cadence®

Virtuoso Liberate AMS Reference Manual

Product Version 16.1
March 2017

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

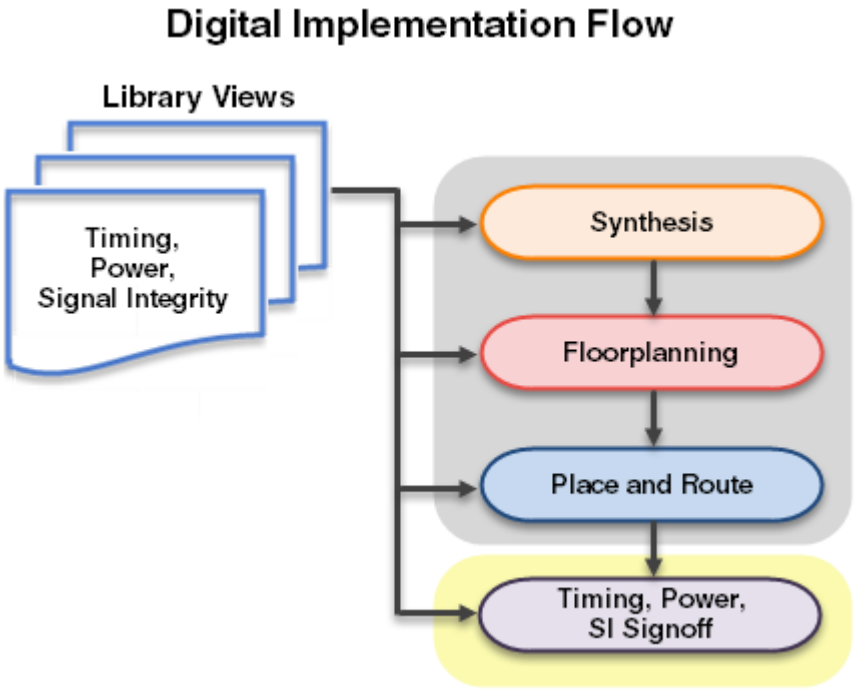
Contents

Preface

Introduction to Characterization

The Role and Importance of Libraries

Creation of electrical views is a prerequisite for any digital design flow. The electrical information stored in the library views is used throughout design implementation from logic synthesis, through design optimization to final signoff verification. Accurate library view creation is essential to ensure close correlation between the design intent and the final silicon.

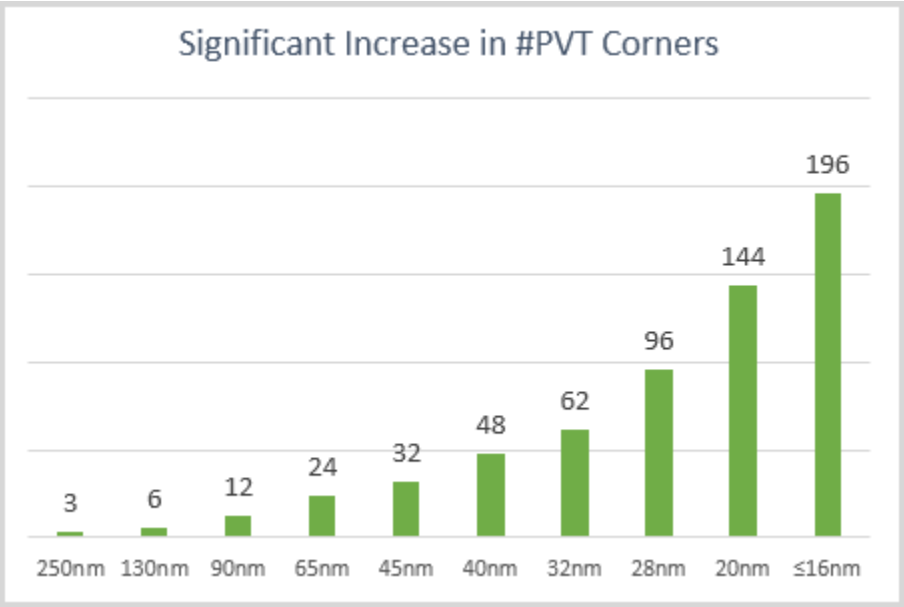


A Growing Problem

In nanometer geometries (65nm or below), the required number of library views is growing dramatically because of issues related to power leakage and process variation. To minimize

power leakage at deep submicron nodes, we see process variations such as LVT, RVT, and HVT (low/regular/high voltage) being utilized. For example, to manage power at 65nm, it is common to have library cells with two or three different threshold values (high threshold to reduce leakage power, low thresholds to improve performance), and to use two or more on-chip supply voltages. In this scenario, the number of views needed for 65nm will be six times greater than what is needed for 130nm.

The figure below shows the growing trend that requires PVT corners to accurately model the circuit behavior:

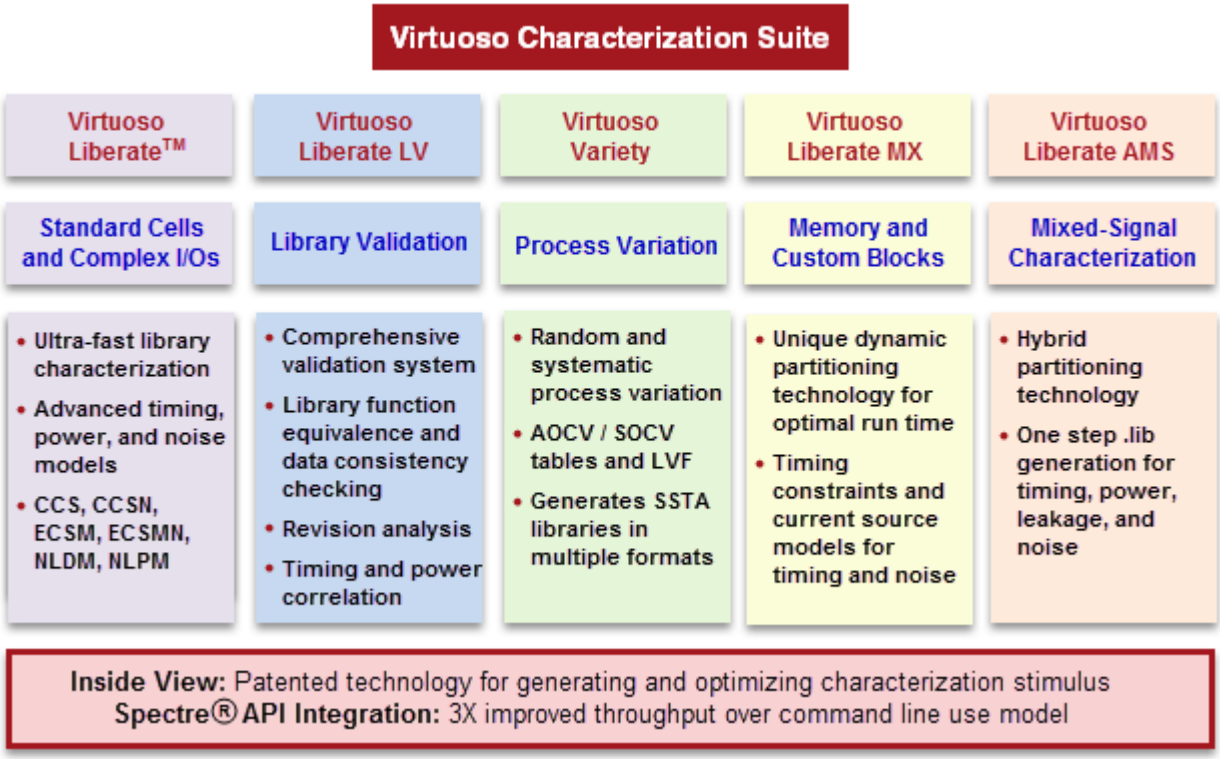


In addition, library views require more advanced models like:

- Current source models CCS and ECSM
- Statistical models – AOCV/SOCV/LVF
- Netlist extraction at various temperatures for Nanometer Process Nodes
- Support multiple foundries to assure flexibility for yield issues
- Support for many more functional designs – 1000+ STD cell, I/O, custom datapath, memory and Analog IP

Virtuoso Characterization Suite

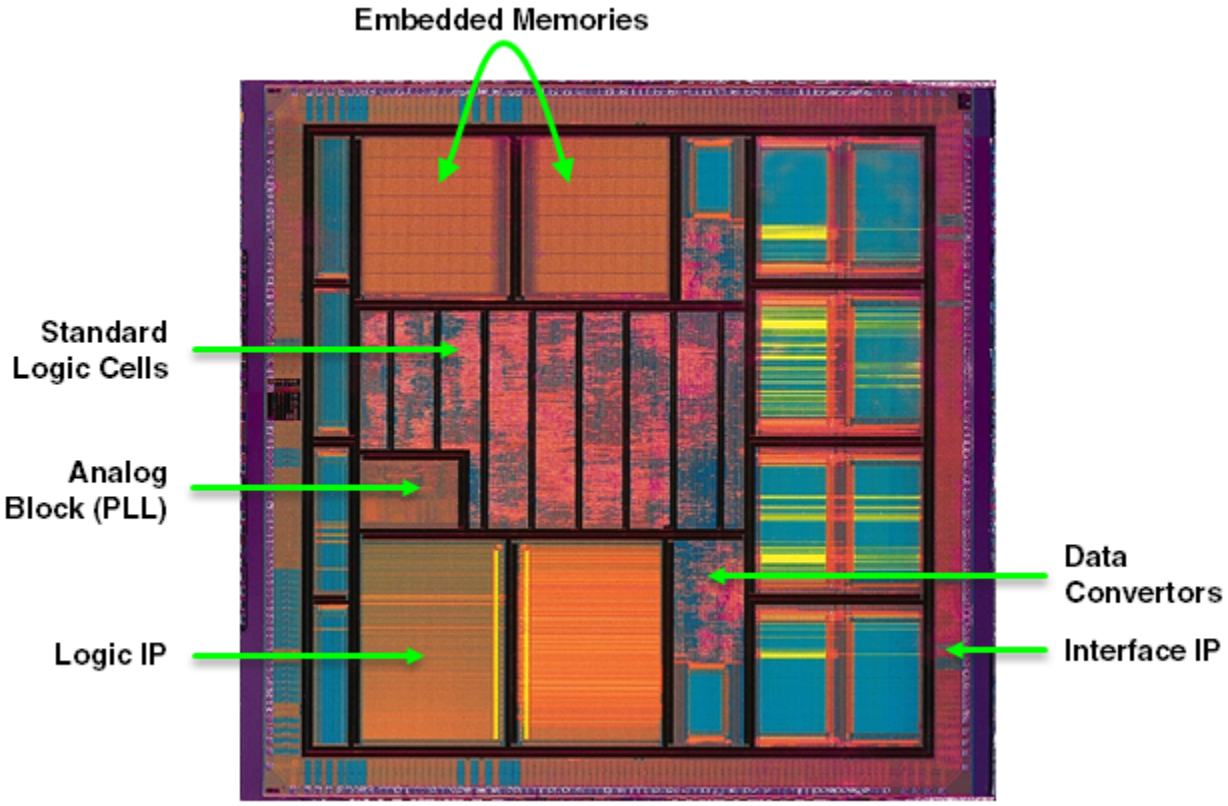
To address all the challenges, Cadence offers Virtuoso® Characterization Suite that covers the complete portfolio of characterization solutions given below:



The Virtuoso Characterization Suite intends to provide highly efficient and automated electrical view creation and validation for all IP blocks that include the following:

- Logic and I/O cells (GPIO, PCI, SSTL, PECL, and so on)
- Embedded memory (SRAM, ROM, Register files, CAM, and so on)
- Custom digital blocks (custom cells, datapath, cores, and so on)

- Interface IP and analog blocks (USB, Serdes, DDR, and so on)



System Requirements

Liberate, Variety, Liberate MX, Liberate LV, and Liberate AMS run exclusively on Linux operating system. The following table lists the supported platforms:

Architecture	Development OS	Supported Environments
x86_64 (32/64)	RHEL 5.5	RHEL 6
		SLES10
		SLES11

For detailed information about the requirements, see Computing Platforms.

Software and Licensing Requirements

- LIBERATE 15.1

  The following table lists the required server and client product numbers for each product in the Virtuoso Characterization Suite:

Product Name	Server Product Number	Client Product Number
Liberate	ALT110	ALT111
Variety	ALT210	ALT211
Liberate MX	ALT410	ALT411
Liberate LV	ALT610	ALT611
Liberate AMS	ALT810	ALT811 or ALT812

- MMSIM 15.1

Product Name	Product Number
Spectre XPS	91600 or 90004
Spectre APS	3500 (restricted for characterization), 91050, or 90004

About This Manual

The *Virtuoso Liberate AMS Reference Manual* describes the Cadence® Virtuoso® Liberate AMS tool. The document includes opening chapters that describe what Liberate AMS does and how to get started with the tool. Later chapters discuss the commands and variables that can be used with Liberate AMS.

Audience Profile

This manual is aimed at developers and designers who want to work on library creation for Analog Mixed Signal (AMS) macro blocks. It assumes that you are familiar with:

- SPICE simulations

- Basic expected behavior of the design being used

Additional Documents for Reference

For information about known problems and solutions, see *Virtuoso Characterization Suite Known Problems and Solutions*.

For a list of new features in a release, see *Virtuoso Characterization Suite What's New*.

For information about other products in Virtuoso Characterization Suite, refer to the following manuals:

- *Virtuoso Liberate Reference Manual* describes the Liberate tool—an accurate, highly efficient and easy-to-use library characterizer that creates electrical views (timing, power, and signal integrity) in formats such as the Synopsys Liberty (`.lib`) format.
- *Virtuoso Liberate LV Reference Manual* describes the Liberate LV library validator—a tool that provides a collection of capabilities used to validate and verify the data consistency, accuracy, and completeness of cell libraries.
- *Virtuoso Variety Reference Manual* describes the Virtuoso Variety process variation cell characterizer—a tool that characterizes process variation aware timing models and generates libraries for multiple statistical static timing analyzers (SSTA) without requiring re-characterization for each unique format.
- *Virtuoso Liberate MX Reference Manual* describes Liberate MX—a tool that provides library creation capabilities to cover memory cores.
- *Virtuoso Liberate API Reference Manual* describes a Tcl interface that allows access to the Liberate characterized Library DataBase (LDB).

Rapid Adoption Kits

Cadence provides Rapid Adoption Kits that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

Demos and Videos

To learn how to characterize mixed signal blocks using Liberate AMS and generate a Liberty library (.lib) to enable timing, power and noise signoff for a full chip including mixed signal blocks, see *Liberate AMS Demo*.

Typographic and Syntax Conventions

This section describes the typographic and syntax conventions used in this manual.

`literal`	Non-italic words indicate keywords that you must enter literally. These keywords represent command or option names.
`argument`	Words in italics indicate text that you must replace with an appropriate value.
`< >`	Angle brackets indicate text that you must replace with a single appropriate value. When used with vertical bars, they enclose a list of choices from which you must choose one.
`|`	Vertical bars separate a choice of values. They take precedence over any other character.
`-`	Hyphens denote arguments of commands or variables. Usually arguments denoted in this way are optional but, as noted in the syntax, some are required. The hyphen is part of the name and must be included when the argument is used.
`{ }`	Braces indicate values that must be denoted as a list. When used with vertical bars, braces enclose a set of values from which you must choose one or more. When you specify a list, the values must be enclosed by either quotation marks or braces. For example, `{val1 val2 val3}` and `"val1 val2 val3"` are legal lists.

Some argument are positional and must be used in the order they are shown. Any positional arguments that are used must be given after any arguments denoted with hyphens.

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

  Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit: https://www.cadence.com/support

- Log on to Cadence Online Support

Customers with a maintenance contract with Cadence can obtain the latest information about various tools at: https://support.cadence.com

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

1

Overview of AMS Characterization

The Digital Implementation flow requires the characterization of top-level AMS blocks. The timing, power, and pin capacitance information is captured in a Liberty format file.

Contents of Liberty Format File

The characterization information captured in a Liberty format file includes the following:

- **Pin Information**

  The Liberty file contains information about the external characteristics of the input and output pins, that is,

  - Pin capacitance for input pins
  - Noise immunity for input pins
  - Holding strength for output pins

- **Timing Information**

  The Liberty file contains information about the timing relationships of the pins of an instance. Within the Liberty file, the following timing arcs are captured:

  - Delay arcs for input to output timing
  - Setup and hold arcs of input pins related to clock
  - Minimum pulse width (MPW) for clock pins
  - Timing arcs for an Internal node
  - Minimum period

- **Power Information**

  The Liberty file contains information about the static and dynamic power consumed by the memory instance.

- Static leakage for each power supply
- Dynamic power for each power supply resulting from events on each input

Pin Characterization

As part of the Liberty file, each input pin is characterized for pin capacitance. Optionally, the pins can also be characterized for noise characteristics.

Pin Capacitance Characterization

Each input pin should be characterized for pin capacitance. This will be used to determine the total loading of the driving signals. Simulations should be run to determine the equivalent capacitance value based on the early stages of the circuit.

Noise Characterization

In certain advanced Liberty models such as CCSN and ECSMN, it is necessary to model the noise characteristics of the input and the output pins. This includes noise immunity of the input pins and holding strength of output pins.

Methods and techniques of noise characterization will not be covered in this manual.

Timing Characterization

The purpose of the timing characterization of any AMS instance is to capture the timing relationships of the instance pins to other pins and themselves.

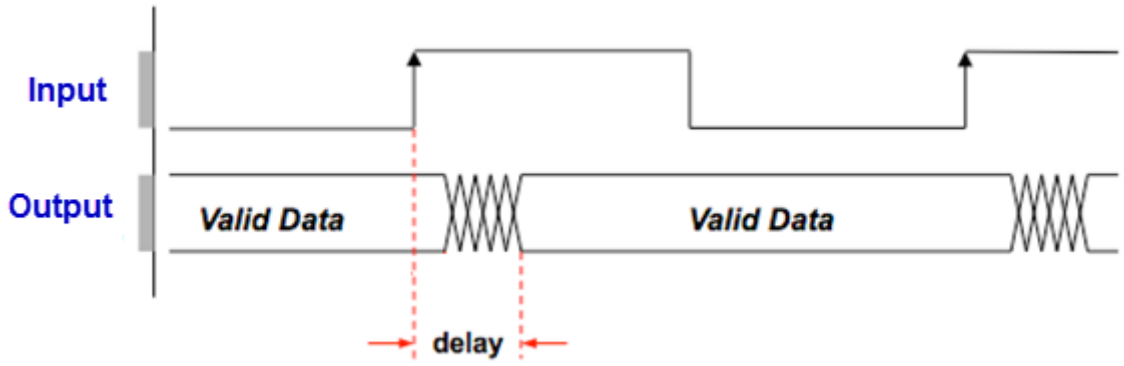
Delay Arc Characterization

The delay arcs describe the timing relationship between:

- The input and output pins
- The input and an internal node
- The internal node and output pin

These timing arcs are used when an event on an input pin results in an event on the output pin.

The delay arc describes the worst case time for the new output data to be available. The delay describe the interval of time for which the output can be expected to be valid.



This is the result of multiple internal timing paths leading to the output and the bus nature of the output.

When creating input stimulus for delay arc, it is important to capture the slowest paths to ensure the correct data delay is captured for delay. This means that for each output direction, rise and fall, there should be vectors that cover the following:

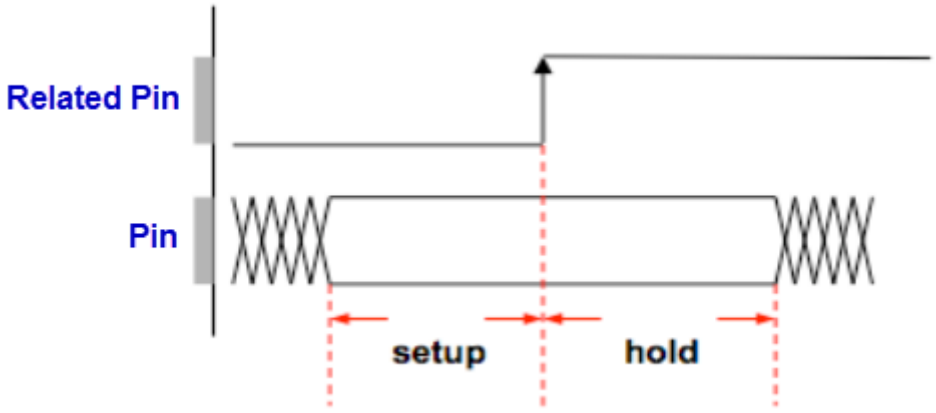
- Worst case outputs switching
- All functional modes that can result in an output event

The delay values should be characterized for all input slew values and output load values.

Constraint Arcs Characterization

The constraint (setup and hold) arcs describe the required timing relationship of a pin against a related pin, usually a clock. The setup time describes the duration for which a related pin must be stable ***before*** triggering the clock edge to transition a pin. The hold time is the duration for which the related pin should be stable ***after*** triggering the clock edge to transition a pin. Both the setup and hold times are allowed to be negative; in this case, the specified transition is allowed to occur on the opposite side of the related pin event. The setup and hold

together describe an interval of time surrounding the related pin event for which the pin is required to be stable.



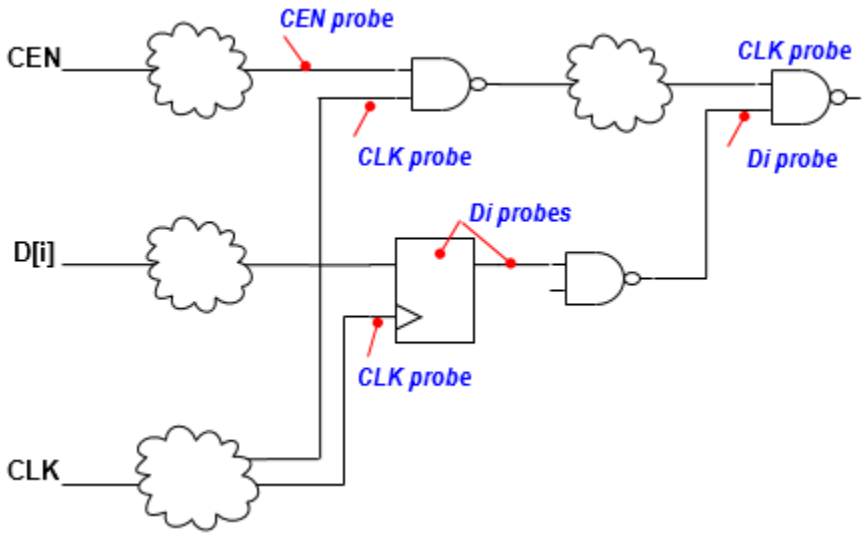
The setup and hold times need to be characterized for all values of pin and related pin input slew.

A valid setup time ensures that a pin's proper value for a cycle arrives before the clock at all locations where the clock propagation is dependent on the state of the input pins. In addition, it ensures that the proper value is stored in the input latches. The clock must not be allowed to propagate for a path that is reliant on a different state of the input.

A valid hold time ensures that a new value for the input pin will not corrupt the existing cycle. In this case, if the hold time is satisfied, the clock will prevent the propagation of the input and prevent it from corrupting the cycle.

The most common production method for characterizing setup and hold time of large AMS block is through the path delay method. In this method, the delay is measured for each pin and related pin, and this information is used to compute the required relationship of these two pins at the pin level. The circuit locations where the pin and related pin intersect are identified

and the delay is measured from the pin and related pin to their corresponding probe locations. The difference of these delays is the setup or hold time.



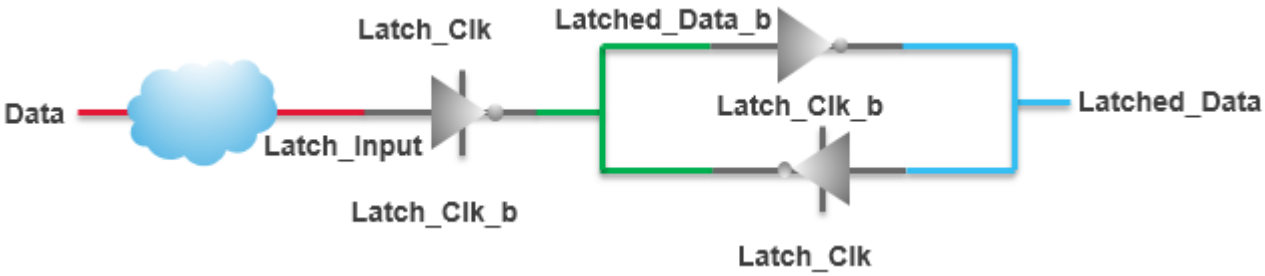
If the first stage is transparent during the situation that is being characterized for, then the following stage should be characterized as well. Typically, the first stage will be a latch and subsequent stages will be some form of combinational logic. In the case of setup time, the latch will be transparent when the setup is exercised, so the next stage should be considered as well. For hold time, when hold is exercised, the latch should be closed. So, if hold is satisfied at the latch it is not necessary to monitor later stages.

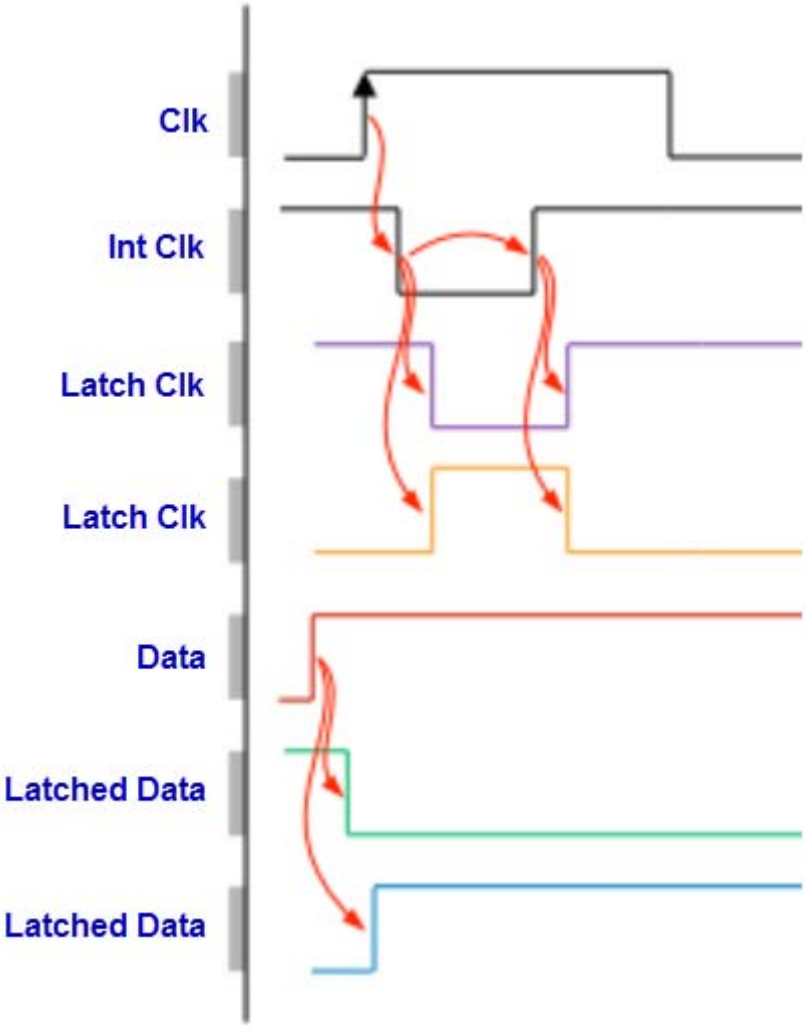
All valid probing locations should be measured and the worst case value should be captured in the Liberty file.

For pins that influence the clock propagation it is important to use the clock propagation for the correct state of the input pin. The clock propagation for signal high should be used for setup rising and hold falling. The clock propagation for signal low should be used for setup falling and hold rising. This is most important for signals that can change the clock propagation, such as, a chip enable pin.

Computing Setup and Hold Times using the Path Delay Method

As an example, let us review the computing of setup and hold times at the input latch.





For setup time, you need to ensure that the latched data is fully transitioned before the latch closes.

```
Setup Time = Max (Data   Latched_Data_b, Data   Latched_Data)
               -  Min(Clk   Latch_Clk_b, Clk   Latch_Clk)
```

For hold time, you need to ensure that the input to the latch does not switch until the latch is closed.

```
Hold Time = Max (Clk   Latch_Clk_b, Clk   Latch_Clk)  -  Latch_Input
```

For hold time, if the timing is satisfied at the input latch, the new data will not propagate further into the design. This means that it is sufficient to probe for hold only at the latch.

Setup and Hold Times for Bus Inputs

In many designs, buses are the input pins of an instance.

Within the Liberty syntax, the timing information is allowed under the bus or the individual bits. When the timing information is under individual bits, all bits can have the same data or each individual bit can have specific data. There can be limitations in the downstream tools to fully optimize the individual bit timing. Optimizing data by bits can be costly in terms of high simulation time. Therefore, the most efficient solution is to use the same data for all bits of the bus.

If the same data is to be used for all bits of the bus, it is necessary to identify the worst case of each bit and use that data for all bits. When doing this it is important to use the clock and data paths from the same probing location.

Tip

Do not use the clock path from input[x] and the data path from input[y].

Bisection Method

A less common approach to AMS characterization is through the use of bisection methods. In this case, the pin and related pin timing relationship will be adjusted until the point of failure to determine the worst case arc value.

There are two possible criteria for determining whether an external value is valid. Delay push-out is used for cases where the output of the gate is expected to switch. This states that the output of the pin and related pin intersection can not change by more than a specified amount when comparing a minimum timing to a relaxed timing. Glitch is used when the output of the intersecting gate is not expected to switch. In this case, the disturbance on the output signal is measured and compared to a specified pass criteria.

Bisection method tends to lead to significantly longer runtimes than path delay. This is because every characterized number will need several iterations to achieve an optimal result. For this reason, most production AMS characterization is done with path delay method.

Path delay numbers will tend to be more conservative than bisection numbers because they require certain signal arrival relationships at the input of the gate rather than a non-failing output of the gate.

Bisection methods are sometimes used in AMS characterization to quantify margin found in production path delay characterization.

Minimum Pulse Width Arcs Characterization

The minimum pulse width (MPW) specifies the minimum time between two consecutive opposite edges of an input signal. MPW is characterized for clock inputs and certain other asynchronous inputs.

The Liberty file will contain values for MPW varying with the input slew of the signal.

Power Characterization

The purpose of the power characterization of any AMS instance is to capture the power effect of each of the pins as well as the standby leakage. This information is then used to compute the total power consumed by the instance.

Static Power (Leakage) Characterization

The static power or leakage is the nominal amount of power consumed by the instance, even if there is no activity. To measure this, the design should be configured idle for a long period of time to eliminate any transient effects. The current should be measured for each power supply.

The static power can be affected by the state of leakage reduction modes in the instance. These modes reduce the static power by placing the circuit into a lower leakage state. These modes would usually be captured in the `.lib` file as `when` conditions on the static leakage.

Dynamic Power Characterization

The dynamic power is the power consumed by any AMS design during active operation. The power is characterized separately for each input and output pin. Depending on which pins are active in a cycle, the power is summed to arrive at the total power consumption.

Dynamic power is measured separately for all power supply pins.

Power for clock pins is characterized by leaving all other pins stable and toggling the clock. This measurement should be done for each valid operation mode separately (usually reported with different input conditions in liberty).

The power for the other input pins should be characterized by toggling that pin separately from other pins and measuring the power consumed.

Power for output pins switching is the difference between a similar cycle with the output switching and a cycle without the output switching. This isolates the power contribution of the output pin and can be used to determine total power.

2

Overview of Liberate AMS

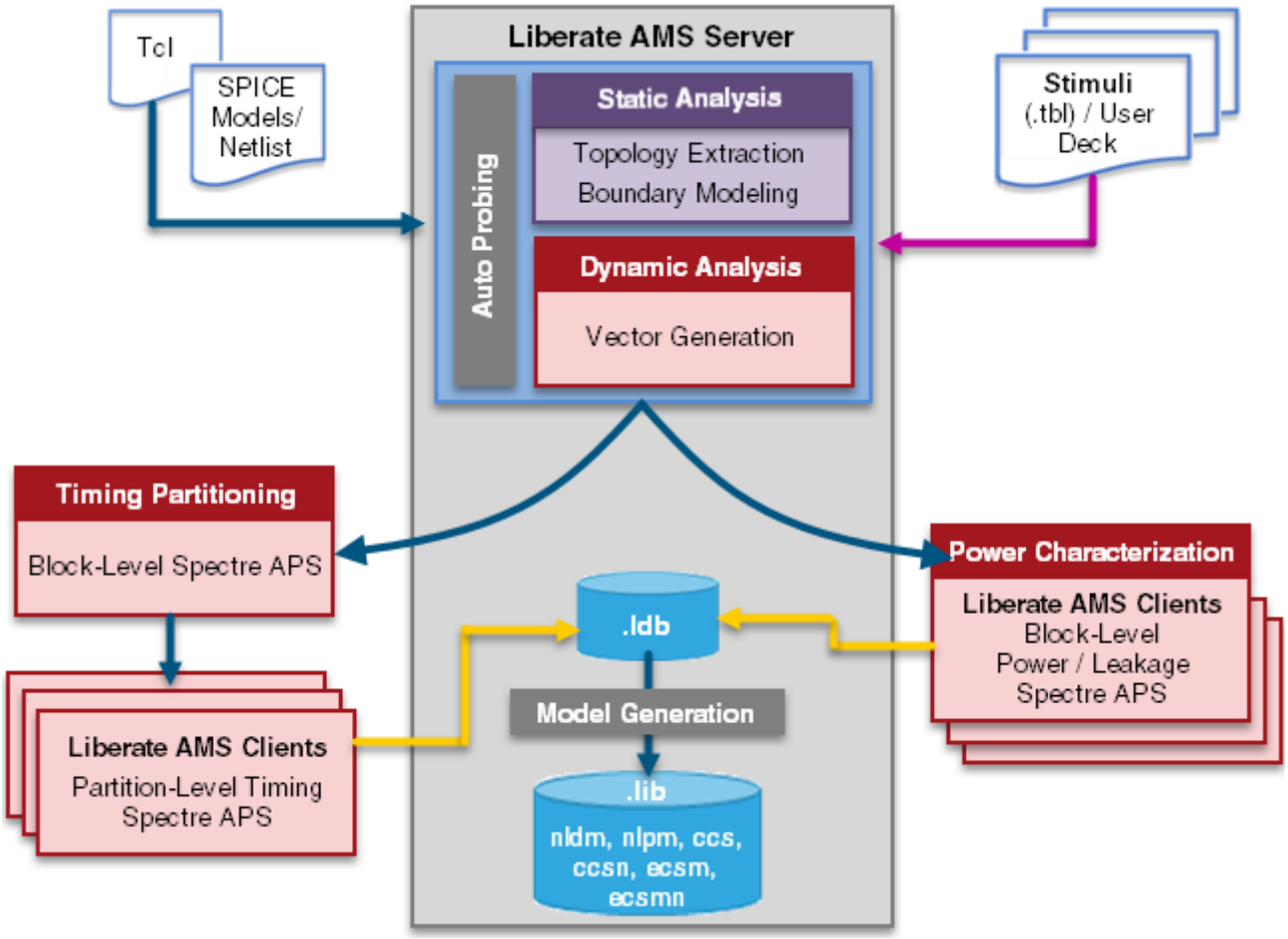
What is Liberate AMS?

Cadence® Virtuoso® Liberate AMS is a tool for characterizing large mixed-signal (AMS) blocks. It creates libraries of electrical views for the digital paths within such blocks.

Successful logic synthesis, design optimization, and verification depend on the existence of accurate libraries of electrical views for the components included in circuits. Liberate AMS provides you with the ability to create such libraries for large AMS blocks containing complex custom digital and analog components. With this capability, you can create characterized libraries for the digital paths within blocks such as phase-locked loops (PLL), serializers and deserializers (SerDes), and analog-to-digital and digital-to-analog (AD/DA) converters.

The completed libraries, which Liberate AMS produces in industry-standard formats such as the Synopsys Liberty (`.lib`) format, can then be used to analyze the static timing and power performance for full chips that include the characterized AMS components.

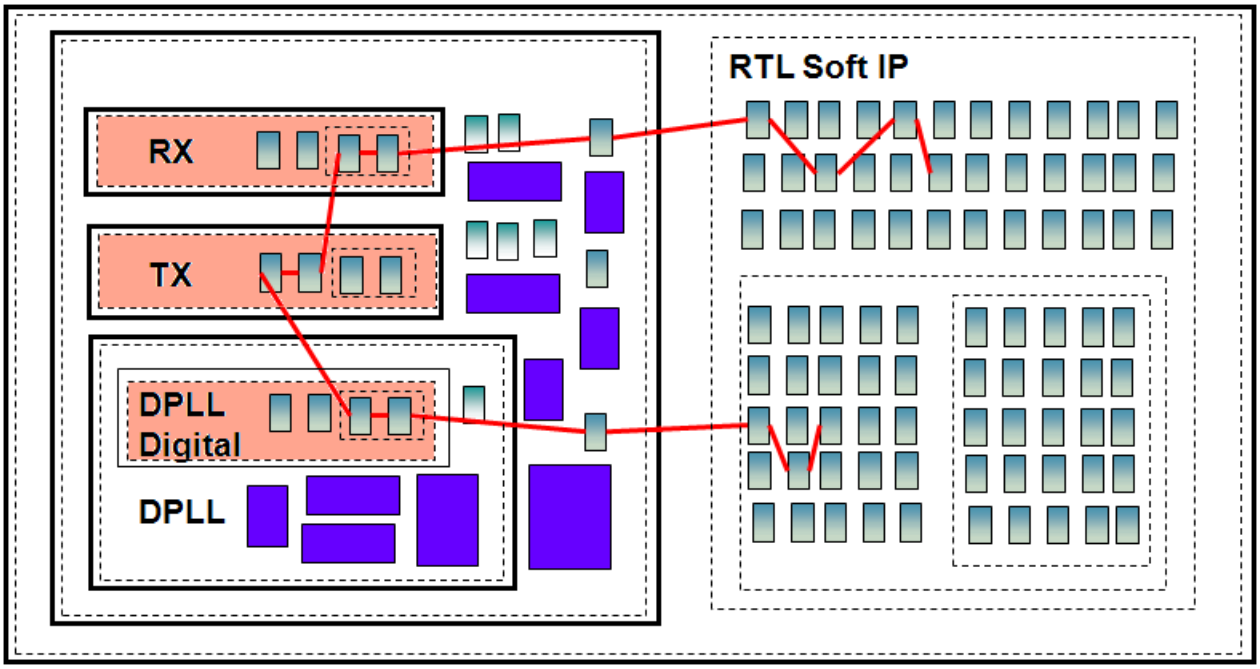
Liberate AMS works in server and client model. The AMS server takes the primary inputs, does the pre-processing (netlist processing, topology extraction, boundary modeling) and creates the database. The AMS clients are used for simulations and for creating partitions.



How Liberate AMS Characterizes AMS Blocks

Full chip system on a chip (SoC) static timing signoff needs to cover paths through the digital portions of AMS blocks, but timing models for AMS blocks and sub-blocks are often inaccurate estimates that cannot be used to model power, timing, and signal integrity. Figure 2-1 illustrates the problem, with the arcs of interest running through AMS components such as the DPLL.

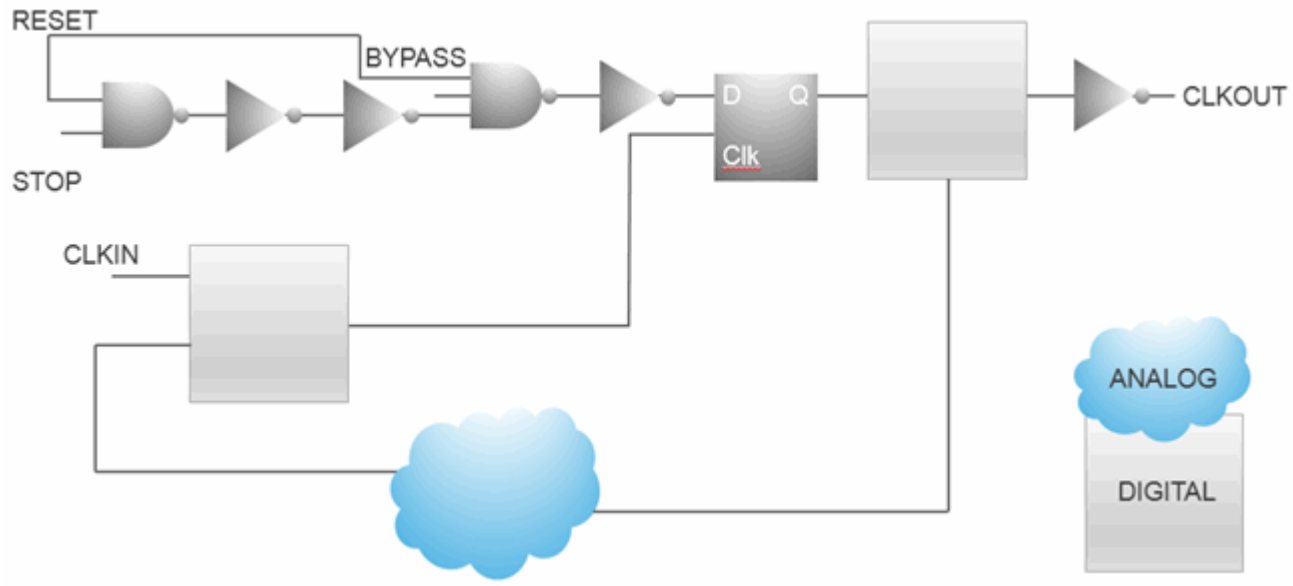
Figure 2-1 Arcs Through an AMS Block



Liberate AMS has the ability to analyze such circuits. By identifying the portions of the circuits that are needed for a high-accuracy simulation, the tool can quickly generate timing models for the digital paths through an AMS block.

The Liberate AMS flow consists of three main steps: partitioning, characterization, and library creation. The following sections use the simplified AMS block, resembling a PLL, in Figure 2-2 to illustrate these steps.

Figure 2-2 Simplified AMS Block to Illustrate Characterization Steps



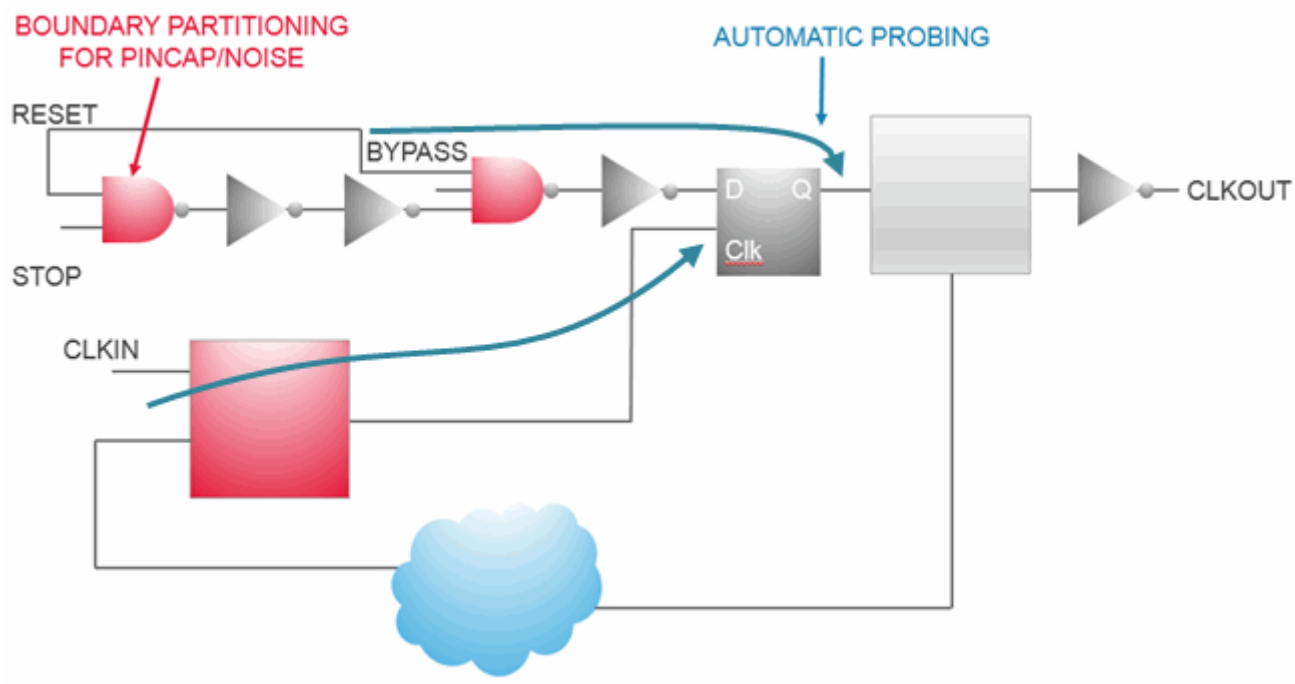
Partitioning

In this step, Liberate AMS automatically identifies the *probes* and portions of the circuit (referred to as *partitions*) that are key to characterization, and concentrates the simulation effort on those probes and partitions. To identify nodes that are relevant to characterization, Liberate AMS first performs a transistor-level static analysis. This step identifies

- Nodes where the setup and hold characterizations between reference and constrained pins should occur.
- Input and output boundary partitions, which are used for input pin capacitance and current-based noise (ECSMN, CCSN) characterization.

Figure 2-3 illustrates the identification of static partitions and probes.

Figure 2-3 Identification of Static Partitions and Probes

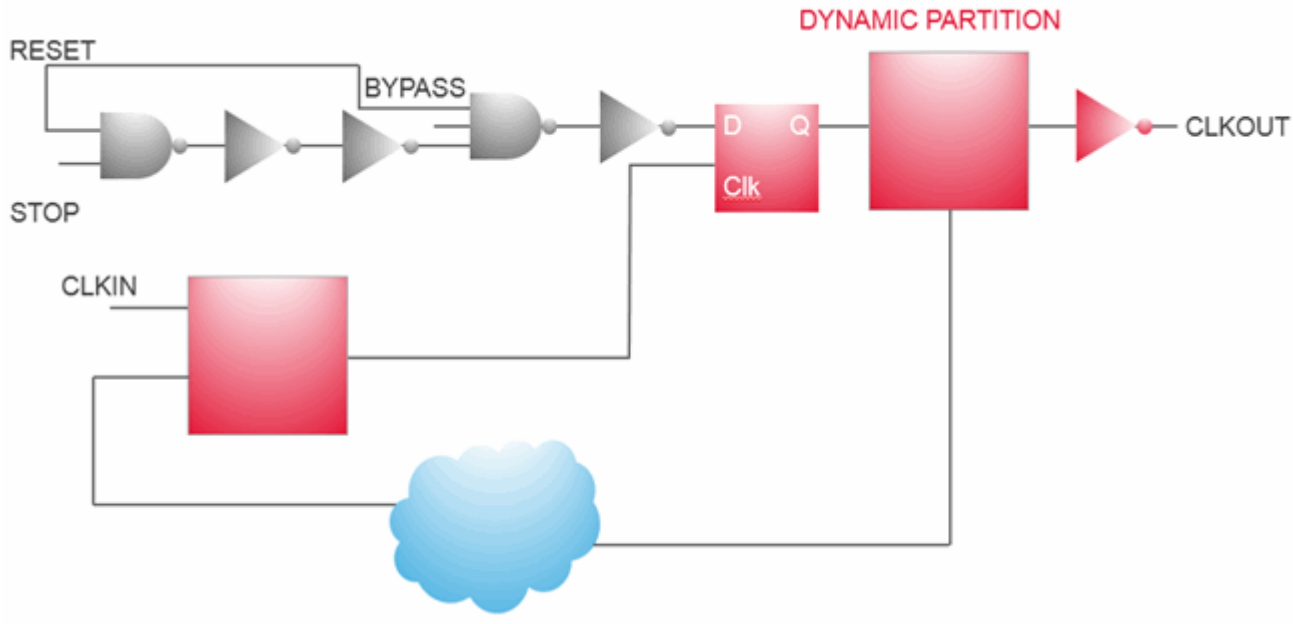


After the transistor-level static analysis, Liberate AMS performs a dynamic simulation using vectors derived directly from your testbenches. This step

- Identifies the vector, or *arc*, for which each measurement is maximized.
- Translates the state and control of the circuits for each specific arc into a snapshot of the original circuit—a dynamic partition. This partition contains all the circuitry and information necessary to carry out the characterization step.

Figure 2-4 illustrates the identification of dynamic partitions.

Figure 2-4 Identification of Dynamic Partitions for an Arc



Characterization

Each static and dynamic partition identified in the previous step is an accurate representation of the original circuit for the specific arcs with which it is associated. Being significantly smaller than the original circuit, the partitions can be simulated quickly, with full SPICE accuracy, and for all the specified slew/load and slew/slew combinations that might be necessary. In general, Liberate AMS can apply to these partitions all the characterization and modeling techniques that are available for standard Liberate cells.

This characterization step can be parallelized across multiple cores and distributed across multiple machines.

Library Creation

After all characterization runs are complete, Liberate AMS merges the results and stores them as a final library. During this step, the tool can also generate different current-based and noise models.

3

Getting Started with Liberate AMS

This chapter describes briefly how to start using Cadence® Virtuoso® Liberate™ AMS. A systematic approach to tool setup is covered here with the intent to help new users of the tool. Once you are familiar with the tool, these can be refined.

Tool Installation and Setup

Installing Liberate AMS

To install Liberate AMS:

1. Familiarize yourself with the installation tools, InstallScape, and the license manager. To find guidance materials for these tools, see https://support.cadence.com.
2. To obtain the Liberate AMS software, see https://downloads.cadence.com.
3. Select the *LINUX* tab.
4. Select the appropriate release (for example, `LIBERATE161`).

   The first two numbers in the release name designate the year of the release and remaining numbers begin with one and increment with each additional release during that year. Therefore, `LIBERATE161` is the first LIBERATE base release of 2016.
5. Download and install the product.

   This step utilizes the tools, InstallScape and the license manager, that you learned about in step 1.
6. Use commands such as the following to include Liberate AMS in your software path.

   ```
   % setenv ALTOSHOME <install_dir>/<liberate_release_name>
   % set path=($path $ALTOSHOME/bin)
   ```
7. Set the following to include integrated Spectre in your executable path:

   ```
   % set path ($path $ALTOSHOME/tools.lnx86/spectre/bin)
   ```

Managing Licenses

This section introduces the concept of licensing. For more detailed information about setting up and configuring the license server, see step 1.

Liberate AMS uses a server-client licensing scheme. The tool uses a server license for partitioning, preprocessing, and for starting and monitoring the characterization run on the server machine. It uses client licenses for running characterization on the client machines and for any post-processing of the library database. Each Liberate AMS server can access all the available client licenses. For example, with two server licenses and forty client licenses, the following configurations are all valid.

- A single characterization run using 40 client processes
- Two simultaneous characterization runs, each with 20 client processes
- Two simultaneous characterization runs, one with 30 client processes and one with 10 client processes

Note: It is important not to request more client licenses than are available because doing so can increase the wall-clock time required to complete the characterization.

Important

Ensure that the license daemon (`cdslmd`) and the license server (`lmgrd`) have the same version and that this version is the same as that required for a release. For example, v11.11.1 is required for the Liberate 15.1 release. If a mismatch is detected, unexpected license behavior might be observed. For example, the license search path can be reset to `<none>` after a failed license check out request. This can result in incorrect license checking in process.

On a 64-bit license host, the 64-bit `cdslmd` and `lmgrd` must be used instead of the default 32-bit ones.

Waiting for a License

When you submit an AMS job, Liberate AMS requests a license. If you want the tool to wait until a license becomes available, set the following environment variable:

```
setenv ALTOS_QUEUE 1
```

How Liberate AMS Uses Licenses

When a Liberate AMS server starts, it checks out one `Liberate_AMS_Server` license. Later, when simulations are ready to begin, Liberate AMS tries to check out N

`Liberate_AMS_Client` licenses, where N is the number of threads specified in the char_ams command. This command has different options available to control the number of threads, for example, `-partthread`, `-charthread`, `-part_fastsim_thread` and so on.

When `ALTOS_QUEUE` is set to `1`,

- If Liberate AMS acquires fewer than N licenses, it waits for up to the value of the `lic_max_timeout` variable, trying to get all N licenses. After `lic_max_timeout` is reached, if Liberate AMS acquires M licenses and M > 0, it starts M threads.
- If M is 0, it again waits for another `lic_max_timeout` seconds to acquire client licenses. This process is repeated until at least one client license is acquired.

When `ALTOS_QUEUE` is set to `0`,

- If Liberate AMS acquires all N licenses, it starts simulations using N threads.
- If Liberate AMS acquires M licenses, 0 < M < N, it starts M threads.
- If Liberate AMS does not acquire a license, it quits.

Environment Variables for Controlling Licensing Checks

- ALTOS_LIC_MAX_TIMEOUT
  `setenv ALTOS_LIC_MAX_TIMEOUT <value>`

  where;

  `value` is duration in seconds.

  This shell variable specifies the duration, in seconds, for which Liberate AMS (both server and client) should wait for licenses.

  For a server process, if the `ALTOS_QUEUE` variable is enabled, Liberate will attempt to check out 1 Server license. If the max timeout is reached, and no server license has been checked out, then Liberate will reset the timer and loop back to continue waiting for a license. For a client, when the max timeout is reached and at least one license was checked out, then the Liberate client will start to run with the licenses it has. No additional licenses are checked out.

- ALTOS_LIC_CHECK_ALT_TIMEOUT
  `setenv ALTOS_LIC_CHECK_ALT_TIMEOUT <value>`

  where;

  `value` is duration in seconds.

Some Cadence characterization products can run using more than one product license. This variable controls both the server and client timeout before trying to check out an alternative license feature if there are any such licenses in the license pool.

64-bit Machine Support

Liberate AMS also ships with support for a 64 bit machines. To use the 64 bit port, set the `ALTOS_64` environment variable prior to starting Liberate AMS:

```
% setenv ALTOS_64 1
```

Inputs Required for Liberate AMS Characterization

Various types of data are required to run Liberate AMS:

- Extracted AMS block netlists in SPICE format
- Foundry device models in SPICE format
- Liberate AMS command files in Tcl format
- A testbench

  See define_deck for more information about converting a testbench into an input stimulus file.

The files required to get started with Liberate AMS characterization can be grouped into the following two categories:

1. **Simulation setup files**: These files are typically available by the time an instance needs to be characterized. This set of files consists of:

   a. **Netlist for the instance (Specifying extracted AMS block netlists)**

   The transistors, diodes, resistors, capacitors, and extracted parasitic elements (RCs) that compose the AMS block netlists are passed to Liberate AMS in SPICE format. Spectre, Berkeley SPICE, and HSpice® netlist formats are supported. The information passed as files to Liberate AMS must include a `.subckt` definition for the block to be characterized. To specify the extracted netlists, use the `read_spice` Tcl command, as shown below:

   ```
   read_spice {pll.sp}
   ```

   b. **Model files (Specifying and using foundry device models)**

   Device models, which represent the electrical parameters of the devices, are supplied by foundries. The device models include P- and N-channel transistors,

diodes, capacitors, and resistors. Most device model files include different parameters for different process corners, such as, a typical corner, a fast corner, and a slow corner. To prepare for a Liberate AMS run, you must include the device model files and specify the operating conditions.

c. Vector or testbench

Vector is needed to set up the circuit in wanted state and transition the output and/or internal nodes to exercise desired arc. The importance of vector is to help the tool to understand what input transition causes what output or internal node transitions.

- ***Truth table format***: User can write the stimulus in truth table format. For more information, see Appendix D, "Truth Table Format."
- ***Using existing vector***: Testbenches for simulating the AMS blocks are available with the designer by the time the instance needs to be characterized. These testbenches can be used directly (see define_deck). For the arcs that need to be characterized, it is important to check if the testbenches exercise these arcs by providing appropriate vectors.

See also Setting Up a Stimulus (tbl) File: An Example.

2. **Files needed to setup characterization**: This group of files needs to be created. It contains:

a. Template file

A template file is used to define or specify the arcs needed from a characterization run. There are three primary sections in this file that you need to create.

- ***Template definitions***: input/clock slew, output load (`define_template`)
- ***Cell definition***: details of input/output pins in design
- ***Arc definitions***: timing, power, and leakage arcs

A template file can be created in the following two ways:

i) Manually create a `template.tcl` file where content can be written in the three sections listed above.

ii) If an existing library (`.lib`) is available for the block (maybe from an older process node), it can be read in and used to create the template file.

Use the `write_template` Tcl command to read in the reference library that you provided and generate a template or macro (`template.tcl`), which is used as an input file for the next script.

```
read_library [pwd]/ref.lib
```

```
write_template -verbose [pwd]/tmpl/template.tcl
```

See also Setting Up a Template File: An Example.

b. Primary setup Tcl file

The commands that run in Liberate AMS are Tcl commands, typically embedded in Tcl scripts. It is convenient to create a shell file to run the various Tcl scripts in the proper sequence. The Tcl commands available for Liberate AMS are detailed in Chapter 6, “Liberate AMS Commands.”

Tcl scripts are used to specify the AMS netlist, SPICE models, and operating conditions. Tcl scripts also define the range of data, such as, input slew and output-loading conditions, for the characterization. Liberate AMS simulates and measures the AMS netlist using each of the specified input slews and loads and generates the appropriate delay tables, timing checks (setup, hold, and so on) and power information (switching power, hidden power, state-dependent leakage). Liberate AMS can also generate library information for the composite current source (CCS) model and the effective current source model (ECSM), for both timing and noise.

See also Setting Up a Top-Level Tcl Run File: An Example.

Setting Up a Stimulus (tbl) File: An Example

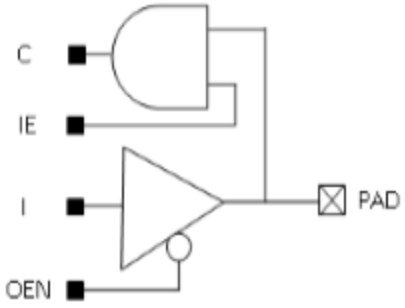
The table files contain the vectors to be used for characterization. These are specified in the `amschar.tcl` file using the define_table command. The vectors in these files are used for the various measurement types, delay, constraint, power, and leakage. Along with the vectors, there are specifiers in the output column to denote which cycles are appropriate for measuring each arc type.

```
arctypes delay
table functional
pins            clkin   ce      din     tm      dout
valid_11        R       L       1       1       X
valid_00        R       L       0       0       X
valid_00        R       L       ?       0       X
test_11         R       L       ?       1       D
test_00         R       L       ?       0       D
endtable
```

Setting Up a Template File: An Example

The template file, `template.tcl`, contains the characterization information about the slews and loads to be characterized, the pinout of the instance specified using the define_cell

command, and the define_arc statements for all the arcs that need to be in the final library file. The example `template.tcl` in this section shows how you can manually define arcs for a simple Input/Output (I/O) test case:



Input				Output	
IE	OEN	I	PAD	PAD	C
0/1	0	0	-	0	0
1	0	1	-	1	1
0	0	1	-	1	0
0/1	1	X	0	-	0
0	1	X	1	-	0
1	1	X	1	-	1
0	1	X	Z	-	0
1	1	X	Z	-	X

template.tcl

```
define_cell \
    -input {OEN I} \
    -output {C} \
    -bidi {PAD} \
    -pinlist {IE OEN I PAD C} \
    -delay delay_template_7x7 \
    IOX1

####IO INPUT
#########################
#OEN=> PAD
define_arc -type enable -vector "XF0FX" -related_pin OEN -pin PAD IOX1
```

```
define_arc -type enable -vector "XF1FX" -related_pin OEN -pin PAD IOX1
define_arc -type disable -vector "XRXRX" -related_pin OEN -pin PAD IOX1
define_arc -type disable -vector "XRXFX" -related_pin OEN -pin PAD IOX1

#I=> PAD
define_arc -vector "X0FFX" -related_pin I -pin PAD IOX1
define_arc -vector "X0RRX" -related_pin I -pin PAD IOX1

####IO OUTPUT
#########################
#IE =>C
define_arc -vector "R00XF" -related_pin IE -pin C IOX1
define_arc -vector "R01XR" -related_pin IE -pin C IOX1
define_arc -vector "R1X0F" -related_pin IE -pin C IOX1
define_arc -vector "R1X1R" -related_pin IE -pin C IOX1
define_arc -vector "FXXXF" -related_pin IE -pin C IOX1

#OEN =>C
define_arc -vector "1F0XF" -related_pin OEN -pin C IOX1
define_arc -vector "1F1XR" -related_pin OEN -pin C IOX1
define_arc -vector "1RX0F" -related_pin OEN -pin C IOX1
define_arc -vector "1RX1R" -related_pin OEN -pin C IOX1

#PAD =>C
define_arc -vector "11XFF" -related_pin PAD -pin C IOX1
define_arc -vector "11XRR" -related_pin PAD -pin C IOX1
```

Setting Up a Top-Level Tcl Run File: An Example

The top-level Tcl run file is the primary file that is provided to Liberate AMS. Most of the settings in this file remain the same from test case to test case and therefore, you can use this file as a template while setting up your own blocks for characterization.

Note: You can setup Liberate AMS for any design by following the guidelines given below. Once you get familiar with tool, you can start using many more variables. The inputs highlighted in **`bold`** typeface are the minimum requirements to run the tool for characterization. However, if not specified, the tool picks the default values for these variables.

1. Set the corners, that is, Process, Voltage, and Temperature (PVT) information, as shown below:

   ```
   set process ff
   ```

```
set voltage 0.95
set temperature 0
set_operating_condition -temp $temperature -voltage $voltage
```

2. Setup the root directory, as shown below:

```
# SETUP DIRECTORY STRUCTURE
    set cell dig_dfnrq4
    set datadir [pwd]
```

3. Setup the location of directories containing the input files, as shown below:

```
# INPUT FILES
    set tcldir   ${datadir}/tcl
    set tmpldir  ${datadir}/tmpl
    set tbldir   ${datadir}/table
    set spicedir ${datadir}/netlist
    set deckdir  ${datadir}/user_deck
```

4. Setup the output directories, as shown below:

```
# OUTPUT FILES
    set outdir ${datadir}/out
    set_var px_dir ${outdir}/ams    ;#specify partition sim dir    set ldbdir
    ${outdir}/ldb; exec mkdir -p $ldbdir
    set libdir ${outdir}/lib; exec mkdir -p $libdir
```

5. Specify the paths to read in the netlist and model files. Also, define leafcells and have Liberate AMS parse these files.

```
# READ IN SPICE
    set netlistfile [pwd]/input.scs
    set modelfile [pwd]/include_${process}
    set_var extsim_model_include $modelfile
    define_leafcell -type nmos -pin_position {0 1 2 3}   {g45n1svt}
    define_leafcell -type nmos -pin_position {0 1 2 3}   {g45p1svt}
    define_leafcell -type black_box -pin_position {0 1} {g45rnsnp}
    read_spice -format spectre [list $netlistfile]
    read_spice -format spectre [list $modelfile]
```

In Liberate AMS, there is no need to parse a model. Therefore, you can define model leafcell definitions using the `define_leafcell` command.

The `define_leafcell` command is used to control the flattening of the hierarchy when the netlist is parsed by Liberate AMS. It stops flattening the hierarchy once it reaches the leafcells specified using this command. Following are guidelines for using this command:

a. Specify all active device (pmos, nmos, and so on) models in the design.

b. Set type as `black_box` for resistors.

c. Capacitors, diodes, and so on can also be defined using the appropriate `-type` option with this command.

d. Do not use the `black_box` option to define design elements (subcircuits) in the netlist as blackbox items.

Note: Cadence recommends not to parse models (that is, `read_spice $modelfile`) and use all leafcell definitions. This reduces the internal database size of Liberate AMS and therefore, speeds up the overall process.

6. Specify the rails. The `set_vdd` and `set_gnd` commands are used to identify power and ground nets and their values. These can be used multiple times to specify all the different supplies in the design.

   The primary purpose of the `set_operating_condition` command is to specify the temperature.

   Liberate AMS takes care of the virtual nets by finding and defining them automatically. See the px_find_virtual_rails variable for more details on virtual rail.

```
# SPECIFY RAILS
    set_vdd VDD $voltage
    set_gnd VSS 0
    set_var px_find_virtual_rail 2
```

7. Specify the simulation settings. You have a fine level of control over supplying the choice of simulator, simulator version, simulator options, and so on.

```
# SPECIFY SPICE ENGINES and OPTIONS
    set partsim "aps"
    set charsim "aps"

    if {$partsim == "aps"} {
        set_var fastsim_cmd "spectre"
        set_var fastsim_cmd_option "+aps -64 +errpreset=liberal"
        set partsim "spectre"
    }
    if {$charsim == "aps"} {
        set_var extsim_cmd "spectre"
        set_var extsim_cmd_option "+aps -64"
        set charsim "spectre"
    }
```

8. Source the input template file, as shown below:

```
source ${tmpldir}/template.tcl
```

   For a sample input template file, see Setting Up a Template File: An Example.

9. Specify the testbench. In Liberate AMS, you have the following two ways to input testbench:

   a. Use the define_deck command.

```
define_deck -observation_window 100e+06 -dut_inst I0 ${deckdir}/input.scs
$cell
```

   b. Use the truth table specified with the `define_table` command.

```
define_table [list tcl/setuphold.tbl] $cell
```

10. You can also specify the following miscellaneous commands:

```
;#Charcterize only user-defined arcs
    set_var px_create_if_uda 1
# reuse the top-level simulation output if available
    set_var px_fastsim_reuse 1
```

11. Initiate characterization by running the char_ams command.

```
# MAIN COMMAND
    char_ams -partsim $partsim \
             -charsim $charsim
```

    The only options set using this command are the simulators for full instance run and characterization run. The tool is also capable of running all the top-level full instance simulations and the individual arc characterization as a distributed job using a workload management, load-sharing platform like LSF. This can significantly reduce the overall run time for large test cases and when there are many arcs to characterize. Refer px_distributed_sim and `rsh_cmd` for more details.

    The `char_ams` command has many user-level controls available to control the characterization.

12. Write the library database and models by using the `write_ldb` and `write_library` commands, respectively.

```
# WRITE MODELS
    write_ldb ldb/${cell}.ldb
    write_library -rename -filename lib/${cell}.lib ${cell}
```

    The `write_ldb` command creates Liberate AMS library database (LDB). This database can be then read back in a later session for formatting library data. It is strongly recommended that this command is specified after the `char_ams` command to ensure availability of a clean unmodified copy of the LDB for future use.

The `write_library` command outputs the library in Liberty format to the specified file.

Note: You need to add ***only*** the `-ccs`, `-ecsm`, `-ccsn`, and `-ecsmn` options to the `char_ams` and `write_library` commands to get CCS, ECSM, CCSN, and ECSMN liberty models. There is no external stimulus input needed for these models. Refer to Chapter 4, "Liberate AMS Flows" for more details about CCS/ECSM and noise characterization.

Running Liberate AMS

Before using Liberate AMS, make sure that it is installed correctly and that all the necessary prerequisite data are available.

To use the 64-bit port of Liberate AMS, set the `ALTOS_64` environment variable prior to running the tool, as shown below:

```
% setenv ALTOS_64 1
```

Start the characterization by typing `liberate_ams` followed by the Tcl command file, as shown below:

```
% liberate_ams ams.tcl
```

By default, Liberate AMS utilizes `stdout` and `stderr` for messages and does not create a log file. However, to run Liberate AMS so that it uses a log file, you can use a command such as following:

```
% liberate_ams ams.tcl |& tee ams.log
```

4

Liberate AMS Flows

This chapter discusses the various flows supported by Liberate AMS in the following sections:

Using Deck-Driven Characterization

Liberate AMS uses a dynamic simulation-driven approach for characterization. As it is built on the Liberate MX technology, Liberate AMS needs to be provided with tables of vectors. However, in a mixed-signal block the analog part plays an important role. As a result, the initialization can be done correctly only when you use an appropriate analog simulation testbench.

For an analog or mixed-signal designer who has the know-how of the functionality of an instance, capturing the design intent becomes crucial during mixed-signal instance characterization. Therefore, the generation of the tables of vectors is automated by utilizing the existing testbenches from the analog transistor-level simulation of the block.

The stimuli is captured by running a full block simulation using a FastSPICE engine, like Spectre® XPS in MS mode, to identify circuit activity. The objective is to provide the Liberate AMS tool with vectors to:

- Configure the mixed-signal block into a specific state for the initialization of the analog part.
- Generate the stimuli to exercise the desired arcs for outputs and internal nodes.

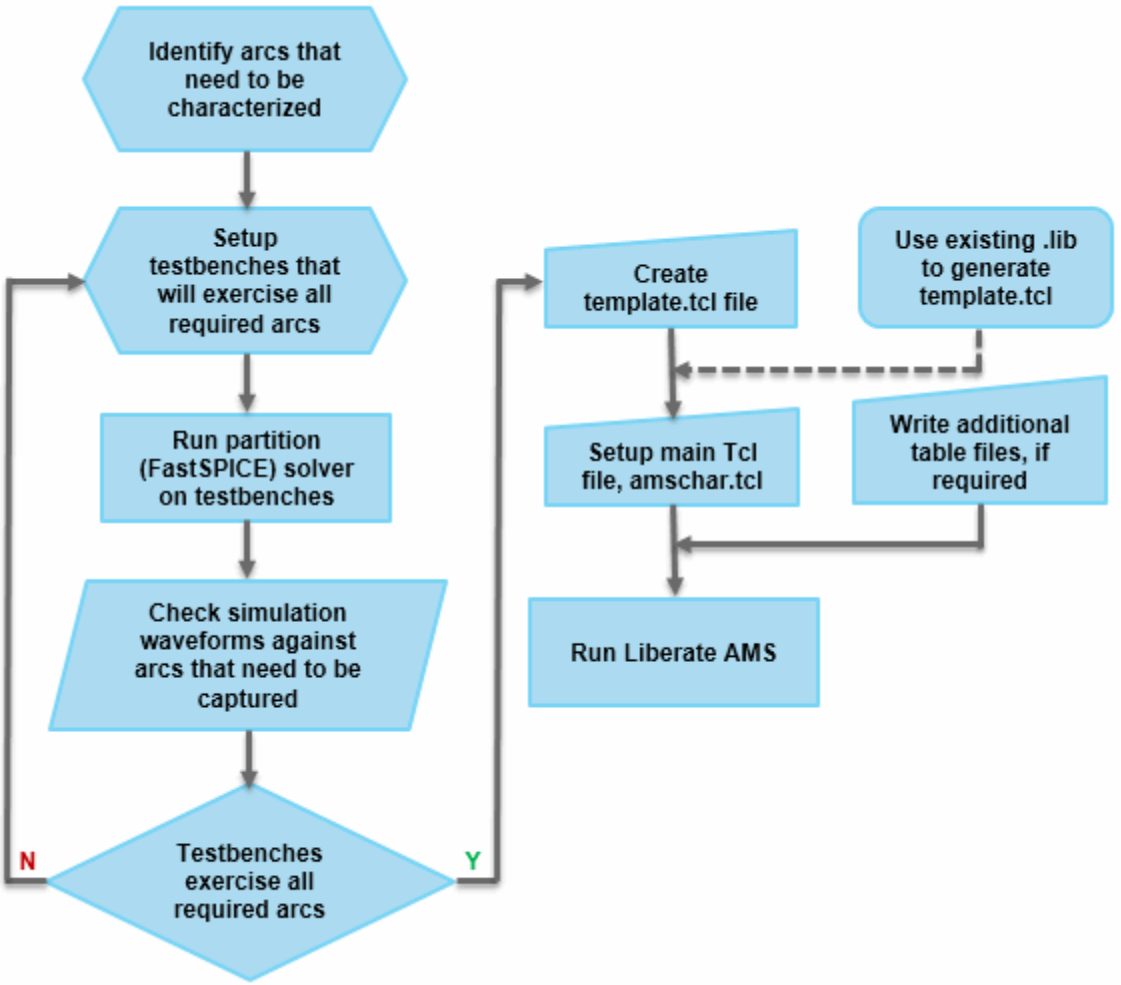
For this, Liberate AMS directly re-uses the stimuli from the deck.

The testbenches are used to capture the stimulus information at the input pins of the block. Liberate AMS then creates the required internal table format automatically for full instance simulation.

Then the use model requires you to set up the testbenches needed to exercise the timing arcs. Liberate AMS partition solver runs on those testbenches to generate the transition database for fastsim and create the partitions.

Verify whether the fastsim results correspond to the original results. Then, double check that the simulation waveforms exercise the needed arcs. If this is the case, you can move forward to update the `template.tcl` file and finalize the main characterization script. Otherwise, reiterate the process of preparing the testbenches and verifying the waveforms until the flow is clean.

This process is defined in the flow diagram below.



Use the define_deck command to specify the testbenches. The following example illustrated the use of the `define_deck` command for leakage, timing, and power characterization:

```
# SPECIFY TESTBENCHES/TABLES

    puts "Loading testbenches ... ${tmpldir}/${cell}.testbench"
    set_var leakage_sim_duration 2e-06 ; # measure leakage after 2us.
    define_deck  -observation_window 1.25e-07 -dut_inst I0 -arctypes "leakage" ${testbenchdir}/${cell}_leakage_en.scs $cell
    define_deck  -observation_window 1.25e-07 -dut_inst I0 -arctypes "timing" ${testbenchdir}/${cell}_timing.scs $cell
    define_deck  -observation_window 1.25e-07 -dut_inst I0 -arctypes "power" ${testbenchdir}/${cell}_power.scs $cell
    puts "Loading testbenches  done..."
```

The `template.tcl` file is the script where you define the templates and arcs that need to be characterized. This template can be generated automatically if there is an existing `.lib` file. Liberate AMS, as the other Liberate tools, has a useful re-characterization flow, where you run the `read_library` and `write_template` commands, as shown below:

```
read_library ${RUNDIR}/Liberty/${LIB}_${pvt}.lib
write_template \
    -verbose \
    ${TCL_DIR}/template.tcl
```

In any case, the `template.tcl` file needs the definition of the arcs that need to be characterized. At the same time, the testbenches provided shall be able to exercise those arcs. The example below shows the definition of the `define_arc` command:

```
#-----------------------------
#     Define Timing Arcs
#-----------------------------

# constraints nm - 2 arcs
foreach dir {R F} { \
    foreach type {setup hold} { \
        define_arc \
            -type $type \
            -pin {din<7:0>} \
            -pin_dir $dir \
            -related_pin_dir R \
            -related_pin I_ckgen.ckd8 \
            -when !margin \
            serializer
    }
}
```

It is also possible to let Liberate AMS identify the arcs on the fly by setting the following variable:

```
set_var px_create_if_uda 0; # Def. 0. Set to 1 for user-defined-arcs only.
```

Liberate AMS can identify arcs using dynamic simulation information. The px_create_if_uda variable controls which of the identified arcs should be characterized.

- If the variable is set to `1`, only the arcs defined in `template.tcl` are characterized.

- If the variable is set to `0`, a partition is created for any identified potential arcs and they are subsequently characterized (useful to identify arcs that the arc specification missed).

The `define_deck` command that tells Liberate AMS how to simulate the different timing arcs. It specifies use this deck that was directly created from the Virtuoso database and use this observation window to capture the characterization measurements, as shown below:

```
# SPECIFY TESTBENCHES/TABLES

    puts "Loading testbenches ... ${tmpldir}/${cell}.testbench"
    set_var leakage_sim_duration 2e-06 ; # measure leakage after 2us.
    define_deck  -observation_window 1.25e-07 -dut_inst I0 -arctypes "leakage" ${testbenchdir}/${cell}_leakage_en.scs $cell
    define_deck  -observation_window 1.25e-07 -dut_inst I0 -arctypes "timing" ${testbenchdir}/${cell}_timing.scs $cell
    define_deck  -observation_window 1.25e-07 -dut_inst I0 -arctypes "power" ${testbenchdir}/${cell}_power.scs $cell
```

The setup is also simplified because it uses the existing testbenches to drive the characterization.

Considerations for Power Characterization

The simulation-based approach also enables power characterization that does not partition the design because it is requires simulation to be run on the full instance.

You need a specific testbench to exercise the power arcs (internal power and switching output power). This testbench can be similar to one used for delay.

The leakage power can also be characterized using a deck. For a leakage type of deck, the clock is exercised for a few cycles and then it is allowed to assume the still state. All the input signals also remain in a stable state. You can define the leakage transition time by using the following variable:

```
set_var leakage_sim_duration 2e-06; # measure leakage after 2us.
```

Considerations for Constraint Measurements

To measure constraints, such as setup, hold, recovery and removal times, you can follow the guidelines described in this section.

The main consideration is that the deck-provided clock and data should be lined up properly for constraint characterization whether or not you use the `-shift_clocks` option with the define_deck command.

The transition event should occur between the defined observation window. Liberate AMS uses this window to infer causality between an output pin and its related pin or between a data pin and a constrained pin for an arc.

The recommended value of an observation window is a quarter of clock period. You can define the observation window directly by using the `define_deck` command, as shown below:

```
define_deck -observation_window {2.5e-9} -dut_inst dut -arctypes "timing" -
shift_clocks ${deckdir}/input.scs $cell
```

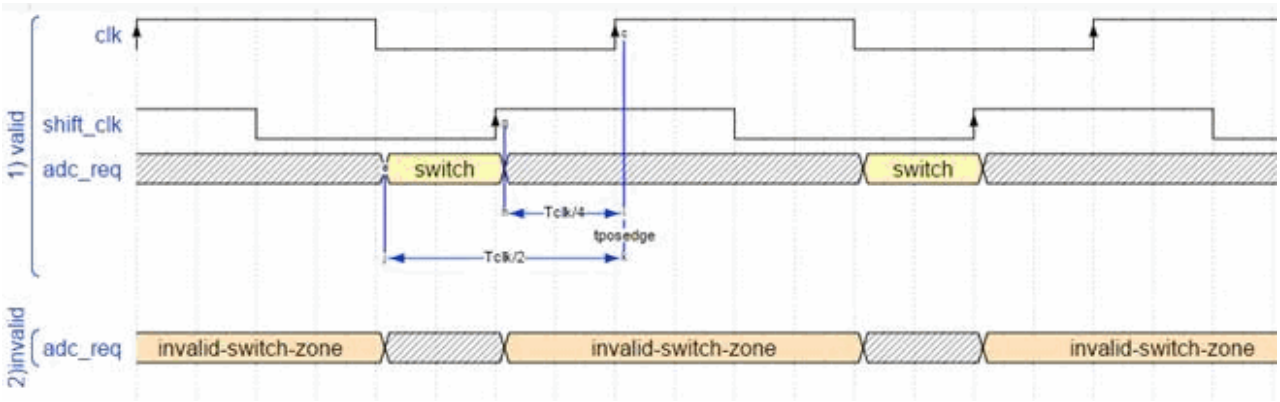
The proper setup for this (or any other case using `define_deck` and one or more main clocks) is to have:

- no delay on clock (or a delay which is a multiple of clock period).
- constrained pins lined up with clock.

This usually involves a few adjustments to the deck, if not already done in that configuration. To do so, use the `define_deck` command with the `-shift_clocks` option and the `-observation_window` option set to `0.25*clock_period`. The second option just shifts the clocks by half of the simulation interval, which is called the observation window. It gives Liberate AMS the extra flexibility to align constraints and clock edges.

It is necessary to update the deck for the constrained pin to change in a quarter of the period (observation window) according to the clock active edge. For an input clocked by the positive edge of the clock, the testbench should be setup so that the stimulus signal connected to this input should be switched within the following time interval:

- From `(tposedge - Tclk/2)` to `(tposedge - Tclk/4)`
- All other zones are invalid intervals for constraint characterization, as shown in the following timing diagram for constraint measurement:



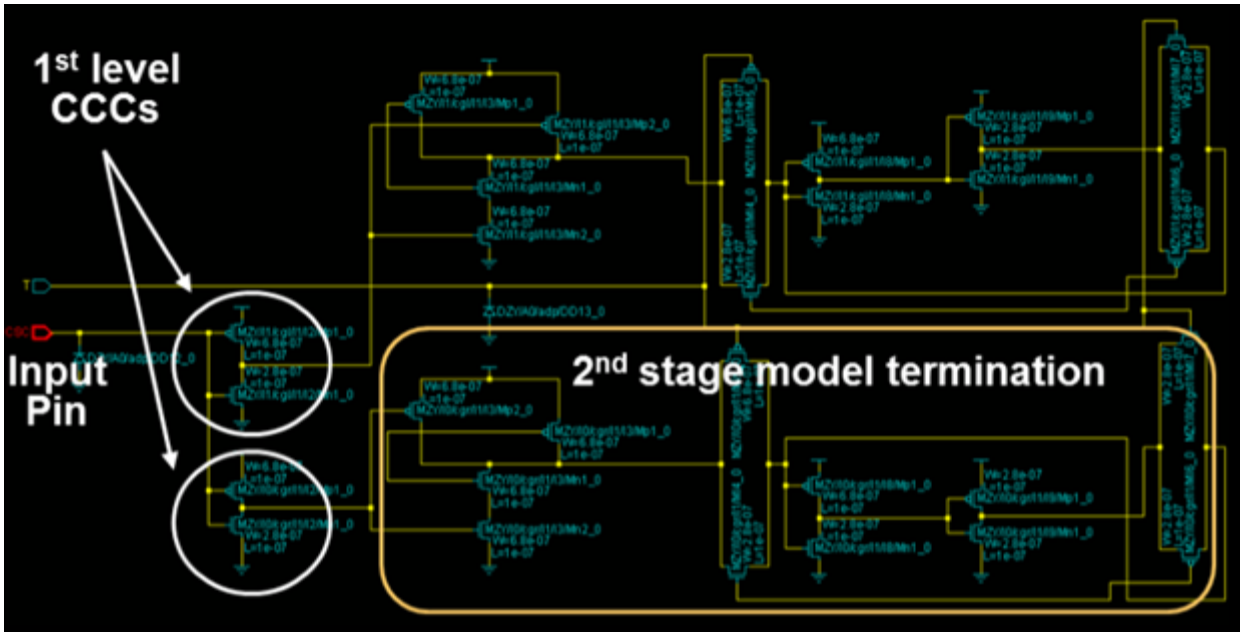
For debugging purposes, if a given testbench does not provide the needed arcs, it can be because the switching is happening in an invalid switch zone.

Considerations for Pin Capacitance and Noise (SI) Models

For pin capacitance and noise (SI) models, do the following:

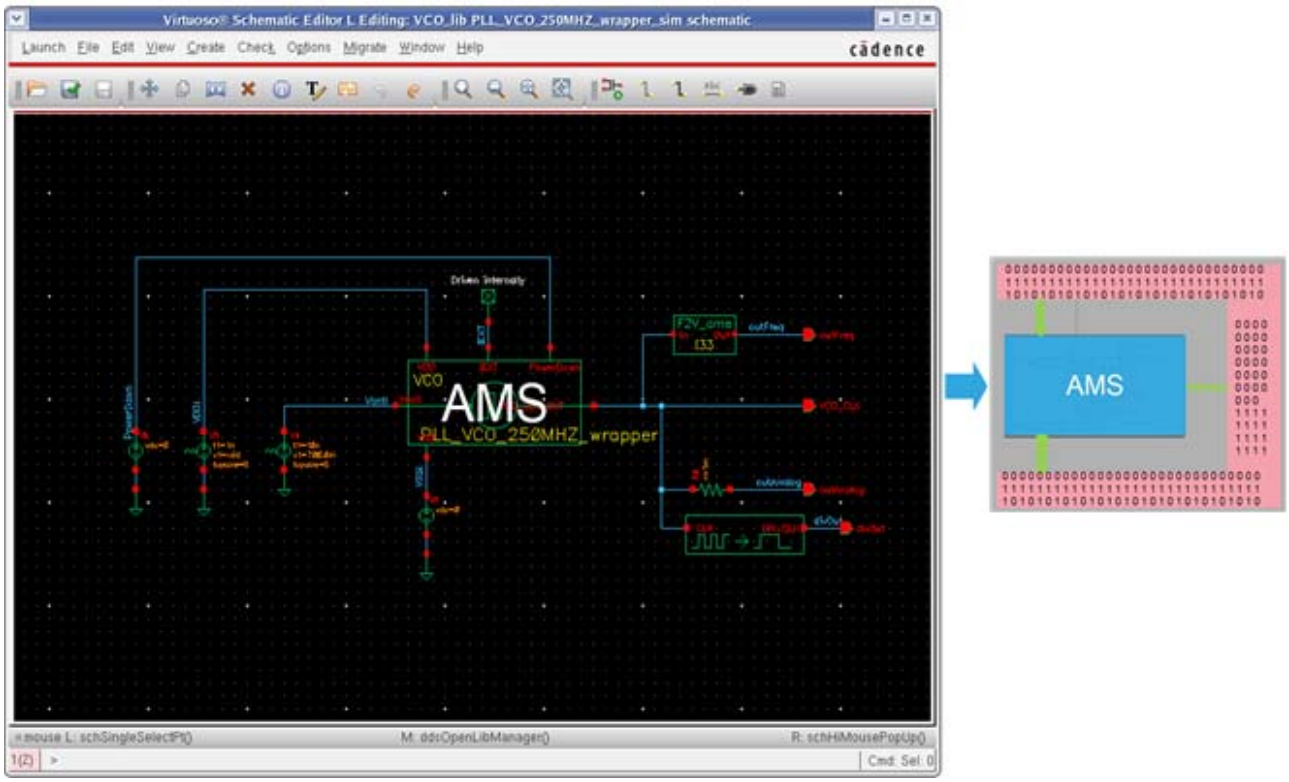
- Create a static boundary partition consisting of the first level of channel-connected transistors attached to each I/O port, which are typically <50 xtrs.
- Characterize each boundary partition separately for pin capacitance and noise (CCSN/ECSMN)
- Integrate each boundary partition noise and pin capacitance data back into the final model.

The figure below illustrates these considerations:

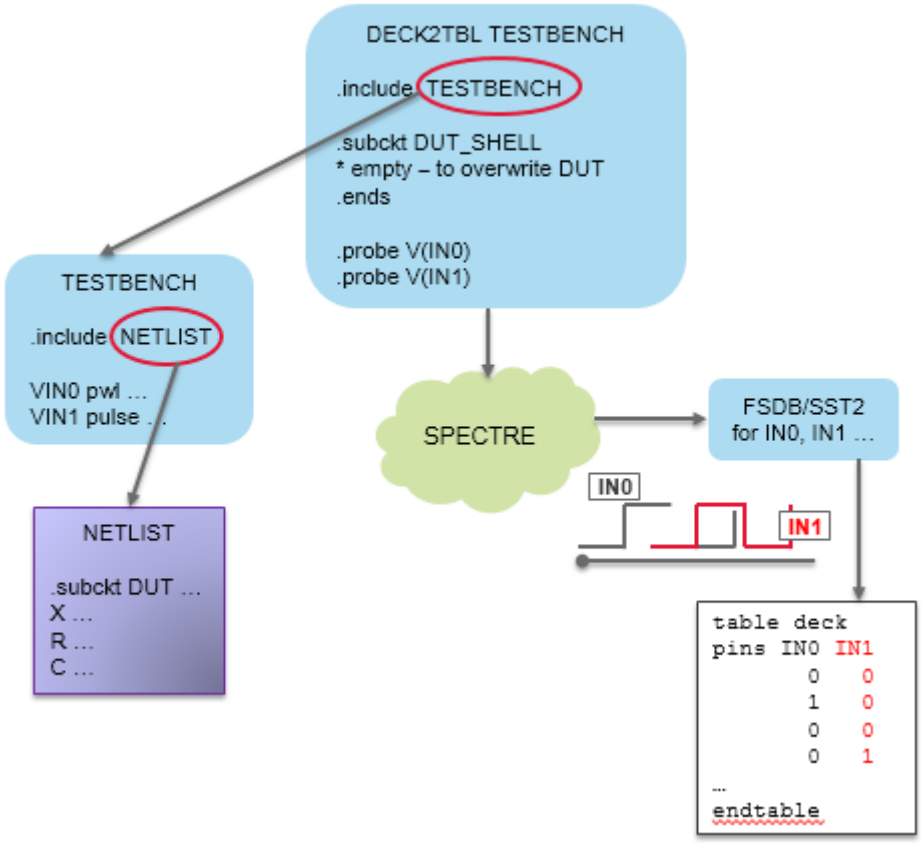


Vector Creation – Testbench Reuse

Use existing testbenches to create vector tables to drive characterization. Each testbench is simulated with the Liberate AMS block hollowed out to capture input switching that is in turn converted into Liberate AMS truth table.



The figure below capture a designer's intent while using the `define_deck` command:



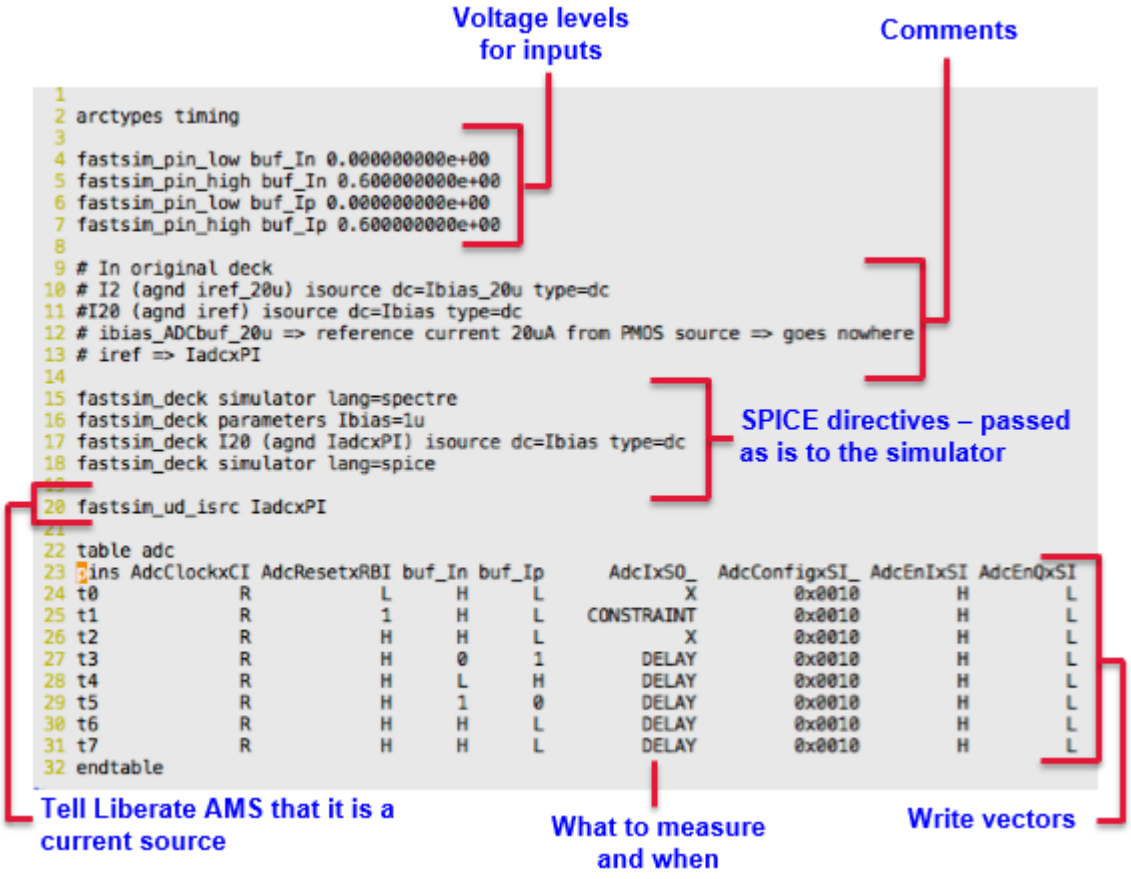
Using Truth Table-Based Characterization

As Liberate AMS is based on the Liberate MX technology, it is still possible for you to provide the stimuli based on tables. The `define_table` command points Liberate AMS to a specific table that reproduces the desired stimuli.

When using the truth table-based flow, specify the simulation interval that also defines the clock period to generate the stimuli, as shown below:

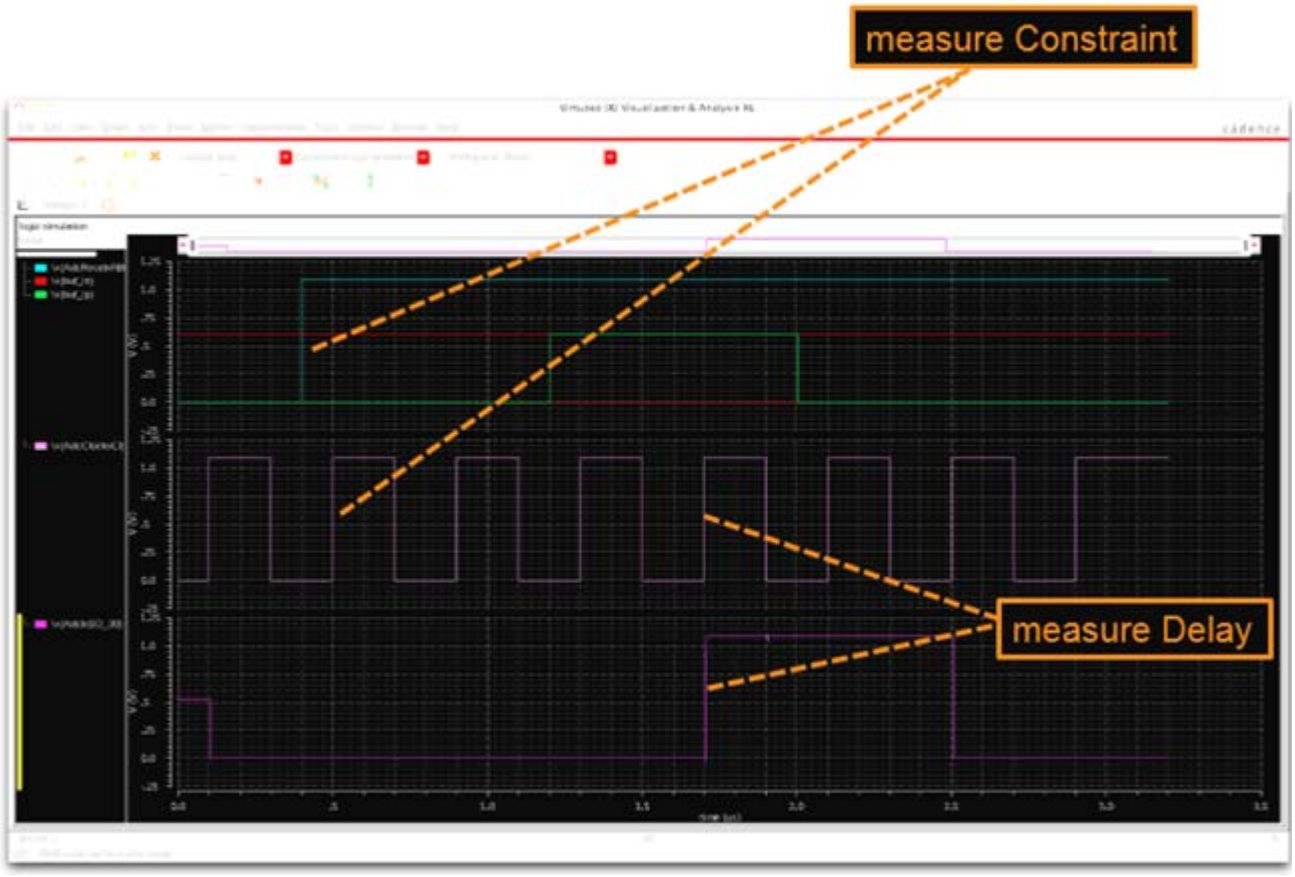
```
set_var px_simulation_interval 10e-09
```

The following figure illustrates the Liberate AMS timing input table format:




```
1
2  arctypes timing
3
4  fastsim_pin_low buf_In 0.000000000e+00
5  fastsim_pin_high buf_In 0.600000000e+00
6  fastsim_pin_low buf_Ip 0.000000000e+00
7  fastsim_pin_high buf_Ip 0.600000000e+00
8
9  # In original deck
10 # I2 (agnd iref_20u) isource dc=Ibias_20u type=dc
11 #I20 (agnd iref) isource dc=Ibias type=dc
12 # ibias_ADCbuf_20u => reference current 20uA from PMOS source => goes nowhere
13 # iref => IadcxPI
14
15 fastsim_deck simulator lang=spectre
16 fastsim_deck parameters Ibias=1u
17 fastsim_deck I20 (agnd IadcxPI) isource dc=Ibias type=dc
18 fastsim_deck simulator lang=spice
19
20 fastsim_ud_isrc IadcxPI
21
22 table adc
23 pins AdcClockxCI AdcResetxRBI buf_In buf_Ip      AdcIxSO_  AdcConfigxSI_ AdcEnIxSI AdcEnQxSI
24 t0             R            L      H      L           X          0x0010         H         L
25 t1             R            1      H      L    CONSTRAINT       0x0010         H         L
26 t2             R            H      H      L           X          0x0010         H         L
27 t3             R            H      0      1       DELAY          0x0010         H         L
28 t4             R            H      L      H       DELAY          0x0010         H         L
29 t5             R            H      1      0       DELAY          0x0010         H         L
30 t6             R            H      H      L       DELAY          0x0010         H         L
31 t7             R            H      H      L       DELAY          0x0010         H         L
32 endtable
```

The following figure illustrates a waveform generated from Liberate AMS table:



Supporting Complementary Signals

To define complementary input signals, the following two approaches can be used:

- If define_deck is used and the signals are complementary in the testbench, they will be complementary by default in the partitioned simulations; however, they will not be complementary in the top-level fastsim simulation. If complementary signals are required for correct DUT operation with fastsim, use the `define_table` command as explained below.
- If define_table is used, code the table such that equal and opposite transition occur on the same table line (see lines `t_3` and `t_4` below). Avoid using `"R"` and `"F"` table values for these inputs.

For example, the following table will drive complementary inputs on `CLKIN_T/CLKIN_C`, while using a `"B"` value on outputs `CLKOUT_T/CLKOUT_C` to allow for measurement using define_measure.

```
table.tbl

arctypes delay measure
simulation_interval 32e-9

table comp_drive_meas
pins RST_N   CLKIN_T CLKIN_C CLKOUT_C CLKOUT_T
init L       0       1       X        X
t_1  L       0       1       X        X
t_2  H       0       1       X        X
t_3  H       1       0       B        B
t_4  H       0       1       B        B
endtable

End of table.tbl
```

Measuring complementary signals requires the use of `define_measure` statements. In `template.tcl`, you can incorporate updates as suggested below:

```
#To measure complementary output signals, two define_measure are required assuming
the cell definition is as follows (set $cell to your cellname):


define_cell \
    -input { RST } \
    -clock { CLKIN_T CLKIN_C }\
    -output { CLKOUT_T CLKOUT_C } \
    -delay  delay_7x7 \
    -power  power_7x7 \
    $cell


#Set a threshold for the trigger at 50% of your power supply (held in variable
$voltage)
set v50 [expr $voltage*0.5]


#Set up measurement of rising edge of CLKIN_T to rising edge of CLKOUT_T - CLKOUT_C
#Set threshold for CLKIN_T at 50% of VDD, and CLKOUT_T-CLKOUT_C at 0V, which is when
#the outputs are equal.
#Note that you do not have to measure the zero cross of CLKIN_T/CLKIN_C because
#Liberate AMS forces both input waveforms to cross @ 50%.
```

```
define_measure \
    -name dval1 \
    -trig {CLKIN_T} \
    -trig_val $v50 \
    -trig_dir R \
    -targ {CLKOUT_T CLKOUT_C} \
    -targ_val 0 \
    -targ_dir R \
    -targ_type "delay" \
    -duration 0.5 \
    -keep max \
    $cell

#A second measure is required to use the results of the first measurement.
#The same trigger value must be used because this just triggers the measurement.

define_measure \
    -name dval2 \
    -trig {CLKIN_T} \
    -trig_val $v50 \
    -trig_dir R \
    -duration 0.5 \
    -equations dval1 \
    $cell

#The last step is to set the R to R arc for CLKIN_T to CLKOUT_T equal to result of
#the second measurement.

define_arc \
    -pin CLKOUT_T \
    -pin_dir R \
    -related_pin CLKIN_T \
    -related_pin_dir R \
    -measure dval2 \
    $cell
```

Using Static Mode

Static mode enables a vector-less usage of the Liberate AMS characterization tool. Working in this mode does not require vectors, tables, or testbenches. If a testbench or vectors (table file format .tbl) is provided, Liberate AMS automatically operates in dynamic mode instead.

In static mode, Liberate AMS conducts the following steps sequentially:

1. Determines the simulation vectors at the Channel-Connected Components (CCC) level automatically.
2. Performs CCC simulations.
3. Propagates the CCC output waveforms to its fanouts.
4. Simulates the fanout CCCs.
5. Analyzes statically the circuit for worst-case timing.

This cycle repeats until the entire measurement path is completed. These measurement path delays are computed and written to the output library in the same manner as the dynamic usage model. Timing report can be generated to help you visualize the details of the timing paths that comprise any library-level timing arc.

Setting Up Static Mode

To set up the Liberate AMS static mode, follow the command sequence given below:

1. define_cell
2. define_template
3. define_arc
   - Provide the user-defined arcs using this command.
4. `set_var` px_hybrid_enable `"static"`
   - Set this variable to run static mode only.
5. set_false_path `-through {A} -through {B C}`
   - Use the `set_false_path` command to consider the paths that include A && (B || C) as false path.
6. `set_var` px_spv_api `1`
   - Set this variable to enable SpiceVision (SPV) write out.

7. char_ams
8. write_library

Running Designs in Static Mode

To run a design in static mode, set the following variable:

```
set_var px_hybrid_enable "static"
```

In static mode, no testbench or vector (table) file is required. In other words, no `define_deck` or `define_table` is required in the Tcl characterization script.

Note: You can also opt to run a design in hybrid mode that enables use of dynamic and static characterization modes in the same run. For detailed information, see Using Hybrid Mode.

While opting to run your design in static mode, you also have the choice to use one of the following two methods:

- ***Method 1***: Characterize user-defined arcs only

  ```
  set_var px_hybrid_enable "static"
  set_var px_create_if_uda 1
  report_static_char -report_name design_static_timing.rpt
  char_ams -partsim $partsim -charsim $charsim
  ```

- ***Method 2***: Automatically enumerate possible delay and constraint arcs, and characterize them

  ```
  set_var px_hybrid_enable "static"
  set_var px_create_if_uda 0
  report_static_char -report_name design_static_autoarcs_timing.rpt
  char_ams -partsim $partsim -charsim $charsim
  ```

Understanding the Outputs of Static Mode

When you run Liberate AMS characterization in static mode, the following outputs are generated:

- `ldb` file
  - This file includes all probes timing tables.
- `./ams` directory
  - `<design_name>.static_timing.tcl` file

This file contains the main Tcl script inside a partition with commands such as following:

- `char_library -skip {power leakage} -ams -extsim spectre -thread 0 -static`

- <*design_name*>.static_timing.ldb.gz file
  - This file contains data related to all partitions, cells, and LDB. In addition, the file includes the CCC data.
- ./<*design_name*>.constraint_<*uniqueNumber*> subdirectory
  - Partition cell information is saved in the following files that are saved in this directory: `arc.tcl`, `cell.tcl`, and `sim.sp`.
- ./spv directory
  - This directory is created to enable CCC level for netlist and make SpiceVision reading easier.

- ./decks
  - All CCC simulation decks are saved in this directory.

Saving of all CCC simulation decks in the design can take up large disk space based on the design size.

Enabling Static Timing Report Through Tcl Command

The `report_static_char` Tcl command can be used to enable generation of the static timing report captured in the `static_timing.rpt` file.

Using report_static_char Tcl Command

Use the following syntax for the `report_static_char` Tcl command:

`-index_1 [ min | mid | max | all ]`

Specifies the `index_1` selection criteria using one of the following valid values: `min`, `mid`, `max`, or `all`.
Default: `mid`

`-index_2 [ min | mid | max | all ]`

Specifies the `index_2` selection criteria using one of the following valid values: `min`, `mid`, `max`, or `all`.
Default: `mid`

`-path [ worst | all ]`

Specifies the path selection criteria for a given arc using one of the following valid values: `worst` or `all`
Default: `worst`

`-type [ delay | setup | hold ]`

Specifies the supported types using one of the following valid values: `delay`, `setup`, or `hold`.
Default: `"*"` (reports all three types of arcs: `delay`, `setup`, and `hold`)

`-format [ load | cg | fanout ]`

Specifies the supported types using one of the following valid values: `load`, `cg`, or `fanout`.
Default: `""` (none of the extra information is reported

`-report_name <`*`string`*`>`

Specifies the file name of the generated timing report.
Default: `"static_timing.rpt"` (this file is by default saved in the `./ams` directory.

`-pin <`*`list`*`>` Lists the output or constrained pins.

`-related_pin <`*`list`*`>`

Lists the intput or related pins.

`-cells <`*`list`*`>` Lists the cell names.

Viewing Contents of a Sample Static Timing Report

The following figure shows the contents of a sample static timing report:

```
Path 1:
Type: setup_rising  rise_constraint
Pins:   pin: byteselb_0[0]  related: e2ph1
Probes: pin: Xdp/Xbyte[0]/Xmux[8]/XI9/sregs_mqb  related: Xdp/Xbyte[0]/Xmux[8]/net62
When:

Table Indices:  pin: 0.06  related: 0.06
Point                                 Load        Cg          Fanout      Incr        Path
-------------------------------------------------------------------------------------------
byteselb_0[0]                         2.222       1.112       6           0           0           r
Xdp/Xbyte[0]/net29                    16.73       4.954       36          0.03332     0.03332     f
Xdp/Xbyte[0]/Xmux[4]/net115           1.392       0.2752      2           0.00815     0.04147     r
Xdp/Xbyte[0]/Xmux[8]/d                2.038       0.2752      2           0.0191      0.06057     r
Xdp/Xbyte[0]/Xmux[8]/XI9/sregs_mqb    1.979       0.1884      2           0.03465     0.09522     r
data arrival time:                                                                    0.09522

e2ph1                                 30.41       13.69       82          0           0           r
Xcent/net80                           16.86       6.692       36          0.01678     0.01678     f
stgph1_0                              86.74       19.82       144         0.0108      0.02758     r
Xdp/Xbyte[0]/Xmux[8]/net62            2.948       0.7404      7           0.01075     0.03833     f
clock arrival time:                                                                   0.03833
-------------------------------------------------------------------------------------------
setup time:                                                                           0.05689
```

Using Hybrid Mode

When you need to use a dynamic testbench and perform static mode characterization in the same run, Liberate AMS provides the option to run the design in hybrid mode. Such designs require one of the following:

- A dynamic testbench to configure and simulate timing arcs associated with analog circuit, non full-swing voltage levels, and piece-wise linear (PWL) models driving the internal signals.

  OR

- A testbench to stimulate and exercise specific timing arcs. The rest of timing arcs are associated with digital logic.

Running Designs in Hybrid Mode

To run a design in hybrid mode,

1. Set the following variable:

```
set_var px_hybrid_enable "hybrid"
```

2. Provide a testbench to characterize the defined timing arcs:

```
define_deck -observation_window {2e-9} \
    -dut_inst I_dut ${deckdir}/margin.scs $cell
```

3. Include the definition of all timing arcs (`define_arc`) in the template file for dynamic testbench and static mode characterization.

Important

If a static mode arc is covered by a dynamic testbench, it will be characterized using dynamic mode. To override this behavior, use the `-static` option with the define_arc command.

In addition to the method described above, you can use the two additional methods described in Running Designs in Static Mode to run a design in hybrid mode. The only change will be in setting the `px_hybrid_enable` variable to `"hybrid"`.

Sample Tcl Script for Hybrid Mode

```
### AMS STATIC MODE with user-defined timing arcs (define_arc)
### AMS STATIC MODE with automatic timing arcs generation (no define_arc)

# SET CORNER
    set process tt
    set voltage 1.0
    set temperature 75
    set_operating_condition -temp $temperature -voltage $voltage

# SETUP DIRECTORY STRUCTURE
    set cell serializer
    set datadir [pwd]

# INPUT FILES
    set tcldir   ${datadir}/tcl
    set tmpldir  ${datadir}/tcl
    set tbldir   ${datadir}/tcl
    set spicedir ${datadir}/spice
    set deckdir  ${datadir}/deck

# OUTPUT FILES
    set outdir ${datadir}/out_hybrid
```

```
    set_var px_dir ${outdir}/ams ;#specify partition sim dir
    set ldbdir ${outdir}/ldb ; exec mkdir -p $ldbdir
    set libdir ${outdir}/lib ; exec mkdir -p $libdir

# READ IN SPICE
    set spectre_netlist true
    set netlistfile ${spicedir}/${cell}.scs

    set modelfile ${spicedir}/models/include_${process}
    set_var extsim_model_include $modelfile
    set_var extsim_deck_include 1
    set_var px_read_spice_exit_on_missing_file 1
    ## define primitive models
    define_leafcell -type nmos -pin_position {0 1 2 3}  {g45n1svt g45n1hvt}
    define_leafcell -type pmos -pin_position {0 1 2 3}  {g45p1svt g45p1hvt}
    define_leafcell -type black_box -pin_position {0 1 2} {g45rnsnp}

    puts "reading netlist ..."
    if {$spectre_netlist} {
        read_spice -format spectre $netlistfile
    } else {
        read_spice $netlistfile
    }

# SPECIFY RAILS
    set_vdd avdd $voltage
    set_gnd avss 0

# SPECIFY SPICE ENGINES and OPTIONS
    set partsim "aps"
    set charsim "aps"

    if {$partsim == "aps"} {
        set_var fastsim_cmd "spectre"
        set_var fastsim_cmd_option "+aps +spice -64 -format fsdb +mt=4
        +errpreset=liberal"
        set partsim "spectre"
    }

    if {$charsim == "aps"} {
        set_var extsim_cmd "spectre"
```

```
        set_var extsim_cmd_option "+aps +spice -64 -mt +errpreset=moderate"
        set_var extsim_option "soft_bin=allmodels"
        set charsim "spectre"
    }

    set_var extsim_deck_dir ${outdir}/decks

# DEFINE MACRO TEMPLATE
    source ${tmpldir}/template_hybrid.tcl
    # source ${tmpldir}/template_static.tcl
    # source ${tmpldir}/template_static_autoarcs.tcl

# SPECIFY TESTBENCHES/TABLES (required for dynamic mode)

    define_deck -observation_window {2e-9} -dut_inst I_dut ${deckdir}/margin.scs
    $cell
    define_deck -observation_window {2e-9} -dut_inst I_dut -arctypes "power"
    ${deckdir}/power.scs $cell
    define_deck -observation_window {2e-9} -dut_inst I_dut -arctypes "leakage"
    ${deckdir}/leakage.scs $cell

########## hybrid flow settings
set_var px_hybrid_enable "hybrid"
set_var px_create_if_uda 1
set_var px_spv_api 1
report_static_char -report_name hybrid_timing.rpt
######### end of hybrd flow settings

########## static flow with user-defined timing arcs (define_arc)
set_var px_hybrid_enable "static"
set_var px_create_if_uda 1
set_var px_spv_api 1
report_static_char -report_name static_timing.rpt
######### end of static flow with user-defined arcs
########### static flow with automatic timing arcs generation (no need for
define_arc)
set_var px_hybrid_enable "static"
set_var px_create_if_uda 0
set_var px_spv_api 1
report_static_char -report_name static_autoarcs_timing.rpt
######### end of static flow with auto-arcs generation
```

```
# MAIN COMMAND
    char_ams -partsim $partsim \
        -charsim $charsim

# WRITE MODELS
    write_ldb ${ldbdir}/${cell}.ldb
    write_library \
        -rename -filename ${libdir}/${cell}_hybrid.lib ${cell}
```

5

Liberate AMS Characterization Stages

To create instance-specific memory models for timing, power, and noise, Liberate AMS mixed-signal characterization solution enables fast and accurate characterization of large mixed-signal macros through different stages discussed in this chapter.

Timing Characterization

The various steps involved in running characterization for timing information are shown below:

Static Analysis

In this step, the tool reads in the netlist and automatically identifies probe locations and also partitions for pin capacitance characterization.

Vector Generation

Liberate AMS uses a dynamic (simulation driven) approach to characterization. Liberate AMS is built on Liberate MX technology which means that the tool needs to be provided with table(s) of vectors. The generation of this table(s) of vectors is automated by utilizing the existing test benches for simulating the block.

The objective is to provide the tool with vectors to:

- Setup block in a specific state.
- Transition circuit inputs to “exercise” the desired ARC(s) to output and internal nodes.

The testbenches are run to capture the stimulus information at the input pins of the block. The tool then are then creates the required internal table format automatically for the full instance simulation.

Full Instance Simulation

The table(s) generated in the previous steps is used as vectors to drive simulation for the full instance. This simulation is usually done using a FastSPICE solver, as the intent is to capture only the activity in the circuit.

Partitioning

The waveform data from the full instance simulation is read back into the tool. This data is used to:

- Prune the statically-probed locations with the dynamic information from simulation.
- Generate partitions from main netlist for every arc that needs to be characterized.

The generated partitions include the associated side path circuitry including the associated parasitic network. The partitions are typically less than a 2000 transistors.

Characterization

Using a full SPICE solver (like Spectre® APS), accurate characterization is performed on all the partitions created for all slews and loads.

Model Generation

Once characterization of all the arcs are complete, the results are merged and a final `.lib` file is written out.

Power Characterization

Power characterization does not involve any partitioning and power numbers are calculated by running simulation on the full instance.

Pin Capacitance and Noise (SI) Models

The approach for generating pin capacitance and noise models is as follows:

1. Create static boundary partition consisting of first level of channel connected (CCC) transistors attached to each I/O port (typically less than 50 transistors).
2. Characterize each boundary partition separately for pin capacitance and noise (CCSN/ ECSMN).
3. Integrate the characterized data back into final model.

6

Liberate AMS Commands

This chapter lists the Tcl commands that can be used with Cadence Virtuoso Liberate AMS.

Available Commands

Table 6-1 lists the commands that can be used with Liberate AMS.

- Commands that have a `Liberate` link are shared with other Liberate tools. Descriptions for these commands are covered in *Virtuoso Liberate Reference Manual*.
- Commands that have a `Liberate AMS` link can be used with Liberate AMS. The descriptions for these commands are included on the indicated pages in later sections of this chapter.

Table 6-1 Commands Available for Use With Liberate AMS

Command	Link to Description
`add_cell_attribute`	Liberate
`add_lib_attribute`	Liberate
`add_margin`	Liberate AMS: 78
`add_pin_attribute`	Liberate
`char_ams`	Liberate AMS: 80
`check_delay_monotonicity`	Liberate
`compare_ccs_nldm`	Liberate
`compare_library`	Liberate
`compare_path`	Liberate AMS: 83
`define_arc`	Liberate AMS: 85
`define_bundle_pins`	Liberate

Table 6-1 Commands Available for Use With Liberate AMS, *continued*

define_bus	Liberate
define_cell	Liberate AMS: 100
define_deck	Liberate AMS: 105
define_duplicate_pins	Liberate AMS: 109
define_group	Liberate
define_index	Liberate AMS: 110
define_input_waveform	Liberate
define_leafcell	Liberate AMS: 112
define_leakage	Liberate AMS: 114
define_map	Liberate
define_measure	Liberate AMS: 115
define_table	Liberate AMS: 124
define_template	Liberate AMS: 125
get_var	Liberate
merge_library	Liberate
packet_slave_cells	Liberate
printvars	Liberate
px_set_domainprop	Liberate AMS: 128
read_ldb	Liberate
read_library	Liberate
read_spice	Liberate
set_client	Liberate
set_constraint	Liberate
set_constraint_criteria	Liberate
set_default_group	Liberate
set_dependent_load	Liberate
set_driver_cell	Liberate
set_false_path	Liberate AMS: 130

Table 6-1 Commands Available for Use With Liberate AMS, *continued*

set_gnd	Liberate AMS: 131
set_max_fanout	Liberate
set_operating_condition	Liberate
set_pin_capacitance	Liberate
set_pin_delay_threshold	Liberate
set_pin_gnd	Liberate
set_pin_slew_threshold	Liberate
set_pin_vdd	Liberate
set_receiver_cap_thresholds	Liberate
set_rsh_cmd	Liberate
set_three_state	Liberate
set_units	Liberate
set_var	Liberate
set_vdd	Liberate AMS: 133
set_virtual	Liberate AMS: 134
write_datasheet	Liberate
write_ldb	Liberate AMS: 135
write_library	Liberate AMS: 136
write_template	Liberate
write_userdata_library	Liberate
write_verilog	Liberate AMS: 143

Commands Used in Liberate AMS

This section describes the following commands that are specifically for Liberate AMS:

- add_margin
- char_ams
- compare_path
- define_arc
- define_cell
- define_deck
- define_leafcell
- define_leakage
- define_measure
- define_table
- define_template
- px_set_domainprop
- set_gnd
- set_vdd
- set_virtual
- write_ldb
- write_library
- write_verilog

For information about additional commands that can be used with Liberate AMS but that are also shared with other Liberate tools, see the links given in Table 6-1.

Command options that are preceded with a hyphen (-) are optional, except where explicitly indicated.

Tip

All commands have a `-help` option. When you run a command that includes the `-help` option, Liberate AMS outputs a message that lists the currently available options for that command. Be aware that there might be options that print out when the `-help` option is used that are not officially supported. The only supported options are those that are documented in this manual. When you use the `-help` option, all other command options are ignored.

add_margin

Adds margin (padding) to values in the library. Margin is always added to all cells in a library.

Options

`-abs <value>`	Specifies the absolute amount of margin to add. Default: `0.0` (no margin)
`-cells {list}`	List of cells
`-constraint`	Applies the same margin to all constraint types: `setup`, `hold`, `recovery`, `removal` and `mpw` (minimum pulse width).
`-direction <rise | fall | both>`	Specifies direction of data to add margin. Default: `"both"`
`-index_1 {list}`	Specifies the `index_1` points. Default: all points
`-index_2 {list}`	Specifies the `index_2` points. Default: all points
`-pin {list}`	List of pins.
`-related {list}`	List of related pins.
`-rel <value>`	Specifies the relative amount of margin to add. Note that this option always makes the library value larger, whether the original value was positive or negative. Default 0.0 (0%). Example: 0.05 = 5% margin.

`-type {list}`

The type of data to be modified. If the type option is not specified, the requested margin will be applied to all data types. Supported types are: `cap`, `constraint`, `delay`, `delay_ccs` *(only accepts positive "abs" margin)*, `hidden`, `hold`, `leakage`, `mpw`, `power`, `recovery`, `removal`, `retain`, `retain_ccs` *(only accepts positive "abs" margin)*, `retain_trans`, `setup`, `trans`. Default: apply margin to `all` types.

Note: Margin can be added to the `power` and `hidden` types on individual cells by specifying a cell name with the `-cells` option. In this case, the arc must be completely specified, that is, the pin, related, and when must also be specified. For example:

```
add_margin \
    -type {power|hidden} \
    -cells {celllists} \
    -pin {pin lists} \
    -related {related_pin list} \
    -when "when_condition" \
    -rel rel \
    -abs abs
```

`-when "string"`

State dependent arc.

This command can be used after `char_ams`, `read_ldb`, and `read_library`, but it must be used before model generation such as with `write_library`. Multiple `add_margin` commands can be specified.

Example:

```
read_ldb test.ldb.gz
write_library no_margin.lib

# Add 10% to power
add_margin -type power -rel 0.1

# Add 50ps to delay
add_margin -type delay -abs 50e-12
write_library margin.lib
```

char_ams

Controls the characterization of AMS blocks. Each AMS block listed in a `define_cell` command is characterized if the SPICE `subckt` definition for that block is defined in the netlists passed to the `read_spice` command.

Options

`-ccs` Characterize CCS (delay) data.

`-ccsn` Characterize CCSN (noise) data.

`-char_params {`*`parameters`*`}`

Parameters for characterization. Default: `none`

`-char_script <`*`script_name`*`>`

Specifies the path and name of a Tcl script to be sourced prior to characterization. Default: `none`.

`-charsim "`*`simulator_name`*`"`

Specifies the SPICE simulator used during characterization. Default: `aps`

`aps`	Use the Spectre Accelerated Parallel Simulator (APS).
`spectre`	Use the Spectre simulator.
`xps`	Use the Spectre eXtensive Partitioning Simulator (XPS).

`-charthread <`*`number`*`>`

Specifies the maximum number of threads to use during characterization. Default: `0` (Number of threads is determined automatically).

`-ecsm` Characterize ECSM (delay) data.

`-ecsmn` Characterize ECSMN (noise) data.

`-part_arc_thread <`*`number`*`>`

Specifies the number of threads to use for updating the arcs after reading the fastsim waveforms. This requires corresponding number of client licenses. Default: `1` (Use one CPU thread).

`-part_fastsim_thread <number>`

Specifies the number of threads to use for fastsim partitioning simulation (overwrites the `-partthread` value only for the number of threads related to fastsim functionality). This requires corresponding number of client licenses. Default: `0` (Use all available CPU threads).

`-part_wave_thread <number>`

Specifies the number of threads to use for waveform update after fastsim. This requires corresponding number of client licenses. Default: `1` (Use one CPU thread).

`-part_write_thread <number>`

Specifies the number of threads to use for writing partitions (overwrites `-partthread` value for the number of threads for writing partitions). This requires corresponding number of client licenses. Default: `0` (Use all available CPU threads).

`-partsim "simulator_name"`

Specifies the SPICE simulator to be used for partitioning. Default: `aps`

`aps`	Use the Spectre APS.
`spectre`	Use the Spectre simulator.
`xps`	Use the Spectre XPS.

`-partthread <number>`

Specifies the maximum number of threads to use for fastsim partitioning simulations and for writing partitions. This requires corresponding number of client licenses.
Default: `0` (Use all available CPU threads).

Note: Define a valid value for the `-partthread` option instead of keeping it unlimited. (Recommended)

`-user_arcs_only`

Specifies to characterize only the user-specified arcs.

`-write_thread <number>`

Specifies the maximum number of threads to use during the writing portion of preprocessing. Default: `0`. (Number of threads is determined automatically).

The different `-thread` arguments of `char_ams` define the maximum number of threads to use on the current machine. If you do not specify any of these `-thread` arguments, Liberate AMS automatically uses multiple threads based on the available CPUs. Steps that are multithread-capable are FastSPICE simulation, results acquisition, arcs selection, and file I/O operations. The number of parallel FastSPICE runs is determined by the maximum of the number of different AMS table files provided and the number specified with the different `-thread` arguments.

The `-char_script` argument specifies the name of a Tcl script to be sourced prior to characterization. Specify the complete path to the Tcl script. It is possible to specify commands that apply only to a specific characterization step—rather than all—by using variables `g_timing_char`, `g_inputcap_char`, `g_noise_char`, and `g_power_char`. Such variables are automatically set during the corresponding characterization run and can therefore be used to selectively apply settings to a specific step.

Example

```
char_ams -partsim $fastspice -charsim $fullspice
```

compare_path

Compares automatically the top-level and partition-level incremental delay path files to quickly identify where the top level and partition diverge.

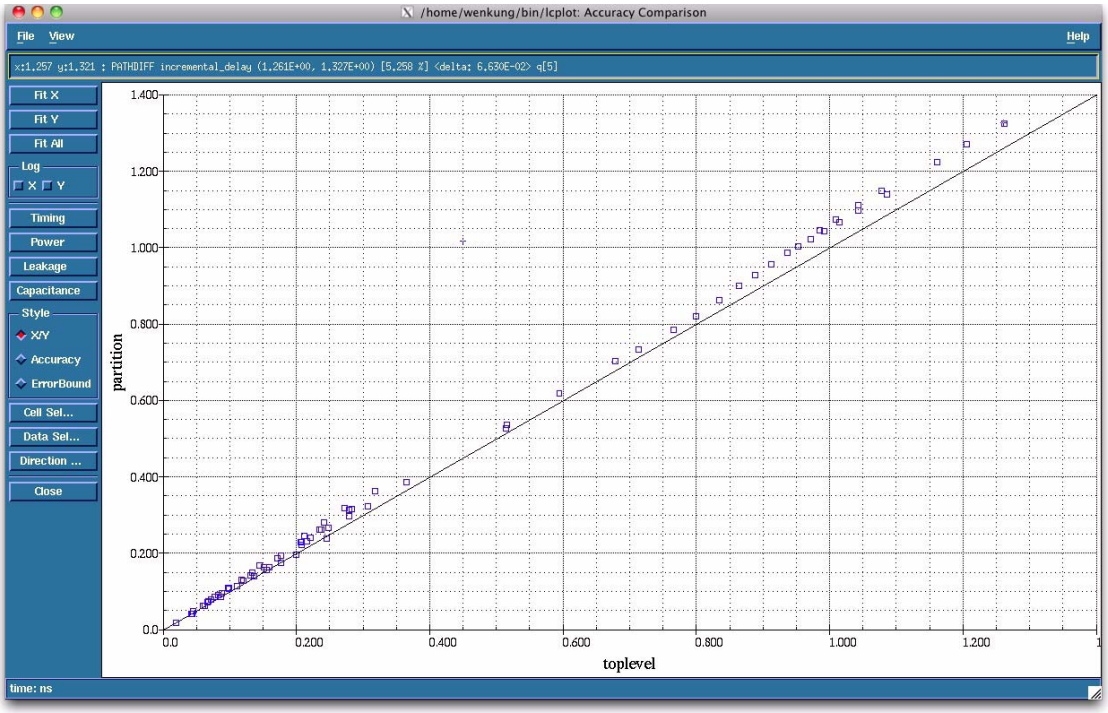
Options

`-abstol <double>`	Specifies the error flag tolerate in nanoseconds (Default: 1e-2)
`-debug < 0 | 1 >`	Enables debug mode (Default: 0)
`-gui <string>`	Output the comparison result in graphical and lwave formats (Default: diff)
`-lcplot < 0 | 1 >`	Automatically output lcplot on result (Default: 0)
`-lwave < 0 | 1 >`	Automatically output lwave on result (Default: 0)
`-part_report <string>`	A report of the cumulative delay path as seen from the partition level (RealSPICE numbers). (Default: sim.rpt)
`-report <string>`	Output the comparison result in text format (Default: diff.rpt)
`-top_report <string>`	A report of the cumulative delay path as seen from the top level (FastSPICE numbers). (Default: sim.top.rpt)

The `sim.top.rpt` file is automatically generated at each run and for each delay partition. It contains the cumulative delay along the path as seen from the top level. The sim.rpt is automatically generated at each run and for each partition. It contains the cumulative delay along the path, as seen from the partition level. These reports help determine potential issues in the partitioning process itself. The command must be run in a delay, constraint, or measure partition directory.

The comparison report is output into both a graphical format (`diff.lcplot`) and a text format (`diff.rpt`).

The following is an example of a graphical version of the comparison report output by the `compare_path` command:



define_arc

Allows to override Liberate AMS auto-probing/partitioning, specify a `when` condition for an arc, and specify retain arcs that should be characterized.

Options

`-attribute {altos_clone_arcs <`*`string`*`>}`

Instructs Liberate AMS to duplicate the characterization results across equivalent arcs involving a bus.

It can also be used to clone specific data types from a pin to a completely unrelated pin. These can be bus pins or non-bus pins. For more information, see Using the -attribute {altos_clone_arcs <string>} option.

`-attribute {altos_px_bisection 1}`

Instructs Liberate AMS to use bisection for constraint (setup/hold) characterization instead of path-delay method. For example:

```
# Use bisection method to characterize setup constraint
define_arc \
    -type setup \
    -pin {addr[0]} \
    -related_pin {clk} \
    -attribute {altos_px_bisection 1} \
    $cell
```

`attribute {altos_px_bundle <`*`bundle_name`*`>::<`*`bundle_criterion`*`>}`

Instructs Liberate AMS on how to choose a representative of a specific measurement group before going to full SPICE characterization, where `<bundle_name>::<bundle_criterion>` means,

`<bundle_name>` == any string

`<bundle_criterion>` == **<**`min | max | absmin | absmax | average`**>**

All arc definitions of the same arc, with the same string in the `altos_px_bundle`, are considered for the characterization of the arc. The `min` or `max` value specifies whether the minimum or the maximum of all `define_arcs` is used in characterization. The `bundle_name` value can be any string, but should be the same for all definitions of the same arc and not conflicting with the names chosen for different arcs.

This is usually used along with `-measure` to consider multiple measurement definitions for a single arc. In that case, the specified `<bundle_criterion>` operation is performed on all measurement arcs with matching `<bundle_name>` and only the resulting one generates a partition and characterizes with full SPICE.

The `<bundle_criterion>` is applied to the value of the first equation evaluated by the arc if there are more than one equations. For example, the following three measurement arcs:

```
define_arc -measure ABC_0
define_arc -measure ABC_1
define_arc -measure ABC_2
```

…will generate a partition and each will be characterized with full SPICE, but for the following ones:

```
define_arc -measure ABC_0 -attribute {altos_px_bundle
ABC::min}
define_arc -measure ABC_1 -attribute {altos_px_bundle
ABC::min}
define_arc -measure ABC_2 -attribute {altos_px_bundle
ABC::min}
```

…only the minimum one will generate a partition and be characterized with full SPICE.

The two different `min_period` components are measured using define_measure and using `altos_px_bundle` to take the maximum of both. In the example below, both components get measured in multiple cycles during fastsim. At the end of fastsim, `ck_min_period` gets one worst (maximum) value for which partition is created and then the partition is run with full SPICE.

For example:

```
set vc50 [expr $voltage * 0.5]
set vs90 [expr $voltage * 0.9]
set vc10 [expr $voltage * 0.1]


define_measure -name tcyc1_1 -trig CKR -trig_val
$vc50 -targ $probe1_1 -targ_type delay -targ_val
$vc90 $cell
define_measure -name tcyc1_2 -trig CKR -trig_val
$vc50 -targ $probe1_2 -targ_type delay -targ_val
$vc10 $cell


define_measure -name tcyc1 -trig CKR -trig_val $vc50
-targ_type delay -equations {"tcyc1_1-tcyc1_2"} $cell
define_arc -type min_period -measure tcyc1 -pin CKR -
pin_dir $dir_r -attribute [list altos_px_bundle
ck_minperiod::max] $cell


define_measure -name tcyc3 -trig CKR -trig_val $vc50
-targ $probe3 -targ_type delay -targ_val $vc90 $cell
define_arc -type min_period -measure tcyc3 -pin CKR -
pin_dir $dir_r -attribute [list altos_px_bundle
ck_minperiod::max] $cell
```

`-attribute {altos_px_existence <string>}`

Identifies the existence of specified switching. It uses the string as a cycle identifier in the table to check if it actually does switch as dictated by the arc. Example:

```
altos_px_existence <identifier>
```

Where `<identifier>` is `<table_id>::<mode_id>::<line_id>` and:

- `<table_id>` is an existing table file name (used in the run)
- `<mode_id>` is an existing table name (used in the run inside table_id)
- `<cycle_id>` is the table cycle name for the specific cycle that needs to be verified

The existence of the specified switching of all involved measure points is checked, starting at the simulation time identified by the attribute and for a number of sub-intervals, as indicated by the corresponding measure `-duration` option. *(See code example #3 below.)*

`-attribute {altos_px_force_timing_type <type>}`

Forces a `timing_type` for a specific arc. For example, for a delay arc that models a dynamic output, force the `timing_type` to be `"rising_edge"` instead of the automatically found `"combinational"`.

```
define_arc \
    -pin {dataout} \
    -related_pin {clk} \
    -attribute {altos_px_force_timing_type rising_edge} \
    $cell
```

`-attribute {altos_px_simulation_interval <value>}`

Specifies a simulation interval on an arc-basis.

There are two other methods for specifying a simulation interval:

- Globally with the px_simulation_interval variable.
- Within a table, using the `simulation_interval` command as explained in the Specifying Input Stimuli section. Also, see the px_simulation_interval section for a listing of precedence when there are multiple definitions of simulation interval. Example:

```
define_arc -attribute {altos_px_simulation_interval
20e-9}
```

`-attribute {altos_px_autoprobing_skip_probe "<list_of_nodes>"}`

Specifies nodes to skip when considering probes during automatic probing for setup / hold characterization.

`-attribute {altos_px_autoprobing_skip_probe_regexp "<list_of_regular_expressions>"}`

Using regular expressions, specifies nodes to skip when considering probes during automatic probing for setup / hold characterization.

`-attribute {altos_px_autoprobing_skip_rel_probe "<list_of_nodes>"}`

Specifies related nodes to skip when considering probes during automatic probing for setup / hold characterization. For example, the following instructs to not use `CLK` as related probe for `CLK->EZ` setup:

```
define_arc \
    -type setup \
    -pin EZ \
    -related_pin CLK \
    -attribute {autoprobing_skip_rel_probe CLK} \
    $cell
```

`-attribute {altos_px_autoprobing_skip_rel_probe_regexp "<list_of_regular_expressions>"}`

Using regular expressions, specifies related nodes to skip when considering probes during automatic probing for setup / hold characterization. For example, the following instructs to not use nodes that begin with "BL" or "WORD" for probes:

```
define_arc \
    -type setup \
    -pin EZ \
    -related_pin CLK \
    -attribute {autoprobing_skip_rel_probe_regexp BL* WORD*} \
    $cell
```

`-attribute {altos_autoprobing_level <value>}`

Overwrites the values of both `px_autoprobing_hold_level` and `px_autoprobing_setup_level`.

`-deck <string>`

Specifies the full path to a user testbench SPICE deck that the defined arc corresponds to. Typically, this should be one of the strings specified using one of the define_deck commands. For example:

```
set my_test_bench1 "test1.scs"
set my_test_bench2 "test2.scs"
define_deck ....  $my_test_bench1
define_deck ....  $my_test_bench2


define_arc ....  -observation_window 15e-9 -deck $my_test_bench1
define_arc ....  -observation_window 30e-9 -deck $my_test_bench2
```

Note: The `-deck` and `-observation_window` options must be used together to uniquely associate an arc with a single observation window and a single testbench.

`-delay_threshold {list}`

Specifies the four delay measurement thresholds (`in_rise`, `in_fall`, `out_rise`, `out_fall`).

`-equation <"equation">`

Allows a SPICE equation to be used in place of a characterized value. Any valid equation can be used. The calculated value is used instead of running simulation to generate a value. (See also the `-name` option.)

Example:

```
define_arc -type mpw -pin CLK -pin_dir rise -name "mpwh" $cell
define_arc -type mpw -pin CLK -pin_dir fall -name "mpwl" $cell
define_arc -type min_period -pin CLK -equation "mpwl+mpwh" $cell
```

`-extsim_deck_header`

Allows to provide external simulator commands directly to the external simulator on an individual arc basis without using the Liberate process or reviewing them. This argument is intended to be used when an external simulator is used (refer to the `-extsim` argument of the char_ams command). It is a local arc specific version of the `extsim_deck_header` variable. As Liberate does not parse the string specified by this argument, ensure that the contents are valid and consistent with the arc simulation. The value string can contain the return character ("\n"). The value string is included at the top of simulation deck. For example:

```
define_arc -extsim_deck_header ".ic n128 0" -related_pin ck -pin Q ...
```

`-force` Allows only the probes specified by the `-probe` and `-related_probe` arguments.

Note: The use of `-probe` and `-related_probe` options (when `-force` is not used) adds probes to the list of considered probes. The worst case of the Liberate AMS-determined probes and the define_arc-declared probes are used in the partition simulations.

`-measure "name"` Name of associated `define_measure` command.

`-name "name"` Name (identifier) of the value produced by characterization. This name can be used as part of an equation to calculate a value.

`-observation_window <double>`

Specifies the duration (in seconds) of observation window used to infer causality between related_pin and pin for the defined arc. When this option is specified, it overrides any observation window values specified using the define_deck command. The `define_deck` windows are applied to all arcs unless this arc-level override is provided.

Note: The `-observation_window` option must be used together with the `-deck` option. This enforces explicit correspondence of an arc to a single observation window for a single testbench deck.

`-pin {pins}` List of destination pins or buses for the arc. (REQUIRED)

`-pin_dir <R | F>` Transition direction (R = rise, F = fall) of pin(s). If direction is not specified, arcs are created only for R.

`-pin_probe_threshold {list}`

List of pin monitor nodes thresholds.

`-probe {names}` List of names of data nodes to monitor for setup/hold characterization.

`-probe_dir <R | F>` Transition direction (R = rise, F = fall) of probe pin(s). If direction is not specified, arcs are created only for R.

`-related_pin {pins}`

List of related pins or buses for the arc.

`-related_pin_dir <R | F>`

Transition direction (R = rise, F = fall) of related pin(s). If direction is not specified, arcs are created only for R.

`-related_probe {names}`

List of names of clock nodes to monitor for setup/hold characterization.

`-related_probe_dir <R | F>`

Transition direction of related probe pin(s).

`-related_probe_threshold {list}`

List of related monitor nodes thresholds.

`-start_measure_after <double>`

Specifies to wait for the specified duration (in seconds) before evaluating this arc. This option must be used with the `-observation_window` option.

Note: If this option is not specified, it defaults to `0` and starts looking for the specified pin transitions at the beginning of the simulation. Note that this value is NOT derived from the define_deck command even if the arc-level overridden observation window of the corresponding testbench deck and arc matches, and will default to `0` if not specified explicitly on the arc.

`-slew_threshold {list}`

Four slew measurement thresholds (`lower_rise`, `upper_rise`, `lower_fall`, `upper_fall`).

`-static` Enables characterization of the arcs in static mode even if a table or deck is defined while using hybrid mode with timing arcs.

Note: This option applies only in Liberate AMS.

`-type <arc_type>` Type of arc. This option accepts one of the following values: `combinational`, `hold`, `max_clock_tree_path`, `min_clock_tree_path`, `minperiod`, `mpw`, `non_seq_hold`, `non_seq_setup`, `power`, `retain`, or `setup`. Default: `combinational`

The following are examples of how this argument can be used with `min_clock_tree_path` and `max_clock_tree_path`:

```
define_arc -type min_clock_tree_path -pin CKLA -pin_dir R
$cell
define_arc -type min_clock_tree_path -pin CKLA -pin_dir F
$cell
define_arc -type max_clock_tree_path -pin CKLA -pin_dir R
$cell
define_arc -type max_clock_tree_path -pin CKLA -pin_dir F
$cell
```

`-when <expression>` Conditional expression to be associated with the arc. It defines the logic conditions of the other pins of the cell to enable the arc using the Liberty™ `when` syntax. It corresponds to the Liberty `when` attribute.

`{cell_names}` List of cells.

Liberate AMS automatically extracts arcs from the provided table files. However, the `define_arc` command allows you to:

- Override Liberate AMS auto probing/partitioning.
- Specify a `when` condition for an arc.
- Specify retain arcs that should be characterized.

Using the -attribute {altos_clone_arcs <string>} option

The `altos_clone_arcs <string>` attribute is used to clone (duplicate) characterization results to other arcs. These can be bit of a bus; whole busses; or even completely unrelated pins. `<string>` is a list of `{related_pin | pin}` pairs that should receive the characterization results.

Note how it works internally. The `altos_clone_arcs` attribute is usually derived internally by Liberate AMS to instruct the tool on how to duplicate the characterization results across equivalent arcs involving a bus. For example, for an access-time characterization—`CLK` to bus `Q[31:0]`—Liberate AMS, by default, characterizes only the worst case among all possible `CLK` to pin `Q[i]`, where `Q[i]` is a pin of bus `Q`. The result will then be "cloned" or copied to the remaining bits of the bus.

You can overwrite or augment the automatic setting of this attribute in the following ways:

Determine the ranges for cloning. It might be necessary to model arcs involving a bus not as a whole, but separated in to different groups. For example, for constraint arcs on an address bus `A[7:0]`, it may be necessary to group column and row addresses in to separate groups, such as `A[0:2]`, `A[3:6]`, and `A[7]`. The specific grouping can then be specified in the define_arc itself and it will be respected when the `clone_attribute` attribute is generated. For this specific example, you would issue three separate `define_arc` commands as shown below:

```
define_arc -type setup A[2:0] ...
define_arc -type setup A[6:3] ...
define_arc -type setup A[7] ...
```

and the tool will infer that three instead of one representative partitions and characterization runs will be needed to model these arcs in the final library.

Note: Having different values for different bits of the same bus requires the library to be written in a bit-blasted format. For this, you need to use the `-expand_buses` option with the `write_library` command.

Important

Use one arc characterization value for another arc.

The following clones the hold arc from pin `ena1` to `ena2`:

```
define_arc
    -type hold \
    -pin {ena1} \
    -related_pin {clk} \
    -attribute {altos_clone_arcs "clk ena2"} \
    myCell
```

This following clones the setup arcs from bus `addrA` to bus `addrB`:

```
set clone_arcs_str ""
for {set i 0} {$i < 8} {incr i} {
    set clone_arcs_str [concat $clone_arcs_str "clk addrB<$i>"]
}

define_arc
    -type setup \
    -pin {addrA<7:0>} \
    -pin_dir R \
    -when "!en" \
    -related_pin {clk} \
    -related_pin_dir R \
    -attribute [list altos_clone_arcs $clone_arcs_str] \
    myCell
```

Examples of define_arc

Example 1

```
# force when conditions on bypass arcs
define_arc -type combinational -pin dout[63:0] \
           -related_pin clk -when "BYP" $cell
define_arc -type combinational -pin dout[63:0] \
           -related_pin clk -when "!BYP" $cell

#define_arc to enable measurement flow to generate min period timing groups,
# define_arcs for minperiod
```

Example 2

When you add the following to your run:

```
define_arc -type minperiod -pin {clk} <cell>
```

A new timing group will be generated in the library as following:

```
pin (CLK) {
    clock : true;
    direction : input;
    capacitance : 0.0100244;
    rise_capacitance : 0.0101743;
    rise_capacitance_range (0.00756358, 0.0116889);
    fall_capacitance : 0.00987448;
    fall_capacitance_range (0.00711549, 0.0117947);
    timing () {
        related_pin : "CLK";
        timing_type : minimum_period;
        rise_constraint (mpw_constraint_template_7x7) {
        index_1 ("0.004, 0.015, 0.038, 0.083, 0.174, 0.355, 0.717");
        values ( \
        "4.66069e-10, 4.69605e-10, 4.75735e-10, 4.83523e-10, 4.93672e-10,
        5.06612e-10, 5.23597e-10" \
        );
    }
    fall_constraint (mpw_constraint_template_7x7) {
                  index_1 ("0.004, 0.015, 0.038, 0.083, 0.174, 0.355, 0.717");
                  values ( \
                 "4.66069e-10, 4.69605e-10, 4.75735e-10, 4.83523e-10, 4.93672e-10,
        5.06612e-10, 5.23597e-10" \
                  );
        }
      }
```

Example 3

The following syntax will trigger a check for `wgbt_r<7>` rising and `Xg/Xgcolr/Xgc<3>/phi1wx` falling at the simulation time corresponding to cycle `write00` in table functional of table file, `timing.tbl`.

```
define_measure \
    -name      MCS_GWD_CLK_DIF_RR1_sig1 \
    -trig      {clk} \
```

```
    -trig_dir  rise \
    -trig_val  0.45 \
    -trig_d    "" \
    -targ      {"wgbt_r<7>" ""} \
    -targ_type "delay" \
    -targ_dir  rise \
    -targ_val  0.72 \
    -targ_d    "" \
    -failed_val "" \
    G40SP16384X4R3F1VTHSBSIR

define_measure \
    -name      MCS_GWD_CLK_DIF_RR1_sig2 \
    -trig      {clk} \
    -trig_dir  rise \
    -trig_val  0.45 \
    -trig_d    "" \
    -targ      "Xg/Xgcolr/Xgc<3>/phi1wx" \
    -targ_type "delay" \
    -targ_dir  fall \
    -targ_val  0.72 \
    -targ_d    "" \
    -failed_val "" \
    G40SP16384X4R3F1VTHSBSIR

define_measure \
    -name      MCS_GWD_CLK_DIF_RR1 \
    -keep      max \
    -duration  0.25 \
    -trig_d    "" \
    -targ      {"wgbt_r<7>" ""} \
    -targ_type "delay" \
    -targ_dir  rise \
    -targ_val  0.72 \
    -targ_d    "" \
    -failed_val "" \
    G40SP16384X4R3F1VTHSBSIR

define_measure \
    -name      MCS_GWD_CLK_DIF_RR1_sig2 \
    -trig      {clk} \
```

```
    -trig_dir  rise \
    -trig_val  0.45 \
    -trig_d    "" \
    -targ      "Xg/Xgcolr/Xgc<3>/phi1wx" \
    -targ_type "delay" \
    -targ_dir  fall \
    -targ_val  0.72 \
    -targ_d    "" \
    -failed_val "" \
    G40SP16384X4R3F1VTHSBSIR

define_measure \
    -name      MCS_GWD_CLK_DIF_RR1 \
    -keep      max \
    -duration  0.25 \
    -trig      {clk} \
    -trig_dir  rise \
    -trig_val  0.45 \
    -equations { \
        "MCS_GWD_CLK_DIF_RR1_sig1 - MCS_GWD_CLK_DIF_RR1_sig2" \
        "MCS_GWD_CLK_DIF_RR1_sig1 * 0.9 - MCS_GWD_CLK_DIF_RR1_sig2 * 1.1" \
        "100 * (MCS_GWD_CLK_DIF_RR1_sig1 - MCS_GWD_CLK_DIF_RR1_sig2) /
        (MCS_GWD_CLK_DIF_RR1_sig1 + MCS_GWD_CLK_DIF_RR1_sig2) " \
        "100 * (MCS_GWD_CLK_DIF_RR1_sig1 * 0.9 -
        MCS_GWD_CLK_DIF_RR1_sig2 * 1.1) / (MCS_GWD_CLK_DIF_RR1_sig1 +
        MCS_GWD_CLK_DIF_RR1_sig2) " \
        } \
    G40SP16384X4R3F1VTHSBSIR

define_arc -pin {clk} -measure MCS_GWD_CLK_DIF_RR1 -attribute
    {altos_px_existence timing.tbl::functional::write00} \
    G40SP16384X4R3F1VTHSBSIR
```

Example 4

This will characterize output power for combinational delays:

```
define_arc \
    -when {DFTRAMP & X10} \
    -type power \
    -related_pin_dir R  \
```

```
-pin_dir R  \
-related_pin {A[4:0]} \
-pin {AY[4:0]} \
G40SP16384X4R3F1VTHSBSIR
```

define_cell

Defines how an AMS block is to be characterized.

Options

`-async {`*`pin_names`*`}` List of asynchronous pin names. The `-async` option specifies pins that require recovery, removal, non_seq_setup, or non_seq_hold timing arcs**.**

`-clock {`*`pin_names`*`}` List of clock pin names. For more information, see Using the -input, -output, and -clock options.

`-constraint <`*`name`*`>` Name of template for constraint tables. This option enables characterization of timing constraints (setup, hold). The range of input slews to use for the data and clock signals is defined by the name of the template pre-defined using the define_template command.

`-delay <`*`name`*`>` Name of template for delay tables. This option enables characterization of cell delay and output slew for the non-linear delay model (NLDM). The range of input slews and output loads to use for a construct is defined by the name of the template pre-defined using the define_template command.

`-ignore_input_for_auto_cap {`*`pin_names`*`}`

Ignore arcs originating from the specified pin when calculating `auto_index` and `auto_max_capacitance`.

`-ignore_output_for_auto_cap {`*`pin_names`*`}`

Ignore arcs to the specified output pin when calculating `auto_index` and `auto_max_capacitance`.

`-ignore_pin_for_ccsn {`*`pin_names`*`}`

List of input pin names not to have CCSN data. This option accepts a list of input/bidi pins. No CCSN data is characterized for the specified pins.

`-input {`*`pin_names`*`}` List of input pin names. For more information, see Using the -input, -output, and -clock options.

`-internal {`*`node_names`*`}`

List of internal nodes in the library. Use this option to expose the timing characteristics of a physical node of interest, which is internal to the macro, without needing to change the macro interface. It also helps to split input to output timing arcs into separate segments. For more information, see Using the -internal option.

Note: The `-internal` option can be used with the `define_cell` command only in Liberate AMS.

`-internal_clock <`*`list_of_pairs`*`>`

Tcl flat list of pairs specified in the following format to define mappings between internal and external clock nodes:

`{ <`*`internal_clock_node`*`> <`*`correpsonding_external_clock_port`*`> ...}`

For more information, see Using the -clock and -internal clock options.

`-output {`*`pin_names`*`}` List of output pin names. For more information, see Using the -input, -output, and -clock options.

`-power <`*`name`*`>` Name of template for power tables. This option enables characterization of switching power. The range of input slews and output loads to use for a construct is defined by the name of the template pre-defined using the define_template command.

`{`*`cell_names`*`}` List of cell names to be characterized.

Using the -input, -output, and -clock options

The `-input`, `-output`, and `-clock` options define the pin type for the given list of pin names. All pins of a cell must have a defined pin type. The same pin name cannot appear in multiple pin types within a single `define_cell` command.

The input/output bundles – buses – can be specified in a compressed form using any of the following delimiters: `| |`, `<>`, `[]`, `{}`, `()`, or _ enclosing a range specification in the form `N:M`, where `N` and `M` are integers. In Liberate AMS, use the following *bus* notation for a pin:

`<`*`name`*`><`*`left_delimiter`*`>msb:lsb<?<`*`right_delimiter`*`>>`

where,

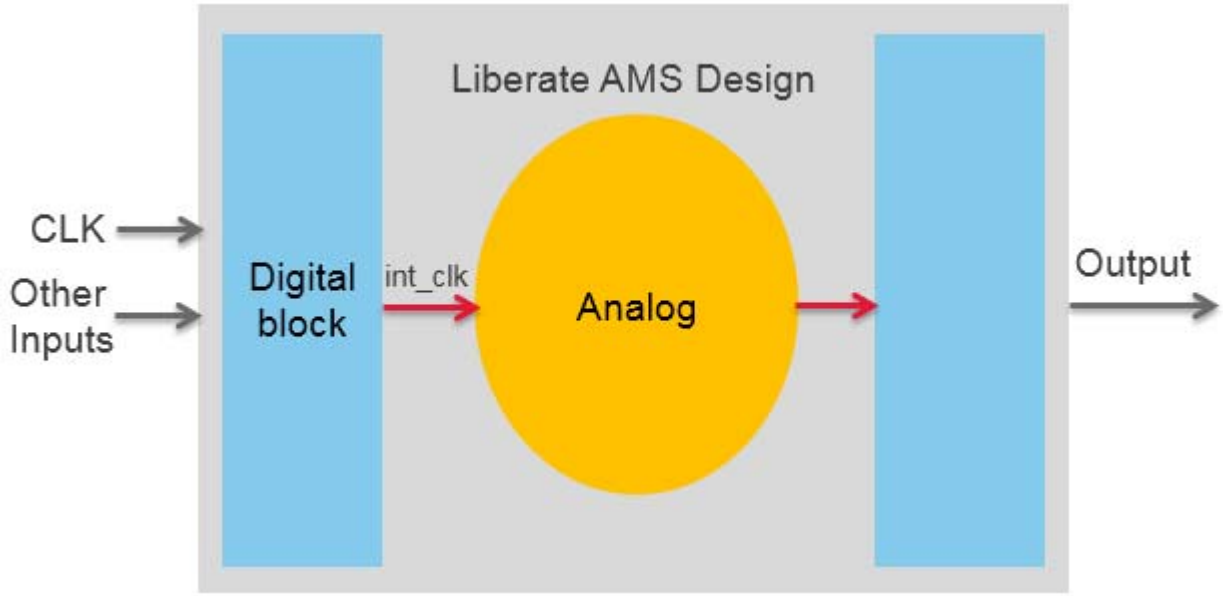
- `name` is the name of the bus

- `left_delimiter` indicates what to use as left delimiter when expanding the bus
- `msb:lsb` indicates the bit range the expansion must be done on
- `right_delimiter` indicates what to use as right delimiter when expanding the bus. The supported delimiters are: `[]`, `<>`, `{}`, `_`, and `()`. When the character `|` is specified, no delimiter is used. For example:
  - `A[5:0]` expands into `A[5], A[4], ... , A[0]`
  - `B|1:0|` expands into `B1, B0`
  - `C_0:3` expands into `C_0, ... , C_3`

The remaining options define which template to use for characterizing each library construct. If a template is specified, the appropriate construct is characterized for the given set of cells. If a template is omitted, the construct is not characterized.

Using the -internal option

Liberate AMS supports arcs from external input to an internal node and internal node to an external output, as shown in the figure below:



With the `-internal` option, you can specify internal nodes that are of the following types:

- Real nodes

A real node exists with the same name in both the netlist and the library. User-defined arcs that exist from and to such nodes are passed to Liberate AMS using the `write_template` command. Arcs are then characterized in Liberate AMS and reported to the generated library using the write_library command.

- Mapped nodes

  A mapped node exists in the netlist, but with a different name than the one used in the library. This is usually done for library legibility. User-defined arcs exist from and to such nodes are passed to Liberate AMS using the `write_template –map {mapped_node real_node}` command. Arcs are then characterized in Liberate AMS and reported to the generated library using the `write_library –map {real_node mapped_node}` command.

- Fake nodes

  A fake node does not correspond to any physical node in the netlist and is only used as a placeholder in the reference library. In this case, Liberate AMS attempts to combine the arcs ending in, and originating from, the internal node into complete input to output arcs, and characterizes those. Consequently, there are no internal node in the final generated library.

Using the -clock and -internal_clock options

Liberate AMS allows you to define mappings between internal and external clock nodes. To do this, use the `-clock` and `-internal_clock` options with the `define_cell` command. The list specified with the `-internal_clock` option provides this mapping in the following format:

```
{ <internal_clock_node> <correpsonding_external_clock_port> ...}
```

For example,

```
-clock { analog_clk1  analog_clk2   dig_clk1  dig_clk2  dig_clk3 }
-internal_clock { I_clk1_node  analog_clk1   I_clk2_node  analog_clk2  I_clk3_node  analog_clk1 } # note that this definition is a list of pairs
```

The above command defines the following:

- 5 external clocks (as in the `-clock` list)
- 3 internal clocks (where two are mapped to `analog_clk1` and one is mapped to `analog_clk2`)

Mapping from internal to external clock must be specified for each internal clock. Multiple internal clocks can be mapped to the same external clock, such as, a single clock that is divided into various different digital clocks with different frequencies. You can see this in the example given above. For cases where there are multiple external clocks mapping to a

digital clock (like for complementary clocks), just provide one of them because their relevant complementary information would be in the arc definition (like using the `-dual_related` option with the define_arc command).

Note: The corresponding external clocks must be defined in the `-clock` list in `define_cell`, otherwise the command will be rejected.

When internal clocks are specified, any constraint arcs with these clocks as a related_pin result in sequential setup/hold arcs. For modeling, these pins are handled as regular clocks and the `"clock:true"` attribute is written in the Liberty model for the internal clock pins.

define_deck

Drives AMS characterization using a simulation testbench.

Options

`-arctypes <timing | measure>`

Specifies the arc type. All arcs of the specified type should get updated by dynamic information coming from the deck. Default: `timing`

When this option is set to `timing`, the following arcs are updated: `delay`, `retain`, `setup`, `hold`, `enable`, `disable`, `mpw`, `min_clock_tree_path`, `max_clock_tree_path`, minimum period, and generic measure.

When this option is set to `measure`, any custom measurement arcs get updated.

{`-dut_inst <`*`string`*`>`} Specifies the name of the design under test (DUT) instantiation in the deck. Use this **mandatory** option when the top-level deck signals do not match the DUT ports. By default, this option is set to `none`, which means it is assumed that the top-level deck signals and the DUT ports match.

For example:

In deck:

```
X123 a a a b c sub123
```

In netlist:

```
.subckt a0 a1 a2 b c
...
.ends
```

`-help` Displays a list of the options for this command. All other command options are ignored.

`-include` Specifies the full path to a SPICE `include` file that should be used for characterization along with the specified deck.

`-isrcs {`*`list`*`}` Specifies a list of inputs with current sources. Default: `none`.

`-observation_window {`*`list`*`}`

Specifies the duration, in seconds, of the observation window used to infer causality between a related pin and a pin for an arc. Specify multiple values when different arcs require different observation windows for characterization. For example, for a clock at `100MHz`, specifying an observation window of `2.5ns` (`1/4*100e6`) would allow to correctly observe delay arcs from that clock.

`-shift_clocks <`*`boolflag`*`>`

Shifts the clock waveforms by the observation window. Default: `false`, that is, the clock waveforms are unchanged.

`-start_measuring_after`

Specifies a delay in the time when measurements get evaluated for a specific deck.

`-table <`*`string`*`>` Specifies the full path to a file containing commands in the table syntax. This file is included in the final table generated by the `define_deck` command.

`-use_charsim <`*`boolflag`*`>`

Specifies to use characterization simulator and options (specified with the `extsim_cmd` and `extsim_cmd_option` variables) for the deck to table command. Default: `false` (Use whatever is specified as partition simulator and values specified with the `fastsim_cmd` and `fastsim_cmd_option` variables.)

`<`*`deck_name`*`>` (Required positional option) Specifies the full path to user deck to be used for characterization.

`<`*`cell_name`*`>` (Required positional option) Specifies the cell name of the AMS block to be characterized.

Note: The `-ref_freq` and `-ref` options of the `define_deck` command have been deprecated. Use the `-observation_window` option instead.

This command is used to specify a simulation deck to be used to exercise the arcs to be characterized by Liberate AMS. Multiple decks can be specified via multiple `define_deck` commands.

Liberate AMS generates a library view of the digital arcs in a mixed signal block. The concepts of libraries, arcs, and characterization may not be most familiar to an AMS designer; at the

same time, knowledge of the design under test, and how to simulate it, may not be readily available to the person in charge of characterization. A simulation deck can be used as a mean to transfer knowledge from the designer to the person or group in charge of generating the timing, noise and power model.

Liberate AMS uses the deck to gain knowledge on how to generate input stimuli for the characterization. In order to extract measurement that can be used toward the specific arcs that need to be characterized, Liberate AMS automatically fits the input stimuli in to a simulation grid. The grid is usually determined by the observation window specified in seconds using the `-observation_window` option. When different arcs require different observation windows for characterization, you can specify multiple values for this option.

By default, the `define_deck` command transfers the input stimuli from deck to characterization without making any changes. To make any changes to it, shift all clocks by half a reference period using the `-shift_clocks` option. This is usually done when the reference signals are primary clocks.

If sources of current have been specified in the deck, such information should be transferred to LIberate AMS using the `-isrcs` option.

Example

```
define_deck \
    -observation_window 125e+06 \
    ${spicedir}/deck.sp \
    $cell
```

Example 1

Use deck `deck.sp` to drive characterization for arcs between an internal digital clock and constrained digital inputs. The internal digital clock is a divided down (by 8) version of analog clock inputs at `1GHz`.

```
define_deck -observation_window 125e+06 deck.sp $cell
```

Example 2

Use deck `deck.sp` to drive characterization for arcs using an input digital clock, `CLKP`, at `1GHz`. The deck instantiates current reference sources on input `IREF` and `IQREF`.

```
define_deck -observation_window 1e+09 -shift_clocks  -isrcs {IREF IQREF} deck.sp
     $cell
```

Frequency is inferred from `CLKP` waveform analysis; `IREF`, `IQREF` are transferred to the characterization.

Deprecated Options of define_deck

Only the options covered below have been deprecated. Use the `-observation_window {list}` option instead. For detailed information about this command and other options that it supports, see define_deck.

`-ref_freq <list>`	Specifies a list of reference frequencies. This allows you to specify multiple observation grids for a given deck. For example, for a serializer or a deserializer, you can specify the frequency of the divided internal clock.
`-ref <string>`	Specifies the reference signal. This signal is an input to periodic waveform from which a reference frequency is inferred for the design under test. For example, for a PLL, the reference input clock

Liberate AMS uses the deck to gain knowledge on how to generate input stimuli for the characterization. In order to extract measurement that can be used toward the specific arcs that need to be characterized, Liberate AMS automatically fits the input stimuli in to a simulation grid. The grid is usually determined by a reference frequency specified using the `-ref_freq` option or, alternatively, by inferring the frequency of a reference signal specified using the `-ref` option. Reference signal is usually a clock used as related event in the characterization arcs. If such signal is a primary input to the clock, either of `-ref` or `-ref_freq` can be used. If instead the signal is internal to the block, the `-ref_freq` option should be used.

define_duplicate_pins

Specifies a list of pins for a cell that will not be characterized directly, instead duplicate data will be passed to it from a characterized pin.

Options

<cellname>	The cell name to apply the duplication to.
<pin>	The pin to be duplicated.
{*duplicate_pins*}	List of pin names to be duplicated.

When you use this command, all the pin data gets copied from the `<pin>` to each of the duplicated pins including any data where the `<pin>` is a related_pin. If the pin is an input pin, the duplicate input pin includes pin capacitance, hidden power, and constraints. In addition, all input to output arcs where the pin is a related_pin is duplicated for each `duplicate_pin`. For example:

```
define_duplicate_pin mux A { B C D E F G H I }
```

This command also supports duplicating a complete bus. For example:

```
define_duplicate_pins myCell addrA addrB
```

where, `addrA` and `addrB` are buses to which the following restrictions apply:

1. `addrA` and `addrB` must have the same "direction" (cannot duplicate an input to an output.)
2. `addrB` cannot be of a lower range than `addrA`, but `addrB` can be of a larger range than `addrA`. In this case, only the first `n` bits of `addrB` will be duplicates of `A` and the rest will remain unique. In other words, if `addrA` is `16` bits, and `addrB` is `8` bits, you can clone `addrA[7:0]` to `addB[7:0]`. However, if `addrA` is `8` bits, and `addrB` is `16` bits, you cannot clone anything into `addrB[15:8]` from `addrA`, because `addrA` does not contain that bit range.
3. `addrB` does not need to exist—it can be created on the fly when the `write_library` command is run.
4. Duplicating bundles are not supported.

define_index

Overrides the indices specified in the templates referenced by the define_cell command. The following are the valid table types: `constraint`, `delay`, `retain`, `mpw`, `power`, and `si_immunity`. The overrides apply only to the cells listed in the define_index command. See also Important Points to Note below.

Options

`-index_1 {`*`indices`*`}`	List indices to use as index_1.
`-index_2 {`*`indices`*`}`	List indices to use as index_2.

`-pin {`*`pins`*`}`	List of pin names (REQUIRED). This option accepts wildcards.
	The following examples illustrate the use of an asterisk "`*`" wildcard for pin names:
	`define_index -pin D*  # All pins beginning with "D"`
	`define_index -pin  *  # All pins`

`-related_pin {`*`pins`*`}`	List of related pin names. This option accepts wildcards. **Note:** `-related_pin` is required for most arcs, except for arcs that need only one pin, such as hidden power arcs and MPW or `min_period` arcs.
`-type {`*`data_types`*`}`	List of data types: `constraint`, `delay`, `mpw`, `power`, or `retain`.
`-when <"`*`function`*`">`	Logic state of side inputs.
`{`*`cell_names`*`}`	List of cell names. This option accepts wildcards.

Important Points to Note

- `-pin`, `-related_pin`, and at least one of `-index_1` or `-index_2` must be specified.
- The size of the `-index_1` and `-index_2` lists must be equal to the equivalent template type specified by define_cell.
- The `-when` option helps to define unique indexes by using the Liberty `when` syntax.
- Multiple `define_index` commands can be used to specify different overrides for different arcs for the same set of cells, or for different cells.
- The `define_index` command must follow the define_cell command and precede the char_ams command.

Examples:

```
define_template -type delay \
    -index_1 {0.16 0.5 1.6 } \
    -index_2 {0.04 0.08 0.2 } \
    delay_template

define_cell \
    -clock { clk_in } \
    -input { add_in<6:0> chip_en data_in<31:0> wr_in } \
    -output { data_out<31:0> } \
    -delay delay_template \
    rf_cell

### Here the load values getting overwritten by define_index below
define_index \
    -pin {data_out<31:0>} \
    -related_pin {clk_in} \
    -type delay \
    -index_2 {0.05 0.01 0.5 } \
    rf_cell
```

define_leafcell

Defines the level of hierarchy that resides at the bottom of a cell-level netlist. This allows Liberate AMS to correctly identify devices in the cell netlist even when the process model file cannot be parsed. This command can be used in combination with the `extsim_model_include` control variable to enable external simulation with the process models and the compiled netlist. It supports identification of mosfets, diodes, resistors, and capacitors.

Options

`-area <"string">`	Name of area diode parameter in the cell. Default: `"area"`
`-element`	Enables circuit element-based leafcells. The model name(s) must be specified.
`-length "string"`	Name of the length MOS parameter in the cell. Default: `"l"`
`-multiple "string"`	Name of the multiple MOS parameter in the cell. Default: `"m"`
`-nfin "parameter_name"`	Name of nfin parameter for FinFET devices. Default: `'NFIN`
`-pin_position {list of pin positions}`	The pin positions. (REQUIRED) There should be one number for each pin in the cell. The pins usually start from `0` and the `pin_positions` start from `0 <drain gate source [bulk(s)]> | <terminal_p terminal_n[bulks]>`.
`-pj <"string">`	Name of diode PJ parameter in the cell. Default: `"pj"`
`-scale <"value">`	The scale factor MOS parameter in the cell. Default: `"1.0"` **Note:** This scale factor is used only by the Liberate *Inside View* to determine device sizes, and is not applied to the device sizes in the simulation netlist.
`-type "string"`	Type of the cell. Specify one of the following supported values: `blackbox`, `nmos`, `pmos`, `diode`, `r`, or `c`.
`-width "string"`	Name of the width MOS parameter in the cell. Default: `"w"`
`{cell_names}`	List of leafcell names.

Important

This command must be used before the `read_spice` command.

Examples

- Mosfets

```
define_leafcell -type nmos -pin_position {0 1 2 3} nch
define_leafcell -type pmos -pin_position {0 1 2 3} pch
```

- Resistors

```
define_leafcell -type r -pin_position {0 1 2} rppoly
```

- Diodes

```
define_leafcell -type diode -pin_position {0 1} ndio
```

- Capacitors

```
define_leafcell -type c -pin_position {0 1} ccap
```

define_leakage

Defines the logic conditions to use for calculating leakage power for the given cell name.

Options

`-ignore_for_default`	Instructs that the specified leakage should not be considered in any calculations when computing default `leakage_power` groups or attributes. (See `set_default_group`)
	
	For example:
	
	`...`
	`read_ldb My.ldb.gz`
	`define_leakage -when "a*!b" -ignore_for_default MyCell`
	`write_library My.lib`
	
	Note: This option must be set in a `define_leakage` command before the write_library command.

`-when <"function">`	User-specified logic conditions using the Liberty `when` format syntax. (REQUIRED)
`-vector <"stimulus">`	Vector stimulus used to simulate the leakage, each bit can be one of `X`, `1`, or `0`. **Note:** This option allows for partial `when` conditions for leakage in the output library, while having secondary pins set to a mix of `0s` and `1s`.
`{cell_names}`	List of cell names.

Note: Enhanced leakage characterization is also achievable by the tool honoring the `-when` condition specified in the define_cell command.

Important

This command must be used before the `read_spice`, `char_ams`, and `write_library` commands.

Examples

```
# Define the leakage conditions of a ADC instance
define_leakage -when " !CEN" ADC128
define_leakage -when " CEN" ADC128
```

define_measure

Specifies a measurement in a form similar to that of the HSpice `.meas` command.

Options

`-duration <#_of_cycles>`

Specifies the duration, in number of cycles, for the measurement. The measurement interval starts at this trigger. Default: `1`

A cycle is equivalent to `2 * simulation_interval`.

`-equations <list>` Specifies a set of related equations to be evaluated for the measurement. Default: none

The equations consist of previous measurement names specified in SPICE syntax. Only two levels of nesting and redirection are supported.

Example:

```
define_measure -name A
define_measure -name B
define_measure -name C -equations "A+B" <- OK
define_measure -name D -equations A     <- OK
define_measure -name E -equations D     <- NOT SUPPORTED
```

You can specify one or more equations associated with a measurement. If you specify multiple equations, they are each evaluated independently.

`-failed_val <string>`

Specifies a value to be used for a measurement that fails to evaluate. Default: none

`-failed_val_reuse_factor <value>`

Specifies a multiplication factor to be applied to the `failed_val` value during the partitioning step. Default: `1`.

`-help` Outputs a list of the options for this command. All other command options are ignored.

`-keep <min | max>`

Specifies how to resolve the final value when the measurement evaluates in multiple simulation intervals. Default: `max`

`min` Specifies that the minimum value is used as the final value.

`max` Specifies that the maximum value is used as the final value.

`-name <`*`name`*`>` (Required) Specifies the name to be used for the measurement. Default: none.

Because the name can be used as an argument of other measurements equations, make sure that *`name`* does not include any of the following characters that might be interpreted as mathematical operators:

`. - + * / , ( ) { } > < % : ^`

`-targ {`*`signal_name`*`}`

(Required) Specifies one or two target signal names. Default: none

`-targ_d <`*`name`* `|` *`#_of_cycles`*`>`

Specifies the duration before the target is to be considered. Default: none

You can specify the delay either in terms of cycles or by providing the name of a different measurement.

`name` Specifies the name of a measurement.

`#_of_cycles` Specifies the number of cycle to wait before the target is to be considered.

`-targ_dir <rise | fall>`

(Required) Use this option to specify the target signal direction. Default: none

`rise` Specifies that the target signal direction is rising.

`fall` Specifies that the target signal direction is falling.

`-targ_e <first | last |` *`pos_integer_signal_#`*`>`

Use this option to specify the edge of the target signal that is to be considered. Default: `last`

`first` Specifies that the first edge of the target signal is to be considered.

`last`	Specifies that the last edge of the target signal is to be considered.
`pos_integer_signal_#`	Specifies the number of the edge of the target signal to be considered.

`-targ_s <`*`measure`* `|` *`#_in_sec`*`>`

Shifts the resulting target crossing time point. You can specify the value as either an absolute number (in seconds) or as the result of another measurement. Default: `0`

`measure`	Specifies the name of a measurement.
`#_in_sec`	Specifies the number of seconds to shift the resulting target crossing point.

`-targ_type <delay | voltage>`

Specifies the type of target measurement. Default: `delay`

`delay`	Specifies that a delay measurement is to be used.
`voltage`	Specifies that a voltage measurement is to be used.

`-targ_val <`*`voltage`*`>`

(Required) Use this option to specify the target voltage at which the target event is to be considered. Default: `-100` (The absolute value in volts).

`-targ_val_reuse_factor <`*`value`*`>`

Specifies a multiplication factor to be applied to the `-targ_val` value during the partitioning step. Default: `1`

`-trig {`*`signal_name`*`}`

(Required) Specifies the trigger signal name. Default: none

`-trig_d <`*`measure`* `|` *`#_of_cycles`*`>`

Specifies the duration after which the trigger is to be considered. Default: none

You can specify the delay either in terms of cycles or by providing the name of a different measurement.

`measure`	Specifies the name of a measurement.

#_of_cycles Specifies the number of cycles to wait before considering the trigger.

`-trig_dir <rise | fall>`

(Required) Specifies the trigger signal direction. Default: none

`rise` Specifies that the target signal direction is rising.

`fall` Specifies that the target signal direction is falling.

`-trig_e <first | last |` *`pos_integer_signal_# >`*

Specifies the edge of the target signal that is to be considered. Default: "`first`"

`first` Specifies that the first edge of the target signal is to be considered.

`last` Specifies that the last edge of the target signal is to be considered.

`pos_integer_signal_#` Specifies the number of the edge of the target signal to be considered.

`-trig_end {`*`signal_name`*`}`

Specifies the end trigger signal name (used when specifying a window for a voltage measurement).

The `-trig_start` and `-trig_end` options

Must be used together.

Cannot be used with `-trig` (if they are, `-trig` is ignored.)

The only supported measurement types (`-targ_type`) are the minimum and maximum voltage.

`-trig_end_d <`*`measure | #_of_cycles`*`>`

Specifies the duration after which the end trigger is to be considered. Default: `0`

`measure` Specifies the name of a measurement.

`#_of_cycles` Specifies the number of cycles to wait before considering the trigger.

`-trig_end_dir <rise | fall>`

(Required if `trig_end` is specified) Specifies the end trigger signal direction. Default: none

`rise`	Specifies that the target signal direction is rising.
`fall`	Specifies that the target signal direction is falling.

`-trig_end_e <first |last | # >`

Specifies the edge of the end trigger that is to be considered. Default: `first`

`first`	Specifies that the first edge of the end trigger signal is to be considered.
`last`	Specifies that the last edge of the end trigger signal is to be considered.
`#`	Specifies the number of the edge of the end trigger signal to be considered.

`-trig_end_s <`*`measure`* `|` *`#_in_sec`*`>`

Shifts the end trigger crossing time point. You can specify the value as either an absolute number (in seconds) or as the result of another measurement.

`measure`	Specifies the name of a measurement.
`#_in_sec`	Specifies the number of seconds to wait before considering the trigger.

`-trig_end_val <`*`voltage`*`>`

Specifies the voltage at which the end trigger is to be considered.

`-trig_end_val_reuse_factor <`*`value`*`>`

Specifies a multiplication factor to be applied to the value of `-trig_end_val` in the partitioning step. Default: `1`

`-trig_s <`*`measure`* `|` *`#_in_sec`*`>`

Shifts the resulting trigger time point. You can specify the value as either an absolute number (in seconds) or as the result of another measurement.

measure	Specifies the name of a measurement.
#_in_sec	Specifies the number of seconds to shift the resulting target time point.

`-trig_start {`*signal_name*`}`

Specifies the name of the start trigger signal (used with `trig_end` when specifying a window of observation for a voltage measurement).

The `-trig_start` and `-trig_end` options:

- must be used together.
- cannot be used with `-trig` (if they are, `-trig` is ignored.)

The only supported measurement types (`-targ_type`) are the minimum and maximum voltage.

`-trig_start_d <`*measure* `|` *#_of_cycles*`>`

Specifies the duration after which the start trigger is to be considered. Default: `0`

You can specify the delay either in terms of cycles or by providing the name of a different measurement.

measure	Specifies the name of a measurement.
#_of_cycles	Specifies the number of cycles to wait before considering the start trigger.

`-trig_start_dir <rise | fall>`

Specifies the start trigger signal direction. Default: none

`rise`	Specifies that the start target signal direction is rising.
`fall`	Specifies that the start target signal direction is falling.

`-trig_start_e <first | last | # >`

Use this option to specify the edge of the start trigger that is to be considered. Default: `first`

`first`	Specifies that the first edge of the start trigger signal is to be considered.

`last`	Specifies that the last edge of the start trigger signal is to be considered.
#	Specifies the number of the edge of the start trigger signal to be considered.

`-trig_start_s <`*measure* | *#_in_sec*`>`

Use this option to shift the start trigger time point. You can specify the value as either a number (in seconds) or as the result of a named measurement.

measure	Specifies the name of a measurement.
#_in_sec	Specifies the number of seconds to shift the resulting start trigger time point.

`-trig_start_val <`*voltage*`>`

Specifies the voltage at which the start trigger is to be considered. Default: `-100`

`-trig_start_val_reuse_factor <`*value*`>`

Specifies a multiplication factor to be applied to the value of `-trig_start_val` in the partitioning step. Default: `-1` (Not applied)

`-trig_val <`*voltage*`>`

(Required) Specifies the absolute value of the voltage (in volts) at which the triggering event is to be considered. Default: `-100`

`-trig_val_reuse_factor <`*value*`>`

Specifies a multiplication factor to be applied to the value of `-trig_val` in the partitioning step. Default: `1`

`-when <`*expression*`>`	Specifies a conditional expression to be associated with the arc. The expression uses the Liberty™ `when` syntax (corresponding to the Liberty `when` attribute) to define the logic conditions of the other pins of the cell to enable this arc. Default: none
`{cell_names}`	Specifies a list of cell names. Default: none

The `define_measure` command uses the same process of FastSPICE top-level evaluation, partitioning, and full SPICE characterization that is used for delay and constraint characterization. The results of the `define_measure` measurements, for both the FastSPICE and real SPICE simulations, are written to the files specified by the `px_measure_report` variable.

Each measurement defined by a `define_measure` command is evaluated in each simulation interval—as dictated by the MX or AMS table used for the measurement—and across as many intervals as specified by the `-duration` option. For a measure that evaluates more than one simulation interval, the maximum value is used, unless otherwise specified by the `-keep` option.

Typically, the `define_measure` command is used to specify one or more raw measurements, followed by a second-level measurement that uses the raw measurements in its specified equations.

Example

This example shows how the HSpice `.meas` statements shown in the comment statements are converted into equivalent `define_measure` commands. In this particular example, the measurements are such that the raw measurements need to be taken only in the first (`sig1` and `sigb`) or in the second (`sig2`) cycle, but the second-level measurement needs to be evaluated across 2 clock cycles.

```
set cell SRAM
#.param cyc = 20n
#.param tc1 = 85n
#.param tc2 = tc1+cyc
#.param tc3 = tc2+cyc
#.param tc4 = tc3+cyc
#.meas tran sig1 trig v(clk) val=0.45 td=tc4 rise=1
#+targ v(x1.Xr/Xrblk<0>/reset_phi:1) val=0.18 td=tc4 fall = 1

define_measure \
    -name sig1 \
    -trig {clk} \
    -trig_dir rise \
    - 0.45 \
    -targ "reset_clk:1" \
    -targ_dir fall \
    -targ_val 0.18 \
    $cell

#.meas tran sig2 trig v(clk) val=0.45 td='tc4+cyc' rise=1
#+targ v(buf:1) val=0.18 td='tc4+cyc' rise=1
define_measure \
    -name sig2 \
    -trig {clk} \
    -trig_dir rise \
    - 0.45 \
    -trig_d 1 \
    -targ "buf:1" \
    -targ_dir rise \
    -targ_val 0.18 \
    -targ_d 1 \
    $cell

#.meas tran sigb trig v(st:1) val=0.45 td=tc4 fall=1
#+targ v(buf:1) val=0.45 td=tc4 fall=1
#.meas tran sigb_final param='max(sigb,0)'
```

```
# sigb actually switches in both cycles; force it to
# take the first falling transition for the target
define_measure \
    -name sigb \
    -trig {st:1} \
    -trig_dir fall \
    - 0.45 \
    -targ "buf:1" \
    -targ_dir fall \
    -targ_val 0.45 \
    -targ_edge first \
    $cell
#.meas tran reset param='s
ig1-sigb_final-sig2'
define_measure \
    -name reset \
    -keep min \
    -duration 2 \
    -trig {clk} \
    -trig_dir rise \
    - 0.45 \
    -targ_d 1 \
    -equations { \
        "sig1 - sigb - sig2" \
        "(sig1 - sigb)*0.9 - sig2 *1.1" \
        "100 * (sig1 - sigb - sig2)/ (sig1 - sigb + sig2) " \
        "100 * ((sig1 - sigb)*0.9 - sig2 *1.1) /(sig1 - sigb + sig2) " \
        } \
    $cell
define_arc -pin {clk} -measure reset $cell
```

define_table

Specifies the vector table files. (See Specifying Input Stimuli.)

Options

`mxtables {table_files}`
: List of vector table files. (REQUIRED)

`cell <name>`
: Cell name. (REQUIRED)

Example

```
define_table delay.tbl myMemCell
```

define_template

Defines a template to be used for characterization.

Options

`-type {delay | power | ccs | ccsn_dc | constraint | ecsm | mpw}`	The type of template being defined. (REQUIRED) How each cell is to be characterized is defined by associating the defined template with the appropriate option of the define_cell command. Multiple `define_cell` commands can reference a single template. See also Uses of different template types.
`-index_1 {values}`	List values to be used as the first index. (REQUIRED)
`-index_2 {values}`	List values to be used as the second index.
`-index_3 {values}`	List values to be used as the third index.
`<template>`	Name of the template.

This command must be used before the `char_ams` command.

Note: Internally, `define_template` uses a fixed set of units (listed below). These cannot be changed, *however*, units can be changed when writing a library with the `set_units` command.

current	1mA (milliamps)
power	1nW (nano watts)
resistance	1kohm (kilohm)
time	1ns (nano seconds)
voltage	1V (volts)
capacitance	1pf (pico farad)

Uses of different template types

The `delay` template type is used for delay characterization using input slew and output load. It requires both `-index_1` and `-index_2` to be specified, where `-index_1` represents the range of input slews and `-index_2` represents the range of output loads.

The `power` template type is used for switching and hidden (internal) power characterization using input slew and output load. It requires both `-index_1` and `-index_2` to be specified, where `-index_1` represents the range of input slews and `-index_2` represents the range of output loads.

The `ccs` template type can be used for composite current source model (CCS) delay characterization. It requires `-index_1` to be specified where `-index_1` represents the range of the normalized voltage values to measure.

The `ccsn_dc` template type can be used for composite current source DC noise model characterization. It requires `-index_1` and `-index_2` to be specified where they represent a range of input/output voltages. If not specified, Liberate uses a range of `29` voltage points from `-Vdd` to `2*Vdd`. DC simulations are very fast, especially on a small CCB group of transistors extracted for noise stage simulations and therefore, do not use much CPU time compared to the transient CCS noise models. It usually is not necessary to change the size of these tables from the default `29x29`. It can be useful to optimize the size of the tables to avoid non-convergence errors at the extremes of the voltage range. The `ccsn_dc` template is global to all cells and is not included in the define_cell command.

The `constraint` template type can be used for timing constraint (setup, hold, removal, recovery) characterization. It requires both `-index_1` and `-index_2` to be specified, where `-index_1` represents the range of input slews of the data signal and `-index_2` represents the range of input slews of the reference signal (clock, reset etc.).

The `ecsm` template type can be used for effective current source model (ECSM) characterization. It requires `-index_1` to be specified, where `-index_1` represents the range of the normalized voltage values to measure.

The `mpw` template type is used when a two dimensional mpw table is required. You must provide both `-index_1` and `-index_2`. In addition, the `define_cell -mpw` option must reference the `mpw` template. By default, if you do not provide an `mpw` template, the `mpw` timing constraint is characterized as a single attribute. If the `mpw_table` variable is set to a `1`, Liberate outputs a one-dimensional minimum pulse width (MPW) table using the `define_cell` constraint `-index_1` list of values. Use the `define_template -type mpw` only when a two-dimensional MPW table is required.

Tip

All `-index_*` entries for all the library constructs should be monotonically increasing.

Examples

```
# Delay template for 3 input slews, 3 output loads
define_template -type delay \
    -index_1 {0.025 0.1 0.25} \
    -index_2 {0.0010 0.015 0.100} \
    delay_3x3
# Power Template for 3 input slews, 3 output loads
define_template -type power \
    -index_1 {0.025 0.1 0.25} \
    -index_2 {0.0010 0.015 0.100} \
    power_3x3

# Timing constraint template for 2 input slews
define_template -type constraint \
    -index_1 {0.025 0.25} \
    -index_2 {0.025 0.25} \
    constraint_2x2

# ECSM  template for 5 intervals
define_template -type ecsm \
    -index_1 {0.05 0.2 0.5 0.8 .95} \
    ecsm_5

# CCS Noise DC Curve with 11 input and 11 output voltages.
define_template -type ccsn_dc \
    -index_1 {-1.0 -0.5 -0.2 -0.1 0.0 \
        0.1 0.2 0.5 1.0 1.2 1.5} \
    -index_2 {-1.0 -0.5 -0.2 -0.1 0.0 \
        0.1 0.2 0.5 1.0 1.2 1.5} \
    ccsn_dc_template
```

px_set_domainprop

Lists the nodes to start, stop, enable, or force domain propagation, or a clock, as defined in the define_cell command.

Options

The required positional options for this command must be given in the order shown.

`{ { <node> <prop_control> <domain> }...}`

Specifies a list of lists.

<node> (Required positional option) A pin or internal wire node name.

<prop_control>

(Required positional option) Allowed arguments are `start`, `stop`, `enable`, and `force`.

- Use `start` to begin propagation for the specified domain at the specified node.
- Use `stop` to stop propagation for the specified domain at the specified node.
- Use `enable` to allow propagation through a gate that is controlled by the specified node.
- Use `force` to make the node a descendant of the given domain. When forcing a domain to a node using `force`, be sure that an appropriate domain is declared in the define_cell command. For example, if forcing a clock domain on a node, ensure that the domain node is present in the `-clock` list in the `define_cell` command.

<domain> (Required positional option) A primary input name.

`-help` Outputs a list of the options for this command. All other command options are ignored.

By default, domain propagation starts at primary input pins and continues through the logic until it reaches a node that does not propagate the domain further, either statically or dynamically.

See also px_domain_propagation.

Example

```
# For primary clock CLK: stop it at node A, restart it at node B,
# force it at node C. Anytime a CLK derived node meets with D,
# let it go through.
px_set_domainprop {\
    {A stop CLK} {B start CLK} \
    {C force CLK} {D enable CLK}\
}
```

set_false_path

Enables the support for user-specified false path for path tracing in Liberate AMS static mode characterization flow. This command ensures that the path trace ignores those paths that match the user-specified false path.

Note: This command supports use of wildcard characters.

Options

`-through <list>`
: Specifies a list of wire names. The `-through` option denotes that both directions should be taken into consideration during false path handling.

  Note: When multiple `-through` options are specified, the `-through` options mean AND. If multiple pins are specified in a `-through` option, the OR of the specified pins is considered.

`-rise_through <list>`
: Considers the rising edge for the given list of wire names.

`-fall_through <list>`
: Considers the falling edge for the given list of wire names.

Example

```
set_false_path -through {A} -through {B C}
```

Considers the paths that include A && (B || C) as false path.

set_gnd

Identifies the names of ground nodes.

Options

`-ignore_power`	Ignore the power contribution from this supply net. If this option is set, the contribution of the specified supply net will be ignored. This means that the current in this supply net will not be summed into any power measurement.
`-no_model`	Request to not include this supply in the output `.lib`.
`-virtual`	Treat the net as logic 0 for static analysis, but as a regular node for dynamic simulation. If this option is set, the node is treated as a logic 0 or logic 1 for the static analysis step, but as a regular node for dynamic simulation and characterization. No contribution to internal power or leakage will come from such nodes. See px_find_virtual_rails to instruct the tool on how to report design nodes that should be defined as virtual**.**
`-waveform <name>`	Provide a file (full path name) containing a text description of a waveform as a time/value pair list.
`<net_name>`	Name of ground supply net.
`<voltage>`	Voltage value (in Volts).

Note: Multiple `set_gnd` and **set_vdd** commands can be specified.

Liberate AMS automatically identifies the following net names (case insensitive) as ground supplies and sets them to zero volts: 0, GND, and VSS. Use the `set_gnd` command to set them to alternative values. It is not recommended to attempt to change the voltage of the ground net `0` because this is considered the reference ground.

At 45nm and below it is not uncommon to find power-gating structures used to generate internal rails. These virtual nodes can be considered as logic `0` or `1` when using static analysis techniques to identify clock trees, latches, and so on, but should be considered as regular nodes during dynamic simulation. The `-virtual` option can be used for such nodes. If the names for such nodes are not identifiable easily, for example in a fully RC extracted netlist, the px_find_virtual_rails parameter can be used. Example:

```
set_vdd -no_model vss1 1 0.9
```

```
set_vdd -type back_up -cells cell1 vdd2 0.9
```

In the example above, using the `-no_model` option ensures that `vdd1` will not appear in `cell1` because it is globally set as `no_model` and no local `vdd1` will be set to `cell1`.

set_vdd

Identifies the names of power nodes.

Options

`-ignore_power`	Ignore the power contribution from this supply net. If this option is set, the contribution of the specified supply net will be ignored. This means that the current in this supply net will not be summed into any power measurement.
`-no_model`	Request to not include this supply in the output `.lib`. See set_gnd for related information.
`-virtual`	Treat the net as logic 0 for static analysis, but as a regular node for dynamic simulation. If this option is set, the node is treated as a logic 0 or logic 1 for the static analysis step, but as a regular node for dynamic simulation and characterization. No contribution to internal power or leakage will come from such nodes. See px_find_virtual_rails to instruct the tool on how to report design nodes that should be defined as virtual**.**
`-waveform <name>`	Provide a file (full path name) containing a text description of a waveform as a time/value pair list.
`<net_name>`	Name of power supply net.
`<voltage>`	Voltage value (in Volts).

Note: Multiple `set_gnd` and `set_vdd` commands can be specified.

Liberate AMS automatically identifies the VDD and VCC net names (case-insensitive) as power supplies and sets them to the default voltage specified by the `set_operating_condition` command. Use the `set_vdd` command to set them to alternative values.

set_virtual

Specifies the virtual ground and power nodes.

Options

`-vdd`	Treat the net as logic 1 for static analysis.
`-gnd`	Treat the net as logic 0 for static analysis.
`<net_name>`	Name of virtual rails net.
`<voltage>`	Voltage value (in Volts).

Note: Use this command for virtual net definitions, but the selection of a virtual net is controlled by the px_virtual_rail_opposite_device_minimum_factor variable.

write_ldb

Creates a library database (LDB). The LDB can then be used in a later Liberate session for formatting the library data, for example, creating a datasheet or generating a Verilog file.

Options

`<filename>`	A library database file in LDB format.

Liberate automatically saves each cell as it is characterized to an LDB in the current directory named `altos.ldb.<#>`, where `#` is the process ID. You can use the `write_ldb` command to rename this file.

It is recommended that the `write_ldb` command is executed immediately after the `char_library` command and before any model creation commands such as `write_library`. This is highly recommended to ensure that a clean, unmodified copy of the LDB is saved for future use. This is important because, for example, when user data is loaded with the `write_library -user_data` command, the internal database will be modified by the user data and any LDB saved subsequently will contain those modifications.

The LDB is automatically g'zipped if the gzip utility from GNU is in the search path. The library database is named as `<filename>.gz`. For example:

```
# characterize the library
char_library
# save the library database to tt.ldb
write_ldb tt.ldb
```

write_library

Outputs the library in Liberty format.

Options

`-bus_syntax {"<>" | "[ ]" | "( )"}`

Controls the `bus_syntax` used when outputting the library and the `bus_naming_style` attribute.

`-capacitance_only` Disables the output of `rise_capacitance` and `fall_capacitance` attributes. The output library will only have a single capacitance attribute.

Note: This option is useful for backward compatibility. Care should be taken when using this option.

`-capacitance_range {0 | 1 | 2}`

Controls whether the `rise/fall_capacitance_range` attributes should be output into the library. The supported values are:

- `0`: Omit
- `1`: Include the rise and fall range spanning from the minimum of the `min_capacitance` values to the maximum of the `max_capacitance` values. (Default)

  Note: This method has been reported to cause timing issues in PrimeTime.
- `2`: Include the rise and fall capacitance ranges where both range limits are both set to the rise/fall capacitance attribute values:

  ```
  rise_capacitance_range = "<rise_capacitance>,
  <rise_capacitance>"
  fall_capacitance_range = "<fall_capacitance>,
  <fall_capacitance>"
  ```

`-ccs` Include CCS data.

`-ccsn` Include CCSN (noise) data.

`-cells {`*`cell_names`*`}` List of cell names. Default: *all cells* This option supports the use of a wildcard.

`-dcnoise_abstol <`*`tolerance`*`>`

Controls the tolerance used to group similar DC templates while merging the DC noise templates.
Default: `1e-6` Volts (`1 uV`)

Note: If the noise values are less than the specified tolerance, the templates will be merged into a single group.

`-dcnoise_prefix <prefix>`

Controls the prefix that should be used when naming the templates while writing the DC noise templates.
Default: `"DC_"`

`-driver_waveform` Controls output of normalized driver waveform into the output library. For the output to include the driver waveform, the ldb or vdb must contain the driver waveform data. If the Tcl contains multiple `write_library` commands, the first command using this option enables the waveform output for all subsequent `write_library` commands. Normalized driver waveforms do not be output for user-defined PWLs that are specified incompletely or use wildcards.

`-driver_waveform_size <value>`

Sets the number of voltage points in the normalized driver waveform `index_2`.
Default: `500`

A normalized waveform currently uses an arbitrary number of voltage points uniformly distributed from gnd to vdd. The number of points can be controlled using this option.

`-ecsm` Include ECSM data.

`-ecsmn` Include ECSM noise data.

`-exclude` Exclude cells specified with the `-cells` option.

`-expand_buses` Turns off the creation of buses and instead outputs individual pins.

`-filename <filename>`

Output filename.

`-gzip` Compress the output library using the gzip utility.

`-indent <number>` Specifies the number of space to indent.
Default: `2`

`-map {`*`list`*`}`

List of name-map pairs, used to map internal names to an external name. Default: none (No name mapping.)

This option modifies the final name used for the internal node in the library. It is usually the case that the internal pin node name contains character that can not be used in a `.lib` format. Therefore, it is required to use a simpler name for such internal pins.

`-overwrite`

Overwrite the existing `.lib` file.

`-precision <`*`precision`*`>`

Controls the precision used when writing out the output values to the library. The value for this option must conform to standard Tcl formatting.
Default: `"%g"`

`-preserve_user_data_precision {`*`attributes`*`}`

Lists the attributes for Liberate to preserve the original precision in the `user_data` file. By default, Liberate uses the same precision as all other attributes.

`-remove_failed <none | data | cell>`

Instructs Liberate how to handle failed characterization data when writing the library. Default: `""` (this is equivalent to `data` for Liberate MX, Liberate AMS, Variety and `none` for other products.)

The following values are supported:

- `none`: Does not remove any failed data. See the variables `mark_failed_data`, `mark_failed_data_replacement`, and `constraint_failed_value` for the available user controls.
- `data`: Removes the arcs with failed data from the cell in the output library.
- `cell`: Removes the cell that includes any arc with failed data from the output library.

`-sdf_cond_equals {"==" | "===" | "== logical" | ""}`

Specifies how the `sdf_cond` attributes are written.
Default: `""` (that is, none)

The following table explains supported values:

Value	Output
"=="	"a == 1'b1 && b == 1'b0 …"
"==="	"a === 1'b1 && b === 1'b0 …"
"== logical"	"a == 1 && b == 0 …"
default	"a && ~b …"

`-sdf_edges` Enables output of the `sdf_edges` attribute.

The `sdf_cond_equals` argument

`-si` Include SI data.

`-skip {leakage | power | hidden_power | conditional_hidden_power}`

Specifies the data type for which the output of power arcs into the `.lib` file should be disabled.
Default: do not skip any data

This capability is useful when characterizing a library using different SPICE models for timing and power. The characterization of power cannot be skipped when CCS data is desired because Liberate needs the hidden power simulations to generate the receiver pin caps.

Specifying `hidden_power` skips the output of `hidden_power` arcs.

Specifying `conditional_hidden_power` skips the output of `conditional_hidden_power` arcs.

`-swap_index_order` Swaps the index order for 2d tables.
Default: Use the order specified by the one of the following commands: `read_library`, `define_template` or `read_ldb`.

`-user_data <filename>`

Specifies a user-provided library in Liberty format to be merged with the current library. This is useful to include non-characterized data such as wire-load models in the output library. Once this user-data is merged into the current library, all subsequent `write_library` commands output the merged constructs as part of the output library. If this is not desired, execute separate runs of Liberate consisting of `read_ldb` and `write_library`. Any valid construct present in the user-provided library that is not present in the current library database will be copied to the output library, with the following exceptions:

- Attribute `slew_derate_from_library` is not copied.
- Attributes `function`, `state_function` and `area` will override values in the current library.
- Groups `state_table`, flip-flops, and latch will override the equivalent groups in the current library.

`-user_data_override <filename>`

Specifies the `user_data` attributes that should be allowed to override the characterized values.

`<libname>` The output library name.

The `write_library` command outputs the library to the file specified by the `-filename` option. If filename is not specified, the library is written to `library_name.lib`. The `-gzip` option ensures that the output file gets compressed using the gzip utility. If the output library file already exists, a warning is issued and a unique filename is generated using the specified name suffixed with a unique number. The `-overwrite` option, if used, disables this automatic version control. Also, if the output library already exists, it will be overwritten. The `-cells` option controls which cells get written to the output library. If the `-exclude` option is also set, only the cells not listed in the `-cells` list will be output. By default, all cells get written.

The `-si`, `-ecsm`, `-ecsmn`, `-ccs`, `-ccsn`, and `-em` arguments enable inclusion of Liberty SI, ECSM, ECSM noise, CCS timing, and CCS noise data in the output library if it exists in the characterized database (LDB). By default, only leakage values and NLDM timing and power table data are written. For example:

```
read_ldb <ldb>
write_library -ecsm  <ecsm.lib>
```

Bus Support

`write_library` also supports buses. Buses can be defined using the `define_cell` command in the LDB, or by the `define_bus` command. For each defined bus, a bus template is created in the library header (group name `"type"`). A `bus_naming_style` attribute is also created. For each bus, all the timing, power, CCSN data, and so on for that bus pin is represented once under the bus group with only `capacitance` and `min/max_transition` attributes given for each pin. However, this can result in a loss in accuracy because the entire data is taken from the first bus bit.

You can use the `-expand_buses` option of the `write_library` command to output a library with individual pins and no buses. In addition, the `-bus_syntax` option can be used to change the bus syntax characters.

Bit-Level Delay Modeling

By default, for an arc involving a bus, only the *worst case* representative is chosen for characterization. The worst case values are then applied to all elements of the bus. This is controlled by adding the `-attribute altos_clone_arcs` option to the define_arc command. You can directly set these attributes or can indirectly control them through the syntax shown below.

- **Worst-case determined using all elements of a bus** (default)

  The worst case is determined from the full bus, whether it is specified as a full range of bits ***or*** as single bits.

  Full range:

  ```
  define_arc -type <...> -pin bus[MSB:0] ...
  ```

  Single bits:

  ```
  define_arc -type <...> -pin bus[MSB] ...
  define_arc -type <...> -pin bus[MSB-1] ...
  ...
  define_arc -type <...> -pin bus[0] ...
  ```

- **Worst-case determined within a range of bits**

  The worst case is calculated for the bits within the specified ranges, and applied to all bits in that range. In the example below, the worst case will be applied to bits in the range `R0:R1` separately from bits in range `R2:R3`.

  ```
  define_arc -type <...> -pin bus[R0:R1] ...
  define_arc -type <...> -pin bus[R2:R3] ...
  ```

- **No worst-case; each bit considered separately**

All bits of the bus are characterized separately (no worst-casing). This is achieved by specifying "single bit ranges", as shown below.

```
define_arc -type <...> -pin bus[MSB:MSB] ...
define_arc -type <...> -pin bus[1:1] ...
define_arc -type <...> -pin bus[0:0] ...
```

Note: When separate ranges are specified, only a fully-expanded (bit-blasted) library is able to properly represent the different values for the different bits (write_library -expand_buses). If you instead chose to generate a fully-compressed library (default in `write_library`) a worst-casing step will be done at the LDB level prior to generating the library. Therefore, if both an expanded and a bit-blasted library needs to be generated, the bit blasted should be generated first.

write_verilog

Creates a Verilog file for the current library. The Verilog content is written to the given <*verilog_filename*>. See also Additional Tcl Variables to Control Verilog Output.

Note: A `.v` suffix is added to filenames that do not end in `.v`.

Options

`-cells {`*cell_names*`}` Controls which cells should be written into the output file. Default: write all cells

Note: If the `-exclude` option is also set, only the cells not listed in the `-cells` list will be output. This option supports the use of a wildcard.

`-delayed` Controls the naming convention for creating "delayed" signals. Default: `"delayed_%P"` (where `%P` is the pin name.)

When using user-data with `write_verilog`, it is necessary to match these delayed signals with the equivalent signals used in the user-provided function description. By default, delayed output signals are created for the signals passed to timing checks such as setup-hold and/or `recrem` in Verilog.

For more details, see Using the -delayed option.

`-exclude` Exclude cells from -cells list.

`-fwire_prefix <"`*prefix*`">`

Prefix for internal wires for pin functions. Default: `"int_fwire_"`

`-indent <`*number*`>` Specifies the number of spaces to use for indentation. Default: use tab

`-merge` Includes the cell modules not specified in the library, but present in the `user_data` file.

`-mpw_include_output_state`

Includes output pin logic state in MPW timing checks for "clear" or "preset" input signals.

`mpw_include_output_state` requires that the `sdf_cond_style` variable is set to `1`. If not set, it will force the setting. This variable should be set prior to creating an equivalent library (`.lib`) with `write_library` to ensure consistency between the library and the Verilog file; otherwise, there might be warnings during SDF back-annotation.

The `-mpw_include_output_state` option should be used before the `read_library`, `char_ams`, or `read_ldb` command. When this variable is used, the MPW check (`$width`) is checked by Verilog only when the output pin of the cell is high for clear inputs and low for preset inputs. The Verilog file will contain extra "timing" gates to add the logic necessary to "and" the logic state of the output pin to logic representing the "when" condition given in the library for each `min_pulse_width` timing arc.

`-mux <`*`type`*`>`

Creates MUX user-defined primitives (UDPs) for MUX functions.
Default: use the basic logic primitives

Note: The `-mux` option converts the pins whose functions are a 2x1 or 4x1 MUX into a pre-defined UDP named `altos_mux2` and `altos_mux4` respectively.

`-no_edge`

Exclude 'posedge' or 'negedge' on edge triggered arcs, default is to include edges.

`-path <`*`path`*`>`

String used to denote a path, for example, `"=>"` and `"*>"`.
Default: `"=>"`

`-sdf_version <`*`version`*`>`

Controls the format of the output Verilog for use with SDF annotation. The version must be `2.1` or `3.0`.
Default: `3.0`

Set to `3.0` to generate a Verilog file that is compatible with SDF version 3.0. SDF version 3.0 permits `recrem` constructs in the Verilog file to represent recovery and removal of timing constraints.

`-specparams` Causes delay assignments in the Verilog file to be assigned to `specparam` variables rather than directly to values. The `-path` option controls the delimiter used for delay assignments, that is, `=>` or `*>`.

`-split_notifier` When writing verilog modules for multi-bit cells, it is required to output separate notifier commands for each DFF. The `-split_notifier` option is used to output separate notifier commands for each DFF.

`-timescale <timescale>`

Verilog timescale. (1ns/10ps)

`-twire_prefix <prefix>`

Prefix for internal wires of conditional timing constraint functions. Default: `int_twire_`

twire_prefix is the prefix used for internal wires created when generating additional functions for state dependent timing constraints. The `fwire_prefix` is the prefix used for internal wires created when generating logic functions. The `udp_prefix` is the prefix used for user defined primitives that are created for latches and/or flip-flops. Set `udp_prefix` to a null string to exclude generating user defined primitives.

`-udp_prefix <prefix>`

Prefix for built-in user defined primitives (UDPs). Set to "" to exclude UDPs.
Default: `altos_`

`-user_data <filename>`

User Verilog file to merge timing info with.

Specifies a user-provided Verilog file to merge the generated Verilog data with. If a `user_data` file is provided, timing information (paths and any additional wires required to specify the conditions for those paths) are merged with the user-provided Verilog file and written to the output file, replacing any existing user-provided timing information. Without a `user_data` file, a complete Verilog file is written including function descriptions.

`<verilog_filename>` Output Verilog filename.

The `write_verilog` command should not be used in the same run as the `char_ams`, `read_ldb`, or `write_library` commands. Instead, it should be used in a separate Liberate run after a `read_library` command. This is because the Liberty file might have proper formatting that is required for the Verilog output to be properly formatted. For example:

```
read_library my.lib
write_verilog my.v
```

The `write_verilog` command must be used after a database has been loaded. For example:

```
read_library my.lib
# Output a Verilog file
write_verilog -user_data my_verilog my.v
```

Using the -delayed option

The delayed option uses a special variable `%P` to return the pin name and combine it with a user-defined string. For example:

```
write_verilog -delayed "delayed_%P"
```

For a pin named `myPin`, the command above produces a delayed signal name `delayed_myPin`. Some examples are given below.

Example 1:

```
module DFFSRN (QN, D, CP, RN, SN);
    output QN;
    input D, CP, RN, SN;
    reg notifier;
    wire delayed_D, delayed_CP, delayed_RN, delayed_SN;

    // Function
    ....
    // Timing
    specify
        ...
        $setuphold (posedge CP, posedge D, 0, 0, notifier,,, delayed_CP, delayed_D);
        ...
        $recrem (posedge RN, posedge CP, 0, 0, notifier,,, delayed_RN, delayed_CP);
        ...
    endspecify
endmodule
```

Example 2:

```
write_verilog -delayed "dly_%P"
```

... will produce:

```
wire dly_D, dly_CP, dly_RN, dly_SN;
...
$setuphold (posedge CP, posedge D, 0, 0, notifier,,, dly_CP, dly_D);
```

Example 3:

```
write_verilog -delayed "%P_d"
```

... will produce:

```
wire D_d, CP_d, RN_d, SN_d;
...
$setuphold (posedge CP, posedge D, 0, 0, notifier,,, CP_d, D_d);
```

The default name for these delayed signals is "delayed_<*pin_name*>" (delayed_%P) where <*pin_name*> is a pin that is involved in a timing check.

To turn off generating delayed signals, use "" (empty double quotes). For example:

```
module DFFSRN (QN, D, CP, RN, SN);
    output QN;
    input D, CP, RN, SN;
    reg notifier;

    // Function
    ....
    // Timing
    specify
        ...
        $setuphold (posedge CP, posedge D, 0, 0, notifier);
        ...
        $recrem (posedge RN, posedge CP, 0, 0, notifier);
        ...
    endspecify
endmodule
```

Additional Tcl Variables to Control Verilog Output

The following Tcl variables can also be used to control the format of the Verilog output:

verilog_delay_value	The delay value. Default: 0

`verilog_delay_Zvalue` The delay value for tristates.
Default: `0`

`verilog_delay_clk2q_value`

The delay value for clock to Q arcs on sequential cells.
Default: `0`

`verilog_IQ` The name map for the internal state of flip-flops (such as, IQ) to the Verilog state function.

`verilog_IQN` The name map for the internal state of flip-flops (such as, IQN) to the Verilog state function.

`verilog_start_skip` Line to mark the start of the timing section in the user_data file which is to be replaced. Must match exactly apart for leading or trailing white space.
Default: `"specify"`

`verilog_stop_skip` Line to mark the end of the timing section in the user_data file which is to be replaced. Must match exactly apart for leading or trailing white space.
Default: `"endspecify"`

7

Liberate AMS Variables

This chapter lists the variables that can be used with Cadence® Virtuoso® Liberate AMS.

Available Variables

Table 7-1 on page 149 lists the variables that can be used with Liberate AMS.

- Variables that have a `Liberate` link are shared with other Liberate tools. Descriptions for these variables are located in Chapter 5 of the *Liberate Reference Manual*.
- Variables that have a `Liberate AMS` link can be used only with Liberate AMS. The descriptions for these variables are included on the indicated pages in later sections of this chapter.

Table 7-1 All Variables Available for Use With Liberate AMS

Variable	Link to Description
`adjust_tristate_load`	Liberate
`amstable_dontcare_value`	Liberate AMS: 165
`binning_detail`	Liberate
`bundle_when`	Liberate
`bus_syntax`	Liberate
`capacitance_force_hidden` (*deprecated*)	Liberate
`capacitance_pin_rollup_k`	Liberate
`capacitance_pin_rollup_mode`	Liberate
`capacitance_range_mode`	Liberate
`ccs_abs_tol`	Liberate
`ccs_base_curve_points`	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

ccs_base_curve_share_mode	Liberate
ccs_cap_hidden_pin	Liberate
ccs_cap_mode_add_missing	Liberate
ccs_cap_use_input_transition	Liberate
ccs_cap_use_input_transition_tristate	Liberate
ccs_force_grid_delay	Liberate
ccs_infer_output_dir	Liberate
ccs_init_voltage_comp_thresh	Liberate
ccs_max_current_thresh	Liberate
ccs_max_pts	Liberate
ccs_rel_tol	Liberate
ccs_segmentation_effort	Liberate
ccs_voltage_smooth_thresh	Liberate
ccs_voltage_tail_tol	Liberate
ccsn_active_ccr_recognition_mode	Liberate
ccsn_allow_duplicate_condition	Liberate
ccsn_allow_multiple_input_switching	Liberate
ccsn_allow_overlap_when	Liberate
ccsn_allow_partial_voltage_swing	Liberate
ccsn_arc_channel_check	Liberate
ccsn_arc_consistent_cut	Liberate
ccsn_arc_high_effort	Liberate
ccsn_bus_holder_mode	Liberate
ccsn_channel_inputs_high_effort	Liberate
ccsn_consistent_side_inputs	Liberate
ccsn_dc_template_size	Liberate
ccsn_default_group_add_when	Liberate
ccsn_default_group_criteria_mode	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

ccsn_dual_tie_enable	Liberate
ccsn_extra_default_stages	Liberate
ccsn_fanout_select_mode	Liberate
ccsn_io_mode	Liberate
ccsn_model_unbuffered_output	Liberate
ccsn_one_sided_tristate	Liberate
ccsn_pin_high_effort	Liberate
ccsn_pin_stage_merge_mode	Liberate
ccsn_pin_unconditional	Liberate
ccsn_prefer_two_sided_stages	Liberate
ccsn_prop_noise_peak_mode	Liberate
ccsn_prop_retry_duration_incr	Liberate
ccsn_prune_last_stage	Liberate
ccsn_xfr_ccc_probe_mode	Liberate
cleanup_tmpdir	Liberate
combinational_out_to_out_arc	Liberate
combinational_risefall	Liberate
conditional_arc	Liberate
conditional_constraint	Liberate
conditional_expression	Liberate
conditional_hidden_power	Liberate
conditional_immunity	Liberate
conditional_include_constant	Liberate
conditional_include_output	Liberate
conditional_leakage	Liberate
conditional_mpw	Liberate
constraint_async_probe_internal	Liberate
constraint_check_final_state	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

constraint_check_final_state_threshold	Liberate
constraint_check_rebound	Liberate
constraint_clock_gater	Liberate
constraint_combinational	Liberate
constraint_combinational_step_limit	Liberate
constraint_combinational_step_size	Liberate
constraint_delay_degrade	Liberate
constraint_delay_degrade_abstol	Liberate
constraint_delay_degrade_abstol_max	Liberate
constraint_delay_degrade_minimize_dtoq	Liberate
constraint_delay_degrade_minimize_dtoq_tol	Liberate
constraint_delay_min_check	Liberate
constraint_dependent_recrem	Liberate
constraint_dependent_setuphold	Liberate
constraint_dependent_setuphold_input_threshold	Liberate
constraint_dependent_setuphold_margin	Liberate
constraint_dependent_setuphold_margin_ratio	Liberate
constraint_dependent_setuphold_pessimism	Liberate
constraint_failed_value	Liberate
constraint_glitch_hold	Liberate
constraint_glitch_peak	Liberate AMS: 166
constraint_glitch_peak_internal	Liberate
constraint_glitch_peak_max	Liberate
constraint_glitch_peak_mode	Liberate
constraint_hold_probe	Liberate
constraint_info	Liberate
constraint_linear_waveform	Liberate
constraint_margin	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

constraint_merge_state	Liberate
constraint_output_load	Liberate
constraint_output_pin	Liberate
constraint_probe_internal	Liberate
constraint_probe_lower_fall	Liberate
constraint_probe_lower_rise	Liberate
constraint_probe_upper_fall	Liberate
constraint_probe_upper_rise	Liberate
constraint_search_bound	Liberate
constraint_search_bound_bisection_mode	Liberate
constraint_search_bound_estimation_mode	Liberate
constraint_search_bound_probe_mode	Liberate
constraint_search_time_abstol	Liberate
constraint_search_time_reltol_mode	Liberate
constraint_slew_degrade	Liberate
constraint_snap_to_bound	Liberate
constraint_vector_mode	Liberate
constraint_width_degrade	Liberate
debug_flow	Liberate
def_arc_drive_side_bidi	Liberate
def_arc_msg_level	Liberate
def_arc_vector_consistency_check	Liberate
default_capacitance	Liberate
default_group_method	Liberate
default_leakage	Liberate
default_non_unate_rcvr_cap_adjust	Liberate
default_power	Liberate
default_power_avg_mode	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

default_timing	Liberate
default_unateness	Liberate
define_arc_ignore_mode	Liberate
define_arc_preserve_when_string	Liberate
define_duplicate_cap_mode	Liberate
delay_constrained_by_setup_recovery	Liberate
delay_inp_fall	Liberate
delay_inp_rise	Liberate
delay_out_fall	Liberate
delay_out_rise	Liberate
disable_current_measure_effort	Liberate
disable_method	Liberate
disk_wait_time	Liberate
driver_cell_acc_mode	Liberate
driver_cell_info	Liberate
driver_cell_load_all_outputs	Liberate
driver_waveform_arcs_only	Liberate
driver_waveform_lib_mode	Liberate
driver_waveform_pulse_mode	Liberate
duplicate_pin_attr_mode	Liberate
duplicate_risefall_power	Liberate
ecsm_cap_hidden_pin	Liberate
ecsm_cap_input_slew_mode	Liberate
ecsm_cap_mode	Liberate
ecsm_cap_use_input_transition	Liberate
ecsm_measure_output_range	Liberate
ecsm_version	Liberate
extsim_ccs_option	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

`extsim_cells_use_nodeset_for_io_pad`	Liberate
`extsim_cmd`	Liberate
`extsim_cmd_option`	Liberate
Note: The Liberate AMS override to set this variable to `+spice` for static mode is not need.	
`extsim_deck_dir`	Liberate
`extsim_deck_header`	Liberate
`extsim_deck_include`	Liberate AMS: 166
`extsim_deck_style`	Liberate
`extsim_exclusive`	Liberate
`extsim_flatten_netlist`	Liberate
`extsim_immunity_option`	Liberate
`extsim_interactive`	Liberate
`extsim_leakage_option`	Liberate
`extsim_lic_keep`	Liberate
`extsim_line_length_limit`	Liberate
`extsim_model_include`	Liberate AMS: 166
`extsim_model_include_leakage`	Liberate
`extsim_model_include_mode`	Liberate
`extsim_model_include_multi_vector_mode`	Liberate
`extsim_monitor_deck_dir`	Liberate
`extsim_monitor_enable`	Liberate
`extsim_monitor_timeout`	Liberate
`extsim_option`	Liberate
`extsim_option_presim`	Liberate
`extsim_reuse_ic`	Liberate
`extsim_sanitize_param_name`	Liberate
`extsim_save_driver`	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

extsim_save_failed	Liberate
extsim_save_passed	Liberate
extsim_save_verify	Liberate
extsim_tar_cmd	Liberate
extsim_timestep	Liberate
extsim_tran_append	Liberate
extsim_use_node_name	Liberate
fastsim_cmd	Liberate AMS: 167
fastsim_cmd_option	Liberate AMS: 167
floating_node_consistency_check	Liberate
force_avg_default_select_order	Liberate
force_condition	Liberate
force_default_group	Liberate
force_edge_timing_type	Liberate
force_leakage_if_no_pg_pin	Liberate
force_related_power_pin	Liberate
force_unconnected_pg_pin	Liberate
heartbeat_initial_timeout	Liberate
heartbeat_timeout	Liberate
hidden_power	Liberate
immunity_glitch_peak	Liberate
immunity_noise_skew_ratio	Liberate
init_comb_num_cycles	Liberate
init_comb_related_pin_period	Liberate
init_constraint_period	Liberate
init_pin_hidden_num_cycles	Liberate
init_pin_hidden_period	Liberate
input_noise	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

keep_dcap_leakage	Liberate
keep_default_leakage_group	Liberate
keep_empty_cells	Liberate
keep_user_defined_arc_failed_data	Liberate
ldb_checkpoint_dir	Liberate
ldb_precision	Liberate
ldb_save_all_cells	Liberate
library_copyright	Liberate
library_revision	Liberate
library_revision_mode	Liberate
lic_max_timeout	Liberate
lic_queue_timeout	Liberate
logic_and	Liberate
logic_not	Liberate
logic_or	Liberate
mac_address_query_timeout	Liberate
mark_failed_data	Liberate
mark_failed_data_replacement	Liberate
max_capacitance_attr_limit	Liberate
max_capacitance_auto_mode	Liberate
max_capacitance_factor	Liberate
max_capacitance_limit	Liberate
max_leakage_vector	Liberate
max_noise_width	Liberate
max_transition	Liberate
max_transition_attr_limit	Liberate
max_transition_factor	Liberate
max_transition_for_outputs	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

max_transition_include_power	Liberate
measure_cap_lower_fall	Liberate
measure_cap_lower_rise	Liberate
measure_cap_upper_fall	Liberate
measure_cap_upper_rise	Liberate
measure_ccs_cap_lower_rise	Liberate
measure_ccs_cap_upper_fall	Liberate
measure_output_range	Liberate
measure_slew_lower_fall	Liberate
measure_slew_lower_rise	Liberate
measure_slew_upper_fall	Liberate
measure_slew_upper_rise	Liberate
measure_target_occurrence	Liberate
merge_related_preset_clear	Liberate
min_capacitance_for_outputs	Liberate
min_output_cap	Liberate
min_transition	Liberate
msg_level	Liberate
msg_level_user_data_override	Liberate
non_seq_copy_dst_pin	Liberate
non_seq_copy_src_pin	Liberate
non_seq_pin_swap	Liberate
non_seq_probe_mode	Liberate
nonseq_as_recrem	Liberate
output_internal_pin	Liberate
packet_arc_notification_interval	Liberate
packet_arc_notification_limit	Liberate
packet_arc_notification_list	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

packet_client_resubmit_count	Liberate
packet_client_timeout	Liberate
packet_clients	Liberate
packet_log_filename	Liberate
packet_rdb_mode	Liberate
parenthesize_not	Liberate
parenthesize_sdf_cond	Liberate
parse_space_bang_is_comment	Liberate
pin_based_leakage	Liberate
pin_based_power	Liberate
pin_capacitance_matching_mode	Liberate
pin_type_order	Liberate
pin_vdd_supply_style	Liberate
predriver_waveform	Liberate
predriver_waveform_ratio	Liberate
preserve_user_function	Liberate
prevector_period	Liberate
prevector_slew	Liberate
process_match_pins_to_ports	Liberate
px_active_coupling_threshold	Liberate AMS: 169
px_active_gate_load	Liberate AMS: 169
px_active_load_thresh	Liberate AMS: 170
px_active_wire_threshold	Liberate AMS: 170
px_arc_report	Liberate AMS: 170
px_autoprobing_hold_level	Liberate AMS: 171
px_autoprobing_setup_level	Liberate AMS: 172
px_bisection	Liberate AMS: 172
px_char_bundle_size	Liberate AMS: 173

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

px_mpw_probe_upper_rise	Liberate AMS: 191
px_negedge_clock	Liberate AMS: 191
px_partition_name_use_arc	Liberate AMS: 191
px_pathdelay_hold_clock_margin	Liberate AMS: 192
px_pathdelay_hold_data_margin	Liberate AMS: 192
px_pathdelay_setup_clock_margin	Liberate AMS: 192
px_pathdelay_setup_data_margin	Liberate AMS: 193
px_pincap_char	Liberate AMS: 193
px_posedge_clock	Liberate AMS: 193
px_power_assign	Liberate AMS: 193
px_power_single_point	Liberate AMS: 194
px_probes_report	Liberate AMS: 194
px_read_spice_exit_on_missing_file	Liberate AMS: 195
px_remove_rc_pincap	Liberate AMS: 195
px_remove_rc_timing	Liberate AMS: 196
px_retaining_time	Liberate AMS: 196
px_seq_probing	Liberate AMS: 197
px_setup_comb	Liberate AMS: 198
px_setup_seq	Liberate AMS: 199
px_simulation_interval	Liberate AMS: 199
px_spv_api	Liberate AMS: 200
px_transistor_variance	Liberate AMS: 200
px_transistor_vth	Liberate AMS: 201
px_verbose	Liberate AMS: 201
px_virtual_rail_auto_mode	Liberate AMS: 202
px_virtual_rail_opposite_device_minimum_factor	Liberate AMS: 202
px_virtual_rail_minimum_xtrs	Liberate AMS: 203
ramp_vsrc	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

rcp_cmd	Liberate
rdb_exit_if_source_differ	Liberate
removal_glitch_peak	Liberate
res_merge	Liberate
res_open_tol	Liberate
res_tol	Liberate
reset_leakage_current_mode	Liberate
reset_negative_constraint	Liberate
reset_negative_delay	Liberate
reset_negative_leakage_power	Liberate
reset_negative_power	Liberate
retry_count	Liberate
rsh_cmd	Liberate
scale_load_by_template	Liberate
scale_tran_by_template	Liberate
sdf_cond_equals	Liberate
sdf_cond_prefix	Liberate
sdf_cond_style	Liberate
sdf_logic_and	Liberate
sdf_logic_not	Liberate
sdf_logic_or	Liberate
server_timeout	Liberate
set_var_failure_action	Liberate AMS: 203
sim_duration	Liberate
sim_estimate_duration	Liberate
sim_init_condition	Liberate
sim_init_condition_estimation_mode	Liberate
sim_init_duration	Liberate

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

sim_power_duration_extend	Liberate
sim_use_init_duration	Liberate
simultaneous_switch	Liberate
simultaneous_switch_from_cell_when	Liberate
simultaneous_switch_offset	Liberate
simultaneous_switch_use_arc_when	Liberate
simultaneous_switch_worst_vector	Liberate
ski_enable	Liberate
slew_lower_fall	Liberate
slew_lower_rise	Liberate
slew_normalize	Liberate
slew_upper_fall	Liberate
slew_upper_rise	Liberate
sort_cells	Liberate
sort_groups_under_pin	Liberate
sort_pins	Liberate
sort_pins_under_when	Liberate
spectre_character_map	Liberate
spice_character_map	Liberate
spice_delimiter	Liberate
spice_delimiter_replacement	Liberate
spice_instance_name_require_x_prefix	Liberate
spice_logical_netname_mode	Liberate
static_autoprobing_max_depth	Liberate AMS: 203
static_autoprobing_max_states	Liberate AMS: 204
static_clock_tree_state	Liberate AMS: 204
static_measure_output_range	Liberate AMS: 204
static_path_skip_double_latches	Liberate AMS: 204

Table 7-1 All Variables Available for Use With Liberate AMS, *continued*

static_sat_mos_limit	Liberate AMS: 205
supply_define_mode	Liberate
test_cell_at_end	Liberate
timing_group_unateness	Liberate
tmpdir	Liberate
toggle_leakage_state	Liberate
tristate_disable_transition	Liberate
use_pid_tmpdir	Liberate
user_data_attr_order	Liberate
user_data_override	Liberate
user_data_quote_simple_attr	Liberate
vector_side_input	Liberate
verilog_cg_filter_edge	Liberate
voltage_map	Liberate
voltage_map_ldb_char_mode	Liberate
vsrc_slope_mode	Liberate
waveform_report	Liberate
wnflag	Liberate
write_logic_function	Liberate
write_logic_function_mode	Liberate
write_min_transition_attr	Liberate
write_template_force_power	Liberate

Variables for Only Liberate AMS

This section describes the variables that are for use only with Liberate AMS. These variables cannot be used when you run Liberate or Liberate MX. For information about additional variables that can be used with Liberate AMS and are also shared with other Liberate tools, see the links given in Table 7-1 on page 149.

amstable_dontcare_value

{ <*bus_name* | *pin_name*> <0 | 1 | *hex_number*> }

Sets a default table entry for a pin or bus. Default: `none`

bus_name	A bus to which the default table entry value applies.
pin_name	A pin to which the default table entry value applies.
`0`	Establishes a default table entry value of `0` for the specified bus or pin.
`1`	Establishes a default table entry value of `1` for the specified bus or pin.
hex_number	Establishes a default table entry value of the specified hex number for the specified bus or pin.

Important

The options for this variable must be given in the same order as given above.

Default table entries are specified as a list of name-value pairs. The value specified for the given pin or bus is used whenever there is no other value specified in the table. For example, if a pin is omitted from a table, or if there is a question mark (?) entry in a table, the default value is used.

Example

```
set_var amstable_dontcare_value {oe 0 ctrl 0xc}
```

constraint_glitch_peak

<value>	Specifies the maximum glitch height as a fraction of the supply used in characterizing timing constraints (setup, hold, recovery, removal). Default: `0.1` (10%)

Use this variable to specify the maximum height that a logic glitch on a constraint output pin is allowed to have. If the height is greater than this specified maximum, the arriving signal is deemed to fail a timing constraint.

The `set_constraint_criteria` command can also be used to set this variable. If both this variable and the `set_constraint_criteria` command are used, the last one executed sets the value to be used.

If you use the `constraint_glitch_peak` variable, it must be used before the `char_ams` command.

extsim_deck_include

`<0 | 1>`	Controls how the FastSPICE simulation deck is written out. Default: `1`

When set to `1`, only the original netlist is included via an `.inc` statement. To have the full netlist reported into the deck, set the `extsim_deck_include` variable to `0`.

For example:

```
#To include original netlist used by FastSPICE simulation.
set_var extsim_deck_include 0
```

extsim_model_include

<string>	Specifies the full path to a file that loads the SPICE models. Default: Uses the flattened models in the external simulation input decks.

Use this variable to specify the full path to a file that loads the SPICE models when using an external SPICE simulation engine. If you do not provide a full path, an error results. By default, Liberate AMS uses the flattened models in the external simulation input decks but when the `extsim_model_include` variable is used, Liberate AMS uses the specified file instead. Liberate AMS places a statement like the following in the external simulation SPICE decks:

```
.include <full_path_to_extsim_model_include_file>
```

Tip

To maximize speed and minimize memory requirements, Cadence recommends using both the `extsim_model_include` and `define_leafcell` variables for designs that are 40nm and below.

If you use the `extsim_model_include` variable, it must be used before the `char_ams` command.

Example

```
set_var extsim_model_include "/home/user1/models/include_ff"
set_var extsim_deck_include 1
```

Where `include_ff` is:

```
.include '/home/user1/models/models.l' ff
```

fastsim_cmd

`<path_to_executable>`

Specifies the full path to an executable to be used for simulating this table file.

fastsim_cmd_option

`<command_options>` Specifies the command line options to be passed to the executable used for simulating this table file.

packet_arc_notification_interval

`<value>` Specifies the minimum time interval between two informational notifications (see `packet_arc_notification_list`). The range of value is between 0 to 72000.
Default: `600` (in Seconds = 10 minutes)

This variable must be used before the `char_ams` command.

Example

```
set_var packet_arc_notification_interval 3600
```

The above example requests no more than one informational notification per hour.

packet_arc_notification_limit

<value>	Specifies the maximum number of informational notifications per run. This variable is effective when the `packet_arc_notification_list` has been set. The specified value should be between 0 to 100. Default: `10`

This variable must be used before the `char_ams` command.

Example

```
set_var packet_arc_notification_limit 5
```

The above example limits the notifications to no more than 5.

packet_arc_notification_list

<string>	Sets the e-mail addresses or SMS equivalent e-mail addresses that can receive notifications. Multiple e-mails or SMS numbers can be specified by using a comma-separated list. By default, no notifications are sent. You can set this variable to a valid e-mail address to enable notifications to that address. Default: `""` (empty list)

Requirements:

- The main Liberate AMS job must run on a machine that is able to send e-mails.
- Any SMS numbers provided for notifications should be able to receive messages by e-mail. Some carriers block this ability to prevent spam messages.

This variable must be used before the `char_ams` command.

Example

```
set_var packet_arc_notification_list "1111111111@mms.att.net,\
```

```
2222222222@messaging.sprintpcs.com,3333333333@tmomail.net,abc@def.com"
```

The above example has three SMS numbers (ATT/Sprints PCS/TMobile) and one e-mail address.

px_active_coupling_threshold

double — Sets the threshold coupling between victim and aggressor, which affects dynamic partitioning. Default: `-1`

For example, if the total coupling between aggressor A and victim V is greater than the threshold, then aggressor A is transversed during the dynamic transversal of victim V.

px_active_gate_load

"{ all | clock | none | wordline | {*net_name*} }"

Controls loading on a per-net basis. Default: `none`

`all`	Loads every node that has active devices.
`clock`	Loads clocks that have active devices.
`none`	Loads any (non-probe) node with equivalent passive loads.
`wordline`	Loads word lines that have active devices.
net_name	Loads the specified net, if it has active devices.

Note: This variable is useful only to achieve correlation on internal margin measurements and should not be used for regular library characterization.

Example

```
# Set active load on clocks, word lines, and signal "sig1"
set_var px_active_gate_load "clock wordline sig1"
```

px_active_load_thresh

"<*value*>"	Specifies the threshold for determining if the active load on a node can be replaced with a passive load. Default: `1.0` (farads)

Liberate AMS simplifies partitions by substituting a passive load for active devices. If the required passive load is larger than the threshold, the tool does not perform this substitution. Default:`1.0` farads

Example

```
set_var px_active_load_thresh "1e-15"
```

px_active_wire_threshold

double	Sets the threshold for a wire to be considered active. Default: `0.01` volts

px_arc_report

"<*filename*>"	Specifies the file where arc discrepancies are to be reported. Default: `px_dir/arc.rpt`

Use this variable to specify the file where arcs that are specified by you but are not found during partitioning are to be reported.

Liberate AMS uses the specified file to report the following discrepancies:

- User-specified arcs that are not found during partitioning.
- Arcs found during partitioning that fail during characterization.

See also px_check_arcs and px_check_arcs_exit_on_missing.

Example

```
# Report user-specified arcs that are not in the final library
set_var px_check_arcs 1
# File where arcs are reported
set_var px_arc_report [pwd]/arc.info
# Exit if one or more user-specified arcs does not generate a partition
set_var px_check_arcs_exit_on_missing 1
```

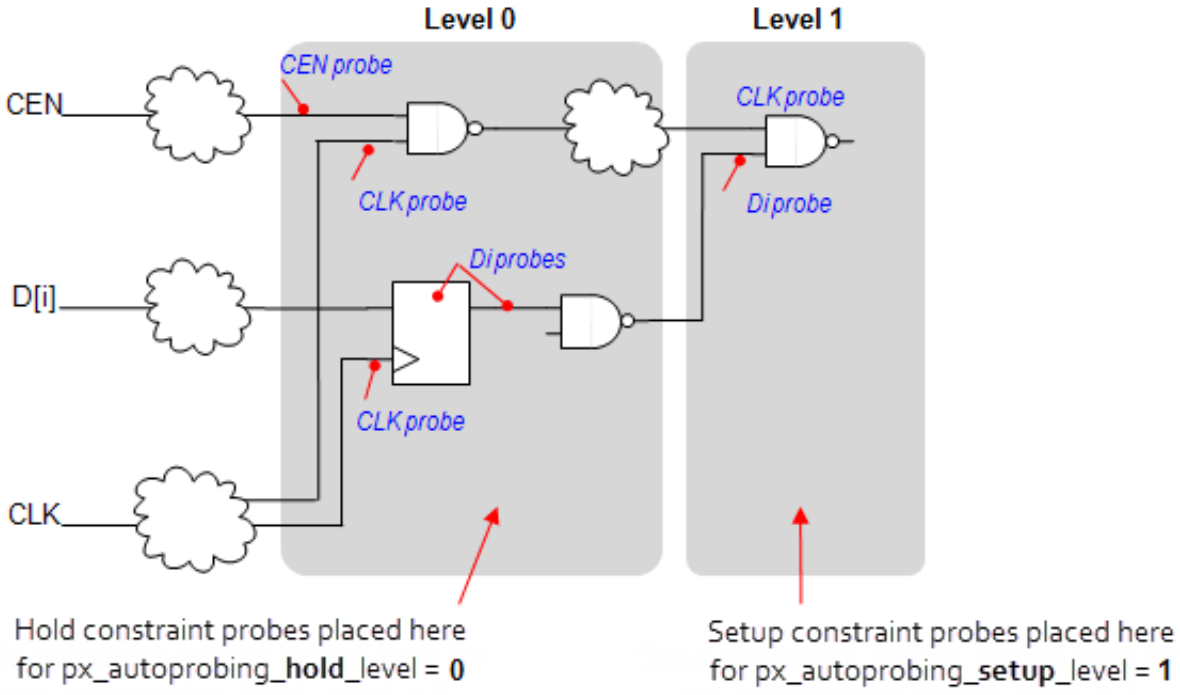
px_autoprobing_hold_level

<number>	Specifies where the *hold* constraint probing is inserted based on the number of levels of intersection between a pin and a related pin. Default: `0` (Only the first level of intersection of pin and related pin—data and clock, for example—lead to probe insertion.)

This variable allows for the selection of a specific logic region as a candidate to be probed based on the number of times a pin and related pin intersect on any path propagating from the inputs of that logic region. For this variable, a logic region is a channel connected region, such as a NAND gate, or a strongly coupled component, such as a latch, domino-stage, flip flop, or memory array. The level `0` is usually the input latch. See Figure 7-1 for an illustration of Level 0 and Level 1.

This variable is similar to px_autoprobing_setup_level, except that it applies to hold constraint.

Figure 7-1 Illustration of Level 0 and Level 1



Example

```
set_var px_autoprobing_hold_level 0;
```

px_autoprobing_setup_level

<number>	Specifies where setup constraint probing is inserted based on the number of levels of intersection between a pin and a related pin. Default: `1` (The first and second levels of intersection lead to probe insertion, except for flip-flop based designs, where the default is `0`.)

This variable is similar to px_autoprobing_hold_level, except that it applies to setup constraint. Level `0` is usually the input latch and level 1 is usually a later intersection with the clock. See Figure 7-1 for an illustration of Level 0 and Level 1.

Example

```
set_var px_autoprobing_setup_level 1;
```

px_bisection

`<0 | 1>`	Specifies whether Liberate AMS should use bisection for the full SPICE phase of characterization. Default: `0`	
	`0`	Liberate AMS uses the path-delay method for characterization.
	`1`	Liberate AMS uses bisection for characterization.

During the partitioning and automatic probing phase, Liberate AMS uses the path-delay differences method to identify worst-case constraints arcs. As for any other arc in the Liberate AMS flow, the resulting worst case arcs are partitioned and characterized in full SPICE. The full SPICE characterization phase uses either the bisection or the path-delay method, depending on the value of the `px_bisection` variable.

If you specify the use of bisection, the probes actually being measured may differ from the ones used when the path-delay method is used. When bisection is used, probes are usually placed at the outputs of slave stages, whereas when the path-delay method is used, probes are usually placed at the inputs of slave stages. You can control automatic probing in bisection mode by using the `-bis_probe` argument in the corresponding `define_arc` command.

Example

```
# Turns off bisection for all constraint arcs except user-defined arcs.
set_var px_bisection 0
```

Note: For correct bisection, Liberate AMS can accept only single switching on a pin in tables. If the table has multiple switching vectors as inputs, Liberate AMS characterization does not perform bisection on that arc.

px_char_bundle_size

<number>	Controls whether the characterization phase groups partitions into sets that are run sequentially. Default: `0`	
	`0`	Does not group partitions into separate runs.
	`1` or greater	Groups the specified number of partitions (or fewer) as a set, and then runs each set sequentially.

This variable instructs Liberate AMS to split the characterization phase into separate runs. Splitting the characterization phase can help avoid excessive memory usage during the `read_spice` process, a problem that is often caused when there are huge RC trees in the partitions. Accordingly, this variable is useful for heavily extracted netlists, particularly those with extracted power rails or extracted virtual rails.

Note: The grouping specified by this variable:

- Is independent from any load sharing or queuing specified for the characterization run.
- Is not related to the bundle or parallel bundle flow (which offers no advantage for runtime or memory usage in Liberate AMS).

Example

Given a timing characterization containing 201 timing partitions, 10 pin capacitances, and 1 power, and with

```
set_var px_char_bundle_size 20
```

Liberate AMS generates the following scripts automatically:

```
<cell>.timing.0.tcl# characterizes partitions 0 through 19
<cell>.timing.1.tcl# characterizes partitions 20 through 29
...
<cell>.timing.9.tcl# characterizes partitions 180 through 199
<cell>.timing.10.tcl# characterizes partition 200

<cell>.pincap.0.tcl
<cell>.power.0.tcl
```

px_char_virtual_as_rail

<all | *net_name*>	Controls the behavior of virtual nodes during dynamic partitioning and characterization. Default: `all`	
	`all`	Simplifies the dynamic partitioning and characterization of all the nodes that are defined as virtual by replacing their original logic with PWL models.
	net_name	Simplifies only the specified net.

The `set_vdd` command and this variable work together.

Use this variable to control whether nodes that are defined as virtual by the `set_vdd -virtual` option are simplified to piece-wise linear (PWL) models or retain their original logic.

Example

```
set_vdd -virtual VDD1 0.9
set_var px_char_virtual_as_rail VDD1
```

px_check_arcs

<0 | 1>	Controls whether a report is produced when there are discrepancies between the list of user-defined arcs and the list of arcs found during partitioning. Default: `0`	
	`0`	Does not report discrepancies.
	`1`	Reports arcs specified by the user that are not present in the final library.

Use this variable to compare the list of user-defined arcs to the list of arcs found during partitioning. Discrepancies can occur because arcs are not found by partitioning (which might be caused by errors in user-specified vectors, or by issues encountered during automatic probing or dynamic partitioning), or because of a failure during characterization (which might be caused by an issue in the deck, or by an incorrect simulation option.)

The variable px_arc_report specifies the file where arc discrepancies are reported. (Default: `px_dir/arc.rpt`). See also px_check_arcs_exit_on_missing.

Example

```
# Report user-specified arcs that are not in the final library
set_var px_check_arcs 1
# File where arcs are reported
set_var px_arc_report [pwd]/arc.info
# Exit if one or more user-specified arcs does not generate a partition
set_var px_check_arcs_exit_on_missing 1
```

px_check_arcs_exit_on_missing

`<0 | 1>` Specifies whether to end the run if there is a mismatch between the arcs specified by the user and the arcs found by partitioning. Default: `0`

- `0` Does not end the run if there is a mismatch.
- `1` Ends the run if there is a mismatch.

See also px_arc_report and px_check_arcs.

Example

```
# Report user-specified arcs that are not in the final library
set_var px_check_arcs 1
# File where arcs are reported
set_var px_arc_report [pwd]/arc.info
# Exit if one or more user-specified arcs do not generate a partition
set_var px_check_arcs_exit_on_missing 1
```

px_clock2clock_constraints

`<0 | 1>` Controls whether CLK to CLK constraint characterization is allowed. Default: `0`

- `0` Does not allow CLK to CLK constraint characterization.
- `1` Allows CLK to CLK constraint characterization.

px_clone_if_uda

`<0 | 1>` Control whether the characterized data for the worst case bit of a bus is duplicated for every arc involving the bus, or for only user-defined arcs involving the bus. Default: `1`

- `0` Worst-case arc data is copied to all arcs involving other bits of the bus.
- `1` Worst-case arc data is copied to only user-defined arcs involving other bits of the bus.

The `px_clone_if_uda` variable controls duplication (cloning) of characterized data in an arc involving a bus. During characterization, the worst case representative for the bus is chosen and fully characterized. According to the value of this variable, the worst-case data is then copied (cloned) over to all arcs involving other bits of the bus, or is copied only to user-defined arcs involving other bits of the bus.

Example

Assume that the tool identifies potential timing arcs between input bus `IN[0:3]` and output bus `OUT[0:3]` and that `IN[0]->OUT[0]` is the arc that gets characterized. Then,

- If `px_clone_if_uda` is set to 0, all possible arcs `IN[`*j*`]->OUT[`*j*`]` for *j*=`1,...,3` are assigned the characterized data found for `IN[0]->OUT[0]`.
- If `px_clone_if_uda` is set to 1, only user-defined arcs between `IN` and `OUT` are assigned the characterized data found for `IN[0]->OUT[0]`.

px_constraint_ocv_factor

`<`*value*`>` Specifies the factor by which the constraint measurements should be corrected. Default: `0`

For a hold, the maximum measured clock path delay is increased by this factor and the minimum measured data path delay is decreased by this factor. For a setup, the maximum measured data path delay is increased by this factor and the minimum measured clock path delay is decreased by this factor. The default of `0` provides no correction factor.

Example

Setting the variable like this,

```
# set 3% variation
set_var px_constraint_ocv_factor 0.03
```

increases the maximum path delay to 103% of the original value, and decreases the minimum path delay to 97% of the original value.

px_create_if_uda

`<0 | 1>` Controls how the existence or non-existence of a user-defined arc determines whether corresponding potential arcs identified during dynamic simulation are created and characterized. Default: `1`

- `0` A potential arc identified during dynamic simulation is created, and if partitionable, characterized even when there is no corresponding user-defined arc.
- `1` A potential arc identified during dynamic simulation is created and characterized only when there exists a corresponding user-defined arc.

To control directly what arcs are generated and characterized, you can use the `define_arc` command to specify arcs explicitly and set the `px_create_if_uda` variable to `1` so that only the arcs you explicitly define are characterized.

Example

Assume that:

- The tool dynamically observes that there could exist an arc from input `IN0` to output `OUT1`.
- There is no user-defined arc between `IN0` and `OUT1`.

With these assumptions:

- If `px_create_if_uda` is set to `1`, the potential arc is not created and not characterized.
- If `px_create_if_uda` is set to `0`, the arc is created and, if partitionable, characterized.

px_debug

`{ none | activity | all | arc | clock | dynamic | pattern | power | sim | static }`

Controls what debug messages are issued. You can specify any combination of these values. Default: `none`.

`activity`	Reports information on wire waveforms, as read back from FastSPICE.
`all`	Reports every type of debug information.
`arc`	Reports information on characterization arcs being identified during the partitioning process.
`clock`	Reports information on clock propagation.
`dynamic`	Reports information on dynamic partitioning.
`none`	Turns off debug messaging.
`pattern`	Reports information on AMS table expansion results.
`power`	Reports information about power characterization.
`probing`	Reports information on the probing step.
`sim`	Reports information on FastSPICE simulation decks and commands being generated during the run.
`static`	Reports information on topology. For example, reports information on connected channels, latches, and combinational circuits, and functional extraction.

px_delay_ocv_factor

<value>	Specifies a factor by which to the delay measurements should be corrected. Default: `0` (Provides no delay measurement correction.)

The maximum measured delay measurement is increased by this factor and the minimum measured delay measurement is decreased by this factor.

Example

Setting the variable like this,

```
# Set 3% variation.
set_var px_delay_ocv_factor 0.03
```

increases the maximum delay to 103% of the original value, and decreases the minimum path delay to 97% of the original value.

px_dir

<string>	Specifies the directory where Liberate AMS temporary files are stored. Default: `./ams`

Example

```
# Write ams files into the specified directory.
set_var px_dir /home/user/test_macro/ams
```

px_distributed_sim

`"{`*options*`}"`	Passes switches and options to an external program. Default: `""` (no options or switches are passed).

This variable passes switches and options to external programs such as a simulator or job management system. This allows you to run a program with the same options that you use standalone.

Examples

```
# Pass switch to SPICE.
set_var px_distributed_sim "spice +mt=8"

# Pass info to job queue.
set_var px_distributed_sim "bsub -q <queue_name> -o <log> -I spice"
```

px_domain_propagation

"<dynamic | static | static dynamic>"

Controls the domain propagation behavior.
Default: `static dynamic`

`dynamic` Uses dynamic information coming from the FastSPICE simulation to propagate the domain through the circuit. In other words, a domain on the input of a gate propagates to the output if they both switch during at least one common simulation interval.

`static` Uses static information to propagate the domain through the circuit, using these heuristics:

- Combinational logic: (a mix of clock and data domain among its inputs)
  - Clock propagates to the output
  - Data stops
- Sequential logic (a mix of clock and data domain among its inputs)
  - Clock stops
  - Data propagates through

`static dynamic`

Uses a combination of static and dynamic information to propagate the domain through the circuit. Heuristics used in the static propagation are augmented or corrected with dynamic information available from the FastSPICE simulation.

Use this variable to specify what kind of domain propagation should be used. A domain force or stop on a domain node overwrites the results of all these methods.

px_dpartition_inactive_node_voltage_resolution

<value> — Sets the continuous inactive values that are discretized using the specified resolution. Default: `0.005 volts.`

In Liberate AMS, when an inactive node is met during dynamic partitioning, the node is connected to a voltage source of value specified by the top-level simulation result. Using this variable, you can set the continuous inactive values.

px_failed_char_report

<string> — Specifies the name of the file that is used to report detailed information about any partitions that fail characterization. Default: `./failed_char_arcs.rpt`

px_fastsim_clock_slew

<double> — Specifies the clock slew to use for the FastSPICE simulation. When not specified, the first entry in the clock slew table is chosen as the slew. Default: `-1` (Default to the first entry in the table.)

Example

```
# Units are seconds.
set_var px_fastsim_clock_slew 1e-11
```

px_fastsim_input_slew

<double> — Specifies the input slew to use for the FastSPICE simulation. When not specified, the first entry in the input slew table is chosen as the slew. Default: `-1` (The first entry in the input slew table is used.)

Example

```
# Units are seconds.
set_var px_fastsim_input_slew 1e-11
```

px_fastsim_load

<double>	Specifies what load to use for the FastSPICE simulation. When not specified, the first entry of the output load table is chosen as the load. Default: `-1` (The first entry in the output load table is used.)

Example

```
# Unit is F.
set_var px_fastsim_load 5e-15
```

px_fastsim_reuse

`<0 | 1>`	Specifies whether to reuse the FastSPICE simulation results from a previous run. Default: `1`	
	`0`	Does not reuse simulation results.
	`1`	Reuses previous simulation results.

Liberate AMS stores the results of FastSPICE runs in the directory `./px_fastsim` under the filename *<macro_name>_<table_file_name>*`/transient1.tran.fsdb`. If the AMS tables do not change from one run to the next, Cadence recommends that you reuse previous run results by setting this option to `1`.

Note: Liberate AMS automatically reruns the FastSPICE simulator for the result files that cannot be found. See the table-based `fastsim_reuse` option also.

Reuse of FastSPICE Data

When you set the `px_fastsim_reuse` variable as shown below, Liberate AMS looks for FastSPICE simulation results in `$TMPDIR/ams_reuse/ams_fastsim` to reuse:

```
set_var px_fastsim_reuse 1
```

The reuse of the FastSPICE simulation results can speed up debugging of a Liberate AMS setup. However, it should be used carefully. The FastSPICE simulation results must not be reused in the following cases:

- The stimuli (tables) have changed.
- The netlist has changed.
- The FastSPICE options have changed.
- The simulation interval has changed.

Note: Liberate AMS does some consistency checks, but it is recommended that you remove the data manually if any of the changes mentioned above happen.

Liberate AMS performs the `checksum` generation program on the input testbench SPICE decks to check that the input SPICE simulation is ***not*** rerun if the decks have not changed. However, when you set the `px_fastsim_reuse` variable, it instructs the tool to automatically rerun the testbench simulation if the netlist or any parameters involved in generating the testbench table file(s) have changed, including observation windows, start measure times, and so on.

The `rechar_chksum` variable can be used to set the `checksum` generation program (defaults to `/usr/bin/md5sum`). If this program is not available on your system, which happens only in rare cases, use this parameter to point to a similar `checksum` generation program. Make sure that this program can accept data through an input pipe and it outputs a `checksum` string followed by a space or newline character.

Example

```
# No table has changed from previous run so reuse
# FastSPICE results for current run.
set_var px_fastsim_reuse 1
```

px_find_virtual_rails

`<0 | 1 | 2>` Specifies whether to identify and report power switched nodes that should be treated as logic constant during static analysis. Default: `0`

- `0` Disables the identification.
- `1` Enables the identification and reporting.
- `2` Enables auto-recognition of all the virtual rails that drive pmos/nmos number to be larger than px_virtual_rail_minimum_xtrs (default is `50`) and match the virtual rail topology. This value reports which wire is set as virtual rail and which other wires can be virtual rails [`ams/virtual.rpt`].

Set this variable to `1` when the static partitioning step takes an unusually long time, such as 10 minutes on a 10M transistors block. Use the `px_find_virtual_rails` variable as follows:

1. Set the variable and run to have the first set of virtual rail candidates reported.
2. Modify the Tcl script to include `set_vdd`, `set_gnd`, and `-virtual` commands on the reported nodes and rerun, checking whether static partitioning runtimes are as expected.
3. Repeat as needed; although usually one iteration is sufficient.

Example

```
# Tool seems to be hanging during static partitioning:
# (AMS-info) - Partitioning - static - start ...
#    (AMS-info) - Partitioning  24/1693326 xtrs
#

# Stop execution and add to your tcl script:
set_var px_find_virtual_rails 1
#

# Rerun; following is reported:
| (AMS-info) - Finding virtual rails ...   wall clock time +=    0 sec
|  | (AMS-info) - xtop/net112:1 - was not set as virtual rail. If the next
partitioning steps take longer than expected, check the node and add 'set_vdd
(set_gnd) -virtual xtop/net112:1 $vdd ($gnd)' to your script
#

# Stop execution and add to your tcl script:
set_vdd -virtual xtop/net112:1 0.99
#
# Rerun.
```

The following is an example of a virtual rail report [`ams/virtual.rpt`]:

```
*** wire_name                drive_pmos      drive_nmos      virtual_vdd
N_XI0-Q_3_47_c_1031883_n        2204              0            set (has been
set as virtual_vdd)

*** wire_name                drive_pmos      drive_nmos      virtual_gnd
N_XI0-VSS_c_697803_n            0               870               set (has been
set as virtual_gnd)

*** wire_name                drive_pmos      drive_nmos      other_unset_wires
N_XI0-DBL_c_1280305_n           10               68              unset (not set
as virtual rail)
```

px_greybox

`<0 | 1>`

Generates a library directly from the results of the partitioning simulator. Default: `0`

Note: This variable runs full instance simulations for all slews and loads.

`0`	Proceeds with the normal flow that includes a full SPICE run.
`1`	Generates a library directly from the partitioning results, without running a full SPICE simulation.

px_hold_comb

`{ < pin_name | all | none > }`

Drives the automatic probe identification step in identifying candidates on combinational logic for hold constraints probes. Default: `all`

`all`	Specifies that probing for hold characterization can be done on combinational logic for all input pins.
`pin_name`	Specifies an input pin for which probing for hold characterization can be done on combinational logic.
`none`	Specifies that combinational logic cannot be used for any pin for hold characterization.

Using the `:probe_inputs` modifier in conjunction with the `all` or *`pin_name`* options instructs the tool to widen the field of probe candidates to include the inputs of the combinational logic being probed.

For more information, see Appendix C, “Autoprobing in Liberate AMS.”

px_hold_seq

`{ < pin_name | all | none > }`

Drives the automatic probe identification step in identifying candidates on sequential logic for hold constraints probes.
Default: `all`

`all`	Specifies that probing for hold characterization can be done on sequential logic for all input pins.
`pin_name`	Specifies an input pin for which probing for hold characterization can be done on sequential logic.
`none`	Specifies that sequential logic cannot be used for any pin for hold characterization.

Using the `:probe_inputs` modifier in conjunction with the `all` or *`pin_name`* options instructs the tool to widen the field of probe candidates to include the inputs of the combinational logic being probed.

For more information, see Appendix C, “Autoprobing in Liberate AMS.”

px_hybrid_enable

`{ "dynamic" | "static" | "hybrid" }`

Selects the mode to use for characterization.
Default: `"dynamic"`

`"dynamic"`	Runs dynamic mode only.
`"static"`	Runs static mode only.
`"hybrid"`	First, runs dynamic mode and then, runs static mode for arcs that are not covered by a dynamic table or deck.

Liberate AMS hybrid mode is a combination of traditional table/deck-based characterization, and a new, experimental static timing characterization flow. Liberate AMS static mode partitions the circuit into Channel-Connected Components (CCCs), simulates each CCC while propagating simulated waveforms, and statically analyzes the circuit for worst-case

timing. You do not need to provide any simulation stimuli. The final values tend to be more conservative compared to those from the traditional dynamic flow.

This variable must be used before the `char_ams` command.

px_internal_pin_slews_use_internal

`<0 | 1>`

When an internal pin (a pin defined through the `-internal` option in the `define_cell` command) is used as related pin in an arc, there are two options for what values to use for its corresponding "index" template values. Use this variable to select the value to used. Default: `1`

- `0` — Use the primary (input) slews for the internal pin.
- `1` — Use the actual slews as measured from simulation for the internal pin.

Example

```
Delay template defined as:

    lu_table_template (delay_3x2) {
        variable_1 : input_net_transition;
        variable_2 : total_output_net_capacitance;
        index_1 ("0.01, 0.02, 0.03");
        index_2 ("0.004, 0.005");
    }
```

Timing group for delay `ARC INT_CLK->ckout`, when `px_internal_pin_slews_use_internal` is `1`:

```
pin (ckout) {
    direction : output;
    related_ground_pin : avss;
    related_power_pin : avdd;
    max_capacitance : 0.005;
    timing () {
        related_pin : "INT_CLK";
        timing_sense : negative_unate;
        timing_type : combinational;
```

```
        cell_rise (delay_3x2) {
            index_1 ("0.0170449, 0.017062, 0.017079");
            index_2 ("0.004, 0.005");
            values ( \
                "0.0592466, 0.0603861", \
                "0.0592606, 0.0603983", \
                "0.0592759, 0.0604104" \
    );
}
```

Timing group for delay `ARC INT_CLK->ckout`, when `px_internal_pin_slews_use_internal` is `0`:

```
pin (ckout) {
    direction : output;
    related_ground_pin : avss;
    related_power_pin : avdd;
    max_capacitance : 0.005;
    timing () {
        related_pin : ""INT_CLK"";
        timing_sense : negative_unate;
        timing_type : combinational;
        cell_rise (delay_3x2) {
            index_1 (""0.01, 0.02, 0.03"");
            index_2 (""0.004, 0.005"");
            values ( \
                ""0.0592466, 0.0603861"", \
                ""0.1456709, 0.1568979"", \
                ""0.2458760, 0.3126658"" \
    );
}
```

px_measure_report

`"<filename>"`	Specifies the name of the report where the actions of `define_measure` commands are reported. Default: `"measure.rpt"`

To generate this report, specify a filename (using a full path if desired). To omit this report, set this variable to `""` (empty quotes).

px_mpw_measurement_duration

<value> Sets the duration of the minimum pulse width (MPW) measurement as a fraction of the `px_simulation_interval` value. Default: `0.75` (assumes clock period of 1).

Usually the period is two times the simulation interval (`clk period == 2 * px_simulation_interval`), so a default of 0.75 ensures that the duration of the measurement covers a rising and a falling edge.

px_mpw_probe

`"<seq | comb>"`

Specifies what type of logic should be examined when looking for MPW probes. Default: `seq`

`seq` Examines sequential logic.

`comb` Examines combinational logic.

px_mpw_probe_lower_fall

<value> Specifies the lower fall threshold for MPW probe measurements. Default: `0.3`

px_mpw_probe_lower_rise

<value> Specifies the lower rise threshold for MPW probe measurements. Default: `0.3`

px_mpw_probe_upper_fall

`<value>` Specifies the upper fall threshold for MPW probe measurements. Default: `0.7`

px_mpw_probe_upper_rise

`<value>` Specifies the upper rise threshold for MPW probe measurements. Default: `0.7`

px_negedge_clock

`{ clock_name }` Specifies the negative edge active clocks, otherwise Liberate AMS assumes that clocks are positive edge active clocks. Default: none.

Example

```
# To specify clk1 & clk2 being negative edge active clocks.
set_var px_negedge_clock {clk1 clk2}
```

px_partition_name_use_arc

`<0 | 1>` Controls the naming of Liberate AMS partitions. Default: `1`

- `0` Generates names that do not include information about the arc that the partition represents.
- `1` Generates names that include information about the arc that the partition represents.

Examples

With `px_partition_name_use_arc` set to `0`,

- A constraint partition might be named:

  `pll0_constraint_553`

- A delay partition might be named:

  `pll0_delay_558`

With `px_partition_name_use_arc` set to `1`,

- The constraint partition would be named:
  `pll0_div0_hold_f_clkin_r_553`
- The delay partition would be named:
  `pll0_clckout_r_clkin_r_558`

because the specific arc the partition represents is incorporated into the partition name.

px_pathdelay_hold_clock_margin

<double>	Specifies the percent margin to add to the hold on clock paths. Default: `0`

Example

The following statement adds 5% to the hold on clock paths:

```
set_var px_pathdelay_hold_clock_margin .05
```

px_pathdelay_hold_data_margin

<double>	Specifies the percent margin to add to the hold on data paths. Default: `0`

Example

The following statement adds 5% to the hold on data paths:

```
set_var px_pathdelay_hold_data_margin .05
```

px_pathdelay_setup_clock_margin

<double>	The percent margin to add to setup on clock paths. Default: `0`

Example

The following statement adds 5% to the setup on clock paths:

```
set_var px_pathdelay_setup_clock_margin .05
```

px_pathdelay_setup_data_margin

<double>	The percent margin to add to setup on data paths. Default: 0

Example

The following statement adds 5% to the setup on data paths:

```
set_var px_pathdelay_setup_data_margin .05
```

px_pincap_char

<0 | 1>	Controls whether pin capacitance characterization is performed or skipped. Default: 1	
	0	Does not perform.
	1	Performs.

px_posedge_clock

{ *string* }	Specifies the names of positive edge active clocks. Default: all

This variable allows you to specify the names of positive edge active clocks. This is useful when a deck has both positive and negative active edges.

Example

```
# To specify clk1 and clk2 being negative edge active clocks and
# clk3 and clk4 being positive edge active clocks.
set_var px_negedge_clock {clk1 clk2}
Set_var px_posedge_clock {clk3 clk4}
```

px_power_assign

<all | clock | input | output | *<list of pins>*>	Specifies how internal power contribution is distributed among switching pins. Default: all

`all`	Distributes the internal power contribution equally among switching pins. This is equivalent to setting both `input` and `output`.
`clock`	Assigns all the switching power to the clock, independently from other input switching.
`input`	Distributes internal power contribution equally among switching input and clock pins.
`output`	Calculates the output switching power and assigns it to the output.
`list of pins`	Distributes the internal power contribution among the listed pins when switching.

px_power_single_point

`<0 | 1>` Allows power characterization to be performed for multiple input slews and output loads, based on the power template that is provided. Default: `1`

`0`	Characterizes all slew and load combinations specified in the power table.
`1`	Characterizes a single slew and load combination from the power table.

Use this variable to control whether a single point or all points in the power template are used for power characterization.

This variable must be used before the `char_ams` command.

px_probes_report

`<filename>` Specifies the filename or full path name of files to be used as the basename for reporting the results of automatic probing.

Liberate AMS produces the following two reports:

- *`<filename>`*: Contains a report of all possible probes found statically.

- *<filename>*`.red`: Contains a less-verbose report, which is shortened by considering switching activity.

px_read_spice_exit_on_missing_file

`<0 | 1>` Specifies whether Liberate AMS issues an error and exits if there is a file missing during the `read_spice` phase. Default: `0`

`0`	Does not exit if there is a missing file.
`1`	Exits and issues an error if there is a missing or unreadable file during the `read_spice` phase.

This variable must be set before the `read_spice` command.

Example

```
set_var px_read_spice_exit_on_missing_file 1
```

px_remove_rc_pincap

`{ "all" | "none" | "rail" |` *`net_name`* `}`

Controls the removal of parasitic RC networks from the selected nets of the input pin-capacitance partitions. Default: `rail`

`all`	Removes parasitic RC networks from all nets.
`none`	Does not remove any parasitic RC networks.
`rail`	Removes parasitic RC networks from rails (vdd, vss).
`net_name`	Removes parasitic RC network from the specified net.

This variable controls removing parasitic networks for a partition. The controls can be applied on a per-net basis. For example, the controls can be applied for supply rails, or for specific nets. The `all` and `none` values take precedence if used together with net names.

Example

```
# Remove parasitic RC network for rails and node net23.
set_var px_remove_rc_pincap "rail net23"
```

px_remove_rc_timing

{ "all" | "none" | "rail" | *net_name* }

Controls the removal of parasitic RC networks from selected nets of the input timing partitions. Default: `none`

`all`	Removes parasitic RC networks from all nets.
`none`	Does not remove any parasitic RC networks.
`rail`	Removes parasitic RC networks from rails (vdd, vss).
net_name	Removes parasitic RC network from the specified net.

This variable controls removing parasitic networks for a partition. The controls can be applied on a per-net basis. For example, the controls can be applied for supply rails, or for specific nets. The `all` and `none` values take precedence if used together with net names.

Example

```
# Remove parasitic RC network for rails and node net123
set_var px_remove_rc_timing "rail net123"
```

px_retaining_time

<0 | 1>

Controls the characterization of the `retaining_rise` and `retaining_fall` arcs. Default: `1`

`0`	Do not characterize `retaining_rise` and `retaining_fall` arcs.
`1`	Characterize the `retaining_rise` and `retaining_fall` arcs.

Note: Retaining (also known as valid time) arcs are characterized only when they are user defined.

Example

```
# Do not generate / characterize retaining time arcs.
set_var px_retaining_time 0
```

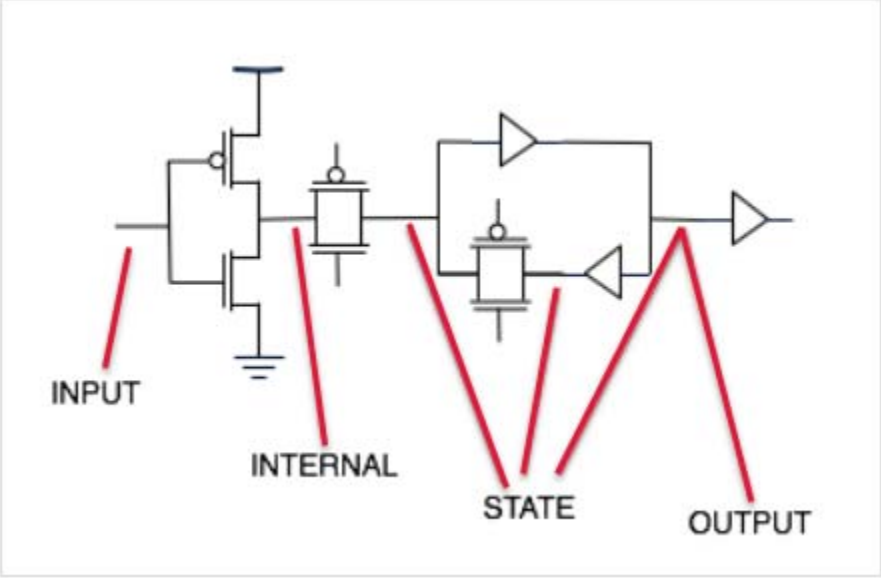
px_seq_probing

<string>

Sets the node types to be considered on a sequential component as possible candidates for probes in setup and hold characterization. Default: `state output internal input` (All node types are considered.)

You can use the following valid values all at the same time or in any combination: `state`, `output`, `internal`, and `input`.

This is the main variable used for controlling how automatic probing is done. It identifies which node candidates should be considered in sequential logic as per the diagram below of a typical latch input:



This parameter can also be specified in a table format using the the `define_table` command. Parameters specified in a table override parameters that are specified with a Tcl command.

px_setup_comb

`{ < pin_name | all | none > }`

Controls the automatic probe identification step in identifying candidates on combinational logic that is used for setup constraints probes. Default: `all`

`all`	Specifies that probing for setup characterization can be done on combinational logic for all input pins.
`pin_name`	Specifies an input pin for which probing for setup characterization can be done on combinational logic.
`none`	Specifies that combinational logic cannot be used for any pin for setup characterization.

Using the `:probe_inputs` modifier in conjunction with the `all` or *`pin_name`* options instructs the tool to widen the field of probe candidates to include the inputs of the combinational logic being probed.

For more information, see Appendix C, “Autoprobing in Liberate AMS.”

px_setup_seq

`{ < pin_name | all | none > }`

Controls the automatic probe identification step in identifying candidates on sequential logic that is used for setup constraints probes. Default: `all`

`all`	Specifies that probing for setup characterization can be done on sequential logic for all input pins.
`pin_name`	Specifies an input pin for which probing for setup characterization can be done on sequential logic.
`none`	Specifies that sequential logic cannot be used for any pin for setup characterization.

Using the `:probe_inputs` modifier in conjunction with the `all` or *`pin_name`* options instructs the tool to widen the field of probe candidates to include the inputs of the sequential logic being probed.

For more information, see Appendix C, “Autoprobing in Liberate AMS.”

px_simulation_interval

<value>	Specifies the simulation interval, in seconds, used by the FastSPICE step. Default: `10ns`

Set the value of this parameter to at least twice the value of the maximum delay that needs to be measured during the run. This is a global attribute and is applied to all tables and arcs.

Note: There are two other methods of specifying the simulation interval:

- Table-based, using `simulation_interval`. (See Appendix C, “Autoprobing in Liberate AMS.”)
- Arc-based, using an `-attribute` to `define_arc`.

If there are multiple definitions of the simulation interval, the precedence order is as follows:

1. Arc-based (highest precedence)
2. Table-based

3. Global, `set_var` for example (lowest precedence)

Example

```
# If max clk->out delay is expected to be around 5.2 ns …
set_var px_simulation_interval 12e-9
```

px_spv_api

`<0 | 1>`

Controls generation of API scripts and data to be used in conjunction with SpiceVision (SPV), a third-party product from Concept Engineering.
Default: `0`

`0`	Disables SpiceVision (SPV) write out.
`1`	Enables SpiceVision (SPV) write out.

px_transistor_variance

`<value>`

Specifies the voltage variance beyond which the device is considered to be switching.
Default: `50mv` (Typically, this default value does not need to be changed.)

See also *Manually specifying the threshold voltage (V_{th}) for inactive transistors*.

px_transistor_vth

<list>	Specifies a key/value pair list of transistor name and the corresponding threshold voltage. See also *Manually specifying the threshold voltage (V_{th}) for inactive transistors*.

If you specify the custom threshold voltage (V_{th}) variables, px_transistor_variance ***and*** px_transistor_vth***, ensure that they cover all the transistors in the design; otherwise, an error will be reported. This means that mixing and matching V_{th} and auto-logic level-based detection is not permitted.***

Example

```
set_var mx_transistor_vth [list \
    nch_lvt   400mV \
    nch_hvt  460mV \
    nch_xvt  500mV \
    pch_lvt   400mV \
    pch_hvt  460mV \
    pch_xvt  500mV \
    ]
set_var mx_transistor_variance 50mV
```

px_verbose

<0 | 1>	Controls the printing of basic flow and runtime information. Default: 1
0	Turns off the printing.
1	Turns on the printing.

Example

```
#Report basic flow info
set_var px_verbose 1
```

px_virtual_rail_auto_mode

`<0 | 1 | 2>` Automatically sets the virtual power supplies in a design. This variable enables you to find (px_find_virtual_rails) and define virtual power nodes in a design.
Default: `2`

`0`	There are no virtual rail findings. It is equivalent to `px_find_virtual_rail` set to `0`.
`1`	The `virtual.rpt` report is created. However, there is no automatic setting. You can refer to the report and set the virtuals accordingly.
`2`	Automatic settings along with the `virtual.rpt` report. (Default)

This variable works when `px_find_virtual_rails` is set to `2`.

px_virtual_rail_opposite_device_minimum_factor

`<integer>` Sets a threshold value for the count of maximum opposite devices. The net below the maximum opposite device counts is not considered as a virtual net.
Default: `1000`

For example:

```
wire_name    drive_pmos    drive_nmos    other_unset_wires
VDD_1         13038           8              unset
VSS_2         508            1038            unset
```

`VDD_1` is set as a virtual VDD because of connections to PMOS, and eight opposite NMOS are connected to the same net.

Similarly, `VSS_2` where 508 PMOS are connected, which is below the default value and because of 1038 NMOS connections, is set as a virtual net.

px_virtual_rail_minimum_xtrs

<integer> — Sets a lower limit for the number of wire drive MOS.
Default: `50`

The wire that has more drive MOS than `mx_virtual_rail_minimun_xtrs` is a virtual rail.

For example:

```
# Defining minimum number of mos to "70"
set_var mx_virtual_rail_minimum_xtrs 70
```

set_var_failure_action

`<warning | error>` — Notifies the `set_var` command how to consider a failure.
Default: `warning`

- `error` — Displays an error message when a `set_var` command fails and suppresses execution of any subsequent commands like `char_library` that might result in characterization or library generation. However, subsequent `set_var` commands are still allowed to check those for correctness.
- `warning` — Displays a warning when a `set_var` command fails. The failed `set_var` command is ignored and the execution of subsequent commands continues.

This variable must be set before any other `set_var` command.

static_autoprobing_max_depth

<integer> — Specifies the maximum level of CCCs to traverse during autoprobing for constraint arcs in Liberate AMS static mode.
Default: `100`

static_autoprobing_max_states

<integer>

Specifies the maximum number of states (that is, latches) on a path when autoprobing.
Default: `0` (automatically decides the number of states from setup/hold probing levels)

static_clock_tree_state

`<0 | 1 | 2>`

Allows sequential elements in the clock path for timing constraint characterization. Adjust this variable if the clock path incorrectly passes through sequential elements.
Default: `2`

`0`	Allows combinational logic only.
`1`	Allows combinational logic and latches.
`2`	Allows combinational logic, latches, and flip-flops.

static_measure_output_range

<boolflag>

Controls the support for non-full swing voltage propagation. Simulation, measure, and path trace all use the voltage range of the real waveform. This variable works together with `ecsm_measure_output_range=1`.
Default: `false`

static_path_skip_double_latches

<boolflag>

Skips tracing paths through two consecutive latch Channel-Connected Components (CCCs). Use this variable mainly to avoid the data path that goes through the master-slave flip-flops. This variable lets you handle the corner case where a flip-flop is partitioned into two separate CCCs.
Default: `false`

static_sat_mos_limit

<integer>

Specifies the maximum number of transistors allowed in one CCC for the CCC to be sensitized. Increasing this variable allows an oversized CCC to be sensitized, which in turn causes the sensitization time to increase significantly.
Default: `500`

A

Liberate AMS Flow Examples

Example 1: Tcl Script to Generate a Template File from a Reference Library

`write_template.tcl` reads in a reference library that you provide and generates a template or macro (`template.tcl`), which is used as an input file for the next script.

```
read_library [pwd]/ref.lib
set_var default_timing off
set_var default_power off
exec mkdir -p [pwd]/tmpl
write_template -expand_buses -map {INT_CLK PLL_CK_0} -verbose [pwd]/tmpl/
                template.tcl
```

Example 2: Tcl Script to Characterize a Block and Generate Models

`ams.tcl` uses `template.tcl` as the input TCL file to define the macro, sets the required characterization corner, and specifies the SPICE input and how to read it in to the tool. It then defines the user specified deck to provide the stimuli for the run. The heart of this script is the `char_ams` command, which performs the requested characterizations. The `write_library` commands then write the results into `.lib` files.

```
# SET CORNER
# SETUP DIRECTORY STRUCTURE
# INPUT FILES
# OUTPUT FILES
# READ IN SPICE
# SPECIFY SPICE ENGINES and OPTIONS
# SPECIFY RAILS
# DEFINE MACRO
# DESCRIBE FUNCIONALITY
# SETUP PARTITION/PROBING/CHAR. PARAMETERS
# MAIN COMMAND
```

```
# WRITE MODELS

### SET CORNER
    set process tt
    set voltage 1.0
    set temperature 75
    set_operating_condition -temp $temperature -voltage $voltage

### SETUP DIRECTORY STRUCTURE
    set cell dac_current
    set datadir [pwd]

### INPUT FILES
    set tcldir   ${datadir}/tcl
    set spicedir ${datadir}/spice

### OUTPUT FILES
    set outdir ${datadir}/out.$process
    set_var px_dir ${outdir}/ams ;#specify partition sim dir
    set ldbdir ${outdir}/ldb; exec mkdir -p $ldbdir
    set libdir ${outdir}/lib; exec mkdir -p $libdir

### SPECIFY RAILS and VIRTUAL RAILS
    set_vdd avdd $voltage
    set_gnd agnd 0
    set_vdd -virtual -no_model VDD_sw $voltage

### READ IN SPICE
    set spectre_model true
    set spectre_netlist true
    set netlistfile ${spicedir}/${cell}.spf
    set modelfile ${spicedir}/models/include_${process}
    set_var extsim_model_include $modelfile
    set_var extsim_deck_include 1

    ## define primitive models
    ## DESIGN ELEMENTS:
    ## - N,P fets : define_leafcell -type nmos/pmos …
    ## - anything else : define_leafcell -type black_box
    define_leafcell -type nmos -pin_position {0 1 2 3}  {g45n1svt g45n1hvt}
    define_leafcell -type pmos -pin_position {0 1 2 3}  {g45p1svt g45p1hvt}
```

```
    define_leafcell -type black_box -pin_position {0 1 2} {g45rnsnp}

    set spicefiles {}
    lappend spicefiles $modelfile
    lappend spicefiles $netlistfile
    if {$spectre_model && $spectre_netlist} {
        read_spice -format spectre $modelfile
        read_spice -format spectre $netlistfile
    } elseif {!$spectre_model && $spectre_netlist} {
        read_spice $modelfile
        read_spice -format spectre $netlistfile
    } elseif {$spectre_model && !$spectre_netlist} {
        read_spice -format spectre $modelfile
        read_spice $netlistfile
    } else {
        read_spice $spicefiles
    }

### SPECIFY SPICE ENGINES and OPTIONS
    set partsim "aps"
    set charsim "aps"

    if {$partsim == "aps"} {
        set_var fastsim_cmd "spectre"
        set_var fastsim_cmd_option "+aps -64 +errpreset=liberal"
        set partsim "spectre"
    }
    if {$charsim == "aps"} {
        set_var extsim_cmd "spectre"
        set_var extsim_cmd_option "+aps -64"
        set charsim "spectre"
    }
    set_var extsim_deck_dir ${outdir}/decks
    set_var extsim_save_failed deck
    set_var extsim_save_passed none

### DEFINE MACRO
    # template.tcl sourced below can be generated
    # automatically from a reference library ref.lib
    # with the following commands:
    # - read_library ref.lib
    # - write_template -verbose template.tcl
    source ${tcldir}/template.tcl
```

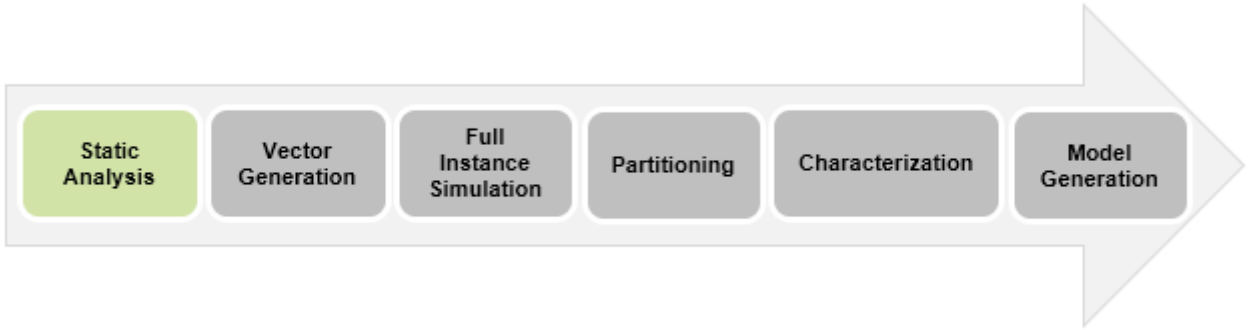
```
    # Use sim. deck to drive characterization
    define_deck -observation_window 125e+06 ${spicedir}/deck.sp $cell

### MISCELLANEOUS
    set_var px_partition_name_use_arc 1
    set_var px_fastsim_reuse 1 ;# 0=Don't reuse partition simulation results

### DEBUG
    set_var px_debug "memory"

### MAIN COMMAND
    char_ams -partsim $partsim -charsim $charsim -user_arcs_only
```

B

Debugging during Liberate AMS Run

Liberate AMS can be configured to output a lot of critical information to aid debugging. In this appendix, let us look at the information that this tool can provide. The information available for debugging at every stage in the timing characterization process is discussed below.

Static Analysis Debugging



Static analysis takes only a few minutes even on large designs. The log file captures the information such as following during static analysis:

```
(AMS-info) - Partitioning - static - start ...
    | (AMS-info) - Partitioned 815521/818357 xtrs wall clock time += 22 sec
    | (AMS-info) - Merging loops ... wall clock time += 4 sec
    | (AMS-info) - Propagating constants ... wall clock time += 0 sec
(AMS-info) - writing gate-level ... wall clock time += 3 sec
    | (AMS-info) - ccc to gates ... wall clock time += 9 sec
(AMS-info) - Partitioning - static - ... end wall clock time __ 38 sec
```

If Liberate AMS gets stuck during static analysis, check whether all rails have been defined. Switched rails can cause hindrances while conducting static analysis. Therefore, set the following flag and rerun the analysis:

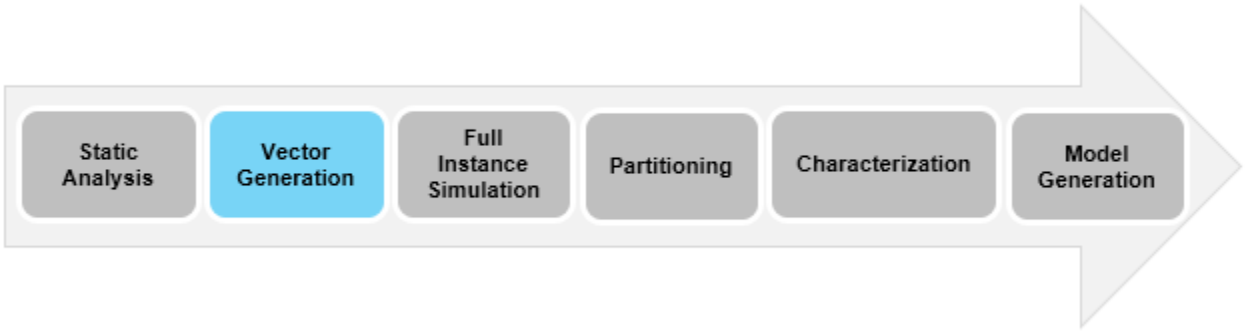
```
set_var px_find_virtual_rails 1
```

Consequently, the log file would also contain messages about the virtual rails found, the power switched nodes that should be treated as logic constant during static analysis, and instructions on how to define them as virtual rails.

```
(AMS-info) - Partitioning - static - start ... (AMS-info) | Finding
virtual rails ... wall clock time += 0 sec
| | (DEBUG_STATIC) - I_reg.vdig - was no set as virtual rail. If the next
static partitioning step takes longer than expected (i.e. > 1 sec / 100K xtrs)
or if dynamic partitions contain more than few thousands xtrs, check the node
and add 'set_vdd (set_gnd) -virtual -no_model I_reg.vdig $vdd ($gnd)' to your
script. Then restart your run. Attached channel ports: 1000
```

For detailed information, see Defining the Virtual Rails.

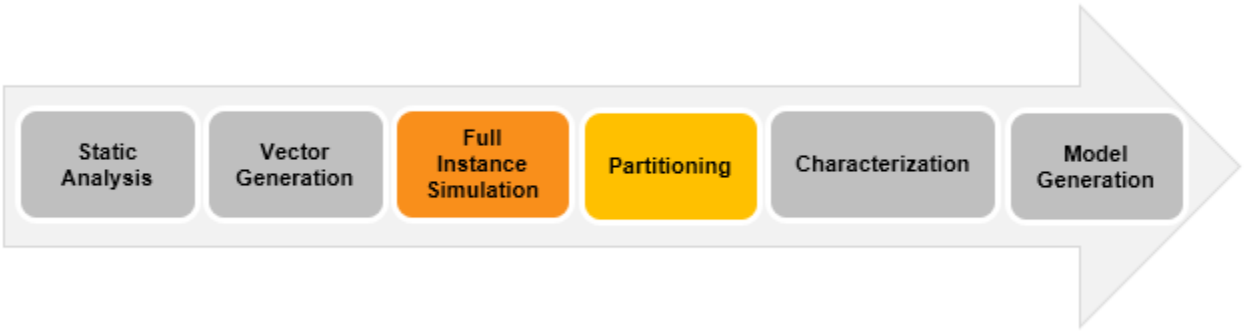
Vector Generation Debugging



During vector generation, the testbenches are simulated and table files are generated. This is also a quick simulation and should not take more than a few minutes. An empty subcircuit shell is used for the block being characterized because the intent is to capture only the stimuli at the input pins of the block.

The names of the run decks and waveform databases generated in this step contain the characters **in**. This is used to differentiate these databases from the files generated for the Full Instance simulation. The waveform databases can be used to check whether the input setup is correct.

Full Instance Simulation and Partitioning Debugging



The table files generated during vector generation are used to drive simulation on the full instance. The waveform databases generated from the full instance simulation can be used to verify waveforms from the different testbenches. This is done in conjunction with the report in the `ams.info` file. This file contains the list of arcs that are found after processing the waveform database for the full instance simulation.

Using the ams.info File

The `ams.info` file is one of the first files to look at if there are missing arcs in the generated library. If the arc information is not in this file, it could be due to incorrect testbench setup. If this is the case, first examine the waveform database for the top-level full instance run.

The `ams.info` file also provides information on the size of the partition created for this arc. The size of the partition should not exceed a couple of thousand transistors even for big designs. If the partitions are bigger than this, one approach would be look for virtual rails. For detailed information, see Defining the Virtual Rails.

This file also provides similar information for constraint arcs. There is additional information provided here on the internal probe location found for the constraint arc.

Using Variables for Debugging

This section covers the variables that can be enabled to further aid in debug after the full instance simulation.

- To generate a report file with arcs that are specified by you, but are not found after partitioning, set the following commands:

```
set_var px_check_arcs 1
set_var px_arc_report "<filename>"
```

- To abort the tool if there is a mismatch between the arcs specified by you and the arcs found after partitioning, set the following command:

  ```
  set_var px_check_arcs_exit_on_missing 1
  ```

- As pin capacitance characterization is very straightforward, you can disable it while debugging the missing timing arcs. To save on run time and reduce the amount of information in the `ams.info` file, set the following command:

  ```
  set_var px_pincap_char 1
  ```

- Use the px_transistor_variance and px_transistor_vth variables to manually specify the threshold voltage (V_{th}) for inactive transistors.

  In general, Liberate AMS does not require threshold voltage information about transistors and assumes that transistors are on/off depending on the gate voltage corresponding to its logic level of `0` or `1`. Liberate AMS automatically figures out the on/off voltage for transistors by looking at the minimum and maximum voltage swing on the net connected to the transistor gate during the top-level FastSPICE simulation.

  However, if the design has a case where a transistor gate voltage is held at steady-state throughout the simulation and the transistor is on all the time, Liberate AMS is not be able to determine the on/off voltage due to lack of voltage swing. As a result, it prunes out these active transistors during partitioning. For this rare scenario, you can specify a custom threshold voltage using the `px_transistor_variance` and `px_transistor_vth` variables.

Characterization and Model Generation Debugging

After the partitions are generated, the next step is to send them for characterization. It is possible that an arc found during full instance simulation fails during characterization. You might also like to see the partition netlist that is created for verification.

A common cause of failure of characterization is the use of a very large load capacitance in the delay table that results in failure of the output to transition properly with this capacitance value. In this case, there will be only one delay folder. This issue can be verified as follows:

1. Open the `sim.sp` file from the delay folder.
2. Search and remove the option `save=nooutput` to enable saving of the waveform data.
3. Save the `sim.sp` file and run characterization simulator on it.

Tip

The net names are mapped to numbers in the `sim.sp` file. The top of the file provides a mapping of the net names to these numbers.

Plot the waveforms for pin and related pin. You should see that the waveform for pin would not have transitioned completely. The final step to confirm that the load value specified is too high is to run the full instance simulation with this load. By default, full instance simulation is run with the first value in the table for slew and load. There is flexibility to change this value in case of the debug scenario mentioned above using the commands listed below:

```
set_var px_fastsim_input_slew <value>
set_var px_fastsim_load <value>
```

To verify any other issues associated with partitioning (including accuracy loss with partitioning), it is also possible to generate a library directly from the full instance simulation without doing any partition (graybox flow). This mode of running will significantly increase the run time. In addition, it is recommended to use a simulator like Spectre® APS for the full instance run as results of such runs are used directly to create the library.

```
set_var px_greybox 1
```

Defining the Virtual Rails

For Liberate AMS to efficiently understand the top-level AMS instance, it is necessary to define the virtual rails that are present in the design. To define the known virtual rails in the flow, the `set_vdd` and `set_gnd` commands are used with the `-virtual` option, as shown below:

```
set_vdd -no_model -ignore_power -virtual VDD_int 1.0
set_gnd -no_model -ignore_power -virtual VSS_int 0
```

Sometimes, you might know the partial name of a virtual rail and not the full name. This situation tends to occur when the netlist is the output of an extraction tool and the names are changed. In this case, you can use the `set_virtual` command to specify a pattern to be used for virtual rail names, as shown below:

```
set_virtual -vdd *VDD_int* 1.0
set_virtual -gnd *VSS_int* 0
```

To identify the virtual rails contained within a top-level AMS instance, you can set up Liberate AMS using the following command:

```
set_var px_find_virtual_rails 1
```

By doing this, Liberate AMS finds out the nodes that are connected to a large number of drain terminals and print them in the log file, as shown below:

```
(AMS-info) memory usage [] (partition) - virtual=776.375 MB - resident=40.7734 MB
  (AMS-info) - Partitioning - static - start ...
(AMS-info)  | Finding virtual rails ...
  |  | (DEBUG_STATIC) - I_regin_6_.n30 - was no set as virtual rail. If the next static partitioning step takes
   longer than expected (i.e. > 1 sec / 100K xtrs) or if dynamic partitions contain more than few thousands
   xtrs, check the node and add 'set_vdd (set_gnd) -virtual -no_model I_regin_6_.n30 $vdd ($gnd)' to your
   script. Then restart your run Attached channel ports: 4

  |  | (DEBUG_STATIC) - I_mux_3_.n0 - was no set as virtual rail. If the next static partitioning step takes
   longer than expected (i.e. > 1 sec / 100K xtrs) or if dynamic partitions contain more than few thousands
   xtrs, check the node and add 138_1          5%
```

For each node that is reported as a possible virtual rail, you need to add the definition for its setup and run again. Once all virtual rails are identified, the characterization is run for completion.

Liberate AMS also reports all possible nodes once. In addition, the nodes that meet certain criteria are automatically set as virtual rails. This feature can be setup as shown below:

```
set_var px_find_virtual_rails 2
```

A report file is generated containing all suspected virtual rails. This report is located at `ams/virtual.rpt`.

```
*** wire_name              drive_pmos    drive_nmos    virtual_vdd
N_XIO-Q_3_47_c_1031883_n      2204          0          set  (has been set as virtual_vdd)

*** wire_name            drive_pmos    drive_nmos    virtual_gnd
N_XIO-VSS_c_697803_n         0             870          set (has been set as virtual_gnd)

*** wire_name            drive_pmos    drive_nmos    other_unset_wires
N_XIO-DBL_c_1280305_n        10            68          unset (not set as virtual rail)
```

This report contains a summary of each possible virtual rail, its PMOS and NMOS drain connections.

Following are the requirements for automatic setting of a node as virtual VDD.

- PMOS drain connections are greater than px_virtual_rail_minimum_xtrs. Default is `60`.

- Zero NMOS drain connections.

Following are the requirements for automatic setting of a node as virtual GND.

- NMOS drain connections are greater than `px_virtual_rail_minimum_xtrs`. Default is `60`.
- Zero PMOS drain connections.

Other nodes that do not meet the criteria are reported and can be set by the user using the previously explained techniques.

Automatic setting of virtual rail also depends on the opposite polarity device connection on same net, which is controlled by px_virtual_rail_opposite_device_minimum_factor. (Default `1000`)

Recommended setting:

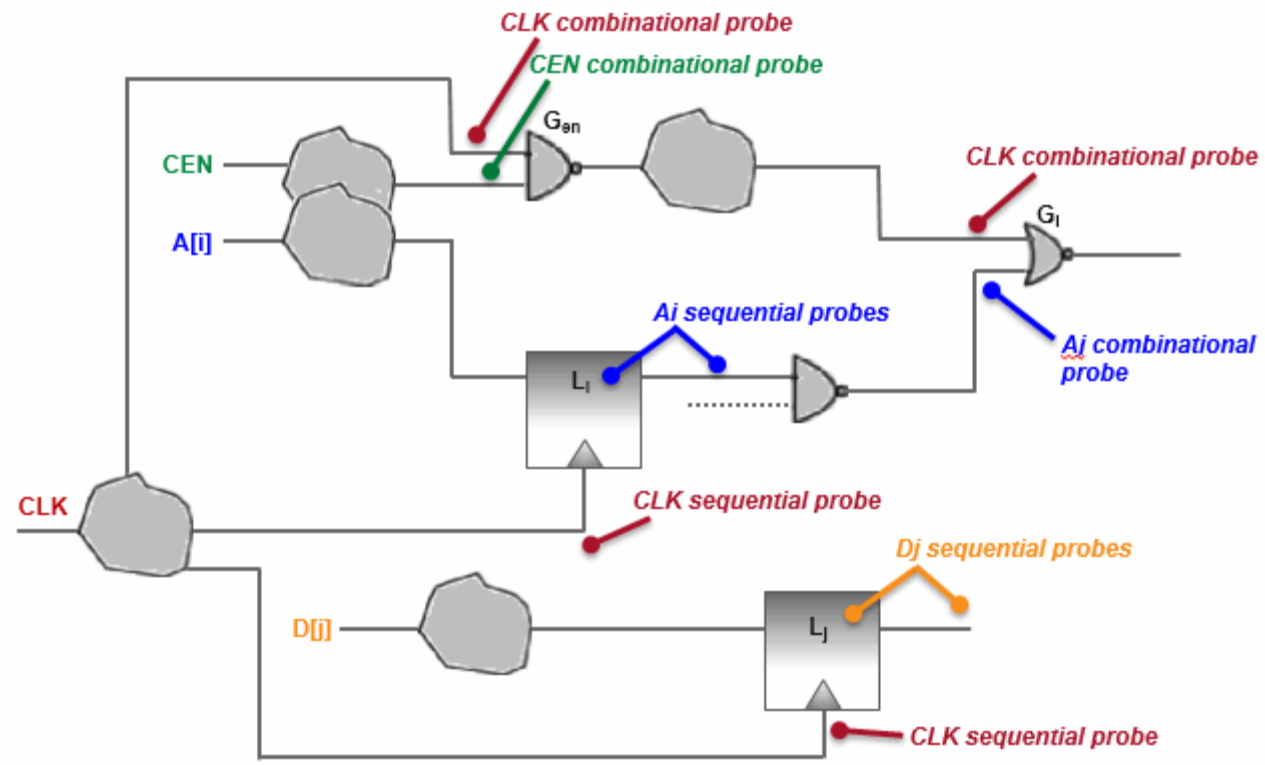
You should set up the flow to use automated mode using the px_virtual_rail_auto_mode variable which is defaults to `2`.

C

Autoprobing in Liberate AMS

When probing for constraint characterization, Cadence® Virtuoso® Liberate AMS automatically probes any structure that lies on the intersection between the constraining and constrained domains. Reconvergent paths are allowed up to two levels of reconvergency; there is no assumption on domain types (that is, clocked vs. non-clocked) or intersection logic type (that is, sequential vs. combinational). The figure below shows some examples of probing on different logic types.

- Clock gated combinational logic on non-clock inputs (G_{en}).
- First-level latches (L_i, L_j).
- Clock gated combinational logic on outputs of first-level latches (G_i).



By default, Liberate AMS uses first-level sequential and combinational probes for hold constraint characterization for all pins. It uses first and second-level sequential and combinational nodes for SETUP constraint characterization for all pins.

This corresponds to the following values for the variables:

```
set_var px_setup_seq  "all"
set_var px_hold_seq   "all"
set_var px_setup_comb "all"
set_var px_hold_comb  "all"

set_var px_autoprobing_hold_level 0
set_var px_autoprobing_setup_level 1
```

Often, probing for constraint characterization is a function of the input pin for which constraints are to be characterized. For this reason, flags can take pin-dependent values. Referring to the figure above, you would specify:

```
set_var px_setup_comb "CEN A[i]"
```

to allow for combinational probes on `CEN` and `A[i]` pins (and those pins only) to be used for setup constraint calculation for that pin.

D

Truth Table Format

Specifying Input Stimuli

In Liberate AMS, you can specify input stimuli by passing one or more truth table files to the `define_table` command.

define_table command

The `define_table` command specifies a list of truth table files. The number of specified table files determines the number of FastSPICE runs that will be performed in parallel—use the different options of the char_ams command available to control the number of threads, for example, `-partthread`, `-charthread`, `-part_fastsim_thread` and so on.

Following is the file syntax for the `define_table` command:

```
arctypes <arc_type>
<fastsim_option>
# <comment_string>
table <table_id>
pins <bus_id> <bus_id> … <bus_id>
    <cycle_id> <value> <value> … <value>
endtable
```

Where:

`<arc_type>`
: Vectors specified in the table file will be used for `<arc_type>` characterization. It accepts one or more of the following values: `delay`, `retain`, `setup`, `hold`, `minperiod`, `mpw`, `power`, `leakage`, `measure`, `timing`.

  Note: Timing equals delay retain setup hold. If `<arc_type>` is not specified, the default is `timing` measure.

<fastsim_option>	It accepts one the following values: fastsim, fastsim_reuse, fastsim_deck_include, fastsim_deck, simulation_interval, fastsim_auto_ic, fastsim_model_include, fastsim_cmd, fastsim_cmd_option.

fastsim

<fastsim_indentifier>

One of the following: spectre, aps, or xps

Specifies which FastSPICE engine will be used to simulate the table file. Overwrites the output of the char_ams -charsim command.

This example specifies Spectre as the FastSPICE simulator:

```
fastsim spectre
fastsim_deck .option rawfmt=fsdb
```

fastsim_auto_ic

<value>	Either 1 or 0 (May also be specified as true, on, false, off)

Instructs Liberate AMS to add or not add .ic statements to bit-lines and core nodes, as recognized in the static topology recognition step. It overwrites the global variable px_fastsim_auto_ic for the table file it is specified in. If not specified, the default value is given by the value of px_fastsim_auto_ic, which has a default of true.

The simulation_interval command overwrites px_simulation_interval for the table it is specified in. This can be used in a table of any type: power, leakage, timing, measure, and so on.

Syntax:

```
simulation_interval <time>
<time> == <value><?unit>
<value> == a number (in any notation)
<?unit> == one of f,p,n,u,m,fs,ps,ns,us,ms
```

Note: The following are equivalent:

```
simulation_interval 20e-9
```

```
simulation_interval 20n
simulation_interval 20ns
```

Example:

```
simulation_interval 20e-9
```

The command above if specified in the `timing.tbl` table ensures that vectors listed in that table will use a simulation interval of 20ns rather than the default 10ns (which will be used for any other table that does not have a `simulation_interval` command).

fastsim_clock_slew

<double> Specifies what clock slew to use in FastSPICE simulation. For more information, see px_fastsim_clock_slew.
Default: `-1pf`

fastsim_cmd

<path_to_executable>

Specifies the path to an executable to be used for simulating this table file.

fastsim_cmd_option

<command_options> Specifies command line options to be passed to the executable used for simulating this table file.

Example:

```
fastsim_cmd /home/tools/mmsimcm_v4/lnx86/latest/bin/spectre
fastsim_cmd_option +spice -64
```

Note: When Spectre XPS is used, the output format defaults to FSDB during partitioning. This overwrites any value coming from `fastsim_deck` options in tables *or* the `extsim_cmd_options` command in scripts.

Re-using fastsim results:

```
fastsim_reuse
```

When present, instructs Liberate AMS to look for previous fastsim results on this table file. fastsim_reuse results are stored for each run inside the directory:

```
./px_fastsim - with name <cellname>.<tablename>.alwf
```

This overwrites `px_fastsim_reuse`.

fastsim_deck

<fastsim_deck_string>

A valid text (for the engine specified in fastsim) that will be included (as is) to the deck being simulated in FastSPICE.

This variable is used to specify truth table files FastSPICE simulator options in Liberate AMS. This optimizes how options are passed to FastSPICE and unifies it into a single "`fastsim_deck`" line. You can then specify anything that needs to go to the fastsim deck, meaning there will be no interpretation of the option/command as previously done. Example:

```
# Using timing table to pass FastSPICE simulator specific options
arctypes leakage
fastsim spectre
fastsim_deck .param simpreset=5
fastsim_deck .param pn_level=5
fastsim_deck .param cgnd=1e-15
fastsim_deck .param sfe_compaction=0
fastsim_deck .param keepparaname=0
fastsim_deck .param rshort=2
fastsim_deck .param hier_delimiter=.
fastsim_deck .param dc_turbo=3
fastsim_deck .param rcr_fmax=1G

table...
  ...
endtable
```

fastsim_deck_include

<netlist_path_name>

Full path name of a SPICE netlist

Specifies what netlist to run fastsim on for this table, when different than the main netlist read into the database through the `read_spice` command. It is usually used to run power characterization on a different netlist than the one used for timing. It assumes:

- The netlist specified has the exact same .subckt interface as the main one and, if used for other than the power table, all nodes contained in this netlist must be contained in the main one as well.
- That `extsim_model_include` is specified, otherwise the option will be ignored.

The name of the file must be a full path, else the option will be ignored.

fastsim_input_slew

<double>	Specifies the input slew to be used in FastSPICE simulation. For more information, see px_fastsim_input_slew. Default: `-1pf`

fastsim_load

<double>	Specifies the load to be used in FastSPICE simulation. For more information, see px_fastsim_load. Default: `-1pf`

fastsim_model_include

<model_path_name>	Specify the full path name of a SPICE model file for power or leakage characterizations. Default: none

Specifies a full path to a model file to be used with fastsim for power or leakage characterizations.

Liberate AMS will only use this model file if it is different from the file read into the database through the read_spice command. The name of the file must be a full path or the option will be ignored. Example:

```
fastsim_model_include "/home/users/models/XYZ/power.mod"

table...
...
endtable
```

Required Keywords

The table, pins, and endtable entries are required keywords.

```
<table_id> : <string>
<cycle_id> : <string>
<bus_id> : <string>
<value_string> : R F ? 0 1 B C D - X H L A P N onehot onecold
```

The `table_id` is an arbitrary string which is used to identify the table. This ID must be unique for each table.

The `cycle_id` is any string and is used as a placeholder to position the pin data. If the `cycle_id` is set to `minperiod` or `output_period`, special functionality is enabled.

The `bus_id` is the name used in the `define_cell` command to refer to a circuit port.

Valid characters that can be used in the `value_string` are:

Inputs

Value	Description
R F	Rising (Falling) edge; expands the table to generate a 0->1 (1->0) sequence while modifying the other values on the same line. Specify 1 (one) edge per row.
1	One, bus-size independent; expands in to 00...00->11...11 when on the same line with an edge; to …1111 otherwise.
0	Zero, bus-size independent; expands in to 11...11->00...0 when on the same line with an edge; to …0000 otherwise.
B	Binary; expands in to …0000->…1111->…1111->…0000 when on the same line with an edge; to …0000->…1111 otherwise.
I	Inverted binary; expands in to …1111->…0000->…0000->…1111 when on the same line with an edge; Opposite of B.
A	a, bus-size independent; expands in to …0101->…1010 when on the same line with an edge; to …1010 otherwise.
L	Low, bus-size independent; tied to 00...00 independently from any expansion it's involved in.
H	High, bus-size independent; tied to 11...11 independently from any expansion it's involved in.
?	Don't care; expanded to ??...?? and tied to 00...00 independently from any expansion it's involved in. Does not affect other measurement options.
X	Overwrite any other output letter on the line

5	Five, bus-size independent; expands in to …1010->…0101 when on the same line with an edge; to …0101 otherwise.
P N	Positive (negative) pulse; expands the table to generate a 0->1->0 (1->0->1) sequence while keeping the other values on the same line untouched.
onehot onecold	Expands the table to generate a 00…01->00…10->...->01…00->10…00 (11…10->11…01->…->10…11->01…11) sequence while keeping the other values on the same line untouched.

Outputs

Value	Measurement	
B	All *(constraints, delay, measure, and power)*	
C *or* -	Constraints *(setup & hold)*	
D *or* **delay**	Delay	
H *or* **hold**	Hold constraint	
J *or* **power**	Switching & hidden power	
M	"define_measure" + mpw + minperiod + min & max clock tree paths	
These offer further granularity over "M"	mpw	Minimum pulse width (mpw) only.
	min_period *or* **minperiod**	Minimum period only.
	min_clock_tree_path	Minimum clock tree path only.
	max_clock_tree_path	Maximum clock tree path only.
S *or* **setup**	Setup constraint	
W *or* **leakage**	Leakage power	
X	- nothing - (Do not measure constraints, delay, or power)	
Z	Tri-state arcs	

In case the output is a bus, it is possible to instruct Liberate AMS to measure only specific bits, using hex notation as bit-mask:

```
### only look at bit 0 of output 0…
pinsCLK AWT TME ME  WE  WEM D   ADR OE  Q
readA   0   0   1   0   0   ?   b   1   0x1
```

Values are automatically expanded to the proper size of the bus they are specified for.

Note: The first line in the table must not perform any measurements (in other words, it acts like a dummy line.) This allows the memory cell to achieve a stable condition in simulation

before measurements are taken. In the code example below, measurement on the Q pin is set to "don't care" in the first line of the table.

Default Values

Pins or busses can be given a default value, which can simplify entering data into a table. Default values may be specified with the Tcl variable, amstable_dontcare_value, or within a table file. Within a table file, use the keyword, `dontcare_value`.

Within a table file, `dontcare_value` can be specified globally for all tables, or on a table-by-table basis. To specify for all tables, place within the file after the “arctypes” statement.

Example:

```
arctypes delay enable disable
dontcare_value din 1
```

To specify on a table-by-table basis, place within a table immediately following the "pins" identifier inside the table. The `dontcare_value` is used if there is no explicit value assigned, or wherever the table has a question mark (?) for a value.

Example:

```
dontcare_value tx_q 1
# tx_q=1 will be used throughout this table
table seldesel
pins            clk    cs      bw      din    oen dout
deselect        R      1       L       ?      ?       X
select          R      0       L       ?      ?       X
deselect        R      1       L       ?      ?       X
endtable
```

Precedence of `dontcare_value` directives is:

```
(table level) > (table file level) > tcl command level > default (=0)
```

E

Legacy Commands and Variables

This chapter lists all the commands and variables that are either deprecated, or included for backward compatibility. We would like to discourage users from relying on these commands and variables. Instead, find the alternate command or variable along with its settings to achieve the best results from Liberate AMS.

Deprecated Commands

The following commands are being phased out, and have been replaced by either new commands (or related options), new variables, or new behaviors of the tool.

px_set_finesim_para

<list>	Passes a list of FineSim parameters as name-value pairs. The specified parameters are used during FastSPICE simulation.

This variable is used to pass parameters to the FineSim simulator. Parameters are passed as a list of name-value pairs.

Parameters can also be passed by specifying them in a table and using the define_table command. Parameters specified in a table override parameters that are specified with a Tcl command.

px_set_hsim_param

<list>	Passes a list of HSIM parameters that is used during FastSPICE simulation.

This variable is used to pass parameters to the HSIM simulator. Parameters are passed as a list of name-value pairs.

Parameters can also be passed by specifying them in a table and using the `define_table` command. Parameters specified in a table override parameters that are specified with a Tcl command.

px_set_nanosim_param

<list>	Passes a list of NanoSim parameters that is used during FastSPICE simulation.

This variable is used to pass parameters to the NanoSim simulator. Parameters are passed as a list of name-value pairs.

Parameters can also be passed by specifying them in a table and using the `define_table` command. Parameters specified in a table override parameters that are specified with a Tcl command.

px_set_ultrasim_param

<list>	Passes a list of UltraSim parameters that is used during FastSPICE simulation.

This variable is used to pass parameters to the UltraSim simulator. Parameters are passed as a list of name-value pairs.

Parameters can also be passed by specifying them in a table and using the `define_table` command. Parameters specified in a table override parameters that are specified with a Tcl command.

Example

```
px_set_ultrasim_param { \
    {simpreset 5} \
    {pn_level 5} \
    {cgnd 1e-15} \
    {sfe_compaction 0} \
    {keepparaname 0} \
    {rshort 2} \
    {hier_delimiter .} \
    {dc_turbo 3} \
    {rcr_fmax 1G} \
}
```

F

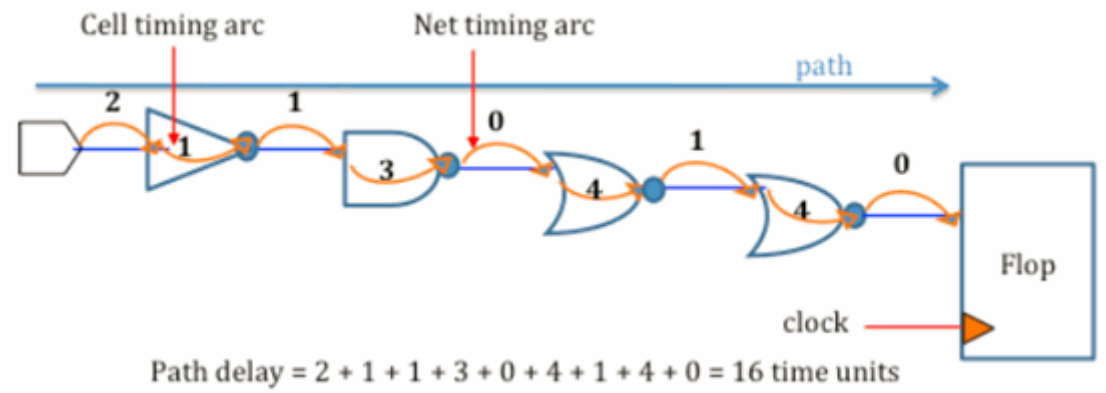
Glossary

Inside View

All characterization solutions available in the Virtuoso Characterization Suite use a unique “inside view” pre-characterization circuit analysis technique to perform vector generation and binning, automatic indices selection, and optimization of timing constraint characterization. This enables fully-automated library creation mechanism.

Static Timing Analysis (STA)

It is an approach to verify timing of digital designs that uses cell delays and net delays to obtain path delays which is used to validate timing specification. It validates if the design can operate at the rated speed. The figure below shows a simple example in which the cell and net delays are used to obtain the path delay.



Timing Libraries

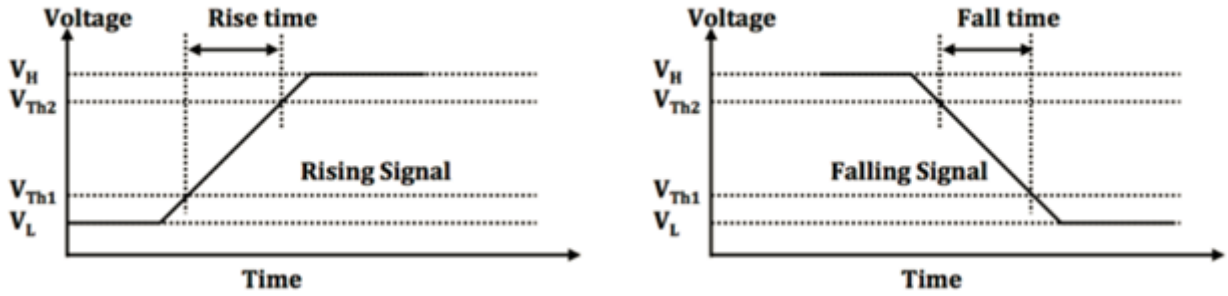
The Liberty format (`.lib`) is the industry standard for specifying the timing information. It is an ASCII file containing timing and power numbers associated with a cell. These are obtained by running SPICE simulations on the cell under a range of conditions.

Timing Arcs

A timing arc is a construct used to represent a single causal (change in input causes change in output) relationship. These arcs form the building blocks for STA.

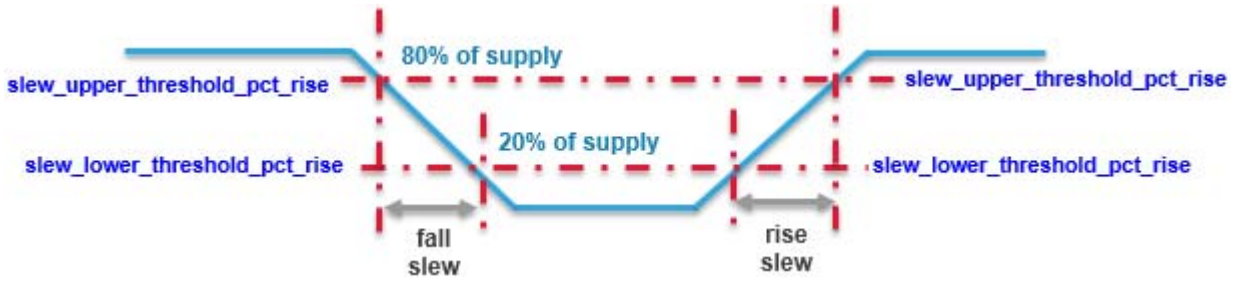
Transition Time

It is defined as the time it takes for a signal to change states between two specific levels. Rise and Fall transitions times, as shown in the figure below, are properties of a timing arc.



Slew

As slew rate is the rate of change, slew is typically measured in terms of transition time. Thresholds of signal transition times are used to measure slew. In the figure below, the threshold setting specifies that the falling slew is the difference between time points when the falling edge reaches 80% and 20% of supply value.



The slew thresholds are typically chosen to correspond to the part of the waveform that is linear. In newer technologies, the timing libraries will have these thresholds set to 30% and 70% of supply.

Cell Delay

Propagation delay through a cell is commonly known as cell delay. The slew of the input waveform and the load connected to the cell influence the delay value. The delay values are characterized for different values of input slew and output load. Typical delay measurements are done from 50% of input signal to 50% of output signal.

Related Pin

`Related_pin` attribute defines pin(s) that represent the beginning point of the timing arc. This attribute is required in all timing groups.

Timing Arc Types

Following types of timing arc can be used:

- Combinational timing arcs:

  These are used to describe the timing arcs for combinational element. The timing arc will be attached to an output pin and the related_pin will be an input or an output. Types of combinational timing arcs:

  - combinational
  - combinational_rise
  - combinational_fall
  - three_state_disable
  - three_state_disable_rise
  - three_state_disable_fall
  - three_state_enable
  - three_state_enable_rise
  - three_state_enable_fall

- Sequential timing arcs:

  These describe timing arcs for sequential elements. It can be a delay arc (if it describes relation between clock transition to data output i.e. input to output) or a constraint arc (if it describes relation between clock transition and data input i.e. input to input). Sequential timing arcs can be one of the following:

  - Edge-sensitive (rising_edge or falling_edge)
  - Preset or clear
  - Setup or hold (setup_rising, setup_falling, hold_rising, or hold_falling)
  - Nonsequential setup or hold (non_seq_setup_rising, non_seq_setup_falling, non_seq_hold_rising, non_seq_hold_falling)
  - Recovery or removal (recovery_rising, recovery_falling, removal_rising, or removal_falling)
  - No change (nochange_high_high, nochange_high_low, nochange_low_high, nochange_low_low)

Timing Sense

This attribute is used in the library to specify unateness in the .lib file.

Setup and Hold

These are synchronous timing checks that ensure proper propagation of data through sequential cells. Setup time is the duration for which input data must be stable before the triggering edge of clock. Hold time is the duration for which the synchronous input should be stable after the triggering edge of clock.