cadence®

Liberate™ MX Memory Characterization Reference Manual

Product Version LIBERATE 21.1
May 2021

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

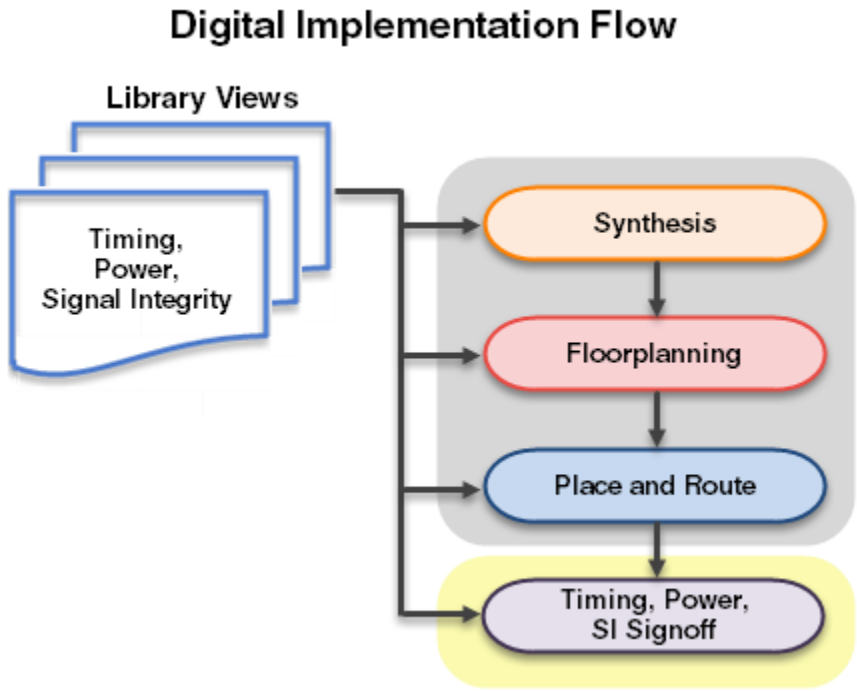
Contents

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.

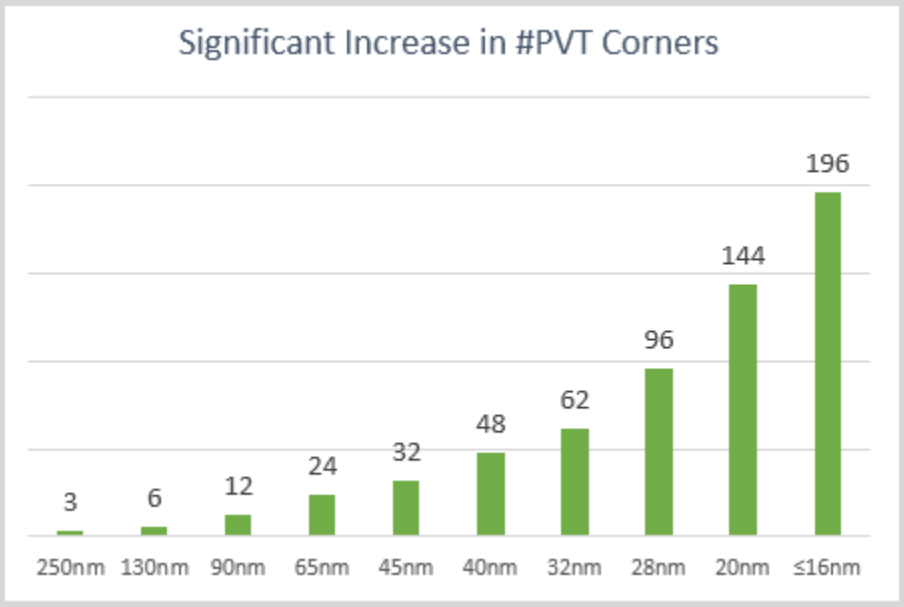


A Growing Problem

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:

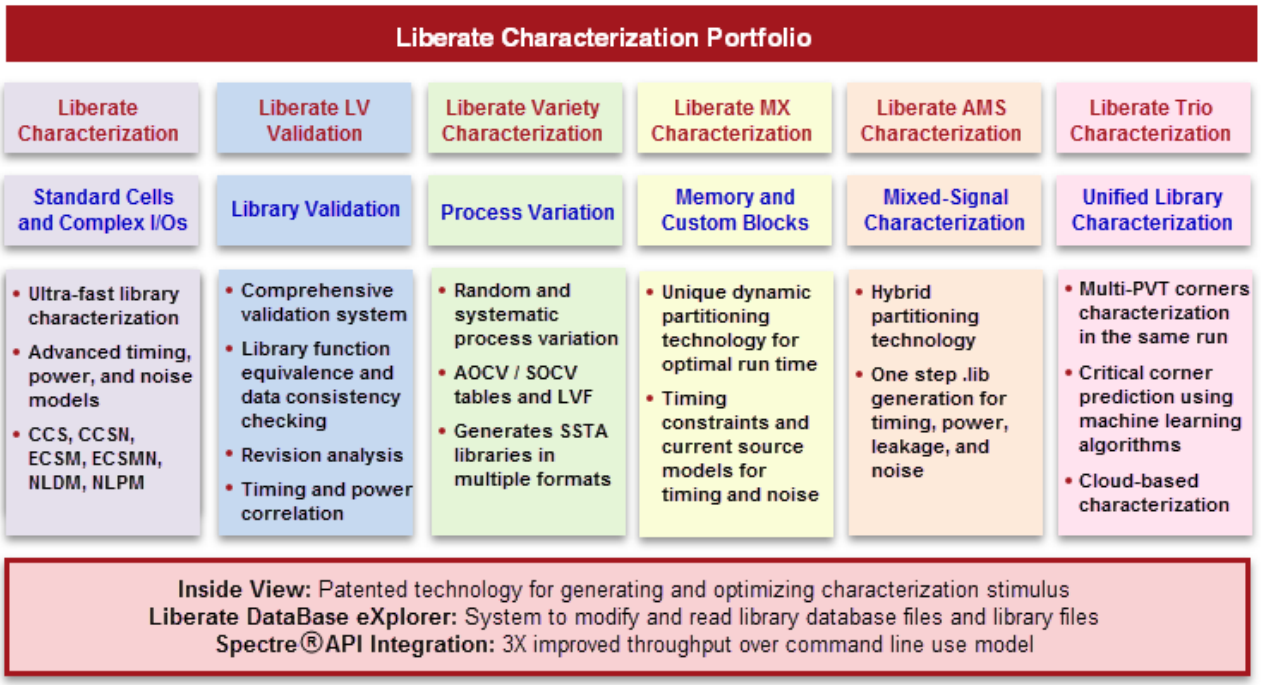


In addition, library views require more advanced models like:

- 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

Liberate Characterization Portfolio

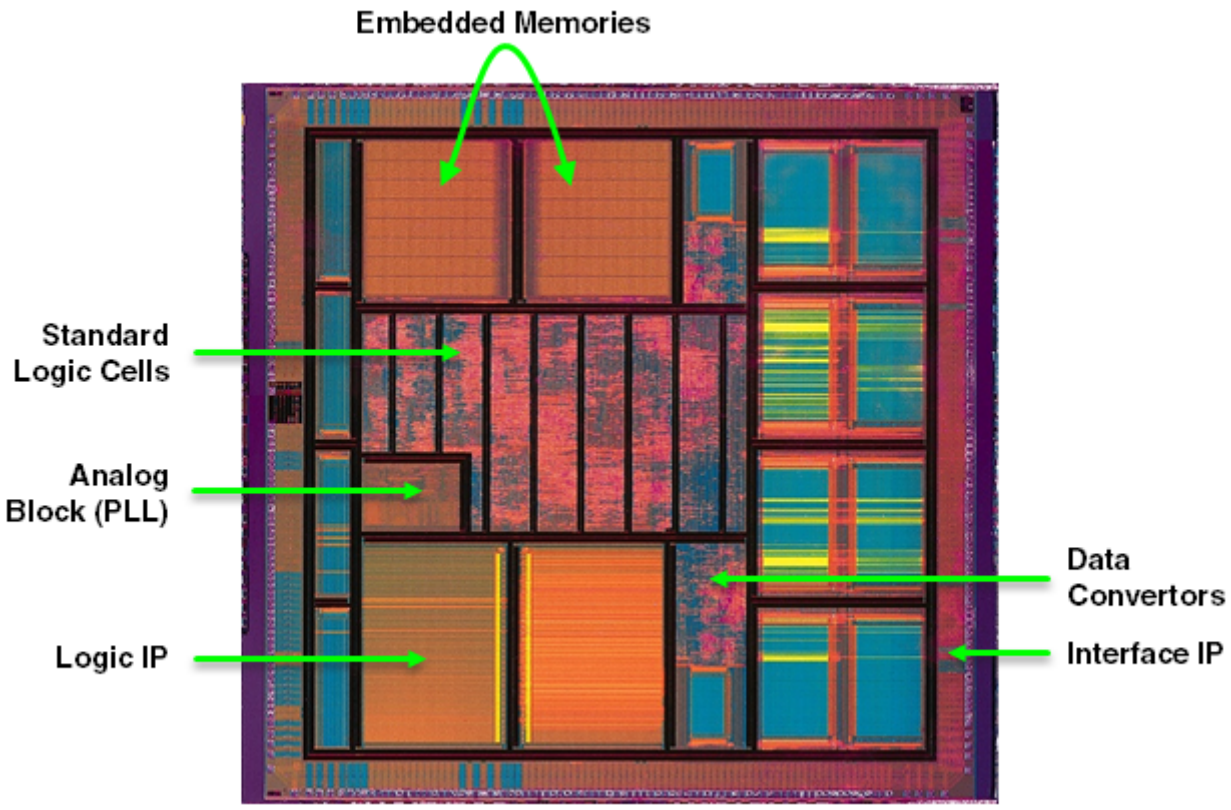
To address all the challenges, Cadence offers Liberate™ Characterization Portfolio that includes the complete set of characterization solutions given below:



The Liberate characterization portfolio intends to provide highly efficient and automated electrical view creation and validation for all IP blocks that including 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)



In addition, the Liberate DataBase eXplorer (Liberate DBX) system of the Liberate characterization portfolio lets you load and manage the contents of library database files (`.ldb`) and library files (`.lib`).

System and Licensing Requirements

Refer to *LIBERATE Software Licensing and Configuration Guide* for information about the different types of software licenses available to use the products of Liberate characterization portfolio. This guide also describes how to configure the licenses for efficient utilization of the available server and client resources.

For detailed information about the system requirements, see Computing Platforms.

About This Manual

The *Liberate MX Memory Characterization Reference Manual* describes the Cadence® Liberate™ MX Memory Characterization tool. The manual includes opening chapters that describe what Liberate does and how to get started with the tool. Later chapters discuss the commands and parameters that can be used with Liberate MX.

Audience Profile

This manual is aimed at developers and designers who want to create electrical views in industry standard formats such as Synopsys Liberty (.lib) format. 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 *Liberate Characterization Portfolio Known Problems and Solutions*.

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

For information about other products in Liberate characterization portfolio, refer to the following manuals:

- *Liberate Characterization Reference Manual* describes the Liberate characterization tool that creates electrical views (timing, power, and signal integrity) in formats such as the Synopsys Liberty (`.lib`) format. This manual also covers information about the Liberate Trio Characterization Suite.
- *Liberate LV Library Validation Reference Manual* describes the Liberate LV validation tool that provides a collection of capabilities used to validate and verify the data consistency, accuracy, and completeness of cell libraries.
- *Liberate Variety Statistical Characterization Reference Manual* describes Liberate Variety characterization 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.

- *Liberate AMS Mixed-Signal Characterization Reference Manual* describes Liberate AMS characterization tool that provides library creation capabilities for Analog Mixed Signal (AMS) macro blocks.
- *Liberate API Reference Manual* describes a Tcl interface that allows access to the Liberate characterized Library DataBase (LDB).
- *Liberate DataBase eXplorer 2.0 Reference Manual* describes the Liberate DBX 2.0 tool that can be used for enhanced post-processing of the data.

Rapid Adoption Kits

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

Typographic and Syntax Conventions

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

`literal`	Indicates text that you must type as presented in the manual. Typically used to denote command, parameter, routine, or argument names that must be typed literally.
`argument`	Indicates text that you must replace with an appropriate argument 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 parameters. 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 Memory Characterization

The Digital Implementation flow requires the characterization of the sub blocks of a full chip. 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 and retain arcs for input to output timing
  - Setup and hold arcs of input pins related to clock
  - Minimum period and minimum pulse width (MPW) for clock pins

- **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

Types of Memory to be Characterized

In some cases, the characterization of the memory instance is dependent on the design of the instance.

Memory instances can be divided into the following two categories:

- Those that are fully timed by the external clock
- Those that generate an internal clock to time the cycles

The internal cycle of the memory instance is timed to provide enough time for all operations to complete. The end of this internal cycle results from termination of the external clock or from completion of an internal delay loop.

These two memory types are referred to as externally-timed (only dependent on external clock) or self-timed (with a delay loop controlled internal clock).

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.

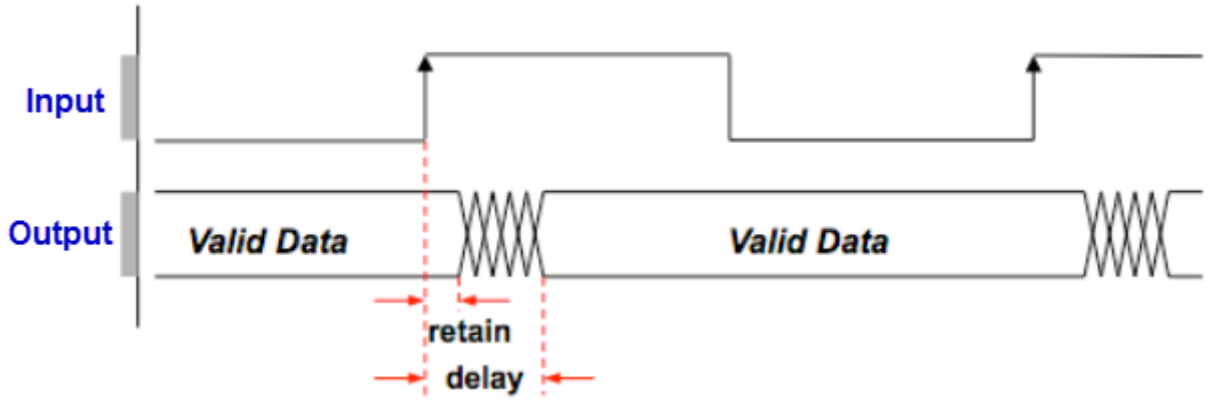
Timing Characterization

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

Delay and Retain Arcs Characterization

The delay arcs describe the timing relationship between the input and output pins. 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 retain arc describes the interval of time for which the previous data will be available after the input event. Retain is sometimes referred to as a minimum delay arc. The delay and retain arcs together describe the interval of time for which the output can be expected to be valid.



The use of delay and retain is especially important in a memory instance due to the range of times that the output event can occur. This is the result of multiple internal timing paths leading to the output and the bus nature of the output.

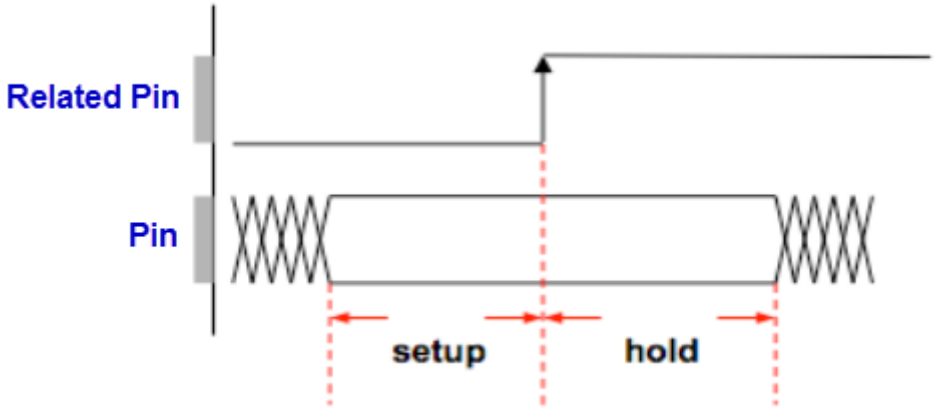
When creating input stimulus for a memory instance, it is important to capture the fastest paths and the slowest paths to ensure the correct data for both delay and retain. This means that for each output direction, rise and fall, there should be vectors that cover the following:

- Near and far output switching
- Near and far addressing
- All functional modes that can result in an output event such as read, write, bypass modes, and pipeline modes

The delay and retain 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.



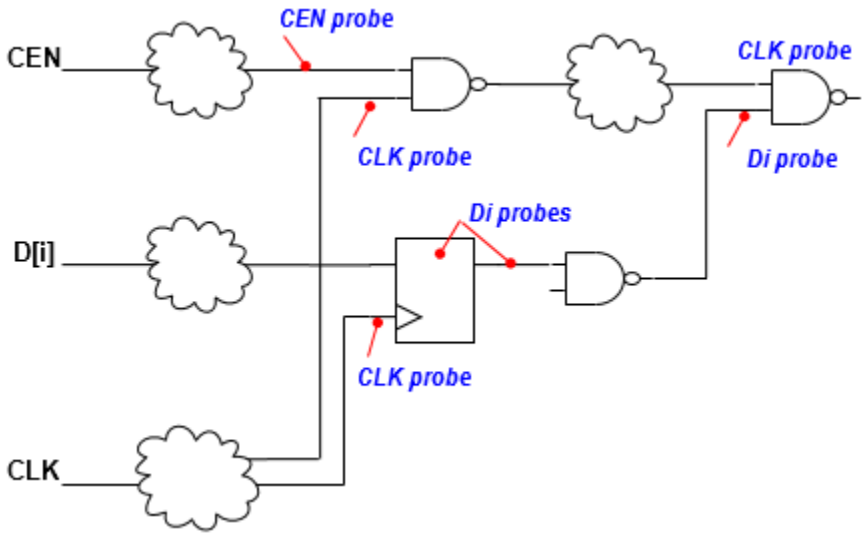
The setup and hold times need to be characterized for all values of pin and related pin input slew.

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 a memory 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.



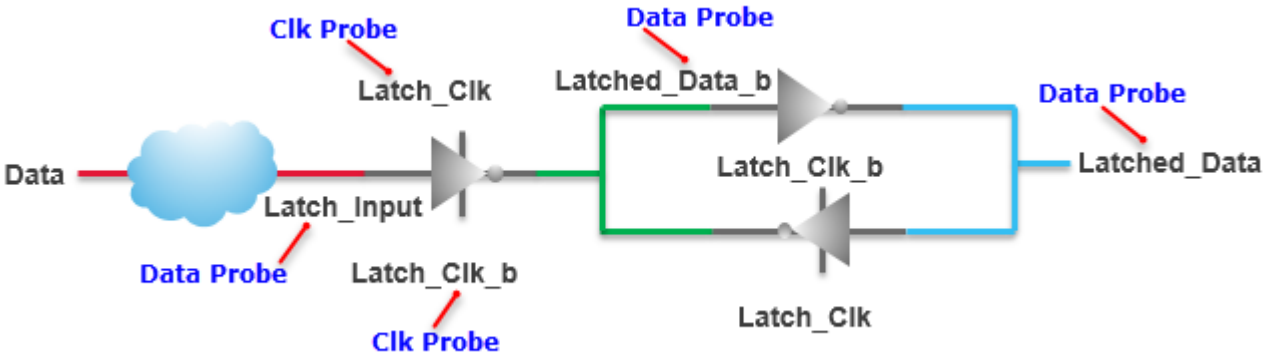
If the first stage is transparent during the situation that is being characterized for, then the following stage should be characterized as well. Typically, the first stage will be a latch and subsequent stages will be some form of combinational logic. In the case of setup time, the latch will be transparent 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.

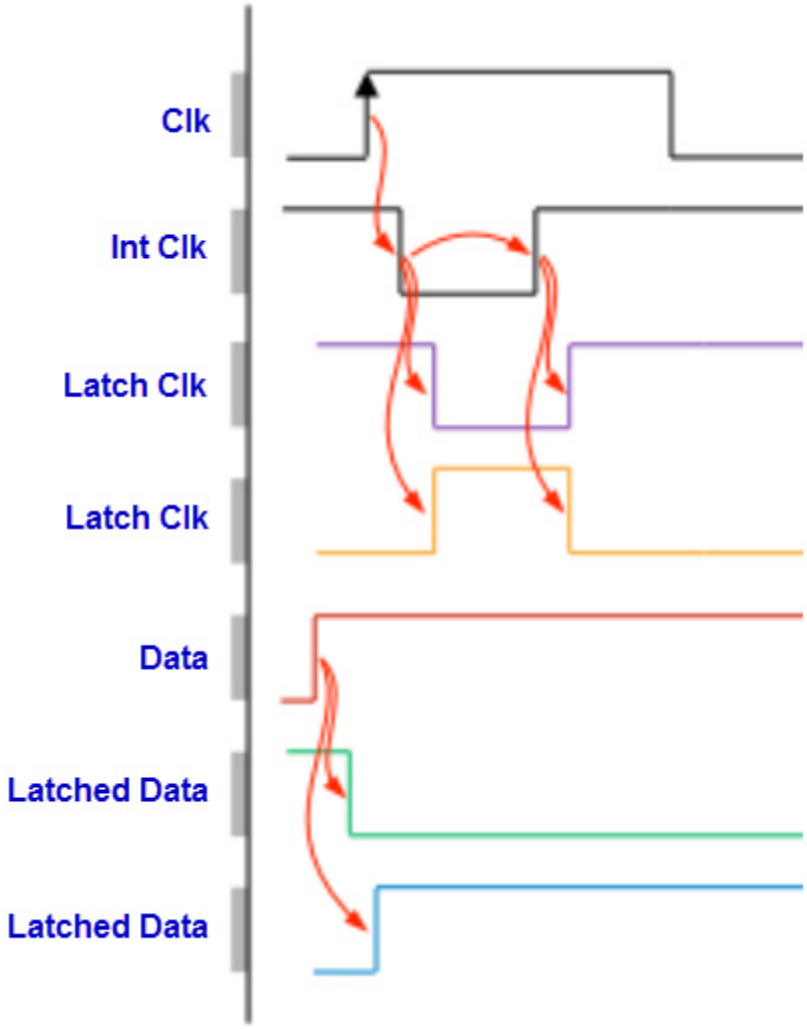
All valid probing locations should be measured and the worst case value should be captured in the Liberty file.

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 setup time, to maximize the data path delay, Liberate MX measures 50% data signal to 70% data probe signal (if probe is rising) or 30% (if probe is falling). To minimize the clock path delay, 50% Clk signal to 30% Clk probe signal (if probe is rising) or 70% (if probe is falling) is measured.

The setup probe threshold is controlled by the following Liberate MX parameters:

- `constraint_probe_lower_fall (constraint_probe_setup_lower_fall) (default: 0.3)`
- `constraint_probe_lower_rise (constraint_probe_setup_lower_rise) (default: 0.3)`
- `constraint_probe_upper_fall (constraint_probe_setup_upper_fall) (default: 0.7)`
- `constraint_probe_upper_rise (constraint_probe_setup_upper_rise) (default: 0.7)`

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) - Min (Data -> Latch_Input)
```

Similarly, for hold time, to maximize the clock path delay, Liberate MX measures 50% Clk signal to 70% Clk probe signal (if probe is rising) or 30% (if probe is falling). To minimize the Data path delay, 50% data signal to 30% data probe signal (if probe is rising) or 70% (if probe is falling) is measured.

The hold probe threshold is controlled by the following Liberate MX parameters:

- `constraint_probe_lower_fall (constraint_probe_hold_lower_fall) (default: 0.3)`
- `constraint_probe_lower_rise (constraint_probe_hold_lower_rise) (default: 0.3)`
- `constraint_probe_upper_fall (constraint_probe_hold_upper_fall) (default: 0.7)`
- `constraint_probe_upper_rise (constraint_probe_hold_upper_rise) (default: 0.7)`

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.

After the input latches, there can be other combinational logic where the internal clock is used to synchronize the signals that are used in the memory operation. These synchronized signals include wordlines that depend on the address value and bitlines that depend on inputs like address, data in, and write enable. These dependencies require that the internal clock and the input signals should be gated together. These locations where they meet require monitoring for setup time.

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 memory 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 cannot 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 memory 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 memory 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.

The characterization of the MPW will be different in case of the externally timed memory and the self-timed memory.

Minimum Pulse Width in the Externally-Timed Memory

As the externally timed memory has the external clock determining the start and end of the internal operations, it is necessary to ensure that those operations complete within the clock pulse.

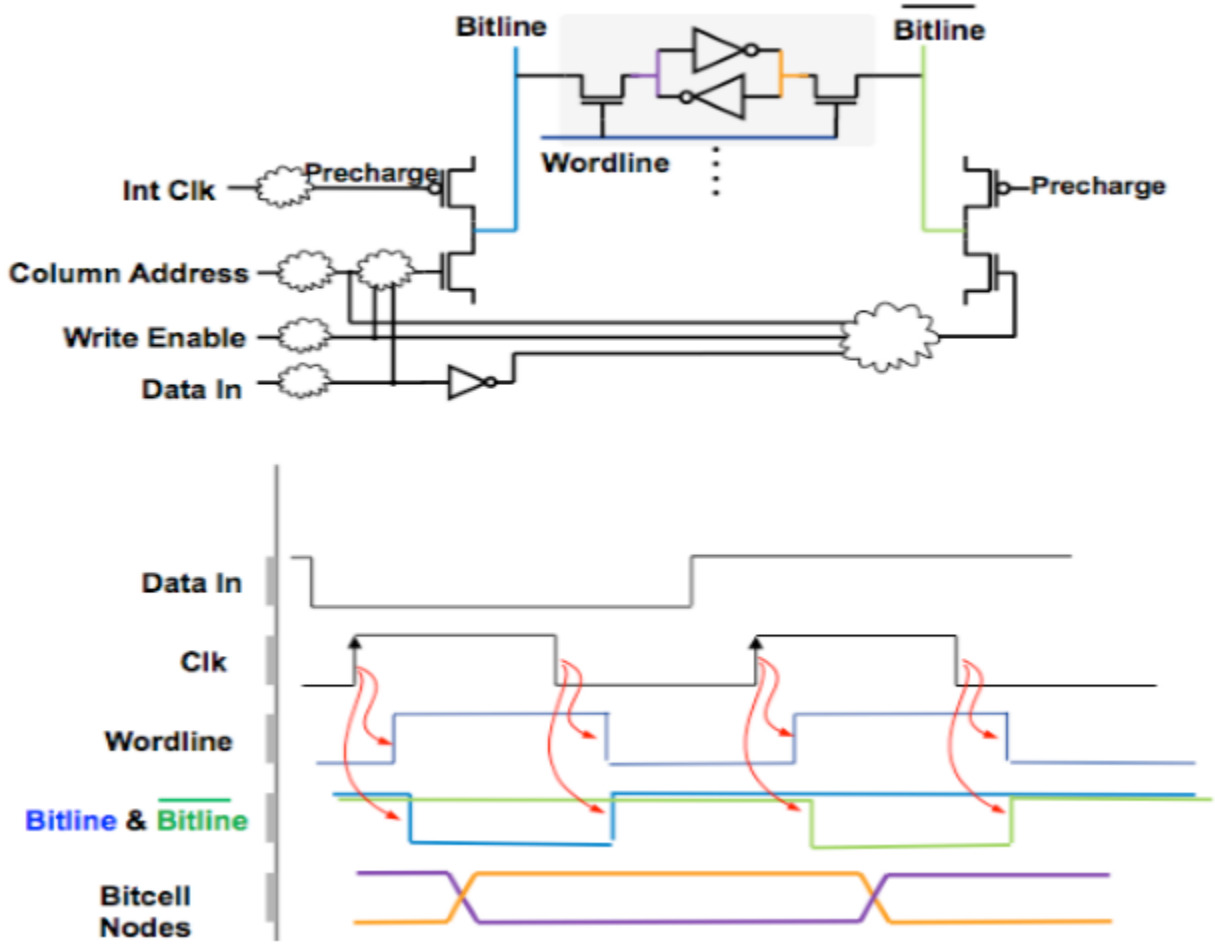
During the active portion (*usually high*) of the clock cycle, the following events must complete:

- Cell needs to reliably written for write cycles.
- Data must be reliably read for read cycles.

The inactive portion (*usually low*) of the clock cycle requires the following events to complete:

- Bitlines must be restored to their beginning of cycle state.
- Latches must open and propagate new values of input signals.

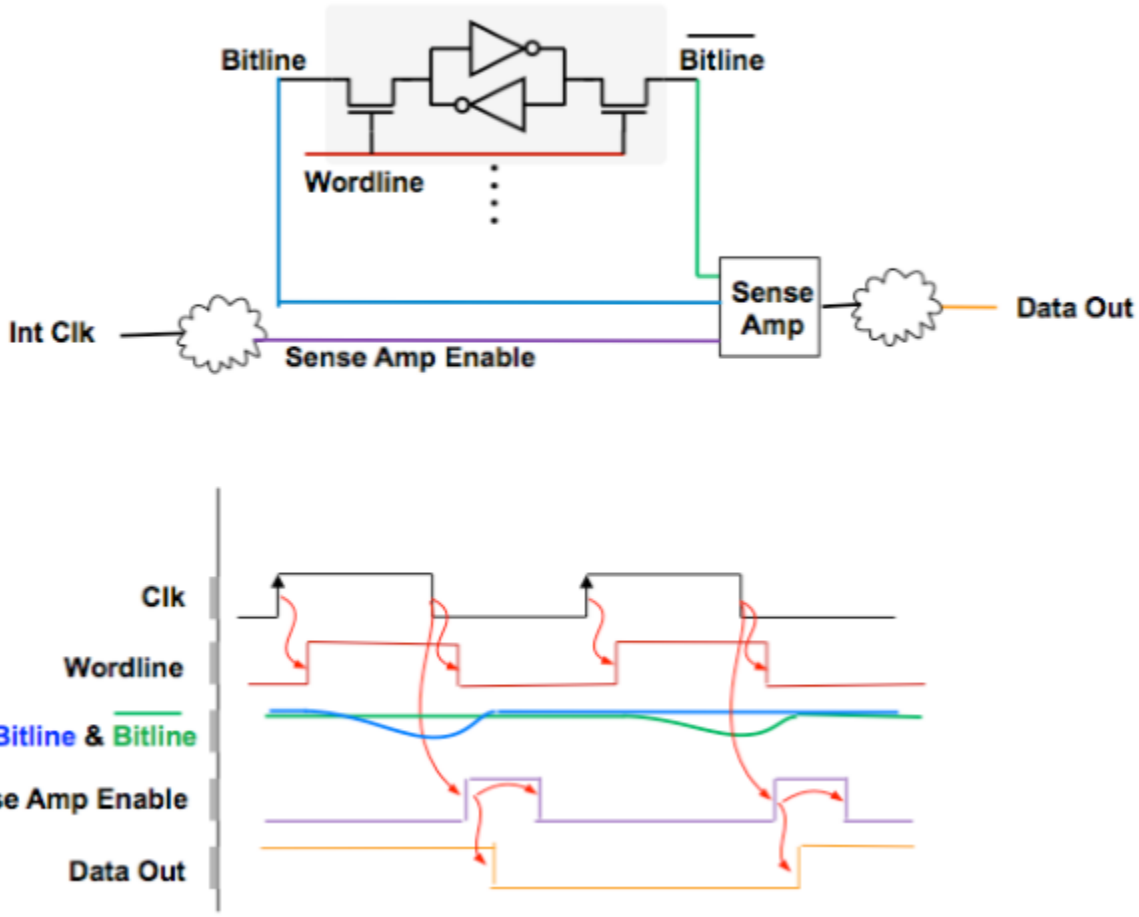
The following figure shows MPW during write:



In the figure above, the wordline and the bitline are controlled by an external clock. The wordline should stay high and the bitline should stay low, but long enough to reliably write data into the bitcell. As this is a critical operation, some margin should usually be added beyond the actual time to write the cell.

```
Minimum Pulse Width High = (Clk rise -> cell flip) - Min( ( Clk fall -> Bitline
rise) , ( Clk fall -> Wordline fall ) )
```

The following figure shows MPW during read:



In the figure above, the wordline and enabling of the sense amplifier are controlled by an external clock. To ensure a reliable read operation, the `Bitline` must sufficiently discharge before the `Wordline` falls and the `Sense Amp Enables` signal rises, as given in the equation below. The voltage requirement of the `Bitline` discharge will be design dependent.

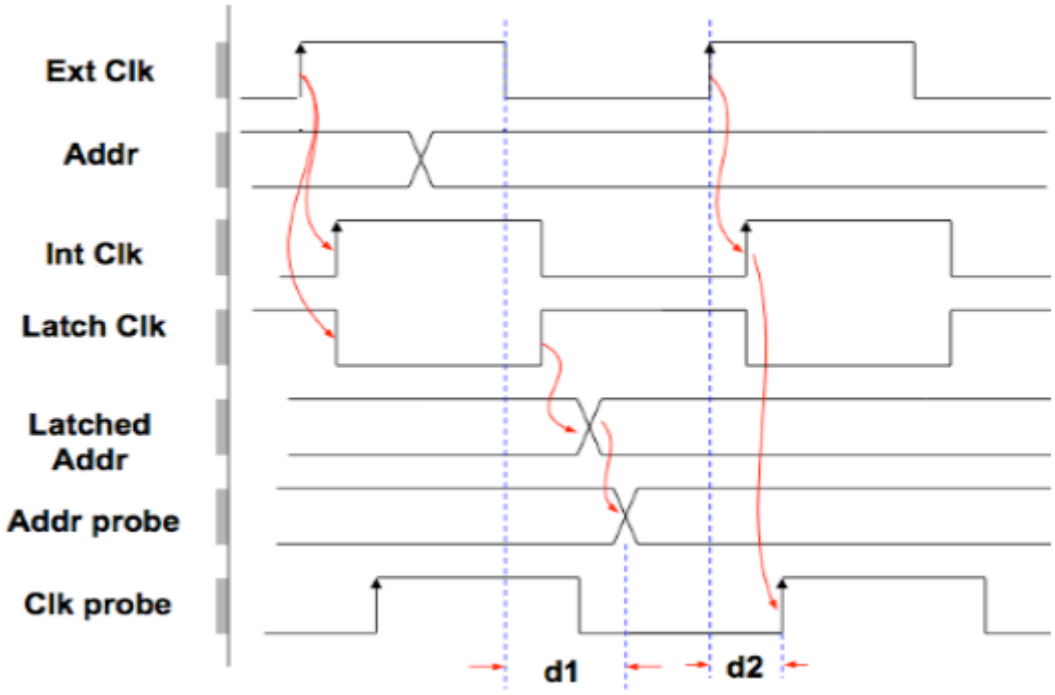
```
Minimum Pulse Width High = (Clk rise -> Bitline discharged) - Min( ( Clk fall ->
Sense Amp Enable rise) , ( Clk fall -> Wordline fall ) )
```

During the inactive portion of the cycle, it will be required that the bitlines return to their initial value before the next clock uses the bitlines. This should be measured for both the write and the read operation.

```
Minimum Pulse Width Low = (Clk fall -> Bitline returned to initial value) - ( Clk
rise -> Precharge rise )
```

The threshold used for the bitline value will be different from the threshold used for other internal signals and will usually be greater than 95%.

It will also be required that the latches open and propagate new data through before that data is needed by the clock of the next cycle. The probing locations for this component will be similar to the probing needed for the setup time of those signals. The difference will be that the probe will switch with clock instead of signal because it propagates as a result of clock opening the latch.



In the figure above, `Addr` probe must switch before `Clk` probe. The distance between these two events is dictated by the low pulse width (LPW) given to the `Ext Clk` signal, so that it determines the MPW.

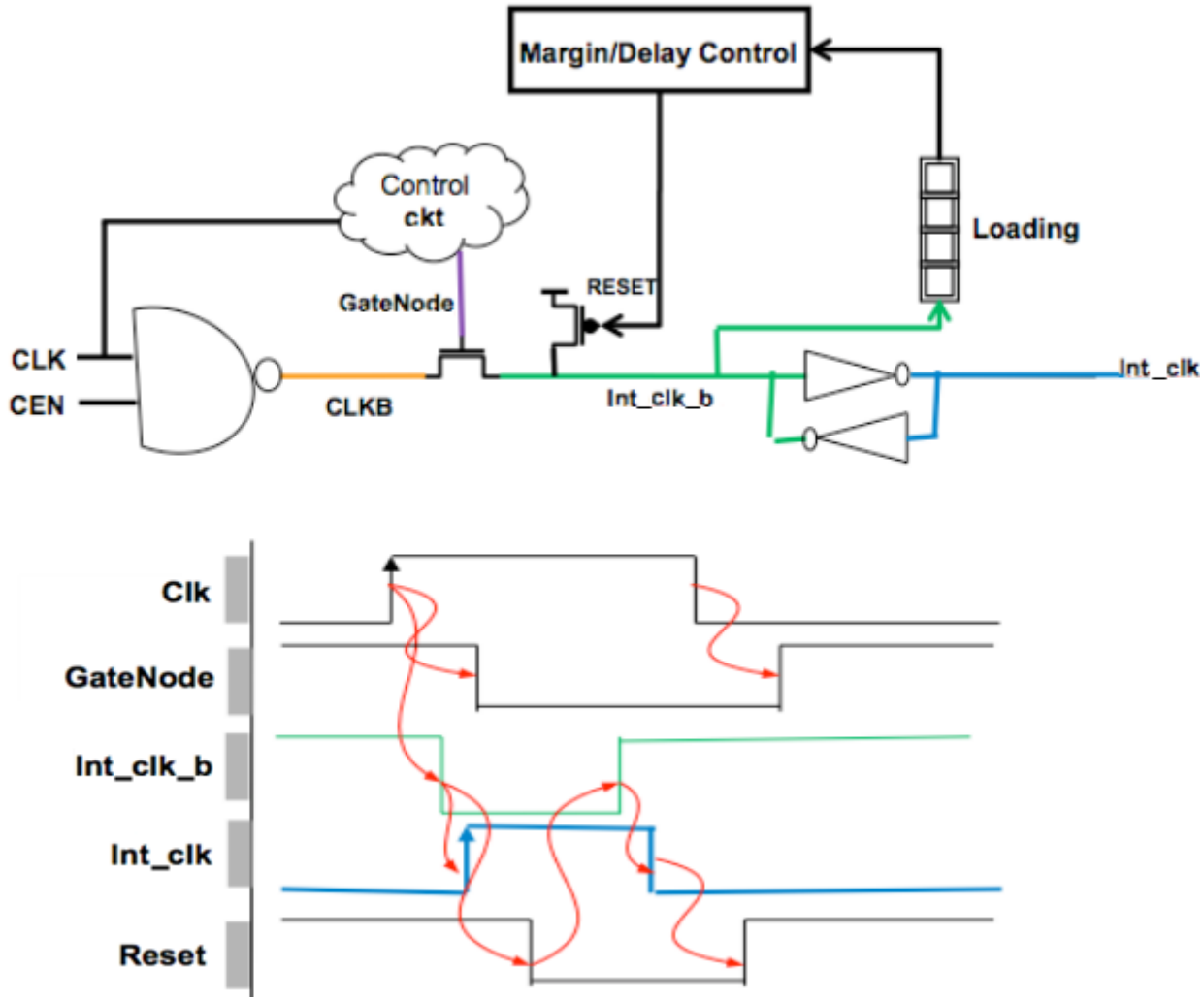
```
Minimum Pulse Width Low = (Clk fall -> Addr Probe) - ( Clk rise -> Clk probe rise )
```

The thresholds used for the internal nodes of MPW should also follow the same internal thresholds used for setup and hold characterization.

Minimum Pulse Width in the Self-Timed Memory

In the self-timed memory, the active edge of the clock is used to start the internal clock. Control over the internal clock then passes to the internal clock return. Once the internal clock returns, the internal clock terminates. The inactive edge of the external clock, combined with

the internal clock return, passes control of the internal clock generation back to the external clock for the next cycle.



In the example above, the external clock needs to stay high until `GateNode` goes low. If the external clock were to go low before `GateNode`, `Int_clk` would be corrupted.

```
Minimum Pulse Width High = (Clk rise -> GateNode fall) - (Clk fall -> ClkB rise)
```

At the end of the cycle, `GateNode` needs to be high before the next `Clk` rising makes `ClkB` fall. If this is not satisfied, the generation of `Int_clk` would be delayed.

```
Minimum Pulse Width High = (Clk fall -> GateNode rise) - (Clk rise -> ClkB fall)
```

As with other internal measurements, the appropriate thresholds should be used.

Minimum Period Arcs

The minimum period defines the time that is needed between two of the same direction edges of an input signal, usually clock.

The measuring of the minimum period is significantly different for the externally timed memory compared to the self-timed memory.

Minimum Period in the Externally-Timed Memory

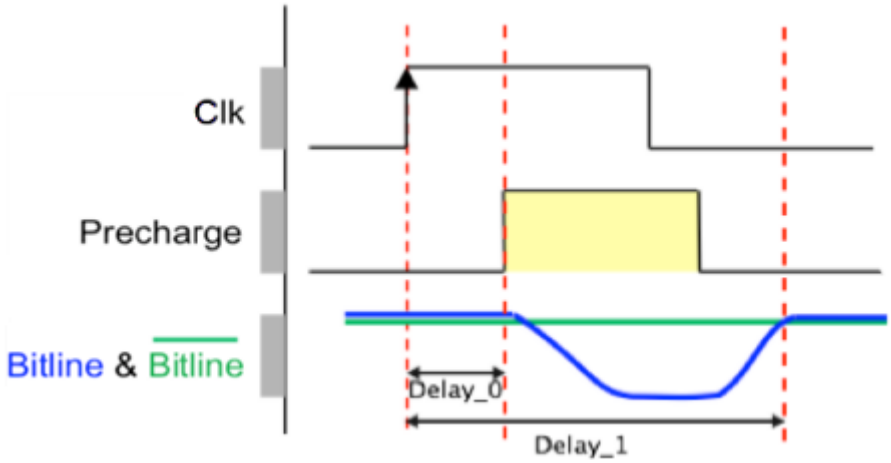
In the externally-timed memory, the two phases of a clock separately control the different portions of the active cycle. As there is no further dependence on the spacing between two like edges, the following equation is used:

```
Minimum Period = (Minimum Pulse Width High) + (Minimum Pulse Width Low)
```

Minimum Period in the Self-Timed Memory

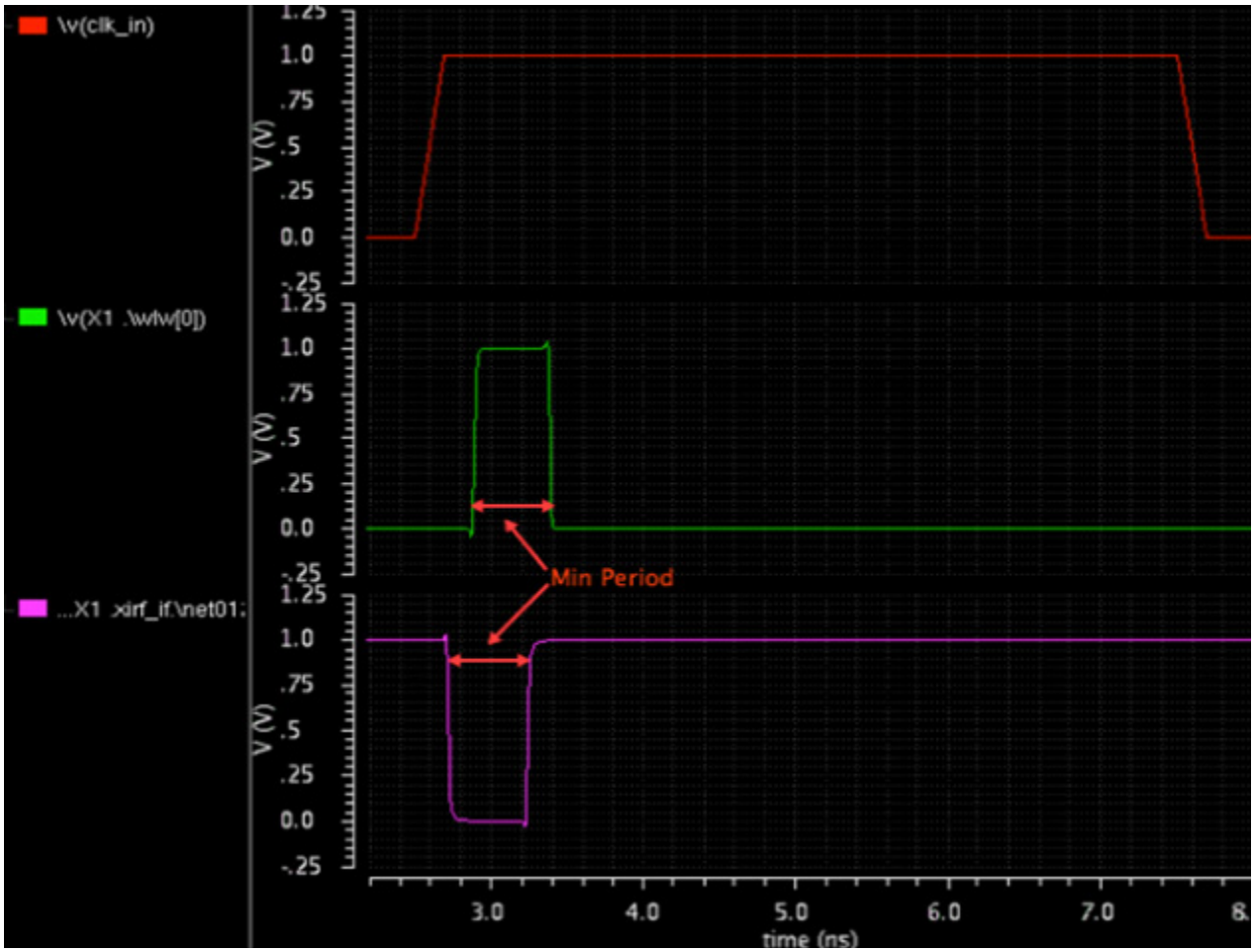
In the self-timed memory, the duration of the cycle is not determined by the values of the minimum pulse widths. These should be accounted for, but there are other requirements, listed below, for the clock cycle that need to also be satisfied.

- The sum of the MPW high and MPW low
- The bitlines must be returned to their initial state before the next cycle
- Any internal pulsing signals need to return to their initial state
- Latches must open in enough time before the next cycle to propagate the new input values



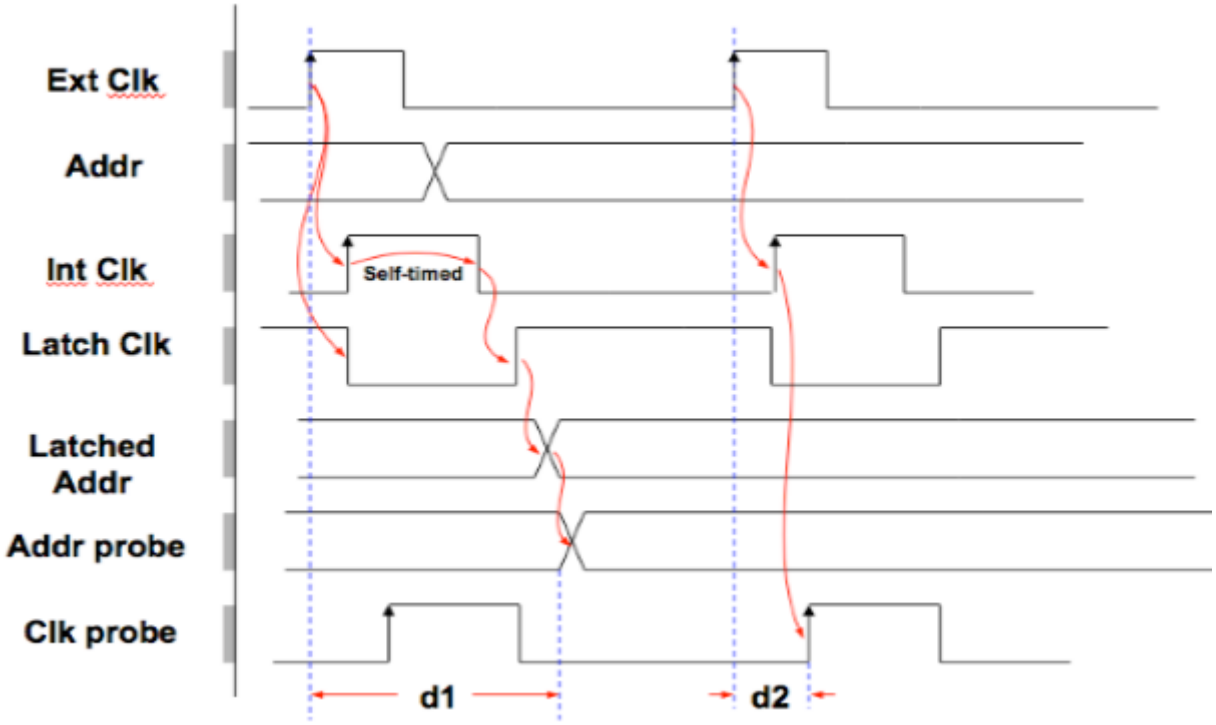
The bitline must return to its initial state (usually high) before the precharge turns off in the next cycle. This dictates the spacing between the two active edges of the clock. The threshold used for the bitline voltage will be higher than thresholds used for other internal signals and will usually be greater than 95%.

```
Minimum Period = (Clk rise -> Bitline high) - (Clk rise -> Precharge fall)
```



All internal pulsing signals must finish pulsing before they are activated again in the next cycle. This means that the minimum period must be greater than the width of any of these pulses.

The threshold used for this measurement is usually closer to the rails than the threshold used for setup and hold characterization.



The latch component of minimum period requires that the latches must open in enough time for the new data to propagate through the latch and arrive before the clock at the setup probing locations. If the external input switches while the latch is closed, the output of the latch switches with clock when the latch opens. This is the earliest the latch output can switch at the end of the cycle.

Basically, this means that even if the setup time of the input signal is satisfied, the latch needs to be open when the signal arrives there or it will be a violation. This is specified in minimum period to allow for enough time.

```
Minimum Period = (Ext Clk rise -> Addr probe) - ( Ext Clk rise -> Clk probe rise)
```

The thresholds that should be used here are the same as those used in setup time characterization.

Power Characterization

The purpose of the power characterization of an embedded memory 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 memory 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 the memory 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 the read and write operations separately. During the read operation, the output should not change to ensure that the output power is not counted. This is usually done by reading the same address location on consecutive cycles. For the write operation, there should be a predictable number of switching core cells, usually half. This is accomplished by writing a known pattern to the memory and then writing again to measure power.

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

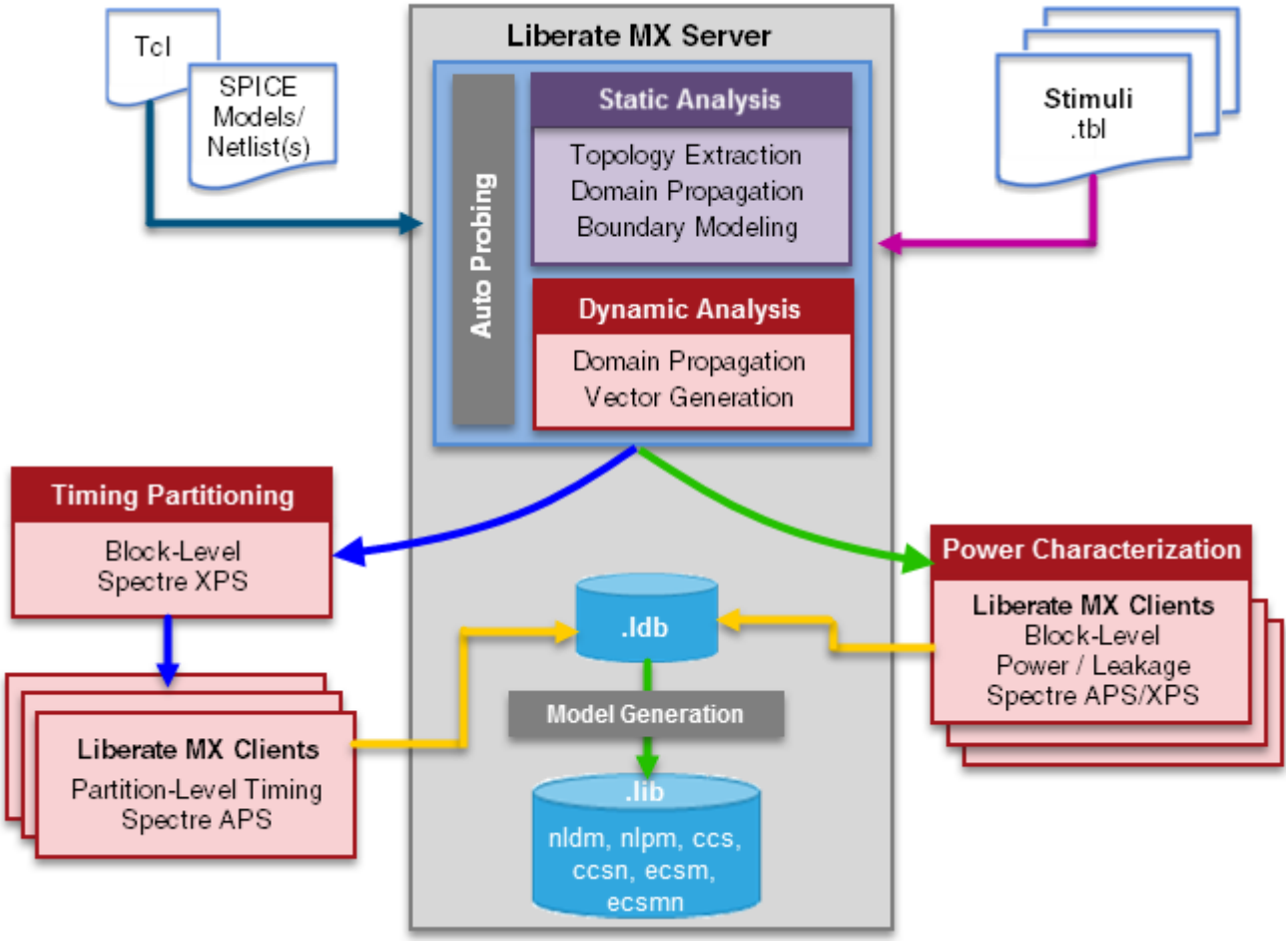
Overview of Liberate MX Memory Characterization

What is Liberate MX Memory Characterization?

Liberate MX provides library creation capabilities to cover the memory cores. Embedded and custom memories comprise a large percentage of silicon area on most chips and consequently, can often be major contributors to chip performance and power consumption. To validate a design's electrical performance, it is essential to have a highly accurate electrical model for each memory equivalent in accuracy of the electrical models used for standard cells and I/Os. Using pre-packaged models from an IP provider or memory compiler might not be sufficiently accurate, especially because the exact context of the memory is not known until it is placed on the chip. It is common, for example, to operate a memory block at a lower voltage to save power. To get an accurate electrical model that reflects the exact usage of the memory a design-specific, instance-specific characterization of each block is required.

Liberate MX implements additional techniques of spatial analysis, automatic probing, and temporal partitioning to make accurate characterization feasible. Using the divide and conquer algorithm, memories and cores circuits are reduced to manageable sizes and used by the unique Inside View technology of Liberate that optimizes the characterization run time. Consequently, the memories and cores are characterized with the same accuracy and techniques as standard cells. For detailed command and parameter description of standard cell library generation, refer to *Liberate Characterization Reference Manual*.

The completed libraries, which Liberate MX 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 memory instance.



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

The Liberate MX flow consists of two main steps: preprocessing and characterization.

In the preprocessing step, spatial partitioning techniques are used to identify the following basic building blocks:

- nand, inverter, latch, flop, arrays
- the clock trees
- internal probes of interest, such as latch/flop data

- clock nodes for constraint checks
- input and output nodes of combinational clock gates

User-provided stimuli—or truth tables—are then used to drive an activity-based temporal partitioning step to extract the transistor sets to be sent to characterization.

In the characterization step, partitions created from the first stage are characterized using a full SPICE simulator such as Spectre APS.

3

Getting Started with Liberate MX

This chapter describes briefly how to start using Liberate MX. 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.

Before using Liberate MX, ensure that it is installed correctly and that all the necessary prerequisite data is available. For information about the prerequisites, see the following manuals:

- *Cadence Installation Guide*
- *Cadence License Manager*
- *LIBERATE Software Licensing and Configuration Guide*

Tool Installation and Setup

Installing Liberate MX

To install Liberate MX:

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 MX software, see https://downloads.cadence.com.
3. Select the *LINUX* tab.
4. Select the appropriate release (for example, `LIBERATE151`).

   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, `LIBERATE151` is the first LIBERATE base release of 2015.
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 MX 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)
```

Important points to note:

- For Spectre X, it is recommended to use SPECTRE19.1 ISR9 or later.
- Liberate MX requires `LIBNG_807` to run Spectre X.
- Following is the recommended setting to use Spectre X:

```
set_var extsim_cmd_option "-64 +spice +preset=ax -
preset_override +liberate"
```

- Tune `preset=[cx|ax|mx|lx|vx]` to select accuracy mode for Spectre X. For more information about the accuracy modes, refer to the *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*.

System Libraries

Liberate MX is shipped with dynamic-linked system libraries. To verify Liberate MX is capable of running on your system, just try executing it. If Liberate MX fails to start properly, it may be possible that you have an old system and that there are missing or incorrect system libraries. If this occurs and you have already checked your environment setup is correct, you can try using static-linked binaries by setting the following environment variable:

```
setenv ALTOS_USE_STATIC_BINARIES 1
```

Inputs Required for Liberate MX Characterization

Various types of data are required to run Liberate MX:

1. Extracted memory block netlists in SPICE format.
2. Foundry device models in SPICE format.
3. A Liberate MX command file in Tcl format.
4. An input stimuli file.

The files required to get started with Liberate MX 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 memory block netlists)**

      The transistors, diodes, resistors, capacitors, and extracted parasitic elements (RCs) that compose the memory top-level netlists are passed to Liberate MX in SPICE format. Spectre and Berkeley SPICE netlist formats are supported. The information passed as files to Liberate MX 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 {sram.lpe}
      ```

   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 MX 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 A, “Truth Table Format.”

      Note: In automatic flow, truth tables are created by Liberate MX.

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 MX 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 MX are detailed in Chapter 5, “Liberate MX Commands.”

Tcl scripts are used to specify the memory instance 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 MX simulates and measures the MX 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 MX can also generate library information for the composite current source (CCS) model and the effective current source model (ECSM), for both timing and noise.

For an example of a setup Tcl file, see the Setting Up Manual Characterization Flow (Full Custom Flow) section in Liberate MX Flows.

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:

```
#%%%%%%%%%%%%%%%%%%%%%%%% template.tcl starts %%%%%%%%%%%%%%%%%%%
 ##################################################
# defining slews and measurement thresholds
##################################################
set_var slew_lower_rise 0.1
set_var slew_lower_fall 0.1
set_var slew_upper_rise 0.9
set_var slew_upper_fall 0.9

set_var measure_slew_lower_rise 0.3
set_var measure_slew_lower_fall 0.3
set_var measure_slew_upper_rise 0.7
set_var measure_slew_upper_fall 0.7

set_var delay_inp_rise 0.5
set_var delay_inp_fall 0.5
set_var delay_out_rise 0.5
set_var delay_out_fall 0.5

set_var max_transition 6e-09
set_var min_transition 2e-11
set_var min_output_cap 5e-15

set cells { \
    rf_top \
}

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

define_template -type constraint \
    -index_1 {0.16 0.5 1.6 } \
    -index_2 {0.16 0.5 1.6 } \
    constraint_template

define_template -type power \
```

```
    -index_1 {0.16 0.5 1.6 } \
    -index_2 {0.04 0.08 0.2 } \
    power_template

if {[ALAPI_active_cell "rf_top"]} {

################################################
# Top-level design definition
################################################
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 \
    -power power_template \
    -constraint constraint_template \
    rf_top

################################################
# Overriding pin-wise slew indexes (if needed)
################################################
define_index -pin {clk_in} -type power \
    -index_1 {0.2 0.3 0.4} \
    -index_2 {0.04 0.08 0.2 } \
    rf_top

################################################
# ARC definitions
################################################
#--------------
# Leakage
#--------------
define_leakage rf_top

#--------------
# Delay arc
#--------------
define_arc \
    -related_pin_dir R -pin_dir F  \
    -related_pin {clk_in} \
    -pin {data_out[31:0]} \
```

```
    rf_top

#--------------
# Retain arc
#--------------
define_arc \
    -type retain \
    -related_pin_dir R -pin_dir F  \
    -related_pin {clk_in} \
    -pin {data_out[31:0]} \
    rf_top

#--------------
# Setup arc
#--------------
define_arc \
    -type setup \
    -related_pin_dir R -pin_dir R  \
    -related_pin {clk_in} \
    -pin {add_in[6:0]} \
    rf_top

#--------------
# Hold arc
#--------------
define_arc \
    -type hold \
    -related_pin_dir R -pin_dir R  \
    -related_pin {clk_in} \
    -pin {add_in[6:0]} \
    rf_top

#--------------
# MPW arc
#--------------
define_arc \
    -type mpw \
    -pin_dir R  \
    -related_pin {clk_in} \
    -pin {clk_in} \
    rf_top
```

```
#--------------
# minimum_period arc
#--------------
define_arc \
    -type min_period \
    -pin_dir R  \
    -related_pin {clk_in} \
    -pin {clk_in} \
    rf_top

#--------------
# power arcs
#--------------
define_arc \
    -when "wr_in" \
    -type power \
    -pin_dir R  \
    -pin {clk_in} \
    rf_top

define_arc \
    -type power \
    -pin_dir R  \
    -pin {add_in[6:0]} \
    rf_top

define_arc \
    -type power \
    -pin_dir R  \
    -pin {data_out[31:0]} \
    rf_top
}
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% template.tcl ends %%%%%%%%%%%%%%%%%%%
```

Running Liberate MX

Before using Liberate MX, ensure that it is installed correctly and the necessary prerequisite data is available.

To use the 64-bit port of Liberate MX, set the `CDS_AUTO_64BIT` environment variable to `ALL` prior to running the tool, as shown below:

```
% setenv CDS_AUTO_64BIT ALL
```

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

```
% liberate_mx mx.tcl
```

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

```
% liberate_mx mx.tcl |& tee mx.log
```

Alternatively, to launch specific executables in 64-bit mode, you can set the `CDS_AUTO_64BIT` environment variable as shown below:

```
UNIX> setenv CDS_AUTO_64BIT liberate:spectre:liberate_mx
```

Invoking Liberate MX Help

Help content for Liberate MX commands and parameters can be viewed in two ways:

- **In the tool's command prompt**: Suffix the `-help` option to a command name to print a help message listing all the options it supports. Note that if you specify `-help` along with other supported options, only the help content is printed on the tool's command prompt, the other specified options are ignored.

  Note: Some options that get printed may not be officially supported. To view the list of supported options for each command, refer to the Chapter 5, "Liberate MX Commands."

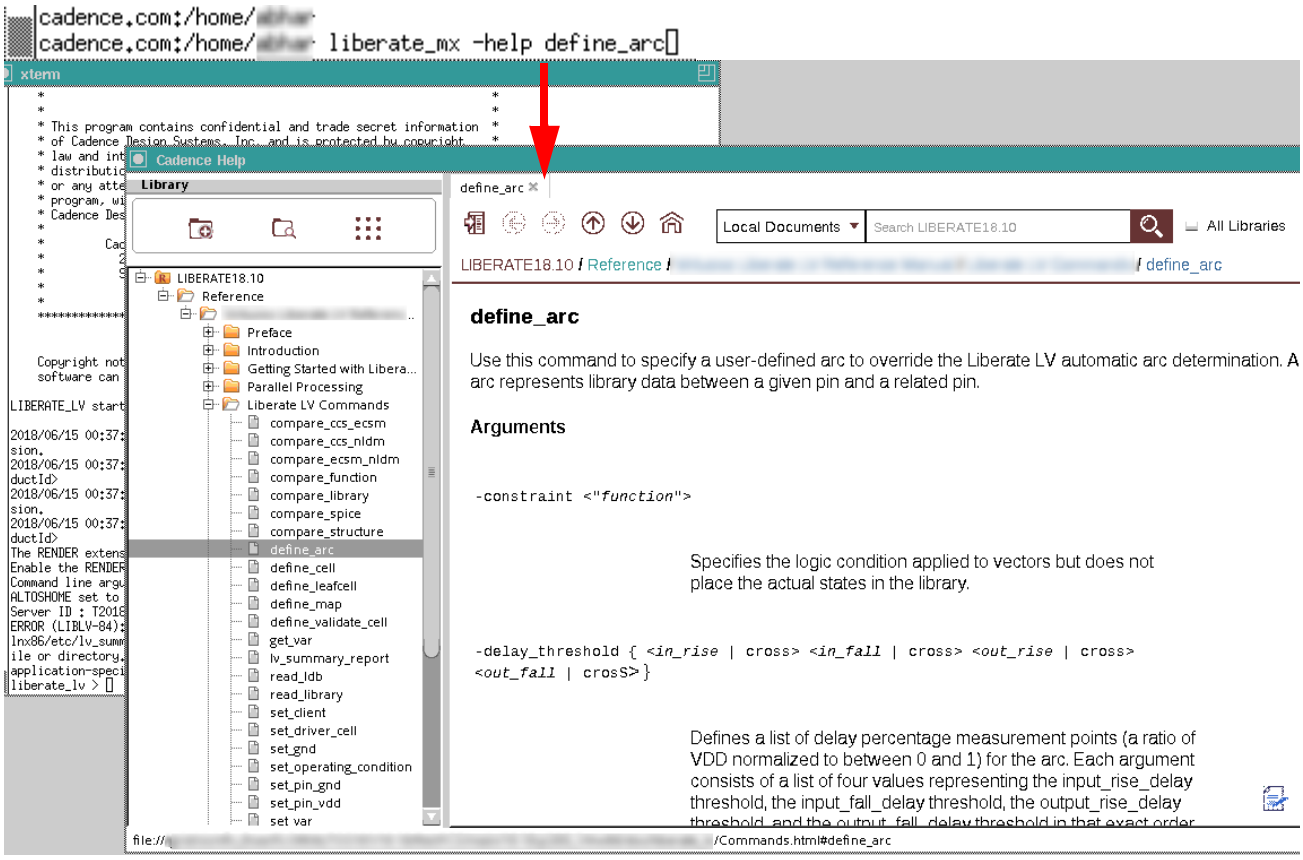
- **In Cadence Help**: On the UNIX command prompt, set the path to the Liberate MX installation directory. Then use the `-help` option with the command or parameter name to view the supported and formatted help content in Cadence Help. The supported syntax is the following:

  - `liberate_mx -help <`*`command_name`*` | `*`parameter_name`*`>`
  - `liberate_mx --help <`*`command_name`*` | `*`parameter_name`*`>`
  - `liberate_mx -h <`*`command_name`*` | `*`parameter_name`*`>`
  - `liberate_mx --h <`*`command_name`*` | `*`parameter_name`*`>`

  Once you are on the tool's command prompt, use the `help` command to view the required help content in Cadence Help. For example:

  ```
  liberate_mx> help define_arc
  ```

This will open the define_arc topic in Cadence Help.



Caution

When Cadence Help is displayed, the tool command prompt is invoked. Type `exit` in the tool command prompt to return to the UNIX prompt.

Searching Documents for Help Content

You can search for content in Cadence Help using the `-searchdoc` option.

Syntax for using `-searchdoc` in the tool executable command:

```
liberate -searchdoc <search_string>
```

Syntax for using `-searchdoc` in the tool's command prompt:

```
liberate_mx> help -searchdoc <search_string>
```

When either of the above commands is used, the document hierarchy is searched based on the specified search string and the search results are displayed in the Cadence Help window. This feature serves as an access point through the tool to Cadence Help and allows you to get the search results in one step.

Note: Use double quotes (`""`) to provide multiple search strings. For example,

```
help -searchdoc "static mode"
```

Reporting of Health Incidents

When a client health incident occurs, that is, if low memory or high CPU load is detected on the remote client machine, the tool will automatically write a report file with the output of the UNIX `top` command. This is useful for checking which user or process is using up the machine's resources.

The report file is named: `$output_dir/ldbs.gz/client_N_HOST_health.rpt`

where, `N` is the client ID and `HOST` is the host that the client is running on.

This file can also be found in the temporary run directory while the client is still running.

Following is the default command that is run when a health incident occurs:

```
"/bin/env HOME=$PWD /usr/bin/top -b -n 1"
```

The output of the `top` command can be customized by placing a file named `.toprc` in the working directory ($PWD).

If needed, the command itself can also be overridden by setting the `ALTOS_HEALTH_CMD` environment variable to a desired UNIX command.

This allows flexibility to run other commands or scripts that may produce more information about the health statistics of the client.

To disable writing of health reports, set the environment variable to an empty string as follows:

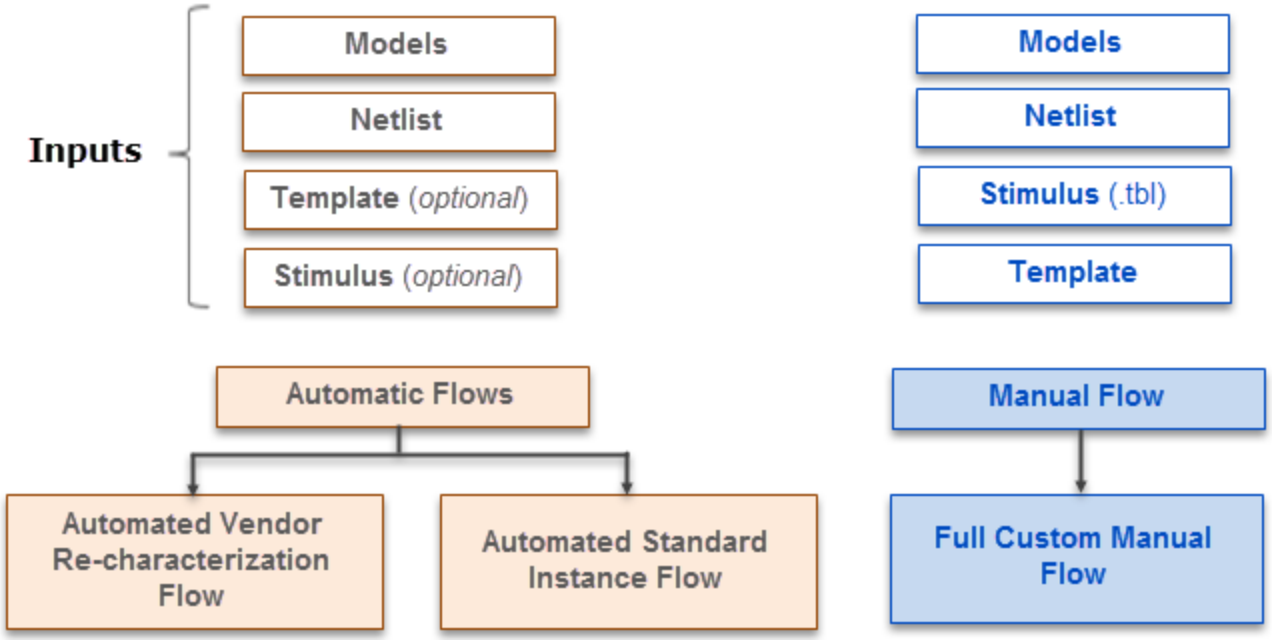
```
setenv ALTOS_HEALTH_CMD ""
```

Note: With default settings, the `top` command captures information about all users on a particular host and writes it to the report file. If this is not desired, you can disable this feature.

4

Liberate MX Flows

Liberate MX supports multiple use models for characterization that are classified as automatic flow and manual flow. The use model that is selected is determined by the instances that need to be characterized.



To validate whether a memory instance functions as per your requirements, Liberate MX provides the validation flow where you can simulate the memory instance and study the timing arcs. The validation flow can also be classified as automatic and manual.

This chapter discusses the following flows and solution and how Liberate MX functions in them:

- Characterization Flows
  - Automatic Characterization Flow
    - Automated Vendor Re-characterization Flow

Characterization Flows

This section covers information about the automatic and manual characterization flows available in Liberate MX.

Automatic Characterization Flow

The Liberate MX automatic flow (also called the define_memory flow) is intended to provide a simplified interface for characterizing standard instances and instances obtained from commercial IP vendors. There are certain criteria that an instance should satisfy in order to use the automatic flow. This section briefly discusses these requirements. For detailed usage and debugging information, refer to Appendix D, "Using Liberate MX Automatic Flow."

Automated Vendor Re-characterization Flow

In case of the vendor instances, the requirements are as follows:

- The instance should be from one of the main IP vendors (Virage, ARM, or TSMC)
- The design is known and supported by the flow. The designs that are supported by the flow are the standard available designs. These include single and dual port, SRAM, register files, and ROM designs.

When the automated vendor re-characterization flow is invoked, Liberate MX internally creates a custom setup base for the requested design requiring minimal input beyond the existing files generated from the compiler.

Automated Standard Instance Flow

For the non-vendor instances, you can describe the instance in terms of the functions of its pins. To ensure that the automated standard instance flow is an appropriate solution, the instance must satisfy the following criteria:

- The behavior of instance should be such that if the instance is enabled and in write mode, the contents of data in are written to the location specified by the address when the clock is activated.
- The behavior of instance should be such that if the instance is enabled and in read mode, the contents stored in the location specified by the address are reflected on data out when the clock is activated.
- Input signals are all latched by the clock signal
- Output signals are latched
- There are no disallowed combinations of address or data in. If the upper address locations do not exist, the location can be specified.
- The design is a SRAM or a ROM based design. This flow does not support CAM, CAMRAM, or flash.

Setting Up Automatic Flow

The Liberate MX automatic flow requires the following files:

- Tcl file containing the define_memory and char_memory commands
- Instance netlist
- SPICE or Spectre model files
- Reference library file or template file (*optional*)
- Tcl file containing any additional needed settings (*optional*)
- Additional table files (*optional*)

Primary Tcl File

The primary Tcl file of the automatic flow contains the `define_memory` and `char_memory` commands. The purpose of the `define_memory` command is to define the instance and the characterization conditions. The `char_memory` command runs the characterization.

The `define_memory` command is used for both vendor instance re-characterization and standard instance characterization.

- **Automated Vendor Re-characterization Flow**

  In case of automated vendor re-characterization flow, you need to use the `define_memory` flow to define the vendor, the instance netlist, and the Process,

Voltage, and Temperature (PVT) conditions. The pin function and rail definitions are not required because the naming conventions of the pins and rails are known.

```
define_memory \
    -ref_lib [pwd]/ref_lib/SRAM_2048x16.lib \
    -netlist [pwd]/netlist/SRAM_2048x16.spf \
    -vendor "TSMC" \
    -global_voltage 1.1 \
    -temp 25 \
    -models [pwd]/models/include_tt_model.sp \
    -mx_setting settings.tcl \
    SRAM_2048x16
```

Some common options available in the `define_memory` command to define the design that is being re-characterized are listed below:

`-vendor`	The IP vendor who designed the instance. Currently, the following IP vendors are supported: TSMC, ARM and Virage.
`-process_node`	The process node of the instance. This is used to identify process node-specific design characteristics for the instance.
`-cfg_file`	The configuration file generated by the compiler can sometimes be useful to define `process_node` and other design characteristics.
`-compiler_name`	The name of the generating compiler.

■ Automated Standard Instance Flow

In case of automated standard instance flow, you need to define the netlist, the pins and their functions, and the PVT information.

```
define_memory \
    -netlist SRAM_2048x16.spf \
    -ref_lib SRAM_2048x16.lib \
    -clock CLK \
    -address ADR \
    -data_in DIN \
    -data_out Q \
    -chip_enable {CEN L} \
    -write_enable {WEN L} \
    -rail {VDD 1.0 VSS 0} \
    -temp 25 \
    -foundry TSMC \
```

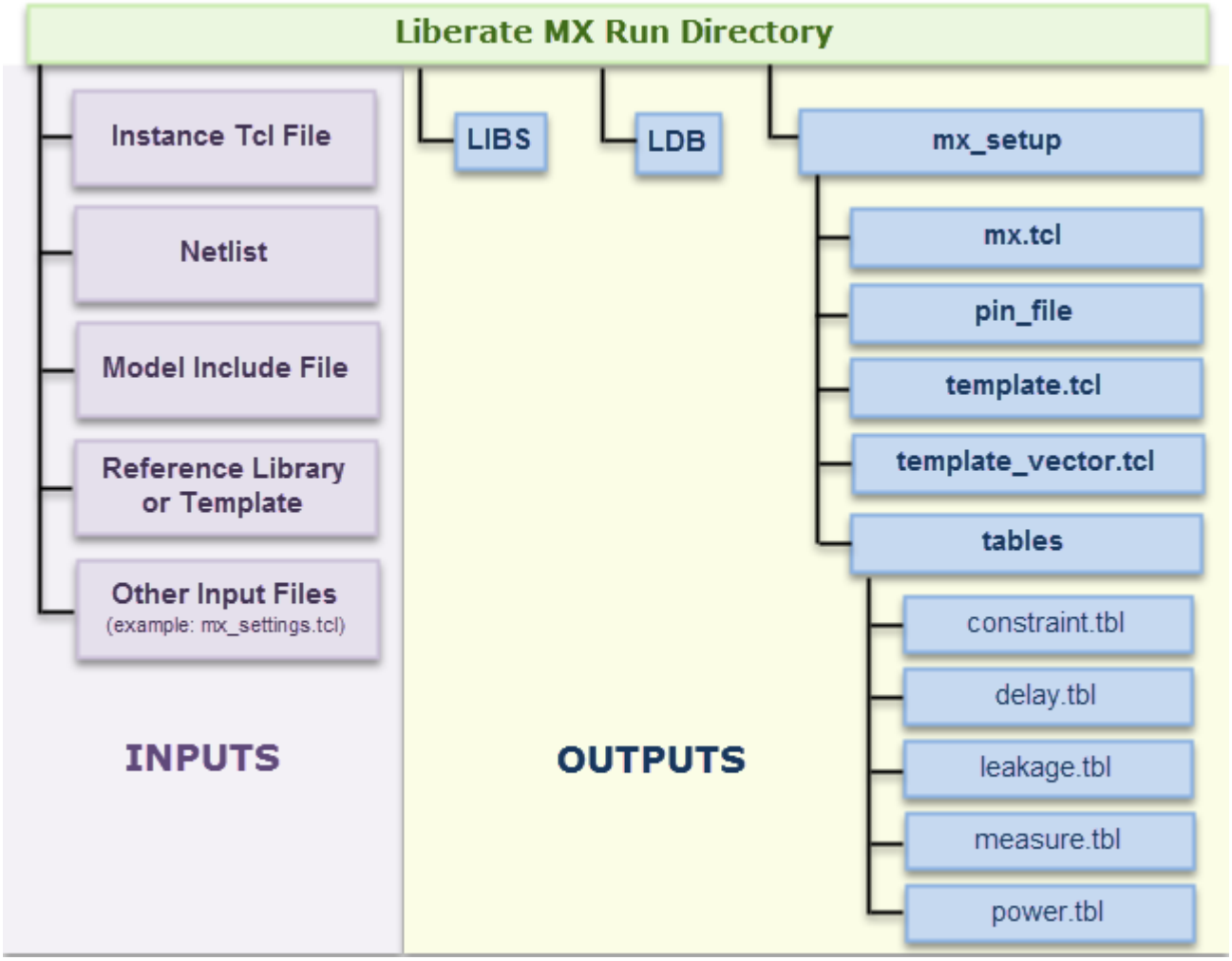
```
-mx_setting settings.tcl \
SRAM_2048x16
```

In both the automatic flows, the arc definitions can be passed in the form of the reference library using the `-ref_lib` option. In this case, Liberate MX characterizes the arcs as defined in the reference library file. If it is necessary to modify the arc definitions, it is best to use the `read_library` and `write_template` commands to create the template and modify it. The template can then be included using the `-template` option. If there is no template or reference library available, Liberate MX creates arcs and `when` conditions based on the pins described in `define_memory`.

The temperature will always need to be defined using the `-temp` option.

Additional Liberate MX commands and options can be specified in a Tcl `include` file. This file needs to be sourced immediately before you run characterization to override all previous definitions. Use the `-mx_setting` option to specify the Tcl file containing Liberate MX commands and options.

When you run the Liberate MX automatic flow, a directory structure as illustrated below is created with all of the files needed for the characterization:



Manual Characterization Flow

The Liberate MX manual characterization flow (also called the full custom flow) provides you the flexibility to customize the steps involved in characterization as per your own requirements.

Setting Up Manual Characterization Flow (Full Custom Flow)

In the full custom flow, all the settings, commands, and vectors are provided to stimulate the memory instance and configure Liberate MX. This data is provided in the form of table files for the vectors and `.tcl` control files to configure the Liberate MX parameters.

For setting the full custom flow, the required vectors, commands, and configuration settings should be provided to Liberate MX. The `mx.tcl` file is the main file in this flow. In this file, you specify the Liberate MX options and the path to the other needed files such as the template, netlist, models, and table files.

Note: The commands highlighted in **`bold`** typeface are mandatory or minimum requirements to get correct behavior from the tool.

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

# INPUT FILES
    set tcldir   ${datadir}/tcl
    set tmpldir  ${datadir}/tmpl
    set tbldir   ${datadir}/tbl
    set spicedir ${datadir}/spice

# SET CORNER
    set_operating_condition -temp 25 -voltage 1.2

# SPECIFY RAILS
    set_vdd VDD 1.2
    set_gnd VSS 0

# LEAFCELL
    define_leafcell -type nmos -pins {0 1 2 3} {xmn xmn_sram}
    define_leafcell -type pmos -pins {0 1 2 3} {xmp xmp_sram}

# READ IN SPICE
    set modelfile ${spicedir}/include_${process}
    set_var extsim_model_include $modelfile

    set netlistfile ${spicedir}/${cell}.spf
    set_var extsim_use_node_name 1
    set_var extsim_deck_include 1
```

```
    read_spice $netlistfile

# DEFINE MACRO
    source ${tmpldir}/template.tcl

# DESCRIBE FUNCTIONALITY
    set tbls [list ${tbldir}/delay.tbl ${tbldir}/const.tbl ${tbldir}/power.tbl
    ${tbldir}/leakage.tbl ${tbldir}/mpw.tbl ${tbldir}/minp.tbl ]
    define_table $tbls $cell

# SETUP PARTITION/PROBING/CHAR PARAMETERS
    set_var mx_dynamic_include_full_core 1
    set_var sim_init_condition "ic"
    set_var mx_remove_rc_timing "rail"
    set_var mx_remove_rc_pincap "rail"

# MAIN COMMAND
    set part_sim "xps"
    set char_sim "aps"
    char_macro -extsim [list $part_sim $char_sim] -ccs

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

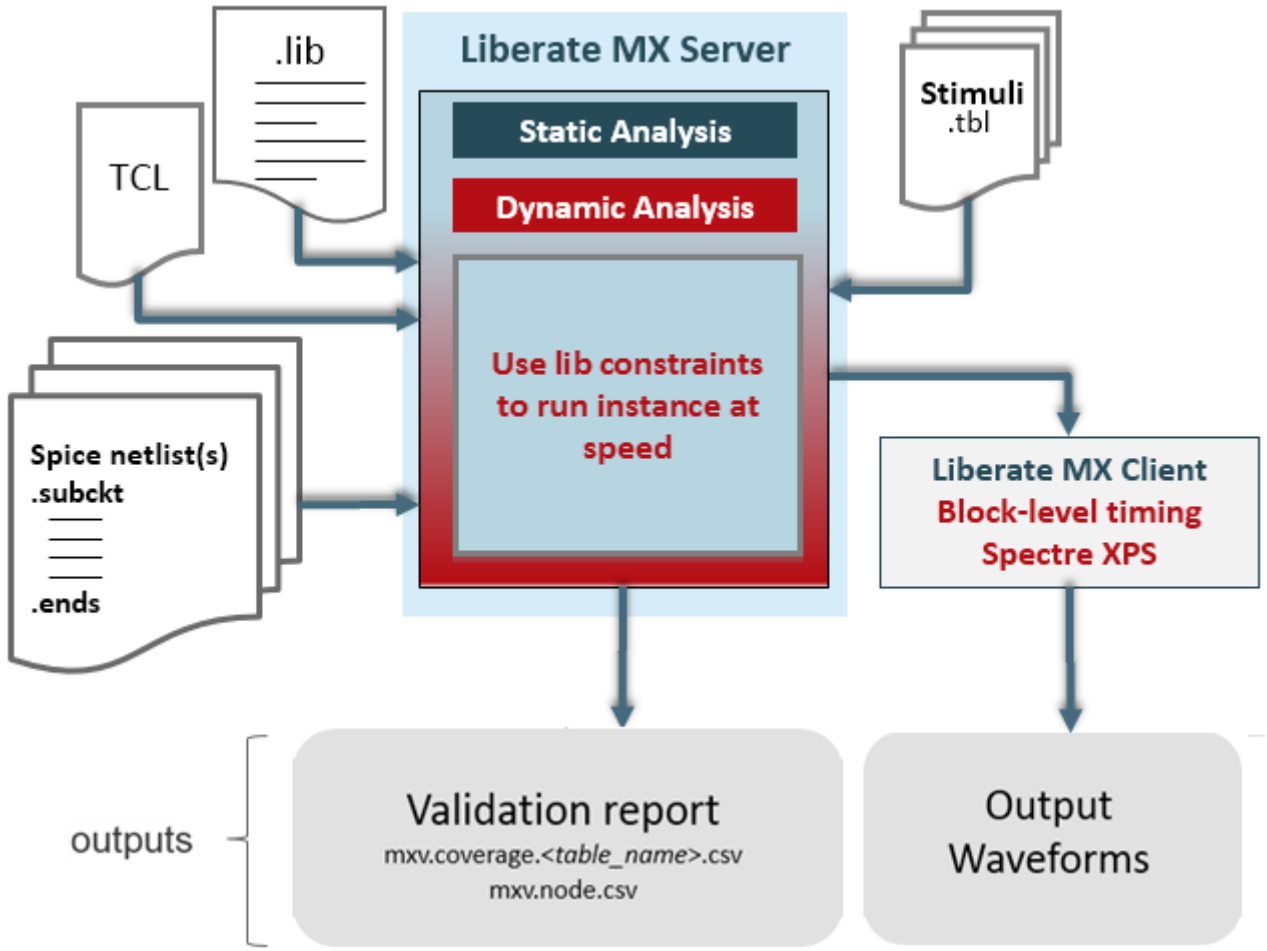
The netlist and model files are included by using the `read_spice` command. The successful reading of the netlist is always required, but the models only need to be read if the netlist contains MOS devices, as opposed to macro model devices. The models should always be specified using `extsim_model`.

The template file is included along with the `mx.tcl` file. This file contains the 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. If there is a reference library file, the template file can be automatically generated by using the `read_library` and `write_template` commands.

Validation Flow

Liberate MX validation flow demonstrates how the memory will function with different timing values specified in the library file. This is done by simulating the memory with minimum setup,

hold, and period. A passing result demonstrates the accuracy of the library file numbers. This is particularly useful to determine if a number is not sufficiently conservative.



The validation run is a top-level simulation and it is a single step process with no partitioning.

The validation flow runs two simulations per vector table. The first simulation is run with relaxed timing and is run to determine the correct functionality of the vectors. The second simulation is run as a stress with minimum timing. It is recommended to run the validation flow using a FastSPICE simulator such as Spectre XPS. Waveforms for both the simulations are provided to you. You can compare them by using the standard waveform viewers. For details, see Appendix E, “Liberate MX Timing Validation Flow.”

Most problems can be found by running a single slew. It is recommended to run a single slew first. This can be followed by minimum and maximum of other slews. Output load is not expected to have a significant effect for this test.

Setting Up Validation Flow

The Liberate MX validation flow is similar to running characterization with the following exceptions:

- The template file is written to include the timing values from the library file.
- The run command is validate_macro instead of char_macro.
- The tables are written to ensure that an arc failure will be observable.

The validation flow can also be setup using the automatic validation flow or the full custom manual validation flow.

- In automatic flow, the setup from conventional `define_memory` flow can be used as it is with the last command changed to `validate_macro` (that is, replace `char_macro` with `validate_macro`). Here, the flow automatically creates the validation tables.
- In full custom flow, you need to manually create the validation table and validation template.

Automatic Validation Flow

The automatic validation flow is supported in the `define_memory` flow when you do the following:

- Code the define_memory command as usual. For example:

```
define_memory \
-ref_lib instance_name.lib \
....
....
....
instance_name
```

- Add the validate argument to the validate_memory command.

By doing this, Liberate MX takes the library generated template for validation, creates validate tables, and runs the relax and stress runs.

Manual Validation Flow

This flow involves the steps discussed in the sections below:

- Writing the Validation Template
- Writing the Validation Table Files
- Determining the Completeness of the Tables

See also Appendix F, "Basic Flow for Validating User-Defined Criteria."

Writing the Validation Template

To provide the necessary timing information for the validation run, the validation flow uses the following command for template generation:

```
write_template -mx_validate <file_name>
```

The resulting template contains the timing information for the arcs:

```
define_arc \
    -type setup \
    -related_pin_dir R -pin_dir R  \
    -related_pin {CLK} \
    -pin {CEN} \
    -validation_values {0.219269 0.219272 0.219271 0.219274 0.219267 0.219268
    0.219268 0.229949 0.229953 0.229952 0.229954 0.229947 0.229948 0.229949
    0.244105 0.244108 0.244108 0.24411 0.244103 0.244104 0.244105 0.267319 0.267322
    0.267322 0.267324 0.267317 0.267318 0.267319 0.310398 0.310402 0.310401
    0.310404 0.310397 0.310398 0.310398 0.334074 0.334078 0.334077 0.33408 0.334073
    0.334074 0.334074 0.353111 0.353115 0.353114 0.353117 0.35311 0.353111 0.353111
    } \
    SRAM512x8
```

Writing the Validation Table Files

The tables for the validation flow should conform to the following criteria:

- Every arc should be exercised by the vectors. If an input transitions on the same line as the active edge of clock, it will be a setup stress. If an input transitions on the line after the active edge of clock, it will be a hold stress.
- The vectors should be written such that any single arc failure should lead to a difference in output.
- Input pins should have only one transition between clock active edges. This is done to ensure that there is enough time to satisfy setup, hold, and transitions within the clock period.
- Every signal should also contain a minimum pulse vector, where the setup and hold are both stressed around the same clock edge. This ensures that the latches are getting enough time to transition.
- It is recommended to use the relaxed simulation waveforms to verify the vectors before debugging the stress vectors.

Following is an example of the contents of a table file:

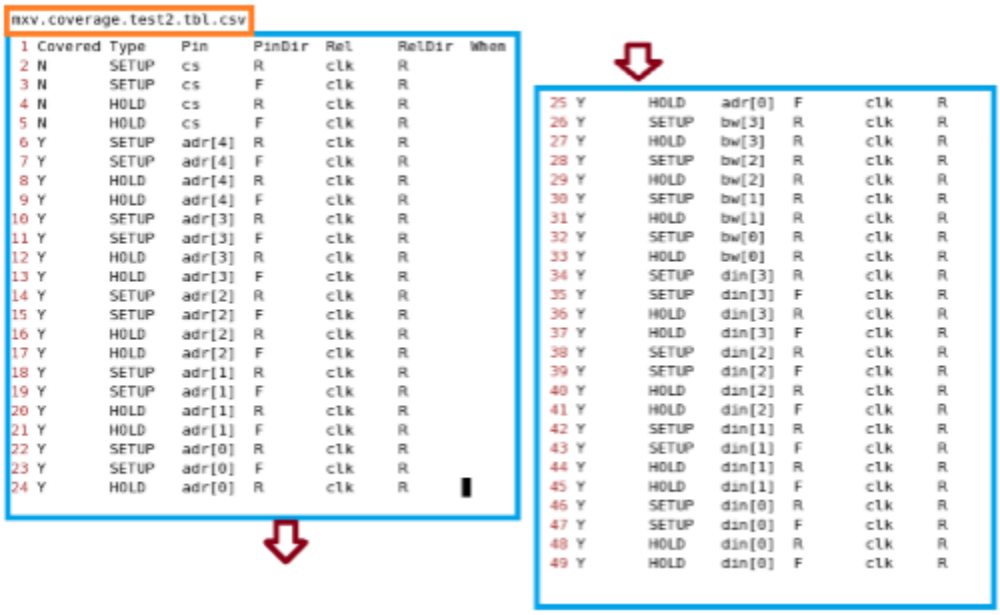
```
table write_read
pins     clk     addr      din   bw     cs        dout
W_a_A0   0        H        0x5   L      L         X
Clock1   1        L        0xa   H      H         D
disable  0        L        0x5   H      H         X
Clock2   1        L        0x5   H      L         D
W_5_A1   0        L        0x5   H      H         X
Clock3   1        H        0x5   H      H         D
disable  0        H        0xa   H      L         X
Clock4   1        H        0xa   H      L         D
R_a_A0   0        H        0xa   H      L         X
Clock5   1        L        0x5   L      H         D
endtable
```

Determining the Completeness of the Tables

To determine if all arcs have been tested, Liberate MX writes out a coverage report. This report is a summary of all arcs that existed in the template and whether they were tested by the tables.

The generated report is saved with a name of the following format:
`mxv.coverage.<table_name>.csv`

This file that has content presented in the following format can be read in Microsoft Excel:



Determining Whether the Run is Passing

There are multiple criteria that can be used to consider a run to be passing. Some criteria are given below:

- All outputs switch in the expected interval with no push-out.
- All internal race conditions should not be worsened by running 'at speed' test.
- There should be no internal illegal states as a result of running 'at speed' test.
- All critical signals should switch full rail.

Checking Switching Activity

The `mx_reuse` or `mx_fastsim` directory has the following two waveform directories for each table file:

- The relaxed timing simulation:

  *<inst_name>*_*<table_name>*_relax_*<number>*_
- The stress timing simulation:

  *<inst_name>*_*<table_name>*_stress_*<number>*_

These waveform files can be viewed in a standard waveform viewer. The user should compare the results of the stress run against the results of the corresponding relaxed run. The output delays in the stress run should be close to the delays in the relaxed run. Any differences should be investigated by the user.

Generating Node Comparison Report

A report can be generated to compare the activity of specified nodes between the relax run and the stress run. Use the `validate_node` and `report_node` commands to generate this comparison.

The first step is to run the `validate_node` command in the Tcl file, as shown below:

```
validate_node "bitline" instance_name
```

In this case, activities of all bitlines in the memory instance are compared. Other supported keywords are `wordline`, `core`, `bitline_precharger`, `senseamp_precharger`, `senseamp_enable`, and `output`. You can also specify a regular expression.

To report the node comparison, use the `report_node` command.

```
report_node -all "bitline" rf_top
```

By default, this command prints the values of only the failing comparisons. Using the `-all` option enables the printing of the passing and failing nodes.

These commands create a report named mxv.node.csv that can be opened in Microsoft Excel to view the node activity comparison, as shown below.

```
-------------------------------------------------
Table,  Node,   Type
-------------------------------------------------
test2.tbl,      xsingle_port_sram_ext.ZY/BLCR<0>:CL1,   bitline
,       Relaxed :,      H,      G,      G,      G,      G
,       At Speed:,      H,      G,      G,      G
,       Diff    :,       ,       ,       ,       ,      ^
test2.tbl,      xsingle_port_sram_ext.ZY/BLR<2>,        bitline
,       Relaxed :,      H,      G,      G,      G,      G,      G
,       At Speed:,      H,      F,      R,      G,      G
,       Diff    :,       ,      ^,      ^,       ,       ,      ^
test2.tbl,      xsingle_port_sram_ext.MZY/M0/xxRNB/I10/MI6:S,   bitline
,       Relaxed :,      H,      F,      R,      G,      F,      R,      G,      G
,       At Speed:,      H,      F,      G,      G,      G
,       Diff    :,       ,       ,      ^,       ,      ^,      ^,      ^,      ^
test2.tbl,      xsingle_port_sram_ext.ZY/BLCR<2>:CL1,   bitline
,       Relaxed :,      H,      G,      G,      G,      G,      G
,       At Speed:,      H,      G,      G,      G
,       Diff    :,       ,       ,       ,       ,      ^,      ^
test2.tbl,      xsingle_port_sram_ext.ZY/BLCL<0>:CL1,   bitline
,       Relaxed :,      H,      G,      G,      G,      G
,       At Speed:,      H,      G,      G,      G
,       Diff    :,       ,       ,       ,       ,      ^
test2.tbl,      xsingle_port_sram_ext.ZY/BLL<2>,        bitline
,       Relaxed :,      H,      G,      G,      G,      G,      G
,       At Speed:,      H,      F,      R,      G,      G
,       Diff    :,       ,      ^,      ^,       ,       ,      ^
test2.tbl,      xsingle_port_sram_ext.MZY/M1/xxLFB/I10/MI6:S,   bitline
,       Relaxed :,      H,      F,      R,      G,      F,      R,      G,      G
,       At Speed:,      H,      F,      G,      G,      G
```

Checking Internal Margins

Running the 'at speed' test can sometimes cause some internal signals to push out, affecting their relationship with other internal signals (setup time violation). These margins can also be compromised by the circuit not being fully reset at the end of the previous cycle (minimum period violation). Some examples of these margins that should be checked are:

- Write Margin

  In a typical design, the bitcell can write when the wordline is high and the bitline is low. The write window is defined as the time between the later of wordline rise and bitline fall and the earlier of wordline fall and bitline rise. This value should not be compromised by running the 'at speed' test.

- Read Margin

The read margin defines how much bitline differential has developed before the sense amplifier is enabled to amplify the difference in the read cycle. This is quantified by measuring the voltage difference between the two complimentary bitlines when the sense amplifier enabled signal becomes active. This value should not be compromised by running the simulation 'at speed' test.

Depending on the design, there might be other important margins that will need to be measured. It is important to measure these margins because it is possible that they might be reduced even if the correct output behavior is maintained.

Checking For Illegal States

There are several internal states within the memory instance that are illegal. These states usually are the result of the address decoding circuitry. An example of this can be the internal address lines that should have only one active line decoded. It is possible that under the 'at speed' test, multiple active lines overlap. These sort of conflicts might be temporary in nature, but they impact the robustness of the memory instance. To check this, all wordlines should be plotted and the user should ensure that only the intended wordlines go high during the clock cycle. Likewise, the column select lines at the column multiplexor should also be checked to ensure that only the intended lines are active on the cycle.

Checking For Full Rail Switching

All critical signals should switch full rail inside the memory instance to ensure robustness. There are two different causes for signals to not switch full rail in an 'at speed' test.

- The start of the internal signal is delayed at the beginning of the active cycle and the termination occurs before the signal can switch full rail. This situation is usually caused by setup time violations.
- The termination of the internal signal at the end of the active cycle does not go full rail before the next cycle starts and reactivates the line. This situation is usually caused by minimum period or minimum pulse width violations.

The signals that should be checked for full rail transition include the wordlines, bitlines during write, sense amp enable, and internal clocks.

Debugging the Failing Arcs

Once it is determined that the 'at speed' test failed, the next step is to determine which arc caused the failure. A circuit failure is usually a result of one or more arcs having incorrect values. There are two methods that can be used to determine such faulty arcs:

- Selectively increase the arc values to get a passing result.
- Debug the circuit to determine if a measurement location was not included in the original library characterization run.

Increasing Arc Values

To get a failing 'at speed' test to pass, one can employ the technique of selectively increasing the arc values. This is best done by modifying the values in the template file. It is best to increase one value at a time to isolate which arc is causing the failure. It can also be a combination of two arcs where either would pass by itself, but together they lead to a failure. It is recommended to increase minimum period first, followed by other arcs. Increasing other values before increasing minimum period can lead to illegal time sequences. After the minimum period has been increased, compare the failure traits with the expected effects of the arc failures to make an educated guess on which arcs might be causing the failure.

Using the Waveforms to Isolate the Failing Arc

If the user has some familiarity of the design, it is possible to use the waveform results to determine what part of the circuit is failing and therefore which arc does not have a correct value. The user can compare the circuit behavior between the relax waveforms and the stress waveforms. It is useful to first compare the failure against the expected failures based on the vectors to narrow down the expected location of the problem.

Using the User-defined Criteria

The validation flow supports user-defined measurement criteria. To do this, you can use the following commands:

Inputs: define_measure -> validate_measure -> report_measure

Results: `mxv.meas.csv`, `mxv.meas.rpt`

measure.tcl

```
define_measure \
-name valid_meas \
-trig clk \
-trig_dir rise \
-trig_val $vs50 \
-equations {"max(_valid_meas_A,_valid_meas_B)"} \
-keep min \
-duration 1.0 \
$cell

define_arc -measure valid_meas -pin clk -pin_dir R $cell
```

mx.tcl

```
validate_measure "valid_meas" $cell
validate_node "bitline" $cell
validate_macro -thread 1 -validsim $fastspice
report_measure -all "valid_meas" $cell
```

mxv.meas.csv

```
-------------------------------------------------------------------------------------------------------------
table_name,     measname,       line_no,        line_head,      relexs_time,    stress_time,    relax_value,  stress_value, diff_value, diff%
-------------------------------------------------------------------------------------------------------------
test2.tbl,      valid_meas,     20,     read_00,        1.700000e-07,   8.500000e-09,   2.2346e-09,           Failed,        -,        -%,
-----------------------------------------------------------
total number of difference: 0
-----------------------------------------------------------
```

Multiple Top-Level Simulations with Different Slew/Load

In Liberate MX, you can run multiple top-level simulations with different slew/load for worst path detection. This can be done by specifying multiple values for one or more fastsim_clock_slew, fastsim_input_slew and fastsim_load in the MX table file. All the three table directives should be specified. Use a single value if you do not want to sweep multiple top-level simulations for a specific table directive. `fastsim_clock_slew` and `fastsim_input_slew` should use the same number of variables. A top-level simulation is run for each combination of clock/input slew and load.

Example

```
fastsim_input_slew 0.16 4.0
fastsim_clock_slew 0.08 0.32
fastsim_load 0.02 0.2
```

This setting will run four top-level simulations with (input_slew, clock_slew, load) set to (0.16, 0.08, 0.02), (4.0, 0.32, 0.02), (0.16, 0.08, 0.2), and (4.0, 0.32, 0.2).

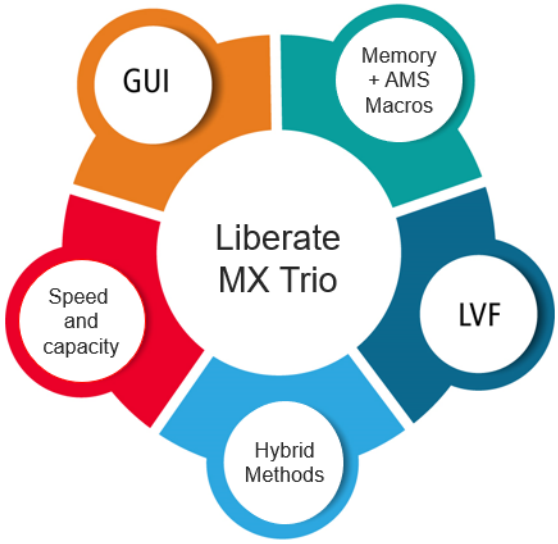
Bolt Job Distribution System

The Bolt job distribution system is a robust and highly scalable job distribution system that allows high scalability for utilizing thousands of CPUs in parallel on user networks and on cloud infrastructures. For detailed usage information, see the following:

- The Liberate MX Memory Characterization - Parallel Job Distribution application note.
- The Bolt Job Distribution System section in Liberate Characterization Reference Manual.

Liberate MX Trio Solution

Cadence® Liberate™ MX Trio, a unified characterization system, brings together features to generate and validate libraries of embedded memories and analog mixed-signal (AMS) blocks. It simplifies the challenge of dealing with an enormous collection of corners across multiple libraries for memory characterization.



The Liberate MX Trio solution brings together:

- Performance improvements with the ***multi-PVT characterization*** flow where multiple corners can be characterized in parallel.
- ***Unification of statistical and nominal modeling*** in a single characterization flow that generates Liberty Variation Format (LVF) and nominal libraries. The solution offers an accurate LVF capability for memory and analog mixed-signal macros.

- ***Real-time progress monitoring*** for each characterization job on a compute cluster through a graphical user interface (GUI) that shows status and run time for all PVT corners, cells, arcs, and data types, on every cloud machine.
- **Unique ease-of-use** for mixed-signal characterization as the solution provides access to Virtuoso integration using Liberate AMS.

Simulator-level multi-threading is not supported and recommended in Liberate MX Trio. The Liberate characterization tool manage CPUs better when it is controlled by clients (packet_clients) and thread (char_macro/char_ams -thread).

Liberate MX Trio uses a simplified flexible token-based licensing scheme. See the *Liberate MX Trio (ALT940) Licensing Requirements* section in the *LIBERATE Software Licensing and Configuration Guide* for information on licensing.

5

Liberate MX Commands

This chapter describes the Tcl commands that control memory library creation.

Note: The command arguments that are prefixed with a hyphen (`-`) are optional except where explicitly indicated. All commands have a `-help` option.

To access officially supported context-sensitive help information on a command or a parameter from within the tool, follow the procedure covered in the Invoking Liberate MX Help.

To review tool-wise support information about each command and parameter available in the Liberate characterization portfolio, see *Liberate Characterization Portfolio Command and Parameter Support Matrix*.

g...	
get_var_default	
h...	
help	hspice_lis_2_waves
i...	
interpolate_define_axis_expr	interpolate_set_default_group
interpolate_define_axis_expr	interpolate_use_worst_value
m...	
mx_cmdlog	mx_recover_setup
mxcore_def	mx_report
mx_match_node	mx_report_supply_current
mx_merge	mx_set_constprop
mx_rampup_check	mx_set_domainprop
mx_recover_clean	mx_set_spectre_param
mx_recover_info	mx_utils_sort_mx_info
mx_recover_merge	mx_set_ultrasim_param
r...	
read_spice	report_node
report_measure	
s...	
set_gnd	set_vdd
set_pin_gnd	set_virtual
set_pin_vdd	spv_tcl_2_waves
set_var	
u...	
unset_var	
v...	
validate_macro	validate_memory
validate_measure	validate_node

w...	
write_ldb	write_verilog
write_library	

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 index_1 points. Default: all points.
`-index_2 {list}`	Specifies index_2 points. Default: all points.

`-pin {list}`	List of pins. `-pins` accepts the bus syntax, such as `{ A[7:0] }`. This option also accepts the asterisk "`*`" wildcard character, for example:
	`add_margin -pin addr* #All pins beginning with "addr", such as addr[3]`
	`add_margin -pin * # All pins`

`-related {list}`	List of related pins. `-related` accepts the bus syntax, such as `{ 0[7:0] }`. This option also accepts the asterisk "`*`" wildcard character.
`-rel <value>`	Specifies the relative amount of margin to add. Default 0.0 (0%). Example: 0.05 = 5% margin. **Note:** This option always makes the library value larger, whether the original value was positive or negative.

`-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. Valid types include: `cap`, `constraint`, `delay`, `delay_ccs` *(only accepts positive "abs" margin)*, `hidden`, `hold`, `leakage`, `mpw` or `minimum_period`, `nochange`, `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_macro`, `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.

Examples:

```
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_macro

Performs memory characterization. Each memory listed in a `define_cell` command will be characterized if the SPICE `subckt` definition for that memory 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`.

Note: This option is for backward compatibility for setups created prior to Liberate 13.1 release.

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

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

`aps`	Use the Virtuoso Accelerated Parallel Simulator (APS).
`spectre`	Use the Spectre simulator.
`xps`	Use the 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` Characterizes ECSM (delay) data.

`-lvf` Enables Liberty Variation Format (LVF) generation from the Liberate MX run.

`-lvf_method [sens | vvo | mc]`

Enables LVF generation.

Note: `-lvf_method` must be set with `-lvf`.

`sens` Enables sensitivity based methodologies (RSS|UM).

`vvo` (Default) Enables Spectre VVO (fast Monte Carlo).

Note: `-monte_trials` should be >= 50, otherwise the tool will report an error.

`mc` Enables Spectre traditional Monte Carlo.

`-monte_trials` Specifies the number of Monte Carlo trials. Default: `2000`

`-ecsmn` Characterize ECSMN (noise) data.

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

Specifies the maximum number of threads to use during the arcs update after reading fastsim waveforms and requires corresponding number of client licenses.
Default: `1` (no multithreading is used).

`-part_fastsim_thread <`*`number`*`>`

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

`-part_wave_thread <`*`number`*`>`

Specifies the maximum number of threads to use during the waveforms update portion of partitioning after fastsim and requires corresponding number of client licenses. Default: `1` (No multithreading is used).

`-part_write_thread <`*`number`*`>`

Specifies the maximum number of threads to use during the components writing portion of partitioning (overwrites `-partthread` value for the number of threads for writing partitions) and requires corresponding number of client licenses. Default: 0 (To use all available CPU threads).

`-partsim "`*`simulator_name`*`"`

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

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

`-partthread <`*`number`*`>`

Specifies the maximum number of threads to use duringfastsim partitioning simulations, and for writing partitions. Default: `0` (To use all available CPU threads).

Note: It is recommended to define a valid value to `-partthread` instead of keeping it unlimited.

`-user_arc_only`

Specifies to characterize only the user-specified arcs.

The different thread arguments of `char_macro` define the maximum number of threads to use on the current machine. If you do not specify any of these thread arguments, Liberate MX 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 MX table files provided and the number specified with the different thread arguments. See Specifying Input Stimuli for more information.

The `char_params` argument specifies a list of options that can be provided to Liberate characterization. Valid arguments are `-ccs`, `-ccsn`, `-ecsm`, `-ecsmn`, `-partthread`, `-charthread`, `-partsim`, `-charsim`. For more information on these options, see the `char_macro` command in the Liberate Characterization Reference Manual.

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.

Examples:

- Example 1:

```
# Spectre XPS for dynamic partitioning with 2 threads and spectre APS for
#characterization (partition) runs with 5 threads
    char_macro -partsim xps -partthread 2 -charsim aps -charthread 5
```

- Example 2

```
# Characterize memory(s) defined via a previous define_cell
# command. Use Spectre-XPS for dynamic partitioning and
# use Spectre for delay and constraint characterization on
# dynamic partitions.
# Partition using 2 threads and characterize using 4
# threads. Generate CCS timing and CCS noise models
# Use distributed runs but only for timing characterization
    char_macro -partsim xps -partthread 2 \
        -charsim aps -charthread 4 -ccs -ccsn \
```

- Example 3

  Refer the scenarios below. The `-partthread` argument (earlier referred to as `-thread`) globally sets several sub-options. However, you can overwrite each individual option with corresponding sub-command. For example:

```
-partthread 10
automatically sets:
-part_fastsim_thread 10
-part_wave_thread 10
-part_arc_thread 10
-part_write_thread 10


-partthread 1
automatically sets:
-part_fastsim_thread 1
-part_wave_thread 1
-part_arc_thread 1
-part_write_thread 1


-partthread 1
-part_fastsim_thread 10
automatically sets:
-part_wave_thread 1
-part_arc_thread 1
-part_write_thread 1
```

List of Deprecated Options

Following is a list of deprecated options for the `char_macro` command. These arguments are used only for backward compatibility.

`-arc_thread <`*`number`*`>`

How to multithread arcs update in preprocessing; overwrites -thread value. Default: *0* (Deprecated. Use `-part_arc_thread`)

`-extsim "`*`simulator_name`*`"`

External SPICE simulator program. (Deprecated. Use `-charsim`).

`-fastsim_thread <`*`number`*`>`

How to multithread FastSPICE in preprocessing; overwrites -thread value. Default: `0` (Deprecated. Use `-part_fastsim_thread`)

`-thread <`*`number`*`>` Maximum number of threads to use on the current machine. Default: `0` (Deprecated. Use `-partthread`)

`-wave_thread <`*`number`*`>`

How to multithread waveforms update in preprocessing; overwrites -thread value. Default: `0` (Deprecated. Use `-part_wave_thread`)

`-write_thread <`*`number`*`>`

How to multithread components writing in preprocessing; overwrites -thread value. Default: 0 (Deprecated. Use `-part_write_thread`)

char_memory

Runs the characterization of an instance of standard functionality (Standard Custom Instance Flow) or an instance designed by a third-party IP vendor (Vendor Recharacterization Flow).

Options

`-ccs`	Enables CCS characterization and library generation.
`-ccsn`	Enables CCSN characterization and library generation.
`-ecsm`	Enables ECSM characterization and library generation.
`-ecsmn`	Enables ECSMN characterization and library generation.
`-workdir "string"`	The complete path where the `mx_setup` directory containing the reuseable setup files is created. The default for this is the current directory.

This command is used along with the `define_memory` command to create all needed characterization files and run the characterization. It is required that `define_memory` is executed before the `char_memory` command.

compare_path

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

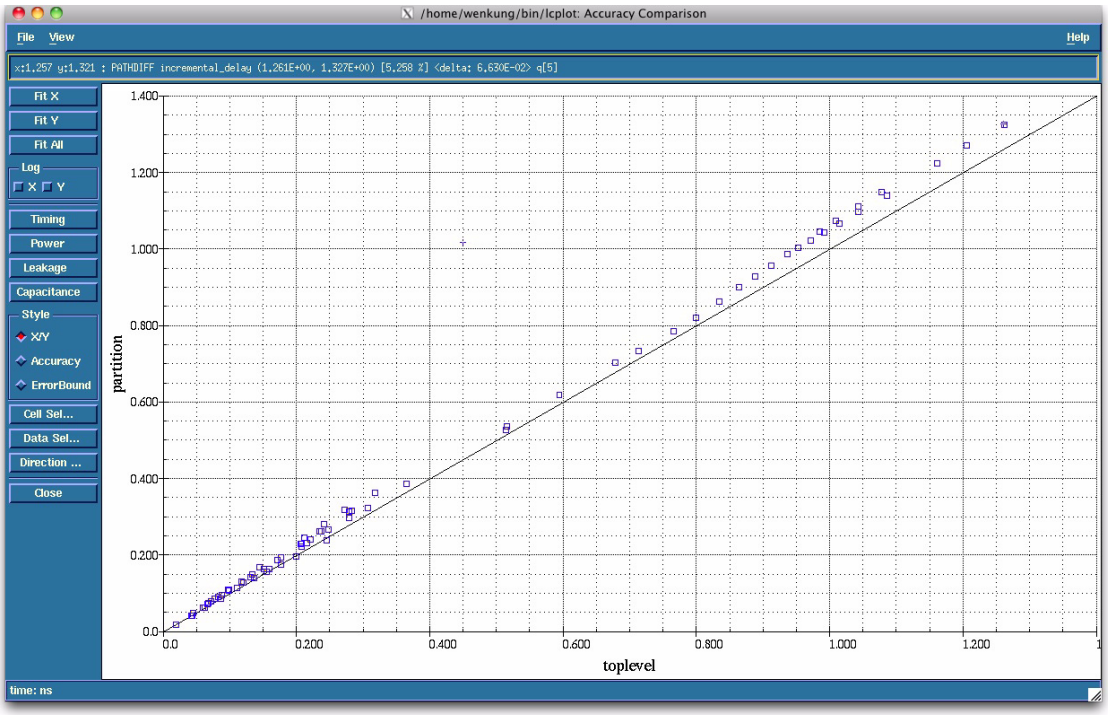
Options

`-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:



compare_timing

Generates a report in which top and partiton-level timing information is reported and compared in the same file.

To be able to compare directly on the nets in schematic, the target partition needs to be identified first, by showing the partition's path, through the existing `ShowPath` submenu.

Options

`cell <string>`	Specifies the cell name.
`-debug < 0 | 1 >`	Enables debug mode and print information. Default: `0`
`-gui < 0 | 1 >`	Displays generated report using `/usr/bin/vim`. Default: `0` (only generate the report)
`-mx_dir <string>`	Specifies the complete path to mx directory. Default: `mx_dir`.
`-nets <list>`	Specifies the specific net name to compare timing reports. Default: all nets.
`-part <string>`	Specifies the individual partition name to compare timing reports. Default: all partitions.
`-table_idx`	Index of arc type template entry (slew/slew/load) used for top level run. For example, for a delay partition, use `0` to indicate that the first slew/load point needs to be chosen for the correct comparison.

compare_waveforms

Generates an ocean script to be loaded in ViVA to compare top-level and partition waveforms. Waveforms are shifted in time so that they line up and refer to the same exact physical node. Nodes names and order are as listed in timing reports.

To be able to compare directly on the nets in schematic, the target partition needs to be identified first, by showing the partition's path, through the existing `ShowPath` submenu.

Options

`cell <string>`	Specifies the cell name.
`-debug < 0 | 1 >`	Enables debug mode and print information. Default: `0`
`-gui < 0 | 1 >`	Calls ViVA on generated script. Default: `0` (only generate the ocean scripts).
`-map <0 | 1>`	Specifies the map names before drawing waveforms.
`-mx_dir <string>`	Specifies the complete path to mx directory. Default: `mx_dir`.
`-nets`	Specifies the specific net name to compare waveforms. Default: all nets.
`-part <string>`	Specifies the individual partition name to compare waveforms. Default: all partitions.
`-table_idx`	Index of arc type template entry (slew/slew/load) used for top level run. For example, for a delay partition, use `0` to indicate that the first slew/load point needs to be chosen for the correct comparison.

define_arc

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

Options

`-attribute { list of attribute_names }`

Lists the pairs of attribute and values that can be specified to define arcs.

Example:

```
define_arc \
-attribute {attribute_name1  <value>  \
            attribute_name1  <value>  \
            ....
            ...
          attribute_nameN  <value> } \
<cell_name>
```

The following attributes that can be listed within `-attribute`.

`altos_clone_arc <list>`

Instructs Liberate MX 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.

`altos_constraint_path_delta_probe_mode < 0 |1 |2>`

Controls the functionality of constraint_path_delta_probe_mode for specific arcs.

- 0: Disables the functionality.
- 1: Disables set_var constraint_path_delta_probe_mode 2 for the specified arc.
- 2: Enables set_var constraint_path_delta_probe_mode 2 for the specified arc.

altos_minmax_probes <0|1|worst>

0 - Default.

1 - Chooses worst among all transistor terminals by computing constraint value for probe/ related_probe pairs.

worst - Selects the worst-case transistor terminals for clock probe and data probe.

altos_mx_bisection 1

Instructs Liberate MX 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_mx_bisection 1} \
$cell
```

altos_mx_bundle <bundle_name>::<bundle_criterion>

All arc definitions of the same arc, with the same string in the `altos_mx_bundle`, are considered for the characterization of the arc. The `min` or `max` specifies whether the minimum or the maximum of all `define_arcs` is used in characterization. The `bundle_name` 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_mx_bundle ABC::min}
define_arc -measure ABC_1 -attribute
{altos_mx_bundle ABC::min}
define_arc -measure ABC_2 -attribute
{altos_mx_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_mx_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_mx_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_mx_bundle ck_minperiod::max] $cell
```

`altos_mx_existence <string>`

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

```
altos_mx_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.)*

`altos_mx_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_mx_force_timing_type
    rising_edge} \
$cell
```

`altos_mx_ignore_select_index 1`

Disables the settings of `select_index`.

`altos_mx_seq_probing <list_of_probing_locations>`

Sets the node type to consider sequential component as possible candidate for data path probes in setup/hold characterization. This attribute will override the Tcl level variable mx_seq_probing. Allowed values are a list of input, internal, state or output.

`altos_mx_pathdelay_clock_margin`

Adds margin to setup or hold on clock path.

`altos_mx_pathdelay_data_margin`

Adds margin to setup or hold on data path.

`altos_mx_simulation_interval <value>`

Specifies a simulation interval on an arc-basis.

There are two other methods for specifying a simulation interval:

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

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

`altos_mx_transition_arc <value>}`

Measures transition arc with the help of measurement that is defined using the define_measure command.

Example:

```
set vdd50 [expr $VDD * 0.5]
set vdd20 [expr $VDD * 0.2]
set vdd80 [expr $VDD * 0.8]
define_measure -name slew_rise_raw -trig out -trig_val $vdd20 -trig_dir R -targ out -targ_val $vdd80 -targ_dir R -targ_type "delay" -duration 0.5 -keep max  cellname
define_measure -name slew_rise -trig out -trig_val $vdd20 -trig_dir R -duration 0.5 -equations {slew_rise_raw} cellname
define_arc -attribute {altos_mx_transition_arc 1} -measure slew_rise -related_pin clk -related_pin_dir R -pin out -pin_dir R cellname
```

`altos_autoprobing_skip_probe "<list_of_nodes>"`

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

`altos_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.

`altos_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
    {altos_autoprobing_skip_rel_probe CLK} \
$cell
```

`altos_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
    {altos_autoprobing_skip_rel_probe_regexp
    BL* WORD*} \
    $cell
```

`altos_autoprobing_level <value>`

Overwrites the values of both `mx_autoprobing_hold_level` and `mx_autoprobing_setup_level`.

`altos_use_fastsim_value_for_partition 1`

Instructs Liberate MX to adjust the final timing data to match the fastsim results.

`altos_check_rampup <pin>`

Enables the ramp-up check to be used for the arc and also specifies the trigger for the ramp-up measurement. The specified pin must be either the -pin or -related_pin of the arc.

```
altos_constraint_probe_lower_rise <value>
altos_constraint_probe_lower_fall <value>
altos_constraint_probe_upper_rise <value>
altos_constraint_probe_upper_fall <value>
```

When computing setup/hold times using path delay method, use this attribute to specify probe threshold for partition simulations of the specified arc. It will override values specified with `constraint_probe_[upper|lower]_[rise|fall]` and `constraint_probe_[setup|hold]_[upper|lower]_[rise|fall]` for partition simulations of the specified arc.

Example:

```
define_arc \
       -type setup \
       -related_pin_dir R -pin_dir R  \
       -related_pin {clk_in} \
       -pin {add_in[6:0]} \
       -attribute
"altos_constraint_probe_lower_rise 0.2
       altos_constraint_probe_lower_fall 0.2
       altos_constraint_probe_upper_rise 0.8
       altos_constraint_probe_upper_fall 0.8"
\
       $cell
```

```
altos_switching_output_probe_intersection 1
```

When enabled, a probe and a related probe pair for a constraint arc intersecting at a channel connected component (CCC) are considered invalid if at least one output of that CCC is not switching.

```
altos_ignore_arc_automeas 1
```

Instructs liberate MX to not create automatic measures for the arc. The user must specify `define_measure` for the arc. This is applicable to min_period or mpw arc types.

`altos_virtual_rail_waveform_shift_pin <pin>|<pin_relpin>`

When pin name is specified, the input pin will be used as trigger for the virtual rail waveform.

When <*pin_relpin*> is specified, the attribute will automatically model the virtual rail for pin and the related pin.

```
define_arc \
       -type hold \
       -related_pin_dir R -pin_dir R \
       -related_pin {clk} \
       -probe {dataprobe} \
       -related_probe {clkprobe} \
       -probe_dir R -related_probe_dir R \
       -force \
       -table {./constrainthold.tbl} \
       -pin {addr[4:4]} \
       -attribute
"altos_virtual_rail_waveform_shift_pin
        pin_relpin" \
       IP_NAME
```

In the above example, the `altos_virtual_rail_waveform_shift_pin` attribute will model the virtual rail for pin and the related pin.

Refer to mx_virtual_rail_waveform_shift_threshold, mx_remove_rc_timing, and virtual_rail_waveform_probe for more information on usage of this attribute.

`-delay_threshold {list}`

Four delay measurement thresholds (`in_rise`, `in_fall`, `out_rise`, `out_fall`).

`-data_slew`

Allows you to define the data slew when performing `mpw` and `min_period` characterization. The number of entries in the `-data_slew` list should be same as `define_template -type mpw` or `index_1` of the `define_template -type` constraint. This option is supported only when `define_arc -type` is set to `mpw` or `min_period`.

`-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
```

`-equation_only` Specifies for a measure-based `define_arc` command that the arc should be used only for evaluation of the equation it is used in. This means that the arc should never generate a group in the library.

`-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_memory command). It is a local arc specific version of the parameter `extsim_deck_header`. 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 ...
```

`-fastsim_arc_types <setup|hold|delay|retain|mpw|minperiod|power|leakage>`

Specifies a list of arc types for which to run characterization in the greybox mode. For example:

```
define_memory -fastsim_arc_types minperiod
```

`-force`

Allows only the probes specified by the `-probe` and `-related_probe` arguments.
Note: The use of `-probe` and `-related_probe` arguments (when not using `-force`) adds probes to the list of considered probes. The worst case of the Liberate MX determined probes and the define_arc declared probes are used in the partition simulations.

`-ignore`

Prevents characterization of the specified arc. The following define_arc options can be used to specify the arc: `-pin`, `-pin_dir`, `-related_pin`, `-related_pin_dir`, `-when`, `-vector`, `-pin_probe`, `-related_probe`.

You can also specify multiple arcs in one command.

If `-pin` or `-related_pin` is not specified, Liberate MX will include all the pins in `$cell` and their transition direction will be X. X includes Rise (R) and Fall (F).

If `-pin` (`-related_pin`) is specified but `-pin_dir` (`-related_pin_dir`) is not specified, the tool will assume the transition direction to be R. In this scenario, `-type` will have one more attribute `all`, which will include all types of arcs.

Examples:

The following example code will ignore all the arcs of `combinational` type that have `CK` as related pin and `R` as `related_pin_dir`, regardless of pin or pin direction.

```
define_arc \
-ignore \
-related_pin CK \
cell
```

The following will ignore all the arcs of `setup` type that have `CK` as related pin, regardless of related pin direction or pin or pin direction.

```
define_arc \
-ignore \
-type setup \
-related_pin CK -related_pin_dir X\
cell
```

The following will ignore all the arcs of `setup` type, regardless of pin, pin direction, related pin, or related pin direction.

```
define_arc \
-ignore \
-type setup \
cell
```

The following will ignore all the arcs.

```
define_arc \
-type all \
-ignore \
cell
```

`-measure "`*`name`*`"` Name of associated `define_measure` command that has an equation.

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

`-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.

`-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).

`-slew_threshold {list}`

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

`-table {list}`

Specifies a list of specific table files to be evaluated for an arc. If this option is not specified, the arc is evaluated in all tables according to existing rules. The specified table files should include full path (the path can include symbolic link that gets resolved automatically).

Example:

```
define_arc \
    -type power \
    -pin { data_out[12:0] }
    -related_pin CLK \
    -pin_dir R \
    -related_pin_dir R \
   -table { /home/mxuser/mx/mx_setup/tables/power_0.tbl \
        /home/mxuser/mx/mx_setup/tables/power_1.tbl } \
    myCell
```

Note: This arc is evaluated in the `power_0.tbl` and `power_1.tbl` table files only. If the run includes more table files that were specified in the `-table` option, the additional files are ignored for this arc evaluation.

`-table_line "string"`

Specifies the line of a truth table (specified using the `-table` option) to be evaluated for an arc. Default: ""

Note: This option is compatible with the Liberate MX characterization and Liberate AMS characterization flows only.

Example:

```
measure.tbl:
--------------------------
arctypes delay setup

table add_in_clk_in
pins   clk_in add_in chip_en data_out
init   0      L      H       X
1a    1      0      H       B
1b    0      L      H       B
2a    1      0      H       B
2b    0      L      H       B
...
endtable
------------------------------


template.tcl:
-----------------------
define_arc -type setup \
    -pin {add_in[0]} -related_pin {clk_in} -pin_dir R -
related_pin_dir R \
    -table_line "2b" \
    rf_top
-----------------------
```

In the above example, the arc will be evaluated only for line `2b` only.

`-type <arc_type>`

Type of arc. This option accepts one of the following values: `async`,`ccsn_first`, `ccsn_last`, `combinational`, `dc_current`, `disable`, `edge`,`enable`, `hidden`, `hold`, `max_clock_tree_path`, `min_clock_tree_path`, `min_period`, `minperiod`, `mpw`, `nochange_high_high`, `nochange_high_low`, `nochange_low_high`, `nochange_low_low`, `non_seq_hold`, `non_seq_setup`, `power`, `recovery`, `removal`, `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
```

In addition, here is an example of measurement that are not for the library file and uses `-type measure`. This is important in case there is a `define_measure` written to check for some user specified condition which does not need to go in the liberty file. Liberate MX requires that `define_measure` must be referenced by `-measure` option of the `define_arc` command in order to be evaluated. For measurements that the user does not want in the library file, use the `measure` type of the `define_arc` command .

```
define_arc \
-type measure \
-when "RET1N" \
-related_pin_dir R -pin_dir F \
-related_pin {CLK} \
-pin {CEN} \
-measure cen_f_setup \
instance_name
```

`-value <value | {list}>`

	Overrides the characterized values. Default: use the characterized values. The value(s) override the characterized results and forces a value (or list of values) for all entries into the data table for the specified arc. This option accepts a single value or a list of values. If a single value is provided, then the table gets a scalar type of data. However, if a list is specified, there must be one entry for each point in the data table. For example, a 7x7 table would require 49 values. For any time-based arc, the value is in seconds (i.e.: 5e-9 = 5ns).
`-value_trans <value | {list}>`	
	Overrides the values in the transition table. Default: use the characterized values. The `-value_trans` option works similarly to the `-value` option except that it applies to transition table. See the `-value` option for more information.
`-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 MX automatically extracts arcs from the provided table files. However, the `define_arc` command allows you to:

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

Using the -attribute {} option

When specifying multiple attributes in a `define_arc` command, use `-attribute` only once and specify the other attributes as space-separated list. For example:

```
-attribute {altos_autoprobing_skip_probe_regexp "BL* WORD*" \
       altos_autoprobing_skip_rel_probe_regexp "BL* WORD*" \
       altos_process_probe_relprobe_intersection 1 \
       altos_use_fastsim_value_for_partition 1 } \
```

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 the Liberate MX 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 MX, 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 single entity, 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
# force when conditions on memory's leakage power
define_arc -type leakage -when "ME" -cell $cell
define_arc -type leakage -when "!ME" -cell $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*.

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_mx_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
```

Example 5

This will demonstrate the use of multiple attributes in `define_arc`, in this case `altos_mx_autoprobing_skip_probe` and `altos_autoprobing_level`:

```
define_arc
    -type hold \
    -pin {ena1} \
    -related_pin {clk} \
    -attribute {altos_mx_autoprobing_skip_probe "node1" altos_autoprobing_level 1}
    \
    myCell
```

define_cell

Defines how a memory 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.

`-mxcore {`*`core_files`*`}`	List of one or more core cell description files used to match memory core cells in the design during automatic probing. For detailed information, see Appendix B, "Specifying Memory Core Cells."
`-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.
`-when <"`*`function`*`">`	Specifies user-defined cell-level logic constraints using the Liberty format *when* syntax, constraining the automatic vector generation of Liberate for the given cell. The `define_cell -when` logical condition applies only to steady state signals such as leakage states and side input states that are non-switching. Liberate does not automatically infer simultaneous switching inputs based on the logical condition. You can use `define_arc` to specify simultaneous switching inputs, or specify a truth table to be translated automatically.
`{`*`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 MX, 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.

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:

- addrA and addrB must have the same "direction" (cannot duplicate an input to an output.)
- 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.
- addrB does not need to exist—it can be created when the write_library command is run.
- 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.
`-index_3 {`*`indices`*`}`	List indices to use as index_3.

`-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`, `si_immunity`, `ccsn_vout`, `ccsn_prop`, `ccsn_prop_high`, `ccsn_prop_low`, `ccsn_dc`, 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_memory and the and char_macro commands.

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 0.07 0.5 } \
    rf_cell
```

define_leafcell

Defines the level of hierarchy that resides at the bottom of a cell-level netlist. This allows Liberate MX 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 parameter 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`, `nmos_soi`, `pmos_soi`, `diode`, `r`, or `c`. The `nmos_soi` and `pmos_soi` types are used to identify SOI (Silicon on Insulator) leafcell mosfets. The supported pinout of `nmos_soi` and `pmos_soi` related types is: `MXX d g s e [p] [b] [t] mname [L=val] [W=val] ...` where: `MXX` is the instance name. `d` is the drain. `g` is the gate. `s` is the source. `e` is the back gate (or substrate). `p` is the external bulk. `b` is the bulk. `t` is the temperature node. `mname` is the model name. **Note:** To use a vsource device in the characterization process, you must define it as a `blackbox`. For example, `define_leafcell -type black_box -position {0 1} vsource`
`-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
```

- Support 3-terminal transistors (PODE) with define_leafcell `-pin_position` to map a leafcell terminal to multiple MOS ports (d/g/s/b). If the PODE (S=D) is a 3-terminal subckt with pins (D G B), it can be defined as:

```
define_leafcell -type nmos -pin_position {0 1 0 2} {npode_mac}
```

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.

`-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`.
`-when <"function">`	User-specified logic conditions using the Liberty `when` format syntax. (REQUIRED)
`{cell_names}`	List of cell names.

`-value { supply value }`	Specifies a list of pairs of supply name and leakage value.
	
	Default: use the characterized leakage value
	
	Example:
	
	`define_leakage -value { \`
	`VDD $vdd_leak_val VSS $vss_leak_val } \`
	`$cellname`

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_memory`, and `write_library` commands.

Examples:

```
# Define the leakage conditions of a RF instance
define_leakage -when " !CEN" rf128
define_leakage -when " CEN" rf128
```

define_measure

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

Note: The `define_measure` flow for constraint arcs is not supported in the greybox mode.

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 *mx_ 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.

`-exit_on_failure` Terminates the Liberate MX run if the measurement fails.

`-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 <none | min | max>`

Instructs MX to keep the result of the measurement as a group in the LDB. It also specifies how to resolve the final value when the measurement evaluates to different values in multiple simulation intervals. Default: `none`

`min`	Specifies that the minimum value is used as the final value.
`max`	Specifies that the maximum value is used as the final value.
`none`	No value is used as the final value.

`-max_val_reuse_factor <`*`value`*`>`

Multiplies `max_val` during the partitioning step. Default: `1`

`-min_val_reuse_factor <`*`value`*`>`

Multiplies `min_val` during the partitioning step. Default: `1`

`-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:

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

Example:

```
define_measure -name rawmeas .... $cell
define_measure -name eqmeas .. -equations $rawmeas $cell
define_arc -measure eqmeas -when "W"
```

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

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

Examples:

- If you set `-targ {A}`, the tool will find time when `v(A)=-targ_val`.
- If you set `-targ {A B}`, the tool will find time when `abs(v(A)-v(B))=-targ_val`.

Note: This option does not support two nodes.

`-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` or `R`	Specifies that the target signal direction is rising.
`fall` or `F`	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`

Note: When using `-targ_s` with a measurement, it must be a time-based measurement.

`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.

A few points to be noted:

- If `voltage` is specified without `-trig_start/-trig_end`, Liberate MX reads the voltage of `-targ {B}` where `-trig {A}` crosses `-trig_val`.
- If `voltage` is specified with `-trig_start/-trig_end`, Liberate MX searches for a `min/max` voltage in that time frame.
- When using `-targ_type voltage`, set `-targ_val 0`.

`-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`

Note: This option does not support two nodes.

`-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` or `R` Specifies that the target signal direction is rising.

`fall` or `F` 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.

Note: When using `-trig_s` with a measurement, it must be a time-based 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`

`-val_reuse_factor <`*value*`>`

Specifies all reused factors to be applied. Default: `-1` (`Not Used`)

`-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 `mx_margin_report` parameter.

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.

If you define `define_arc` followed by `define_measure`, then Liberate MX generates partition for that path and reports fastsim top-level evaluation as well as partition values. The results of the measurement is reported to files `measure.rpt.fastsim` and `measure.rpt`.

For example:

```
define_measure -name N -when "W"
define_arc -measure N -when "W"
```

If only `define_measure` is defined, then Liberate MX evaluates it in fastsim and the measurement result is reported in the `measure.rpt.fastsim` file.

For example:

```
define_measure -name N -when "W"
```

Options Associated with Reuse Flow

The `reuse_factor` options that to be used are as follows:

- failed_val_reuse_factor
- max_val_reuse_factor
- min_val_reuse_factor
- targ_val_reuse_factor
- trig_end_val_reuse_factor
- trig_start_val_reuse_factor

- trig_val_reuse_factor
- val_reuse_factor

These options are associated with the Reuse flow, which requires that during the partition phase, waveforms are scaled from the old to the current values of rails. This is done automatically for automatic types of measurements. However, for user specified measurement - with user specified voltage levels - (i.e. `define_measure` commands) it is unknown whether a specified level is to be considered a voltage level (to be scaled) or an absolute value (to be left untouched). These options are used to explicitly specify scaling. The option `val_reuse_factor` is a `"master"` scaling factor that sets all the reuse scaling factors at once.

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 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_memory

Describes the attributes and characterization settings of a memory instance to be characterized.

Options

`-additional_arcs list()`

List of one or more arcs to be included in the characterization.

`-additional_tables {list_of_table_files}`

List of one or more behavioral tables to be included in the characterization. These specified table files supplements the tables automatically generated by the `define_memory` flow.

`-address <address_base_name>`

Base Address bus name. This is the root name, not the full bus name and without any port or test prefix or suffix.

For example, if a dual port memory has addresses named `ADRA[9:0]` and `ADRB[9:0]`, then the root name would be `ADR` and `{A B}` would be `-port_suffix`. (Standard Custom Instance flow).

`-autoprobing_hold_level  list ()`

Level value followed by base pin names.

`-autoprobing_setup_level list ()`

Level value followed by base pin names.

`-banks int (1)` Number of banks.

`-bisection <0 | 1>`

Enable bisection for constraints. Set to `0` for path delay flow or `1` for bisection flow. Default: `0`

`-bist_prefix <`*`list`*`>`

Signal prefix for test mode. This specifies the prefix to be added to the base signal names to specify the signal names to be used for the test mode operation. This affects any defined address (`data_in`, `data_out`, `chip_enable`, `write_enable`, `bit_write`, or `test_enable`). (Standard Custom Instance flow)

`-bist_suffix <list>`

Signal suffix for test mode. This specifies the suffix to be added to the base signal names to specify the signal names to be used for the test mode operation. This affects any defined address (`data_in`, `data_out`, `chip_enable`, `write_enable`, `bit_write`, or `test_enable`). (Standard Custom Instance flow)

`-bits` It is the width of one word in memory.

`-bit_mask list()` Base write enable bus name, followed by its state for bit is masked (L or H).

`-bit_write {base_name masked_state}`

Base Write Enable bus name, followed by its state for bit is masked (L or H). For example, if a dual port memory has bit write enables named `BWENA` and `BWENB`, then the root name would be `BWEN` and `{A B}` would be `-port_suffix`. If the bit is masked when the state is low, the correct argument would be `{BWEN L}`. (Standard Custom Instance flow)

`-bit_specific_when <0 | 1>`

When set to 1, maintains bit specific `when` conditions in the design. In default behavior neither bit-specific `when` conditions is maintained nor existing ones in the reference library is maintained. Default: 0

For example:

```
define_memory -bit_specific_when 1
```

`-bitcell <rom|single_port|dual_port|2prf|10t>`

Specify the type of bitcell. Available values are `rom`, `single_port`, `dual_port`, `2prf`, or `10t`. Default is to determine the probable bitcell based on the pinout of the instance. (Standard Custom Instance flow)

`-bypass_enable list()`

Base Bypass Enable pin followed by its state for chip is in bypass mode (L or H).

`-bypass_suffix list()`

Signal suffix for bypass mode.

`-cfg_file <cfg_file_name>`

The cfg file generated from the compiler along with the instance (Virage and TSMC recharacterization flows).

`-char_script <`*`string`*`>`

File containing the characterization commands.

`-char_spice <simulator_name>`

Characterization SPICE simulator to be used for characterization. The valid values are `aps`, `spectre`, and `xps`. Default: `aps`.

`-char_thread <number>`

Number of threads to be used for characterization simulations. Default is to use maximum available threads or licenses. Overrides the value specified in `-thread`.

`-chip_enable {base_name active_state}`

Base Chip Enable pin name, followed by its state for chip is active (L or H). For example, if a dual port memory has chip enables named `CENA` and `CENB`, then the root name would be `CEN` and `{A B}` would be `-port_suffix`. If the chip enable is active low in this case, the correct argument would be `{CEN L}`. (Standard Custom Instance flow)

`-clk_bist_prefix <`*`string`*`>`

Clock prefix for test mode. This specifies the prefix to be added to the base clock name to specify the clock name to be used for the test mode operation. (Standard Custom Instance flow)

`-clk_bist_suffix <`*`list`*` >`

Clock suffix for test mode. This specifies the suffix to be added to the base clock name to specify the clock name to be used for the test mode operation. (Standard Custom Instance flow)

`-clk_port_suffix {list}`

List of clock port suffixes. This option is needed for instances with more than one port. These are the suffixes added to the base names to differentiate between the signals that are used for the various ports. This affects any defined clock. (Standard Custom Instance flow)

`-clock <clock_base_name>`

Base Clock pin name. This is the root name without any port or test prefix or suffix. For example, if a dual port memory has `CLKA` and `CLKB`, the `-clock` would be `CLK` and `{A B]` would be the `-clock_bist_suffix`. (Standard Custom Instance flow)

`-clock_active_edge <`*`string`*`>`

Active edge of clock for edge triggered designs. The valid values are `rise` and `fall`. Default: `rise`

`-column_mux`

Column MUX is used to optimize designs (Power/Performance/Area). A MUX number shows a muxing done for a column and controlled by address bits that is the column address. Higher the column address lower is the number of physical rows in designs.

`-compiler_name <`*`string`*`>`

Name of the generating compiler.

`-complimentary_address_low <`*`string`*`>`

The low address to toggle all the output pins in the ROM design. You also need to specify the `-complimentary_address_high` for switching the output pins.

`-complimentary_address_high <`*`string`*`>`

The high address to toggle all the output pins in the ROM design. You also need to specify the `-complimentary_address_low` for switching the output pins.

`-const_arc_attr list ()`

List of constraint arc attribute.

`-data_in <data_in_base_name>`

Base Data In bus name. This is the root name, not the full bus name and without any port or test prefix or suffix. For example, if a dual port memory has addresses named `DA[9:0]` and `DB[9:0]`, then the root name would be `D` and `{A B}` would be `-port_suffix`. (Standard Custom Instance flow)

`-data_out <data_out_base_name>`

Base Data Out bus name. This is the root name, not the full bus name and without any port or test prefix or suffix. For example, if a dual port memory has addresses named `QA[9:0]` and `QB[9:0]`, then the root name would be `Q` and `{A B}` would be `-port_suffix`. (Standard Custom Instance flow)

`-debug int (0)`

Print Verbose debug information. The valid values are `0` and `1`. Default: `0`

`-derive_table_from_tmpl int (1)`

Derive MX tables from template.

`-design <sram|rf|uhdrf|rom>`

Name of design. Allowed values are `sram`, `rf`, `uhdrf`, or `rom`. (Standard Custom Instance flow)

`-expand_buses <string>`

Controls whether the buses are written out expanded in the library file. The valid values are `yes` or `no`. Default: `no`

`-extsim_cmd_option <string>`

Specify circuit simulation options. This option is used to specify simulator control options for the extsim simulator. Default is `""`. The input option string is automatically added into the `mx.tcl` file.

`-fastsim_arc_types <setup|hold|delay|retain|mpw|minperiod|power|leakage>`

Specifies a list of arc types for which to run characterization in the greybox mode. For example:

```
define_memory -fastsim_arc_types minperiod
```

`-fastsim_cmd_option <string>`

Specify fast simulator options. This option is used to specify simulator control options for the fastsim simulator. Default is `""`. The input option string is automatically added into all MX table files.

`-foundry  <TSMC | SMIC>`

Foundry for determination of device names. Allowed foundries are `TSMC` and `SMIC`. For all other foundries, it will be necessary to define the device leafcells in the `mx_setting` file.

`-global_voltage <voltage_value>`

Global Power supply voltage value in `volts`. Specifies the power supply value for vendor re-characterization instance. These instances do not use the `-rail` option since the power supply names are defined internally. For custom instances with `-rail` defined, it overrides the maximum rail voltage to define the instance operating condition.

`-greybox <0 | 1>`

Enables Grey Box Mode (no partitioning). Set to `0` for standard flow or `1` for grey box mode. Default: `0`

`-internal_threshold {lower_threshold upper_threshold}`

List of two values to be used for the lower and upper internal probing thresholds in percent. Default is `{20 80}`.

`-latch_probing list ()`

Override for latch internal probing option. Default is Input, State, Internal, Output.

`-loads {list_of_loads}`

List of loads to be used for characterization in `pf`. Overrides internal or template or library definition.

`-model_format <spice | spectre>`

Model Format. Allowed values are `spice` or `spectre`. Default: `spice`

`-models <model_include_file>`

SPICE Model include file.

`-model_leafcell`

Used to confirm if `-element` is necessary for the specialized foundry. Default: `false`
If the `-foundry` option is used for determining device names (for example, `mos` is defined as `.model` in model file and as `M` in netlist) then the `-element` option of the `define_leafcell` command is required to be defined. For example:

```
define_memory -foundry TSMC -model_leafcell 1
```

`-models_leakage <model_leakage_include_file>`

Spice Model include file for leakage characterization. If not defined, then `-models` is used.

`-models_power <model_power_include_file>`

Spice Model include file for power characterization. If not defined, then `-models` is used.

`-mx_setting <mx_tcl_file>`

Name of user-provided Tcl file containing Liberate MX settings and commands. This is used to override the internal Liberate MX settings that are defined in the `define_memory` flow.

`-netlist <netlist_file>`

The instance netlist to be used for the characterization.

`-netlist_format <spice | spectre>`

Netlist Format. Allowed values are `spice` or `spectre`. Default: `spice`.

`-number_of_ports int ()`

Number of ports in the bitcell, For dual port memory it is `2`.

`-other_input_pins list ()`

List of Input Pins of other functions.

`-output_enable list ()`

Base Output Enable pin name, followed by its state (L or H).

`-part_spice  <simulator_name>`

Partition SPICE simulator to be used for characterization. Allowed values are `xps` and `aps`. Default: `xps`.

`-part_thread <number>`

Number of threads to be used for partitioning simulations. Default is to use maximum available threads or licenses. Overrides the value specified in *-thread*.

Note: It is recommended to define a valid value to `-part_thread` instead of keeping it unlimited.

`-part_waveform_format <`*`string`*`>`

Format of partition simulation result. Allowed values are `sst2` or `fsdb`. Default: `sst2`

For example, to use `fsdb` as the waveform format for Liberate MX top-level simulation runs, specify:

```
part_waveform_format -string "fsdb"
```

`-pin_file <pin_file>`

File containing the pin name information. For more information on usage of `-pin_file`, see Guidelines for Usage of -vendor and -pin_file in define_memory Flow.

`-port_suffix {list_of_q_loads}`

List of port suffixes. This option is needed for instances with more than one port. These are the suffixes added to the base names to differentiate between the signals that are used for the various ports. This affects any defined address (`data_in`, `data_out`, `chip_enable`, `write_enable`, `bit_write`, or `test_enable`). (Standard Custom Instance flow)

`-process_node <process_node>`

Process Node like `65nm`, `55nm`, `45nm`, `40nm`, `32nm`, or `28nm`. (Vendor Recharacterization flow if `cfg` file is not provided).

`-qloads {list}` List of loads to be used for characterization for the Data out pin in `pf`. Overrides internal, template or library, or `-load` definition.

`-rail {paired_list_of_supplies_and_values}`

Name and value pairs for each Voltage supply. Value is in `volts`. Format is `{<supply1> <supply1_voltage> <supply2> <supply2_voltage> ...}` (Standard Custom Instance flow)

`-rcdb <0|1>` Enable the RCDB flow. Set to `0` for standard flow or `1` for rcdb flow. Default: `0`

`-read_enable list ()`

Base Read Enable pin name, followed by its state for chip is reading (L or H).

`-read_port list ()` List of read ports pins.

`-read_timer list()` Base Read Timer pin name.

`-read_timer_enable list ()`

Base Read Timer Enable pin name, followed by its state (L or H).

`-redundancy_address list ()`

Redundancy Address bus name.

`-redundancy_enable list ()`

Base Redundancy Enable pin followed by its state (L or H).

`-ref_lib <string>` Name of reference library. Do not use if `-template` is specified.

`-remove_tables {list_of_tables}`

List of automatically generated tables to be removed from the run. Available values are: `leakage`, `delay`, `constraint`, `power`, `measure`, `bist`, `sleep`, `bist_pwr`.

`-scan_enable list ()`

Base Scan Enable pin name, followed by its state for chip is in scan mode (L or H).

`-scan_in list ()` Base Scan In bus name.

`-scan_out list ()` Base Scan Out bus name.

`-setup_reuse<0|1>` Reuse setup files from the previous run if available. Allowed values are `0` (do not reuse) and `1` (reuse). Default: `0` (not to reuse)

`-shutdown list ()` Base Shut Down Enable pin name, followed by its state (L or H).

`-simulation_interval <number>`

Value for `mx_simulation_interval` in seconds. Default is `20e-9`.

`-skip_read_model int (0)`

Controls whether the models will be read in with read_spice.

`0`: Reads the SPICE Models.

`1`: Skips reading the SPICE models.

`-sleep list ()` Base Sleep Enable pin name, followed by its state (L or H).

`-slews {list_of_slews}`

List of slews to be used for characterization in `ps`. Overrides internal or template or library definition.

`-targ_thresh <value>`

Defines target (end point) threshold as percentage of VDD.

`-tbl_file_limit int (200)`

Table file line number limitation. Default: `200` lines

`-temp <temperature>`

Temperature in degrees C for characterization.

`-template <string>` Name of template file. Do not use if `-ref_lib` is specified.

`-template_vec <string>`

Name of template vector.

`-test1 list ()` Base Test1 Enable pin followed by its state for chip is in test1 mode (L or H).

`-test_enable {base_name active_state}`

Base Test Enable pin followed by its state for chip is in test mode (L or H). This is for specifying the a test mode that controls whether regular inputs or test inputs are active. For example, if a dual port memory has test enables named `TENA` and `TENB`, then the root name would be `TEN` and `{A B}` would be `-port_suffix`. If the instance is in test mode when the pin is low, the correct argument would be `{TEN L}`. (Standard Custom Instance flow)

`-thread <number>` Number of threads to be used for all aspects of characterization. Default is to use maximum available threads or licenses.

`-trig_thresh <value>`

Defines trigger (starting point) threshold as percentage of VDD.

`-unique_power int (0)`

Enables `-unique power` in `write_template`.

`-vendor <ARM|Virage|TSMC>`

Name of Vendor. (Vendor Recharacterization flow). For more information on usage of `-vendor`, see Guidelines for Usage of -vendor and -pin file in define memory Flow.

`-virtual_rails {paired_list_of_supplies_and_values}`

Paired list of virtual rails and their expected voltage level. Value is in `volts`. Format is `{<supply1> <supply1_voltage> <supply2> <supply2_voltage> ...}`.

`-words <number_of_words>`

Number of Words in the memory instance. (Standard Custom Instance Flow and Vendor Recharacterization flow if `cfg` file is not provided).

`-write_enable {base_name active_state}`

Base Write Enable pin name, followed by its state for chip is writing (L or H). For example, if a dual port memory has write enables named `WENA` and `WENB`, then the root name would be `WEN` and `{A B}` would be `-port_suffix`. If the write enable is active low, the correct argument would be `{WEN L}`. (Standard Custom Instance flow).

`-write_library_args` Provides additional arguments to be used in the `write_library` phase of characterization.

`-write_port list()` List of write ports pins.

`-write_timer list ()`

Base Write Timer pin name.

`-xps_smode_power <`*`string`*`>`

Use XPS s-mode for power. XPS must be selected as FastSPICE to use this option.

`{cell_names}` List of Cells (required). Default: `none`.

This command is used along with the `char_memory` command to create all needed files for characterization and run the characterization. It is required that the `define_memory` command must be executed before the `char_memory` command. This command can be used to describe an instance of standard functionality (Standard Custom Instance Flow) or an instance designed by a third-party IP vendor (Vendor Recharacterization Flow). In this flow a directory called `mx_setup` is created containing the files needed to characterize the memory instance. Once a memory instance has been run through the flow, the user has the ability to edit the gerneated files and rerrun by setting the `setup_reuse` flag to `1`.

The define_memory command supports different models for timing, power, and leakage. For example:

```
define_memory \
-models [pwd]/DATA/include_models \
-models_power [pwd]/DATA/include_models_power \
-models_leakage [pwd]/DATA/include_models_leakage \
... ...
```

Note: If there is no `-models_power` or `-models_leakage`, the **define_memory** command uses `-models` as the default model files.

Usage of `define_memory` flow

The two flows are supported in the `define_memory` flow.

Automated Vendor Re-characterization Flow

When you are using memories from a third-party vendor, you might be adding an uncharacterized PVT or verifying the compiler accuracy. An example of the usage of `define_memory` for the recharacterization of a Vendor Instance:

- Example 1

```
define_memory \
    -ref_lib [pwd]/ref_lib/SRAM_2048x16.lib \
    -netlist [pwd]/netlist/SRAM_2048x16.spf \
    -vendor "TSMC" \
    -global_voltage 1.1 \
    -temp 25 \
    -models [pwd]/models/include_tt_model.sp \
    -mx_setting [pwd]/mx_settings.tcl \
    SRAM_2048x16
```

- Example 2: Two port Register file: Memory with 2 dedicated ports (1R 1W)

```
    CLKR: clock pin for read port
    CLKW: clock pin for write port B
    AA: address bus for port A [read]
    AB: address bus for port B [write]
    D : data bus  [write port only]
    Q : data out bus [read port only]
    WEB: write enable (active low)

define_memory \
    -netlist [pwd]/regfile.sp \
    -template [pwd]/template.tcl \
    -mx_setting [pwd]/mx_setting.tcl \
    -additional_tables {[pwd]/new_table.tbl} \
    -vendor tsmc \
    -foundry tsmc \
    -design rf \
```

```
    -bitcell 2prf \
    -number_of_ports 2 \
    -global_voltage 0.9 \
    -temp 0 \
    -models [pwd]/models/include_ss \
    -process_node 45nm \
    -virtual_rails [list \
        VDDI 0.9 \
        VSSI 0 \
        ] \
    regFile
```

Automated Standard Instance Flow

The design is of standard functionality and can be characterized using `define_memory` flow. Anexample of the usage of `define_memory` for the characterization of a Standard Custom Instance:

- Example1 pinout (Single port SRAM: Standard SRAM without test mode):

```
    CLK:       clock pin
    ADR[10:0]:  address bus
    DIN[15:0]: data bus
    Q[15:0]:   data out bus
    CEN:       chip enable (active low)
    WEN:       write enable (active low)

define_memory \
    -netlist SRAM_2048x16.spf \
    -clock CLK \
    -address ADR \
    -data_in DIN \
    -data_out Q \
    -chip_enable {CEN L} \
    -write_enable {WEN L} \
    -rail {VDD 1.0 VSS 0} \
    -temp 25 \
    -foundry TSMC \
    -models [pwd]/models/include_tt_model.sp \
    -template [pwd]/template.tcl \
    -mx_setting [pwd]/mx_settings.tcl \
    SRAM_2048x16
```

- Example 2 pinout (Dual port SRAM (DPSRAM) with test mode)

  Dual port SRAM, Standard SRAM with two ports which are symmetric and capable of reading and writing.

```
CLKA:        clock pin for port A
CLKB:        clock pin for port B
TCLKA:       test mode clock pin for port A
TCLKB:       test mode clock pin for port B
ADRA[10:0]:  address bus for port A
ADRB[10:0]:  address bus for port B
TADRA[10:0]: test address bus for port A
TADRB[10:0]: test address bus for port B
DINA[15:0]:  data bus for port A
DINB[15:0]:  data bus for port B
TDINA[15:0]: test mode data bus for port A
TDINB[15:0]: test mode data bus for port B
QA[15:0]:    data out bus for port A
QB[15:0]:    data out bus for port B
CENA:        chip enable for port A (active low)
CENB:        chip enable for port B (active low)
TCENA:       test mode chip enable for port A (active low)
TCENB:       test mode chip enable for port B (active low)
WENA:        write enable for port A  (active low)
WENB:        write enable for port B  (active low)
TWENA:       test mode write enable for port A  (active low)
TWENB:       test mode write enable for port B  (active low)
TENA:        test mode select for port A  (active low)
TENB:        test mode select for port B  (active low)

define_memory \
    -netlist SRAM_2048x16.spf \
    -clock CLK \
    -address ADR \
    -data_in DIN \
    -data_out Q \
    -chip_enable {CEN L} \
    -write_enable {WEN L} \
    -test_enable {TEN L} \
    -port_suffix {A B} \
    -clk_bist_prefix T \
    -bist_prefix T \
```

```
    -rail {VDD 1.0 VSS 0} \
    -temp 25 \
    -foundry TSMC \
    -models [pwd]/models/include_tt_model.sp \
    -template [pwd]/template.tcl \
    -mx_setting [pwd]/mx_settings.tcl \
    SRAM_2048x16
```

- Example 3: Read Only Memory (ROM) with standard functionality where each read operation finishes in single clock cycle

```
define_memory \
    -netlist [pwd]/rom.sp \
    -template [pwd]/rom_template.tcl
    -mx_setting [pwd]/mx_setting.tcl
    -bitcell rom \
    -design rom \
    -global_voltage 1.08 \
    -temp 125 \
    -words 4096 \
    -bits 32 \
    -models [pwd]/include_ss \
    -rail {VDD 1.08 VSS 0} \
    -clock CLK \
    -data_out Q \
    -address A \
    -chip_enable {CEN L} \
    -unique_power 1 \
    rom
```

define_pincap

Enables dynamic pin capacitance characterization for the defined pins.

Options

`-pin {pins}`	List of input pins for pin cap arc.
`-pin_dir < R | F >`	Specifies the pin direction. R=rise and F=fall.
`-when <expression>`	Defines the state dependency of the arc.
`-attribute <attribute_value>`	List of user-defined attribute/value pairs.
`{ cellNames }`	List of cell names. (REQUIRED)

Example

```
define_pincap -pin clk_in -pin_dir R -when "wr_in" rf_top
define_pincap -pin clk_in -pin_dir F -when "wr_in" rf_top
define_pincap -pin clk_in -pin_dir R -when "!wr_in" rf_top
define_pincap -pin clk_in -pin_dir F -when "!wr_in" rf_top
```

Notes

- It is not mandatory to define a delay arc or a hidden arc with a `define_pincap` command. To define a delay arc or hidden arc along with dynamic pin capacitance for characterization, use `define_arc` in addition to `define_pincap`. Both would be required if you need the arcs. If the pin does not require any arcs, and you are only characterizing pin cap for the pin, use `define_pincap`.
- To enable dynamic pin capacitance characterization the following two things are required:
  - Pin cap table file
  - Template pin cap definition of the pins and `when` conditions to characterize

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. Multiple `define_cell` commands can reference a single template.

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. (REQURED)
`-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_macro` command.

Note: Internally, `define_template` uses a fixed set of units (listed below). These cannot be changed; *however*, units may 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)
`capacitence`	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 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` parameter 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
```

Guidelines for Usage of -vendor and -pin_file in define_memory Flow

- If vendor information is unknown and unsupported, specify `-data_in`, `-data_out`, `-clock`, or any other information.

- If vendor information is known and supported, specify `-vendor`. The **define_memory** command generates a `pin_file` [`%outdir/mx_setup/pin_file`].
- If `pin_file` is correct, use it and also use the table that **define_memory** generates.
- If `pin_file` has an error, modify it manually (`pinfile.modified`). Also, modify the removing `-vendor` from the **define_memory** command and instead add `-pin_file [pwd]/pinfile.modified`. This implies that the `-vendor` and `-pin_file` options should not be used simultaneously.

fastsim_pin_high

Sets a high voltage value for an input or output pin for a fast simulation.

Arguments

<pin>	Specifies the pin name for which a high voltage value is to be set.
<voltage_value>	Specifies the voltage value.

fastsim_pin_low

Sets a low voltage value for an input or output pin for a fast simulation.

Arguments

<pin>	Specifies the pin name for which a low voltage value is to be set.
<voltage_value>	Specifies the voltage value.

get_var_default

Returns the default value of the specified parameter. The parameters are defined using the `set_var` command.

Arguments

<parameter_name>	Specifies the parameter name for which the default value needs to be returned.

help

Displays detailed description of the specified command or parameter in Cadence Help.

Note: This command will work only after you set the path to the Liberate installation directory. For more information, see Invoking Liberate MX Help.

Options

`{`*`command_name`* | *`parameter_name`*`}`

Name of the command or parameter.

`-searchdoc <search_string>`

Displays search results for the specified strings in Cadence Help. Multiple search strings can be provided within double quotes.

Example

```
help -searchdoc "static mode"
```

This command will display search results related to static and mode.

hspice_lis_2_waves

Use this command to have Liberate MX convert HSpice waveforms from a `.lis` file into a data file format that the Liberate MX lwave program can read.

Options

`lis <`*`file`*`>`	The HSpice *.lis* filename.
`-nets <list>`	List of nodes to show. Specify the `-nets <list>` to limit the waveform display to a select number of nets. If `-nets` option is not specified, all nodes available for display in the lwave program.

Example:

```
hspice_lis_2_waves sim.lis
```

interpolate_define_axis_expr

Use this command to apply the specified `axis_expression` only to the specified types during interpolation. The default is to apply the `axis_expression` to all types.

Options

`-expression "string"`

Multiplication expression of compiler axis parameters, `bit*bit` or `bit*word`.

`-type "string"` A list of types that use the axis parameter for interpolation.

Valid types include:

`all cap delay hold leakage power mpw minp retain retain_trans setup trans`

`name "string"` The interpolate axis name.

interpolate_set_axis_exprs

Use this command to apply pin and state dependency to the specified types during interpolation.

Options

`-type {list}` A list of types that use the axis parameter for interpolation.

Valid types include:

`cap delay hold leakage power mpw minp retain retain_trans setup trans`

`-pin {pin}` The output or constrained pin.

`-related_pin {pin}` The input or related pin.

`-when "string"` State dependency.

`-axis_exprs "string"`

A list of axis expression names to be used during interpolation.

`-use_worst_value <boolflag>`

Specifies not to perform interpolation and to use worst value worst value found across all the characterized instances.

Example

define an exit expression `bit*bit` that is not used by any interpolation types:

```
interpolate_define_axis_expr \
    -expression {bit*bit} \
    -type {} \
    bit_bit
```

define an exit expression `word*word` that is not used by any interpolation types:

```
interpolate_define_axis_expr \
    -expression {word*word} \
    -type {} \
    word_word
```

#use `word_word` and `bit_bit` for power on pin A[0]:

```
interpolate_set_axis_exprs -type power -pin {A[0]} -axis_exprs {word_word bit_bit}
```

#use `bit_bit` for power on any pin of bus `A`:

```
interpolate_set_axis_exprs -type power -pin {A} -axis_exprs {bit_bit}
```

#use worst value for hold on bus `A` for a given when condition:

```
interpolate_set_axis_exprs -type hold -pin {A} -related_pin {CLK} -when {PGEN&CEN}
-use_worst_value
```

interpolate_set_default_group

Allows the user to specify how multiple values for the same interpolation group type are merged. By default, the group with the maximum value is used for all types except `retain` and `retain_trans`, where the group with the minimum value is used.

Options

`-type "string"`	A list of types that use specified criteria and method. Valid types include: `all cap delay hold leakage power mpw minp retain retain_trans setup trans` Default: `all`
`-criteria`	Criteria used to set table values when multiple characterized entries are found. By default, maximum value is used except for `retain` and `retain_trans`, where minimum value is used. Valid values are: `default min max`
`-method`	Method used to set table values when multiple characterized entries are found. By default, the table that has the largest or smallest value is used. If bitwise method is specified then the worst value from any table is used. Valid values are: `table bitwise`

interpolate_use_worst_value

Use this command to apply worst-case value to the specified types during interpolation. The default value is `all`.

Option

`-type "string"`	A list of types that use the worst-case value for interpolation. Valid types include: `all cap delay hold leakage power mpw minp retain retain_trans setup trans` The `retain` and `retain_trans` arc types will use the minimum value across all characterized instances. Other arc types will use the maximum value across all characterized instances.

mx_cmdlog

Specifies the Tcl commands that you want to log.

Options

`-cmds <list>`	List of Tcl commands to be logged.
`-mx_default_cmds`	Specifies whether to use the default list of MX commands. Default: `0`
`-delete`	Deletes specified commands from list of commands being logged. Default: `0`

Example:

```
mx_cmdlog -mx_default_cmds
```

The above command will display an output like this:

```
liberate_mx > set_vdd ABC 12
CMD_LOG (Aug. 31, 2017 05:21:26 PM) >>> set_vdd ABC 12
```

mxcore_def

Use this command to specify the custom memory core definitions. For more information, see Specifying Memory Core Cells.

Options

`<mxcore_filename>` The `mxcore` filename.

Example:

```
mxcore_def <cell>.mxcore
```

mx_match_node

Searches the netlist which has been read by `read_spice` and return the nodes satisfied by the filtering options and the pattern or keyword argument. Intersection criteria are handled as nodes that are connected through a corecell (in the case of `bitline`, `wordline` or `core`) or through physical connection (for precharge and senseamp).

Options

`-bitline <bitline_name>`

Filters result of the command by returning only nodes that intersect the specified bitline.

`-wordline <wordline_name>`

Filters result of the command by returning only nodes that intersect the specified wordline.

`-core <core_node_name>`

Filters result of the command by returning only nodes that intersect the specified core node.

`-data_in <data_in_name>`

Filter result of the command by returning only nodes that propagate from the specified data input.

`-senseamp <sense_amp_node_name>`

Filter result of the command by returning only nodes that propagate into the senseamp containing the specified node.

`{pattern_or_keyword}`

Pattern or keyword to use when matching nodes. Pattern is any TCL regexp pattern. Valid keywords are: `bitline`, `wordline`, `core`, `bitline_precharger`, `senseamp_precharger`, `senseamp_enable`.

`{cell name}` List of cells.

Examples:

- To return all bitlines in the instance:

  ```
  mx_match_node bitline <cellname>
  ```

- To return all bitlines that propagate from D<0>:

  ```
  mx_match_node -data_in D<0> bitline <cellname>
  ```

- To return all nodes that contain “BLB”:

  ```
  mx_match_node *BLB* <instname>
  ```

mx_merge

Merges all LDB data to create top-level LDB.

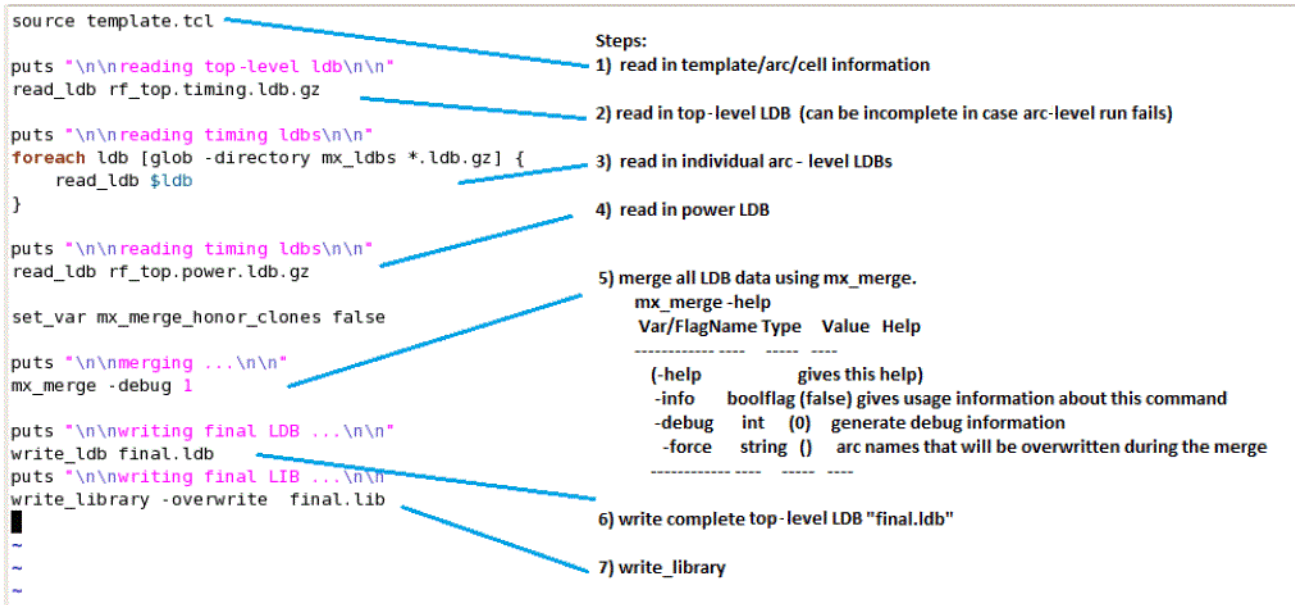
Options

`-debug`	Generates debug information.
`-force`	Arc names that are overwritten during the merge.

Following are some of the scenarios in which the command is used.

- In cases where few arc-level runs failed in the first run and the user only performs an incremental run to create arc-level LDBs. After arc-level LDBs are generated, the user can use `mx_merge` to merge all LDBs and create a top-level LDB.
- User has timing, pincap, and power LDBs available and needs to create top-level LDB and/or top-level library (`.lib`).

The following figure shows the steps of the use-model along with the description of each step.



mx_rampup_check

The threshold of virtual supply voltage to be used for rampup check measurements.

mx_recover_clean

Analyzes characterization data for issues and corrects them, if possible. *(*See *mx_recover_info* for full description of flow.

Options

`-debug`	Prints debug information on cleanup steps.
`{remove_double_groups}`	List of errors to look for / correct.

mx_recover_info

Options

`<no options>`	Outputs a usage statement detailing the commands below.

Liberate MX provides for a recovery flow using the following three commands:

- `mx_recover_setup`: Generates a place holder LDB ready to receive characterization LDBs for subsequent merging.
- `mx_recover_clean`: Analyzes characterization data for issues and possibly corrects them.
- `mx_recover_merge`: Merges characterization LDBs in to final LDB.

The procedure for this flow is to create three separate Tcl scripts, each one containing a separate command (as shown in the examples below). To make a complete flow, create one more script that calls each of these scripts in turn:

```
# mx_recover_setup.tcl
    mx_recover_setup <cell>.ldb.gz
# mx_recover_clean.tcl ... This is optional if cleaning is not needed.
    mx_recover_clean {remove_double_groups}
# mx_recover_merge.tcl
    mx_recover_merge
```

Note: The "clean" step is optional, and may be omitted if there's no need to correct existing characterization LDBs.

mx_recover_merge

Merges (possibly cleaned up) characterization LDBs in to final LDB. (See *mx_recover_info* for full description of flow.)

Options

`-debug`	Debug info on merging step and intermediate merging LDBs.
`-merged_ldb <name>`	User name for resultant LDB

mx_recover_setup

Generates a place holder LDB ready to receive characrization LDBs for subsequent merging. (See *mx_recover_info* for full description of flow.)

Options

`<ldb>`	Full path name to LDB that needs to be regenerated.

mx_report

Generates a series of reports from a regression run.

Options

`-ldb {list}`	Paths to mx reference/compare ldb's - as written by `write_ldb` command in respective runs.
`-lib {list}`	Paths to MX reference/compare libraries - as written by `write_library` command in respective runs.
`-rpt <file_name>`	Name of output report (do not include file name extension.)

The output report is an HTML file (and supporting directories) that can be viewed as-is or can be opened as a Microsoft Excel document containing a workbook with 6 separate sheets:

- Summary
- Lib Compare
- Measurement Compare
- Partitions Compare
- Fast-Spice vs. Full-Spice
- Statistics

mx_report_supply_current

Reports the additional inrush, peak, or average current in the library database (LDB) for the specified cell, arc, and power supply.

Options

`-name <library_name>`

Specifies a name for the library attribute. (REQUIRED)

`-power_type <peak | inrush | average>`

Specifies the type of data to report. (REQUIRED)

It can be one of the following:

- `peak` – Reports the peak current. (Default)
- `inrush` – Reports the inrush current.
- `average` – Reports the average current.

`-supply_pin <supply_pin_name>`

Specifies the power supply pin to be used for evaluation. If not specified, the sum of all power supplies is used.

`-when <when_condition>`

Specifies the when condition for which the current is to be evaluated. By default, all the conditions are included for evaluating the current.

`-table <list_of_truthtables>`

Specifies a list of truth table files to be included for evaluation. By default, all the power table files are included.

`-pin <pin_name>` Specifies the pin name.

`-pin_dir <pin_direction>`

Specifies the pin direction.

Note: This is required if the pin name is specified.

`-cycle_name <list_of_cycle_names>`

Specifies a list of cycle names to be included for evaluation. Even if a cycle name is repeated multiple times within a specified truth table, the average or peak of all of the given cycles is used. By default, all the cycles in the specified truth tables are included for evaluation.

`<cell_name>` Specifies the cell name.

Examples:

The following example shows how to use this command to report the peak current for all the cycles in the specified tables:

```
mx_report_supply_current -name curr_dynamic_peak_vddce -supply_pin vddce
-power_type peak -table $peak_power_tables MY_CELL_NAME
```

The following example shows how to use this command to report the inrush current for all the cycles in the specified table:

```
mx_report_supply_current -name curr_dynamic_inrush_vddce -supply_pin vddce
-power_type inrush -table inrush.tbl MY_CELL_NAME
```

The following example shows how to use this command to report the peak pin current for a specific pin:

```
mx_report_supply_current -name curr_powerdown_PGEN_r_peak_vddpe -supply_pin vddpe
-when PGEN -pin PGEN -pin_dir R -power_type peak -table power_inputs.tbl
MY_CELL_NAME
```

The command in the example given above reports the peak PGEN rise current as PGEN rise transition in the specified table.

Note: When `power_type` is set to `peak`, `-pin` and `-pin_dir` are optional. If `-pin` and `-pin_dir` are not specified, the overall peak current for the specified truth tables is reported.

The following example shows how to use this command to report the average pin current for a specific pin:

```
mx_report_supply_current -name curr_pin_CEN_r_avg_vddpe -supply_pin vddpe -pin CEN
-pin_dir R -power_type average -table power_inputs.tbl MY_CELL_NAME
```

The command in the example given above reports the average CEN rise current over the same cycle where CEN rises in the given table. In case multiple occurences of `pin`, `pin_dir`, and `-when` are found in the given table, the average of all the average cycles is used.

The following example shows how the generated attributes appear in the LDB:

```
altos_mx_report_supply_current : "curr_dynamic_peak_vddce 1.14675
curr_dynamic_inrush_vddce 0.74675 curr_powerdown_PGEN_r_peak_vddpe 0.4
curr_pin_CEN_r_avg_vddpe 0.2;
```

mx_set_constprop

Lists the nodes to start, stop, enable, or force constant propagation.

Options

`<node> <value>}`	Lists nodes and their logic values, where to start constant propagation from. The `node` is the pin, bus, or bus range that must be considered during constant propagation.

If a bus or a partial bus is specified, its constant value must be specified in hexadecimal notation. Constant propagation is not a required step but helps in various spatial analysis steps during the preprocessing step. Once a value is specified, via the `mx_set_constprop` command for a pin or a bus, that same value is used during the FastSPICE simulation run and therefore does not need to be specified again in the stimuli or truth table.

Example:

```
# Set test mode pins to '1000' value
mx_set_constprop {{tm[3:0] 0x8}}
```

mx_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 combinational logic gates until it reaches a sequential element that has a non-clocked pin.

See also mx_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
mx_set_domainprop {\
    {A stop CLK} {B start CLK} \
    {C force CLK} {D enable CLK}\
}
```

mx_set_spectre_param

This command is used to pass parameters to the Spectre simulator.

Options.

<list>	Passes a list of Spectre parameters as name-value pairs. The specified parameters are used during Spectre simulation.

Note: Parameters specified in a table will override parameters that are specified with a Tcl command.

Example:

```
mx_set_spectre_param {\
        {soft_bin allmodels} \
}
```

mx_set_ultrasim_param

These commands are used to pass parameters to the appropriate external simulator.

Options.

`<list>`	Passes a list of UltraSim parameters as name-value pairs. The specified parameters are used during UltraSim simulation.

Note: These parameters can also be specified in a table format using hspice_lis_2_waves. Parameters specified in a table will override parameters that are specified with a Tcl command. (See fastsim_deck.) For example:

```
mx_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} \
}
```

mx_utils_sort_mx_info

Sorts contents of the `mx_info` file by arc types specified in `define_arc -type`.

Options

`-mx_info`	Sorts the contents of the `mx_info` file.
`-keep_orig`	Renames the original `mx_info` file as `mx.info.ORIG` and creates a new sorted `mx_info` file. Default is `false`.

report_measure

Reports the difference between relax or stress simulation results.

Options

`-all`	When set to `true`, reports all activities for both relax or stress run. When `false`, reports only failed activities. Default: `false`.
`-abs_diff <value>`	Reports absolute differences of measure values between relax/ stress simulation results. The absolute error is bigger then `rel_diff`. Default is `1n`.
`-rel_diff <value>`	Reports relative differences of measure values between relax/ stress simulation results. The relative error is bigger then `abs_diff`. Default is `0.05` (which is equal to 5%).
`measname <string>`	Measure name to be validated.
`cell <string>`	Cell names.

report_node

Creates a comparison report for the node validated with the validate_node command.

Options

`-bitline <string>`	Filters result of the command by returning only nodes that intersect the specified bitline.

`-wordline <string>`	Filters result of the command by returning only nodes that intersect the specified wordline.
`-core <string>`	Filters result of the command by returning only nodes that intersect the specified core node.
`-data_in <string>`	Filters result of the command by returning only nodes that intersect the specified data input.
`-senseamp <string>`	Filters result of the command by returning only nodes that intersect the specified senseamp.
`-all <boolflag>`	Controls what activities should be reported. Default: `false` ■ `true`: Reports all activities. ■ `false`: Reports only failed activities.
`-showwave <boolflag>`	Controls whether to display waveform. Default: `false`
`-idx1 <boolflag>`	Specifies whether to perform the node activity comparison per clock cycle. Default: `false`
`-check_relax <boolflag>`	Reports truth table and relax run difference. This option is supported for `output` nodes only. Default: `false`
<*pattern*>	Pattern or keyword to be used when matching nodes or wirenames. Any Tcl regexp pattern can be specified. Default: none Valid keywords are `bitline, wordline`, `core`, `bitline_precharger`, `senseamp_precharger`, `senseamp_enable`, and `output`.
<*cellname*>	The cell name in which the node activity comparison was performed.

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 mx_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 MX 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, for example in a fully RC extracted netlist, the mx_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 mx_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 MX automatically identifies the following net names (case-insensitive) as power supplies and sets them to the default voltage specified by the `set_operating_condition` command: VDD and VCC. Use the `set_vdd` command to set them to alternative values.

set_pin_gnd

See set_pin_vdd for description of options.

Options

`-add_supply`	Notifies Liberate to create a new supply with the name specified using the `-supply_name` option or an arbitrary internal supply name if `-supply_name` is not specified (and if the supply does not exist already.) **Note:** It is recommended to define all supplies using the set_gnd and set_vdd commands. The `-add_supply` option is available only for backward compatibility and should not be used.
`-supply_name <name>`	Name of the supply that drives this pin. (REQUIRED)
{*cell_names*}	List of cell names. (REQUIRED)
{*pin_names*}	List of pin names. For example, {a1 a2 a3 c din ckn}, or regexp like {a* c*}. (REQUIRED)
<*gnd_value*>	Ground supply value. (REQUIRED)

set_var

Use this command to set parameters that are specific to Liberate MX.

Arguments

`-cells`	List of cells. Default: all cells
`-pin {`*`pins`*`}`	List of destination pins for the arc (typically, output pins for combinational arcs, input pins for timing constraint, or hidden power arcs). (REQUIRED)
`-pin_dir <R | F>`	Transition direction of pin(s).
`-pvt`	List of PVT names for which the parameter is applicable. Use of wildcard characters is allowed.
`-related_pin {`*`pins`*`}`	List of related pin names (typically input pins for combinational arcs, clock pins for timing constraint arcs).
`-related_pin_dir <R | F>`	Transition direction of related pin(s).
`-type < constraint | delay | delay and power | hold | leakage | mpw | nochange | power | recovery | removal | setup >`	Type of arc (Default: *all types*)
<*`name`*>	Parameter name (REQUIRED)
<*`value`*>	Parameter value (REQUIRED)

The options `-cells`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-related_pin_dir` are used to specify local cell and arc specific parameters and their corresponding values. All options are not valid for all parameters. If an option is not allowed, an error is issued and the setting is ignored. If an option is omitted, any value for that option is allowed. The options `-cells`, `-pin`, and `-related_pin` support the usage of the wildcards `*` and `?`. Some parameters can only be set at a global level. If you specify local cell and arc parameter that can only be applied globally, a warning is issued in the log file and `set_var` is ignored.

The available Liberate MX variables are defined in Chapter 6, “Liberate MX Parameters.”

Example

```
set_var -cells DFF* write_logic_function false
```

set_pin_vdd

The `set_pin_vdd` and **set_pin_gnd** commands associate a *pin* of a cell with a particular supply domain. This is particularly useful on cells that have multiple power and ground connections, such as level shifters where it may be difficult for Liberate to correctly identify the supplies for each pin. The cell and pin options accepts a list of names. The cell and pin names may be specified with wildcard characters `*` and `?` and `regexp` expressions. In the event that multiple commands are given which map to the same cell and/or pin, then the last command given takes effect.

Options

`-add_supply` — Notifies Liberate to create a new supply with the name specified using the `-supply_name` option or an arbitrary internal supply name if `-supply_name` is not specified (and if the supply does not exist already.)

Note: It is recommended to define all supplies using the set_gnd and set_vdd commands. The `-add_supply` option is available only for backward compatibility and should not be used.

`-leakage_add_to_supply <name>`

Specifies the name of a supply to which all leakage for this supply should be added. This option should be used when gate leakage on an input pin is to be added to a power pin not controlling it. This is useful when characterizing level shifters where the input is driven by a supply domain that does not exist in the cell and you do not want to lose the input power.

`-supply_name <name>`

Specifies the name of the supply associated with the specified pin. (REQUIRED)

`{cell_names}` — List of cells. (REQUIRED)

`{pin_names}` — List of pin names. For example, `{a1 a2 a3 c din ckn}`, or wildcard pattern matching like `{a* c*}`. (REQUIRED)

<vdd_value> Power supply value (Optional)

When you specify the `supply_name` but do not specify the `vdd_value` and the supply exists in the vdd list (see set_vdd), then the vdd voltage value from the `set_vdd` command is used. If you specify both `supply_name` and `vdd_value`, ensure that the vdd value is the same as the one supplied in the `set_vdd` command. Else, an error message is generated and the command is ignored.

1. If there are more than one set_pin_vdd commands for the same pin but with a different value or different supply name, then the second instance of the command overwrites the first.

2. If there are two supplies with different names but with the same voltage value, and a set_pin_vdd command associates a net to this voltage but the supply_name is not given, Liberate reports an error and exits (after reporting all similar errors.)

Example error cases:

```
# -supply_name not given:
set_vdd VDD1 1.2
set_vdd VDD2 1.2
set_pin_vdd NAND2 Y 1.2

# Attempt to redefine vdd2 voltage:
set_pin_vdd -supply_name vdd2 busDriver PAD 3.3
```

This command must be used before char_macro.

Examples

Example 1

```
# Set the voltage swing on the input pin of a level shifter
set_pin_vdd -supply VDD3 level_shifter_3to1 A1 3.0
```

Example 2

```
set_vdd VDD $VOLT
```

```
set_vdd VDD1 $VOLT1
set_pin_vdd -supply_name VDD1 -leakage_add_to_supply VDD \
    LVLHLD1HVT   IN    $VOLT
```

In the above example, gate leakage currents on pin IN will be multiplied by $VOLT1 (controlling VDD1) to get the leakage power that will be added to the supply pin VDD.

set_virtual

Specifies the virtual ground and power nodes. Use this command for virtual net definitions, but the selection of a virtual net is controlled by the mx_virtual_rail_opposite_device_minimum_factor parameter.

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: In earlier releases, the virtual net definitions functionality was used by the `-ignore_power`, `-nomodel`, and `-virtual` arguments of the `set_vdd` and `set_gnd` commands.

spv_tcl_2_waves

Helps debug FastSPICE waveforms when tcl waveform file is too large to be loaded from TCL interpreter.

Options

`-debug` Print debug info

`waveforms_file <string>`

Specify waveforms file name

`nets { list of nets }`

Instructs the tool to require activity information be available for memory core bits. Default: `0`

The command can be used to display waveforms using the Liberate MX lwave utility for the list of nets as stored in file 'waveforms_file' that is output bye the FastSPICE simulation (when `mx_spv_api` is `1`).

unset_var

Resets the specified parameter to its default value. This command is equivalent to setting `set_var <parameter_name> [get_var_default <parameter_name>]`.

Arguments

`<parameter_name>`	Specifies the parameter name for the default value needs to be reset.

validate_macro

Performs memory validation. Each memory that has a `define_cell` command and a netlist loaded using the `read_spice` command is validated.

Options

`-validsim "`*`simulator_name`*`"`

Specifies which external simulator to be used for validation. Supported simulators are: `"aps"` and `"xps"`. Default: `"xps"`

`-thread <number>`

Specifies the maximum number of threads to be used on the host machine. When set to `0`, Liberate MX automatically uses multiple threads up to the total number of available CPUs on the host. The following steps support multithreading: FastSPICE simulation, results acquisition, arcs selection, and file IO operations. The number of parallel FastSPICE runs is determined by the maximum number from among the different tables files provided and the number specified by the thread argument. Default: `0` (Let Liberate MX decide). For more information, see Specifying Input Stimuli.

Examples:

```
# Validate memory(s) defined via a previous define_cell command.
# Use Spectre APS for validation.
# Validation using 2 threads
validate_macro -validsim "aps" -thread 2
```

For more information on library validation, see Appendix E, “Liberate MX Timing Validation Flow.”.

validate_measure

Validates the measure defined by `define_measure`.

Options

`measname <string>`	Measure name to be validated.
`cell <string>`	Cell name.

validate_memory

Creates all needed validation files when used after the define_memory command in the Liberate MX validation flow.

Options

`-workdir "string"`	The path where the `mx_setup` directory containing the reusable setup files is created. Default: `mx_setup`

This command can be used to run the validation on an instance of standard functionality in automated standard instance flow. For more information, see Appendix F, “Basic Flow for Validating User-Defined Criteria.”

validate_node

Compares activity of the specified nodes between the relaxed simulation and the stress simulation in the Liberate MX validation flow.

Options

`-bitline <string>`	Filters result of the command by returning only nodes that intersect the specified bitline.
`-wordline <string>`	Filters result of the command by returning only nodes that intersect the specified wordline.
`-core <string>`	Filters result of the command by returning only nodes that intersect the specified core node.

`-data_in <string>` Filters result of the command by returning only nodes that intersect the specified data input.

`-senseamp <string>` Filters result of the command by returning only nodes that intersect the specified senseamp.

`-glitch_height <float>`

Specifies minimum height for a glitch to be considered.
Default: `-1`
Valid range: `0~0.5vdd`

`-glitch_height_delta <float>`

Reports a glitch if (`abs(glitch(a)-glitch(b))> glitch_height_delta`.
Default: `-1`
Valid range: `0~0.5vdd`

Notes:

- The value of `glitch_height_delta` should be smaller than that of `glitch_height`.
- If both relax and stress glitch values are greater than `glitch_height`, the `glitch_height` values of relax and stress will be compared. If the glitch delta between relax and stress is greater than `glitch_height_delta`, outlier is reported. If one of the relax or stress glitch value is larger than `glitch_height` or both are smaller than `glitch_height` value, then `glitch_height_delta` is not compared.

Example:

```
cell.validation.tbl,     BL[0],          bitline
,          Relaxed:        H,       G,       G,
,          At Speed:       G,       G,       G,
,         Diff   :         ^        ,        ^
```

Here,

In the first line, only one of relax/stress (at speed) has glitch. Therefore, outlier will be reported at the mark of “^”;

In the second line, both relax and stress has glitch, but glitch delta between the two is smaller than `glitch_heigh_delta`. Therefore, it will be passed.

In the third line, both relax and stress has glitch and the glitch delta between the two is larger than `glitch_heigh_delta`. Therefore, outlier will be reported at the mark of “^”.

`-ongrid` Controls which wires should be compared. Default: `false`

- `true`: Compares the wires on clock intervals.
- `false`: Compare wires without grid.

<pattern>	Pattern or keyword to be used when matching nodes or wirenames. Any Tcl regexp pattern can be specified. Default: `output` Valid keywords are `bitline, wordline`, `core`, `bitline_precharger`, `senseamp_precharger`, `senseamp_enable`, and `output`.
<cellname>	The cell name in which the comparison is to be performed.

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.
`-overwrite`	Disables the automatic version control, and if the output library already exists, it is overwritten.
`-rename`	Renames an existing LDB file. The default is to not rename the previous LDB and create a unique file name for the new LDB instead.

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 argument.

`-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.

`-cell_cleanup <gzip | delete>`

Runs a cell-level library cleanup utility after successful generation of full libraries in MPVT or arc packet flow (`write_library_mode=1`). If this option is not specified, by default, all cell-level libraries are saved and preserved. The option supports use of the following values:

- `gzip`: Compresses cell libraries under the LIBS directory. This method uses less drive storage, but requires longer access time for future `write_library` operations.
- `delete`: Removes cell-level libraries after full libraries have been generated successfully. Uses the least disk space, but requires the most time for future `write_library` operations.

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

`-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.

`-em` Include electromigration 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 on how to handle failed characterization data when writing to the library. Default: "" (`data` for Liberate MX, Liberate AMS, Liberate Variety and `none` for other products)

The following are the supported values:

- `none`: Does not remove any failed data. See the parameters `mark_failed_data`, `mark_failed_data_replacement` and `constraint_failed_value` for 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.

`-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`.

`-unique_pin_data` Outputs data, such as unique timing, power, and more for each bus bit or bundle member. Default is to use one pin to represent the data for all the bus or bundle pins.

`-unique_pin_data_list`

Sets `-unique_pin_data` only for the list of named buses.

`-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.

Note: You may see unexpected behavior if the `read_ldb` > `write_library` sequence is followed in Liberate MX.

`-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` parameter is set to `1`. If not set, it will force the setting. This parameter 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_memory`, or `read_ldb` command. When this parameter 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` parameters 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_memory`, `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"`

6

Liberate MX Parameters

This chapter describes the following Liberate MX specific parameters that impacts memory library creation.

Note: Liberate MX specific parameters are set using the `set_var` command.

To access officially supported context-sensitive help information on a command or a parameter from within the tool, follow the procedure covered in the Invoking Liberate MX Help.

To review tool-wise support information about each command and parameter available in the Liberate characterization portfolio, see *Liberate Characterization Portfolio Command and Parameter Support Matrix*.

a...	
add_margin_info	
c...	
ccs_from_ecsm_accurate_mode	constraint_probe_lower_rise
ccs_from_ecsm_linear_interp_factor	constraint_probe_hold_lower_rise
ccs_from_ecsm_smooth_mode	constraint_probe_setup_lower_rise
constraint_glitch_peak	constraint_probe_upper_fall
constraint_glitch_peak_max	constraint_probe_hold_upper_fall
constraint_glitch_peak_mode	constraint_probe_setup_upper_fall
	constraint_probe_upper_rise
constraint_probe_lower_fall	constraint_probe_hold_upper_rise
constraint_probe_hold_lower_fall	constraint_probe_setup_upper_rise
constraint_probe_setup_lower_fall	
d...	
driver_cell_trim_miller	

e...	
extsim_deck_include	extsim_model_include
f...	
fastsim_cmd	fastsim_cmd_option
l...	
lic_max_timeout	lic_queue_timeout
mx...	
mxtable_dontcare_value	mxvw_min_glitch_peak
mx_a...	
mx_accuracy_setting	mx_auto_char_params
mx_active_fanout_channel_include	mx_auto_minp_eq_mpw_h_plus_l
mx_active_load	mx_automeas_filter
mx_active_load_channel_thresh	mx_autoprobing_hold_level
mx_active_load_gate_thresh	mx_autoprobing_setup_level
mx_arc_report	
mx_b...	
mx_bisection	mx_bitline_probe_threshold
mx_c...	
mx_char_bundle_size	mx_clone_if_uda
mx_char_virtual_as_rail	mx_clone_if_uda_singlebit
mx_check_arcs	mx_const_prop
mx_check_arcs_exit_on_missing	mx_constraint_ocv_factor
mx_check_arcs_exit_on_missing	mx_corecell
mx_clock2clock_constraints	mx_cp_cmd_preserve_date
mx_clk2clk_mode	mx_create_if_uda
mx_clock_tree_report	
mx_d...	
mx_debug	mx_distributed_retry_enable
mx_defmem_sdf_cond	mx_distributed_simulation_timeout

mx_delay_ocv_factor	mx_distributed_status_cmd
mx_dir	mx_distributed_sim
mx_distributed_kill_cmd	mx_domain_propagation
mx_distributed_pending_timeout	mx_dpartition_inactive_tie
mx_distributed_retry_count	mx_dynamic_include_full_core
mx_e...	
mx_extsim_process_netlist_cmd	
mx_f...	
mx_fastsim_auto_ic	mx_find_memcore_numbit_threshold
mx_fastsim_clock_slew	mx_find_memcores
mx_fastsim_deck_pwl_precision	mx_find_stack_loads
mx_fastsim_init_virtual_rails	
mx_fastsim_input_slew	mx_find_virtual_rails
mx_fastsim_input_slew	mx_fix_interpolate_tran_mode
mx_fastsim_load	mx_fix_pin_vdd
mx_fastsim_parallel_wait	mx_force_arc_ignore_autoprobing_level
mx_fastsim_reuse	mx_full_rail_tol
mx_find_arrays	mx_fsdb_output_name
mx_g...	
mx_gen_nominal_lvf_mode	mx_greybox_constraint_method
mx_greybox	
mx_i...	
mx_info_file_print_probe_threshold	mx_intersect_domains_check_intra_ccc_dynamic_propagation
mx_inputcap_ldb_reuse	
mx_l...	
mx_ldbs_reuse	mx_leakage_measure_final
mx_leakage_check_time	mx_lvf_monte_entries_per_thread

mx_m...	
mx_margin_report	mx_min_period_pulse_targ_fall_thresh
mx_mcf	mx_min_tran_value
mx_measure_error_file	mx_monitor_memcore
mx_measure_distribution_mode	mx_min_risefall_retain
mx_minp_single_slew	mx_monitor_memcore_lprobe_level
mx_min_clock_tree_mode mx_max_clock_tree_mode	mx_mpw_allow_same_probe_on_both_rise_and_fall_clock_tree
mx_merge_arc_packet_ldbs	mx_mpw_false_probe_delay_threshold
mx_merge_multi_slew_load_mode	mx_mpw_measurement_duration
mx_min_period_latch_component_mode	mx_mpw_mode
mx_min_period_mode	mx_mpw_mode
mx_min_period_pulse_trig_rise_thresh	mx_mpw_probe
mx_min_period_pulse_targ_rise_thresh	mx_mpw_probe_lower_fall mx_mpw_probe_lower_rise mx_mpw_probe_upper_fall mx_mpw_probe_upper_rise
mx_min_period_pulse_trig_fall_thresh	mx_mxtable_interpret_read_write_cycle_keywords
mx_n...	
mx_negedge_clock	mx_noise_ldb_reuse
mx_o...	
mx_output_require_fullrail_switch	
mx_p...	
mx_partition_name_use_arc	mx_power_window_ratio
mx_pathdelay_hold_clock_margin	mx_power_single_point
mx_pathdelay_hold_data_margin	mx_power_use_define_index_clock_slew
mx_pathdelay_margin_mode	mx_preprocess
mx_pathdelay_save_data	mx_probe_peak_currents
mx_pathdelay_setup_clock_margin	mx_probe_real_rails_as_virtual

mx_pathdelay_setup_data_margin	mx_probes_report
mx_pincap_char	mx_process_probe_relprobe_intersection
mx_pincap_report_mode	
mx_posedge_clock	mx_process_probe_relprobe_intersection_max_paths
mx_postprocess_probe_relprobe_intersection	mx_process_probe_relprobe_intersection_max_depth
mx_power_assign	mx_push_probe_inside_array
mx_power_divide_num_switching_mode	mx_pulse_width_threshold
mx_power_ldb_reuse	
mx_r...	
mx_read_spice_exit_on_missing_file	mx_ring_large_ccc_max_paths
mx_remove_false_ic_group	mx_ring_large_ccc_max_path_depth
mx_remove_rc_pincap	mx_ring_large_ccc_min_pass_gate_xtr_cnt
mx_remove_rc_timing	mx_ring_model_fold
mx_report_starnet	mx_ring_max_side_inputs
mx_retaining_time	mx_ring_max_xtr_cnt
mx_require_whitebox_model_file	
mx_s...	
mx_scale_virtual_noise_fraction	mx_spv_api
mx_seq_probing	mx_status_log_file
mx_setup_seq mx_hold_seq mx_setup_comb mx_hold_comb	mx_sst2_output_name
mx_simulation_interval	mx_switch_wire_threshold
mx_skip_autoprobing	mx_switch_wire_threshold_ratio
mx_skip_print	
	mx_switch_wire_threshold_ratio
mx_t...	
mx_timing_ldb_reuse	mx_total_active_load_channel_thresh
mx_timing_report	mx_total_active_load_gate_thresh

mx_timing_report_debug	mx_transres
mx_u...	
mx_update_constraint_pin_threshold	mx_use_fastsim_value_for_partition
mx_use_complex_ccc	
mx_v...	
mx_validation_fastsim_slew_load_off_grid_action	
mx_validation_auto_report_mode	mx_virtual_rail_opposite_device_minimum_factor
mx_validation_shift_idx	mx_virtual_rail_minimum_xtrs
mx_verbose	mx_virtual_rail_modeling_mode
mx_verify_table	mx_virtual_rail_stack_xtrs
mx_virtual_rail_auto_mode	mx_virtual_rail_waveform_shift_threshold
mx_w...	
mx_whitebox_active_coupling_threshold	mx_whitebox_model_file_format
mx_whitebox_active_wire_threshold	mx_whitebox_ring_in_depth
mx_whitebox_model_file	mx_write_greybox_dot_meas_file
mx_z...	
mx_zip_partition_deck	
mx_x...	
mx_xps_inc_str	
p...	
packet_arc_job_retries	packet_arc_xtor_count_based_cell_sort
s...	
set_var_failure_action	setup_constraint_probe_upper_fall
set_virtual_rail_threshold	setup_constraint_probe_upper_rise
setup_constraint_probe_lower_fall	supply_info
setup_constraint_probe_lower_rise	
v...	

virtual_rail_waveform_probe	

add_margin_info

`<0 | 1>` Controls whether an `add_margin` report should be generated while applying the user-specified `add_margin` commands. Default: `0`

- `0` Applies the user-specified `add_margin` commands without generating an `add_margin` report.
- `1` Applies the user-specified `add_margin` commands and generates an `add_margin` report.

  The report is saved by filename, `add_margin.log`. It contains the formulas and computations that explain how the data in the margined library was computed. The report can be useful to understand the application of user-specified additional margin.

This parameter must be set prior to generating a library with the `write_library` command.

ccs_from_ecsm_accurate_mode

`<1 | 2>` Specifies the mode to be used to create voltage waveform from ECSM.
Default: `1`

- `1` Creates voltage waveform from ECSM with time step 1e-6s. If ECSM waveform has a fast slope, conversion to CCS may not be accurate.
- `2` Creates voltage waveform from ECSM with voltage step 0.001. If time step is larger than 1e-6s, the tool adds more points.

ccs_from_ecsm_linear_interp_factor

`<value>` Enhances the ECSM to CCS conversion process. This parameter adds linear points before the voltage curve fitting to avoid unusual time step that causes the curve fitting to fail or accuracy issues if the following condition is met:

```
ecsm time step >  ccs_from_ecsm_linear_interp_factor *
(trans/(measure_slew_upper_threshold -
measure_slew_lower_threshold))
```

For example, if the user-defined slew threshold is 0.25~0.75, then the tool will get `ecsm time step> ccs_from_ecsm_linear_interp_factor*(trans/0.5)`, and perform linear interpolation.

ccs_from_ecsm_smooth_mode

`<0 | 1 | 2>` Checks if there is a large time step with miller voltage change. If found, the sharp segmentation is smoothened.
Default: `0`

- `0` The check is disabled.
- `1` Smoothens sharp segmentation only.
- `2` Recreates the waveform based on ecsm_13pts + delay + upper + lower threshold list.

constraint_glitch_peak

<value>	Glitch height as a ratio of supply used in characterizing timing constraints (`setup`, `hold`, `recovery`, `removal`). Default: `0.1` (`10%`)

Use this parameter to specify the maximum size of logic glitch permitted on the constraint output pin before an arriving signal is deemed to fail a timing constraint.

The `set_constraint_criteria` command can also be used to set this parameter. If both this parameter and the `set_constraint_criteria` are used, the last one executed sets the value to be used by Liberate MX.

This parameter must be used before `char_macro`.

constraint_glitch_peak_max

<value>	Specifies the maximum threshold for `constraint_glitch_peak_mode`. Default: `0.5`

If the new threshold from using `constraint_glitch_peak_mode` exceeds the ratio of this parameter times vdd, then the threshold is limited to the voltage represented by the ratio of vdd specified by this parameter. This can result in a search bound error.

This parameter must be used before `char_macro`.

constraint_glitch_peak_mode

`<0 | 1 | 2>` Apply `constraint_glitch_peak` on top of inherent glitch. Default: `0`

- `0` Do not apply `constraint_glitch_peak` on top of inherent glitch. (Default)
- `1` Liberate measures the inherent glitch magnitude (noise on the net) and then add that to the `constraint_glitch_peak` to use as a new threshold. If the new threshold exceeds `constraint_glitch_peak_max`, the threshold is limited to `constraint_glitch_peak_max`. This helps prevent warnings about "Too close to search bound" for glitch-based constraint measurements in the Liberate log file. (Recommended)
- `2` Operates the same as option 1, but pertains to internal nodes. Only clock-gater hold results are impacted in an entire library. Non-sequential setup/hold are not affected as long as `constraint_async_probe_internal` = `0`.

Many cells exhibit inherent glitches immediately upon clock transition. This is a glitch that occurs on a node as a direct result of the clock switching and is not related to any race condition between data and clock. If this inherent glitch occurs at a node that Liberate identifies as the probe node, then the logfile contains the warning message "Too close to search bound". If the constraint measurement criteria is glitch, then it is possible that an inherent glitch occurs on the probe node. Setting `constraint_glitch_peak_mode` can work around this by accounting for the inherent glitch.

This parameter must be used before `char_macro`.

constraint_path_delta_probe_mode

`<0 | 1 | 2>` Determines the probe threshold when path-delta probe pin is equal to input pin. Default: `1`

- `0` When path-delta probe pin = input pin, always use a probe threshold of `0.5`.

  Note: For backward compatibility only.

- `1` When path-delta probe pin = input pin, allow `define_arc` probe threshold parameters to take effect.

  Note: For Liberate only.

- `2` Similar to `1`, enables constraint arc mode. The constraint probe or rel_probe threshold take effect when there is no logic delay.

  Note: For Liberate MX and Liberate AMS only.

constraint_probe_lower_fall

`<value>` Specifies the lower fall threshold for constraint probe measurements. Default: `0.3`

constraint_probe_hold_lower_fall

`<value>` Specifies the lower fall threshold for constraint probe measurements. Default: `0.3`

constraint_probe_setup_lower_fall

`<value>` Specifies the lower fall threshold for constraint probe measurements. Default: `0.3`

constraint_probe_lower_rise

`<value>` Specifies the lower rise threshold for constraint probe measurements. Default: `0.3`

constraint_probe_hold_lower_rise

`<value>` Specifies the lower rise threshold for constraint probe measurements. Default: `0.3`

constraint_probe_setup_lower_rise

`<value>` Specifies the lower rise threshold for constraint probe measurements. Default: `0.3`

constraint_probe_upper_fall

`<value>` Specifies the upper fall threshold for constraint probe measurements. Default: `0.7`
Valid values are between `.02` and `.98`.

constraint_probe_hold_upper_fall

`<value>` Specifies the upper fall threshold for constraint probe measurements. Default: `0.7`
Valid values are between `.02` and `.98`.

constraint_probe_setup_upper_fall

`<value>` Specifies the upper fall threshold for constraint probe measurements. Default: `0.7`
Valid values are between `.02` and `.98`.

constraint_probe_upper_rise

`<value>` Specifies the upper rise threshold for constraint probe measurements. Default: `0.7`

constraint_probe_hold_upper_rise

`<value>` Specifies the upper rise threshold for constraint probe measurements. Default: `0.7`

constraint_probe_setup_upper_rise

`<value>` Specifies the upper rise threshold for constraint probe measurements. Default: `0.7`

driver_cell_trim_miller

`<0 | 1 | 2>` Enables filtering of nonmonotonic behavior from the waveform. Default: `0`

When an active driver is used (see `set_driver_cell`) to create an input waveform, the waveform may be nonmonotonic due to effects such as the miller capacitance. In this scenario, use the `driver_cell_trim_miller` parameter to filter the nonmonotonic behavior.

- `0` Do not apply any filtering. Use the waveform produced by the active driver.
- `1` Apply filtering to remove nonmonotonic behavior from the input waveform. This setting should only be used for backward compatibility to the LIBERATE 16.1 ISR4 and prior releases to match existing libraries.
- `2` At times, the active driver cell is not able to produce a waveform at the fast slews. In these cases, Liberate adjusts the fastest slew that the driver can create to meet the required fast slews in the slew index. This setting ensures creation of a proper waveform in the rare cases where the active driver is not fast enough. If filtering of nonmonotonic behavior is desired, the setting of 2 is recommended.

This parameter must be set before the `char_macro` command is run.

extsim_deck_include

`<0 | 1 >` Controls how the FastSPICE simulation deck is written out. Default: `1`

- `0` Reports the full netlist into the deck.
- `1` Only the original netlist is included via an `.inc` statement.

Use this command to control how the FastSPICE simulation deck is written out.

Example:

```
#to .include original netlist use by FastSPICE
#simulation.
set_var extsim_deck_include 1
```

extsim_model_include

`<value>` Specify full path to a file that will load the SPICE models. Default: `Use flattened models`.

Use this parameter to specify a full path to a file that will load the models when using an external SPICE simulation engine. If a full path is not provided, an error will occur. Normally, Liberate MX uses flattened models in the external simulation input decks. However, when this parameter is used, Liberate MX uses the file specified instead of the flattened models in the external simulation input deck. Liberate MX places a statement in the extsim SPICE decks such as:

```
.include <extsim_model_include_file>
```

For 40nm and below, the recommended flow is to use all three — `extsim_model_include`, `extsim_deck_include`, and `define_leafcell`.

Example:

```
set_var extsim_model_include "/home/user1/models/include_ff"
set_var extsim_deck_include 1
```

Where `include_ff` looks like:

```
.include '/home/user1/models/models.l' ff
```

This parameter must be used before `char_macro`.

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.

lic_max_timeout

`<value>`	Specifies the duration, in seconds, to wait for each license feature or token before skipping and checking with the next possible license feature or token. Default: `86400`

This parameter only works when the shell environment parameter `ALTOS_QUEUE` is set to `1`. For more information, refer to *LIBERATE Software Licensing and Configuration Guide*.

The `ALTOS_LIC_MAX_TIMEOUT` shell environment parameter will override the value set by this parameter in the Tcl file.

This parameter must be used before `char_macro`.

lic_queue_timeout

`<value>`	Specifies the duration, in seconds, to wait for the required licenses to be acquired. Default: `60` (seconds)

The `ALTOS_LIC_CHECK_ALT_TIMEOUT` shell environment parameter will override the value set by this parameter in the Tcl file.

This parameter must be used before `char_macro`.

mxtable_dontcare_value

`{<bus_name | pin_name>` Sets a default table entry for pin or bus. Default: `none <0 | 1}`

This parameter sets a default table entry value for a pin or bus. parameter 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 "dont_care" (default) value is used. See also Appendix A, “Truth Table Format.”

Example:

```
set_var mxtable_dontcare_value {oe 0 ctrl 0xc}
```

mxvw_min_glitch_peak

`< value>` The value range for this parameter is 0.0 to 1.0. Default: `0.1`

This parameter is used in mx validation flow to identify a glitch in waveform comparison. It is used to define the glitch by specifying the ratio of vdd between 0 to100 percent. This parameter is applied on glitch above gnd, glitch under vdd, undershoot under gnd, and overshoot over vdd.

mx_accuracy_setting

```
< inactive_node_voltage | inactive_node_remove_rc |
ring_side_input_tie | side_load_xtr | inactive_node_use_mean_value
| measure_core_expand_partition_to_full_row |
max_drop_virtual_probe | active_wordline_membit_load |
update_multi_thread_settings | virtual_probing >
```

Specifies the various accuracy settings. Default: `default`

`active_wordline_membit_load`

Includes memory cell as active load.

`inactive_node_voltage`

In Liberate MX, sets inactive wire value to closest rail (vdd or ground).

`inactive_node_remove_rc`

Removes RC on the inactive node.

`inactive_node_use_mean_value`

Sets voltage source with the mean value in the subinterval range.

`max_drop_virtual_probe`

Enables selection of virtual probes having maximum drop.

`measure_core_expand_partition_to_full_row`

If the target of the dynamic partition is a wordline or a memory core, adds all other memory cores on the same row to the traversal.

`ring_side_input_tie`

Ties side input based on the circuit structure and behavior.

`side_load_xtr`

Models side load transistors with active devices.

`update_multi_thread_settings`

Enables extra license checks/checkout in Liberate MX.

`virtual_probing`

Selects different virtual probes (having maximum drop) in each sidx for getting better accuracy.

mx_active_fanout_channel_include

`< none | !memcore >` Determines how the probe node is loaded for modeling purposes. Default: `none`

`none` Load the probe with the transistors directly driven, and model the channel connected terminals with equivalent caps. (Default)

`!memcore` Set this to implement the probe loading scheme of release 3.1 and earlier.

May cause a degrade in performance – use for backward compatibility only

Intended for backward compatibility only. This parameter controls how the probe node is loaded for modeling purposes. Releases 3.1 and earlier employed a scheme that loaded the node in a recursive fashion, including the full active channel (except for memory core nodes.) This caused a degrade in performance for no appreciable gain in accuracy. This parameter should be set to `none` to implement the more efficient modeling scheme of version 3.1p1.

Example:

```
set_var mx_active_fanout_channel "!memory"
```

mx_active_load

<value>	Controls loading on a per-net basis. Valid values are listed below. Default: none	
	all	Load any node with active devices.
	clock	Load clocks with active devices.
	none	Load any (non-probe) node with equivalent passive loads (Default)
	wordline	Load wordlines with active devices.
	net_name	Load specified net name with active devices.

Controls loading on a per-net basis. Note: this is only useful to achieve correlation on internal margin measurements and should not be used for regular library characterization.

Example:

```
# Set active load on clocks, wordlines, and signal "sig1"
set_var mx_active_load "clock wordline sig1"
```

mx_active_load_channel_thresh

`<value>`	Simplifies a partition by substituting a passive load for active devices. If the channel equivalent load of a device is larger than the specified threshold, the substitution is not performed. Default: `1`

mx_active_load_gate_thresh

`<value>`	Simplifies a partition by substituting a passive load for active devices. If the gate equivalent load of a device is larger than the specified threshold, the substitution is not performed. Default: `1`

mx_arc_report

`<filename>`	Specifies file where failed arcs are reported. Default: `"mx_dir/arc.rpt"`

Arcs that are found during partitioning but fail to check against user-defined arcs are reported to this file. Also reported are arcs found by partitioning but fail during characterization.

mx_auto_char_params

`<0 | 1 >`

Informs MX to pass commands and parameters specified in the main script directly to characterization phase. Default: `0`

`0` Don't pass through any commands or parameters. (Default)

`1` Pass-through commands and parameters.

Commands passed through are:

- `define_template`
- `define_leafcell`

All parameters are passed through except:

- parameters specific to Liberate MX (parameters that begin with "mx_")
- parameters needed to be set to a specific value for the flow to work, such as `extsim_deck_include (0)`, `extsim_use_node_name (0)`

mx_auto_minp_eq_mpw_h_plus_l

`<0 | 1 >`

Specifies whether the sum of MPW high and MPW low is considered as a component of the minimum period.

Default: `1`

`0` The sum of MPW high and MPW low is not considered as a component of the minimum period.

`1` The resulting minimum period is enforced to be greater than or equal to the sum of MPW high and MPW low. (Default)

mx_automeas_filter

<value>	Limits the possible combination of measurement points to only defined output pins and corresponding clock or bitline or any switching mode during a functional operation.

In memories measuring all internal activity for all bits is tedious and you can select bits for which activity is to be monitored and measured. This parameter helps to choose a logical area to measure. Depending on the value supplied, Liberate MX chooses a set of clock, bitline, wordline and output for a specific read and write operation.

Example:

```
set_var mx_automeas_filter "Q\[0\]"
```

Liberate MX selects only clock and bit-lines (and prechargers attached to them) that can be reached back from `Q[0]`.

mx_autoprobing_hold_level

<number>	Specifies where hold constraint probing will be inserted based on the intersection between a pin and related pin. Default: `0 (i.e. first latch/combinational; i.e. master)`

Allows for probing to be intersection-level based: the choice of a specific logic region as a valid candidate for probing is based on the number of times pin and related pin intersect on any path propagating from the inputs to that logic. In this case, "logic region" means a channel connected region (i.e. NAND gate) or a strongly coupled component (i.e. latch, domino-stage, flip flop, memory array). See schematic below for an explanation of "Level 0" and" Level 1".

Example:

```
set_var mx_autoprobing_hold_level 0;
```

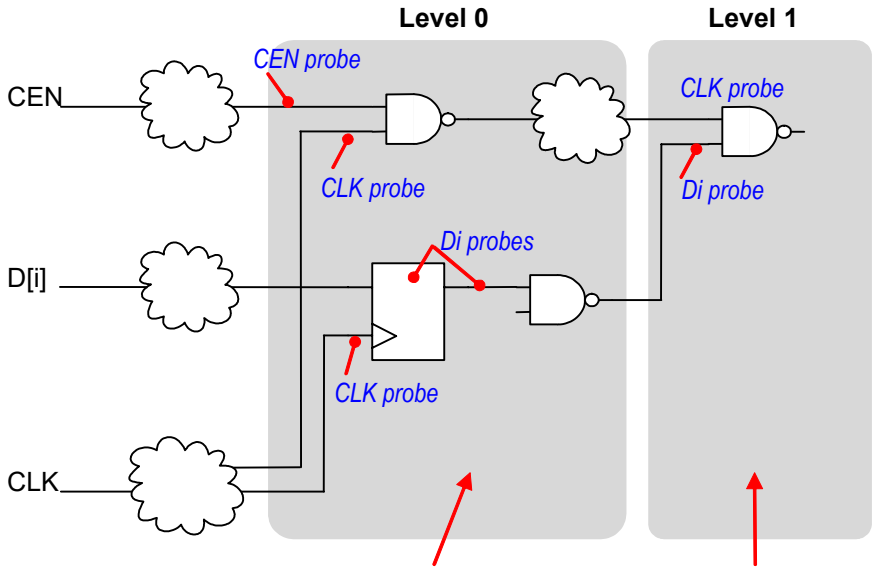
mx_autoprobing_setup_level

<number>	Specifies where setup constraint probing will be inserted based on the intersection between a pin and related pin. Default: `1 (i.e. second latch/combinational; i.e. slave)`

Similar to `mx_autoprobing_hold_level`, except applies to setup constraint (see above).

Note: for flip-flop based designs, this default will be `0`.

```
set_var mx_autoprobing_setup_level 1;
```



Hold constraint probes placed here for mx_autoprobing_**hold**_level = **0**

Setup constraint probes placed here for mx_autoprobing_**setup**_level = **1**

mx_bisection

`<0 | 1 >`

Specifies whether Liberate MX should use bisection to do characterization. Default: `0` (Do not perform bisection, use default path-delay method)

`0`	Liberate MX will use the path-delay method to perform characterization. (Default)
`1`	Liberate MX will use bisection to perform characterization.

During the partitioning and automatic probing phase, worst case constraints arcs are identified using path-delay differences method. As for any other arc in the MX flow, the resulting worst case arcs are partitioned and characterized in full SPICE. The full SPICE characterization phase uses either bisection or path-delay depending on the value of the `mx_bisection` parameter.

If bisection is specified, the probes being measured may differ from the ones used when the path-delay method is used – usually probes are pushed at the outputs of slave stages (as opposed to inputs of slave stages when path-delay is used). The user has the ability to control automatic probing when in bisection mode by using the `-bis_probe` option in the corresponding `define_arc` command.

Example:

```
define_arc \
    -bis_probe myNode \
    -bis_probe_dir R \
    -pin data -pin_dir F \
    -related_pin clk -related_pin_dir R \
    ...
```

Note: The 3.2 version of Liberate MX can only accept single switching on a pin in tables to generate correct bisection. If the table has multiple switching vectors as inputs, then Liberate MX characterization fails to perform bisection on that arc.

mx_bitline_probe_threshold

`<value>`	Measures the delay for bitline precharge at the percentage (%) point on the bitline waveform. This parameter is used only when `mx_min_period_mode` is set to bitline_precharge. Default: `0.98 (98%)`

mx_char_bundle_size

`<number>`	Causes the characterization to group partitions and execute as separate runs. Default: `0` (Don't group partitions into separate runs.)

This parameter instructs Liberate MX to split the characterization portion into separate runs in order to avoid excessive memory usage during `read_spice` (often caused by huge RC trees present in partitions.) Accordingly, this feature is useful for heavily extracted netlists, particularly those with extracted power rails or extracted virtual rails.

Regardless of this parameter, 3 distinct characterization scripts are always generated:

- All **timing** (`delay`, `constraint`, `measure`) partitions are grouped and run at once (through `script timing.tcl`)

- All **pincap/noise** partitions are grouped and run at once (through script `pincap.tcl`)
- **Power** characterization is run separately.

If `mx_char_bundle_size` is set to a number greater than zero (N > 0), then N partitions (or less) will be grouped as a set, and Liberate MX will be called sequentially on each set.

Example:

`mx_char_bundle_size = 20`, and a timing characterization contains 201 timing partitions, 10 pincap, and 1 power, the following scripts will be automatically generated:

```
<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
```

Notes:

- This bundling is independent from any LSF/queuing specified for the characterization run (queuing will happen independently from this bundling.)
- This is not related to the bundle or parallel bundle flow (which offers no advantage for runtime/memory in Liberate MX.)

mx_char_virtual_as_rail

`<list>`	Controls the behavior of virtual rails in dynamic partition and characterization. Default: `all`

This parameter controls the behavior of virtual rails in dynamic partition and characterization. This parameter works together with the `set_vdd` command of Liberate MX.

Example:

```
set_vdd -virtual VDD1 0.9
set_var mx_char_virtual_as_rail VDD1
```

mx_check_arcs

< 0 | 1 >	Check arcs found during partitioning against user-defined arcs and report them to file. Default: 0

Set this to 1 to report arcs specified by the user that will not be present in the final library. This can occur because arcs are not found by partitioning (could be an issue in user-specified vector, automatic probing, or dynamic partitioning), or because of a failure during characterization (issue in deck, or simulation option.)

The parameter mx_arc_report specifies the file where failed arcs are reported. (Default: mx_dir/arc.rpt). See also mx_check_arcs_exit_on_missing.

Example:

```
# Report user-specified arcs that will not be in final library
set_var mx_check_arcs 1
# File where arcs are reported
set_var mx_arc_report [pwd]/arc.info
# Exit if one or more user-specified arcs will not generate a partition
set_var mx_check_arcs_exit_on_missing 1
```

mx_check_arcs_exit_on_missing

< 0 | 1 >	Terminates execution if there is a mismatch between arcs specified by the user and arcs found by partitioning. Default: 0 (do not terminate execution)

Set this to 1 to force MX to terminate execution if there's a mismatch between user-specified arcs and arcs found by partitioning.

mx_clock2clock_constraints

`< 0 | 1 >` Allows for CLK to CLK constraint characterization. Default: `0`

mx_clk2clk_mode

`< bitline_precharge | none>` Controls the mode to be used for clock to clock arcs. Default: `bitline_precharge`

`bitline_precharge`	Checks that write and read operation is completed before the next precharge cycle starts.
`none`	No automatic measurement is generated.

mx_clock_tree_report

`<0 | 1>` Requests output of a clock tree report. Default: `0`

`0`	Does not generate a clock tree report. (Default)
`1`	Generates a clock tree report, `clock_tree.rpt`, in the Liberate MX directory. This report lists the active nets in the clock tree path for each table.

Example:

```
TABLE: /xxx/xxx/xxx/constraint.tbl
clk
->clkb
-->clk_buf_
---->clk_gate
```

This parameter must be set before `char_macro`.

mx_clone_if_uda

<0 | 1>	Specifies whether to clone a worst-case arc. Default: 0
0	Allows a worst-case arc to be cloned even if a user-defined arc for the cloned one is not available.
1	Allows a worst-case arc to be cloned, only if a user-defined arc for the cloned one is available. (Default)

Example:

```
# cloning worst-case arc only if a user-defined arc is available
set_var mx_clone_if_uda 1
```

mx_clone_if_uda_singlebit

`< 0 | 1 >` Controls how an arc defined between single bits of buses is characterized.

When set to `0`, arc `rel[i]->pin[j]` is treated as `rel[i:i]->pin[j:j]`, so the arc is characterized independent of other allowed arc combinations involving bits of pin and related buses. When set to `1`, `arc rel[i]->pin[j]` is worst-cased among allowed arc combinations involving bits of pin and related buses. Allowed arc combinations considered during cloning (worst-casing) are:

- user defined arcs (`mx_clone_if_uda == 1`), or
- any possible arc (`mx_clone_if_uda == 0`) between buses rel and pin.

Default: `1`

mx_const_prop

`< 0 | 1 >` Performs/Skips constant propagation at the top level. Default: `0`

mx_constraint_ocv_factor

`<value>` Factor by which to correct constraint measurements. Default: `0`

For HOLD, maximum measured clock path delay is increased (multiplied) and minimum measured data path delay is decreased (divided) by this factor. For SETUP, maximum measured data path delay is increased (multiplied) and minimum measured clock path delay is decreased (divided) by this factor. The default is `0`, i.e. NO OCV correction.

Example:

```
# set 3% variation
set_var mx_constraint_ocv_factor 0.03
```

mx_corecell

`<identifier>`	Where identifier is one of the following: `single_port`, `dual_port`, `rom`. Default: `single_port`

This parameter offers a way of specifying the type of core cell in a more concise way than with the `-mxcore` option to `define_cell`. It accepts as value the most common configuration of SRAM cells – single or dual port, i.e. 6T or 8T – or just the keyword `rom`. When `rom` is specified, the tool doesn't look for an SRAM structure at all.

Example:

```
# look for a dual_port sram 8T core cell …
set_var mx_corecell "dual_port"
# Command has the same effect of mxcore example command
```

mx_cp_cmd_preserve_date

`< 0 | 1 >` Controls how the `cp` command will function.
Default: `1`

- `0` Uses the `cp` command to copy.
- `1` Uses the `cp -p` command to copy with preserving permission.

mx_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 `mx_create_if_uda` parameter 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 `mx_create_if_uda` is set to 1, the potential arc is not created and not characterized.

- If `mx_create_if_uda` is set to 0, the arc is created and, if partitionable, characterized.

mx_debug

`< clock | measure | memory | pattern | power | sim | arc>`

`clock`	Generates information on clock propagation.
`measure`	Generates information on measure. It also creates `debug_measure.csv` file with all measurement details.
`memory`	Generates information on the system memory footprint (memory consumed during the run).
`pattern`	Generates information on mxtable expansion results.
`power`	Generates information about power characterization.
`sim`	Generates information on FastSPICE simulation decks and commands being generated during the run.
`arc`	Generates information on arc probing. It also creates `debug_arc.csv` file with all delay and constraint details.

Example:

```
set_var mx_debug arc :
```

It generates files `debug_arc.csv` and `debug_arc.log`, which contains all valid candidates considered to get worst case for a given arc.

- `debug_arc.csv`: This file contains a comma separated list of all arcs. It can be directly opened in Excel.
- `debug_arc.log`: This file contains various debug information.

Each line gives detail about a valid path pair for a given arc (setup/hold).

For any given arc, Liberate MX starts with the default value `-1.0`, and the moment a valid value is found, the database gets updated with the new value. You can see the updated value in the column `New` of the xls file (`debug_arc.csv`).

Delay/Retain: For delay and retain arcs, values mapped to `t1` are for delay measurement and to `t2` are for transitions.

```
DELAY = t1_max
TRANS = t2_max
RETAIN = t1_min
RETAIN_SLEW = t2_min
```

Constraints: Liberate MX performs a total of four measurements in order to find setup/hold and these are mapped to `t1_min/t1_max/t2_min/t2_max`. MX takes measurements on both lower and upper thresholds, for example, 30-70 or 20-80. Smaller delays are mapped to min and larger ones are mapped to max. Hence final setup/hold is calculated as given below.

```
SETUP = t1_max-t2_min
HOLD = t2_max-t1_min
```

You can specify any combination of the values listed.

This parameter must be set before the `char_memory` command.

mx_defmem_sdf_cond

`<0 | 1 >` Requests the reuse of the `sdf_cond` from the original library. Default: `0`

- `0` Do not reuse existing `sdf_cond` value. The `sdf_cond` is determined from the `when` arc.
- `1` Reuse the existing `sdf_cond` value.

mx_delay_ocv_factor

`<value>` Factor by which to correct delay measurements. Default: `0`

Example:

```
# set 3% variation
set_var mx_delay_ocv_factor 0.03
```

mx_dir

`<string>` Specifies the complete directory path where MX temporary files should be stored. Default: `[pwd]/mx`

Example:

```
# write mx files in to specified dir
set_var mx_dir /home/user/test_macro/mx
```

Parameters for Job Distribution

- mx_distributed_retry_enable
- mx_distributed_retry_count
- mx_distributed_simulation_timeout
- mx_distributed_pending_timeout
- mx_distributed_status_cmd
- mx_distributed_kill_cmd

mx_distributed_kill_cmd

`<string>` Specifies the name of the command that will kill jobs. Default: `bkill`

Note: This parameter must be specified before char_macro.

mx_distributed_pending_timeout

`<value>` Specifies the time (in seconds) in which the pending jobs timeout. All the pending jobs are killed if they are not submitted to a host before timeout. Default: `-1` (Never kill jobs)

Note: This parameter must be specified before char_macro.

mx_distributed_retry_count

`< 0 | 1 | 2 >`

Specifies the number of tries to restart a distributed job. If the characterization is not successful and the maximum number of retries is reached, then the tool marks the cell as failed and exits.

- `0` (Default): Never resubmit a job.
- `1`: Resubmit a job once.
- `2`: Resubmit a job twice.

Note: This parameter must be specified before char_macro.

mx_distributed_retry_enable

`< 0 | 1 >`

Specifies whether to enable the resubmission of distributed jobs. Default: `0`

Note: This parameter must be specified before char_macro.

mx_distributed_simulation_timeout

`<integer>`

Specifies the time (in seconds) to timeout simulation jobs. If a simulation exceeds the specified timeout value, the job will be killed. Default: `86400`

Note: This parameter must be specified before char_macro.

mx_distributed_status_cmd

`<string>`

Specifies the commands that will return the job status. Default: `bjobs`

Note: This parameter must be specified before char_macro.

mx_distributed_sim

`<"options">`	Passes switches and options to an external program. Default `""` (`none`).

This parameter passes switches and options to external programs such as a simulator or job management system. This allows the user to run a program with the same options they use standalone.

Example:

```
# Pass info to job queue
set_var mx_distributed_sim "bsub -q <queue_name> -o <log> -I spectre"
```

mx_domain_propagation

`<"static" | "dynamic" | "static dynamic">`

Controls the behavior of domain propagation. Default: `"static dynamic"`

`static` Use static information to propagate the domain through the circuit, using these heuristics:

Combinational logic (mix of clock and data domain among its inputs)

- clock will propagate to the output
- data will stop

Sequential logic (mix of clock and data domain among its inputs)

- clock will stop
- data will propagate through

`dynamic` Use dynamic information coming from FastSPICE simulation to propagate the domain through the circuit; i.e. a domain on the input of a gate will propagate to the output if they both switch in at least one common simulation interval.

`static dynamic` Use 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. (Default.)

A domain force or stop on a domain node will overwrite the results of either method.

mx_dpartition_inactive_tie

`< all | inter | ? >` Specifies how to transfer the steady state region of the circuit from the top-level fastsim run to the partition-level fullspice run for a specific vector. Default: `inter`

If set to `all`, all inactive notes are connected.

mx_dynamic_include_full_core

`<0 | 1 | 2>` Instructs the tool to include the full core cell in dynamic partitioning independently from activity. Default: `2`

- `0` Includes only the active portion of the memory core in the partition.
- `1` Forces the entire memory core to be included in the partition even if only half the memory core activity is captured.
- `2` Forces the entire memory core to be included in the partition for the monitored memory core.

This command should be used in conjunction with `mx_monitor_memcore` set to `1`. In that case, activity would be available for memory cores, just like any other node, so only the active portion of the memory core would be included in the partition when traversed in. For example, a memory access traversal.

Using this command alone forces the whole core to be included, which is normally needed in the measurement flow to improve accuracy for measurements that involve core nodes as probes.

mx_extsim_process_netlist_cmd

`<string>`

Specifies a Linux system command that is run on each partition netlist when the partitions are being run. This parameter supports the following three symbols:

- `%L`: Log file name.
- `%I`: Input netlist name.
- `%O`: Output netlist name.

`%I` and `%O` must be specified, otherwise the parameter is ignored.

Example:

```
set_var mx_extsim_process_netlist_cmd "qreduce -cpu 2 -log %L -out %O %I"
```

After the command is run for a partition, it will generate the following files:

- The command log file, if specified, will be below the `sim.sp_pre_process.log`
- The original partition netlist will be preserved below the `sim.sp_pre_process` file.
- The processed netlist will be below the `sim.sp` file.

If the specified command fails, the unprocessed netlist file will be used for partition simulation.

mx_fastsim_auto_ic

`< 0 | 1 >`

Forces on/off the settings of initial condition on memory cores in the FastSPICE decks. Default: `1`

mx_fastsim_clock_slew

`<double>`

Specifies what clock slew to use in FastSPICE simulation. When not specified, slew is chosen as the first entry in the clock slew template. The value defined here is in nano seconds (ns). Default: `-1`

mx_fastsim_deck_pwl_precision

`<integer>` Sets the precision for PWL written into fastsim. The specified value decides the number of digits that will be placed after the floating point.
Default: `9`

mx_fastsim_init_virtual_rails

`<double>` Initializes Liberate MX fastsim virtual rail ratio.
Default: `-1`

mx_fastsim_input_slew

`<double>` Specifies what input slew to use in FastSPICE simulation. When not specified, slew is chosen as the first entry in the input slew template. The value defined here is in nano seconds (ns).
Default: `-1`

mx_fastsim_load

`<double>` Specifies what load to use in FastSPICE simulation. When not specified, load is chosen as the first entry of the output load template. The value defined here is pf.
Default: `-1`

Here is an example on how you can override the fastsim slew/load.

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

define_template -type constraint \
         -index_1 {0.16 0.5 1.6 } \
         -index_2 {0.16 0.5 1.6 } \
         constraint_template
```

```
In Top level TCL :
set_var mx_fastsim_input_slew 0.19
set_var mx_fastsim_clock_slew 0.25
set_var mx_fastsim_load 0.10

When used this way following values will be used
Input slew : 0.19ns
Clock slew : 0.25ns
Output load : 0.10pf
```

mx_fastsim_parallel_wait

`<integer>`	Specifies the time interval for each submitted simulation for `mx_distributed_sim`. Set this parameter to stagger the runs and not overwhelm the system. Default: `0`

mx_fastsim_reuse

`< 0 | 1 >`	Uses FastSPICE simulation results from a previous run. Default: `1`

Results of FastSPICE runs are stored by MX in directory `./mx_fastsim` under file name `<macro_name>_<table_file_name>.alwf`. If the sensitization (i.e. the mxtables) does not change from one run to the next, it is advisable to reuse previous run results by setting this flag to `1`.

Liberate MX will automatically re-invoke the FastSPICE simulator for the expected result files that cannot be found. See also table-based option `fastsim_reuse`.

Example:

```
# No table has changed from previous run - reuse
# FastSPICE results for current run
set_var mx_fastsim_reuse 1
```

mx_find_arrays

`< 0 | 1 >`	Identifies large channel-connected components as possible candidates for memory arrays. Default: `0`

It is recommended that this be turned off for small memories, if memory cores are not properly identified.

mx_find_memcore_numbit_threshold

`<integer>`	Sets a lower bound for the number of memory cores expected to be found in a channel connected or strongly-coupled region. Default: `1`

The use of this parameter helps avoid identifying a regular latch as a memory array when the latch's structure is identical to that of a memory core cell, which could affect the automatic probing and, ultimately, the constraint characterization.

mx_find_memcores

`< 0 | 1 >`	Forces Liberate-MX to stop searching for memory cores recognition when set to zero. Default: `1`

This parameter allows forcing Liberate-MX to stop the searching of memory cores recognition when set to `0`.

Example:

```
#to stop memory core search.
set_var mx_find_memcores 0
```

mx_find_stack_loads

`< 0 | 1 >`	Instructs the tool to identify active loads typically used on tracking lines. Default: `0`

Because they usually come in large numbers, active loads can introduce a large number of errors if modeled as passive caps in partitioning. This command identifies them and forces partitioning to keep them as active loads rather than equivalent passive caps.

mx_find_virtual_rails

`< 0 | 1 | 2 >`	Identifies nodes that should be treated as logic 1, 0 during static analysis. Default: `0`. Recommended: `2`

When set to `1`, the command triggers reporting of power switched nodes that should be considered as logic constant during static analysis. Reported nodes should be then set as vdds, gnds by using the `set_vdd` or `set_gnd` commands with the `-virtual` option for subsequent runs. Use this flag when static partitioning step takes an unusually long time, that is, more than 10 minutes on a 10M xtrs block.

Example:

```
# Tool seems to be hanging in static partitioning:
# (MX-info) - Partitioning - static - start ...
#    (MX-info) - Partitioning  24/1693326 xtrs
#

# Stop execution and add to your tcl script:
set_var mx_find_virtual_rails 1
#

# Rerun; following will be reported:
| (MX-info) - Finding virtual rails ...   wall clock time +=    0 sec
  |  | (MX-info) - xtop/net112:1 - was no 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.
```

Liberate MX also reports all possible nodes once.

When set to `2`, the nodes that meet certain criteria are automatically set as virtual rails. Setting `mx_find_virtual_rails` to `2` generates a report file containing all suspected virtual rails. This report is located at `mx/virtual.rpt`.

Note: Review `virtual.rpt` to ensure all virtual rails are defined as per design.

Sample `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)
```

This report contains a summary of each possible virtual rail, its PMOS and NMOS drain connections.

The following requirements should be met for automatic setting of a node as virtual VDD.

- PMOS drain connections are greater than `mx_virtual_rail_minimum_xtrs`. Default is `50`.
- There are zero NMOS drain connections.

The following requirements should be met for automatic setting of a node as virtual GND.

- NMOS drain connections are greater than `mx_virtual_rail_minimum_xtrs`. Default is `50`.
- There are zero PMOS drain connections.

mx_fix_interpolate_tran_mode

`<0 | 1 | 2 | 3>` Fixes the negative transition values. Default: `2`

- `0` The negative values are not fixed.
- `1` Fixes only the negative values.
- `2` Fixes all the values that are less than the value specified for the `mx_min_tran_value` parameter.
- `3` Fixes the negative values by copying the same simulated value of the load for all slews.

mx_fix_pin_vdd

`< 0 | 1>` Turns on a fix to correct an issue where the automatically generate `pin_vdd` value may be wrong for an IO. Default: `0`

When Liberate MX used in reuse mode and in a different corner than what FastSPICE simulation was run at, the `pin_vdd` value automatically generated for IO pins may be wrong and use the previous PVT value rather than the current one. Set to `1` to implement this fix. (Recommended).

mx_force_arc_ignore_autoprobing_level

`< 0 | 1>` Ignores autoprobing. Default: `0`

mx_full_rail_tol

`< tolerance >` Controls the tolerance that defines what is considered full-rail, as a percentage of VDD. Default: `0.05 (5%)`

If Vmax and Vmin are maximum and minimum voltage levels reached by an output in a transition, the transition is considered to be full-rail if `mx_output_require_fullrail_switch` is set true, and `(Vmax - Vmin) < mx_full_rail_tol * VDD`.

Example:

```
# Outputs can transition to 91% of VDD and still be considered switching
```

```
set_var mx_output_require_fullrail_switch 1
set_var mx_full_rail_tol 0.1
```

mx_fsdb_output_name

`<string>`	Specifies the waveform filename containing the output data in FSDB format. This file is written out by the simulator and is read in for partition. Default: `transient1.tran.fsdb`

This parameter must be set before the char_macro command.

mx_gen_nominal_lvf_mode

`<0 | 1 | 2 | 3>` Controls the type of netlist generation. Default: `0`

`0`	Both nominal and LVF are run on nominal netlist. The `char_macro -lvf` option is used to run `timing.tcl` and `timing_lvf.tcl`.
`1`	Ignores the `char_macro -lvf` option and generates only nominal netlist. Runs `timing.tcl` and skips `timing_lvf.tcl`.
`2`	Generates both nominal netlist and new reduced LVF netlist based on the `set_var -stage` variation option. Runs `timing_lvf.tcl` and skips `timing.tcl`.
`3`	Generates both nominal netlist and new reduced LVF netlist based on the `set_var -stage` variation option. Runs both `timing.tcl` and `timing_lvf.tcl`.

mx_greybox

`< 0 | 1 >`	Generates a library directly out of FastSPICE. Default: `0`

Set this to generate a library directly out of FastSPICE without partitioning, or a full SPICE run.

Note: This must be a standalone and separate run. This means that either a library will be generated directly from FastSPICE (`mx_greybox=1`) or the software will proceed with the normal flow. Default: `0`.

mx_greybox_constraint_method

`< 0 | 1 >`	Controls the number of runs to be done on a constraint table. Default: `1`

This parameter controls the number of runs to be done on a constraint table and sets it to max(i1,i2) where i1 and i2 are number of constrained and related slew indexes.

mx_info_file_print_probe_threshold

`<0 | 1>`	Prints probe threshold. Default: `0`

mx_inputcap_ldb_reuse

`<0 | 1>`	Reuse pin cap LDB from previous run. Default: `0`

See mx_ldbs_reuse for description of all LDB reuse parameters.

mx_intersect_domains_check_intra_ccc_dynamic_propagation

<0 | 1>	Adds false path detection step to avoid creation of large number of false arcs when there are non causal vector provided and redundancy type logic is present in the circuit. This will also generate warning messages during dynamic partitioning and huge runtime caused by domain propagation and auto probing. Default: 0

mx_ldbs_reuse

<0 | 1>	Global parameter to reuse all the LDBs from the previous run. Default: 0

It will also re-characterize the arcs that failed characterization in previous run, if any.

Liberate MX can reuse all or selective LDBs from the previous run by using the following local parameters:

mx_inputcap_ldb_reuse
mx_power_ldb_reuse
mx_noise_ldb_reuse
mx_timing_ldb_reuse

Note: Do not set the global and the local parameters at the same time. You can either set to globally reuse the LDBs or set the local parameters for specific reuse of certain LDB types.

mx_leakage_check_time

`<0 | 1 >`	Controls whether the leakage current should be read. Default: `1`
`0`	Does not read the leakage current.
`1`	Reads the leakage current.

mx_leakage_measure_final

`<0 | 1 >`	Controls the measurement of the leakage value. Default: `1`
`0`	Measures the leakage value as the average value of the beginning and the end of the subinterval range.
`1`	Measures the leakage value at the end of the subinterval range.

mx_lvf_monte_entries_per_thread

<value>	Sets the number of entries each post-partition `.tcl` will characterize in the Monte Carlo or VVO flow. Default: `0`
	Example:
	The following will enable each run Tcl to characterize 2 entries.
	`set_var mx_lvf_monte_entries_per_thread 2`

This parameter must be set before the char_macro command.

mx_margin_report

`{filename}`	Generates a report by the `define_measure` commands. Default: `"measure.rpt"`

To generate this report, specify a filename. To omit this report, set this parameter to `""` (empty quotes.)

mx_mcf

`<double>` Miller Cap Factor; multiplies each coupling cap in the netlist by the specified amount. Default: `1`

mx_measure_error_file

`{filename}` Generates an error report file for the `define_measure` commands. This report lists all measurements that did not evaluate during the run and the reason of the failure. Default: `"measure.err"`

For example:

```
MEASURE NAME ,    ERROR TYPE

bad ,             -trig not found
```

mx_measure_distribution_mode

`<0 | 1 >` Specifies whether the post-partition `.tcl` files must be run in parallel. Default: `0`

The default value will not change the behavior of the tool from the previous releases.

- `0` Post-partition .tcl files are run in sequence.
- `1` Post-partition .tcl files are run in parallel.

mx_min_clock_tree_mode
mx_max_clock_tree_mode

`<clock_nodes | user_attributes | none>`

Controls the mode to use for clock tree path computation. Default: `clock_nodes`

- `clock_nodes` Automated measurements for minimum/maximum delay to any clock node is generated and used for `min_clock_tree_path`.

`user_attributes`	Specifies only clock nodes that are probes of setup and constraint arcs that have the following user-defined attributes: `altos_ctree_probe`, `altos_min_ctree_probe`, or `altos_max_ctree_probe`. These are appropriately used for `clock_tree_path` computation.
`none`	No automatic measurement is done.

mx_merge_arc_packet_ldbs

`<0 | 1 >`

When using the arc packet flow, specifies whether to merge individual LDBS before reading. Default: `0`

- `0` Does not merge individual LDBS before reading.
- `1` Merges all the individual LDBS before reading

mx_merge_multi_slew_load_mode

```
<closest_index_log | worst_case | best_case |
matched_index_worst_case | matched_index_all_max | none>
```

Enables a method to merge slew/load-based LDBs to create arc level LDB. This parameter can be used only in multi slew/load flow for delay arcs.
Default: `closest_index`

- `closest_index` Uses the values from the partitions of matched indexes, otherwise uses the closest index value.
- `closest_index_log` Same as `closest_index`, but uses a log scale for closest index.
- `worst_case` Uses worst case from all the partition values for each index1, index2 combination.
- `best_case` Uses best case from all the partition values for each index1, index2 combination.
- `matched_index_worst_case` Uses the values from the partitions of matched indexes, otherwise uses the worst-case value.
- `matched_index_all_max` Uses the values from the partitions of matched indexes, otherwise uses the max value.

`none`	Does not merge multi slew load data.

mx_min_period_latch_component_mode

`<0 | 1 >` Specifies an enhanced min period latch component calculation. Default: `0 (off)`

`0`	For designs where the latch clock is controlled only by internal clock. (Default)
`1`	For designs where the latch clock is controlled by both an internal and external clock.

Set this parameter to specify an enhanced minimum period latch component calculation for designs where the latch clock is gated by both an external and internal clock.

mx_min_period_mode

`<internal_pulse | latch | bitline_precharge | all | none>`

Controls various minimum period components. Default: `internal_pulse`

`internal_pulse`	Internal pulse width in design is considered.
`latch`	Use latch component.
`bitline_precharge`	Use bitline precharge component.
`all`	Use all automatic components.
`none`	No automatic measurement is done.

Liberate MX supports all measurements contributing to minimum period for self-timed memories. This parameter is used to control the components that should be considered for minimum period calculation.

Note: When using automatic flows (using the `define_memory` command), this parameter is set to `latch`, `user_spec`, `bitline_precharge`, or `internal_pulse`.

You can use any combination of valid values for minimum period calculation.

Example:

```
set_var mx_min_period_mode "latch bitline_precharge"
```

mx_min_period_pulse_trig_rise_thresh

`<value>` Sets the threshold for measuring minimum pulse probe from rising edge of rise pulse. Default: `0.05`

mx_min_period_pulse_targ_rise_thresh

`<value>` Sets the threshold for measuring minimum pulse probe to falling edge of rise pulse. Default: `0.05`

mx_min_period_pulse_trig_fall_thresh

`<value>` Sets the threshold for measuring minimum pulse probe from falling edge of fall pulse. Default: `0.95`

mx_min_period_pulse_targ_fall_thresh

`<value>` Sets the threshold for measuring minimum pulse probe to rising edge of fall pulse. Default: `0.95`

mx_min_risefall_retain

`<0 | 1 >` Allows copying the rise and fall ECSM data for the retain arcs. Default: `0`

- `0` Does not copy rise and fall ECSM data to the other edge for retain arcs.
- `1` Copies rise and fall ECSM data to the other edge for retain arcs.

mx_minp_single_slew

`<slew value>` Forces minimum period characterization to use the single clock slew. This parameter is used independently from any other template that is normally used for minimum period characterization.
Default: `-1`

Minimum period characterization uses slews in the following order of precedence:

- Single number as specified in `mx_minp_single_slew` parameter
- Slews as specified in MPW type template defined for the run
- Clock slews as specified in constraint type template defined for the run

Example:

```
set_var mx_minp_single_slew 0.09

define_template -type mpw \
    -index_1 {0.04 0.2 } \
    mpw_template_1

define_cell \
       -clock { clk } \
       -input { adr[0:4] bw[0:3] cs din[0:3] } \
       -output { dout[0:3] } \
       -delay delay_template_3x2 \
       -constraint constraint_template_3x2 \
       -power power_template_3x2 \
       -mpw mpw_template_1 \
       $cell

produces the following groups in the library:

lu_table_template (minp_single_slew) {
  variable_1 : constrained_pin_transition;
  index_1 ("0.09");
}
lu_table_template (mpw_template_1) {
  variable_1 : constrained_pin_transition;
  index_1 ("0.04, 0.2");
}

....
 timing () {
   related_pin : "clk";
   timing_type : min_pulse_width;
   rise_constraint (mpw_template_1) {
     index_1 ("0.04, 0.2");
```

```
    values ( \
      "0.0699806, 0.144858" \
    );
  }
  fall_constraint (mpw_template_1) {
    index_1 ("0.04, 0.2");
    values ( \
      "0.0803253, 0.175298" \
    );
  }
}
timing () {
  related_pin : "clk";
  timing_type : minimum_period;
  rise_constraint (minp_single_slew)
    index_1 (0.09)
    values ( \
      "2.59703" \
    );
  }
  fall_constraint (minp_single_slew)
    index_1 (0.09)
    values ( \
      "2.59703" \
    );
  }
```

mx_min_tran_value

`<value>` Specifies the minimum transition value which will be used during the interpolation to fix all the transition values that are smaller than this parameter in the table. Default: `1e-12`

mx_monitor_memcore

`<value>` Special debug flag. Default: `"inter"`

`none`	No memory core node is monitored; can lead to slightly larger partition sizes in register files, but to significant reduction in FastSPICE run time and MX footprint for larger testcases.
`inter`	Only memory core nodes that cross CCC are monitored. (Default.)
`intra`	Only memory core nodes that do not cross CCC are monitored. Should be used for debugging purposes only.
`all`	All memory core nodes are monitored. Should be used for debugging purposes only and on small cases. (Recommended)

mx_monitor_memcore_lprobe_level

`<value>` Uses `.lprobe` statement instead of `.probe` statement on memory core nodes. A `.lprobe` statement forces the simulator to generate a digital waveform (that is, only 0, 1, X values) instead of an analog waveform for the node on which it is issued. This results in savings on waveform size and reduces runtime memory requirement during fastsim waveform read back. By default, the `.lprobe` statement is not used.

To enable this in Liberate MX, the `mx_monitor_memcore_lprobe_level` parameter is used and the value passed is half of core VDD. Therefore, Liberate MX knows how to automatically set low and high in the `.lprobe` statement. It should also be known how to reconstruct an analog waveform from a digital waveform.

Example

```
set_var mx_monitor_memcore_lprobe_level [expr 1.215 / 2]
```

Here, 1.215 is supply voltage (VDD) for characterization run.

mx_mpw_allow_same_probe_on_both_rise_and_fall_clock_tree

`<0 | 1 >` Allows a node to be associated with both rising and falling clock trees. Default: `1`

- `0` Requires a node to ONLY switch with clock rising(falling) to be considered on the rise(fall) clock tree.
- `1` A node that switches with both clk:R and clk:F can belong to both trees. (Default)

mx_mpw_false_probe_delay_threshold

`<value>` Filters out nodes that are internal return signals. Default: `1e-9`

Filters out nodes that are just internal return signals and do not contribute to the characterization of MPW. If the delay between primary input and probe is greater than `mx_mpw_false_probe_delay_threshold`, the probe is discarded and not used during MPW calculation.

mx_mpw_measurement _duration

`<value>` Sets duration of the MPW measurement as a fraction of `mx_simulation_interval`. Default: `0.75` (assumes clock period of 1).

Sets the duration of the MPW measurement as a fraction of `mx_simulation_interval`. Usually the period is two-times the simulation interval (`clk period == 2 * mx_simulation_interval`) so a default of `0.75` ensures the duration of the measurement covers a rising and a falling edge.

mx_mpw_mode

`<edge_intersection | none>`

Selects method for calculating MPW.
Default: `edge_intersection`

`edge_intersection`

(Default) Selects method described below:

- Clock trees are traced to identify the rising-edge and falling-edge clock trees. Tracing is done using both static and dynamic methods.
- Identification of probe nodes as nodes that lie on the intersection between rising and falling clock tree.
- Definition of MPW HIGH as setup between CLK:R nodes and CLK:F nodes, and MPW LOW as setup between CLK:F nodes and CLK:R nodes.

`none` No automatic measurement is done.

mx_mpw_probe

`<seq | comb | array | seq_states>`

Specifies the types of logic to be probed for MPW.
Default: `seq`

`seq` Probes the sequential gate inputs. (Default)

`comb` Probes the combinational gate inputs.

`array` Probes the memory array.

`seq_states`

Probes the sequential gate inputs, sequential gate outputs, and the internal state nodes.

When looking for MPW probes, specifies what type of logic should be probed. For minimum and maximum clock tree probe selection, Liberate MX considers first level sequential logic, that is, hold latches.

mx_mpw_probe_lower_fall
mx_mpw_probe_lower_rise
mx_mpw_probe_upper_fall
mx_mpw_probe_upper_rise

`<value>`	Specifies the thresholds for MPW probe measurements.

Defaults are as follows:

```
mx_mpw_probe_lower_fall: 0.3
mx_mpw_probe_lower_rise: 0.3
mx_mpw_probe_upper_fall: 0.7
mx_mpw_probe_upper_rise: 0.7
```

mx_mxtable_interpret_read_write_cycle_keywords

`< 0 | 1 >`	Control interpretation of read and write cycle keywords. Default: `0`

Use this parameter to turn on and off (default OFF) the capability to interpret 'read' and 'write' cycle keywords in a table. These keywords were originally intended to allow for a more concise table functional description of a memory, and they should be used only for simple cases. In general, it is recommended not to use these keywords, particularly for the more complex designs and instead give a fully expanded description of the write and read operations.

mx_negedge_clock

`<list>`	Specifies the active edge(s) of a clock, otherwise Liberate MX assumes positive active edge(s). Default: `none`

This parameter allows specifying negative active edge(s) of a clock otherwise, Liberate-MX assume positive active edge(s).

Example:

```
# to specify clk1 & clk2 being negative active edge.
set_var mx_negedge_clock clk1 clk2
```

mx_noise_ldb_reuse

<0 | 1>	Reuse noise LDB from previous run. Default: 0

See mx_ldbs_reuse for description of all LDB reuse parameters.

mx_output_require_fullrail_switch

< 0 | 1 >	Requires an output transition from FastSPICE to be a fullrail in order to be considered as valid for partitioning. Default: 1

mx_partition_name_use_arc

< 0 | 1 >	Controls the naming of MX partitions. Default: 1

If this parameter is set to 1, Liberate MX uses information on the arc the partition represents.

Examples:

Constraint partition:

```
single_port_sram_ext_constraint_adr0_hold_f_553/
```

Delay partition:

```
single_port_sram_ext_delay_dout0_r_clk_r_558/
```

Without setting this parameter, "adr0_hold_f" and "dout0_r_clk_r" would be missing from above partitions respectively.

mx_pathdelay_hold_clock_margin

`<double>` Adds margin to hold on clock path. The defined value is directly multiplied with the measured clock path.
Default: `0`

mx_pathdelay_hold_data_margin

`<double>` Adds margin to hold on data path. The defined value is directly multiplied with the measured data path.
Default: `0`

mx_pathdelay_margin_mode

`<0 | 1 >` Specifies how to apply margin for the arcs. Default: `1`

- `0` Adds the same margin for all the arcs (pin/rel_pin/pin_dir/rel_pin_dir).
- `1` Considers probe/rel_probe/probe_dir/rel_probe_dir. The arcs that have pin/r_pin/probe/r_probe matching with the user-defined `define_arc -attribute altos_mx_pathdelay_clock|data_margin` will use the margin set in the attribute. Other probe/r_probe pairs will use the parameter setting defined by:

  mx_pathdelay_setup_data_margin

  mx_pathdelay_setup_clock_margin

  mx_pathdelay_hold_data_margin

  mx_pathdelay_hold_clock_margin

mx_pathdelay_save_data

`<0 | 1 >` Specifies whether the clock and data delays must be saved to the LDB file. Default: `0`

- `0` Does not save the clock and data delays. (Default)

1	Saves actual *data_path* and *clock_path* delays.

mx_pathdelay_setup_clock_margin

`<double>` Adds margin to setup on clock path. The defined value is directly multiplied with the measured clock path.
Default: `0`

mx_pathdelay_setup_data_margin

`<double>` Adds margin to setup on data path. The defined value is directly multiplied with the measured data path.
Default: `0`

mx_pincap_char

`< 0 | 1 >` Performs/skips pin capacitance characterization. Default: `1`

This can be done at pin/cell level as well, which makes it convenient when there are many pins for a cell.

Example:

The following would enable pin capacitance for only 3 pins:

```
set_var mx_pincap_char false
set_var -pin {X Y Z}  mx_pincap_char true
```

You can also do the following (similar to `-ignore` type of usage):

```
set_var mx_pincap_char true
set_var -pin {X Y Z}  mx_pincap_char false
```

Note: For correct functionality, always set the parameter globally to `true` or `false` before setting at the pin level.

mx_pincap_report_mode

`<0 | 1 | 2>` Specifies if the pin capacitance report should be generated.
Default: `0` Recommended: `2`

`0` Does not generate the pin capacitance report.

1	Generates the pin capacitance report in the mx_dir directory. This setting is recommended for static pin capacitance debugging.
2	Generates the pin capacitance report in the mx_dir directory. An input pin is dropped off and not characterized in the static flow if the partition size exceeds the maximum transistor or maximum side input limit. It is recommended to rerun the dropped pins using the dynamic flow.

mx_posedge_clock

<list>	Specifies rising edge(s) clocks. Default: all

This parameter allows specifying positive active edge clocks. This is useful when a deck has both positive and negative active edge(s).

Example:

```
# to specify clk1 & clk2 being negative active edge and
# clk3 and clk4 being positive edge.
Set_var mx_negedge_clock clk1 clk2
Set_var mx_posedge_clock clk3 clk4
```

mx_postprocess_probe_relprobe_intersection

`<0 | 1 >`

Both probe and related probe are CCC inputs and there is a path to power/ground controlled by both. If no state or output node crosses this path, this arc will be considered as an invalid pair and will be discarded.
Default:`0`

`0`	The input will be regarded as a probe.
`1`	The input will be discarded.

mx_power_assign

`all | clock | input | output <list of pins>`

Specifies how to distribute internal power among switching pins.
Default: `all`

`all`	Distributes internal power contribution equally among switching pins. Selecting **all** is equivalent to input and output.
`clock`	MX assigns all switching power to clock, independently from other inputs switching. (Default)
`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>`	Internal power contribution is distributed among listed pins when switching. If you use the default setting of `clock`, MX assigns all switching power to clock, independently from other inputs switching.

mx_power_divide_num_switching_mode

`<0 | 1 | 2 | 3>`

Controls how output power is distributed when multiple switching by inputs and outputs.
Default:`3`

0	Divides hidden by num swinputs, switching by num swoutputs.
1	Divides hidden by num swinputs, switching by num swoutputs and num swinputs.
2	Divides hidden by num sw inputs, switching by #swinputs and #swoutputs (for backward compatibility).
3	Automatically divides swoutputs when num of swoutputs = num of swinputs. Else, use mode 1.

This parameter can be set globally for all arcs using the following command

```
'set_var mx_power_divide_num_switching_mode 0'
```

It can also be set per arc as follows:

```
'set_var -cell {RAM1P10240x16} -type power -pin {AY*} -related_pin {TA*} -pin_dir
F -related_pin_dir F mx_power_divide_num_switching_mode 0'.
```

Note: The `'*'` wildcard applies the parameter value for bus pins. For example, `AY[12:0]`. Also, if one of the options is not specified, for example, `no '-pin_dir'`, then the parameter value applies for all arcs matching the specified criteria (`all pin_dir's`).

mx_power_ldb_reuse

`<0 | 1>`	Reuse power and leakage LDB from previous run. Default: `0`

See mx_ldbs_reuse for description of all LDB reuse parameters.

mx_power_window_ratio

`<value>`	Controls the measurement window time for dynamic power calculation as a ratio of `mx_simulation_interval`. The value range for this parameter is 0 to 0.5. Default: `0.125`

This parameter is used to force a window of observation to be smaller than default (`0.5*mx_simulatoin_interval`).

Note: `mx_power_window_ratio` works with an assumption that all `i(VDD)` activity happens within the observation window.

mx_power_single_point

`<0 | 1 >` Allows power characterization to be performed for multiple slews/loads. Default: `1`

- `0` Characterize multiple points for slew/load table.
- `1` Populate slew/load table with a single point. (Default)

Allows power characterization to be performed for multiple input slews and output loads, based on the power template that is provided.

This parameter must be set before `char_macro`.

mx_power_use_define_index_clock_slew

`< 0 | 1 >` Enables use of define index clock slew for power arc. Default: `1`

mx_preprocess

`< 0 | 1 >` Enable preprocess speedup. Default: `1`

The Preprocess Flow prunes FastSim decks to provide initial conditions for better performance and capacity. It needs the RCDB flow enabled. *(See RCDB flow Application Note.)* Only UltraSim is supported as the partition simulator.

mx_probe_peak_currents

`< 0 | 1 >` Triggers calculation for inrush and peak current as well as attribute settings in the final library.
Default: `0`

To enable peak current probing use:

```
set_var mx_probe_peak_currents 1
```

The peak of the sum of all power supply currents is reported. To limit probing to a specifc set of tables, use:

```
set_var mx_peak_current_tables {<list_of_tables>}
```

If `mx_peak_current_tables` is not specified, the default case, then all power tables are used.

Peak current is defined in the library section of the generated liberty file as a float cell attribute and reported in the cell section:

```
library (my_liberty) {
    define (peak_current, cell, float);
    cell (my_cell) {
    peak_current : 12.345;
    }
}
```

mx_probe_real_rails_as_virtual

`< true| false >`	If enabled, the tool models ground and normal rail as virtual rail to be shifted across LUT similar to power virtual rails. Default: `false`

mx_probes_report

`<filename>`	Specifies file(s) to report the results of automatic probing. User may specify a full path name.

Liberate MX produces the following two reports:

- `<filename>`: Contains the results of all possible probes found statically.
- `<filename>.red`: Less-verbose ("reduced") report filtered down based on switching activity.

mx_process_probe_relprobe_intersection

`<0 | 1 | 2>` Controls the post-processing of constraint probes to discard the output of the scan-path latches.

Default: `1`

- `0` Does no post-processing of the constraint probe locations.
- `1` Post-processes the constraint probing locations to discard the output of the scan-path latches. (Default)
- `2` Invokes a more comprehensive method to post-process the constraint probe locations to eliminate the output of the scan-path latches excluding regions containing the memory array.
- `3` Invokes a more comprehensive method to post-process the constraint probe locations to eliminate the output of the scan-path latches including regions containing the memory array.

mx_process_probe_relprobe_intersection_max_paths

`<value>` Used when `mx_process_probe_relprobe_intersection` is set. The value defined here sets max intersection paths that are used.
Default: `10`

mx_process_probe_relprobe_intersection_max_depth

`<value>` Used when `mx_process_probe_relprobe_intersection` is set. The value defined here sets max intersection depth that is used.
Default: `20`

mx_pulse_width_threshold

`<Float>` Specifies alower bound to be considered for a valid pulse when recording wire activity.
Default: `2e-10`

mx_push_probe_inside_array

`<0 | 1 >` Specifies whether Liberate MX can consider probes within the bitcell array. Default: `0`

- `0` Do not consider probes for constraint arcs that are a part of the bitcell array. (Default)
- `1` Consider probes within the bitcell array.

mx_read_spice_exit_on_missing_file

`<0 | 1 >` Informs Liberate MX to issue an error and exit if there is a file missing during read_spice. Default: `0`

- `0` Don’t exit if there is a missing file. (Default)
- `1` Issue an error and exit if there is a missing or unreadable file during the `read_spice` phase.

Example:

```
set_var mx_read_spice_exit_on_missing_file 1
```

This parameter must be set before `read_spice`.

mx_remove_false_ic_group

`<0 | 1 >`

Controls whether measurement results generated from false initial condition arcs will be included in the data base. For backward compatibility only. Default: `1` (Always remove false initial condition groups.)

- `0` — Allows (potentially) false initial condition groups. For backward compatibility only.
- `1` — Always remove false initial condition groups. (Default)

Because dynamic information for memory cores is usually unknown, partitions that cover memory core nodes will have multiple `define_arc` commands to ensure that all possible initial conditions for the memory cores are characterized. The normal behavior is for one – and only one – of these arcs to generate successful measurements, and all others to fail.

However, it may happen that some of the bad initial condition arcs will generate results. In this case, proper results are identified as the one being the closest to the FastSPICE result; all others are removed from the database.

An informational message is always printed to the standard output when results for false initial condition arcs are removed from the data base.

mx_remove_rc_pincap

`{"all" | "none" | "rail" | net_name}`

Controls removing parasitic RC networks on a per-net basis. Default: `"rail"`

- `all` — Remove parasitic RC networks for all nets.
- `none` — Do not remove any parasitic RC networks.
- `rail` — Remove parasitic RC networks for rails (vdd, vss). (Default)
- `net_name` — Remove parasitic RC network for the specified net.

This parameter controls removing parasitic networks for a partition. This may be applied on a per-net basis, for example supply rails, or specific nets. User may use the keywords `"all"` or `"none"` or supply a list of nets.

Note: `"all"` or `"none"` will take precedence if used together with net names.

Example:

```
# Remove parasitic RC network for rails
set_var mx_remove_rc_pincap "rail"
```

mx_remove_rc_timing

`{"all" | "none" | "rail" | net_name}`

Controls removing parasitic RC networks on a per-net basis. Default: `"rail"`

`all`	Remove parasitic RC networks for all nets.
`none`	Do not remove any parasitic RC networks. (Default)
`rail`	Remove parasitic RC networks for rails (vdd, vss).
`net_name`	Remove parasitic RC network for the specified net.

This parameter controls removing parasitic networks for a partition. This may be applied on a per-net basis, for example supply rails, or specific nets. `"all"` or `"none"` or supply a list of nets.

Note: `"all"` or `"none"` will take precedence if used together with net names.

Example:

```
# Remove parasitic RC network for specified nets
set_var mx_remove_rc_timing {myNet123 myNet456}
```

mx_report_starnet

`{"" | "probes" | "starnetmap"}`

Controls removing parasitic RC networks on a per-net basis. Default: `""`

`probes`	Prints star net (*|NET) name.
`starnetmap`	Stores start net map.

mx_retaining_time

`< 0 | 1 >` Generates – and characterizes – partitions for `retaining_rise` / `retaining_fall` arcs. Default: `1`

Note: Retaining (a.k.a. valid time) arcs are characterized only when user-defined.

Example:

```
# Do not generate / characterize retaining time arcs
set_var mx_retaining_time 0
```

mx_require_whitebox_model_file

`< 0 | 1 >` Specifies whether a model file is to be read. Set this parameter to `0` if you do not want a model file to be read. By doing so, you eliminate the need to set the mx_whitebox_model_file parameter to an empty `.scs` file. Default:`1`

mx_ring_large_ccc_max_paths

`<value>` Controls the number of partial paths considered while creating input pin cap CCC partition for pincap characterization. This is helpful for cases where input has too large CCC connected and results in bigger pincap partition and higher runtime. This parameter works with path depth defined by `mx_ring_large_ccc_max_path_depth`.
Default: `1`

This parameter has no effect unless the logic driven by the input exceeds the maximum transistor limit.

mx_ring_large_ccc_max_path_depth

`<value>` Controls path depth to be considered while creating pin cap CCC partitions. This is used with `mx_ring_large_ccc_max_paths` in case of huge CCC connected to primary input.

Default: `10`

This parameter has no effect unless the logic driven by the input exceeds the maximum transistor limit.

mx_ring_large_ccc_min_pass_gate_xtr_cnt

`<value>` Specifies the minimum number of pass gates transistors to be traversed when creating partitions for input capacitance or CCSN. Pass gates beyond the specified minimum value will not be included in these partitions.

Default: `-1`

`-1` implies that the check is off and any number of pass gates may be followed.

mx_ring_max_side_inputs

`<integer>` Specifies the maximum amount of side inputs.

Default: `3`

mx_ring_max_xtr_cnt

`<integer>` Specifies the maximum transistor size in a single CCC for pin capacitance partition.

Default: `200`

mx_ring_model_fold

`< 0 | 1 >` Turns on topological matching of logic. Default:`1`

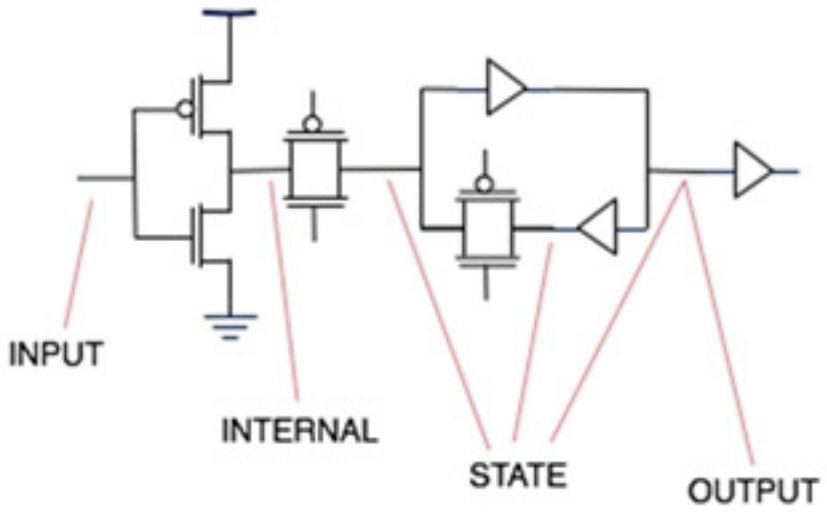
Use this parameter use to turn on topological matching of logic driven by primary inputs / driving primary outputs belonging to the same bus, to speed up pincap and CCSN characterization

mx_scale_virtual_noise_fraction

`<value>` — Introduces pessimism in the final library numbers for the delay and retain arcs using virtual rails. This parameter takes values between `0` and `1`. To disable this parameter, set it to `-1`. Default: `0.01`.

mx_seq_probing

`<list>` — Sets the node type to consider on a sequential component as possible candidates for data path probes in setup/hold characterization. Default: `state output internal input`.

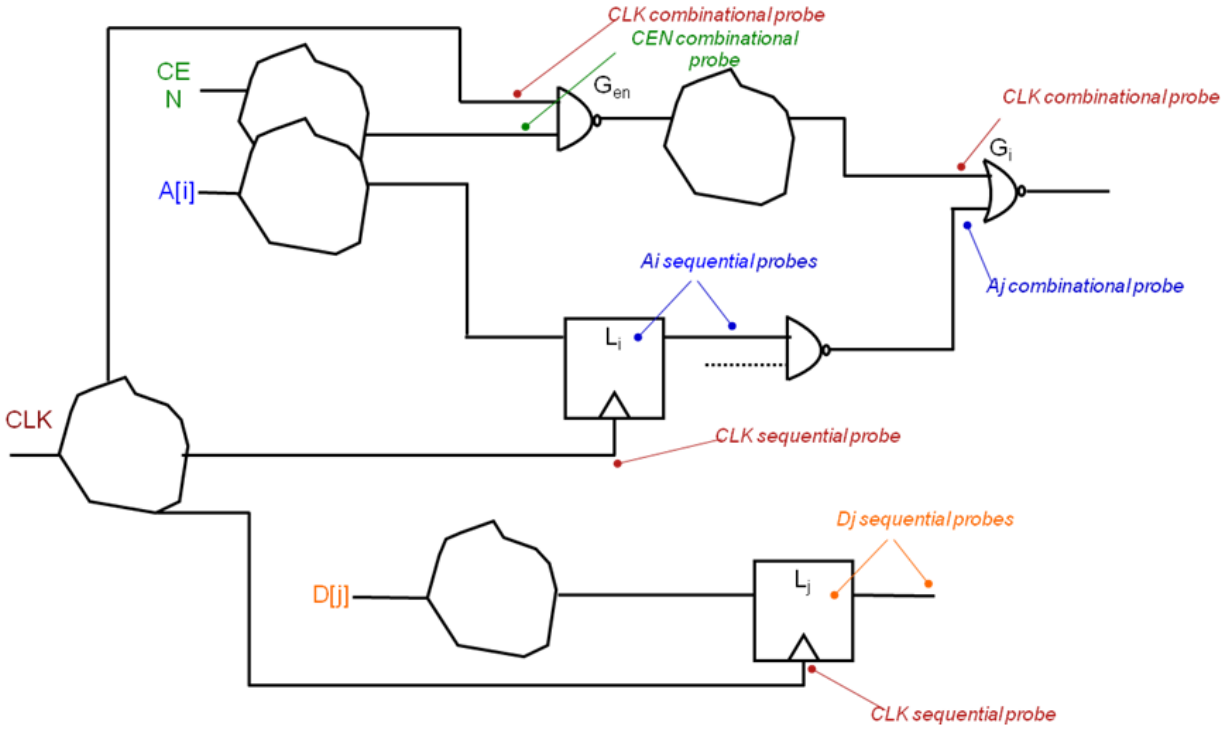


mx_setup_seq
mx_hold_seq

mx_setup_comb
mx_hold_comb

`{<identifier> }` Flags drive the automatic probe identification step in identifying candidates for setup and hold constraints probes.

When probing for constraint characterization, Liberate MX automatically probes the following structures (refer to the figure below): 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 MX uses first-level probes for hold characterization for all pins and both first-level and second-level probes for setup characterization for all pins. Internal and output nodes of sequential elements are used as non-clock probes and clock inputs to the sequential element are used as clock probes. This corresponds to the following values for the flags:

```
set_var mx_setup_seq  "all"
set_var mx_hold_seq   "all"
set_var mx_setup_comb "all"
set_var mx_hold_comb  "all"
```

It is often the case that 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 picture above, the user would specify:

set_var `mx_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.

Moreover, you can specify that the non-clock inputs to sequential elements also be used as probes – using keyword "*:probe_inputs"* modifier after pin name. Referring to the picture above, you would specify:

set_var `mx_setup_seq` *"all Dj: probe_inputs"*, to use inputs to sequential gate L_j as probes for D_j, but leave the default for other pins (i.e. internal and output nodes only).

You can also specify whether sequential slave (i.e. second level) nodes should be considered when probing – usually for setup. Using set var `mx_setup_seq` and post-fixing he :slave keyword to the name allows for a pin or set of pins to be probed at second-level sequential elements when characterizing setup constraint.

In general, to allow for both first (master) and second (slave) level sequential probing for all inputs, you would specify the syntax as follows:

```
set_var mx_setup_seq "all all:slave"
```

This example uses first-level sequential for all pins' setup characterization, except for pin "wtz", which uses second-level sequential as well:

```
set_var mx_setup_seq "all wtz:slave"
```

mx_simulation_interval

<value>	Value, in seconds, of the simulation interval 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 Specifying Input Stimuli.)
Arc-based: using an `-attribute` to `define_arc`.

If there are multiple definitions of the simulation interval, the precedence order is as follows (1st in list means highest precedence)

1. Arc-based(highest)

2. Table-based
3. Global - i.e. `set_var`(lowest)

Example:

```
# if max clk->out delay is expected to be around 5.2 ns …
set_var mx_simulation_interval 12e-9
```

mx_skip_autoprobing

`< identifier | "array">`	Skips autoprobing for channel connected regions. Default: `none`

This command forces Liberate MX to avoid automatic probing on specific channel connected regions (CCC). The command accepts either a string describing the type of logic to avoid (for example: my_cell_25) or the keyword "array".

Example:

```
set_var mx_skip_autoprobing "array"
```

mx_skip_print

`<regular expression>`	Used to reduce FastSPICE runtime by filtering out nodes that should not get monitored. Default: `none`

This parameter accepts one or more regular expression to filter nodes that should not get monitored in the FastSPICE simulation. This is to reduce FastSPICE runtime / disk space / MX footprint when reading results back. Regular expression are in TCL style (see regexp TCL command documentation).

Issuing the command multiple times causes concatenation of patterns rather than overwrite. Command must be issued before `char_macro`. Node names / patterns should refer to pre-layout names.

Example:

```
# No activity needed (and therefore skipping them in FastSPICE) on
# nodes that match pattern below:

set_var mx_skip_print \
"XLPCAM/XLPCAM/XMEMARRAY_(.*)/XCG64MEMARRAY(.*)/XCG64ROW(.*)/XSRCH_B/NET200"
```

mx_spv_api

`< 0 | 1 >`	Turns on/off generation of API scripts and data to be used in conjunction with the SpiceVision GUI from Concept Eng (third-party product). Default: `0`

mx_status_log_file

<filename>	Generates a log file summarizing the status of a Liberate MX run.

This parameter must be set before the `read_spice` command.

mx_sst2_output_name

`<string>`	Specifies the waveform filename containing the output data in SST2 format. This file is written out by the simulator and is read in for partition. Default: `transient1.tran.fsdb`

This parameter must be set before the char_macro command.

mx_switch_wire_threshold

`<value>` — Specifies the minimum voltage change for a wire to be considered switching.

It is recommended to set `mx_switch_wire_threshold` to 30 percent of the power supply value.

Default: `DBL_MAX`

mx_switch_wire_threshold_ratio

`<value>` — Applies `switch_wire_threshold` as a percentage of the absolute value of power rail.

mx_switch_wire_threshold_ratio

`<value>` — Applies `switch_wire_threshold` as a percentage of the absolute value of power rail.

mx_timing_ldb_reuse

`< 0 | 1 >` — Reuse timing LDB from previous run. Default: `0`

See mx_ldbs_reuse for description of all LDB reuse parameters.

mx_timing_report

`<0 | 1>` — Controls generation of detailed timing reports. Default: `0`

When set to `1`, Liberate MX generates the following two detailed timing reports:

- `sim.top.rpt` that reports the data from fastsim.
- `sim.rpt` that reports the data from characterization runs.

mx_timing_report_debug

`< 0 | 1 >` Generates timing reports with extra columns. Default: `0`

When set to `1`, Liberate MX generates the `sim.top.rpt` and `sim.rpt` timing reports with two extra columns, `Incr` and `Tran`.

- **Incr**: From startpoint to current net incremental time (50% ~ 50%).
- **Tran**: Current net transition time (30% - 70%).

For example:

```
PointPath PulseWidthIncrTran
clk 0.02500 r 0 0.02000
MZY/I0/cgr/I2/Mn1_G 0.06577 f 0.04077 0.01894
MZY/I0/cgr/I1/MI6_G 0.06714 f 0.00137 0.02042
MZY/I0/cgr/I4/Mn1_G 0.13050 r 0.06336 0.04266
```

mx_total_active_load_channel_thresh

`<value>` Simplifies a partition by substituting a passive load for active devices. If the sum of the channel equivalent load on a net is larger than the specified threshold, the substitution is not performed.
Default: `1e-16`

mx_total_active_load_gate_thresh

`<value>` Simplifies a partition by substituting a passive load for active devices. If the sum of the gate equivalent load on a net is larger than the specified threshold, the substitution is not performed.
Default: `1e-16`

mx_transres

`<value>` Adds the option `transres=value` to the `.tran` command in the partition deck. If set to `-1`, the `transres` option will not be added to the `.tran` command.
Default: `-1`

Note: This parameter must be specified before char_macro.

Example:

```
set_var mx_transres 1e-15
```

The `.tran` command in the partition deck will be:

```
.tran 1.00e-12 'tran_tend' transres=1e-15
```

mx_update_constraint_pin_threshold

`< 0 | 1 >` Controls whether the constraint arc pin threshold should be updated. Default: `""`

mx_use_complex_ccc

`< 0 | 1 >` Enables combined grouping of closely related circuit components for probing purposes. Default: `1`

mx_use_fastsim_value_for_partition

`{"none" | "all"}`

Specifies whether the partsim results should be used for the specified arcs. Default: `none`

`none` Do not use partsim results. (Default)

`all` Use partsim results for timing arcs that have the define_arc attribute `altos_use_fastsim_value_for_partition` set to `1`.

Each delay/retain/setup/hold arc that is defined using `define_arc -attribute {altos_use_fastsim_value_for_partition 1}` uses adjusted values that match the timing value from the fastsim simulation.

mx_verbose

`< 0 | 1 >` Prints basic flow and runtime information. Default: `1`

Example:

```
#report basic flow info
set_var mx_verbose 1
```

mx_validation_fastsim_slew_load_off_grid_action

`{"ignore" | "warning" | "error"}`

In Liberate MX validation flow, the values of `mx_fastsim_input_slew`, `mx_fastsim_clock_slew` and `mx_fastsim_load` are checked to see if the values are in define_template or define_index. This parameter controls what action will Liberate MX take when an off grid value is detected. Default: `warning`

`ignore`	Does not perform the off grid check.
`warning`	If Liberate MX detects an off grid value, the program will display a warning message and continue the validation flow. (Default)
`error`	If Liberate MX detects an off grid value, the program will display an error message and exit.

Note: This parameter should be set before `validate_macro`.

Example:

```
set_var mx_validation_fastsim_slew_load_off_grid_action "error"
```

mx_validation_auto_report_mode

`<0 | 1 | 2>` Specifies the mode for auto shift index and delay comparison reporting. Default: `0`

- `0` Disables auto shift index.
- `1` Enables auto shift index (Liberate MX ignores mx_validation_shift_idx).
- `2` Enables auto shift index (Liberate MX ignores mx_validation_shift_idx) and creates a report for delay comparison in `mxv.nod.csv` for the output pin.

mx_validation_shift_idx

`<integer>` Specifies the index number of shifting for stress when running the 'at speed' test. Default: `0`

When set to `0`, Liberate MX does not perform multiple index checks. If an integer greater than `0` is set, multiple index check is enabled.

Example:

Relaxed:, L, RF, L, L, L, L,

At Speed:, L, L, L, R, H, F,

"R,H,F" in 'at speed' is performed after 2 index shift after relaxed "RF". If you set `mx_validation_shift_idx` to `2`, then there is no outlier between relaxed and at speed tests.

mx_verify_table

`<0 | 1 | 2>` Enables checks to be performed on the table files to ensure correctness. Default: `0`

- `0` Does not check the table files. (Default)
- `1` Performs checks and issue warnings if tables are incorrect.

2	Performs checks and exits with error if the tables are incorrect.

mx_virtual_rail_auto_mode

<0 | 1 | 2>

Automatically sets the virtual power supplies in a design. Default: 2

0	There are no virtual rail findings. It is equivalent to mx_find_virtual_rail set to 0.
1	The virtual.rpt report is created. However, there is no automatic setting. User can refer to the report and set the virtual rails accordingly.
2	Automatic settings along with the virtual.rpt report. (Default)

This parameter enables you to find (mx_find_virtual_rails) and define virtual power nodes in a design.

This parameter works when mx_find_virtual_rails is set to 2.

mx_virtual_rail_opposite_device_minimum_factor

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: 250

For example:

```
wire_name   drive_pmos   drive_nmos   other_unset_wires
VDD_1        13038          8             set
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.

VSS_2 where 508 PMOS are connected, which is below the default value and because of 1038 NMOS connections, is not set as a virtual net.

mx_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.

Example:

```
# Defining minimum number of mos to "70"
set_var mx_virtual_rail_minimum_xtrs 70
```

mx_virtual_rail_modeling_mode

`< 0 | 1 >`	Enables the new virtual rail modeling method. Default: `0`

When setting up the simulation to use the new code, `mx_reuse` should not be used. Start the simulation from scratch because the new code generates a new `sim.probe` file under `mx_reuse` directory.

mx_virtual_rail_stack_xtrs

`< 0 | 1 >`	Auto-generates virtual rail for the stack of common-gate NMOS/PMOS. Default: `0`

mx_virtual_rail_waveform_shift_threshold

`double`	Specifies the input threshold that will be used to apply virtual rail waveforms to the partition. Default: `0`

mx_whitebox_active_coupling_threshold

`double`	Sets the threshold coupling between victim/aggressor, which will be considered in dynamic partitioning. Default: `-1`

For example, if total couple between aggressor A and victim V is greater than threshold, aggressor A will be transversed during dynamic transversal of victim V.

mx_whitebox_active_wire_threshold

`double`	Sets the threshold for a wire to be considered active. Default: `0.01 volts`

mx_whitebox_model_file

`< 0 | 1 >`	Forces the model file used in both partitioning and characterization to point to the specified file. Default: `0`
`{filename}`	Specifies the full path name for the file.

mx_whitebox_model_file_format

`<string>`	Specifies the format of model file. Default: `""`

mx_write_greybox_dot_meas_file

`< 0 | 1 >`	Controls generation the measure file. Default: `0`

mx_whitebox_ring_in_depth

`< 0 | 1 | 2 >`	Limits pin capacitance partitioning to a single CCC level. When set to `2`, allows extra CCC levels for pin capacitance partitioning until the transistor size specified in mx_ring_max_xtr_cnt is reached. Default: `2`

mx_zip_partition_deck

`< 0 | 1 >`	Controls generation of partition decks in gzip format. Default: `0`

When set to `1`, it generates partition decks in gzip format. You can use this parameter when characterizing large extracted netlists.

This parameter must be used before the `char_macro` command.

mx_xps_inc_str

`<string>` — Sets the correct include command for dspf netlists. If all the netlist in `read_spice` are `*.dspf` files, set `mx_xps_inc_str` to `'.dspf_include'`. Default: `'.include'`

packet_arc_job_retries

`< 0 | 1 | 2 >` — Specifies the number of times to resubmit a failed job. When the maximum number of retries is reached, the job is marked as failed and will not be submitted again. Default: `0`

packet_arc_xtor_count_based_cell_sort

`< 0 | 1 | >` — When enabled, applies weightage on the arc in bottom runs for job submission in the arc packet flow. Default: `0`

set_var_failure_action

`<warning | error>` — Notifies the `set_var` command how to consider a failure. Default: `warning`

- `error` — When a `set_var` fails, an error message is issued and subsequent commands that would result in characterization or library generation (for example, char_memory) are suppressed. Subsequent `set_var` commands are still allowed so they can be checked for correctness.
- `warning` — A warning is issued when `set_var` fails. The failed `set_var` is ignored and execution continues.

This parameter must be set prior to any other `set_var` command.

Sample Message:

The simulation on client `<client>` has been running for `180` minutes. The simulation might be stalled, waiting for a license, or terminated. If the simulation has terminated and the client

is not responding, you might need to restart the characterization job. Check the simulation status, for example, check sim.lis file if using external simulator.

set_virtual_rail_threshold

`<0 | 1>` Executes multi slew runs with a similar load and calculates the threshold that will be used to shift the virtual rail.
Default: `0`

- `0` Does not execute the multi slew runs.
- `1` Automatically executes the multi slew runs to calculate the threshold.

Example:

```
arctypes timing

fastsim_clock_slew 0.0792 0.005
fastsim_input_slew 0.294 0.005
fastsim_load 0.5

set_virtual_rail_threshold
```

This will run two simulations:

- First for clock slew 0.0792ns and input slew 0.294ns and load 0.5fF.
- Second for clock slew 0.005ns and input slew 0.005ns and load 0.5fF.

The log file will show the new calculated value for rise and fall threshold. The new values will be automatically applied in the characterization setup for the bottom simulations.

setup_constraint_probe_lower_fall

`<float>` Sets the lower threshold at falling edge for constraint arcs.
Valid values: `0.02~0.98`
Default: `-1`

setup_constraint_probe_lower_rise

`<float>` Sets the lower threshold at rising edge for constraint arcs.
Valid values: `0.02~0.98`
Default: `-1`

setup_constraint_probe_upper_fall

`<float>` Sets the upper threshold at falling edge for constraint arcs.
Valid values: `0.02~0.98`
Default: `-1`

setup_constraint_probe_upper_rise

`<float>` Sets the upper threshold at rising edge for constraint arcs.
Valid values: `0.02~0.98`
Default: `-1`

supply_info

`<0 | 1>` Controls the printing of messages that summarize the values set for the following commands: `set_vdd`, `set_gnd`, `set_pin_vdd`, and `set_pin_gnd`.
Default: `0`

`0`	Does not print the messages.
`1`	Prints the messages.

Example:

```
set_var supply_info 1
```

Liberate will print messages such as following:

```
(set_vdd): Pin 'vdd' is set to 20 V for cell 'test1' -no_model 'False' -ignore_power
'False' -type 'internal' -attributes {} -combine_rail 'False' -include {}
(set_pin_vdd): Pin 'A1' is set to 3 V for cell 'level_shifter_3to1' -add_supply
'True' -leakage_add_to_supply '' -supply_name {VDD3}
```

virtual_rail_waveform_probe

`<worst | ccc_worst>` Controls the selection of probe point in the virtual rail obtained from dynamic partition to be applied to the partition netlist simulation.

- `worst` Identifies the worst captured PWL among all virtual nodes and use it to drive PWL for all CCC [Channel Connected Regions]
- `ccc_worst` Identifies the worst PWL among all virtual nodes of each CCC [Channel Connected Regions] and apply it to respective CCC.

A

Truth Table Format

Specifying Input Stimuli

In the Liberate MX Memory Characterization tool, 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 `-thread` option of the char_macro command to control the number of threads.

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

Specifies which FastSPICE engine will be used to simulate the table file. Overwrites the output of the char_memory -charsim command.

For example, the following command 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 MX 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 parameter mx_fastsim_auto_ic for the table file it is specified in. If not specified, the default value is given by the value of mx_fastsim_auto_ic, which has a default of true.

simulation_interval

<value>	Specifies the simulation interval, in seconds. Default: 10ns

The simulation_interval command overwrites mx_simulation_interval for the table it is specified in. This can be used in a table of any type: power, leakage, timing, measure, etc.

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 is used for any other table that does not have a `simulation_interval` command).

fastsim_clock_slew

`<double>`	Specifies the clock slew to be used in FastSPICE simulation. For more information, see mx_fastsim_clock_slew. Default: `-1ns`

fastsim_input_slew

`<double>`	Specifies the input slew to be used in FastSPICE simulation. For more information, see mx_fastsim_input_slew. Default: `-1ns`

fastsim_load

`<double>`	Specifies the load to be used in FastSPICE simulation. For more information, see mx_fastsim_load. 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 SST2 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 MX to look for previous fastsim results on this table file. fastsim_reuse results are stored for each run inside the directory:

```
./mx_fastsim – with name <cellname>.<tablename>.alwf
```

This overwrites `mx_fastsim_reuse`.

monitor_memcore

`<inter | intra | all>`	Special debug flag to provide table-level control. For more information, see mx_monitor_memcore. Default: `inter`

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 parameter is used to specify truth table files FastSPICE simulator options in Liberate MX. 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_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_distributed_sim

<fastsim_disctributed_sim>

Passes switches and options to external programs such as a simulator or job management system. This works same as `mx_distributed_sim` and overrides global setting, if defined.

Example:

in TCL script

```
set_var mx_distributed_sim "bsub -q queue -P project -W 600 -n 1 -R \"(OSREL==EE50
|| OSREL==EE60) rusage\[mem=8000,swp=8000\] span\[hosts=1\]\" %C"
```

in table file delay.tbl

```
fastsim_distributed_sim bsub -q queue -P project -W 600 -n 1 -R "(OSREL==EE50 ||
OSREL==EE60) rusage[mem=4000,swp=8000] span[hosts=1]" %C
```

Simulation job for `delay.tbl` will request 4GB of memory. Simulation jobs for any other table requests 8GB.

fastsim_model_include

<model_path_name>	Specify the full path name of a SPICE model file for power or leakage characterizations. Default: none

This specifies a full path to a model file to be used with fastsim for power or leakage characterizations.

Liberate MX 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
```

fastsim_init_virtual_rails

`<fastsim_init_virtual_rails>`

Initializes virtual rails in the full instance simulation. Allowed values range between `0` and `1`, where `1` corresponds to the expected value of the virtual rail and `0` corresponds to the opposite value.

fastsim _output

`<pin_name>` Specifies the output pin. Useful when you want specify a bidi pin as output instead of an input.

mxv_check_mpwh

`mxv_check_mpwh` Enables the stressing of minimum pulse width high in the validation run. Allowed values range between `0` and `1`. If enabled along with `mxv_check_mpwl`, both minimum pulse width high and low will be stressed and minimum period may not be satisfied.

mxv_check_mpwl

`mxv_check_mpwl` Enables the stressing of minimum pulse width low in the validation run. Allowed values range between `0` and `1`. If enabled along with `mxv_check_mpwh`, both minimum pulse width high and low will be stressed and minimum period may not be satisfied.

Guidelines for Writing Truth Table for Multiport Memories

Following are some of the guidelines for writing truth table for multiport memories.

- When there are more output buses and need measures/arcs terminating to both outputs by “assign values”

  For example, there are 2 output buses `output_0` and `output_1` and you would like to measure delay/retain on both buses.

```
# only on one of the output bus
table measure_output0_only
pins        input   ouput0
write0      R       X
write1      R       X
read0       R       X
read1       R       D
read0       R       D
endtable
# both output buses in same cycles
table measure_both_output0_and_output1
pins        input     ouput0    output1
write0      R           X           X
write1      R           X           X
read0       R           X           X
read1       R           D           D
read0       R           D           D
endtable
```

- For `clk hidden power` arc ensure only one of the output switches at a time. This is mostly required to be taken care with multiport memories where multiple outputs can switch due to different read and write operations on ports simultaneously.
- It is recommended to start with one of the port and try to get most of the arcs (Delay/ power) correctly. Once you are sure that the results are fine, you can just replicate the same table by changing bus names for other ports.

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
K *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 MX 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 may be given a default value, which can simplify entering data into a table. Default values may be specified with the Tcl variable, mxtable_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` will be used if there is no explicit value assigned, or wherever the table has a question mark (?) for a value.

Example:

```
table seldesel
pins            clk    cs      bw      din    adr    dout
dontcare_value oe 1
# oe=1 will be used throughout this table
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)
```

Three-State Enable or Disable Arcs

Three state characterization is performed in MX if both following conditions are met:

- Three state enable and disable lines exist in `define_table`
- User-defined arc of type enable/disable are available (`-type enable` or `-type disable`)

A table line is considered an enable line for a specific output when the line contains the identifier "enable" or "E" or "e" in the column corresponding to that output. A table line is considered a disable line for a specific output when the line contains the identifier "disable or "Z" or "z" in the column corresponding to that output.

Note: The table file containing enable and disable lines must also contain `arctypes` identifier to cover `enable` and `disable` arcs.

Example:

```
arctypes delay enable disable
table en_dis
pins    oe clk bw adr dout
disable F  ?  H  ?  Z
set_1   L  R  H  H  X
read_1  R  H  H  H  E
disable F  ?  H  ?  Z
set_0   L  R  H  L  X
```

```
read_0  R  H  H  L  E
endtable
```

Truth Table Example

```
# Specify a full table for a sram with bist and asynch. In particular, table
# delay describes basic write/read operations controlled by rising edge of CLK:
# write D = 11111111111 in to address ADR = 111111111
# write D = 00000000000 in to address ADR = 000000000
# read from address ADR = 000000000 and measure delay
# read from address ADR = 111111111 and measure delay
# write D = 00000000000 in to address ADR = 111111111
# write D = 11111111111 in to address ADR = 000000000
# read from address ADR = 000000000
# read from address ADR = 111111111

arctypes timing power
fastsim spectre
fastsim_cmd_option -64 +spice +xps=s3 +cktpreset=sram_pwr -format fsdb


table delay
pins    BISTE   ME      WE      OE      CLK     WEM     ADR     D       Q
write11 l       h       h       l       R       h       1       1       x  #Dummy
write00 l       h       h       l       R       h       0       0       -
read0_  l       h       l       h       R       l       0       ?       b
read1_  l       h       l       h       R       l       1       ?       b
write10 l       h       h       l       R       h       1       0       -
write01 l       h       h       l       R       h       0       1       -
read0_  l       h       l       h       R       l       0       ?       b
read1_  l       h       l       h       R       l       1       ?       b
endtable

table delay_bist
pins    BISTE   TME     TWE     TOE     CLK     TWEM    TADR    TD      Q
write11 h       h       h       l       R       h       1       1       -
write00 h       h       h       l       R       h       0       0       -
read0_  h       h       l       h       R       l       0       ?       b
read1_  h       h       l       h       R       l       1       ?       b
write10 h       h       h       l       R       h       1       0       -
write01 h       h       h       l       R       h       0       1       -
```

```
read0_  h       h       l       h       R       l       0       ?       b
read1_  h       h       l       h       R       l       1       ?       b
endtable

table constraint
pins    CLK     ADR     D       WEM     WE      OE      ME      AWT     BISTE     Q
const   R       b       l       l       l       l       h       l       l       -
const   R       l       b       l       l       l       h       l       l       -
const   R       l       l       b       b       l       h       l       l       -
const   R       l       l       l       l       l       b       l       l       -
const   R       l       l       l       l       l       h       l       0       -
endtable

table constraint_bist
pins    CLK     TADR    TD      TWEM    TWE     TOE     TME     AWT     BISTE     Q
const   R       b       l       l       l       l       h       l       h       -
const   R       l       b       l       l       l       h       l       h       -
const   R       l       l       b       b       l       h       l       h       -
const   R       l       l       l       l       l       b       l       h       -
const   R       l       l       l       l       l       h       l       1       -
endtable

table asynch
pins    BISTE   AWT     D       WEM     TD      TWEM    OE      TOE     Q
asynch  l       R       b       b       ?       ?       h       l       b
asynch  h       R       ?       ?       b       b       l       h       b
endtable
```

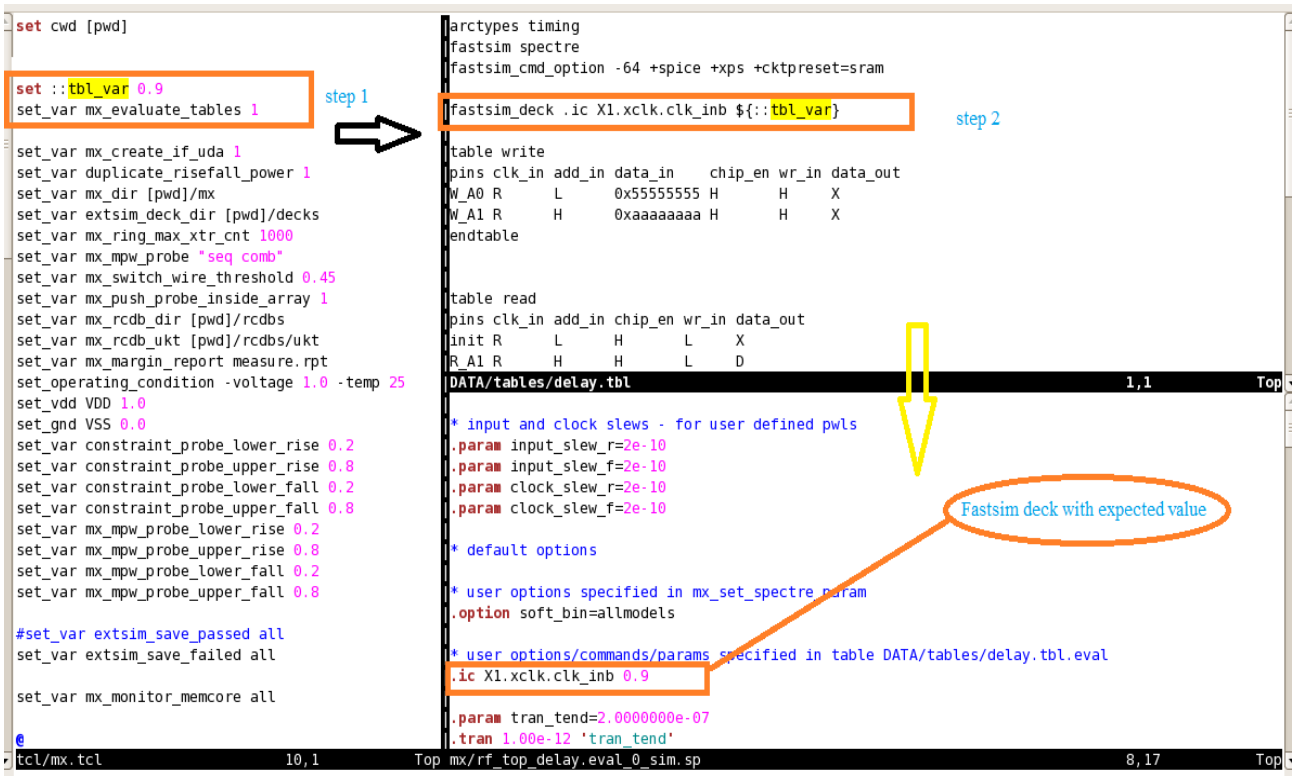
Using Tcl Variable Inside the Table Files

You can use Tcl variable evaluation inside the Liberate MX table files. The Tcl variable needs to be:

- defined at top name space level (i.e. :: prefix)
- defined in Tcl scripts before the `define_table` command or the `-mxtable` option of the `define_cell` command.
- have the same name in the Tcl scripts and the table file.

If a table evaluates to a value different from the original, a `postfix.eval` is appended to the original table name passed to the run. The new table is stored in the same location as the original table. However, if the location is read-only, the table is left unchanged.

The Tcl variable functionality is controlled by variable `mx_evaluate_tables`. The default value of the variable is 0, that is, no Tcl evaluation is performed on the Liberate MX table files. The following figure shows the use of Tcl variable evaluation inside the Liberate MX table file.



B

Specifying Memory Core Cells

In Liberate MX, memory core nodes need to be identified during automatic probing step. By default, the tool looks for a 6T SRAM cell. The default can be overwritten by specifying one or more core cell description files. Each file - typically just one per macro - contains the description of a core cell. Each cell is defined by one storage group and one or more port groups. In turn, each group is a collection of devices - transistors (for SRAMs and DRAMs) and capacitors (for DRAMs). Refer to the syntax and examples that follow.

Note: This step only effects automatic probing – i.e. understanding of what's a memory core node, what is a bit line pair, etc. - and not partitioning, which is based on switching information only.

Syntax:

```
mxcore <core_name> {
    <group_id> {
        <subgroup_id> {
            device {
                model:<model_id>
                side:<side_id>
                ctrl:<ctrl_id>
                use:<use_id>
                sizel:<l_id>
                sizew:<w_id>
                sizec:<c_id>
    }}}}
```

where:

`<core_name>`	<any string> Identifies the core cell.
`<group_id>`	storage, port Identifies the group type.

<subgroup_id>	sram, dram, rom, write, read, writeread Identifies the subgroup type
<model_id>	n, p, c Identifies the device type
<side_id>	1, 0 Identifies the "side" of the core cell the device is in. This is used whenever there are back-to-back drivers forming the storage group of the cell and is useful to refer to a right and a left side of the cell. Choice of 0, 1 is fully arbitrary - but must be consistent among different devices. See examples below.
<ctrl_id>	word, core Identifies device's gate terminal connectivity: controlled by wordline, controlled by mem core node.
<use_id>	pass, pull-up, pull-down, storage Identifies how the device is used; as a pass transistor, a pull up one, a pull down one, a storage.
<l_id>	double Specifies L for device - when <model_id> is n or p
<w_id>	double Specifies W for device - when <model_id> is n or p
<c_id>	double Specifies C for device - when <model_id> is c

Example:

8T SRAM - dual port – see Figure 2 – 8T core cell2 below

```
mxcore 8T_SRAM  {
    storage {
        sram {
            device {
                model:nmos
                side:1
                ctrl:core
                use:storage
            }
            device {
                model:pmos
                side:1
```

```
            ctrl:core
            use:storage
        }
        device {
            model:nmos
            side:0
            ctrl:core
            use:storage
        }
        device {
            model:pmos
            side:0
            ctrl:core
            use:storage
        }
    }
}
port {
    writeread {
        device {
            model:nmos
            side:1
            ctrl:wordbit
            use:pass
        }
        device {
            model:nmos
            side:0
            ctrl:wordbit
            use:pass
        }
    }
}
port {
    writeread {
        device {
            model:nmos
            side:1
            ctrl:wordbit
            use:pass
        }
```

```
device {
    model:nmos
    side:0
    ctrl:wordbit
    use:pass
}}}}
```

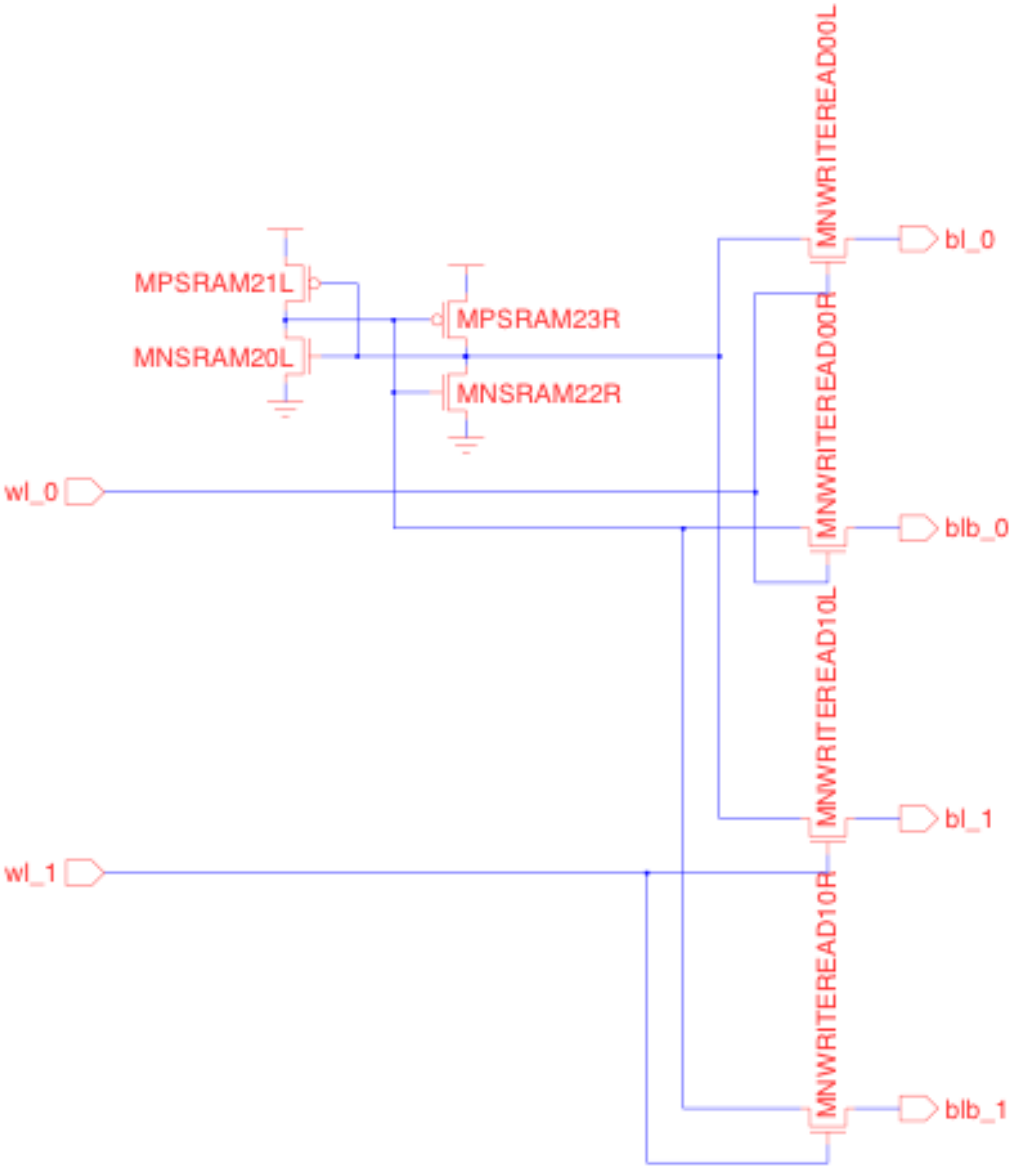


Figure 2 – 8T core cell

Example:

```
10T SRAM - dual port - pass write, pull-down read - see Figure 3 – 10T core
cell3.


mxcore 10T  {
    storage {
        sram {
            device {
                model:nmos
                side:1
                ctrl:core
                use:storage
            }
            device {
                model:pmos
                side:1
                ctrl:core
                use:storage
            }
            device {
                model:nmos
                side:0
                ctrl:core
                use:storage
            }
            device {
                model:pmos
                side:0
                ctrl:core
                use:storage
            }
       }
    }
    port {
        write {
            device {
                model:nmos
                side:1
                ctrl:word
                use:pass
```

```
            }
            device {
                model:nmos
                side:0
                ctrl:word
                use:pass
            }
        }
    }
    port {
        read {
            device {
                model:nmos
                side:1
                ctrl:core
                use:pulldown
            }
            device {
                model:nmos
                side:1
                ctrl:word
                use:pass
            }
            device {
                model:nmos
                side:0
                ctrl:core
                use:pulldown
            }
            device {
                model:nmos
                side:0
                ctrl:word
                use:pass
            }}}}
```

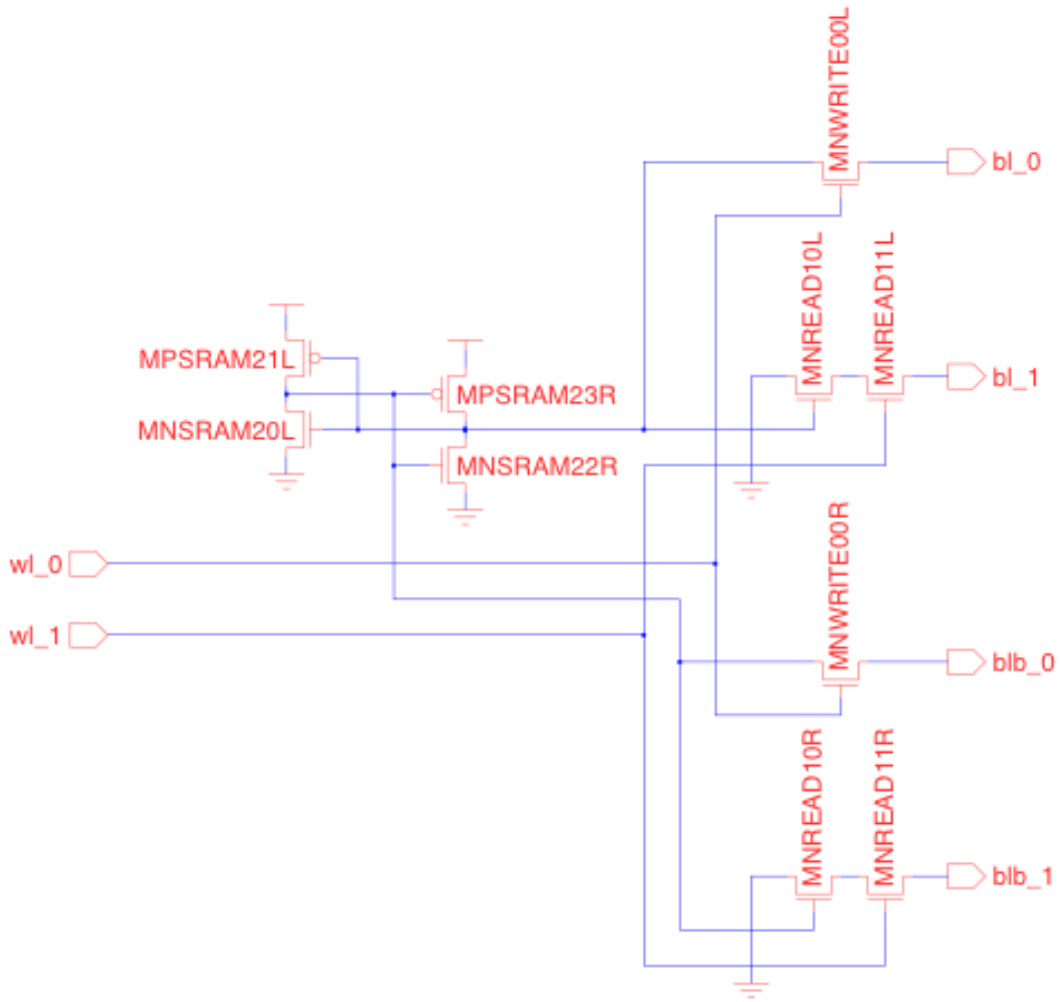


Figure 3 – 10T core cell

C

Performing Compound Calculations in Liberate MX

The `-measure` option of the define_measure command computes the dependent measurements, which can then be referenced in the define_arc statements. The arcs from `define_arc -measure` is further used to calculate new arcs in `define_arc -equation`. However, using this method only one level of computation is possible. To perform compound calculations in Liberate MX, you can create a Python file containing user-defined equations. This allows multiple levels of calculations from measurement arcs created using `define_arc -measure`.

Running Compound Calculations in Liberate MX

The following steps are involved in running compound calculations in Liberate MX:

1. Defining the source measurement arcs for the equation and the target arcs from calculation with unique names in `template.tcl`.
2. Creating a placeholder target arc with a unique name where the calculation results will be stored.
3. Extracting all the source measurement arcs by their names to run the calculation equation in the user-provided Python file.
4. After all the calculations are done, setting the target arcs with their names and value results.

Example

Let us consider an example, where a mpw arc is to be calculated from the following equation using measurement arcs:

```
rise_constraint = max((setup_1_equ + setup_2_equ)/2, (setup_1_equ - setup_2_equ)/
2)
```

Here, `setup_1_equ` and `setup_2_equ` are `altos_measurement` arcs, and `rise_constraint` is the target mpw arcs.

Perform the following steps to run compound calculation for the above-mentioned equation:

1. Create measurement in Liberate MX in `template.tcl`:

```
define_measure -name setup_1 \
    -trig {add_in[0]} -trig_val 0.5 -trig_dir rise \
    -equations "setup_data-setup_clock" -duration 4.0\
    -table_line 17b \
    rf_top
define_measure -name setup_2 \
    -trig {add_in[0]} -trig_val 0.5 -trig_dir rise \
    -equations "setup_data-setup_clock" -duration 4.0\
    -table_line vec0 \
    rf_top
```

2. Create measurement arcs with the previous measurements `setup_1` and `setup_2`, and name them as `setup_1_equ` and `setup_2_equ` in `template.tcl`.

```
define_arc -name setup_1_equ -type setup -equation_only \
    -pin {add_in[0]} -related_pin {clk_in} -pin_dir R -related_pin_dir R \
    -measure setup_1 \
    rf_top
define_arc -name setup_2_equ -type setup -equation_only\
    -pin {add_in[0]} -related_pin {clk_in} -pin_dir R -related_pin_dir R \
    -measure setup_2 \
    rf_top
```

3. Create target arcs placeholder with a unique name `target_mpwl_arc`, and designate a random value with `-value` (`1.0` in used in this example) in `template.tcl`.

```
define_arc -type mpw \
    -related_pin {clk_in} -pin {clk_in} -pin_dir F \
    -value {1.0} \
    -name {target_mpwl_arc} \
    rf_top
```

   The value will be replaced with calculated results.

   Note: It is recommended to use terms such as *`target`* to name the arc that you want to calculate so that irrelevant arcs are not modified by mistake.

4. Create a Python file in the run directory and name it `measure_equation.py`.

5. Write calculation equation in `measure_equation.py` using the source arc names specified in Step 2.

   Write the target arc setting statement with the target name and result values. Also include any Python package that is imported during the calculation. In this example, the Python file will contain the following:

```
# import necessary python package
Import numpy as np
# calculation equation with source arc name
setup_1_A = (setup_1_equ + setup_2_equ)/2
setup_1_B = (setup_1_equ - setup_2_equ)/2
target_mpwl_arc = np.maximum(setup_1_A, setup_1_B)
# set the target arc value with target arc name and calculation results
measObj.setMeasureTarget("target_mpwl_arc", target_mpwl_arc)
```

Note: If `measure_equation.py` is located in the run directory, it will be picked and run automatically. When the calculation is not required anymore, remove the Python file from run directory or rename the file.

D

Using Liberate MX Automatic Flow

This appendix discusses the commands and their options that enable you to run the Liberate MX automatic flow (also known as the define_memory flow). For information about the criteria that an instance should satisfy to use the automatic flow, refer to the Automatic Characterization Flow section in Chapter 4, "Liberate MX Flows."

This appendix also covers information about Debugging Common Issues with Automatic Flow.

Defining Liberate MX Settings for Automatic Flow

The Liberate MX automatic flow requires the following files:

- Tcl file containing the `define_memory` and char_memory commands
- Instance netlist
- SPICE or Spectre model files
- Reference library file or template file (*optional*)
- Tcl file containing any additional needed settings (*optional*)
- Additional table files (*optional*)

Primary Tcl File

The primary Tcl file of the automatic flow should contain the `define_memory` and `char_memory` commands. The sections below explain the various options of these commands that you can use to provide the Liberate MX settings and run characterization.

define_memory

The define_memory command is used to fully describe the run to be performed. The main information covered includes netlist and models location and format, PVT, pin name,

functionality and active level, simulator control options and threading, and high-level settings to control the algorithm (activate the graybox mode, enable the rcdb flow, and so on).

Arc Definition

The arc definitions can be passed in the form of the reference library using the `-ref_lib` option. In this case, Liberate MX characterizes the arcs exactly as defined in the reference library file.

If it is necessary to modify the arc definitions, it is best to use the `read_library` and `write_template` commands to create the template and modify it (take care to generate the template with Liberate MX). The template can then be included using the `-template` option.

If there is no template or reference lib available, Liberate MX will create arcs and `when` conditions based on the pins described in `define_memory`.

`-ref_lib`	Reference Liberty file, containing all arcs and conditions to be used during the characterization.
`-template`	Template file, written in Tcl, containing all arcs, cell description and measurement conditions.

In case of re-characterization, you might want to modify the lists of slews and characterize loads without editing the template file. This is enabled using the following options:

`-slews`	List of slews to be characterized. Overrides template or reference lib definition.
`-loads`	List of loads to be characterized. Overrides template of reference lib definition.
`-qloads`	List of loads to be characterized for the data out pin. Overrides template, reference lib or `-loads` definition.

Netlist and Models

Once arcs and look up tables are defined, you need to point to instance's netlist and models to be used for the different measurements, using the netlist and model arguments below:

`-netlist`	The instance netlist file.
`-netlist_format`	The format of the netlist file. Default is `spice`.
`-models`	The model include file.

-models_leakage	Specific models to be used for leakage simulations.
-models_power	Specific models to be used for dynamic power simulations.
-model_format	The format of the models. Default is spice.
-skip_read_model	Controls whether the models will be read in with read_spice.

The -foundry argument can be used to automatically set the define_leafcell commands. The available foundries are TSMC and SMIC. All other foundries will need to define_leafcell in the mx_settings file (described later in this section).

Corner Definition

The temperature will always need to be defined using the -temp option.

The instance voltage supplies can be defined as follows:

-rail	A paired value list of voltage supplies and values.
-global_voltage	Defines the nominal supply value when the supply names do not need to be defined.
-virtual_rails	The names of the virtual rails of the instance followed by their normal operation value. Supplements the virtual rails that are automatically located.

Port and Physical Implementation of an Instance

To help the tool to properly recognize memory point and efficiently partition the design, the following options can be used to describe the physical implementation of the instance:

-words	Describes the number of address locations in the memory. The default of this will be 2^(number of address bits). It is essential to define the number of words if the number of words is less than 2^(number of address bits) to ensure that the vectors will not try to access an address that does not exist.
-bits	The number of bits in the instance. This is usually not needed if the reference lib or template is provided.
-column_mux	The number of physical columns in the array corresponding to a single data bit.
-banks	The number of separate banks in the instance.

-number_of_ports	The number of ports in the instance.
-bitcell	The type of bitcell used in the instance. Default is a 6t single port cell. Available values are rom, single_port, dual_port, 2prf and 10t.

In the automated standard instance flow, the instance pins need to be defined with their functions. These functions will then be used to create the vectors to be used for the characterization. The following options are used to define the basename of their respective pins:

-clock	-address	-data_in	-data_out
-scan_in	-scan_out	-read_timer	-write_timer

Some other pins will need to be defined along with their active state:

-chip_enable	The base name of the chip enable pin along with the state for chip is enabled.
-write_enable	The base name of the write enable pin along with the state for chip is in write mode.
-bit_mask	The base name of the bit mask or bit write bus along with the state for bit is in masked mode.
-output_enable	The base name of the output enable pin along with the state for output is enabled.
-sleep	The base name of the sleep pin and its state for the instance is in sleep mode.
-shutdown	The base name of the shutdown pin and the state for the memory is in shutdown mode.
-bypass_enable	The base name of the bypass mode enable pin and the state for the instance is in bypass mode.
-read_timer_enable	The base name of the pin to select between and externally specified read timer setting and a default internal value.
-test_enable	The signal that would control the input MUXes and select between regular mode inputs and test mode inputs.

There are other options that will control the prefix or suffix to be appended to the base signal names to determine the final names. This is particularly useful in the case of dual port devices

or BIST control. The base name of the input signals is used with the defined prefixes and suffixes to map the pins from the subcircuit in the netlist file to their function.

`-clock_bist_prefix`	Prefix applied to the test version of the clock.
`-clock_bist_suffix`	Suffix applied to the test version of the clock.
`-clock_port_suffix`	Suffix applied to the clock signals to denote port.
`-bist_prefix`	Prefix applied to the test version of input pins.
`-bist_suffix`	Suffix applied to the test version of input pins.
`-port_suffix`	Suffix applied to the input signals to denote port.
`-bypass_suffix`	Suffix applied to a base input name to get the corresponding bypass mode output name.

You might also need to describe other pins used to control the device. In that case, the `-other_input_pins` option allows you to add them. Related non-standard arcs will have to be described in the template files or reference library, and you might have to provide additional tables (see later) to have arcs fully exercised during the fastsim simulations.

`-other_input_pins`	List of input pins of other functions.

Skipped or Additional Tables

You have the ability to override the tables that were created by the Liberate MX define_memory command. This allows the `define_memory` command to be used for instances that do not fit the exact definition of a standard instance.

`-additional_tables`	A list of tables to be added to the characterization.
`-remove_tables`	A list of tables created by define_memory that are not to be used for the characterization. The allowed values are: `leakage`, `delay`, `constraint`, `power`, `measure`, `bist`, `sleep`, and `bist_pwr`.

Simulator Options

The memory is now properly defined. Next step is to set simulator options and threading. Options below allow you to control speed of the characterization simulations:

`-part_spice`	The simulator to be used during the full instance portion of the Liberate MX run. The default is Spectre XPS.
`-char_spice`	The simulator to be used for the partition characterization portion of the Liberate MX run. The default is Spectre APS.
`-xps_smode_power`	Disables the use of s-mode for dynamic power simulations when using Spectre XPS. The default is 0, use s-mode for power.
`-fastsim_cmd_option`	Options to be used for the full instance simulation.
`-extsim_cmd_option`	Options to be used for the partition characterization simulations.
`-thread`	Number of threads to be used for the Liberate MX run. Default is 0, use all available.
`-part_thread`	Number of parallel jobs to be run during the full instance simulation portion. Overrides `-thread`.
`-char_thread`	Number of parallel jobs to be run during the partition characterization portion. Overrides `-thread`.
`-part_waveform_format`	
	Specifies the waveform format to be used for the full instance simulations. Default is `fsdb` format. This option can also be set to `sst2`.

Probing, Method Control and Additional Setup

Depending on your memory implementation, you might need to control where constraint probing will be inserted, based on the intersection between a pin and related pin. Probing and internal thresholds can also be controlled with the following options:

`-internal_threshold`	Lower and upper thresholds to be used for internal probing, in percent. Default is 20 80.
`-latch_probing`	Controls the nodes within a sequential element to be considered for probing. Default is input, state, internal, output.
`-autoprobing_hold_level`	

	List of pins and values for autoprobing hold level. Default is level 0.
-autoprobing_setup_level	
	List of pins and values for autoprobing setup value. Default is level 1.

The Liberate MX settings file contains the additional Liberate MX commands and options to be used for the characterization. This is used to override the internal Liberate MX settings that are defined, and instead define specific options, like define_leafcell commands or debug options.

-mx_setting	Name of the user-provided Tcl file containing the Liberate MX settings and commands.

The last step is to define master characterization methods, and specify the name of your memory, with the following:

-greybox	Enables the greybox (non-partitioning) flow. Default is 0.
-bisection	Enables bisection characterization for constraints. Default is 0.
-rcdb	Enables the RCDB flow. In this flow, the Liberate MX database size is reduced by leveraging information in the dspf netlist. This should only be used for large instances in dspf format. Default is 0.
user_memory_name	Specify the name of your memory.

char_memory

Once the define_memory command is completed, launch the characterization using the char_memory command. This command accepts few options that control where the run is to be performed and which format has to be generated.

The options of the char_memory command are:

-workdir	Specify the work directory name. It will be created as a subdirectory of the current directory if it does not already exist.
-ecsm	Enables ECSM characterization and lib generation.
-ecsmn	Enables ECSMN characterization and lib generation.

-ccs	Enables CCS characterization and library generation.
-ccsn	Enables CCSN characterization and library generation.

Additional Table Files

It might be necessary to provide additional tables to the Liberate MX define_memory flow. This could be needed if the tables created by Liberate MX are either incorrect or not sufficient to complete the characterization.

To learn more about tables, see the Guidelines for Writing Truth Table for Multiport Memories section.

Example

As an example, look at the following instance with bist inputs:

Pin Name	Function	Pin Name	Function
CLOCK	Clock pin for all modes	BIST	Bist control selects between input and test input - active high
CE	Chip Enable - active high	TCE	Test Chip Enable
WE	Write Enable - active high	TWE	Test Write Enable
A[6:0]	Address	TA[6:0]	Test Address
D[31:0]	Data	TD[31:0]	Test Data In
Q[31:0]	Data Out		

This would be the corresponding define_memory command:

```
define_memory \
    -netlist ./Netlist/rf_top_bist.cir \
    -models ./Models/include_models.scs \
    -model_format spectre \
    -global_voltage 1 \
    -temp 25 \
    -template ./Templates/template.tcl \
    -mx_setting ./Tcl/mx_settings.tcl \
    -bits 32 \
    -words 128 \
```

```
    -clock {CLOCK} \
    -address {A} \
    -data_in {D} \
    -data_out {Q} \
    -write_enable {WE H} \
    -chip_enable {CE H} \
    -test_enable {BIST H} \
    -bist_prefix {T} \
    -rail {  VDD 1.0 \
             VSS 0.0 } \
    rf_top_bist

char_memory
```

Defining the `bist` prefix as `T` instructs Liberate MX to prepend a `T` to the input pin names to derive the test input name. For example, a `T` is prepended to the chip enable name, `"CE"` to get the test chip enable name of `"TCE"`.

Debugging Common Issues with Automatic Flow

The Liberate MX automatic flow is intended as a fully automated solution for characterization of standard and vendor sourced memory instances. There are some cases, such as incorrect setup or a non-conforming instance, that can require some user debugging to understand what went wrong.

Pin Name Recognition Issues

The most common sources of debug in the `define_memory` flow are pin name recognition issues. These can result in incorrect stimulus vectors and missing arcs.

You are provided with a summary of the final pin mapping as `mx_setup/pin_file`:

```
pin_name            function            base_name
A                   address             A
BIST                bist                BIST
CE                  chip_enable         CE
CLOCK               clock               CLOCK
D                   data_in             D
Q                   data_out            Q
TA                  address             A
TCE                 chip_enable         CE
```

```
TD                  data_in             D
TWE                 write_enable        WE
WE                  write_enable        WE
```

If there is a mistake in the pin names defined, there will be question marks inserted into the `pin_file` report:

```
pin_name            function            base_name
A                   address             A
BIST                bist                BIST
CE                  chip_enable         CE
CLOCK               clock               CLOCK
D                   ?                   ?
Q                   data_out            Q
TA                  address             A
TCE                 chip_enable         CE
TD                  ?                   ?
TWE                 write_enable        WE
WE                  write_enable        WE
```

The report has all of the names from the instance netlist subcircuit in the first column. In this example, the user should investigate the name defined for the pins `D` and `TD`.

Missing Arcs

After running the `define_memory` flow, the user may find that some arcs are missing from the lib file. Some possible sources of these missing arcs include:

- Missing arc definitions
- Simulation issues
- Incorrect vectors

As in all Liberate MX flows, in order to characterize and arc, it is necessary to have an arc definition as well as a proper stimulus. If an arc is missing, the user should verify that both the stimulus and the definition are present. This can be corrected by the user by using the `-template` flow and modifying the template file as needed to contain the missing arc. Missing arc definitions in the template file can affect the stimulus created by the `define_memory` flow.

You should delete all reuse results if the template is modified.

If the arc is properly defined, but the arc is still missing from the lib file, the user should review the waveforms from the top level simulations to ensure that the activity is correct. If it is found

to be incorrect, the user should review the `define_memory` command used to make sure that all side pins are properly defined.

If the stimulus cannot be corrected by modifying the `define_memory` command. You might need to create a new table file and include it using the `-additional_tables` command.

E

Liberate MX Timing Validation Flow

The Liberate MX timing validation flow performs validation of the timing arcs of the memory instance by running at speed simulation. The memory is simulated with user provided vectors with the minimum required setup, hold, and clock period found in the lib file to ensure that functionality is maintained.

The output of the MX timing validation flow is the two simulation waveform files. One simulation uses a deck with relaxed timing without any stressed waveforms. The second simulation uses a deck with stressed timing with realignments of signal waveforms against clocks to reflect the minimum timing that is provided. You can check the waveform differences between these two simulations to determine if the functionality and marginality of the instance is preserved under the stress conditions. For example, if the customer provides a table called `validation.tbl`, then two tables, `validation_relax.tbl` and `validation_stress.tbl` are generated. These two tables (vectors) are run and the results are stored.

The procedure for this is as given below.

1. Prepare a reference template file called `ref_template.tcl`:

   Input: A library with setup or hold time constraints of signals relative to clock.

   Output: `ref_template.tcl` file with `define_arc -validation_values {<values>}`.

   Command flow:

   ```
   read_library to_be_verified.lib
   write_template -mx_validate ref_template.tcl
   ```

2. Prepare a Liberate MX control file called `mxv.tcl` using the `ref_template.tcl` from step 1 above, as shown below:

   ```
   source ref_template.tcl
   validate_macro -validsim "xps" -thread 2
   ```

3. Check the results.

The output from `validate_macro` will have waveforms. The waveform data is stored in two directories under `mx_reuse/mx_fastsim/`. If a table called `validation.tbl` was provided, two tables `validation_relax.tbl` and `validation_stree.tbl` are generated. The two simulation results are stored in:

```
mx_reuse/mx_fastsim/SRAM512x8_validation_relax_0_/transient1.tran.trn
mx_reuse/mx_fastsim/SRAM512x8_validation_stress_1_/transient1.tran.trn
```

4. About the simulation points:

The default data point chosen for fastsim simulation will be the minimum number of the delay/constraint indexes. If users want to change the simulation slew/load, they can use:

```
set_var mx_fastsim_input_slew  <input_slew_wanted>
set_var mx_fastsim_clock_slew  <clock_slew_wanted>
set_var mx_fastsim_load        <load_wanted>
```

F

Basic Flow for Validating User-Defined Criteria

To verify user-defined criteria by using the Liberate MX validation flow, perform the following steps:

1. define_measure <options>

   In this step, specify a measurement based on the requirement.

2. validate_measure <options>

   In this step, provide measure name to be validated from `define_measure`.

3. validate_macro <options>

   In this step, Liberate MX runs the validate feature and simulates the design.

4. report_measure <options>

   In this step, report the differences between relax and stress simulation results.

Example:

The example provided below demonstrates the steps that have been explained above. Here, the `define_measure` command is specified to provide the measurement details. Once it is specified, the measurement is done for measure named, `valid_meas`. Next, `validate_measure` is used to specify `valid_meas` to confirm the measurement that is of interest. Then, the `validate_macro` command runs the simulation considering the vectors provided by the user in the table (`.tbl`) file. In the final step, the `report_measure` command is specified to report of error(s), if any.

```
----------------
set cell sram
set voltage 1.2
set tskew 30e-12
set vc50 [expr $voltage * 0.5]
```

```
set vs50 [expr $voltage * 0.5]

define_measure \
-name _valid_meas_A \
-trig clk  -trig_dir rise -trig_val $vs50 \
-targ { dout[0] } -targ_dir fall -targ_val $vs50 \
-failed_val "0" \
$cell

define_measure \
-name _valid_meas_B \
-trig clk -trig_dir rise -trig_val $vs50 \
-targ { dout[0] } -targ_dir rise -targ_val $vs50 \
-failed_val "0" \
$cell

define_measure \
-name valid_meas \
-trig clk -trig_dir rise -trig_val $vs50 \
-equations {"max(_valid_meas_A,_valid_meas_B)"} \
-keep min -duration 1.0 \
$cell

define_arc -measure valid_meas -pin clk -pin_dir R $cell

validate_measure "valid_meas" $cell
validate_macro -thread 1 -validsim $fastspice
report_measure -all "valid_meas" $cell
```

G

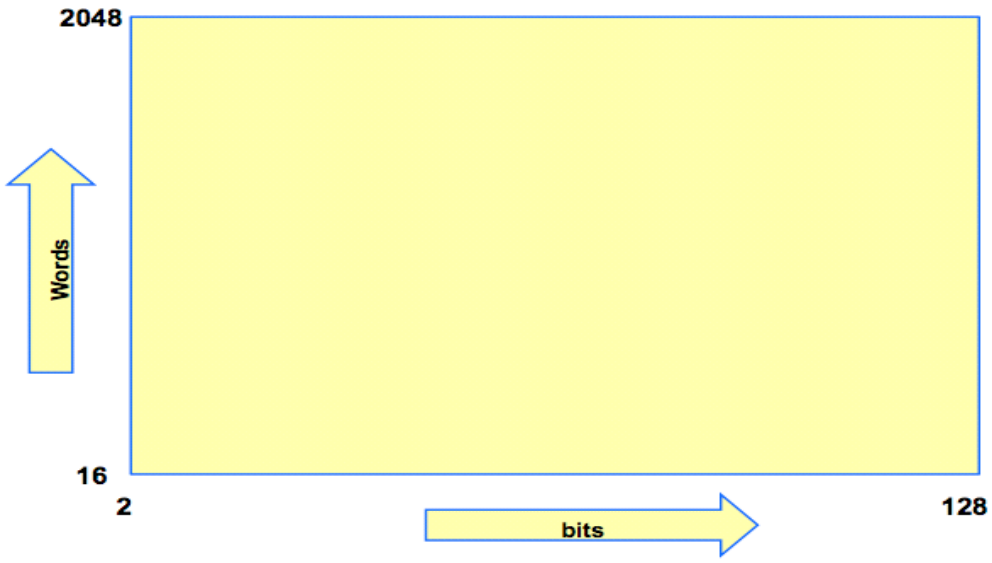
Liberate MX Compiler Characterization Flow

Liberate MX supports characterization and creation of memory compilers. The memory compilers use limited simulation data to predict timing and power for a wide range of instances.

This appendix introduces you to the Liberate MX compiler characterization flow along with its commands and supported flows. it also instructs the user on how to setup the Liberate MX compiler characterization flow, how to populate the characterization data, and how to generate estimated data for the interpolated instances.

Introduction to Memory Compilers

Memory instances can be designed in a modular way that allows the creation of arbitrary-sized instances. These instance sizes fall within a particular physical range. For example, the same building blocks might be used to build an instance with words ranging from 16 to 2048 and with bits ranging from 2 to 128. It would not be practical to always generate an instance, extract, and characterize to determine timing and power for the instance. For this reason, there needs to be a way to create estimated timing data quickly for a generated instance.



This can usually be done by characterizing a small number of instances within the instance range. The timing and power of those instances can then be used to predict the timing and power of other, non-characterized instances. Instances will be selected to cover boundary conditions and to provide enough data to accurately predict the remaining instances.

In this example, the minimum list of instances should be the boundary conditions.

- 2048 words, 2 bits
- 2048 words, 128 bits
- 16 words, 2 bits
- 16 words, 128 bits

Other instances can also be characterized to improve interpolation accuracy.

There are many different formulas that can be used to predict timing and power for non-characterized instances. These include:

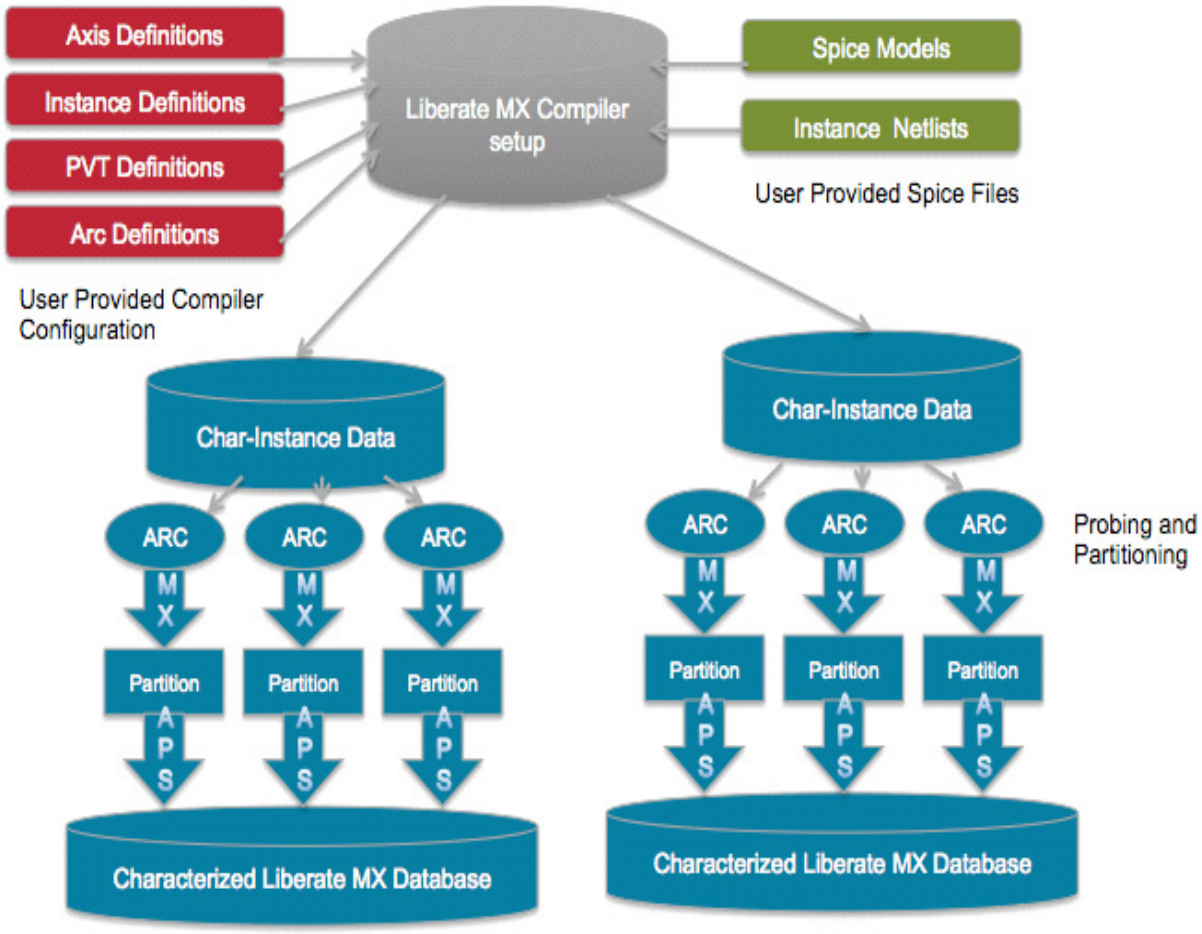
- Linear interpolation
- Curve fitting
- Formula-based

Memory Compiler Characterization with Liberate MX

Liberate MX has the capability to perform compiler characterization. This includes the characterization of the specified instances, assembling of the needed data, and the generation of estimated data for other instances. The compiler characterization or interpolation can be performed incrementally with reuse of previously run characterization.

You will need to define the following:

- Instance space
- Axis names and values
- PVT conditions for characterization
  - SPICE Models
- Arcs to be characterized
- Instances to be characterized
  - Netlists
- Instances to be interpolated



There are Liberate MX commands to define the parameter axis that can vary by instance, the design space where data is expected to be continuous, the characterized instances, and the interpolated instances.

See also:

- Defining the Compiler Space
- Defining the PVT Corner for Characterization
- Defining the Characterized Instances
- Defining the Interpolated Instances

Defining the Compiler Space

To define the compiler space, you need to define the axes and their ranges using the compiler_define_axis command. In addition, you need to define a design using the compiler_define_design command.

compiler_define_axis

This command defines the axis for each instance. Every defined instance should represent a unique combination of values of the axis defined corresponding to it. It is required to either specify the allowed values or the increment. The defined name is by user preference and has no internal definition by Liberate MX.

Options

`-min_value`	The minimum value for the axis. Default: `0`
`-max_value`	The maximum value for the axis. Default: `1`
`-increment`	The increment of allowed values. Default: `1`
`-allowed_values`	The list of allowed values. Default: `min`, `max`, and `all` intermediate values defined by increment
`<name>`	The axis name to be used in instance definitions.

For example:

```
compiler_define_axis \
    -min_value 2 \
    -max_value 128 \
    -increment 1 \
    bit

compiler_define_axis \
    -min_value 16 \
    -max_value 2048 \
    -increment 4 \
    word
```

compiler_define_design

In a design, the characterization data should be continuous. This means that if there is a boundary point where there are circuit differences, the defined design region should not cross that boundary. Also, only one design should contain a defined instance.

Options

`-axis`	Paired list of axis name followed by `min:max`. Unspecified axis uses the ranges defined in `define_axis`.
`-mx_settings`	Additional file containing Liberate MX settings.
`-bitcell`	The type of bitcell. One of the following value can be specified: `rom`, `single_port`, `dual_port`, `2prf`, or `10t`. Default: `single_port`
`-bist_prefix "string"`	Signal prefix for test mode. This specifies the prefix to be added to the base signal names to specify the signal names to be used for the test mode operation. This affects any defined address (`data_in`, `data_out`, `chip_enable`, `write_enable`, `bit_write`, or `test_enable`). (Standard Custom Instance flow)
`-bist_suffix "string"`	Signal suffix for test mode. This specifies the suffix to be added to the base signal names to specify the signal names to be used for the test mode operation. This affects any defined address (`data_in`, `data_out`, `chip_enable`, `write_enable`, `bit_write`, or `test_enable`). (Standard Custom Instance flow)
`-clock`	Name of the clock pin.
`-address`	Name of the address pin.
`-data_in`	Name of the data-in pin.
`-data_out`	Name of the data-out pin.
`-chip_enable`	Name of the chip-enable pin followed by its state for chip set as active.
`-write_enable`	Name of the write-enable pin followed by its state for chip set for writing.
`-read_enable`	Name of the read-enable pin followed by its state for chip set for reading.
`-bit_mask`	Name of the bit-mask/bit-write pin followed by its state for bit set as masked.
`<name>`	Name of the design block.

The defined instance pins are used for the purpose of automatically creating table vectors for the instance. This is done using the same methods as the `define_memory` command. All instances within the design must have the same pinout, although bus sizes can differ.

For example:

```
compiler_define_design \
    -axis { {bit 2:128} {word 16:2048} } \
    -clock {clk_in} \
    -address {add_in} \
    -data_in {data_in} \
    -data_out {data_out} \
    -write_enable {wr_in H} \
    -chip_enable {chip_en H} \
    -mx_setting [pwd]/mx_settings.tcl \
```

In addition to using the `define_memory` flow, the compiler characterization flow can be run using custom top-level Tcl scripts. To specify the custom top-level Tcl command file, you need to specify a directory that contains the top-level command files for instances of each PVT. This can be done using the `-cmd_file_path` option of the compiler_define_pvt command, as well as the custom top-level Tcl command file using the `-cmd_file` option of the compiler_define_instance command.

Defining the PVT Corner for Characterization

The PVT corners to be used for the compiler characterization must be defined using the compiler_define_pvt command. Each PVT corner requires a separate run of Liberate MX.

compiler_define_pvt

Defines the PVT corners.

Options

`-cmd_file_path`	Specifies a directory that contains the top-level command files for instances of each PVT. For example:
	`-cmd_file_path $cmd_dir\`
	`$process`

`-rail`	Name and value pairs for each voltage supply.
`-virtual_rails`	Name and value pairs for each virtual rail.

`-models`	Name of the SPICE model include file.
`-model_format`	Model format to be used. One of the following values can be specified: `spice` or `spectre`. Default: `spice`
`-temp`	Temperature in degree Celsius.
`-ref_lib_path`	Directory for the reference `.lib` files of the PVT corner.
`-template_path`	Directory for the template files of the PVT corner.
`-netlist_path`	Directory for the instance netlists of the PVT corner. **Note:** A different netlist path allows you to use different extracted netlists for different PVT corners.
`-user_data_path`	Directory for the user data files of the PVT corner.
`<name>`	Name of the PVT corner.

For example:

```
compiler_define_pvt \
    -rail {VDD 1.0 VSS 0.0} \
    -models [pwd]/include_models \
    -model_format spectre \
    -temp 25 \
    -netlist_path [pwd]/netlists \
    -template_path [pwd]/templates \
    Typical
```

Defining the Characterized Instances

The characterized instances need to be defined with the compiler_define_instance command. Of the defined instances, only those specified in compiler_char_instances are characterized.

compiler_define_instance

Defines the characterized instances.

Options

`-cmd_file`	Specifies the custom top-level Tcl command file. For example:
	`-cmd_file mx.tcl`
	`$process`

`-axis`	Paired list of axis name followed by value. All axes defined with `define_axis` must be assigned a value.
`-design`	Name of the compiler design as defined with `compiler_define_design`.
`-netlist`	Name of the instance netlist. This can be found in the directory specified by `compiler_define_pvt -netlist_path`.
`-netlist_format`	Netlist format. One of the following values are allowed: `spice` or `spectre`. Default: `spice`
`-ref_lib`	Name of the reference `.lib` file. This can be found in the directory specified by `compiler_define_pvt -ref_lib_path`.
`-template`	Name of the template file. This can be found in the directory specified by `compiler_define_pvt -template_path`.
`-user_data`	Name of the user data file. This can be found in the directory specified by `compiler_define_pvt -user_data_path`.
`-additional_tables`	List of additional tables to be used for the characterization.
`-pin_groups [list []]`	List of bus lists split into multiple groups based on the bits so that interpolation can be done on the groups independently.
`-remove_tables`	List of generated tables not to be used for the characterization. One of the following values can be specified: `leakage`, `delay`, `constraint`, `power`, `measure`, `bist`, `sleep`, or `bist_pwr`.
`{<instance_name>}`	Name of the instance. (Required)

For example:

```
compiler_define_instance \
    -design RF \
    -axis { \
                {word 16} \
                {bit  2} \
```

```
        } \
-netlist rf_16x2.cir \
-template rf_16x2_tmpl.tcl \
rf_16x2
```

compiler_char_instances

Specifies the defined instances and PVTs to be used for the characterization. This command can also be used to specify the simulators and related options.

The following global Tcl variables can be used in the custom Tcl command file:

- `CELL` (name of the characterization instance from `compiler_define_instance` command).
- `PVT` (name of PVT from `compiler_define_pvt` command).
- `DESIGN` (name of design from `compiler_define_design` command).

Options

`-part_spice`	Simulator to be used for the full instance partitioning simulations. One of the following values can be specified: `ultrasim`, `aps`, or `xps`. Default: `xps`
`-char_spice`	Simulator to be used for the partition characterization phase. One of the following values can be specified: `spectre` or `aps`. Default: `aps`
`-part_thread`	Number of threads to be used for the full instance partition creation simulations. Default: `0` (use all available threads)
`-char_thread`	Number of threads to be used for the partition characterization phase. Default: `0` (use all available threads)
`-inst_thread`	Number of instance characterizations to be done simultaneously. Default: `2`
`-extsim_cmd_option`	Command options for the partition characterization phase.
`-fastsim_cmd_option`	Command options for the full instance partition creation phase.
`-pre_lib_script`	Tcl file to be sourced before writing the library.
`-ccs`	Enable CCS characterization. Default: `false`
`-ccsn`	Enable CCSN characterization. Default: `false`

`-ecsm`	Enable ECSM characterization. Default: `false`
`-ecsmn`	Enable ECSMN characterization. Default: `false`
`-pvts`	List of previously defined PVTs to be used for the characterization.
`-instances`	List of previously defined instances to be characterized.
`-workdir`	Working directory for the characterization. Default is the current directory.
`-write_library_args`	Provides additional arguments to be used in the `write_library` phase of characterization.

For example:

```
compiler_char_instances \
    -design RF \
    -char_spice aps \
    -char_thread 8 \
    -part_spice xps \
    -part_thread 8 \
    -instances { \
                     rf_16x2 \
                     rf_16x128 \
                     rf_2048x2 \
                     rf_2048x128 \
                } \
    -workdir [pwd]/comp_char \
    -pvts Typical
```

Defining the Interpolated Instances

The interpolated instances need to be defined with the command interpolate_define_instance. Of the defined instances, only those specified in interpolate_generate_instances are created. In addition, interpolation can be done for each instance using the set of closest characterized instances for interpolation instead of using the set of all characterized instances. This can be done by specifying the `-closests_char_insts` option of the `interpolate_generate_instances` command.

interpolate_define_instance

Defines the interpolated instances.

Options

`-axis`	Paired list of axis name followed by value. All axes defined with `define_axis` must be assigned a value.
`-design`	Name of the compiler design as defined in `compiler_define_design`.
`-extrapolate`	Extrapolates the instances out of the axis range specified by `compiler_define_axis`.
`-pin_groups [list []]`	
	List of bus lists split into multiple groups based on the bits so that interpolation can be done on the groups independently.
`-template`	Name of the template file. This can be found in the directory specified by `compiler_define_pvt -template_path`.
`-user_data`	Name of the user data file. This can be found in the directory specified by `compiler_define_pvt -user_data_path`.
`{<instance_name>}`	Name of the instance. (Required)

For example:

```
interpolate_define_instance \
    -design RF \
    -axis { \
                {word 64} \
                {bit 16} \
            } \
    -template rf_64x16_tmpl.tcl \
    rf_64x16
```

Example for extrapolation:

```
interpolate_define_instance \
    -extrapolate \
    -design RF \
    -axis { \
              {word 32} \
              {bit 36} \
```

```
        } \
    -template rf_32x36_tmpl.tcl \
rf_32x36
```

interpolate_generate_instances

Specifies the defined instances and PVTs that will be generated.

Options

`-pre_lib_script`	Tcl file to be sourced before writing the library.
`-ccs`	Enable CCS characterization. Default: `false`
`-ccsn`	Enable CCSN characterization. Default: `false`
`-closest_char_insts`	When this option is specified, for each interpolated instance a set of `2^n` (here, `n` is the number of design axis.) characterized instances that "surround" the interpolated instance is used for interpolation. For example when using design axis "bit" and "word", for each interpolated instance, a set of four characterized instances that form a box around the interpolated instance is used.
`-compare_char_insts`	Compares the characterized instances with the interpolated instances and lets you debug the interpolation error. **Note:** All the instances to be characterized should be included as interpolation instances when specifying the `-instances` option in the `interpolate_generate_instances` command.
`-delete_ldb`	Removes the LDB generated by `interpolate_generate_instances` command.
`-ecsm`	Enables ECSM characterization. Default: `false`
`-ecsmn`	Enables ECSMN characterization. Default: `false`
`-ecsm_monotonicity_checks`	

	Sets the absolute tolerance and relative tolerance for checking the monotonicity of ECSM vector. Example: `-ecsm_monotonicity_checks {{abstol {intp_monotonicity_check 0.005}} {reltol {intp_monotonicity_check 0.05}}}`
`-debug`	Enables the debug flag in to report interpolation option.
`-pvts`	List of previously defined PVTs to be used for the characterization.
`-intp_comments`	Adds the interpolation formula used in the generated LDB file and the log file.
`-instances`	List of instances to be characterized. **Note:** These instances must have been previously defined with the interpolate_define_instance command.
`-workdir`	Working directory for the characterization. Default is the current directory.
`-exit_on_error`	When this option is specified, Liberate MX exits if any errors are encountered during interpolation.
`-library_name_prefix`	Prefix for the customizing the library name and liberty file names. **Note:** Name of the memory cell is the default library name.
`-library_name_postfix`	Postfix for the customizing the library name and liberty file names.

For example:

```
interpolate_generate_instances \
    -design  RF \
    -instances { \
                    rf_64x16 \
                } \
    -workdir [pwd]/comp_char \
    -pvts Typical
```

The generated instances can be found in the following directory:

```
<workdir>/<design>/<pvt>/<instance>/LIBS
```

In this example, the path would be:

```
comp_char/RF/Typical/rf_64x16
```

interpolator_run_instances

Takes a list of all instances that are to be interpolated and an interpolation script that uses global variables for design, PVTs, and intp_insts. It then sets up and executes a number of parallel runs to generate the interpolated libraries.

Options

`-design`	Name of compiler design. [Required argument]
`-pvts`	List of PVTs to be modeled. [Required argument]
`-instances`	List of instances to be modeled. [Required argument]
`-intp_script`	Tcl file to run interpolation on a set of PVTs and a set of instances. [Required argument]
`-run_dir`	Top level directory to be used for launching interpolation runs
`-bundle`	The number of instances to be bundled in each interpolation run. Default: `0`
`-num_runs`	The number of interpolation runs to execute in parallel. Default: `2`

For example:

```
set intp_insts {inst1 inst2 .... }
set design ram
set pvts {SS}
interpolator_run_instances -instances $intp_insts -pvts $pvts -design $design -
intp_script my_intp_script.tcl -bundle 100
```

`my_intp_script.tcl` script uses global tcl variables for global *`INTERPOLATION_INSTANCES`*, *`INTERPOLATION_PVTS`*, and *`INTERPOLATION_DESIGN`* that will be used in each run.

Using Interpolation

To predict the data for uncharacterized instances, Liberate MX uses characterized instances and interpolation. The interpolation formula is a curve fit of the characterized instances based

on their axis values. The accuracy of the interpolated values depends on the quantity and proximity of the characterized instances.

The Interpolation Formula

Liberate MX uses a curve fit formula for interpolation that is based on the defined axis of the design. For a design with `N` axis defined, the formula would be:

```
value = C0 + C1 * Axis_1 + C2 * Axis_2 + … + Cn * Axis_n
```

Liberate MX solves the coefficient values for `C0` through `Cn` based on the characterized values. This equation will be solved independently for each arc. It is required that there are at least `n+1` characterized instances.

For the previous example with bits and words defined, the formula would be as following:

```
value = C0 + C1 * rows + C2 * bits
```

Additional complex elements can be introduced through the interpolate_define_axis_expr command.

interpolate_define_axis_expr

Specifies the additional complex elements for interpolation.

Options

`-expression`	Arithmetic expression in terms of previously defined axis.
`<name>`	Name of the axis.

The expression can be a value arrived at by multiplying the previously defined axis with the other defined axis or with itself.

For example:

```
interpolate_define_axis_expr \
    -expression  bits*words \
    bits_words

interpolate_define_axis_expr \
    -expression  bits*bits \
    bits_squared
```

The newly defined values are used in the interpolation formula. Following are the expressions for `n` axis and `m` defined axis:

```
value = C0 + C1 * Axis_1 + C2 * Axis_2 + … + Cn * Axis_n + Cn+1 * Axis_Expr_1 +
Cn+2 * Axis_Expr_2 + … + Cn+m * Axis_Expr_m
```

In this case as well, Liberate MX solves the coefficient values for `C0` through `Cn+m` based on the characterized values. This equation will be solved independently for each arc. It is required that there will be at least `n+m+1` characterized instances.

For the previous example, the formula would then become:

```
value = C0 + C1 * rows + C2 * bits + C3 * bits_words + C4 * bits_squared
```

Splitting Bus into Groups for Interpolation

In the Liberate MX compiler flow, bus can be split into multiple groups based on the bits to perform independent interpolation on the split groups. To split bus, specify bus groups using the `-pin_groups` option with the interpolate_define_instance and compiler_define_instance commands. You can provide inputs as a list of lists with this option.

Note: When using this feature, `write_library -expand_buses` must be set in order to see changes in the library.

For example:

```
interpolate_define_instance
     -pin_groups [ list []]

compiler_define_instance
     -pin_groups [ list []]
```

The final input to the `-pin_groups` option should be given as the following:

```
-pin_groups { {delay data_in[0:1] datalsb  data_in[2:7] datamid
data_in[8:10] datamsb} }    {the input should always be a list of
lists}
```

To make the task easier, you can define a function to automatically split the bits of the bus, group them, and produce the output in the above format. For example:

```
-pin_group [gen_pin_group {arc_type bus bus_size}]
```

Similarly, you can write your own function or modify this one as needed.

You can also provide input using a function in the following way:

```
-pin_groups [ list [gen_pin_group delay data_in 11] ]
```

To split two different buses in two different ways, such as {value, value2, value1} and {value1, value2, value3}), you can divide two functions and provide them as input in the following way:

```
-pin_groups [ list [gen_pin_group_1 delay data_in 11]
[gen_pin_group_2 setup add_in 8] ] ….etc
```

Note: When using this feature, ensure that the split bus defined in `interpolate_define_instance` and `compiler_define_instance` is split in the `define_arc` template in the same way.

Example Control File

The commands given in the sections above are used together in the Liberate MX Tcl file to run the characterization.

```
# Define the Compiler Axis Values

compiler_define_axis \
    -min_value 4 \
    -max_value 32 \
    -increment 1 \
    bit

compiler_define_axis \
    -min_value 16 \
    -max_value 128 \
    -increment 4 \
    word

# Define the design space and pinout

compiler_define_design \
    -axis { {bit 4:32} {word 16:128} } \
    -clock {clk_in} \
    -address {add_in} \
    -data_in {data_in} \
    -data_out {data_out} \
    -write_enable {wr_in H} \
    -chip_enable {chip_en H} \
    -mx_setting [pwd]/mx_settings.tcl \
```

```
    RF

# Define the PVT and its associated models, extraction and template paths

compiler_define_pvt \
    -rail {VDD 1.0 VSS 0.0} \
    -models [pwd]/include_models \
    -model_format spectre \
    -temp 25 \
    -netlist_path [pwd]/netlists \
    -template_path [pwd]/templates \
    Typical

# Define the Characterized Instances

compiler_define_instance \
    -design RF \
    -axis { \
        {word 16} \
        {bit 4} \
        } \
    -netlist rf_16x4.cir \
    -template rf_16x4_tmpl.tcl \
    rf_16x4

compiler_define_instance \
    -design RF \
    -axis { \
        {word 128} \
        {bit 4} \
        } \
    -netlist rf_128x4.cir \
    -template rf_128x4_tmpl.tcl \
```

rf_128x4

```
compiler_define_instance \
    -design RF \
    -axis { \
        {word 16} \
        {bit 32} \
```

```
        } \
    -netlist rf_16x32.cir \
    -template rf_16x32_tmpl.tcl \
    rf_16x32

compiler_define_instance \
    -design RF \
    -axis { \
        {word 128} \
        {bit 32} \
        } \
    -netlist rf_128x32.cir \
    -template rf_128x32_tmpl.tcl \
    rf_128x32

# Characterize the Instances

compiler_char_instances \
    -design RF \
    -char_spice aps \
    -char_thread 10 \
    -part_spice xps \
    -part_thread 4 \
    -instances { \
             rf_16x4 \
             rf_16x32 \
             rf_128x4 \
             rf_128x32 \
         } \
    -workdir [pwd]/comp_char \
    -pvts Typical

# Define the Interpolated Instances

interpolate_define_instance \
    -design RF \
    -axis { \
        {word 32} \
        {bit 32} \
        } \
    -template rf_32x32_tmpl.tcl \
```

```
    rf_32x32

interpolate_define_instance \
    -design RF \
    -axis { \
        {word 64} \
        {bit 16} \
        } \
    -template rf_64x16_tmpl.tcl \
    rf_64x16

# Generate the Interpolated Instances

interpolate_generate_instances \
    -design  RF \
    -instances { \
             rf_32x32 \
             rf_64x16 \
         } \
    -workdir [pwd]/comp_char \
    -pvts Typical
```

License Requirements

For licensing requirement related to Liberate MX compiler creation flow, refer to *LIBERATE Software Licensing and Configuration Guide*.

H

Legacy Commands and Parameters

This chapter lists all the commands and parameters that are either deprecated, or included for backward compatibility. We would like to discourage users from relying on these commands and parameters. Instead, find the alternate command or parameter along with its settings to achieve the best results from Liberate MX.

Backward Compatibility Parameter

si_write_output_voltage_compatibility_mode

`<0 | 1>`

Creates an `output_voltage` group in the `merge_library` flow when the input library has a template that includes `variable_1: iv_output_voltage`.
Default: `1`

- `0` — For backward compatibility. Creates an `output_voltage` group in the final merged library.
- `1` — Creates an `output_voltage` group in the final merged library. If the input library does not have the `output_voltage` group, the `output_voltage` group will not be generated for the final merged library and an error message will be printed.

This parameter must be set before the `merge_library` command.

Deprecated Commands

The following commands are being phased out, and have been replaced by either new commands (or related options), new parameters, or new behaviors of the tool.

debug_flow

`<1x1 | 2x2>`

`1x1`	A `1×1` data matrix uses the first value in each index.
`2x2`	A `2×2` data matrix uses the first and last values in each index.

Use this parameter to speed up the characterization by reducing the template as defined by the `define_template` command to either a `1x1` or a `2x2` data matrix. This can significantly decrease the runtime for a quick debug flow.

You can check selected points in the slew/load matrix by shrinking the template to a `2×2` or `1×1`.

Example:

```
set_var debug_flow 2x2
```

This parameter must be set before the first `define_template` command.

mx_mpw_mode

`clock_pulse`	Selects calculation method used with version 3.0p3 and earlier.

mx_set_clockprop

Use the mx_set_domainprop command instead.

mx_set_finesim_param

These commands are used to pass parameters to the appropriate external simulator.

Options.

`<list>`	Passes a list of finesim parameters as name-value pairs. The specified parameters are used during FastSPICE simulation.

Note: These parameters can also be specified in a table format using hspice_lis_2_waves. Parameters specified in a table will override parameters that are specified with a Tcl command. (See fastsim_deck.)

Example

```
mx_set_finesim_param { \
    {simpreset 5} \
    {pn_level 5} \
    {cgnd 1e-15} \
    {sfe_compaction 0} \
    {keepparaname 0} \
    {rshort 2} \
    {hier_delimiter .} \
    {dc_turbo 3} \
    {rcr_fmax 1G} \
}
```

mx_set_hsim_param

<list>	Passes a list of HSIM parameters that is used during FastSPICE simulation. Parameters are passed as a list of name-value pairs.

Parameters can also be passed by specifying them in a table and using the hspice_lis_2_waves command. Parameters specified in a table override parameters that are specified with a Tcl command.

mx_set_nanosim_param

<list>	Passes a list of NanoSim parameters that is used during FastSPICE simulation. 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.

Deprecated Parameters

mx_remove_rc

Use the mx_remove_rc_pincap and mx_remove_rc_timing parameters instead.

mx_whitebox_monitor_memcores

Use the mx_monitor_memcore parameter instead.

mx_active_load_thresh

`<value>`	Threshold for determining if the active load on a node can be replaced with a passive load. Default: `1.0` (farads)

Liberate MX simplifies a partition by substituting a passive load for active devices. If the passive load required is larger than the specified threshold, then don't perform this substitution. Default is `1.0` farads (a large number.)

Example:

```
set_var mx_active_load_thresh "1e-15"
```

Use the mx_active_load_channel_thresh and mx_active_load_gate_thresh parameters instead.

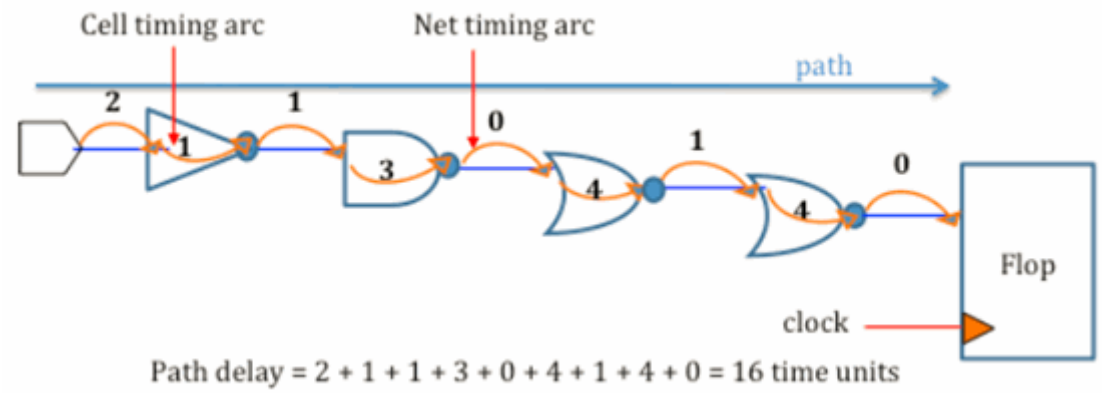
Glossary

Inside View

All characterization solutions available in the Liberate Characterization Portfolio 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

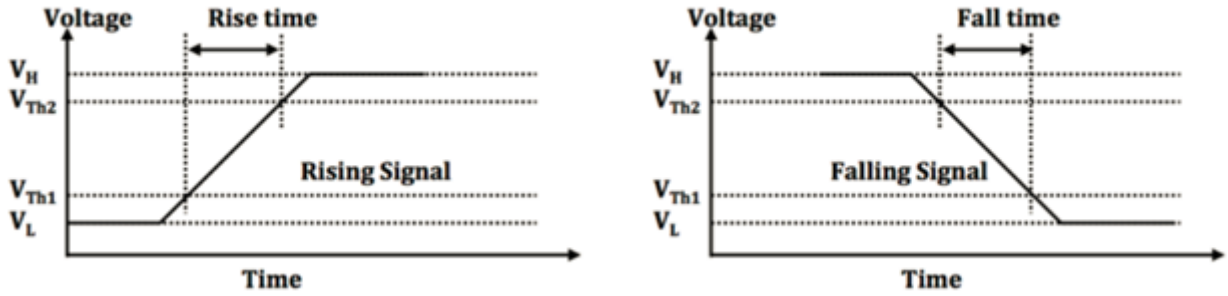
The Liberty format (`.lib`) is the industry standard for specifying the timing information. It is an ASCII file containing timing and power numbers associated with a cell. These are obtained by running SPICE simulations on the cell under a range of conditions.

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.

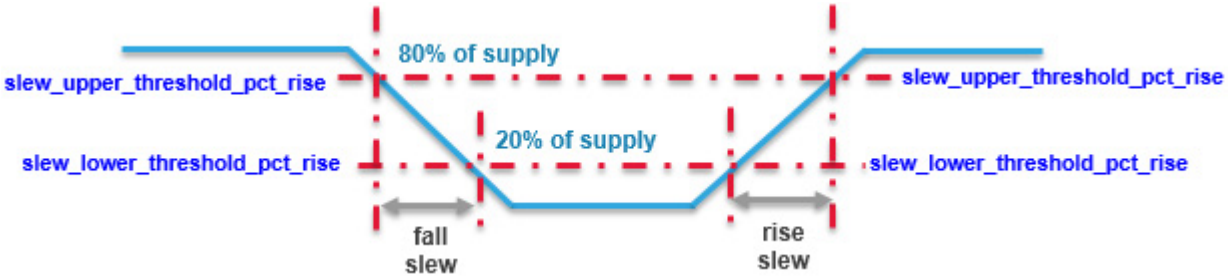
Transition Time

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.