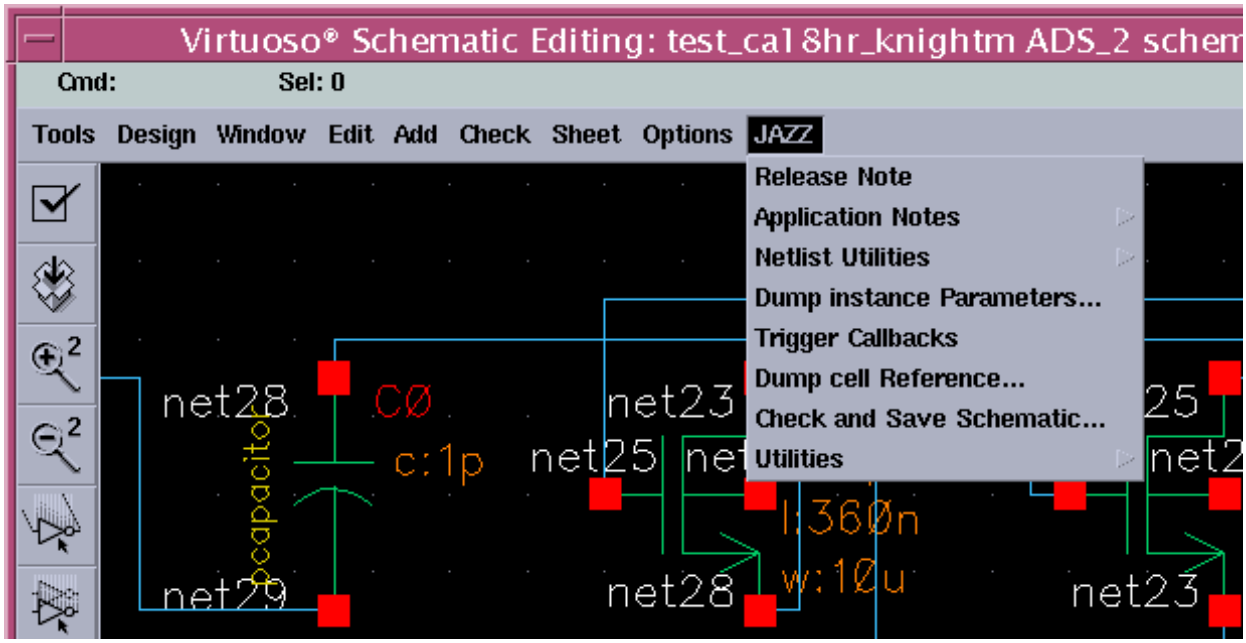


## 5.6. Creating schematic symbols

Many times it is useful to create symbols for cell schematics that you have created in your libraries. Most often you will want to partition your schematics into separate logical blocks to reduce clutter and confusion in your schematics. In this way you can use symbol views of cellblocks to represent the schematic views of the blocks, instead of placing all devices that represent each block. This is known as hierarchical design. Typically, top-level schematic views of a design will contain two or more levels of hierarchy. Note that you need to place pins in your block schematics to make signals available to the symbol views of the blocks. Once you have finished creating a schematic view with appropriate pins, you can create a symbol view automatically as follows:

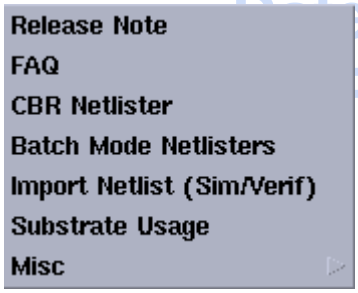
- Open up a schematic for which you want to create a symbol for:
- Select **Design -> Create Cellview -> From Cellview...** to bring up the **Cellview from Cellview** form. The form should default to all the options you want for creating a symbol representation of this schematic, with the correct pin names. **OK** the form.
- In the **Symbol Generation Options** form, you may wish to rearrange the placement of pins to suit your needs before you OK the form.
- You may wish to hand edit the symbol after it has been generated to suit your preferences.

## 5.7. Jazz menu in Composer






- **Release Note:** Describes changes made in each release of the design kit.
- **Application Notes Submenu Contents:**




  - **Release Note:** Same as Release Note, above.
  - **FAQ:** Frequently asked questions about design kit usage.
  - **CBR Netlister:** Obsolete.
  - **Batch Mode Netlisters:** Obsolete.
  - **Import Netlist (Sim/Verif):** Information on importing spice netlist as “black box” for simulation and verification.
  - **Substrate Usage:** Describes the programmable nodes used to represent the substrate in the design kits for most processes.
  - **Misc Submenu Contents:**
    - **Default Plotter:** Describes how to create ~/.defplot.tem and ~/.defLeplot.tem files to define user-specific plotters for composer and layout, respectively.

➢ **Using Sheet Border:** Brief description of the sheet border library included in the Jazz design kit as a utility.

➢ **Netlist Utilities Submenu Contents:**


- **cbr for Calibre:** Generates calibre netlist from schematic for use in Jazz Calibre flow.
- **CBR for Calibre with Include files:** Generates calibre netlist from schematic using added CBR/CDL include files.

➢ **Dump instance Parameters:** Writes instance parameters to a file.

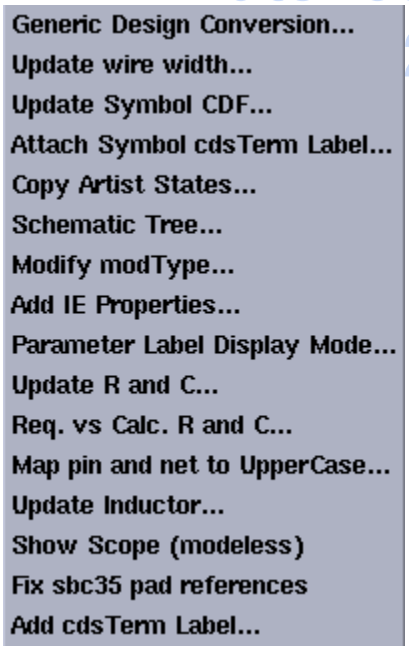
➢ **Trigger Callbacks:** Triggers callbacks for all CDF parameters in cell view or throughout hierarchy or library. Default is dry run, which will not change anything, only run through and display any error messages.

➢ **Dump cell Referencs:** Creates a file containing a list of cells and which library they are from.



➢ **Check and Save Schematic:** Checks and saves all the schematics in the selected library.

➢ **Utilities Submenu Contents:**



- **Generic Design Conversion:** Set of utilities, if they exist for the process in use, that perform specific design migration tasks such as replacing a set of cell names and library names.
- **Update wire width:** (rdsSchUpdateWireWidth) Updates the wire width based on default settings of wide path for buses / multi bit paths and narrow paths for single bit paths. Hierarchy option available.
- **Update Symbol CDF:** (rdsSchUpdateCDFFromSymbol) Updates the CDF and pin list for a symbol to match the symbol contents. It's recommend using create cellview from cellview and selecting modify, in which case this function is not needed.
- **Attach Symbol cdsTerm Label:** (rdsSchAttachLabel) Attaches all cdsTerm() label to their corresponding pins, so that moving the pins will move the labels.
- **Copy Artist States:** (rdsSchCopyArtistStates) Copies all Artist saved states from a given library and cell name to another given library and cell name. The destination state files are also appropriately updated to fix full paths.
- **Schematic Tree:** (rdsSchTree) Recursive function to output a "tree" like display from the specified cellView downwards using the specified view and stop lists. The pins can be displayed as well if desired.
- **Modify ModType:** (rdsSchModTypeModifyCB) Change NPN model type from GP <-> hicum for selected set, cell or hierarchy.
- **Add IE Properties:** (rdsAsiElformCB) Adds Interface Elements to verilog views of cells in a library or individual cell. The IE elements are used in spectreVerilog co-simulation. Please see Cadence documentation for more on spectreVerilog cosimulation.
- **Parameter and Label Display Mode:** (rdsSchParamDisplayCB) Doesn't seem to be working. Changes parameter label display mode between normal, big and bigRotate90.
- **Update R and C:** (rdsSchUpdateRC) Triggers callbacks for resistors or capacitors allowing calculated parameter to be set. For example, recalculation of resistance or capacitance with fixed length or width.
- **Req vs Calc. R and C:** (rdsSchReqVsCalRCInst) Print differences between calculated and requested values of resistance and/or capacitance output file. Reported as %delta.
- **Map pin and net to UpperCase:** (rdsSchUpperPinNetCV) Convert pin na,es, labels and sub names to uppercase, and convert local net names to global.
- **Update Inductor:** (rdsSchRecalcIndCV) Recalculate inductor electrical parameters for library or cellview by rerunning the inductor toolbox.
- **Show Scope (modeless):** (rdsSchShowScope) Displays cellname, library name and window for current schematic, and the path back to the top level schematic for the window. Modeless refers to a display window which does not have to be closed before using other windows.

- **Fix sbc35 pad references:** (rdsSchFixPadsCB) Obsolete.
- **Add cdsTerm Label:** (rdsSchAddcdsTermLabelCv) Add cdsTerm labels on symbols with best guess rotation and orientation to allow backannotation from artist, or edit-> component label display.

# 6. Simulation Environment - Analog Artist

Note that the Analog Artist Simulation Environment has many capabilities, many of which will not be covered in this overview. You should consult the Cadence on-line manuals and/or sign up for Cadence training to learn more details about these capabilities. A few essential ones are covered here to get you started.
Note also that there are many types of analyses and many optional specifications for each analysis, so it is not feasible to cover them all here.

## 6.1. Starting an Analog Artist Simulation Session

- Open a schematic you wish to simulate
- **Tools -> Analog Artist** starts an Analog Artist simulation session with Spectre as the simulator.



## 6.2. Setting up analyses (general)

- From the Simulation window, select **Analyses -> Choose** to bring up the **Choosing Analyses** form. Select the Analysis you want to run and specify the details needed to run the analysis. You can enable/disable the analysis for a simulation by toggling the **Enabled** Boolean at the bottom of the form.
- Note that options specific to this analysis can be modified by clicking on the **Options...** button on the bottom right corner of the form.
- Note that all analyses that you setup, will be displayed in the **Analyses** section of the Simulation window. You can then access these analyses again by double-clicking on the analysis of interest.

## 6.3. DC Operating point analysis

By default, DC Operating points are not saved for a DC simulation with Spectre. If you want DC Operating points to be saved, do the following:

From the Simulation window, select **Analyses -> Choose** to bring up the Choosing Analyses form.

- Select the dc analysis
- Enable Save DC Operating Point Boolean and OK the form.

## 6.4. DC Sweep analysis

In order to perform a DC sweep of a particular dc source in your design, do the following:

- From the Simulation window, select **Analyses -> Choose** to bring up the **Choosing Analyses** form.
- Select the **dc** analysis
- In the Sweep Variable section, enable the Component Parameter Boolean.
- Click on the **Select Component** button and click on the dc source in the schematic to automatically fill in the **Component Name** field.
- Choose dc in the Select Component Parameter form and OK the form.
- In the **Sweep Range** portion of the **Choosing Analyses** form, specify the Start and Stop values of the DC sweep.
- For a linear sweep, change the **Sweep Type** cyclic to **Linear** and enter step size or total number of steps for the DC sweep and then **OK** the form.

## 6.5. Specifying Outputs to be Saved and/or Plotted

By default, all node voltages are automatically saved in a Spectre simulation, but no device currents are saved. For very long transient simulations it is advised you only save enough data as needed, since saving all voltages would take a lot of disk space. To specify certain outputs to be saved:

- Bring up the Save Options form by choosing Outputs -> Save All...
- Disable the **allpub** Boolean to turn off saving of all node voltages. **OK** the form
- Choose Outputs -> To Be Saved -> Select On Schematic
- Click on wires to add individual node voltages, click on device terminals to add currents to be saved. When you are done selecting signals to be saved, press the **Esc** button to end the selection.
- Note that the signals you selected will be displayed in the **Outputs** section of the Simulation window.
- You can have output signals or expressions be plotted automatically to the waveform window following a simulation.
- Note that all the expressions you wish to be plotted must consist of outputs you have chosen to save for the simulation.

To setup automatic plotting of expressions following the simulation do the following:

- Choose Outputs -> To Be Plotted -> Select On Schematic
- Click on wires to add individual node voltages, click on device terminals to add currents to be plotted. When you are done selecting signals to be plotted, press the **Esc** button to end the selection.
- Note that the signals you selected will be displayed in the **Outputs** section of the Simulation window.

## 6.6. Running the simulation

To run a simulation, choose **Simulation -> Netlist And Run**. If you have modified the schematic since the last simulation, make sure you **Check and Save** the schematic before attempting to run the simulation, or the simulation will not run. By default, a view window will appear showing the status of

the simulation. The simulation will not hang up the DFII session, so you are able to perform other DFII functions while the simulation is running.

- If you need to stop the simulation before it completes, use the **Simulation -> Stop** menu item. A beep signifies when the simulation completes.

## 6.7. Looking at results

If you setup outputs to be plotted automatically following a simulation, then the waveform window is automatically updated with the latest results. You can manually select signals to be plotted following a simulation (assuming appropriate signals were saved for the simulation). To do this:

- Choose one of the menu items in the **Results -> Direct Plot** pull down menu, then click on the signals in the schematic you wish to plot results for. When you are done selecting the signals, hit the **Esc** button and all the signals you selected are plotted to the waveform window.

Another more common method of setting up expressions to plot to the waveform window is using the calculator. You can bring up the calculator by choosing **Tools** -> **Calculator**... With the calculator, you can setup simple expressions or many mathematical expressions to be plotted. Consult Cadence on-line documentation for details on using the Calculator.

You can also print simulation results to a view window instead of plotting them to the waveform window. To do this, choose one of the menu entries in the **Results** -> **Print** pull down menu, then click on the signals in the schematic, then hit the **Esc** button. This will print results to a view window. Furthermore, you can annotate results, such as DC node voltages or DC operating points as labels on schematic devices. To do this, choose one of the entries in the **Results** -> **Annotate** pull down menu. You can restore labels on the schematic by selecting **Results** -> **Annotate** -> **Design Defaults**.



## 6.8. Using Design Variables

You can assign many instance properties to variables instead of numbers. By doing so, you have the flexibility to modify the variable in a simulation if you are not sure exactly what the value of the parameter should be, or wish to investigate the effects of a parameter on selected outputs. Furthermore, this allows you to change the value of a schematic parameter without having to renetlist the schematic every time. If you assign variables as instance properties, you should define the variables with associated default values in the Simulation window **Design Variables** section.

To add a design variable to the Simulation window:

- Choose Variables -> Edit... to bring up the Design Variables form.
- Enter the variable name and associated value, then click on the **Add** button. When you are done entering Variables and their values, **OK** the form.
- You can use this same form to modify the value of variables, or various other actions related to Design Variables.

## 6.9. Saving Simulation Sessions

It is often useful to save a simulation setup before ending a simulation session so that you can recall your simulation setup for a schematic without having to redo the setup. To do this:

- Choose Session -> Save State...
- Assign a meaningful name to the current simulation setup, then **OK** the form.

To load a previously stored setup, do the following:

- Choose **Session -> Load State...** then choose the desired state in the **State Name** list box, then **OK** the form.

Note that simulation sessions by default are saved to **$HOME/.artist_states** directory, so they are usually not visible by others in your project.

## 6.10. Parametric Analysis

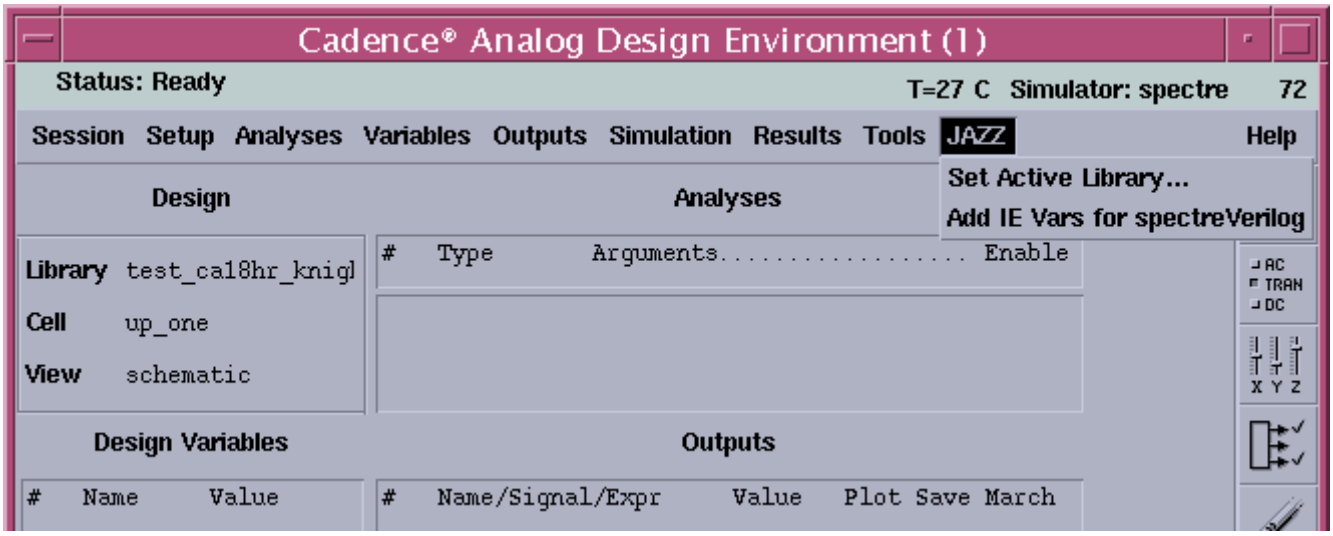
Parametric Analysis tool allows you to sweep any Design Variables you have defined in a Simulation Session. This tool will automatically run a series of simulations using the parameter ranges you specify and generate a set of outputs for each simulation, allowing you to view the set of results following the simulation. Note that you can choose as many variables to sweep in a Parametric Analysis as you wish, so you can perform multidimensional simulations. This is more efficient than manually adjusting parameter values and re-running simulations and comparing results to previous simulations. Note that **temp** is a built-in design variable designed to allow you to sweep temperature. Do not add **temp** as a design variable as this will conflict with the built-in variable. You can bring up the Parametric Analysis by selecting **Tools -> Parametric Analysis...** menu entry. Click the **Help** button on this tool for more details of using this tool.



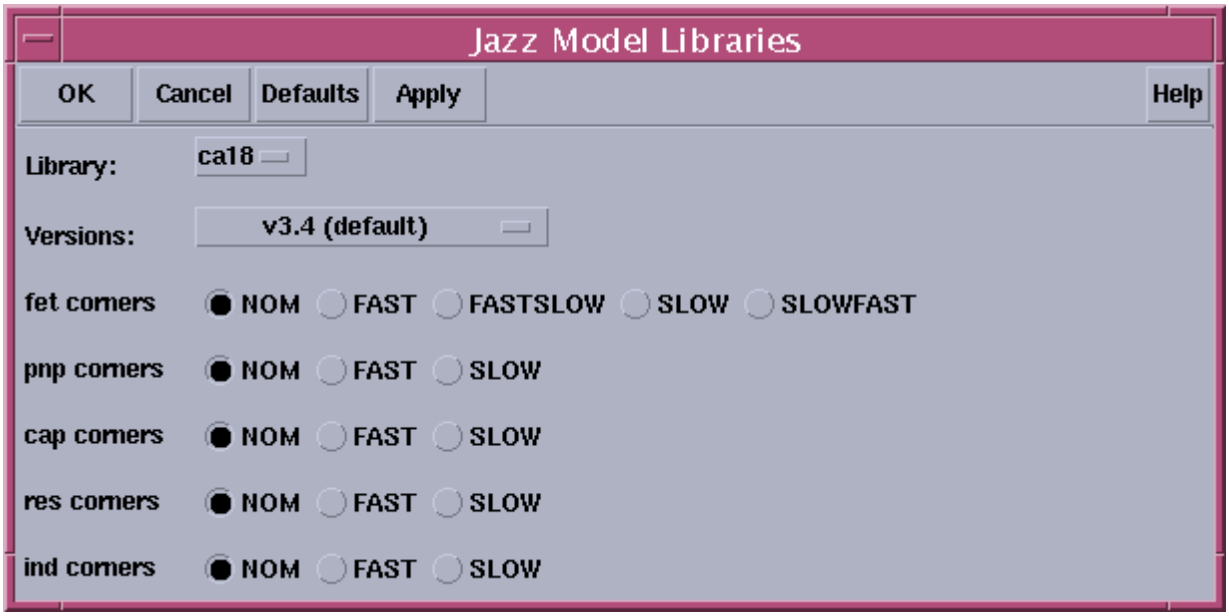
## 6.11. Analysis of Process Variation

The Jazz menu includes a simple utility for setting up to use models at corners other than nominal process conditions. If you wish to simulate at a single process corner (not sweeping through corners), you can use the simple interface as follows from the simulation window. The more sophisticated Corner Analysis tool allows you to setup a set of corner analyses and inspect the impact on circuit performances in tabular or waveform forms. You can further define other attributes such as temperature and design variable settings to define the conditions of each corner. Monte Carlo simulations of process and mismatch allow provide more realistic results by simulating the random interactions of manufacturing variation.

## 6.12. Jazz Menu in Analog Artist:




- **Set Active Library Menu:** Specify version on model library and the desired combination of process corners. For Monte Carlo Simulation, version is set to statistical.




- **Add IE Vars for spectreVerilog:** Adds interface elements to the list of Design Variables for spectreVerilog cosimulation.

# 7. Schematic-driven layout environment - Virtuoso XL

Note that the Virtuoso Layout Editor and Virtuoso XL have many capabilities, many of which will not be covered in this overview. You should consult the Cadence on-line manuals and/or sign up for Cadence training to learn more details about these capabilities.

## 7.1. What is VirtuosoXL?

- VirtuosoXL (VXL) is a layout accelerator built on top of Virtuoso.
- VXL is a schematic-driven layout tool.
- VXL provides a link between schematic and layout views.
- VXL can generate all the components (devices & pins) captured in the schematic, keep the connectivity information.
- VXL uses a built-in extractor, which extracts the connectivity in the layout.
- VXL allows cross-probing of pins, nets, and components between schematic and layout.
- VXL allows updating components or parameters from schematic to layout, and vice versa.



## 7.2. Things Important to VXL

- Instance's Name

  Instance name in layout must match the instance name in schematic. Note that layout instance names have a "|" prefix before the actual instance name.

  - Subcell's Pin

  Subcell layout must have pins same as subcell symbol.

  Pin can be on drawing layer or pin layer. (In display, turn on instance pin, then the pins can be viewed from upper level).

  - Layout cell name

    By default VXL is looking for the layout from same library, same cell name as the symbol. If no layout is found, VXL will flatten the schematic, put lower level layout cells. By adding the following property to schematic instances, the mapping can be redirected:

    Property Name: **lxUseCell**
    Property Type: **string**
    Value: **libName cellName**

## 7.3. Creating a layout from a schematic

- Open schematic, in Composer, Tools->Design Synthesis->Layout XL.
- Select **Create New** in the following popup form, then **OK**. Click **OK** in the following **Create New File** form.

- The tool will open an empty layout. In Virtuoso, **Design->Gen from Source**. In the **Layout Generation Options** form, pick the right metal layer, and size for each pin. You can also set defaults, and **Apply Pin Defaults**. To turn on Create Labels Click **Set Text Style**, then select **Pin Layer** in the popup. (The tool will generate label using the layer as pin shape you can fix the cell height by selecting **Aspect Ratio** button.
- The tool will generate all the devices and pin placed in schematic. There will be a blue box showing the expected area. All the devices and pins are placed below the blue box.

## 7.4. Schematic/Layout cross-reference utilities

- Show incomplete nets (flight lines):
  **Connectivity->Show Incomplete Nets**.
  Select the net(s) you want to view or click on **All** button.
- You can also hide flight lines using:
  **Connectivity->Hide Incomplete Nets**
- Cross-probing between schematic and layout:
  **Connectivity**->**XL Probe**, then select instance, pin, instance's pin, or shape in layout, in schematic it will show the corresponding one, and vice verse.
  To exit probing mode, click in any empty location, and hit **Esc**.
- Update components and nets.
  In VirtuosoXL, **Connectivity->Update->Components and Nets**. The tool should add the extra components to your layout. If a component is deleted from schematic, layout won't delete it, but put a big X on that device.
- If instance cell name is changed (nfet -> n3p3fet), you should use update components and nets for update.
- Update parameters.
  In Composer, select the devices whose parameters have been changed. In VirtuosoXL, **Connectivity**->**Update**->**Layout Parameters**. The tool should popup an Info form tells how many parameters have been changed.
- Verify Parameters
  In Virtuoso XL, **Connectivity->Check->Shorts & Opens**
  **Connectivity**->**Check**->**Against Source**, check to see if parameters are matched.



Re-using existing layouts in VXL

- In library manager, copy only schematic view.
- Make changes on schematic.
- Use Virtuoso XL to create layout from schematic, and generate all the devices without I/O pins and prBoundary.
- In Virtuoso XL, switch to layout mode: **Tools->Layout**
- **Jazz->Misc->Copy Cells**, specify the source layout name. Turn on routings, placements or both depending on your need. Then **OK**. The command will copy the shapes from the

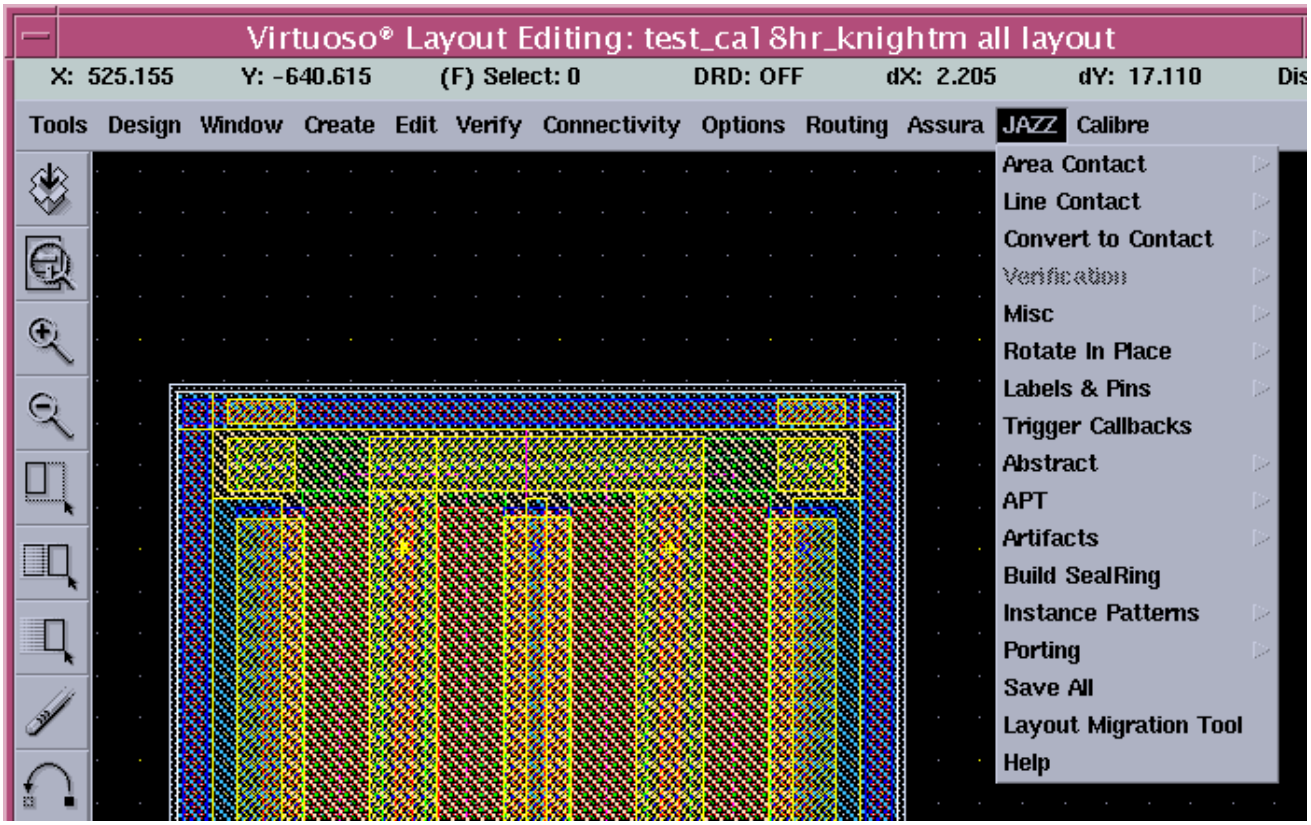
source layout, and move the instances to the corresponding location, if a match (same instance name) can be found in the source layout.

- Switch back to layout XL mode: **Tools->Layout XL**. The first time will take some time depending on the size of the cell.

## 7.5. Checking for potential problems before running VXL

- CIW->JAZZ->Virtuoso-XL Cell Check
- If you have trouble running VXL, run this command on a particular cell. The command checks instance name match, connectivity match.

## 7.6. Jazz menu on Layout Window




- **Area Contact, Line Contact:** These commands will create contacts. For the line contacts, they are ROD contacts in which the line widths and subparts can be changed in the option form (popup by F3).
- **Convert to contact:** Converts existing contacts to contact arrays – does not convert arbitrary shapes to contact. Useful for fixing contact spacing / sizing for layout migrations.
- **Verification Submenu:** When the design kit still supports Jazz Calibre using the Jazz custom netlister and flow, these verification menus are used to run verification and are documented in sections 9, 10 and 11.

- **Misc Submenu:**




- **Object Size:** This function reports the x, y dimension, perimeter and area for selected objects.
- **Zoom To XY:** Enter the comma separated x, y location in the pop-up form to zoom to that location.
- **Rel Copy:** This function copies selected objects to a relative x, y location. Enter the relative x, y, number of copies, and the direction if multiple copies are desired.
- **Rel Move:** This function moves selected objects to a relative x, y location.
- **Rel Stretch:** This function stretches selected objects to a relative x, y location.
- **Rel Move Origin:** This function moves the cell origin to a specified x, y location.

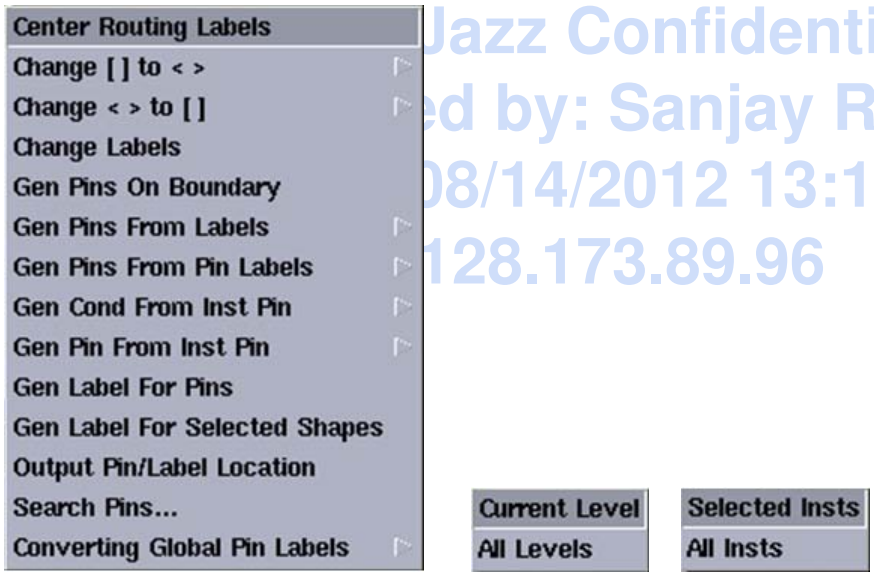
- **Copy To Layers:** This function will copy selected shapes to specified layers at same location.
- **Align Instances:** This function aligns a group of instances along selected left edge, right edge, bottom edge, top edge, center or origin of instance. A form pops up, select the align edge, align direction and optional spacing. If you select not to apply spacing, the instances will keep the original spacing. After you click OK or Apply, you will be prompted to click on the instance to be aligned to.
- **Select By Intersection:** This function selects every selectable object that intersects with the specified drawing area.
- **Set Snapsize:** This function allows you to change the settings (found in the display options window) for the snap size, and the major and minor grid size. The changes are not saved to the next session.
- **Snap:** This function snaps objects in the cell(s) to grid on the specified snap size, by default the minimum manufacturing grid for the process. Options can be toggled to snap just the current cell or through hierarchy and to save or not save.
- **Create Bus:** This function creates a bus on the entry layer using specified width, spacing, and number of wires. After clicking OK or Apply, go to the layout window and click the first point for the bus. This will form the lower or left wire corner, with any other wires will be added to the top or right. So the bus should be drawn in counterclockwise direction.
- **Create Bus Label:** This function creates an array of labels on text layer. A form pops up, enter key label, start index, end index, index step, and select the bracket. After you click OK or Apply, you will be prompted to enter a line to specify the locations of first and second labels.
- **Copy Cell:** This command allows user to copy routing from a source cell, and put placements at the same location if they have the same instance name.
- **Replace Instances:** This command allows user to search instance matching input search libName/cellName/viewName, replace with new libName/cellName/viewName.
- **Gen Characters:** This function generates specified text on the layers you specified. Enter the text, magnification, and select the layers you want to put the text. After you click OK or Apply you will be prompted to click at a point to place the text. Please see the Jazz-> Artifact menu for more complete artifact generators such as copyright and part number generators. This function does not add the artifact marking layer which is needed if the reduced set of design rules for artifacts are desired. Don't generate text on contact or via layers directly. The contacts and vias need to be in arrays with a specified size and spacing. The artifact generator does this automatically, or you can create symbolic contact in the area, flatten them, and delete unnecessary contacts to make it look like characters. All characters must pass the design rule checks.
- **Find Ref Cells:** This command will find referenced cells in place and routing blocks from search path. User will be prompted to enter the search order of libraries. Highest

priority should be put at the beginning. Used for streamed in routing data, this function will also report missing cells.

- **Stretch Fet:** This command will stretch the source/drain metal and contact of selected fet(s) if the fet(s) has parameters sourceTopStretch, sourceBotStretch, drainTopStretch, drainBotStretch. The command pops up a form with following fields: Stretch Left (Right): on/off, No. of Left (Right) Cts: integer (negative number means full contacts), Start From: bottom, center or top.
- **Modify Path Points (Shift+g):** This command allows the user to edit layout paths. First select the desired path to edit and then execute the command (Shift+g) to perform one of three operations, delete point, undelete point, or modify path. The user can also specify the starting path direction, either forward or backward.

➢ **Rotate In Place Submenu:** Rotate selected device in place relative to original position, i.e. each object rotates based on its own center. Select from R90, R180, R270, MX, MXR90, MY, MYR90.

➢ **Labels & Pins Submenu:**






- **Center Routing Labels:** This function will center all the metal labels on the current cell to the center of each routing path. This will help routing tool to recognize the connecting points.
- **Change [] to <>/change <> to []:** These functions search and replace characters [] with <> and vice versa.
- **Change Labels (customized search and replace function):** This command will replace existing labels based on user input. User will be prompted to enter the current label, new label, match whole word, beginning or any. Bus labels are treated as though they don't index. For selection “whole word”, the tool will replace the labels matching

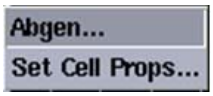
exactly with the searching pattern. For “beginning”, the tool will replace the labels starting with the searching pattern. For “any”, the tool will replace any labels partially matching the searching pattern.

- **Gen Pins On Boundary (works for VXL cells):** This function creates a pin wherever a shape which has connectivity information intersects with cell boundary and the net name is a pin in schematic. A cell boundary can be defined by "outline" or “prBoundary". If no boundary has been defined, the cell extension will be used. To check if a shape has connectivity information, query (Q) the shape, see if you can click on "Connectivity". Normally, if the data has been saved in DLE/VXL environment, a connected shape should have connectivity information. The pin has same size as the intersecting edge.
- **Gen Pins From Labels (works for stream in gds data, not VXL cells):** This function creates a pin wherever a label origin overlaps a polygon on same layer and the label name is a pin in the schematic if the schematic exists. This function will generate pins through hierarchy. This function is useful to create pins for layout data created in non-Virtuoso environment.
- **Gen Pins From Pin Labels (works for stream in gds data, not VXL cells):** This function creates a pin wherever a label on pin purpose layer overlaps a polygon on same layer and the label is a pin in schematic if the schematic exists. This function will generate pins through hierarchy. This function is useful to create pins for layout data created in non-Virtuoso environment, and the connector information is created on special layers, e.g. layer 88, 89, 90, 91, 92...
- **Gen Cond From Inst Pin:** This command will generate conductors from instance pins. The generated conductor has the same shape as instance pin. If the instance has net name(s) associated with (in layoutXL situation), then a proper label will be generated on the created conductor using same instance pin layer. If the instance has no net name associated with it, then the instance pin name will be generated on text layer.
- **Gen Pin (top level) From Inst Pin (for VXL cells):** This command will generate pins from instance pins. The generated pin has the same shape as instance pin. New pin name is coming from the current level. If the instance pin doesn't have connectivity information, no pin is generated, only conductor is generated.
- **Gen Label For Pins:** This command will generate labels for pins where no label is present.
- **Gen Label For Selected Shapes:** This command will generate labels for the selected shapes if the selected shape has connectivity information. The label will be generated at the center of a rectangle, at one end of a path, or one corner of a polygon. If a label is found at those locations, no label will be generated.
- **Output Pin/Label Location:** This command will generate a report on the location and layer of existing pins or labels, also report missing labels in layout if the corresponding schematic exists.

- **Search Pin:** This command will find the pins matched with entered pin name, and add them to select list.
- **Convert Global Pin Labels:** This command will put global pin name like "vss!" on "text" layer and add a metal label like "vss" on top of the pin.

➢ **Trigger Callbacks:** Same as schematic version. Triggers callbacks for all CDF parameters in cellview or throughout hierarchy. Default is dry run, which will not change anything, only run through and display any error messages.

➢ **Abstract Submenu:** Obsolete.



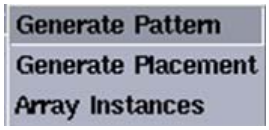

➢ **APT Submenu:** Used in custom R&D work to generate test structures and corresponding index files.




➢ **Artifacts Submenu:** Artifact generators are provided as a convenience. In some cases the artifacts will require hand editing in order to pass all of the design rules. See the README for additional details about these functions.


➢ **Build Seal Ring:** Obsolete. Some processes offer a seal ring pcell in the process design library, but in general the seal ring is added by Jazz during mask generation.

➢ **Instance Patterns Submenu:** These functions are useful for layout migration.

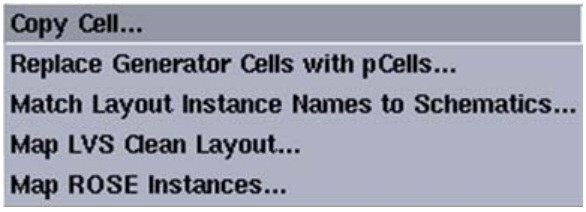



- **Generate Pattern:** This function can be used to create a list of all instance names and their rotations and placements by selecting “Absolute Placement.” There are also

options to generate lists of instance names with just rotation, instance names with rotation and row designation, and bus names.

- **Generate Placement:** This function reads in the list of instance names and rotations and placements and applies them to the selected layout.
- **Array Instances:** This functions takes all the instances in the selected layout and arrays them with specified spacing.

➢ **Porting Submenu:**




- **Copy Cell:** This function allows routing to be copied from a source cell, putting placements at the same location if they have the same instance name.
- **Replace Generator Cells with pCells:**
- **Map LVS Clean Layout to layoutXL compatible layout:** This function allows routing to be copied from a source cell, and puts corresponding placements at the same location even though they may not have the same names. Routing and placement source cells don't have to be the same, i.e. you can move the placements according to one source cell, copy the routings from another source cell. This command can be used to build a layoutXL compatible cell from a layoutXL incompatible but LVS clean cell. To use this function, first, run LVS on source layout using option "-ixf." Calibre LVS will generate a spice file for the layout, and *.ixf file for schematic and layout instance mapping. Second, create a new layout, which has placements (no pins, no boundary) generated from layoutXL. Third, use the function “Map LVS Clean Layout,” entering the correct source cell, spi file, and ixf file. If only placements are needed, turn off "copy routings & pins". Routings can be copied from another cell.
- **Map ROSE Instances:** This command allows user to recover the instance names from ROSE data (*.rsf). User needs to input the path for rsf library, the rsf cell name and gridsize information. This command has hierarchial option.



➢ **Save All:** This command will save all the modified cells in the hierarchy **and** display all the saved cell names in the CIW.

➢ **Layout Migration Tool:** For a few processes, specialized layout mapping utilities may be available.

➢ **Help:** Documentation on all the customized functions. You can also print the following file: $RDS_ROOT/cdsware/layout/bin/txt/rssLayHelp.txt.

## 8. Verification

Jazz supports Calibre and Assura verification for most processes, including DRC, LVS and parasitic extraction. The Calibre support is gradually being migrated from a Jazz custom Calibre flow to Calibre

Interactive flow. For more information on tool version support, please see the design kit installation document. Sections 9, 10 and 11 describe how to use the Jazz custom Calibre flow. For documentation of the Assura and Calibre Interactive flows, please see the Jazz menu in the CIW.

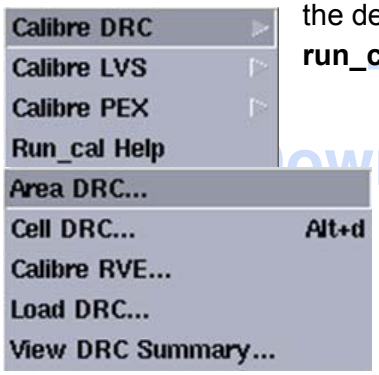
# 9. Calibre DRC from Jazz Menu

When Jazz Calibre is used, the Jazz Design Kit has a GUI to run Calibre DRC from within the Cadence layout environment. When the Calibre Interactive or Assura flow is used, the check is run from the Calibre or Assura menus, respectively, and this section does not apply. The documentation for these flows is available in the CIW Jazz menu.

## 9.1. Running DRC

To bring up the main GUI for Calibre DRC, select the following layout window pull down menu item:
**JAZZ -> Verification -> Calibre DRC -> Cell DRC...**

This brings up the **Calibre DRC** form. There are several options presented in the form, but typically the default options are acceptable. When you OK this form, a Perl script called **run_cal** is started as a foreground process.






The form allows the DRC job to be run in either the background (default option) or foreground. If you run the job in the foreground you must wait until the job completes before you can do anything else, while running in the background allows you to do other tasks in the Cadence session while the job is running. When the job completes, a dialog box pops up indicating whether there were any errors found or not.

The options for **run_cal** are documented in the **Run_cal Help** menu**.**

## 9.2. Viewing DRC Summary File

After running Calibre DRC, you can view the results file generated by Calibre from within the Cadence environment by selecting the following menu item:
**JAZZ -> Verification -> Calibre DRC -> View DRC Summary...**

## 9.3. Debugging violations using RVE

If DRC violations were flagged in your layout, you can use the Mentor Graphics' RVE tool for graphically finding and debugging problems with the layout. You can start the RVE tool automatically pointing to the correct summary file by selecting the following menu item:
**JAZZ -> Verification -> Calibre DRC -> Calibre RVE...**

When the tool appears, all individual DRC rule checks are presented in a tree format. If there was a violation of a particular DRC rule, it will be colored red and will also indicate how many violations of that rule exist in your layout. You can highlight and zoom to problem areas in the layout view as follows using this tool:

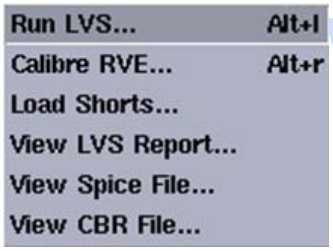
- Expand the violated rule by clicking on the "**+**" symbol. This will show individual violations of this rule as numbered errors.
- Double click on the error to automatically highlight and zoom to the portion of the layout with the violation. Note that the explanation of the DRC rule is given in the **Checktest** section of the RVE form.

## 10. Calibre LVS from the Jazz Menu

When the Calibre Interactive or Assura flow is used, the LVS check is run from the Calibre or Assura menus, respectively, and this section does not apply. The documentation for these flows is available in the CIW Jazz menu.

### 10.1. Running LVS

To bring up the main GUI for Calibre LVS, select the following layout window pull down menu item:
**JAZZ -> Verification -> Calibre LVS -> Run LVS...**




This brings up the Calibre LVS form. There are several options presented in the form, but typically the default options are acceptable. As with the DRC check, pressing OK on this form will start the run_cal perl script. The run_cal options are found in the **Run_cal Help**. As with the DRC check, the form allows the LVS job to be run in the background (default option) or foreground. When the job completes, a view window with the Calibre LVS report will appear.



### 10.2. Viewing LVS Results File

After running Calibre LVS, the results file generated by Calibre is automatically displayed in a view window. You can also bring up the results file manually using:
**JAZZ -> Verification -> Calibre LVS -> View LVS Report...**

### 10.3. Debugging errors using RVE

If LVS errors were reported, you can use the Mentor Graphics' RVE tool as an aid to graphically find and debug problems with the layout and/or schematic. You can start the RVE tool automatically pointing to the correct report file by selecting the following menu item:
**JAZZ -> Verification -> Calibre LVS -> Calibre RVE...**
When the tool appears, all individual errors are presented in a tree format. Use the tool as follows to debug individual discrepancies:

- You can expand details on errors by clicking on the "**+**" symbol close to the error. Once you see a list of discrepancies listed, you can select an individual discrepancy with the left mouse button. This causes the bottom half of the RVE window to display information about this particular discrepancy.
- Note that the left half of the bottom window displays layout information, while the right half displays schematic (source) information. You can double click on a net name or instance name to automatically zoom to the problem area in the layout or schematic.

# 11. Calibre Parasitic Extraction and Re-simulation from the Jazz menu

This capability refers to the ability to extract parasitic capacitance and resistances from layout interconnect and include the effects of these parasitics in simulation. Note that you must have an LVS correct schematic and layout before you can use this capability. Below is a summary of the steps needed to run parasitic extraction and re-simulation. The Jazz menu is not used in the Calibre Interactive flow or the Assura flow, but many of the steps are similar. Additional documentation for the Calibre Interactive and Assura flows is available in the CIW Jazz menu.



## 11.1. Attach labels to layout nets

In order to preserve the appropriate correspondence between schematic and layout after including parasitic elements, you must attach labels to nets in the layout corresponding to pin names in the top-level schematic. Calibre assigns arbitrary names to nets unless you specifically label the nets, so for this flow it is important to preserve the names of the top level nets. This is required in order to be able to perform Coupled C or Distributed RC extractions, but is not necessary for Lumped C extractions. The layout labels must be created on the drawing purpose of the same layer of the net to which it is attached. Note that the origin of the label must be fully covered by a portion of the net. Note that only metal layers can be specifically labeled for Calibre, so if you wish to label nets fully in poly you must use provide a connection to that net through metal, and label the metal portion of the net. Note that the JAZZ layout menu provides many utilities associated with labels.

## 11.2. Extract parasitics from the layout

Use the JAZZ pull down menu from either a schematic or layout view to bring up the Parasitic Extraction form: **JAZZ -> Verification -> Calibre PEX -> Cell Extraction**. Note that you can choose between Lumped C, Coupled C, or Distributed RC options to control how to perform the extraction. Note that Coupled C and especially Distributed RC options can potentially generate many parasitic elements and so it may not be feasible to choose these options for a top level layout at this point in time. If the extraction runs successfully, you should get a Dialog box indicating it was successful and a new view called "extracted" is generated for the cell which

contains all the device and connectivity information from the layout.

### 11.3. Run DIVA LVS

Note that while this step is not essential for re-simulation with parasitics, there are several benefits that make it necessary for most users, the main ones being:

- If you want to be able to plot voltages for nets other than the top level nets connected to the extracted block (i.e. nets further down the hierarchy) you must follow the DIVA LVS flow and create the **analog_extracted** view.
- If you want to back annotate lumped parasitics of nets to wires on your schematic view or print a summary of parasitics you must follow the DIVA LVS flow.
- If you want to filter out parasitics and only include the parasitics on certain nets you must follow the DIVA LVS flow.

In order to return to the Cadence Parasitic Extraction and re-simulation flow, you must re-run LVS using the Cadence DIVA LVS tool. Note that you are not required to perform Diva LVS to re-simulate with parasitics (see comments above). Running DIVA LVS is documented in the Cadence on-line documentation, but the steps are summarized here.

1. Open either the schematic or layout view of the cell
2. Select **JAZZ -> Verification -> Diva LVS...** to bring up Cadence's Diva LVS form pre-filled out with reasonable defaults. Note that you can bring up this form in a different way from Cadence menus, but that will not pre-fill the form with reasonable settings. Make sure that **Create Netlist** is enabled for both schematic and extracted views.
3. Click on the Run button of the LVS form to start Diva LVS. Note that you can view the log file for Diva LVS as the tool is running by clicking on the **Info** button and then choosing the **Log File** button.



Check the Diva LVS log file to make sure that the LVS run completed successfully. Look for the keywords "**The net-lists match**" towards the end of the log file. If Diva does not complete successfully you cannot proceed at this point. Note that Diva LVS may take a very long time to run for very large designs or if there are too many parasitic elements. In some cases because of limitations with DIVA, the LVS job may actually run out of memory and never complete.

### 11.4. Examining extracted parasitics

Once you obtain a successful Diva LVS match you can examine parasitic lumped values if desired by selecting the **Back annotate** button. This brings up **the Parasitic Back annotation** form which allows you to add labels to the schematic nets showing values of the parasitics, or print all parasitic values to a view window. You can also use the **Parasitic Probing** form to display parasitics for nets of your choice or between nets of your choice.

### 11.5. Building "analog_extracted" simulation view

If you wish to simulate the effects of the parasitics using the normal Analog Artist environment you can use the **Build Analog** button to create an **analog_extracted** view of your cell. At this point you

can choose which parasitics you would like to include in your simulation. By default, all parasitics are included, but you can filter out which parasitics you would like to include by adding **sbaLib** library components to your schematic view. Note that adding these components to your schematic view does not affect other types of simulations for your schematic. Use **spcapacitor** to specify to include a net's lumped capacitance to ground in the simulation. Use **spcapacitor2** to add the coupling capacitance between 2 nets. Use **spresistor** to add distributed RC contribution for a net.

### 11.6. Running simulations using "analog_extracted" views

Once you have **analog_extracted** views for blocks in your simulation schematic, you can use them as the simulation views for running Analog Artist simulations, instead of the original schematic views. Note that you will have to use the Hierarchy Editor to choose the appropriate views you would like the netlister to use when generating a netlist. If you would like to include the parasitic effects for certain blocks, set the simulation views of these blocks to **analog_extracted** using the Hierarchy Editor. Please refer to the next chapter or the Cadence on-line documentation for more details on using the Hierarchy Editor.

## 12. Simulation View Partitioning - Hierarchy Editor



The Hierarchy Editor is a tool that allows one to partition a testbench schematic to use different views for netlisting. Thus, for example, if you have both a behavioral view and a schematic transistor level view of a block in the schematic hierarchy, the Hierarchy Editor allows you to select which view to use for this and other blocks that may have multiple simulation views. Similarly, if you have an **extracted** or an **analog_extracted** view of a block obtained by following the Parasitic Extraction and Resimulation flow covered in the previous chapter, you can use the Hierarchy Editor to specify which blocks in your schematic hierarchy should use these extracted views so that you can re-simulate including the effects of extracted parasitics. Note that the Hierarchy Editor use is covered in depth in the Cadence online documentation. The goal of this section is to describe a schematic-driven approach to using the Hierarchy Editor such that in-depth knowledge of how the Hierarchy Editor works is not necessary.

### 12.1. Creating the "config" Hierarchy Editor view

The first step is to create a **config** view for your simulation testbench schematic. This view is used by the netlister to determine how to netlist each view in your schematic hierarchy. Use the Library Manager to create a new view for your schematic testbench. You can use any name you wish for the **config** view, but if you use a name other than **config,** you will have to manually select the **Tool** as **Hierarchy-Editor**. When the **New Hierarchy** form appears, select the Built-in Template as the simulator you are using (normally **Spectre**). Change the **View** value from **myView** to **schematic**, as this is usually the view that you are trying to partition. OK the form. At this point, the main Hierarchy Editor window should automatically update to display the cells and views in your schematic hierarchy. Choose **File -> Save** to save the **config** view and then exit the tool. Note that with a clear

understanding of the Hierarchy Editor, you can define all schematic views within this tool, but the next step describes an easier way of accomplishing this.

## 12.2. Opening a config view schematic

Once you have a config view and wish to use it to define the simulation views to use for the schematic, you should always open schematic view under control of the **config** view. To do this, use the Library Manager to open the **config** view for editing. When prompted, be sure to choose the Top Cell View as the view to open and disable opening of the config view. Once you open the schematic in this way, the window title should display the schematic name along with the **config** view name.

## 12.3. Defining the schematic simulation views to use

Once you have opened a schematic under control of the config view, start Analog Artist environment. Then from the schematic **Hierarchy-Editor** pull down, select **Set Instance Binding** menu item and choose **Yes** when prompted to open the **config** view in edit mode. This will bring up the Hierarchy-Editor window as well as a **Set Instance Binding** form. Use this form to define the views to use for blocks in your schematic hierarchy by clicking on an instance (block) in the schematic and choosing the appropriate **View Binding** to use for the instance. Thus, for example, if you had an instance **I10** of a block called **nd02d1** which has several views including **schematic** and **analog_extracted**, click on instance **I10** in the schematic and choose the simulation view you wish to use for the block - either **schematic** or **analog_extracted**. Select the **Apply** button on the form to apply your change to this instance. Note that if you don't set a specific binding for an instance, default bindings will be used, which is normally the **schematic** view if it exists, or else the **Spectre** view if the simulator is **Spectre**. Once you have finished defining the views to use, select **View -> Update** from the Hierarchy Editor and then save the config view using **File -> Save**. Alternatively, you can perform an update and a save by clicking on the red button on the top of the Hierarchy Editor window.



You can use **Hierarchy-Editor -> Show Views Found...** menu entry from the schematic to display what views will be used for each block in your schematic. This will highlight blocks in different colors according to which view name will be used, and labels also identify the view name to be used for each instance in your schematic. Once you have finished choosing your simulation views, you can exit the Hierarchy Editor tool. Then proceed to use Analog Artist the usual way.

If you configured things properly you should now be able to plot any voltages in the hierarchy from your schematic view, and they will include the effects of the parasitics. It is important to note when traversing hierarchy to select a net to plot, always choose the "schematic" view when descending, not the **analog_extracted** view. You should also notice that when you select an internal net that an "X" will appear on the nearest terminal of the net you selected. This is another indication that the plots will include the effects of parasitics on that net.

## 13. Creating GDS files

To create a GDS file from a layout view, you can use the Cadence Stream Out form. To access this form, from the CIW choose **File -> Export -> Stream...** menu entry. Click on the Help button on this form for details of all the options for stream out.
Similarly, you can import a GDS file to a layout view using the Cadence Stream In form, which can be accessed from the CIW by choosing File -> Import -> Stream... menu entry. Again, click on the Help button on this form for details of all the options for stream in. It is recommended to create the target library first and attach it to the right technology before beginning the stream in. Then, click on the **Options** button and enable **Skip Undefined Layer- Purpose Pair** Boolean in the **Stream In Options** form.