

# **Virtuoso Liberate MX Reference Manual**

**Product Version 16.1**  
**January 2017**

© 2006–2017 Cadence Design Systems, Inc. All rights reserved.

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

**Trademarks:** Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks marks are the property of their respective owners.

**Restricted Permission:** This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

**Disclaimer:** Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

**Restricted Rights:** Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

---

# Contents

---

<b><u>Preface</u></b> .....	13
<b><u>Introduction to Characterization</u></b> .....	13
<u>The Role and Importance of Libraries</u> .....	13
<u>A Growing Problem</u> .....	13
<b><u>Virtuoso Characterization Suite</u></b> .....	15
<u>System Requirements</u> .....	16
<u>Software and Licensing Requirements</u> .....	17
<b><u>About This Manual</u></b> .....	17
<b><u>Audience Profile</u></b> .....	17
<b><u>Additional Documents for Reference</u></b> .....	18
<u>Rapid Adoption Kits</u> .....	18
<u>Application Notes</u> .....	18
<b><u>Typographic and Syntax Conventions</u></b> .....	19
<b><u>Customer Support</u></b> .....	19
<b><u>Feedback about Documentation</u></b> .....	20
<b><u>1</u></b> .....	
<b><u>Overview of Memory Characterization</u></b> .....	21
<u>Contents of Liberty Format File</u> .....	21
<u>Types of Memory to be Characterized</u> .....	22
<u>Pin Characterization</u> .....	22
<u>Pin Capacitance Characterization</u> .....	22
<u>Noise Characterization</u> .....	22
<u>Timing Characterization</u> .....	23
<u>Delay and Retain Arcs Characterization</u> .....	23
<u>Constraint Arcs Characterization</u> .....	24
<u>Minimum Pulse Width Arcs Characterization</u> .....	28
<u>Minimum Period Arcs</u> .....	33
<u>Power Characterization</u> .....	35
<u>Static Power (Leakage) Characterization</u> .....	36

---

<u>Dynamic Power Characterization</u> .....	36
 <b>2</b>	
<u>Overview of Liberate MX</u> .....	37
<u>What is Liberate MX?</u> .....	37
 <b>3</b>	
<u>Getting Started with Liberate MX</u> .....	41
<u>Tool Installation and Setup</u> .....	41
<u>Installing Liberate MX</u> .....	41
<u>Managing Licenses</u> .....	42
<u>System Libraries</u> .....	44
<u>Inputs Required for Liberate MX Characterization</u> .....	44
<u>Setting Up a Template File: An Example</u> .....	46
<u>Running Liberate MX</u> .....	50
 <b>4</b>	
<u>Liberate MX Flows</u> .....	51
<u>Characterization Flows</u> .....	52
<u>Automatic Characterization Flow</u> .....	52
<u>Manual Characterization Flow</u> .....	56
<u>Validation Flow</u> .....	58
<u>Setting Up Validation Flow</u> .....	60
<u>Determining Whether the Run is Passing</u> .....	63
<u>Debugging the Failing Arcs</u> .....	66
 <b>5</b>	
<u>Liberate MX Commands</u> .....	69
<u>add_margin</u> .....	71
<u>char_macro</u> .....	73
<u>char_memory</u> .....	78
<u>compare_path</u> .....	79
<u>compare_timing</u> .....	81

## Virtuoso Liberate MX Reference Manual

---

<u>compare waveforms</u>	82
<u>define arc</u>	83
<u>define cell</u>	100
<u>define duplicate pins</u>	103
<u>define index</u>	104
<u>define leafcell</u>	106
<u>define leakage</u>	108
<u>define measure</u>	109
<u>define memory</u>	119
<u>define table</u>	135
<u>define template</u>	136
<u>hspice lis 2 waves</u>	139
<u>interpolate define axis expr</u>	139
<u>mxcore def</u>	140
<u>mx_match_node</u>	140
<u>mx merge</u>	142
<u>mx_recover clean</u>	142
<u>mx_recover info</u>	143
<u>mx_recover merge</u>	143
<u>mx_recover setup</u>	144
<u>mx_report</u>	145
<u>mx set constprop</u>	146
<u>mx set domainprop</u>	147
<u>mx_set ultrasim param</u>	148
<u>report measure</u>	148
<u>set_gnd</u>	150
<u>set_vdd</u>	151
<u>set virtual</u>	152
<u>spv tcl 2 waves</u>	153
<u>validate_macro</u>	154
<u>validate measure</u>	155
<u>validate memory</u>	155
<u>write_ldb</u>	156
<u>write library</u>	157
<u>write_verilog</u>	164

## 6

<b><u>Liberate MX Variables</u></b> .....	171
<u>constraint glitch peak</u> .....	176
<u>constraint glitch peak max</u> .....	176
<u>constraint glitch peak mode</u> .....	176
<u>constraint probe lower fall</u>	
<u>constraint probe hold lower fall</u>	
<u>constraint probe setup lower fall</u> .....	177
<u>constraint probe lower rise</u>	
<u>constraint probe hold lower rise</u>	
<u>constraint probe setup lower rise</u> .....	178
<u>constraint probe upper fall</u>	
<u>constraint probe hold upper fall</u>	
<u>constraint probe setup upper fall</u> .....	178
<u>constraint probe upper rise</u>	
<u>constraint probe hold upper rise</u>	
<u>constraint probe setup upper rise</u> .....	178
<u>debug flow</u> .....	178
<u>extsim deck include</u> .....	179
<u>extsim model include</u> .....	179
<u>fastsim cmd</u> .....	180
<u>fastsim cmd option</u> .....	180
<u>keep empty cells</u> .....	180
<u>lic_max timeout</u> .....	181
<u>lic_queue timeout</u> .....	181
<u>mxtable dontcare value</u> .....	181
<u>mxvw_min glitch peak</u> .....	182
<u>mx_active fanout channel include</u> .....	182
<u>mx_active load</u> .....	183
<u>mx_active load channel thresh</u> .....	183
<u>mx_active load gate thresh</u> .....	183
<u>mx_arc report</u> .....	183
<u>mx_auto char params</u> .....	184
<u>mx_automeas filter</u> .....	184
<u>mx_autoprobing hold level</u> .....	185
<u>mx_autoprobing setup level</u> .....	185
<u>mx_bisection</u> .....	186

## Virtuoso Liberate MX Reference Manual

---

<u>mx bitline probe threshold</u>	187
<u>mx char bundle size</u>	187
<u>mx char virtual as rail</u>	188
<u>mx check arcs</u>	188
<u>mx check arcs exit on missing</u>	189
<u>mx clock2clock constraints</u>	189
<u>mx clk2clk mode</u>	189
<u>mx clock tree report</u>	190
<u>mx clone if uda</u>	190
<u>mx const prop</u>	190
<u>mx constraint ocv factor</u>	190
<u>mx corecell</u>	191
<u>mx create if dynamic</u>	191
<u>mx create if uda</u>	192
<u>mx debug</u>	192
<u>mx defmem sdf cond</u>	194
<u>mx delay ocv factor</u>	194
<u>mx dir</u>	194
<u>mx distributed sim</u>	195
<u>mx domain propagation</u>	195
<u>mx dpartition inactive tie</u>	196
<u>mx dynamic include full core</u>	196
<u>mx fastsim auto ic</u>	196
<u>mx fastsim clock slew</u>	197
<u>mx fastsim input slew</u>	197
<u>mx fastsim load</u>	197
<u>mx fastsim reuse</u>	198
<u>mx find arrays</u>	198
<u>mx find memcore numbit threshold</u>	198
<u>mx find memcores</u>	199
<u>mx find stack loads</u>	199
<u>mx find virtual rails</u>	199
<u>mx fix pin vdd</u>	201
<u>mx full rail tol</u>	201
<u>mx fullsim measurement</u>	201
<u>mx greybox</u>	201

## Virtuoso Liberate MX Reference Manual

---

<u>mx greybox constraint method</u>	202
<u>mx inputcap ldb reuse</u>	202
<u>mx ldbs reuse</u>	202
<u>mx margin report</u>	202
<u>mx mcf</u>	203
<u>mx measure error file</u>	203
<u>mx min clock tree mode</u>	
<u>mx max clock tree mode</u>	203
<u>mx min period latch component mode</u>	204
<u>mx min period mode</u>	204
<u>mx minp single slew</u>	205
..... <u>mx monitor memcore</u>	207
<u>mx monitor memcore lprobe level</u>	207
<u>mx mpw allow same probe on both rise and fall clock tree</u>	208
<u>mx mpw false probe delay threshold</u>	208
<u>mx mpw measurement duration</u>	208
<u>mx mpw mode</u>	208
<u>mx mpw probe</u>	209
<u>mx mpw probe lower fall</u>	
<u>mx mpw probe lower rise</u>	
<u>mx mpw probe upper fall</u>	
<u>mx mpw probe upper rise</u>	209
<u>mx mxtable interpret read write cycle keywords</u>	210
<u>mx negedge clock</u>	210
<u>mx noise ldb reuse</u>	210
<u>mx output require fullrail switch</u>	211
<u>mx partition name use arc</u>	211
<u>mx pathdelay hold clock margin</u>	211
<u>mx pathdelay hold data margin</u>	211
<u>mx pathdelay setup clock margin</u>	212
<u>mx pathdelay setup data margin</u>	212
<u>mx pincap char</u>	212
<u>mx posedge clock</u>	212
<u>mx power assign</u>	212
<u>mx power divide num switching mode</u>	213
<u>mx power ldb reuse</u>	214
<u>mx power single point</u>	214



## Virtuoso Liberate MX Reference Manual

---

<u>mx preprocess</u>	214
<u>mx probe peak currents</u>	214
<u>mx probes report</u>	215
<u>mx process probe relprobe intersection</u>	215
<u>mx process probe relprobe intersection max paths</u>	216
<u>mx process probe relprobe intersection max depth</u>	216
<u>mx read spice exit on missing file</u>	216
<u>mx remove false ic group</u>	216
<u>mx remove rc pincap</u>	217
<u>mx remove rc timing</u>	217
<u>mx retaining time</u>	218
<u>mx ring large ccc max paths</u>	218
<u>mx ring large ccc max path depth</u>	219
<u>mx ring model fold</u>	219
<u>mx seq probing</u>	219
<u>mx setup seq</u>	
<u>mx hold seq</u>	
<u>mx setup comb</u>	
<u>mx hold comb</u>	220
<u>mx simulation interval</u>	221
<u>mx skip autoprobng</u>	222
<u>mx skip print</u>	222
<u>mx spv api</u>	223
<u>mx timing ldb reuse</u>	223
<u>mx timing report</u>	223
<u>mx timing report debug</u>	223
<u>mx total active load channel thresh</u>	224
<u>mx total active load gate thresh</u>	224
<u>mx verbose</u>	224
<u>mx virtual rail auto mode</u>	224
<u>mx virtual rail opposite device minimum factor</u>	225
<u>mx virtual rail minimum xtrs</u>	225
<u>mx whitebox active coupling threshold</u>	225
<u>mx whitebox active wire threshold</u>	226
<u>mx whitebox model file</u>	226
<u>mx zip partition deck</u>	226

<u>parse auto define leafcell</u> .....	226
<u>set var failure action</u> .....	227
<u>virtual rail waveform probe</u> .....	228

## A

<u>Truth Table Format</u> .....	229
---------------------------------	-----

<u>Specifying Input Stimuli</u> .....	229
<u>define table command</u> .....	229
<u>fastsim</u> .....	230
<u>fastsim auto ic</u> .....	230
<u>fastsim clock slew</u> .....	231
<u>fastsim input slew</u> .....	231
<u>fastsim load</u> .....	231
<u>fastsim cmd</u> .....	232
<u>fastsim cmd option</u> .....	232
<u>monitor memcore</u> .....	233
<u>fastsim deck</u> .....	233
<u>fastsim deck include</u> .....	234
<u>fastsim distributed sim</u> .....	234
<u>fastsim model include</u> .....	235
<u>Guidelines for Writing Truth Table for Multiport Memories</u> .....	235
<u>Required Keywords</u> .....	236
<u>Truth Table Example</u> .....	240
<u>Using Tcl Variable Inside the Table Files</u> .....	241

## B

<u>Specifying Memory Core Cells</u> .....	243
---	-----

## C

<u>Using Liberate MX Automatic Flow</u> .....	251
---	-----

<u>Defining Liberate MX Settings for Automatic Flow</u> .....	251
<u>Primary Tcl File</u> .....	251
<u>Additional Table Files</u> .....	258
<u>Debugging Common Issues with Automatic Flow</u> .....	259

---

<u>Pin Name Recognition Issues</u> .....	259
<u>Missing Arcs</u> .....	260

## D

<u>Liberate MX Timing Validation Flow</u> .....	261
---	-----

## E

<u>Basic Flow for Validating User-Defined Criteria</u> .....	263
--	-----

## F

<u>Liberate MX Compiler Characterization Flow</u> .....	265
---	-----

<u>Introduction to Memory Compilers</u> .....	265
<u>Memory Compiler Characterization with Liberate MX</u> .....	267
<u>Defining the Compiler Space</u> .....	268
<u>Defining the PVT Corner for Characterization</u> .....	271
<u>Defining the Characterized Instances</u> .....	272
<u>Defining the Interpolated Instances</u> .....	275
<u>Interpolation</u> .....	277
<u>The Interpolation Formula</u> .....	277
<u>Example Control File</u> .....	278
<u>License Requirements</u> .....	281

## G

<u>Legacy Commands and Variables</u> .....	283
--	-----

<u>Deprecated Commands</u> .....	283
<u>mx_set_clockprop</u> .....	283
<u>mx_set_finesim_param</u> .....	283
<u>mx_set_hsim_param</u> .....	284
<u>mx_set_nanosim_param</u> .....	284
<u>Deprecated Variables</u> .....	285
<u>mx_remove_rc</u> .....	285
<u>mx_whitebox_monitor_memcores</u> .....	285
<u>mx_active_load_thresh</u> .....	285

H

<u>Glossary</u> .....	287
-----------------------	-----

---

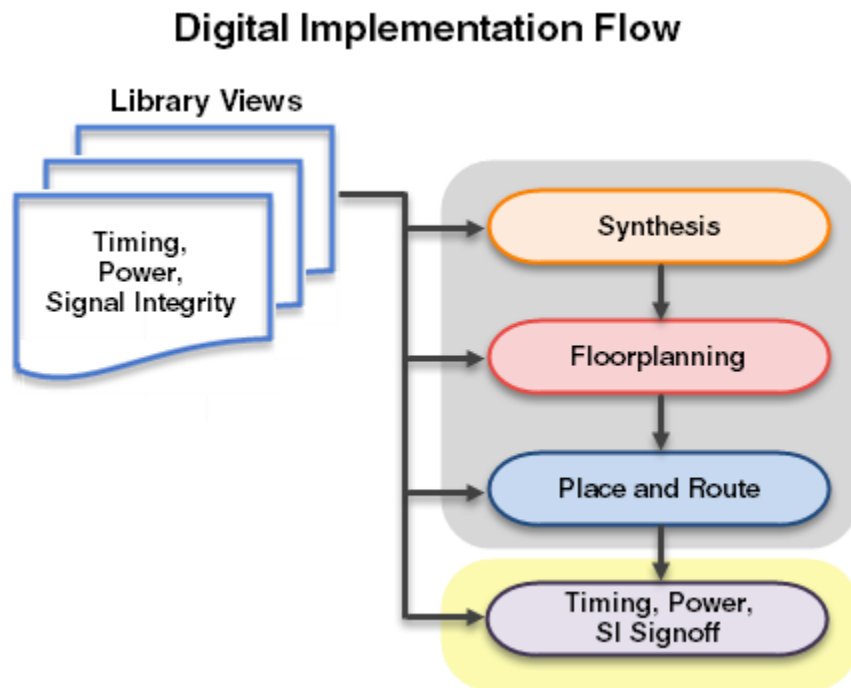
# Preface

---

## Introduction to Characterization

### The Role and Importance of Libraries

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

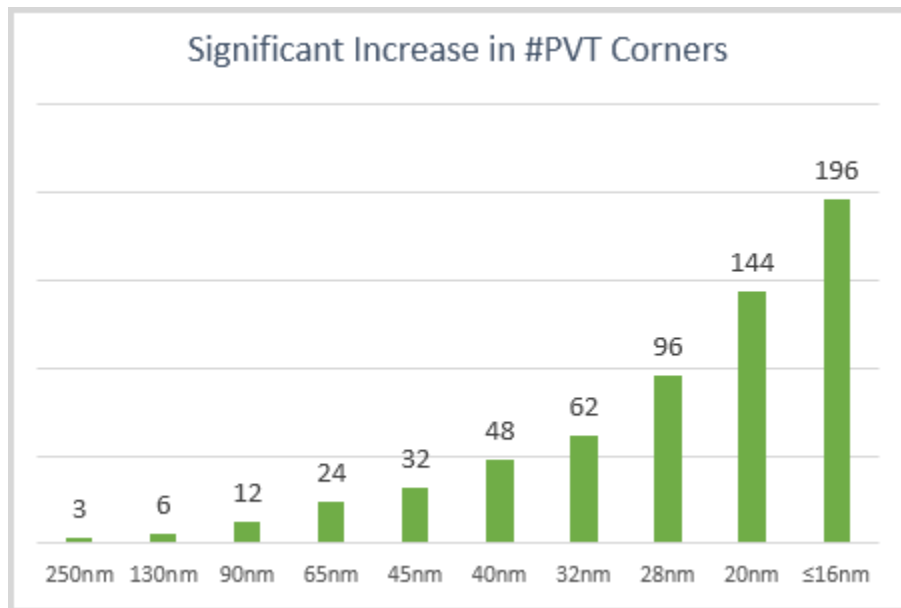


### A Growing Problem

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

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

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

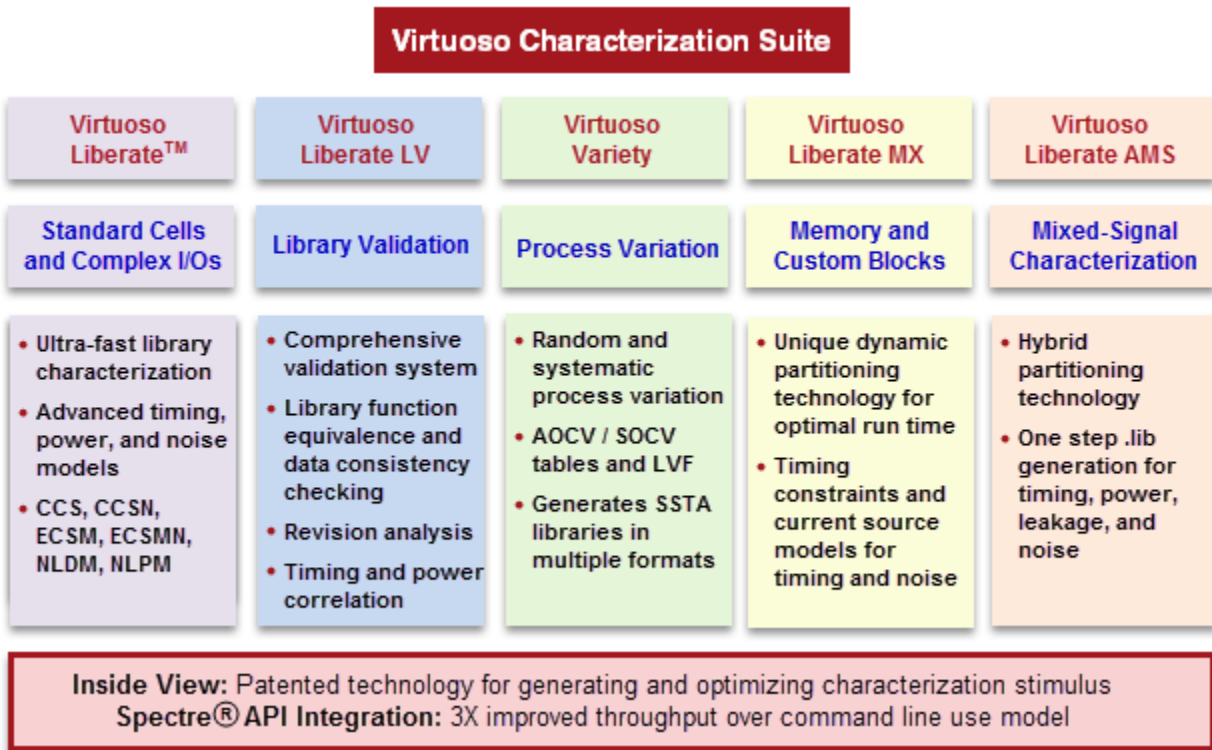


In addition, library views require more advanced models like:

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

## Virtuoso Characterization Suite

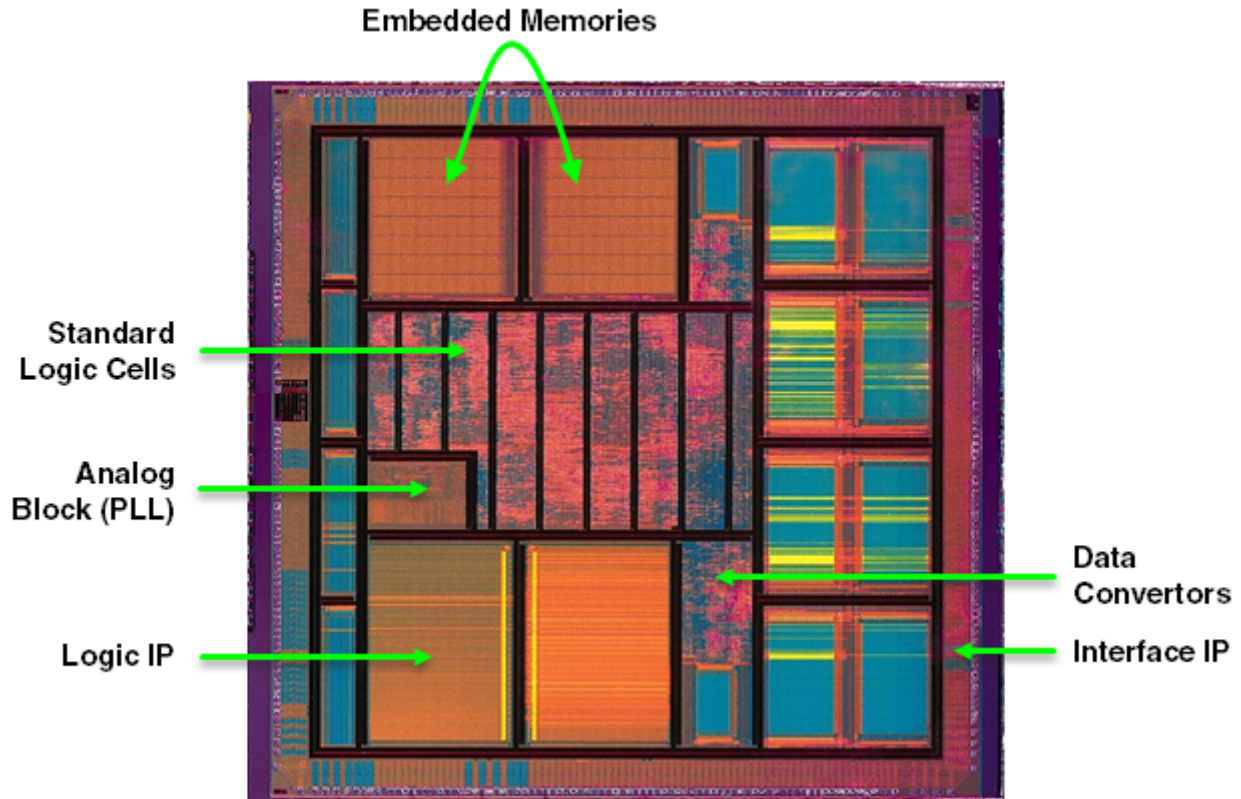
To address all the challenges, Cadence offers Virtuoso® Characterization Suite that covers the complete portfolio of characterization solutions given below:



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

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

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



## System Requirements

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

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

For detailed information about the requirements, see [Computing Platforms](#).



## Software and Licensing Requirements

### ■ LIBERATE 15.1

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

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

### ■ MMSIM 15.1

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

## About This Manual

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

## Audience Profile

This manual is aimed at developers and designers who want to work on library creation for memory instances. It assumes that you are familiar with:

- SPICE simulations
- Basic expected behavior of the design being used

## Additional Documents for Reference

For information about known problems and solutions, see [\*Virtuoso Characterization Suite Known Problems and Solutions\*](#).

For a list of new features in a release, see [\*Virtuoso Characterization Suite What's New\*](#).

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

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

## Rapid Adoption Kits

Cadence provides [\*Rapid Adoption Kits\*](#) that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

## Application Notes

The following [\*application notes\*](#) are available on the Cadence Online Support website:

- Setting up Liberate MX for Various Usage Models
- Liberate MX: Debugging Netlist Issues
- Using and Debugging the Liberate MX Validation Flow

- Characterizing Minimum Period and Minimum Pulse Width Using Liberate MX

## Typographic and Syntax Conventions

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

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

## 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.

---

# 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.

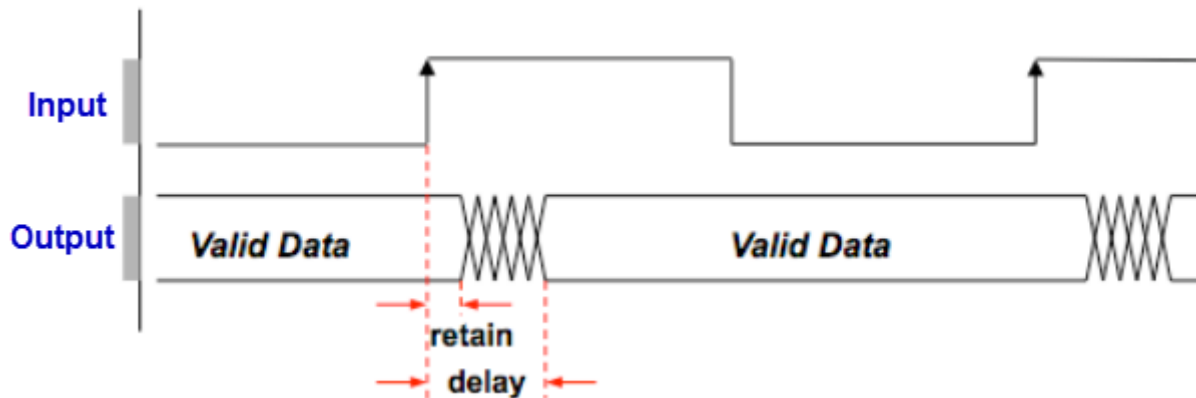
## Timing Characterization

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.

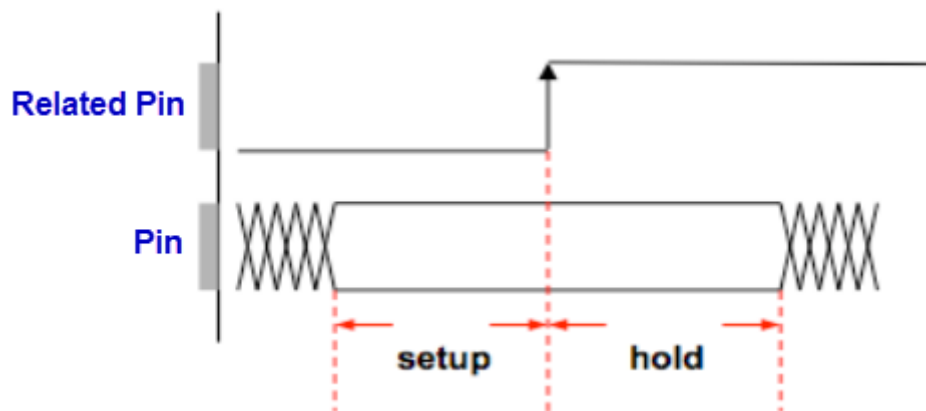
When creating input stimulus for a memory instance, it is important to capture the fastest paths and the slowest paths to ensure the correct data for both delay and retain. This means that for each output direction, rise and fall, there should be vectors that cover the following:

- 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

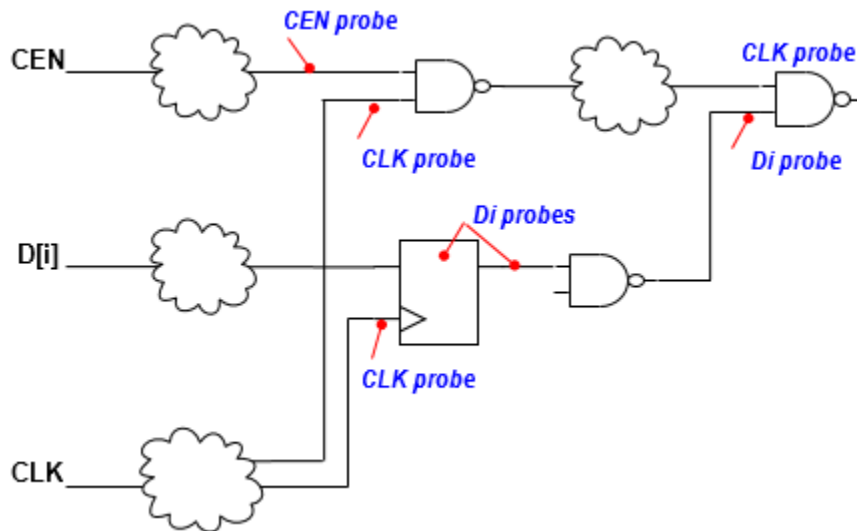


## Virtuoso Liberate MX Reference Manual

### Overview of Memory Characterization

---

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.



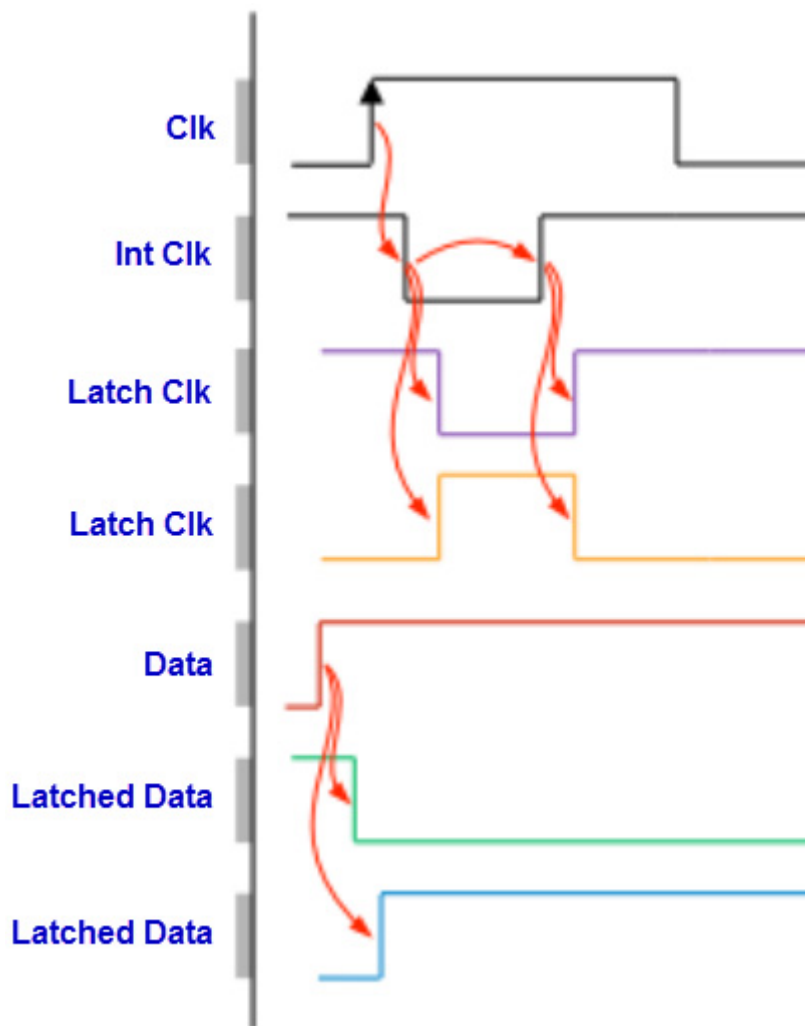
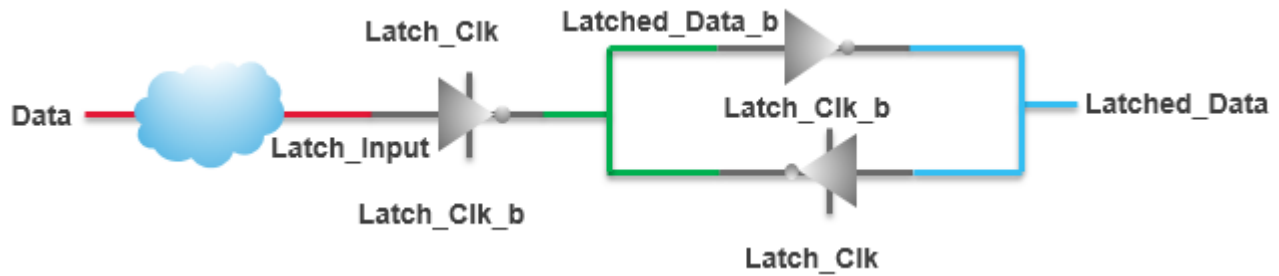
If the first stage is transparent during the situation that is being characterized for, then the following stage should be characterized as well. Typically, the first stage will be a latch and subsequent stages will be some form of combinational logic. In the case of setup time, the latch will be transparent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

$$\text{Setup Time} = \text{Max} (\text{Data} \rightarrow \text{Latched\_Data\_b}, \text{Data} \rightarrow \text{Latched\_Data}) \\ - \text{Min}(\text{Clk} \rightarrow \text{Latch\_Clk\_b}, \text{Clk} \rightarrow \text{Latch\_Clk})$$

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

$$\text{Hold Time} = \text{Max} (\text{Clk} \rightarrow \text{Latch\_Clk\_b}, \text{Clk} \rightarrow \text{Latch\_Clk}) - \text{Latch\_Input}$$

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

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 very 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.

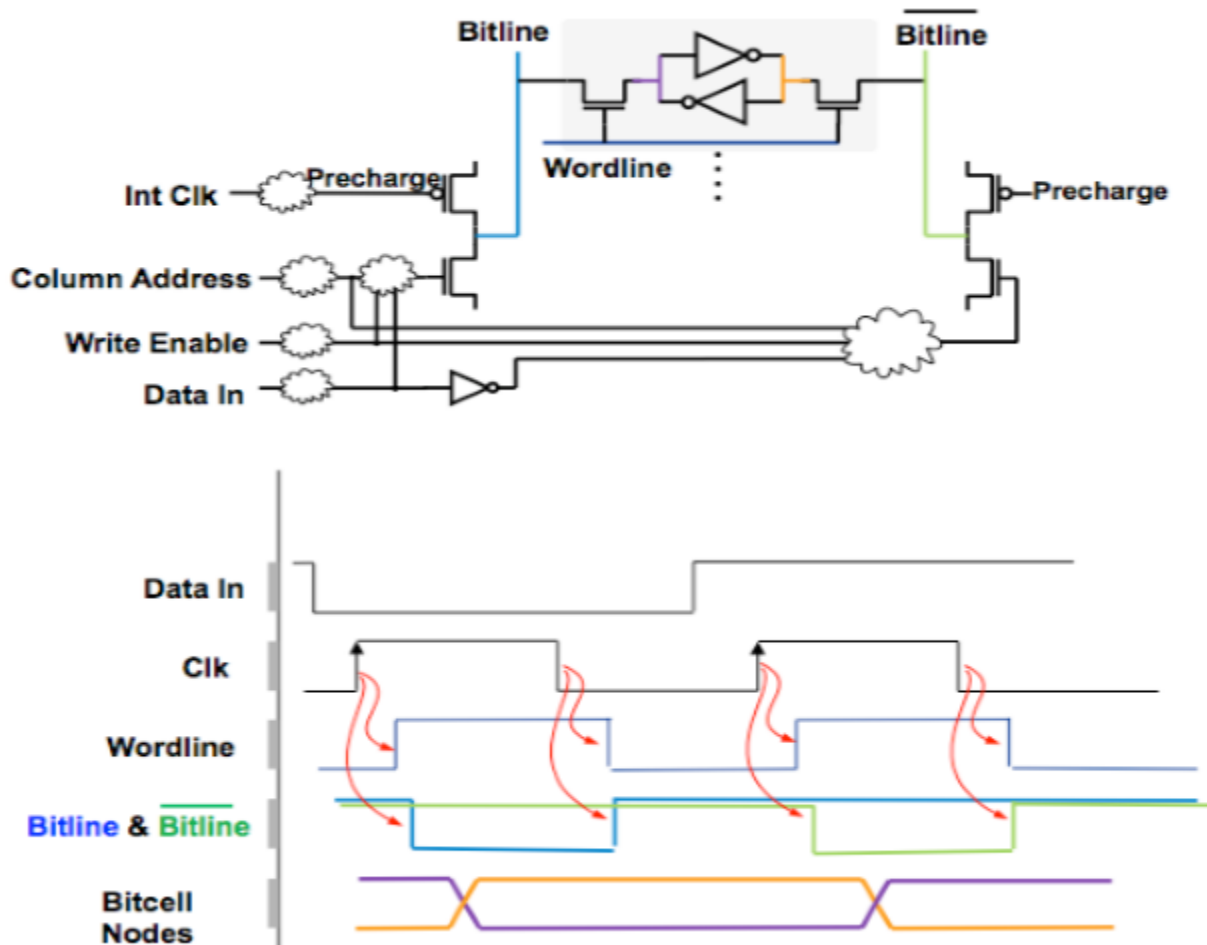
The inactive portion (*usually low*) of the clock cycle requires the following events to complete:

## Virtuoso Liberate MX Reference Manual

### Overview of Memory Characterization

- 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:



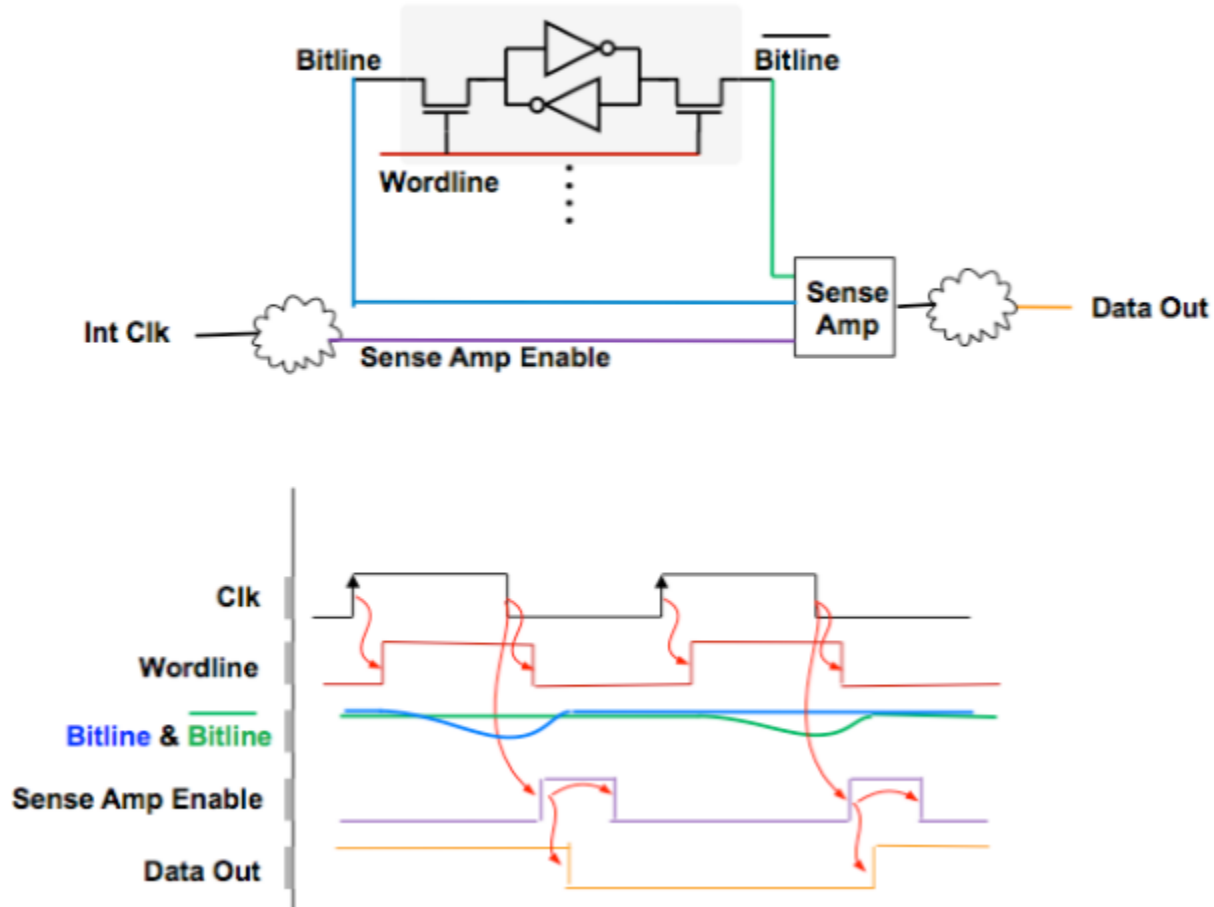
In the figure above, the wordline and the bitline are controlled by an external clock. The wordline should stay high and the bitline should stay low, but long enough to reliably write data into the bitcell. As this is a critical operation, some margin should usually be added beyond the actual time to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

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

## Virtuoso Liberate MX Reference Manual

### Overview of Memory Characterization

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.

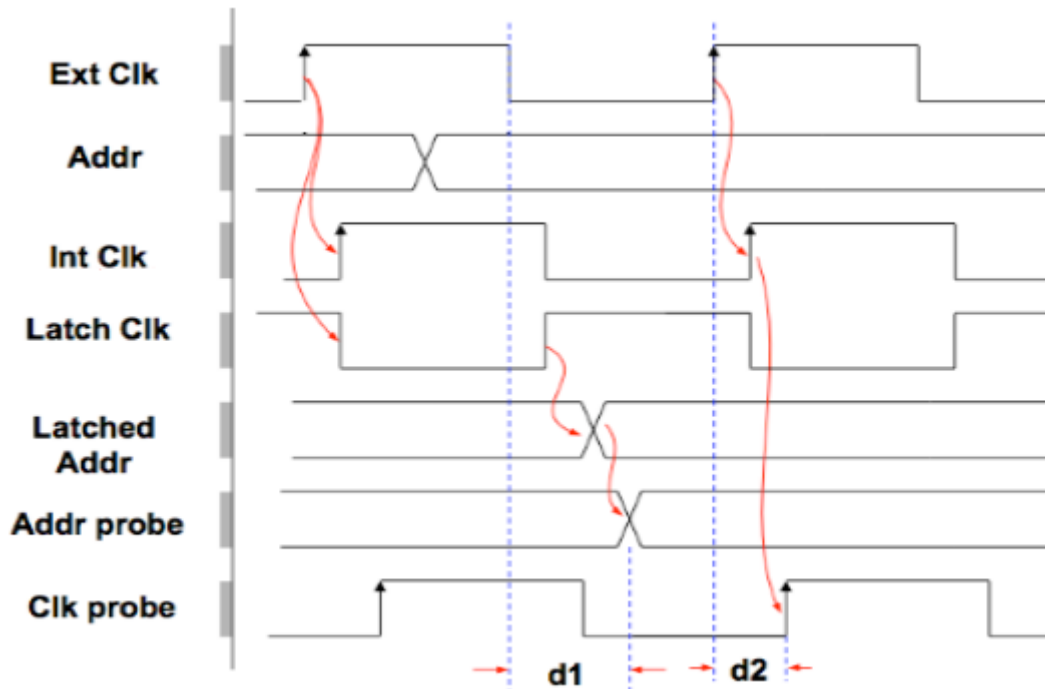
Minimum Pulse Width High = (Clk rise -> Bitline discharged) - Min( ( Clk fall -> Sense Amp Enable rise) , ( Clk fall -> Wordline fall ) )

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.

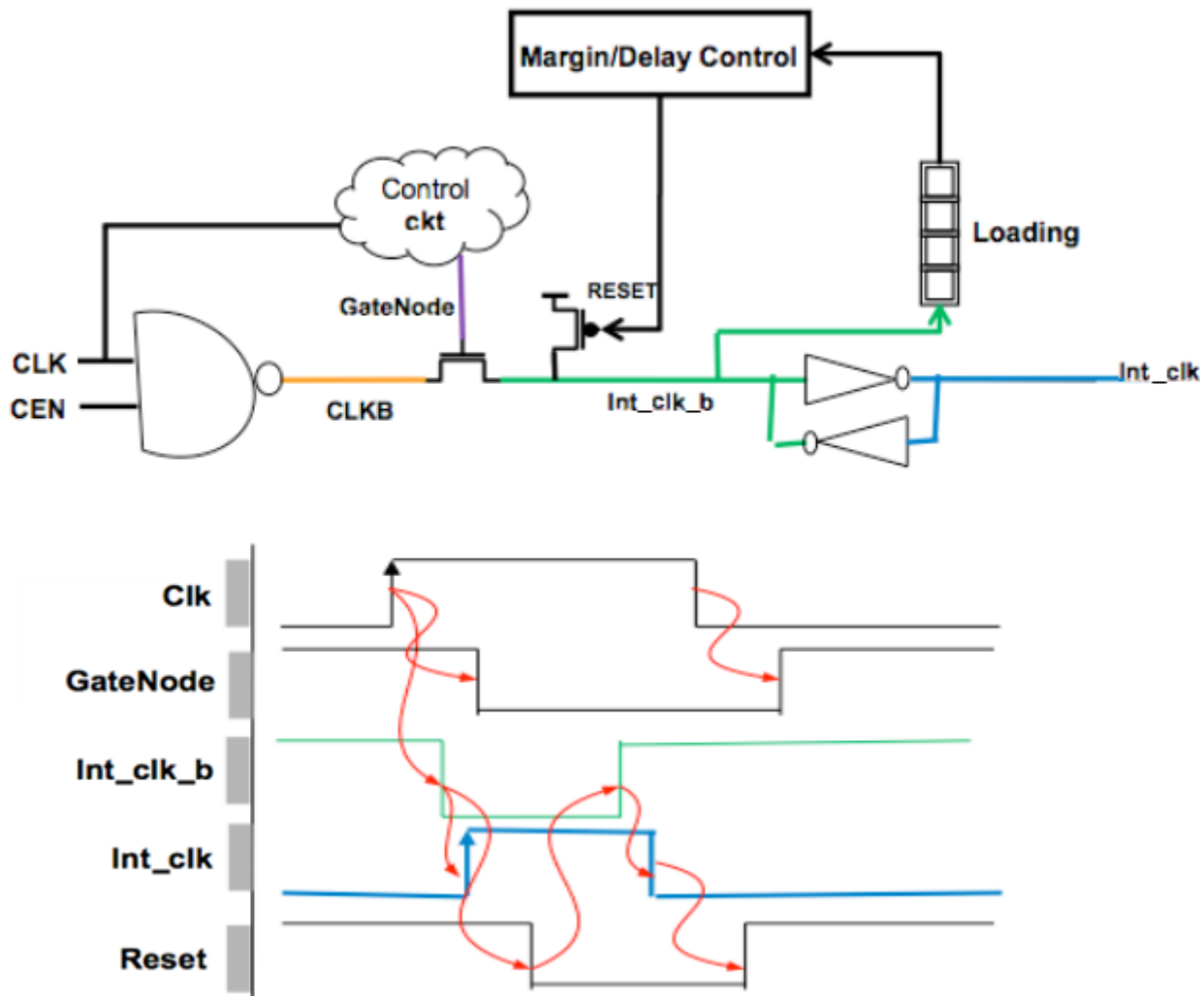
### Minimum Pulse Width in the Self-Timed Memory

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

## Virtuoso Liberate MX Reference Manual

### Overview of Memory Characterization

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

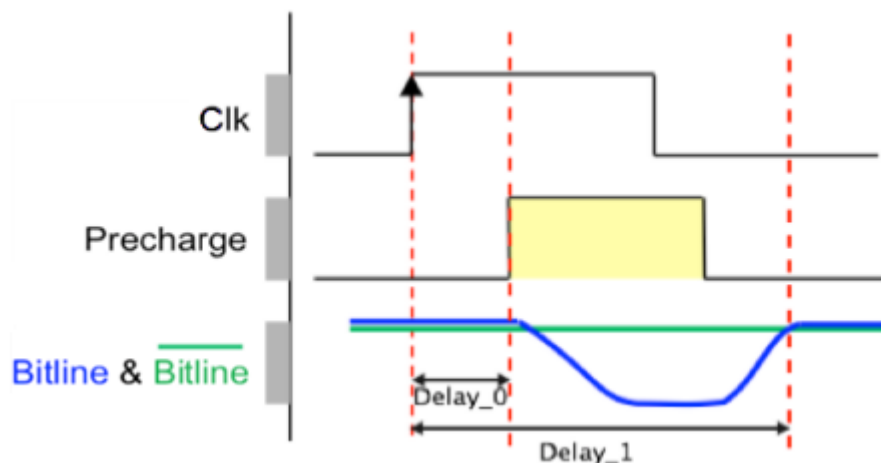
In the externally-timed memory, the two phases of a clock separately control the different portions of the active cycle. As there is no further dependence on the spacing between two like edges, the following equation is used:

$$\text{Minimum Period} = (\text{Minimum Pulse Width High}) + (\text{Minimum Pulse Width Low})$$

### 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

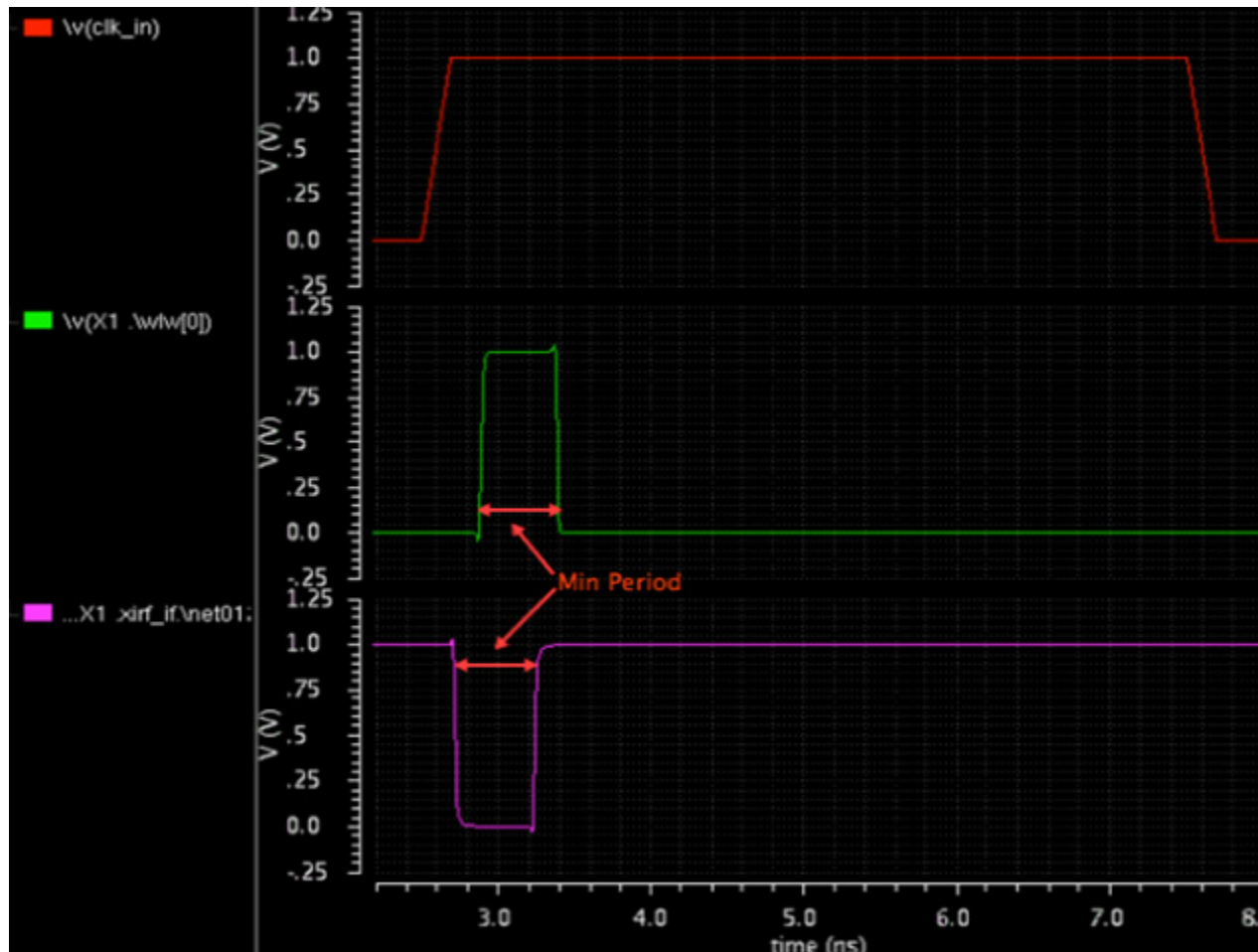


## Virtuoso Liberate MX Reference Manual

### Overview of Memory Characterization

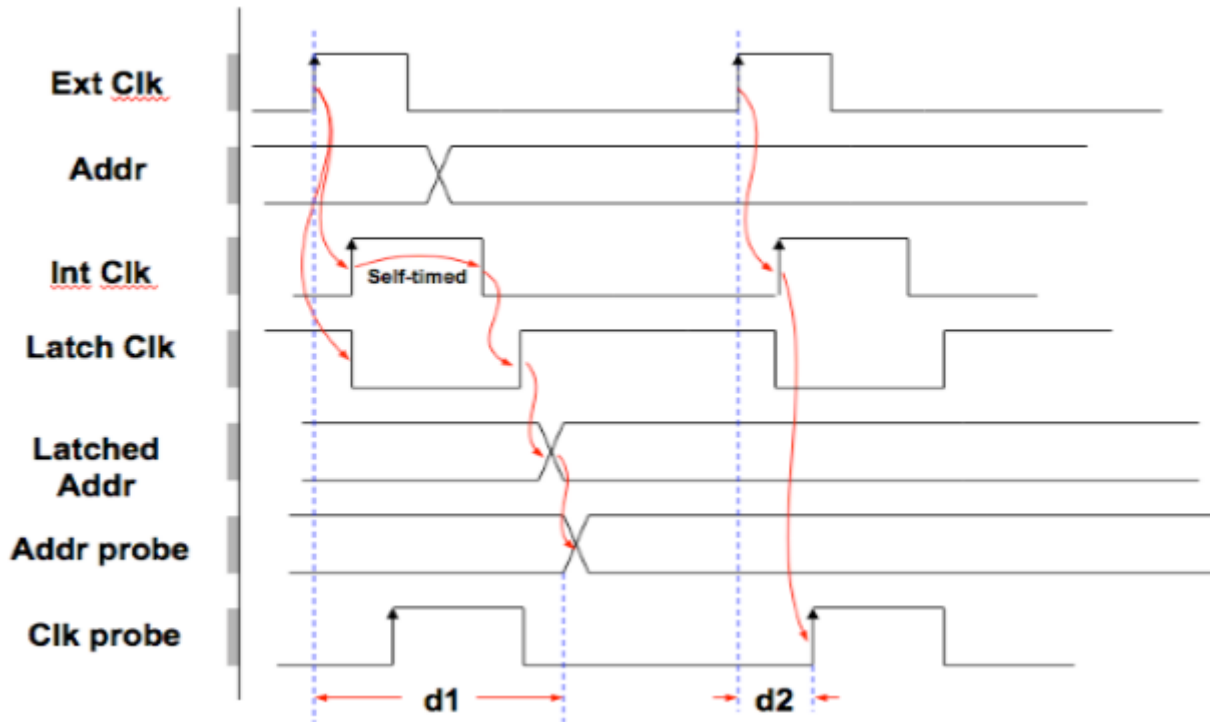
The bitline must return to its initial state (usually high) before the precharge turns off in the next cycle. This dictates the spacing between the two active edges of the clock. The threshold used for the bitline voltage will be higher than thresholds used for other internal signals and will usually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

---

## Overview of Liberate MX

---

### What is Liberate MX?

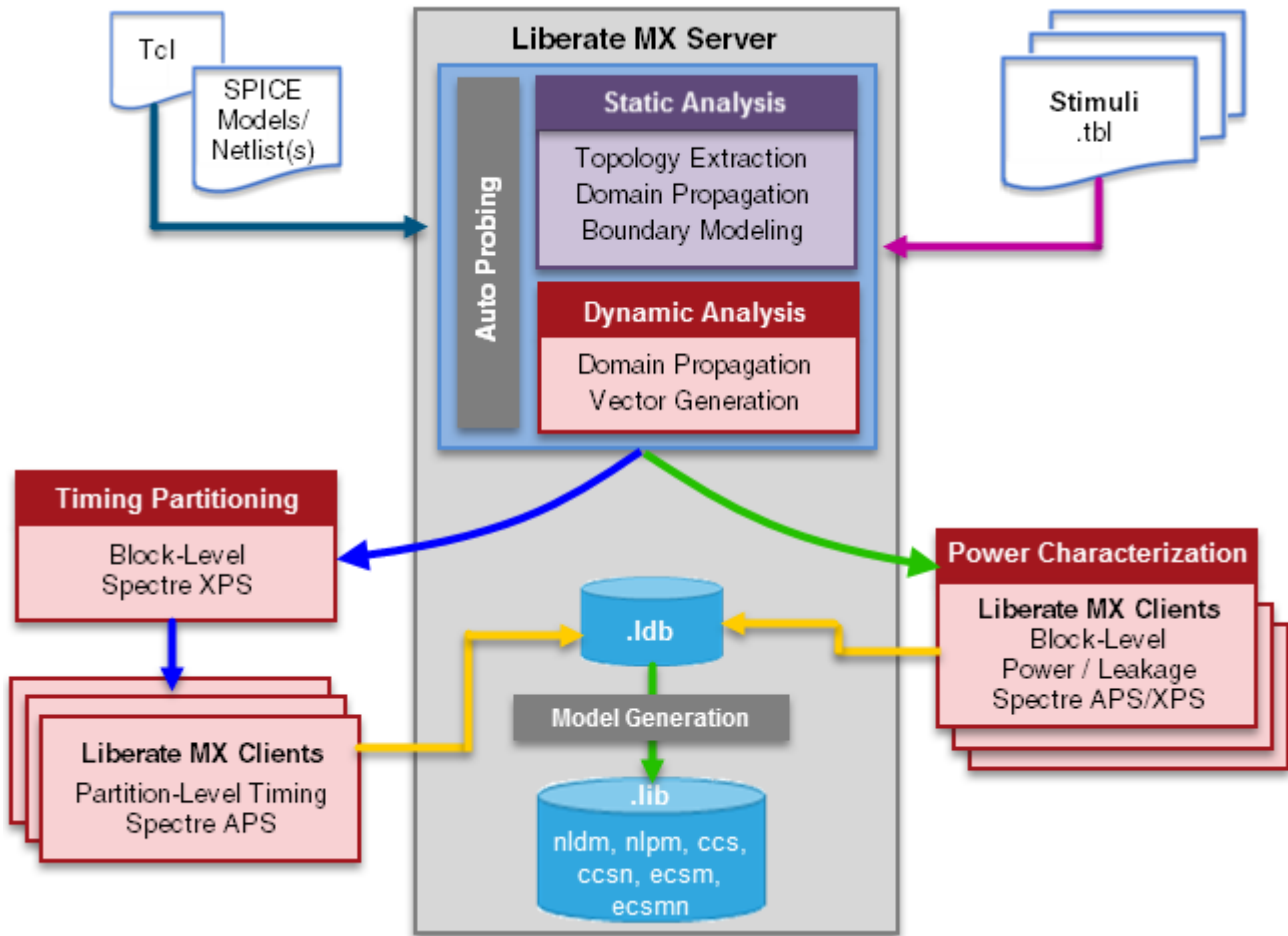
Cadence® Virtuoso® Liberate™ MX provides library creation capabilities to cover the memory cores. Embedded and custom memories comprise a large percentage of silicon area on most chips and consequently, can often be major contributors to chip performance and power consumption. To validate a design's 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 *Virtuoso Liberate Reference Manual*.

## Virtuoso Liberate MX Reference Manual

### Overview of Liberate MX

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

## Virtuoso Liberate MX Reference Manual

### Overview of Liberate MX

---

- 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.

# **Virtuoso Liberate MX Reference Manual**

## Overview of Liberate MX

---



---

## Getting Started with Liberate MX

---

This chapter describes briefly how to start using Cadence® Virtuoso® 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.

### 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)
```

## Managing Licenses

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

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

### *Important*

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

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

## Waiting for Available License

When you submit a Liberate MX job, a request is made for a license. To request that Liberate MX wait until a license becomes available, set the following environment variable:

```
setenv ALTOS_QUEUE 1
```

## How Liberate MX Uses Licenses

Liberate MX clients run using different types of client license features, depending on the product names. Some product licenses can be mixed and matched together. Liberate MX clients can run using the `Liberate_MX_Client` product license.

When a Liberate MX server starts, it checks out the `Liberate_MX_Server` license. Later, when simulations are ready to begin, Liberate MX tries to check out N clients, where N is the number of threads specified in the `char_macro -thread` command option.

## Virtuoso Liberate MX Reference Manual

### Getting Started with Liberate MX

---

When `ALTOS_QUEUE` is set to 1,

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

When `ALTOS_QUEUE` is set to 0,

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

### Environment Variables for Controlling Licensing Checks

#### ■ `ALTOS_LIC_MAX_TIMEOUT`

```
setenv ALTOS_LIC_MAX_TIMEOUT <value>
```

where;

*value* is duration in seconds.

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

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

#### ■ `ALTOS_LIC_CHECK_ALT_TIMEOUT`

```
setenv ALTOS_LIC_CHECK_ALT_TIMEOUT <value>
```

where;

*value* is duration in seconds.

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

## 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, Berkeley SPICE, and HSpice® 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
```

## Virtuoso Liberate MX Reference Manual

### Getting Started with Liberate MX

---

```
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)
#####
```

## Virtuoso Liberate MX Reference Manual

### Getting Started with Liberate MX

---

```
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
```



## Virtuoso Liberate MX Reference Manual

### Getting Started with Liberate MX

---

```
#-----
# 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
```

## Virtuoso Liberate MX Reference Manual

### Getting Started with Liberate MX

---

```
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, make sure that it is installed correctly and that all the necessary prerequisite data are available.

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

```
% setenv ALTOS_64 1
```

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

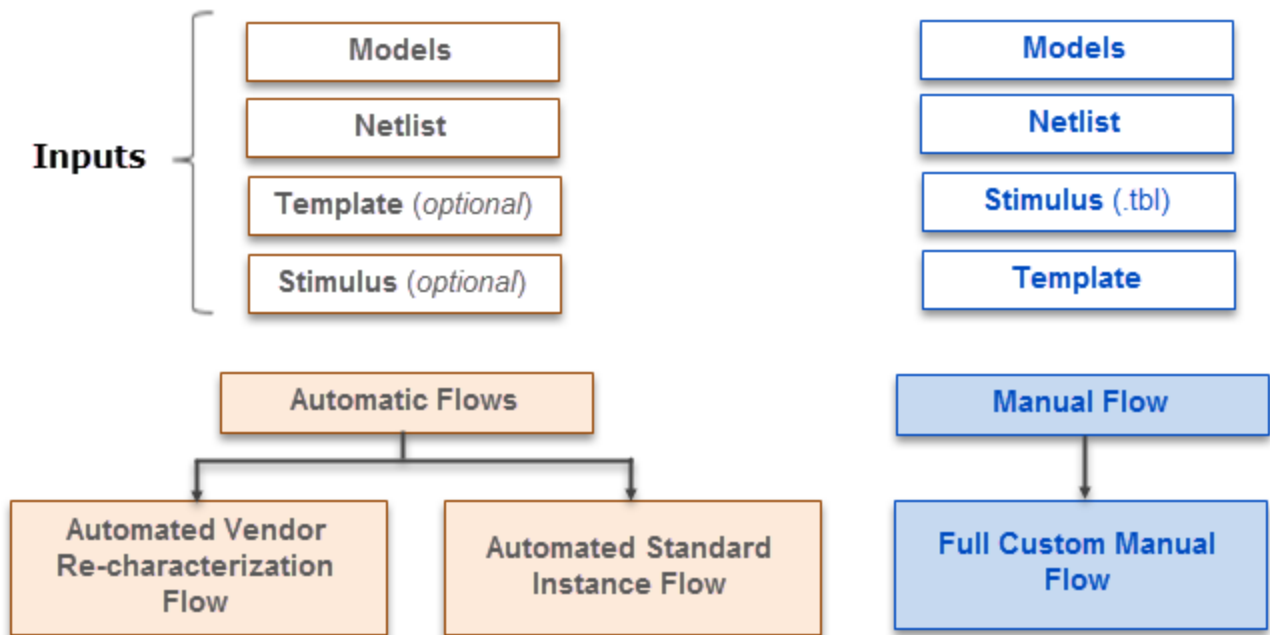
```
% liberate_mx mx.tcl
```

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
```

## 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 how Liberate MX functions in them:

### ■ Characterization Flows

- Automatic Characterization Flow
  - Automated Vendor Re-characterization Flow
  - Automated Standard Instance Flow

❑ Manual Characterization Flow

■ Validation 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 C, “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 \
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Flows

---

```
-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 <code>process_node</code> 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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Flows

---

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 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. 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`.

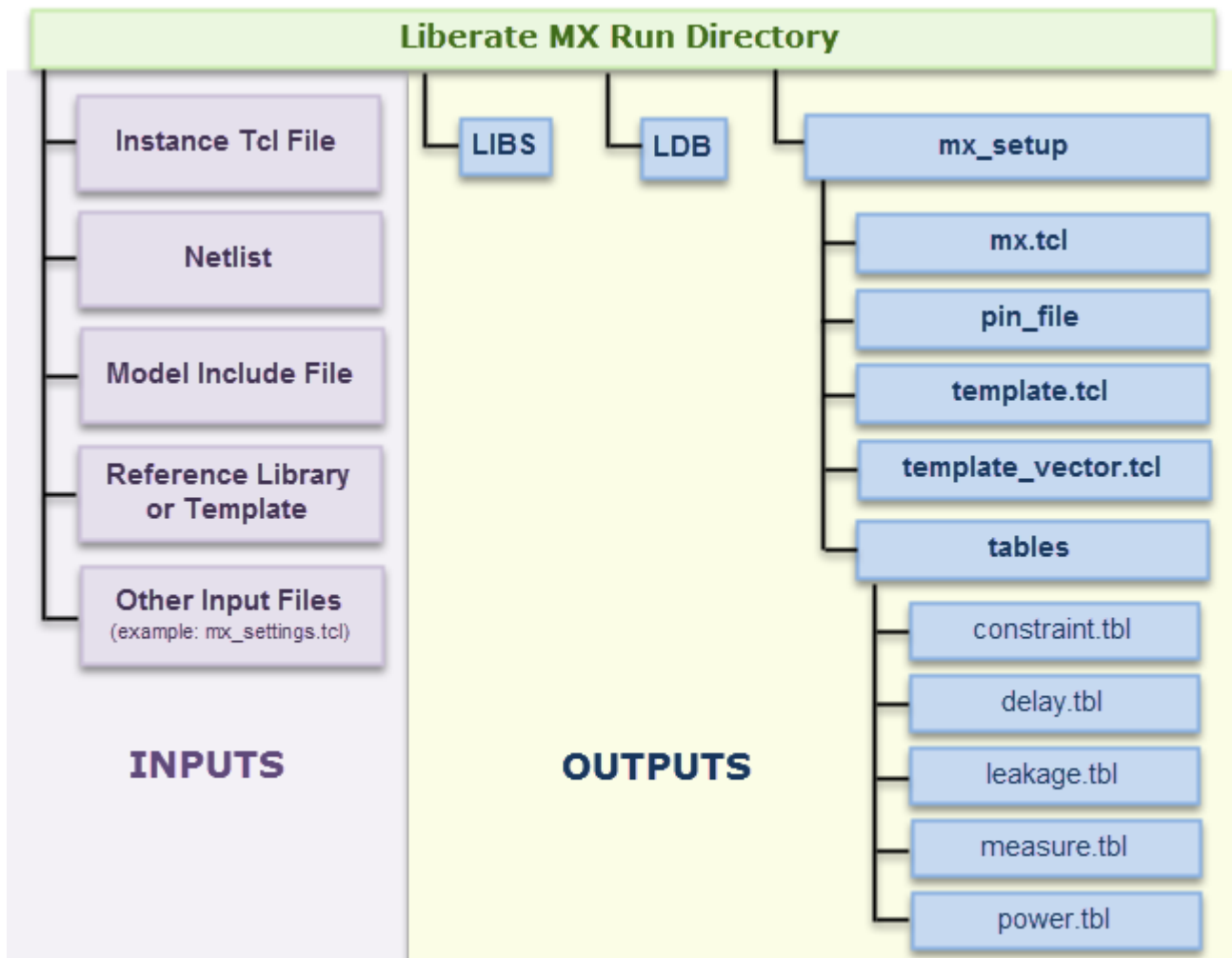
The temperature will always need to be defined using the `-temp` option.

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.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Flows

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 variables.

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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Flows

---

```
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}
```

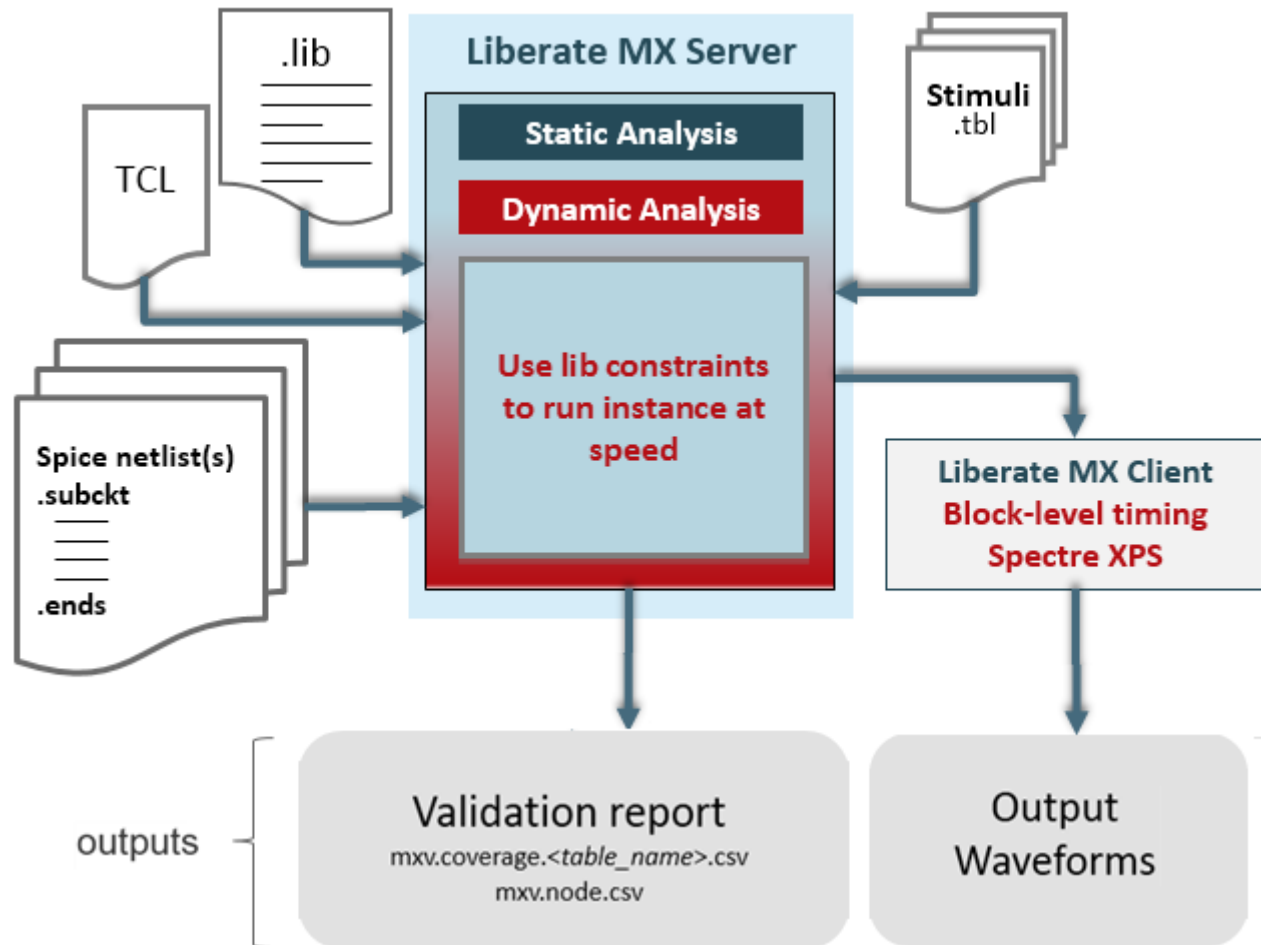
The netlist and model files are included by using the `read_spice` command. The successful reading of the netlist is always required, but the models only need to be read if the netlist contains MOS devices, as opposed to macro model devices. The models should always be specified using `extsim_model`.

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 D, “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 E, “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.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Flows

---

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
```

## Virtuoso Liberate MX Reference Manual


### Liberate MX Flows

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

mxv.coverage.test2.tbl.csv							
1	Covered	Type	Pin	PinDir	Rel	RelDir	When
2	N	SETUP	cs	R	clk	R	
3	N	SETUP	cs	F	clk	R	
4	N	HOLD	cs	R	clk	R	
5	N	HOLD	cs	F	clk	R	
6	Y	SETUP	adr[4]	R	clk	R	
7	Y	SETUP	adr[4]	F	clk	R	
8	Y	HOLD	adr[4]	R	clk	R	
9	Y	HOLD	adr[4]	F	clk	R	
10	Y	SETUP	adr[3]	R	clk	R	
11	Y	SETUP	adr[3]	F	clk	R	
12	Y	HOLD	adr[3]	R	clk	R	
13	Y	HOLD	adr[3]	F	clk	R	
14	Y	SETUP	adr[2]	R	clk	R	
15	Y	SETUP	adr[2]	F	clk	R	
16	Y	HOLD	adr[2]	R	clk	R	
17	Y	HOLD	adr[2]	F	clk	R	
18	Y	SETUP	adr[1]	R	clk	R	
19	Y	SETUP	adr[1]	F	clk	R	
20	Y	HOLD	adr[1]	R	clk	R	
21	Y	HOLD	adr[1]	F	clk	R	
22	Y	SETUP	adr[0]	R	clk	R	
23	Y	SETUP	adr[0]	F	clk	R	
24	Y	HOLD	adr[0]	R	clk	R	



25	Y	HOLD	adr[0]	F	clk	R	
26	Y	SETUP	bw[3]	R	clk	R	
27	Y	HOLD	bw[3]	R	clk	R	
28	Y	SETUP	bw[2]	R	clk	R	
29	Y	HOLD	bw[2]	R	clk	R	
30	Y	SETUP	bw[1]	R	clk	R	
31	Y	HOLD	bw[1]	R	clk	R	
32	Y	SETUP	bw[0]	R	clk	R	
33	Y	HOLD	bw[0]	R	clk	R	
34	Y	SETUP	din[3]	R	clk	R	
35	Y	SETUP	din[3]	F	clk	R	
36	Y	HOLD	din[3]	R	clk	R	
37	Y	HOLD	din[3]	F	clk	R	
38	Y	SETUP	din[2]	R	clk	R	
39	Y	SETUP	din[2]	F	clk	R	
40	Y	HOLD	din[2]	R	clk	R	
41	Y	HOLD	din[2]	F	clk	R	
42	Y	SETUP	din[1]	R	clk	R	
43	Y	SETUP	din[1]	F	clk	R	
44	Y	HOLD	din[1]	R	clk	R	
45	Y	HOLD	din[1]	F	clk	R	
46	Y	SETUP	din[0]	R	clk	R	
47	Y	SETUP	din[0]	F	clk	R	
48	Y	HOLD	din[0]	R	clk	R	
49	Y	HOLD	din[0]	F	clk	R	



## 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 very 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.



## Virtuoso Liberate MX Reference Manual

### Liberate MX Flows

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,	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,	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,	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,	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,	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 generally 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

## Virtuoso Liberate MX Reference Manual

### Liberate MX Flows

---

**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  
-----
```

## 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. When this option is included with a command name, a help message is printed out that lists all currently available arguments for that command. There may be arguments that get printed out when help is used but are not officially supported. The only supported arguments are those that are documented in this manual. When `-help` is used, all other command arguments are ignored.

<b>a...</b>	
<a href="#"><code>add_margin</code></a>	
<b>c...</b>	
<a href="#"><code>char_macro</code></a>	<a href="#"><code>compare_timing</code></a>
<a href="#"><code>char_memory</code></a>	<a href="#"><code>compare_waveforms</code></a>
<a href="#"><code>compare_path</code></a>	
<b>d...</b>	
<a href="#"><code>define_arc</code></a>	<a href="#"><code>define_leakage</code></a>
<a href="#"><code>define_cell</code></a>	<a href="#"><code>define_measure</code></a>
<a href="#"><code>define_duplicate_pins</code></a>	<a href="#"><code>define_memory</code></a>
<a href="#"><code>define_index</code></a>	<a href="#"><code>define_table</code></a>
<a href="#"><code>define_leafcell</code></a>	<a href="#"><code>define_template</code></a>
<b>h...</b>	
<a href="#"><code>hspice_lis_2_waves</code></a>	
<b>i...</b>	
<a href="#"><code>interpolate_define_axis_expr</code></a>	
<b>m...</b>	

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

<u>mxcore def</u>	<u>mx recover setup</u>
<u>mx match node</u>	<u>mx report</u>
<u>mx merge</u>	<u>mx set constprop</u>
<u>mx recover clean</u>	<u>mx set domainprop</u>
<u>mx recover info</u>	<u>mx set ultrasim param</u>
<u>mx recover merge</u>	
<b>r...</b>	
<u>report measure</u>	
<b>s...</b>	
<u>set gnd</u>	<u>set virtual</u>
<u>set vdd</u>	<u>spv tcl 2 waves</u>
<b>v...</b>	
<u>validate macro</u>	<u>validate memory</u>
<u>validate measure</u>	
<b>w...</b>	
<u>write ldb</u>	<u>write verilog</u>
<u>write library</u>	

## **add\_margin**

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

### **Options**

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

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-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 Libearte 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.
- `-charthread <number>`  
Specifies the maximum number of threads to use during characterization. Default: 0 (Number of threads is determined automatically).
- `-ecsm` Characterize ECSM (delay) data.
- `-ecsmn` Characterize ECSMN (noise) data.
- `-part_arc_thread <number>`  
Specifies the 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).

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-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`

<code>aps</code>	Use the Virtuoso Accelerated Parallel Simulator (APS).
<code>xps</code>	Use the XPS simulator.
<code>spectre</code>	Use the Spectre simulator.

`-partthread <number>`

Specifies the maximum number of threads to use during fastsim 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.

`-write_thread <number>`

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

### ***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`)

The different `-thread` arguments of `char_macro` defines 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](#).

The `char_params` argument specifies a list of options that can be provided to Liberate characterization. Valid arguments are `-ccs`, `-ccsn`, `-ecsm`, `-ecsmn`, `-thread`, `-extsim`. For more information on these options, see the `char_macro` command in the Liberate 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 spectre -partthread 2 -charsim spectre -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 -extsim "spectre" -thread 2 \
    -char_params {-extsim "spectre" -thread 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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

```
-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
```

## **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**

<code>-ccs</code>	Enables CCS characterization and library generation.
<code>-ccsn</code>	Enables CCSN characterization and library generation.
<code>-ecsm</code>	Enables ECSM characterization and library generation.
<code>-ecsmn</code>	Enables ECSMN characterization and library generation.
<code>-workdir "string"</code>	The complete path where the <code>mx_setup</code> directory containing the reusable 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**

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

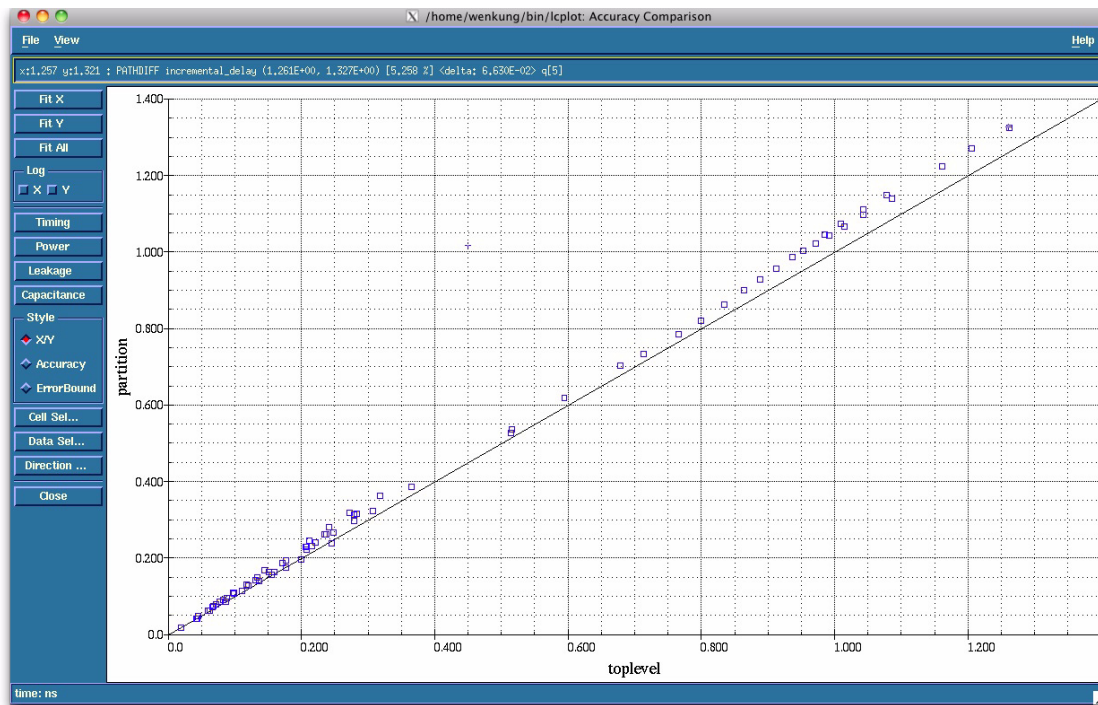
The `sim.top.rpt` file is automatically generated at each run and for each delay partition. It contains the cumulative delay along the path as seen from the top level. The `sim.rpt` is automatically generated at each run and for each partition. It contains the cumulative delay along the path, as seen 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:

# Virtuoso Liberate MX Reference Manual

## Liberate MX Commands





## **compare\_timing**

Generates a report in which top and partition-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**

<code>cell</code>	Specifies the cell name.
<code>-debug &lt; 0   1 &gt;</code>	Enables debug mode and print information. Default: 0
<code>-gui &lt; 0   1 &gt;</code>	Displays generated report using <code>/usr/bin/vim</code> . Default: 0 (only generate the report)
<code>-mx_dir &lt;string&gt;</code>	Specifies the complete path to mx directory. Default: <code>mx_dir</code> .
<code>-nets</code>	Specifies the specific net name to compare timing reports. Default: all nets.
<code>-part &lt;string&gt;</code>	Specifies the individual partition name to compare timing reports. Default: all partitions.
<code>-table_idx</code>	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 very 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**

<code>cell</code>	Specifies the cell name.
<code>-debug &lt; 0   1 &gt;</code>	Enables debug mode and print information. Default: 0
<code>-gui &lt; 0   1 &gt;</code>	Calls ViVA on generated script. Default: 0 (only generate the ocean scripts).
<code>-map &lt;0   1&gt;</code>	Specifies the map names before drawing waveforms.
<code>-mx_dir &lt;string&gt;</code>	Specifies the complete path to mx directory. Default: <code>mx_dir</code> .
<code>-nets</code>	Specifies the specific net name to compare waveforms. Default: all nets.
<code>-part &lt;string&gt;</code>	Specifies the individual partition name to compare waveforms. Default: all partitions.
<code>-table_idx</code>	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 very 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 {altos_clone_arcs <string>}`

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](#).

`-attribute {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
```

`attribute {altos_mx_bundle <bundle_name>::<bundle_criterion>}`

Instructs Liberate MX on how to choose a representative of a specific measurement group before going to full SPICE characterization, where

`<bundle_name>::<bundle_criterion>` means,

`<bundle_name>` == any string

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

All arc definitions of the same arc, with the same string in the `altos_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.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

For example:

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

```
define_measure -name tcycl_1 -trig CKR -trig_val $vc50
-targ $probel_1 -targ_type delay -targ_val $vc90 $cell
```

```
define_measure -name tcycl_2 -trig CKR -trig_val $vc50
-targ $probel_2 -targ_type delay -targ_val $vc10 $cell
```

```
define_measure -name tcycl -trig CKR -trig_val $vc50
-targ_type delay -equations {"tcycl_1-tcycl_2"} $cell
define_arc -type min_period -measure tcycl -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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-attribute {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 actually does switch 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.*)

`-attribute {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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-attribute {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` variable.
- 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}
```

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

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

`-attribute {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.

`-attribute {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 {autoprobing_skip_rel_probe CLK} \  
$cell
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-attribute {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 {autoprobing_skip_rel_probe_regexp BL*  
WORD*} \  
    $cell
```

`-attribute {altos_autoprobing_level <value>}`

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

`-delay_threshold {list}`

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

`-equation <"equation">`

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

Example:

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

`-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.



## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-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 variable `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.

`-measure "name"`

Name of associated `define_measure` command.

`-name "name"`

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

`-pin {pins}`

List of destination pins or buses for the arc. (REQUIRED)

`-pin_dir <R | F>`

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

`-pin_probe_threshold {list}`

List of pin monitor nodes thresholds.

`-probe {names}`

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

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

- `-probe_dir <R | F>` Transition direction (R = rise, F = fall) of probe pin(s). If direction is not specified, arcs are created only for R.
- `-related_pin {pins}` List of related pins or buses for the arc.
- `-related_pin_dir <R | F>`  
Transition direction (R = rise, F = fall) of related pin(s). If direction is not specified, arcs are created only for R.
- `-related_probe {names}`  
List of names of clock nodes to monitor for setup/hold characterization.
- `-related_probe_dir <R | F>`  
Transition direction of related probe pin(s).
- `-related_probe_threshold {list}`  
List of related monitor nodes thresholds.
- `-slew_threshold {list}`  
Four slew measurement thresholds (lower\_rise, upper\_rise, lower\_fall, upper\_fall).

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-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/ghaida/mx/mx_setup/tables/power_0.tbl \  
            /home/ghaida/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.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-type <arc_type>`      Type of arc. This option accepts one of the following values: combinational, hold, max\_clock\_tree\_path, min\_clock\_tree\_path, minperiod, mpw, non\_seq\_hold, non\_seq\_setup, power, retain, or setup. Default: combinational

The following are examples of how this argument can be used with min\_clock\_tree\_path and max\_clock\_tree\_path:

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

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 {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.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

**Note – how it works internally:** The `altos_clone_arcs` attribute is usually derived internally by 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 whole, but separated in to different groups. For example, for constraint arcs on an address bus `A[7:0]`, it may be necessary to group column and row addresses in to separate groups, such as `A[0:2]`, `A[3:6]`, and `A[7]`. The specific grouping can then be specified in the `define_arc` itself and it will be respected when the `clone_attribute` attribute is generated. For this specific example, you would issue three separate `define_arc` commands as shown below:

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

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

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

#### **Important**

Use one arc characterization value for another arc.

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

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

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

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

```
}

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

## Examples of define\_arc

### **Example 1**

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

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

```
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>/philwx* 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 "" \
```



## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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>/philwx" \  
  -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>/philwx" \  
  -targ_type "delay" \  
  -targ_dir  fall \  
  -targ_val  0.72 \  
  -targ_d    "" \  
  -failed_val ""
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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`:

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

```
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**

- |  |   |
|--|---|
| <code>-async {pin_names}</code>                      | List of asynchronous pin names. The <code>-async</code> option specifies pins that require recovery, removal, <code>non_seq_setup</code> , or <code>non_seq_hold</code> timing arcs.  |
| <code>-clock {pin_names}</code>                      | List of clock pin names. For more information, see <a href="#">Using the -input, -output, and -clock options</a> .  |
| <code>-constraint &lt;name&gt;</code>                | 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 <a href="#">define_template</a> command.                              |
| <code>-delay &lt;name&gt;</code>                     | 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 <a href="#">define_template</a> command. |
| <code>-ignore_input_for_auto_cap {pin_names}</code>  | Ignore arcs originating from the specified pin when calculating <code>auto_index</code> and <code>auto_max_capacitance</code> .   |
| <code>-ignore_output_for_auto_cap {pin_names}</code> | Ignore arcs to the specified output pin when calculating <code>auto_index</code> and <code>auto_max_capacitance</code> .  |
| <code>-ignore_pin_for_ccsn {pin_names}</code>        | 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.   |
| <code>-input {pin_names}</code>                      | List of input pin names. For more information, see <a href="#">Using the -input, -output, and -clock options</a> .  |

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

<code>-mxcore {core_files}</code>	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 <a href="#">Appendix B, “Specifying Memory Core Cells.”</a>
<code>-output {pin_names}</code>	List of output pin names. For more information, see <a href="#">Using the -input, -output, and -clock options.</a>
<code>-power &lt;name&gt;</code>	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 <a href="#">define_template</a> command.
<code>-when &lt;"function"&gt;</code>	Specifies user-defined cell-level logic constraints using the Liberty format <i>when</i> syntax, constraining the automatic vector generation of Liberate for the given cell. The <code>define_cell -when</code> 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 <code>define_arc</code> to specify simultaneous switching inputs, or specify a truth table to be translated automatically.
<code>{cell_names}</code>	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**

<code>&lt;cellname&gt;</code>	The cell name to apply the duplication to.
<code>&lt;pin&gt;</code>	The pin to be duplicated.
<code>{duplicate_pins}</code>	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 `addrB[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 on the fly 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`.
- `-pin {pins}` List of pin names (REQUIRED). This option accepts wildcards.
- The following examples illustrate the use of an asterisk "\*" wildcard for pin names:
- ```
define_index -pin D* # All pins beginning with "D"
define_index -pin *  # All pins
```
- `-related_pin {pins}` List of related pin names. This option accepts wildcards.
- Note:** `-related_pin` is required for most arcs, except for arcs that need only one pin, such as hidden power arcs and MPW or `min_period` arcs.
- `-type {data_types}` List of data types: `constraint`, `delay`, `mpw`, `power`, or `retain`.
- `-when <"function">` Logic state of side inputs.
- `{cell_names}` List of cell names. This option accepts wildcards.

### Important Points to Note

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



## Virtuoso Liberate MX Reference Manual

### Liberate MX 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

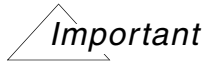
```
define_index \  
  -pin {data_out<31:0>} \  
  -related_pin {clk_in} \  
  -type delay \  
  -index_2 {0.05 0.01 0.5 } \  
rf_cell
```

## define\_leafcell

Defines the level of hierarchy that resides at the bottom of a cell-level netlist. This allows Liberate 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 variable to enable external simulation with the process models and the compiled netlist. It supports identification of mosfets, diodes, resistors, and capacitors.

### Options

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



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

### Examples

#### ■ Mosfets

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

#### ■ Resistors

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

#### ■ Diodes

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

#### ■ Capacitors

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

## define\_leakage

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

### Options

`-ignore_for_default` Instructs that the specified leakage should not be considered in any calculations when computing default `leakage_power` groups or attributes. (See `set_default_group`)

For example:

```
...
read_ldb My.ldb.gz
define_leakage -when "a*!b" -ignore_for_default MyCell
write_library My.lib
```

**Note:** This option must be set in a `define_leakage` command before the `write_library` command.

`-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.

**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.

### 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>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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:

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

`-targ {signal_name}`

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

`-targ_d <name | #_of_cycles>`

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

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

`name` Specifies the name of a measurement.

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

`-targ_dir <rise | fall>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>rise</code> | Specifies that the target signal direction is rising.  |
| <code>fall</code> | 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`

|                                          |                                                                         |
|------------------------------------------|-------------------------------------------------------------------------|
| <code>first</code>                       | Specifies that the first edge of the target signal is to be considered. |
| <code>last</code>                        | Specifies that the last edge of the target signal is to be considered.  |
| <code><i>pos_integer_signal_#</i></code> | 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`

|                              |                                                                               |
|------------------------------|-------------------------------------------------------------------------------|
| <code><i>measure</i></code>  | Specifies the name of a measurement.                                          |
| <code><i>#_in_sec</i></code> | Specifies the number of seconds to shift the resulting target crossing point. |

`-targ_type <delay | voltage>`

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

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>delay</code>   | Specifies that a delay measurement is to be used.   |
| <code>voltage</code> | Specifies that a voltage measurement is to be used. |

`-targ_val <voltage>`

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

`-targ_val_reuse_factor <value>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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

`-trig {signal_name}`

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

`-trig_d <measure | #_of_cycles>`

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

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

`measure` Specifies the name of a measurement.

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

`-trig_dir <rise | fall>`

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

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

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

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

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

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

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

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

`-trig_end {signal_name}`



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

The `-trig_start` and `-trig_end` options

- Must be used together.
- Cannot be used with `-trig` (if they are, `-trig` is ignored.)
- The only supported measurement types (`-targ_type`) are the minimum and maximum voltage.

`-trig_end_d <measure | #_of_cycles>`

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

*measure* Specifies the name of a measurement.

*#\_of\_cycles* Specifies the number of cycles to wait before considering the trigger.

`-trig_end_dir <rise | fall>`

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

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

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

`-trig_end_e <first | last | # >`

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

*first* Specifies that the first edge of the end trigger signal is to be considered.

*last* Specifies that the last edge of the end trigger signal is to be considered.

*#* Specifies the number of the edge of the end trigger signal to be considered.

`-trig_end_s <measure | #_in_sec>`

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

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

*measure* Specifies the name of a measurement.

*#\_in\_sec* Specifies the number of seconds to wait before considering the trigger.

`-trig_end_val <voltage>`

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

`-trig_end_val_reuse_factor <value>`

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

`-trig_s <measure | #_in_sec>`

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

*measure* Specifies the name of a measurement.

*#\_in\_sec* Specifies the number of seconds to shift the resulting target time point.

`-trig_start {signal_name}`

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

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

- must be used together.
- cannot be used with `-trig` (if they are, `-trig` is ignored.)
- The only supported measurement types (`-targ_type`) are the minimum and maximum voltage.

`-trig_start_d <measure | #_of_cycles>`

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

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

*measure* Specifies the name of a measurement.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

*#\_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>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

(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` variable.

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

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

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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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

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

#.meas tran sigb trig v(st:1) val=0.45 td=tc4 fall=1
#+targ v(buf:1) val=0.45 td=tc4 fall=1
#.meas tran sigb_final param='max(sigb,0)'
# sigb actually switches in both cycles; force it to
# take the first falling transition for the target
define_measure \
  -name sigb \
  -trig {st:1} \
  -trig_dir fall \
  - 0.45 \
  -targ "buf:1" \
  -targ_dir fall \
  -targ_val 0.45 \
  -targ_edge first \
  $cell

#.meas tran reset param='s
ig1-sigb_final-sig2'
define_measure \
  -name reset \
  -keep min \
  -duration 2 \
  -trig {clk} \
  -trig_dir rise \
  - 0.45 \
  -targ_d 1 \
  -equations { \
    "sig1 - sigb - sig2" \
    "(sig1 - sigb)*0.9 - sig2 *1.1" \
    "100 * (sig1 - sigb - sig2)/ (sig1 - sigb + sig2) " \
    "100 * ((sig1 - sigb)*0.9 - sig2 *1.1) /(sig1 - sigb + sig2) " \
  } \
  $cell

define_arc -pin {clk} -measure reset $cell
```

## **define\_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 "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"`



## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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>

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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` and `spectre`.  
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 "string"`

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>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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.

`-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>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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

`-expand_buses <string (no)>`

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}`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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}`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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}`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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)`

Skip reading 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.



## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

`-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)`

Enable `-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_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 generated files and rerun 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. An example 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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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\_table**

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

### **Options**

|                                     |                                        |
|-------------------------------------|----------------------------------------|
| <code>mxtables {table_files}</code> | List of vector table files. (REQUIRED) |
| <code>cell &lt;name&gt;</code>      | 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 | ecdm | mpw}`

The type of template being defined. (REQUIRED)

How each cell is to be characterized is defined by associating the defined template with the appropriate option of the `define_cell` command. Multiple `define_cell` commands can reference a single template.

See also [Uses of different template types](#).

`-index_1 {values}` List values to be used as the first index. (REQUIRED)

`-index_2 {values}` List values to be used as the second index.

`-index_3 {values}` List values to be used as the third index.

`<template>` Name of the template.

This command must be used before the `char_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)         |
| capacitance | 1pf (pico farad)   |



### ***Uses of different template types***

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

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

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

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

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

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

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



#### *Tip*

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

### Examples

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

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

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

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

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

### ***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.

## **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**

|                                 |                                                                                                                                                                                                                                                       |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lis &lt;file&gt;</code>   | The HSpice <code>.lis</code> filename.                                                                                                                                                                                                                |
| <code>-nets &lt;list&gt;</code> | List of nodes to show.<br><br>Specify the <code>-nets &lt;list&gt;</code> to limit the waveform display to a select number of nets. If <code>-nets</code> option is not specified, all nodes available for display in the <code>lwave</code> 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

|                                   |                                                                                                                                                                                                         |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-expression "string"</code> | Multiplication expression of compiler axis variables, <code>bit*bit</code> or <code>bit*word</code> .                                                                                                   |
| <code>-type "string"</code>       | A list of types that use the axis variable for interpolation.<br><br>Valid types include:<br><br><code>all cap delay hold leakage power mpw minp retain</code><br><code>retain_trans setup trans</code> |
| <code>name "string"</code>        | The interpolate axis name.                                                                                                                                                                              |

## **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>
```

# Virtuoso Liberate MX Reference Manual

## Liberate MX Commands

### mx\_merge

Merges all LDB data to create top-level LDB.

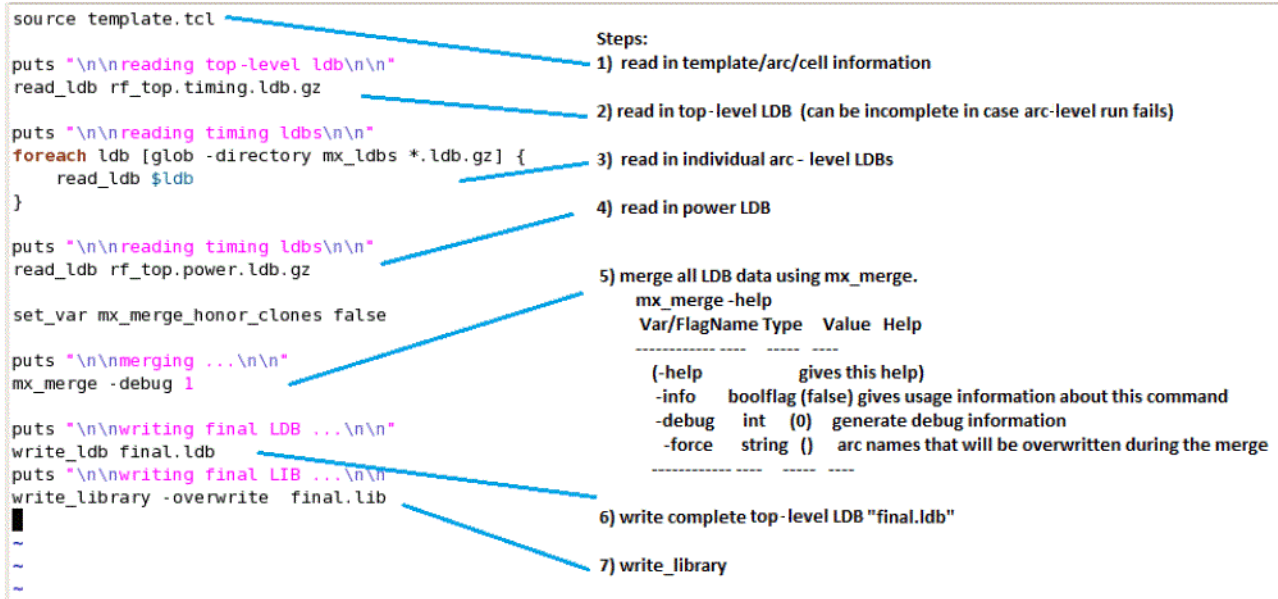
#### Options

- 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\_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**

|                                     |                                                                                                                |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>-ldb {list}</code>            | Paths to mx reference/compare ldb's - as written by <code>write_ldb</code> command in respective runs.         |
| <code>-lib {list}</code>            | Paths to MX reference/compare libraries - as written by <code>write_library</code> command in respective runs. |
| <code>-rpt &lt;file_name&gt;</code> | 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\_set\_constprop**

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

### **Options**

|                                          |                                                                                                                                                                                        |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;node&gt; &lt;value&gt;}</code> | Lists nodes and their logic values, where to start constant propagation from. The <code>node</code> 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 {\n    {A stop CLK} {B start CLK} \n    {C force CLK} {D enable CLK} \n}
```

## **mx\_set\_ultrasim\_param**

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

**Options.**

|                           |                                                                                                                         |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;list&gt;</code> | 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 { \n    {simplereset 5} \n    {pn_level 5} \n    {cgnd 1e-15} \n    {sfe_compaction 0} \n    {keepparaname 0} \n    {rshort 2} \n    {hier_delimiter .} \n    {dc_turbo 3} \n    {rcr_fmax 1G} \n}
```

## **report\_measure**

Reports the difference between relax or stress simulation results.

## Options

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

## **set\_gnd**

Identifies the names of ground nodes.

### **Options**

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-ignore_power</code>          | Ignore the power contribution from this supply net.<br><br>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.                                                                                                                                                                                                                                  |
| <code>-no_model</code>              | Request to not include this supply in the output <code>.lib</code> .                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>-virtual</code>               | Treat the net as logic 0 for static analysis, but as a regular node for dynamic simulation.<br><br>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 <a href="#">mx_find_virtual_rails</a> to instruct the tool on how to report design nodes that should be defined as virtual. |
| <code>-waveform &lt;name&gt;</code> | Provide a file (full path name) containing a text description of a waveform as a time/value pair list.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>&lt;net_name&gt;</code>       | Name of ground supply net.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>&lt;voltage&gt;</code>        | 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 easily, 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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

```
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

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-ignore_power</code>          | Ignore the power contribution from this supply net.<br><br>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.                                                                                                                                                                                                                                  |
| <code>-no_model</code>              | Request to not include this supply in the output <code>.lib</code> . See <a href="#">set_gnd</a> for related information.                                                                                                                                                                                                                                                                                                                                                      |
| <code>-virtual</code>               | Treat the net as logic 0 for static analysis, but as a regular node for dynamic simulation.<br><br>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 <a href="#">mx_find_virtual_rails</a> to instruct the tool on how to report design nodes that should be defined as virtual. |
| <code>-waveform &lt;name&gt;</code> | Provide a file (full path name) containing a text description of a waveform as a time/value pair list.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>&lt;net_name&gt;</code>       | Name of power supply net.                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>&lt;voltage&gt;</code>        | 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\_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 variable.

### **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 by the FastSPICE simulation (when `mx_spv_api` is 1).

## **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 D, "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 E, “Basic Flow for Validating User-Defined Criteria.”](#)

## **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**

|                               |                                                                                                                                    |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;filename&gt;</code> | A library database file in LDB format.                                                                                             |
| <code>-overwrite</code>       | Disables the automatic version control, and if the output library already exists, it is overwritten.                               |
| <code>-rename</code>          | 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)
- 2: Include the rise and fall capacitance ranges where both range limits are both set to the rise/fall capacitance attribute values:

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

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

`-ccs`

Include CCS data.

`-ccsn`

Include CCSN (noise) data.

`-cells {cell_names}`

List of cell names. Default: *all cells* This option supports the use of a wildcard.

`-dcnoise_abstol <tolerance>`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

- `-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, Variety and none for other products)
- The following are the supported values:
- `none`: Does not remove any failed data. See the variables [`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" | ""}`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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`.

`-user_data <filename>`



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

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

`-user_data_override <filename>`

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

`<libname>`

The output library name.

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

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

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

## Bus Support

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

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

## Bit-Level Delay Modeling

By default, for an arc involving a bus, only the *worst case* representative is chosen for characterization. The worst case values are then applied to all elements of the bus. This is controlled by adding the `-attribute altos_clone_arcs` option to the `define_arc` command. You can directly set these attributes or can indirectly control them through the syntax shown below.

### ■ Worst-case determined using all elements of a bus (default)

The worst case is determined from the full bus, whether it is specified as a full range of bits ***or*** as single bits.

Full range:

```
define_arc -type <...> -pin bus[MSB:0] ...
```

Single bits:

```
define_arc -type <...> -pin bus[MSB] ...
define_arc -type <...> -pin bus[MSB-1] ...
...
define_arc -type <...> -pin bus[0] ...
```

### ■ Worst-case determined within a range of bits

The worst case is calculated for the bits within the specified ranges, and applied to all bits in that range. In the example below, the worst case will be applied to bits in the range `R0 : R1` separately from bits in range `R2 : R3`.

```
define_arc -type <...> -pin bus[R0:R1] ...
define_arc -type <...> -pin bus[R2:R3] ...
```

### ■ No worst-case; each bit considered separately

All bits of the bus are characterized separately (no worst-casing). This is achieved by specifying "single bit ranges", as shown below.

```
define_arc -type <...> -pin bus[MSB:MSB] ...  
define_arc -type <...> -pin bus[1:1] ...  
define_arc -type <...> -pin bus[0:0] ...
```

**Note:** When separate ranges are specified, only a fully-expanded (bit-blasted) library is able to properly represent the different values for the different bits (`write_library -expand_buses`). If you instead chose to generate a fully-compressed library (default in `write_library`) a worst-casing step will be done at the LDB level prior to generating the library. Therefore, if both an expanded and a bit-blasted library needs to be generated, the bit blasted should be generated first.

## write\_verilog

Creates a Verilog file for the current library. The Verilog content is written to the given `<verilog_filename>`. See also [Additional Tcl Variables to Control Verilog Output](#).

**Note:** A `.v` suffix is added to filenames that do not end in `.v`.

### Options

- `-cells {cell_names}` Controls which cells should be written into the output file.  
Default: write all cells
- Note:** If the `-exclude` option is also set, only the cells not listed in the `-cells` list will be output. This option supports the use of a wildcard.
- `-delayed` Controls the naming convention for creating "delayed" signals.  
Default: "delayed\_%P" (where %P is the pin name.)
- When using user-data with `write_verilog`, it is necessary to match these delayed signals with the equivalent signals used in the user-provided function description. By default, delayed output signals are created for the signals passed to timing checks such as `setup-hold` and/or `recrem` in Verilog.
- For more details, see [Using the -delayed option](#).
- `-exclude` Exclude cells from `-cells` list.
- `-fwire_prefix <"prefix">` Prefix for internal wires for pin functions.  
Default: "int\_fwire\_"
- `-indent <number>` Specifies the number of spaces to use for indentation.  
Default: use tab
- `-merge` Includes the cell modules not specified in the library, but present in the `user_data` file.
- `-mpw_include_output_state`

Includes output pin logic state in MPW timing checks for "clear" or "preset" input signals.

`mpw_include_output_state` requires that the `sdf_cond_style` variable is set to 1. If not set, it will force the setting. This variable should be set prior to creating an equivalent library (`.lib`) with `write_library` to ensure consistency between the library and the Verilog file; otherwise, there might be warnings during SDF back-annotation.

The `-mpw_include_output_state` option should be used before the `read_library`, `char_memory`, or `read_ldb` command. When this variable is used, the MPW check (`$width`) is checked by Verilog only when the output pin of the cell is high for clear inputs and low for preset inputs. The Verilog file will contain extra "timing" gates to add the logic necessary to "and" the logic state of the output pin to logic representing the "when" condition given in the library for each `min_pulse_width` timing arc.

`-mux <type>`

Creates MUX user-defined primitives (UDPs) for MUX functions.

Default: use the basic logic primitives

**Note:** The `-mux` option converts the pins whose functions are a 2x1 or 4x1 MUX into a pre-defined UDP named `altos_mux2` and `altos_mux4` respectively.

`-no_edge`

Exclude 'posedge' or 'negedge' on edge triggered arcs, default is to include edges.

`-path <path>`

String used to denote a path, for example, "`=>`" and "`*>`".

Default: "`=>`"

`-sdf_version <version>`

Controls the format of the output Verilog for use with SDF annotation. The version must be 2.1 or 3.0.

Default: 3.0

Set to 3.0 to generate a Verilog file that is compatible with SDF version 3.0. SDF version 3.0 permits `recrem` constructs in the Verilog file to represent recovery and removal of timing constraints.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

|                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-specparams</code>                  | Causes delay assignments in the Verilog file to be assigned to <code>specparam</code> variables rather than directly to values. The <code>-path</code> option controls the delimiter used for delay assignments, that is, <code>=&gt;</code> or <code>*&gt;</code> .                                                                                                                                                                                                                                                                                                                                    |
| <code>-split_notifier</code>              | When writing verilog modules for multi-bit cells, it is required to output separate notifier commands for each DFF. The <code>-split_notifier</code> option is used to output separate notifier commands for each DFF.                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>-timescale &lt;timescale&gt;</code> | Verilog timescale. (1ns/10ps)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>-twire_prefix &lt;prefix&gt;</code> | <p>Prefix for internal wires of conditional timing constraint functions. Default: <code>int_twire_</code></p> <p><b>twire_prefix</b> is the prefix used for internal wires created when generating additional functions for state dependent timing constraints. The <code>fwire_prefix</code> is the prefix used for internal wires created when generating logic functions. The <code>udp_prefix</code> is the prefix used for user defined primitives that are created for latches and/or flip-flops. Set <code>udp_prefix</code> to a null string to exclude generating user defined primitives.</p> |
| <code>-udp_prefix &lt;prefix&gt;</code>   | <p>Prefix for built-in user defined primitives (UDPs). Set to "" to exclude UDPs.</p> <p>Default: <code>altos_</code></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>-user_data &lt;filename&gt;</code>  | <p>User Verilog file to merge timing info with.</p> <p>Specifies a user-provided Verilog file to merge the generated Verilog data with. If a <code>user_data</code> 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 <code>user_data</code> file, a complete Verilog file is written including function descriptions.</p>                                                     |
| <code>&lt;verilog_filename&gt;</code>     | Output Verilog filename.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

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:**

## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

```
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



## Virtuoso Liberate MX Reference Manual

### Liberate MX Commands

---

|                                        |                                                                                                                                                                               |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>verilog_delay_Zvalue</code>      | The delay value for tristates. Default: 0                                                                                                                                     |
| <code>verilog_delay_clk2q_value</code> | The delay value for clock to Q arcs on sequential cells.<br>Default: 0                                                                                                        |
| <code>verilog_IQ</code>                | The name map for the internal state of flip-flops (such as, IQ) to the Verilog state function.                                                                                |
| <code>verilog_IQN</code>               | The name map for the internal state of flip-flops (such as, IQN) to the Verilog state function.                                                                               |
| <code>verilog_start_skip</code>        | 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"  |
| <code>verilog_stop_skip</code>         | 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" |

# **Virtuoso Liberate MX Reference Manual**

## **Liberate MX Commands**

---

## Liberate MX Variables

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

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

|                                                                                                                           |                                                                                                                           |
|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>c...</b>                                                                                                               |                                                                                                                           |
| <u>constraint_glitch_peak</u>                                                                                             | <u>constraint_probe_lower_rise</u><br><u>constraint_probe_hold_lower_rise</u><br><u>constraint_probe_setup_lower_rise</u> |
| <u>constraint_glitch_peak_max</u>                                                                                         | <u>constraint_probe_upper_fall</u><br><u>constraint_probe_hold_upper_fall</u><br><u>constraint_probe_setup_upper_fall</u> |
| <u>constraint_glitch_peak_mode</u>                                                                                        | <u>constraint_probe_upper_rise</u><br><u>constraint_probe_hold_upper_rise</u><br><u>constraint_probe_setup_upper_rise</u> |
| <u>constraint_probe_lower_fall</u><br><u>constraint_probe_hold_lower_fall</u><br><u>constraint_probe_setup_lower_fall</u> |                                                                                                                           |
| <b>d...</b>                                                                                                               |                                                                                                                           |
| <u>debug_flow</u>                                                                                                         |                                                                                                                           |
| <b>e...</b>                                                                                                               |                                                                                                                           |
| <u>extsim_deck_include</u>                                                                                                | <u>extsim_model_include</u>                                                                                               |
| <b>f...</b>                                                                                                               |                                                                                                                           |
| <u>fastsim_cmd</u>                                                                                                        | <u>fastsim_cmd_option</u>                                                                                                 |
| <b>k...</b>                                                                                                               |                                                                                                                           |
| <u>keep_empty_cells</u>                                                                                                   |                                                                                                                           |

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

|                                         |                                   |
|-----------------------------------------|-----------------------------------|
| <b>l...</b>                             |                                   |
| <u>lic max timeout</u>                  | <u>lic queue timeout</u>          |
| <b>p...</b>                             |                                   |
| <u>parse auto define leafcell</u>       |                                   |
| <b>s...</b>                             |                                   |
| <u>set var failure action</u>           |                                   |
| <b>v...</b>                             |                                   |
| <u>virtual rail waveform probe</u>      |                                   |
| <b>mx...</b>                            |                                   |
| <u>mxtable dontcare value</u>           | <u>mxvw min glitch peak</u>       |
| <b>mx_a...</b>                          |                                   |
| <u>mx active fanout channel include</u> | <u>mx auto char params</u>        |
| <u>mx active load</u>                   | <u>mx automeas filter</u>         |
| <u>mx active load channel thresh</u>    | <u>mx autoprobing hold level</u>  |
| <u>mx active load gate thresh</u>       | <u>mx autoprobing setup level</u> |
| <u>mx arc report</u>                    |                                   |
| <b>mx_b...</b>                          |                                   |
| <u>mx bisection</u>                     | <u>mx bitline probe threshold</u> |
| <b>mx_c...</b>                          |                                   |
| <u>mx char bundle size</u>              | <u>mx clone if uda</u>            |
| <u>mx char virtual as rail</u>          | <u>mx const prop</u>              |
| <u>mx check arcs</u>                    | <u>mx constraint ocv factor</u>   |
| <u>mx check arcs exit on missing</u>    | <u>mx corecell</u>                |
| <u>mx clock2clock constraints</u>       | <u>mx create if dynamic</u>       |
| <u>mx clk2clk mode</u>                  | <u>mx create if uda</u>           |
| <u>mx clock tree report</u>             |                                   |
| <b>mx_d...</b>                          |                                   |
| <u>mx debug</u>                         | <u>mx distributed sim</u>         |
| <u>mx defmem sdf cond</u>               | <u>mx domain propagation</u>      |

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

|                                                                |                                                                                                                                      |
|----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <u>mx_delay_ocv_factor</u>                                     | <u>mx_dpartition_inactive_tie</u>                                                                                                    |
| <u>mx_dir</u>                                                  | <u>mx_dynamic_include_full_core</u>                                                                                                  |
| <b>mx_f...</b>                                                 |                                                                                                                                      |
| <u>mx_fastsim_auto_ic</u>                                      | <u>mx_find_memcores</u>                                                                                                              |
| <u>mx_fastsim_clock_slew</u>                                   | <u>mx_find_stack_loads</u>                                                                                                           |
| <u>mx_fastsim_input_slew</u>                                   | <u>mx_find_virtual_rails</u>                                                                                                         |
| <u>mx_fastsim_load</u>                                         | <u>mx_fix_pin_vdd</u>                                                                                                                |
| <u>mx_fastsim_reuse</u>                                        | <u>mx_full_rail_tol</u>                                                                                                              |
| <u>mx_find_arrays</u>                                          | <u>mx_fullsim_measurement</u>                                                                                                        |
| <u>mx_find_memcore_numbit_threshold</u>                        |                                                                                                                                      |
| <b>mx_g...</b>                                                 |                                                                                                                                      |
| <u>mx_greybox</u>                                              | <u>mx_greybox_constraint_method</u>                                                                                                  |
| <b>mx_i...</b>                                                 |                                                                                                                                      |
| <u>mx_inputcap_ldb_reuse</u>                                   |                                                                                                                                      |
| <b>mx_l...</b>                                                 |                                                                                                                                      |
| <u>mx_ldbs_reuse</u>                                           |                                                                                                                                      |
| <b>mx_m...</b>                                                 |                                                                                                                                      |
| <u>mx_margin_report</u>                                        | <u>mx_mpw_allow_same_probe_on_both_rise_and_fall_clock_tree</u>                                                                      |
| <u>mx_mcf</u>                                                  | <u>mx_mpw_false_probe_delay_threshold</u>                                                                                            |
| <u>mx_measure_error_file</u>                                   | <u>mx_mpw_measurement_duration</u>                                                                                                   |
| <u>mx_min_clock_tree_mode</u><br><u>mx_max_clock_tree_mode</u> | <u>mx_mpw_mode</u>                                                                                                                   |
| <u>mx_min_period_latch_component_mode</u>                      | <u>mx_mpw_mode</u>                                                                                                                   |
| <u>mx_min_period_mode</u>                                      | <u>mx_mpw_probe</u>                                                                                                                  |
| <u>mx_minp_single_slew</u>                                     | <u>mx_mpw_probe_lower_fall</u><br><u>mx_mpw_probe_lower_rise</u><br><u>mx_mpw_probe_upper_fall</u><br><u>mx_mpw_probe_upper_rise</u> |
| <u>mx_monitor_memcore</u>                                      | <u>mx_mxtable_interpret_read_write_cycle_keywords</u>                                                                                |

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

|                                                                                    |                                                         |
|------------------------------------------------------------------------------------|---------------------------------------------------------|
| <u>mx_monitor_memcore_lprobe_level</u>                                             |                                                         |
| <b>mx_n...</b>                                                                     |                                                         |
| <u>mx_negedge_clock</u>                                                            | <u>mx_noise_ldb_reuse</u>                               |
| <b>mx_o...</b>                                                                     |                                                         |
| <u>mx_output_require_fullrail_switch</u>                                           |                                                         |
| <b>mx_p...</b>                                                                     |                                                         |
| <u>mx_partition_name_use_arc</u>                                                   | <u>mx_power_ldb_reuse</u>                               |
| <u>mx_pathdelay_hold_clock_margin</u>                                              | <u>mx_power_single_point</u>                            |
| <u>mx_pathdelay_hold_data_margin</u>                                               | <u>mx_preprocess</u>                                    |
| <u>mx_pathdelay_setup_clock_margin</u>                                             | <u>mx_probe_peak_currents</u>                           |
| <u>mx_pathdelay_setup_data_margin</u>                                              | <u>mx_probes_report</u>                                 |
| <u>mx_pincap_char</u>                                                              | <u>mx_process_probe_relprobe_intersection</u>           |
| <u>mx_posedge_clock</u>                                                            | <u>mx_process_probe_relprobe_intersection_max_paths</u> |
| <u>mx_power_assign</u>                                                             | <u>mx_process_probe_relprobe_intersection_max_depth</u> |
| <u>mx_power_divide_num_switching_mode</u>                                          |                                                         |
| <b>mx_r...</b>                                                                     |                                                         |
| <u>mx_read_spice_exit_on_missing_file</u>                                          | <u>mx_retaining_time</u>                                |
| <u>mx_remove_false_ic_group</u>                                                    | <u>mx_ring_large_ccc_max_paths</u>                      |
| <u>mx_remove_rc_pincap</u>                                                         | <u>mx_ring_large_ccc_max_path_depth</u>                 |
| <u>mx_remove_rc_timing</u>                                                         | <u>mx_ring_model_fold</u>                               |
| <b>mx_s...</b>                                                                     |                                                         |
| <u>mx_seq_probing</u>                                                              | <u>mx_skip_autoprobing</u>                              |
| <u>mx_setup_seq</u> <u>mx_hold_seq</u><br><u>mx_setup_comb</u> <u>mx_hold_comb</u> | <u>mx_skip_print</u>                                    |
| <u>mx_simulation_interval</u>                                                      | <u>mx_spv_api</u>                                       |
| <b>mx_t...</b>                                                                     |                                                         |
| <u>mx_timing_ldb_reuse</u>                                                         | <u>mx_total_active_load_channel_thresh</u>              |
| <u>mx_timing_report</u>                                                            | <u>mx_total_active_load_gate_thresh</u>                 |

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

|                                              |                                                       |
|----------------------------------------------|-------------------------------------------------------|
| <u>mx_timing_report_debug</u>                |                                                       |
| <b>mx_v...</b>                               |                                                       |
| <u>mx_verbose</u>                            | <u>mx_virtual_rail_opposite_device_minimum_factor</u> |
| <u>mx_virtual_rail_auto_mode</u>             | <u>mx_virtual_rail_minimum_xtrs</u>                   |
| <b>mx_w...</b>                               |                                                       |
| <u>mx_whitebox_active_coupling_threshold</u> | <u>mx_whitebox_model_file</u>                         |
| <u>mx_whitebox_active_wire_threshold</u>     |                                                       |
| <b>mx_w...</b>                               |                                                       |
| <u>mx_zip_partition_deck</u>                 |                                                       |
| <b>v...</b>                                  |                                                       |
| <u>variation_dominant_xtr_ccc_abstol</u>     |                                                       |

## **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 variable 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 variable. If both this variable and the `set_constraint_criteria` are used, the last one executed sets the value to be used by Liberate MX.

This variable 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 variable times vdd, then the threshold is limited to the voltage represented by the ratio of vdd specified by this variable. This can result in a search bound error.

This variable 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                      Don't 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 variable must be used before `char_macro`.

**`constraint_probe_lower_fall`**  
**`constraint_probe_hold_lower_fall`**  
**`constraint_probe_setup_lower_fall`**

<value>                      Specifies the probe lower fall threshold. Default: 0.3

The setup/hold-specific values are used during path-delta constraint characterization and to override the global values. Valid values are between .02 and .98.

**constraint\_probe\_lower\_rise**  
**constraint\_probe\_hold\_lower\_rise**  
**constraint\_probe\_setup\_lower\_rise**

<value>                      Specifies the probe lower rise threshold. Default: 0 . 3

The setup/hold-specific values are used during path-delta constraint characterization and to override the global values. Valid values are between .02 and .98.

**constraint\_probe\_upper\_fall**  
**constraint\_probe\_hold\_upper\_fall**  
**constraint\_probe\_setup\_upper\_fall**

<value>                      Specifies the probe upper fall threshold. Default: 0 . 7

The setup/hold-specific values are used during path-delta constraint characterization and to override the global values. Valid values are between .02 and .98.

**constraint\_probe\_upper\_rise**  
**constraint\_probe\_hold\_upper\_rise**  
**constraint\_probe\_setup\_upper\_rise**

<value>                      Specifies the probe upper rise threshold. Default: 0 . 7

The setup/hold-specific values are used during path-delta constraint characterization and to override the global values. Valid values are between .02 and .98.

**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 variable 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 2x2 or 1x1.

Example:

```
set_var debug_flow 2x2
```

This variable must be set before the first `define_template` command.

## **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 <code>.inc</code> 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.<br>Default: Use flattened models. |
|---------|------------------------------------------------------------------------------------------------|

Use this variable 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 variable 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>
```

Note that, 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 variable 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.

## **keep\_empty\_cells**

|                             |                                                                                                                                                       |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;0   1 &gt;</code> | Determines how to handle a cell when the netlist is empty. Default: 0 (Treat empty cells as an error condition.)                                      |
| 0                           | Generates an error condition when an empty cell subcircuit is found. The cell will not be included in the <code>.lib</code> file. (Default)           |
| 1                           | Generates a warning condition when an empty cell subcircuit is found. The cell is included in the <code>.lib</code> file, but does not have any data. |

A cell is available for characterization when all of the following conditions are true:

- There is a `define_cell` command and a netlist that has been read in using the `read_spice` command.

- The `char_library` command does not have the `-cells` argument, or the `-cells` argument includes the cell name and the `-exclude` argument is not used.

This variable must be used before `char_library`.

## **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 variable only works when the shell environment variable `ALTOS_QUEUE` is set to 1. See the [Waiting for Available License](#) section of Chapter 3, “Getting Started with Liberate MX” for more details about how this variable works.

The shell environment variable `ALTOS_LIC_MAX_TIMEOUT` overrides the value set by this variable in the `Tcl` file. For more information, see [ALTOS\\_LIC\\_MAX\\_TIMEOUT](#).

This variable 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 shell environment variable `ALTOS_LIC_CHECK_ALT_TIMEOUT` overrides the value set by this variable in the `Tcl` file. For more information, see [ALTOS\\_LIC\\_CHECK\\_ALT\\_TIMEOUT](#).

This variable must be used before `char_macro`.

## **mxtable\_dontcare\_value**

{`<bus_name | pin_name>` `<0 | 1>`} Sets a default table entry for pin or bus. Default: none

This variable sets a default table entry value for a pin or bus. Variable 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 variable 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 to 100 percent. This variable is applied on glitch above gnd, glitch under vdd, undershoot under gnd, and overshoot over vdd.

## **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 variable 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 variable 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 really 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.<br>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.<br>Default: 1 |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## **mx\_arc\_report**

|            |                                                                             |
|------------|-----------------------------------------------------------------------------|
| <filename> | Specifies file where failed arcs are reported. Default:<br>"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 variables specified in the main script directly to characterization phase.  
Default: 0

0 Don't pass through any commands or variables. (Default)

1 Pass-through commands and variables.

Commands passed through are:

- `define_template`
- `define_leafcell`

All variables are passed through except:

- variables specific to Liberate MX (variables that begin with "mx\_")
- variables 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\_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 variable 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;
```

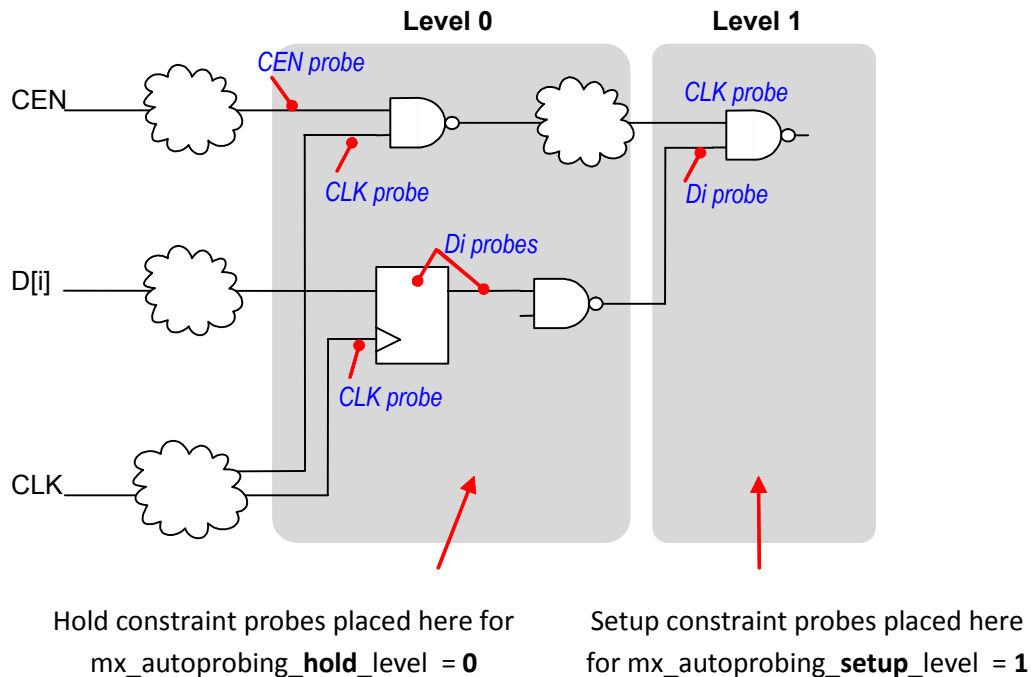
## **mx\_autoprobing\_setup\_level**

`<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;
```



## 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 actually 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 variable 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 variable 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.

Note that regardless of this variable, 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**

`<string>` Controls the behavior of virtual rails in dynamic partition and characterization. Default: `all`

This variable controls the behavior of virtual rails in dynamic partition and characterization. This variable 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 variable `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 >                      Terminate execution if there's a mismatch between arcs specified by the user and arcs found by partitioning. Default: 0 (don't 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<br>  none> | Controls the mode to be used for clock to clock arcs.<br>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.<br>(Default)                                                                                                            |
| 1       | Generates a clock tree report,<br>clock_tree.rpt, in the Liberate MX<br>directory. This report lists the active nets in<br>the clock tree path for each table. |

### **Example:**

```
TABLE: /xxx/xxx/xxx/constraint.tbl
clk
->clkb
-->clk_buf_
---->clk_gate
```

This variable must be set before `char_macro`.

## **mx\_clone\_if\_uda**

|           |                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------|
| < 0   1 > | Allows a worst-case arc to be cloned, only if a user-defined arc<br>for the cloned one is available. Default: 1 |
|-----------|-----------------------------------------------------------------------------------------------------------------|

This variable allows a worst-case arc to be cloned, only if a user-defined arc for the cloned one is available. Example:

```
# Turn off cloning
set_var mx_clone_if_uda 0
```

## **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 variable 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\_create\_if\_dynamic**

< 0 | 1 >               Adjusts internal MX algorithms to move dynamic filtering of statically-found arcs to occur early in the process. Default: 0 (Off)

This variable is used when the number of statically found arcs becomes very large, which may cause excessive memory usage. This can occur when `mx_setup_comb` or `mx_hold_comb` are set to "all". Symptoms may be that the process size grows too large, or the program stalls at this point:

```
(MX-info) - Auto-probing ...
```

If this occurs, please terminate the process, set this variable, and start again.

## **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` variable to 1 so that only the arcs you explicitly define are characterized.

## **Example**

Assume that

- The tool dynamically observes that there could exist an arc from input `IN0` to output `OUT1`.
- There is no user-defined arc between `IN0` and `OUT1`.

With these assumptions,

- If `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. |
|-------|---------------------------------------------|



## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

|                      |                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------|
| <code>measure</code> | Generates information on measure. It also creates <code>debug_measure.csv</code> file with all measurement details.          |
| <code>memory</code>  | Generates information on the system memory footprint (memory consumed during the run).                                       |
| <code>pattern</code> | Generates information on mxtable expansion results.                                                                          |
| <code>power</code>   | Generates information about power characterization.                                                                          |
| <code>sim</code>     | Generates information on FastSPICE simulation decks and commands being generated during the run.                             |
| <code>arc</code>     | Generates information on arc probing. It also creates <code>debug_arc.csv</code> 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
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

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 variable must be set before the `char_memory` command.

### **mx\_defmem\_sdf\_cond**

|                             |                                                                                                                |
|-----------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>&lt;0   1 &gt;</code> | Requests the reuse of the <code>sdf_cond</code> from the original library. Default: 0                          |
| 0                           | Do not reuse existing <code>sdf_cond</code> value. The <code>sdf_cond</code> is determined from the when arc . |
| 1                           | Reuse the existing <code>sdf_cond</code> 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 directory where MX temporary files should be stored. Default: `./mx`

Example:

```
# write mx files in to specified dir
set_var mx_dir /home/user/test_macro/mx
```

## **mx\_distributed\_sim**

<"options"> Passes switches and options to an external program. Default " " (none).

This variable 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.

|                                             |                                                                                                                                                                                                                                                    |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>static</code><br><code>dynamic</code> | 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 nodes are connected.

### **mx\_dynamic\_include\_full\_core**

`< 0 | 1 >` Instructs the tool to include the full core cell in dynamic partitioning independently from activity. Default: `1`

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\_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\_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\_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.

Note that 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 very 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 variable 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 variable 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

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. The usage is as follows:

1. Set the variable and run to have a first set of virtual rail candidates reported.
2. Modify the tcl script to include `set_vdd/set_gnd -virtual` commands on the reported nodes and rerun, checking whether static partitioning runtimes are as expected.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

### 3. Repeat as needed. Usually one iteration is sufficient.

#### 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.
```

when set to "2":

Liberate\_mx could auto recognize all the virtual rails, which drive pmos/nmos number is larger than mx\_virtual\_rail\_minimum\_xtrs(default is 50) and match the virtual rail topology. This value reports which wire is set as virtual rail, and some other wires may be virtual rail [mx/virtual.rpt].

Example: Mx/virtual.rpt

|                                                           |            |            |                   |
|-----------------------------------------------------------|------------|------------|-------------------|
| *** wire_name                                             | drive_pmos | drive_nmos | virtual_vdd       |
| N_XI0-Q_3_47_c_1031883_n<br>(has been set as virtual_vdd) | 2204       | 0          | set               |
| *** wire_name                                             | drive_pmos | drive_nmos | virtual_gnd       |
| N_XI0-VSS_c_697803_n<br>(has been set as virtual_gnd)     | 0          | 870        | set               |
| *** wire_name                                             | drive_pmos | drive_nmos | other_unset_wires |
| N_XI0-DBL_c_1280305_n<br>(not set as virtual rail)        | 10         | 68         | unset             |



## **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\_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  $(V_{\max} - V_{\min}) < mx\_full\_rail\_tol * VDD$ .

### **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\_fullsim\_measurement**

< 0 | 1 >                      Generates dynamic partitions out of `define_measurement` commands. Default: 0

## **mx\_greybox**

< 0 | 1 >                      Generate 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 variable 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\_inputcap\_ldb\_reuse**

<0 | 1> Reuse pin cap LDB from previous run. Default: 0

See [mx\\_ldbs\\_reuse](#) for description of all LDB reuse variables.

### **mx\_ldbs\_reuse**

<0 | 1> Reuse timing, power, pin-cap, noise, power and leakage LDBs from previous run. Default: 0

Liberate MX can reuse all or selective LDBs from the previous run by using the following variables:

[mx\\_inputcap\\_ldb\\_reuse](#)  
[mx\\_ldbs\\_reuse](#)  
[mx\\_power\\_ldb\\_reuse](#)  
[mx\\_noise\\_ldb\\_reuse](#)  
[mx\\_timing\\_ldb\\_reuse](#)

It will also re-characterize the arcs that failed characterization in previous run, if any.

### **mx\_margin\_report**

{filename} Generates a report by the `define_measure` commands.  
Default: "measure.rpt"

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

To generate this report, specify a filename. To omit this report, set this variable 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\_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\_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 variable 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 variable 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 variable 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_minp_period_mode "latch bitline_precharge"
```

## **mx\_minp\_single\_slew**

**<slew value>** Forces minimum period characterization to use the single clock slew. This variable 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` variable
- 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");  
}
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

```
}
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\_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 very small cases.)                                                                                 |

## **mx\_monitor\_memcore\_lprobe\_level**

|         |                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <value> | Uses <code>.lprobe</code> statement instead of <code>.probe</code> statement on memory core nodes. A <code>.lprobe</code> 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.<br>Default: -100 |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

To enable this in Liberate MX, the `mx_monitor_memcore_lprobe_level` variable 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**

<clock\_pulse | edge\_intersection | none>

Selects method for calculating MPW.  
Default: `edge_intersection`

`clock_pulse`



Selects calculation method used with version 3.0p3 and earlier.

`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" | "seq comb array">`

Specifies what type of logic should be probed. Default: `"seq"`

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 variable 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 very 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**

<string>                      Specifies the active edge(s) of a clock, otherwise Liberate MX assumes positive active edge(s). Default: none

-clock                         Specifies the name of negative clock edge(s) clock names.

This variable 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 -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 variables.

### **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 variable 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 variable, "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\_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 cap characterization. Default: 1

### **mx\_posedge\_clock**

<string>                      Specifies rising edge(s) clocks. Default: all

This variable 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\_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.

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

|                |                                                                                                                                                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 <code>clock</code> , 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.<br>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 variable 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 variable 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 variable 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 variables.

## **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.<br>(Default) |

Allows power characterization to be performed for multiple input slews and output loads, based on the power template that is provided.

This variable must be set before `char_macro`.

## **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. (*Please 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 specific 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\_probes\_report**

|                               |                                                                                                  |
|-------------------------------|--------------------------------------------------------------------------------------------------|
| <code>&lt;filename&gt;</code> | 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**

|                            |                                                                                                                                                                                                                                                                                  |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;0   1&gt;</code> | When set, a post processing algorithm filters selected probe and related probe pairs. This feature can be controlled by <code>mx_process_probe_relprobe_intersection_max_paths</code> and <code>mx_process_probe_relprobe_intersection_max_depth</code> variables.<br>Default: 1 |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## **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\_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 <code>read_spice</code> 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.<br>For backward compatibility only. |
| 1 | Always remove false initial condition groups.<br>(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 variable 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"

## Virtuoso Liberate MX Reference Manual

### Liberate MX Variables

---

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| <code>all</code>      | Remove parasitic RC networks for all nets.            |
| <code>none</code>     | Do not remove any parasitic RC networks.<br>(Default) |
| <code>rail</code>     | Remove parasitic RC networks for rails (vdd, vss).    |
| <code>net_name</code> | Remove parasitic RC network for the specified net.    |

This variable 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\_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\_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 variable 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.

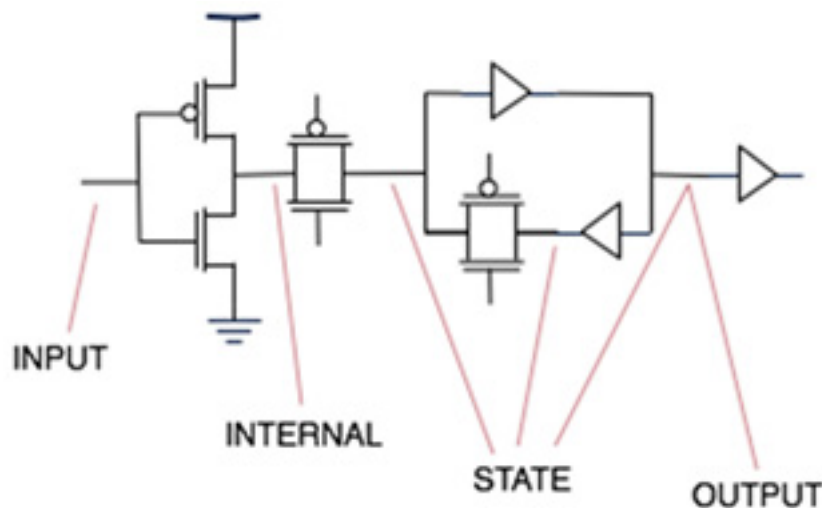
## **mx\_ring\_model\_fold**

**< 0 | 1 >** Turns on topological matching of logic. Default:1

Use this variable 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\_seq\_probing**

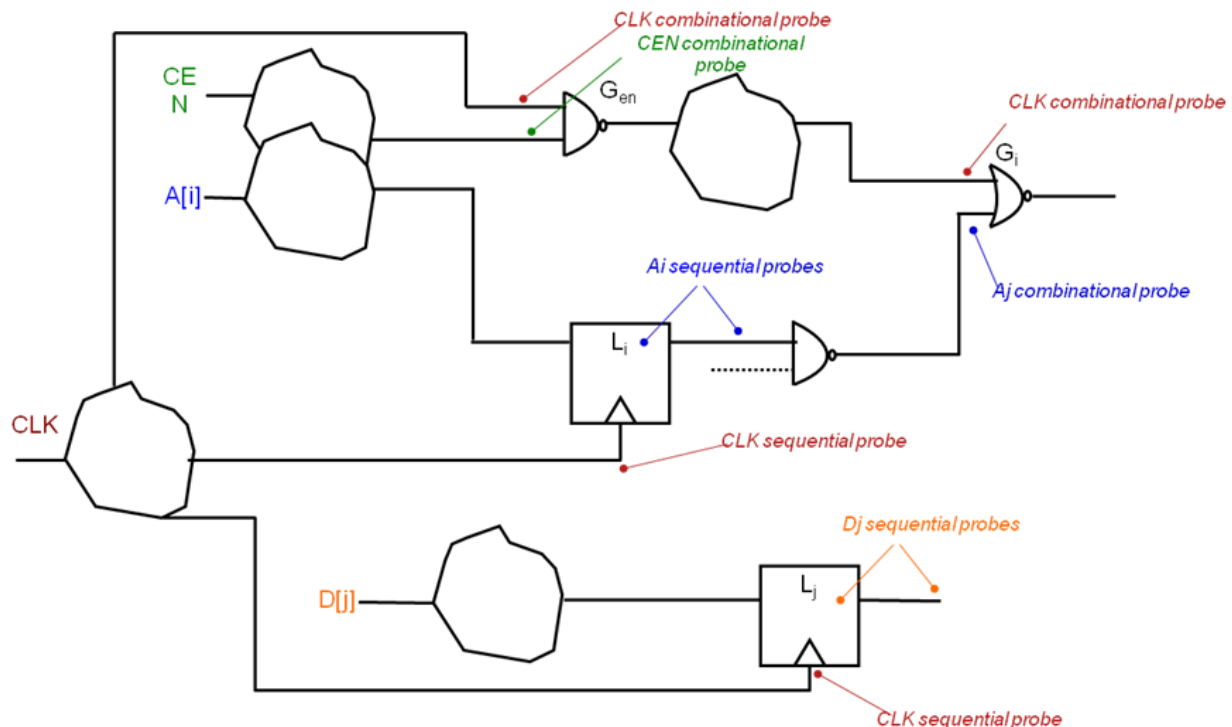
**<string>** Sets the node type to consider on a sequential component as possible candidates for data path probes in setup/hold characterization. Default: state output internal input.



**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 (please 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<sub>j</sub> as probes for D<sub>j</sub>, 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 the `: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**

|                            |                                                                                         |
|----------------------------|-----------------------------------------------------------------------------------------|
| <code>&lt;value&gt;</code> | Value, in seconds, of the simulation interval used by the FastSPICE step. Default: 10ns |
|----------------------------|-----------------------------------------------------------------------------------------|

Set the value of this variable 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 variable 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\_timing\_ldb\_reuse**

< 0 | 1 > Reuse timing LDB from previous run. Default: 0

See [mx\\_ldbs\\_reuse](#) for description of all LDB reuse variables.

## **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\_verbose**

< 0 | 1 >                      Prints basic flow and runtime information. Default: 1

Example:

```
#report basic flow info
set_var mx_verbose 1
```

## **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 <code>mx_find_virtual_rail</code> set to 0.                                                         |
| 1 | The <code>virtual.rpt</code> 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 <code>virtual.rpt</code> report. (Default)                                                                                |



This variable enables you to find (mx\_find\_virtual\_rails) and define virtual power nodes in a design.

This variable 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_pmosdrive_nmosother_unset_wires
VDD_1130388 unset
VSS_25081038unset
```

VDD\_1 is set as a virtual vdd because of connections to pmos. Note that eight opposite nmos are connected to the same net. Similarly, for VSS\_2 where 508 pmos are connected, which is below the default value and because of 1038 nmos connections, VSS\_2 is 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_minimum_xtrs` is a virtual rail.

Example:

```
# Defining minimum number of mos to "70"
set_var mx_virtual_rail_minimum_xtrs 70
```

### **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\_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 variable when characterizing large extracted netlists.

This variable must be used before the `char_macro` command.

## **parse\_auto\_define\_leafcell**

<0 | 1 | 2>                      Controls whether Liberate should recognize leafcells  
                                 when parsing a netlist, if yes, then how should it happen.  
                                 Default: 0

0                      Does not generate `define_leafcell`  
                         commands. This setting is provided for  
                         backward compatibility to LIBERATE 14.1  
                         ISR3 and prior releases.

1                      Recognizes leafcells automatically and  
                         prints `define_leafcell` commands into  
                         a log file. All of the user-provided  
                         `define_leafcell` commands are  
                         preserved and honored.

- |   |                                                                                                                                                                                                                                                                |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2 | Detects leafcells automatically, but does not apply the detected leafcells to the analysis. Print <code>define_leafcell</code> commands into a log file so that you can examine and fill in the missing information. For example: l, w, nfin, cjsw, and so on. |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

A leafcell is the lower most instance that is found when flattening a netlist that is outside of the model file. The `define_leafcell` command can be used to manually specify leaf cells. Also, the auto-detection of leafcells is supported only when using the Spectre Front End (SFE) parser (see `read_spice -format spectre`) and the variable `extsim_model_include` is set.

This variable must be set prior to the `char_memory` command.

### **set\_var\_failure\_action**

|                   |         |                                                                                                                                                                                                                                                                                                               |
|-------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <warning   error> |         | Notifies the <code>set_var</code> command how to consider a failure.<br>Default: warning                                                                                                                                                                                                                      |
|                   | error   | When a <code>set_var</code> fails, an error message is issued and subsequent commands that would result in characterization or library generation (for example, <code>char_memory</code> ) are suppressed. Subsequent <code>set_var</code> commands are still allowed so they can be checked for correctness. |
|                   | warning | A warning is issued when <code>set_var</code> fails. The failed <code>set_var</code> is ignored and execution continues.                                                                                                                                                                                      |

This variable 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.

## **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.

|                        |                                                                                                                           |
|------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <code>worst</code>     | Identifies the worst captured PWL among all virtual nodes and use it to drive PWL for all CCC [Channel Connected Regions] |
| <code>ccc_worst</code> | Identifies the worst PWL among all virtual nodes of each CCC [Channel Connected Regions] and apply it to respective CCC.  |

---

# Truth Table Format

---

## Specifying Input Stimuli

In Liberate MX, 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.

## Virtuoso Liberate MX Reference Manual

### Truth Table Format

---

`<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.

This example specifies Spectre as the FastSPICE simulator:

```
fastsim spectre
fastsim_deck .option rawfmt=fsdb
```

### **fastsim\_auto\_ic**

`<value>` Either 1 or 0 (May also be specified as `true`, `on`, `false`, `off`)

Instructs Liberate 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 variable `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`.

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
```

## Virtuoso Liberate MX Reference Manual

### Truth Table Format

---

```
simulation_interval 20n  
simulation_interval 20ns
```

Example:

```
simulation_interval 20e-9
```

The command above if specified in the `timing.tbl` table ensures that vectors listed in that table will use a simulation interval of 20ns rather than the default 10ns (which will be used for any other table that does not have a `simulation_interval` command).

### **fastsim\_clock\_slew**

<double>                      Specifies the clock slew to be used in FastSPICE simulation.  
For more information, see [mx\\_fastsim\\_clock\\_slew](#).  
Default: -1pf

### **fastsim\_input\_slew**

<double>                      Specifies the input slew to be used in FastSPICE simulation.  
For more information, see [mx\\_fastsim\\_input\\_slew](#).  
Default: -1pf

### **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 variable 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 keep paraname=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
```

## **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
```

## Virtuoso Liberate MX Reference Manual

### Truth Table Format

---

```
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:

# Virtuoso Liberate MX Reference Manual

## Truth Table Format

### 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<br>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 ( <i>constraints, delay, measure, and power</i> ) |
| C or -     | Constraints ( <i>setup &amp; hold</i> )               |
| D or delay | Delay                                                 |
| H or hold  | Hold constraint                                       |

## Virtuoso Liberate MX Reference Manual

### Truth Table Format

|                                          |                                                                 |                                 |
|------------------------------------------|-----------------------------------------------------------------|---------------------------------|
| <b>J or power</b>                        | Switching & hidden power                                        |                                 |
| <b>M</b>                                 | "define_measure" + mpw + minperiod + min & max clock tree paths |                                 |
| These offer further granularity over "M" | mpw                                                             | Minimum pulse width (mpw) only. |
|                                          | <b>min_period or minperiod</b>                                  | Minimum period only.            |
|                                          | min_clock_tree_path                                             | Minimum clock tree path only.   |
|                                          | max_clock_tree_path                                             | Maximum clock tree path only.   |
| <b>S or setup</b>                        | Setup constraint                                                |                                 |
| <b>W or leakage</b>                      | Leakage power                                                   |                                 |
| <b>X</b>                                 | - nothing - (Do not measure constraints, delay, or power)       |                                 |
| <b>Z</b>                                 | 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
```

## Virtuoso Liberate MX Reference Manual

### Truth Table Format

---

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
```

# Virtuoso Liberate MX Reference Manual

## Truth Table Format

---

```
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 = 1111111111 in to address ADR = 111111111
# write D = 0000000000 in to address ADR = 000000000
# read from address ADR = 000000000 and measure delay
# read from address ADR = 111111111 and measure delay
# write D = 0000000000 in to address ADR = 111111111
# write D = 1111111111 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 1      h      h      1      R      h      1      1      x #Dummy
write00 1      h      h      1      R      h      0      0      -
read0_ 1      h      1      h      R      1      0      ?      b
read1_ 1      h      1      h      R      1      1      ?      b
write10 1      h      h      1      R      h      1      0      -
write01 1      h      h      1      R      h      0      1      -
read0_ 1      h      1      h      R      1      0      ?      b
read1_ 1      h      1      h      R      1      1      ?      b
endtable
```

```
table delay_bist
pins      BISTE    TME      TWE      TOE      CLK      TWEM      TADR      TD      Q
write11 h      h      h      1      R      h      1      1      -
write00 h      h      h      1      R      h      0      0      -
read0_ h      h      1      h      R      1      0      ?      b
read1_ h      h      1      h      R      1      1      ?      b
write10 h      h      h      1      R      h      1      0      -
write01 h      h      h      1      R      h      0      1      -
```



## Virtuoso Liberate MX Reference Manual

### Truth Table Format

```

read0_  h      h      1      h      R      1      0      ?      b
read1_  h      h      1      h      R      1      1      ?      b
endtable

```

```

table constraint
pins    CLK      ADR      D      WEM      WE      OE      ME      AWT      BISTE      Q
const   R        b        1        1        1        1        h        1        1        -
const   R        1        b        1        1        1        h        1        1        -
const   R        1        1        b        b        1        h        1        1        -
const   R        1        1        1        1        1        b        1        1        -
const   R        1        1        1        1        1        h        1        0        -
endtable

```

```

table constraint_bist
pins    CLK      TADR      TD      TWEM      TWE      TOE      TME      AWT      BISTE      Q
const   R        b        1        1        1        1        h        1        h        -
const   R        1        b        1        1        1        h        1        h        -
const   R        1        1        b        b        1        h        1        h        -
const   R        1        1        1        1        1        b        1        h        -
const   R        1        1        1        1        1        h        1        1        -
endtable

```

```

table asynch
pins    BISTE      AWT      D      WEM      TD      TWEM      OE      TOE      Q
asynch  1          R        b        b        ?        ?        h        1        b
asynch  h          R        ?        ?        b        b        1        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 namespace 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.

## Virtuoso Liberate MX Reference Manual

### Truth Table Format

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.

```
set cwd [pwd]

set ::tbl_var 0.9
set_var mx_evaluate_tables 1

set_var mx_create_if_uda 1
set_var duplicate_risefall_power 1
set_var mx_dir [pwd]/mx
set_var extsim_deck_dir [pwd]/decks
set_var mx_ring_max_xtr_cnt 1000
set_var mx_mpw_probe "seq comb"
set_var mx_switch_wire_threshold 0.45
set_var mx_push_probe_inside_array 1
set_var mx_rcdb_dir [pwd]/rcdbs
set_var mx_rcdb_ukt [pwd]/rcdbs/ukt
set_var mx_margin_report measure.rpt
set_operating_condition -voltage 1.0 -temp 25
set_vdd VDD 1.0
set_gnd VSS 0.0
set_var constraint_probe_lower_rise 0.2
set_var constraint_probe_upper_rise 0.8
set_var constraint_probe_lower_fall 0.2
set_var constraint_probe_upper_fall 0.8
set_var mx_mpw_probe_lower_rise 0.2
set_var mx_mpw_probe_upper_rise 0.8
set_var mx_mpw_probe_lower_fall 0.2
set_var mx_mpw_probe_upper_fall 0.8

#set_var extsim_save_passed all
set_var extsim_save_failed all

set_var mx_monitor_memcore all

arctypes timing
fastsim spectre
fastsim_cmd_option -64 +spice +xps +cktpreset=sram

fastsim_deck .ic X1.xclk.clk_inb ${::tbl_var}

table write
pins clk_in add_in data_in chip_en wr_in data_out
W_A0 R L 0x55555555 H H X
W_A1 R H 0xaaaaaaaa H H X
endtable

table read
pins clk_in add_in chip_en wr_in data_out
init R L H L X
R_A1 R H H L D
DATA/tables/delay.tbl 1,1 Top

* input and clock slews - for user defined pwls
.param input_slew_r=2e-10
.param input_slew_f=2e-10
.param clock_slew_r=2e-10
.param clock_slew_f=2e-10

* default options

* user options specified in mx_set_spectre param
.option soft_bin=allmodels

* user options/commands/params specified in table DATA/tables/delay.tbl.eval
.ic X1.xclk.clk_inb 0.9

.param tran_tend=2.0000000e-07
.tran 1.00e-12 'tran_tend'
```

---

## 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><br>Identifies the core cell.   |
| <group_id>  | storage, port<br>Identifies the group type. |

## Virtuoso Liberate MX Reference Manual

### Specifying Memory Core Cells

---

|               |                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <subgroup_id> | sram, dram, rom, write, read, writeread<br>Identifies the subgroup type                                                                                                                                                                                                                                                                      |
| <model_id>    | n, p, c<br>Identifies the device type                                                                                                                                                                                                                                                                                                        |
| <side_id>     | 1, 0<br>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. Note that choice of 0, 1 is fully arbitrary - but must be consistent among different devices. See examples below. |
| <ctrl_id>     | word, core<br>Identifies device's gate terminal connectivity: controlled by wordline, controlled by mem core node.                                                                                                                                                                                                                           |
| <use_id>      | pass, pull-up, pull-down, storage<br>Identifies how the device is used; as a pass transistor, a pull up one, a pull down one, a storage.                                                                                                                                                                                                     |
| <l_id>        | double<br>Specifies L for device - when <model_id> is n or p                                                                                                                                                                                                                                                                                 |
| <w_id>        | double<br>Specifies W for device - when <model_id> is n or p                                                                                                                                                                                                                                                                                 |
| <c_id>        | double<br>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
            }
        }
    }
}
```

## Virtuoso Liberate MX Reference Manual

### Specifying Memory Core Cells

---

```
        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
        }
    }
}
```

## Virtuoso Liberate MX Reference Manual

### Specifying Memory Core Cells

---

```

device {
  model:nmos
  side:0
  ctrl:wordbit
  use:pass
}}}}

```

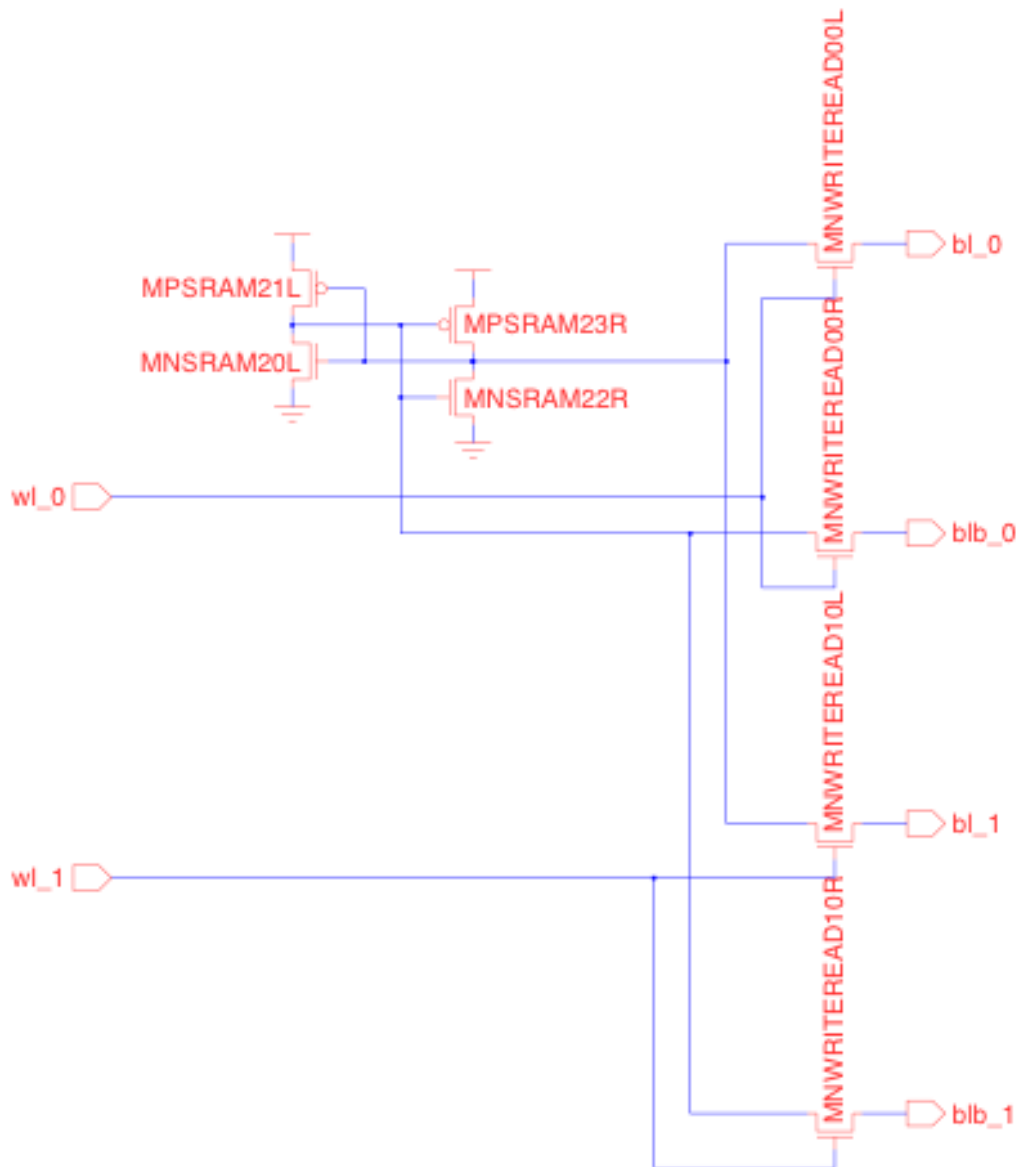


Figure 2 – 8T core cell

## Virtuoso Liberate MX Reference Manual

### Specifying Memory Core Cells

---

#### 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
      }
    }
  }
}
```

## Virtuoso Liberate MX Reference Manual

### Specifying Memory Core Cells

---

```
    }
    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
        }
    }
}
```



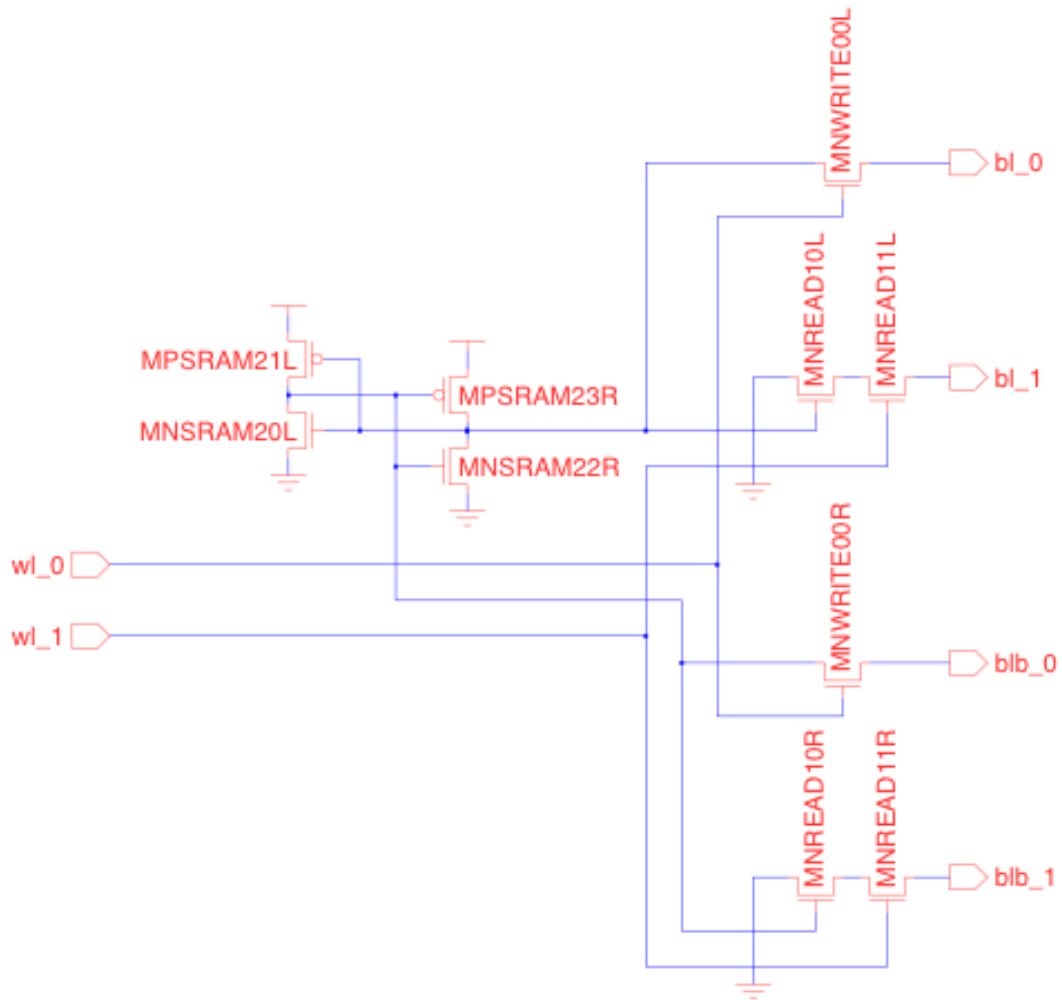


Figure 3 – 10T core cell

## **Virtuoso Liberate MX Reference Manual**

### **Specifying Memory Core Cells**

---

---

## 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,

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

---

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`.

|                        |                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------|
| <code>-ref_lib</code>  | Reference Liberty file, containing all arcs and conditions to be used during the characterization. |
| <code>-template</code> | 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:

|                      |                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------|
| <code>-slews</code>  | List of slews to be characterized. Overrides template or reference lib definition.                                           |
| <code>-loads</code>  | List of loads to be characterized. Overrides template of reference lib definition.                                           |
| <code>-qloads</code> | List of loads to be characterized for the data out pin. Overrides template, reference lib or <code>-loads</code> 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:

|                              |                                                                 |
|------------------------------|-----------------------------------------------------------------|
| <code>-netlist</code>        | The instance netlist file.                                      |
| <code>-netlist_format</code> | The format of the netlist file. Default is <code>spice</code> . |
| <code>-models</code>         | The model include file.                                         |

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

|                  |                                                                                                                  |
|------------------|------------------------------------------------------------------------------------------------------------------|
| -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 <code>spice</code> .                                                        |
| -skip_read_model | Controls whether the models will be read in with <code>read_spice</code> . Default is not to read in the models. |

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 $2^{(\text{number of address bits})}$ . It is essential to define the number of words if the number of words is less than $2^{(\text{number of address bits})}$ 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.                                                                                                                                                                                                                                                        |

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

|                  |                                                                                                                                             |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| -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.           |

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

---

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 <code>define_memory</code> that are not to be used for the characterization. The allowed values are: leakage, delay, constraint, power, measure, bist, sleep, and bist_pwr. |

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

---

#### ***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. |
| -autoprobng_hold_level |                                                                                                                         |



## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

---

|                                       |                                                                          |
|---------------------------------------|--------------------------------------------------------------------------|
|                                       | List of pins and values for autoprobing hold level. Default is level 0.  |
| <code>-autoprobing_setup_level</code> |                                                                          |
|                                       | 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.

|                          |                                                                                      |
|--------------------------|--------------------------------------------------------------------------------------|
| <code>-mx_setting</code> | 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:

|                               |                                                                                                                                                                                                         |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-greybox</code>         | Enables the greybox (non-partitioning) flow. Default is 0.                                                                                                                                              |
| <code>-bisection</code>       | Enables bisection characterization for constraints. Default is 0.                                                                                                                                       |
| <code>-rcdb</code>            | 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. |
| <code>user_memory_name</code> | 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 very 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:

|                        |                                                                                                                              |
|------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <code>-work_dir</code> | Specify the work directory name. It will be created as a subdirectory of the current directory if it does not already exist. |
| <code>-ecsm</code>     | Enables ECSCM characterization and lib generation.                                                                           |
| <code>-ecsmn</code>    | Enables ECSCMN characterization and lib generation.                                                                          |

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

|       |                                                       |
|-------|-------------------------------------------------------|
| -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} \  

```

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

---

```
-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 |
|----------|----------|-----------|
|----------|----------|-----------|

## Virtuoso Liberate MX Reference Manual

### Using Liberate MX Automatic Flow

---

|       |              |       |
|-------|--------------|-------|
| 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.

---

## 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`

## Virtuoso Liberate MX Reference Manual

### Liberate MX Timing Validation Flow

---

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>
```

---

## 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 \  

```

## Virtuoso Liberate MX Reference Manual

### Basic Flow for Validating User-Defined Criteria

---

```
-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
```



---

# 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 generally 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:

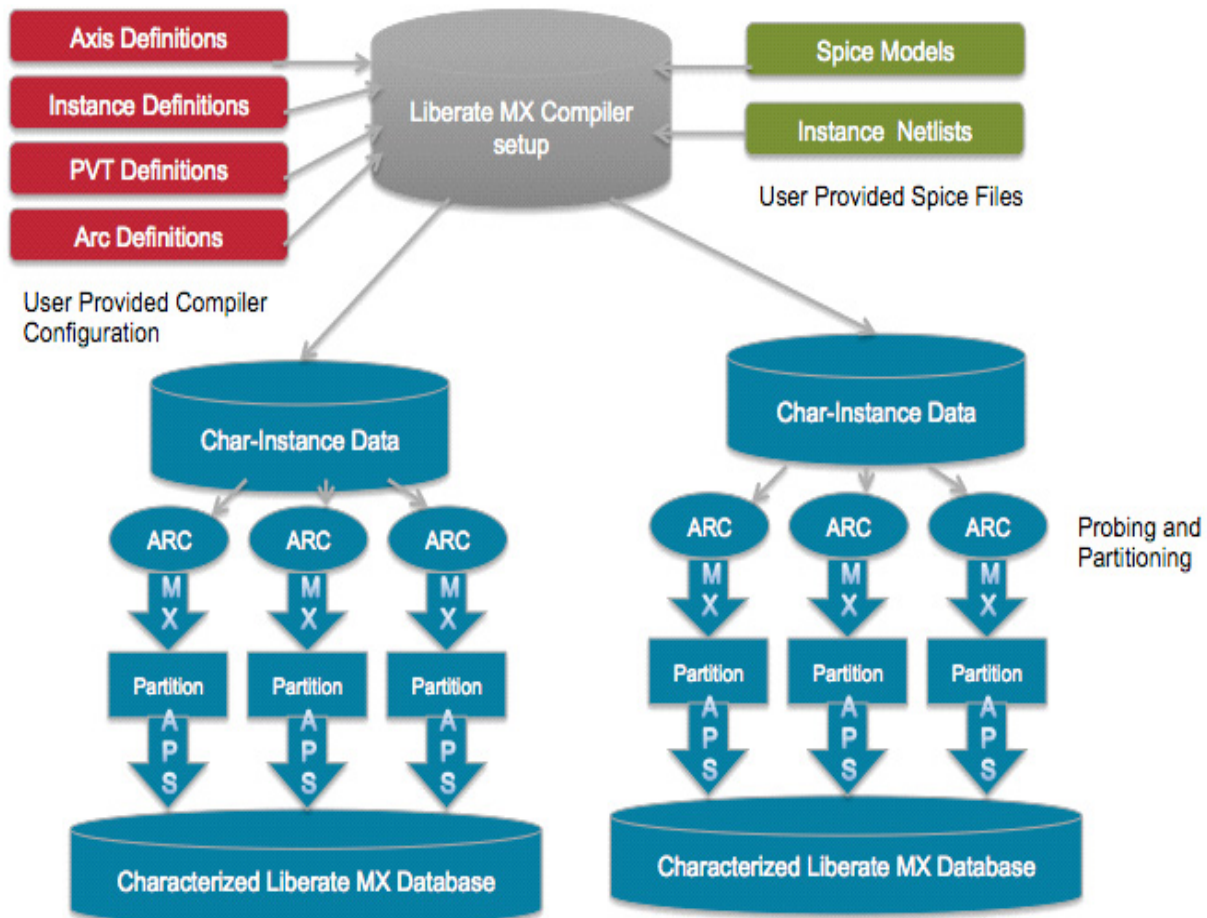
- Linear interpolation
- Curve fitting
- Formula-based

## 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.

## Defining the Compiler Space

1. Define the axes that can vary and their ranges with the `compiler_define_axis` command.

### □ `compiler_define_axis`

`-min_value`                      The minimum value for the axis. Default: 0

`-max_value`                      The maximum value for the axis. Default: 1

|                              |                                                                                                 |
|------------------------------|-------------------------------------------------------------------------------------------------|
| <code>-increment</code>      | The increment of allowed values. Default: 1                                                     |
| <code>-allowed_values</code> | The list of allowed values. Default: min, max, and all intermediate values defined by increment |
| <code>&lt;name&gt;</code>    | The axis name to be used in instance definitions.                                               |

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.

```
compiler_define_axis \
  -min_value 2 \
  -max_value 128 \
  -increment 1 \
  bit

compiler_define_axis \
  -min_value 16 \
  -max_value 2048 \
  -increment 4 \
  word
```

It is generally 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.

2. Define a 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.

#### ❑ **compiler\_define\_design**

|                                    |                                                                                                                                                                                                                                                                                                             |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-axis</code>                 | Paired list of axis name followed by min:max. Unspecified axis uses the ranges defined in <code>define_axis</code> .                                                                                                                                                                                        |
| <code>-mx_settings</code>          | Additional file containing Liberate MX settings.                                                                                                                                                                                                                                                            |
| <code>-bitcell</code>              | The type of bitcell. One of the following value can be specified: rom, single_port, dual_port, 2prf, or 10t. Default: single_port                                                                                                                                                                           |
| <code>-bist_prefix "string"</code> | 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) |
| <code>-bist_suffix "string"</code> |                                                                                                                                                                                                                                                                                                             |

|               |                                                                                                                                                                                                                                                                                                             |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | 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_inst` command.

## Defining the PVT Corner for Characterization

The PVT corners to be used for the compiler characterization must be defined. Each PVT corner requires a separate run of Liberate MX.

### `compiler_define_pvt`

|                              |                                                                                                                                                                     |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-cmd_file_path</code>  | Specifies a directory that contains the top-level command files for instances of each PVT. For example:<br><br><code>-cmd_file_path \$cmd_dir\<br/>\$process</code> |
| <code>-rail</code>           | Name and value pairs for each voltage supply.                                                                                                                       |
| <code>-virtual_rails</code>  | Name and value pairs for each virtual rail.                                                                                                                         |
| <code>-models</code>         | Name of the SPICE model include file.                                                                                                                               |
| <code>-model_format</code>   | Model format to be used. One of the following values can be specified: <code>spice</code> or <code>spectre</code> . Default: <code>spice</code>                     |
| <code>-temp</code>           | Temperature in degree Celsius.                                                                                                                                      |
| <code>-ref_lib_path</code>   | Directory for the reference <code>.lib</code> files of the PVT corner.                                                                                              |
| <code>-template_path</code>  | Directory for the template files of the PVT corner.                                                                                                                 |
| <code>-netlist_path</code>   | Directory for the instance netlists of the PVT corner.                                                                                                              |
| <code>-user_data_path</code> | Directory for the user data files of the PVT corner.                                                                                                                |
| <code>&lt;name&gt;</code>    | Name of the PVT corner.                                                                                                                                             |

The different netlist path allows you to use different extracted netlists for different PVT corners.

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**

|                                      |                                                                                                                                                                                                                                                                                                       |
|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-cmd_file</code>               | Specifies the custom top-level Tcl command file. For example:<br><pre>-cmd_file mx.tcl \$process</pre>                                                                                                                                                                                                |
| <code>-axis</code>                   | Paired list of axis name followed by value. All axes defined with <code>define_axis</code> must be assigned a value.                                                                                                                                                                                  |
| <code>-design</code>                 | Name of the compiler design as defined with <code>compiler_define_design</code> .                                                                                                                                                                                                                     |
| <code>-netlist</code>                | Name of the instance netlist. This can be found in the directory specified by <code>compiler_define_pvt -netlist_path</code> .                                                                                                                                                                        |
| <code>-netlist_format</code>         | Netlist format. One of the following values are allowed: <code>spice</code> or <code>spectre</code> . Default: <code>spice</code>                                                                                                                                                                     |
| <code>-ref_lib</code>                | Name of the reference <code>.lib</code> file. This can be found in the directory specified by <code>compiler_define_pvt -ref_lib_path</code> .                                                                                                                                                        |
| <code>-template</code>               | Name of the template file. This can be found in the directory specified by <code>compiler_define_pvt -template_path</code> .                                                                                                                                                                          |
| <code>-user_data</code>              | Name of the user data file. This can be found in the directory specified by <code>compiler_define_pvt -user_data_path</code> .                                                                                                                                                                        |
| <code>-additional_tables</code>      | List of additional tables to be used for the characterization.                                                                                                                                                                                                                                        |
| <code>-remove_tables</code>          | List of generated tables not to be used for the characterization. One of the following values can be specified: <code>leakage</code> , <code>delay</code> , <code>constraint</code> , <code>power</code> , <code>measure</code> , <code>bist</code> , <code>sleep</code> , or <code>bist_pwr</code> . |
| <code>{&lt;instance_name&gt;}</code> | Name of the instance. (Required)                                                                                                                                                                                                                                                                      |

For example:



## Virtuoso Liberate MX Reference Manual

### Liberate MX Compiler Characterization Flow

---

```
compiler_define_instance \  
  -design RF \  
  -axis { \  
      {word 16} \  
      {bit 2} \  
  } \  
  -netlist rf_16x2.cir \  
  -template rf_16x2_tmpl.tcl \  
  rf_16x2
```

The `compiler_char_instances` command 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).

## **compiler\_char\_instances**

|                                  |                                                                                                                                                                                                               |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-part_spice</code>         | Simulator to be used for the full instance partitioning simulations. One of the following values can be specified: <code>ultrasim</code> , <code>aps</code> , or <code>xps</code> . Default: <code>xps</code> |
| <code>-char_spice</code>         | Simulator to be used for the partition characterization phase. One of the following values can be specified: <code>spectre</code> or <code>aps</code> . Default: <code>aps</code>                             |
| <code>-part_thread</code>        | Number of threads to be used for the full instance partition creation simulations. Default: 0 (use all available threads)                                                                                     |
| <code>-char_thread</code>        | Number of threads to be used for the partition characterization phase. Default: 0 (use all available threads)                                                                                                 |
| <code>-inst_thread</code>        | Number of instance characterizations to be done simultaneously. Default: 2                                                                                                                                    |
| <code>-extsim_cmd_option</code>  | Command options for the partition characterization phase.                                                                                                                                                     |
| <code>-fastsim_cmd_option</code> | Command options for the full instance partition creation phase.                                                                                                                                               |
| <code>-pre_lib_script</code>     | Tcl file to be sourced before writing the library.                                                                                                                                                            |
| <code>-ccs</code>                | Enable CCS characterization. Default: <code>false</code>                                                                                                                                                      |
| <code>-ccsn</code>               | Enable CCSN characterization. Default: <code>false</code>                                                                                                                                                     |
| <code>-ecsm</code>               | Enable ECSM characterization. Default: <code>false</code>                                                                                                                                                     |
| <code>-ecsmn</code>              | Enable ECSMN characterization. Default: <code>false</code>                                                                                                                                                    |
| <code>-pvt</code>                | List of previously defined PVTs to be used for the characterization.                                                                                                                                          |
| <code>-instances</code>          | List of previously defined instances to be characterized.                                                                                                                                                     |
| <code>-workdir</code>            | Working directory for the characterization. Default is the current directory.                                                                                                                                 |

### **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`

|                                      |                                                                                                                                |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>-axis</code>                   | Paired list of axis name followed by value. All axes defined with <code>define_axis</code> must be assigned a value.           |
| <code>-design</code>                 | Name of the compiler design as defined in <code>compiler_define_design</code> .                                                |
| <code>-template</code>               | Name of the template file. This can be found in the directory specified by <code>compiler_define_pvt -template_path</code> .   |
| <code>-user_data</code>              | Name of the user data file. This can be found in the directory specified by <code>compiler_define_pvt -user_data_path</code> . |
| <code>{&lt;instance_name&gt;}</code> | Name of the instance. (Required)                                                                                               |

For example:

```

interpolate_define_instance \
  -design RF \
  -axis { \
    {word 64} \
    {bit 16} \
  } \
  -template rf_64x16_tmpl.tcl \
  rf_64x16

```

The `interpolate_generate_instances` command specifies the defined instances and PVTs that will be generated.

### **interpolate\_generate\_instances**

|                                  |                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-pre_lib_script</code>     | Tcl file to be sourced before writing the library.                                                                                                                                                                                                                                                                                                                                                       |
| <code>-ccs</code>                | Enable CCS characterization. Default: <code>false</code>                                                                                                                                                                                                                                                                                                                                                 |
| <code>-ccsn</code>               | Enable CCSN characterization. Default: <code>false</code>                                                                                                                                                                                                                                                                                                                                                |
| <code>-closest_char_insts</code> | 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. |
| <code>-ecsm</code>               | Enable ECSM characterization. Default: <code>false</code>                                                                                                                                                                                                                                                                                                                                                |
| <code>-ecsmn</code>              | Enable ECSMN characterization. Default: <code>false</code>                                                                                                                                                                                                                                                                                                                                               |
| <code>-pvt</code>                | List of previously defined PVTs to be used for the characterization.                                                                                                                                                                                                                                                                                                                                     |
| <code>-instances</code>          | List of previously defined instances to be characterized.                                                                                                                                                                                                                                                                                                                                                |
| <code>-workdir</code>            | Working directory for the characterization. Default is the current directory.                                                                                                                                                                                                                                                                                                                            |

For example:

```
interpolate_generate_instances \  
  -design RF \  
  -instances { \  
    rf_64x16 \  
  } \  
  -workdir [pwd]/comp_char \  
  -pvt Typical
```

The instances listed with the `-instances` flag must have been previously defined with the `interpolate_define_instance` command.

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
```

## Interpolation

To predict the data for the uncharacterized instances, Liberate MX uses the characterized instances and interpolate. 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***

|                           |                                                            |
|---------------------------|------------------------------------------------------------|
| <code>-expression</code>  | Arithmetic expression in terms of previously defined axis. |
| <code>&lt;name&gt;</code> | 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
```

## 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}
```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Compiler Characterization Flow

---

```
-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 \  

```

## Virtuoso Liberate MX Reference Manual

### Liberate MX Compiler Characterization Flow

---

```
-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 16} \  
    {bit 32} \  
} \  
-netlist rf_16x32.cir \  
-template rf_16x32_tmpl.tcl \  
rf_16x32
```



```
        {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

The Liberate MX compiler creation flow uses a server and client licensing method similar to the Liberate MX instance characterization flow.

The master process of the compiler characterization requires a Liberate MX server license. During compiler characterization, each parallel instance characterization executed by the compiler characterization also requires a Liberate MX server license. Therefore, the number of Liberate MX server licenses required is equal to the number of instance threads specified in `compiler_char_instances` plus one. The number of Liberate MX client licenses required is equal to the number of parallel partition runs for each instance characterization (`char_thread`) times the number of parallel instance characterizations (`inst_thread`). There are a minimum of 10 client licenses per instance in the compiler creation flow.

```
Server_licenses = 1 + inst_thread
Client_licenses = inst_thread * char_thread
```

## **Virtuoso Liberate MX Reference Manual**

### **Liberate MX Compiler Characterization Flow**

---

The minimum requirement to perform compiler characterization is 3 Liberate MX server licenses and 20 Liberate MX client licenses.

For a run that is only generating estimated data for the interpolated instances, a Liberate MX server license is required for the master process and additional server licenses for the instances that are being created. Minimum two server licenses are required, while no Liberate MX client licenses are needed for this step.

---

## Legacy Commands and Variables

---

This chapter lists all the commands and variables that are either deprecated, or included for backward compatibility. We would like to discourage users from relying on these commands and variables. Instead, find the alternate command or variable along with its settings to achieve the best results from Liberate MX.

### Deprecated Commands

The following commands are being phased out, and have been replaced by either new commands (or related options), new variables, or new behaviors of the tool.

#### **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.**

|                           |                                                                                                                         |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;list&gt;</code> | 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.

This variable is used to pass parameters to the HSIM simulator. Parameters are passed as a list of name-value pairs.

Parameters can also be passed by specifying them in a table and using the 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.

This variable is used to pass parameters to the NanoSim simulator. Parameters are passed as a list of name-value pairs.

Parameters can also be passed by specifying them in a table and using the `define_table` command. Parameters specified in a table override parameters that are specified with a Tcl command.

## Deprecated Variables

### **mx\_remove\_rc**

Use the mx\_remove\_rc\_pincap and mx\_remove\_rc\_timing variables instead.

### **mx\_whitebox\_monitor\_memcores**

Use the mx\_monitor\_memcore variable instead.

### **mx\_active\_load\_thresh**

|                            |                                                                                                                     |
|----------------------------|---------------------------------------------------------------------------------------------------------------------|
| <code>&lt;value&gt;</code> | 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 very large number.)

Example:

```
set_var mx_active_load_thresh "1e-15"
```

Use the mx\_active\_load\_channel\_thresh and mx\_active\_load\_gate\_thresh variables instead.

## **Virtuoso Liberate MX Reference Manual**

### Legacy Commands and Variables

---

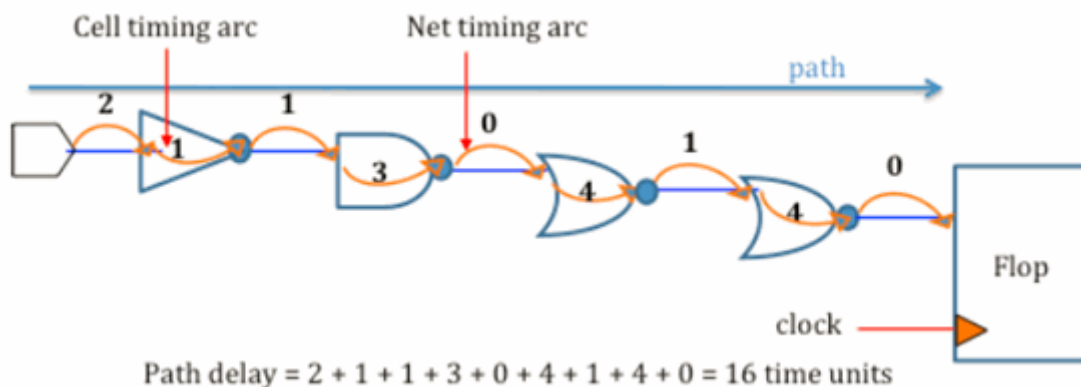
## Glossary

### Inside View

All characterization solutions available in the Virtuoso Characterization Suite use a unique “inside view” pre-characterization circuit analysis technique to perform vector generation and binning, automatic indices selection, and optimization of timing constraint characterization. This enables fully-automated library creation mechanism.

### Static Timing Analysis (STA)

It is an approach to verify timing of digital designs that uses cell delays and net delays to obtain path delays which is used to validate timing specification. It validates if the design can operate at the rated speed. The figure below shows a simple example in which the cell and net delays are used to obtain the path delay.



### Timing Libraries

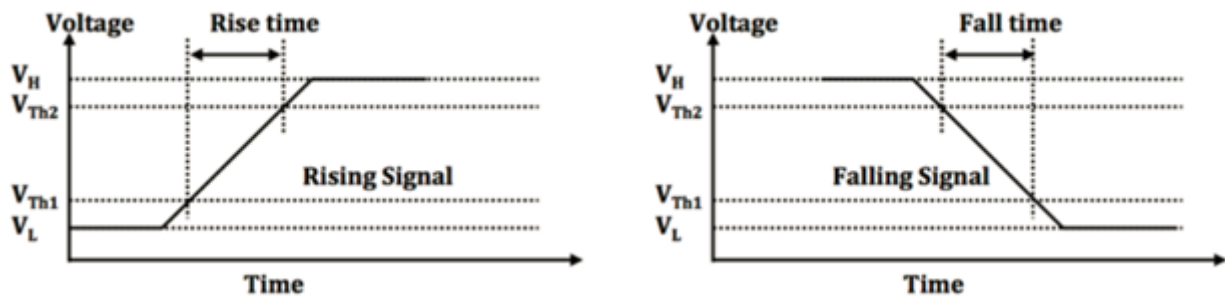
The Liberty format (.lib) is the industry standard for specifying the timing information. It is an ASCII file containing timing and power numbers associated with a cell. These are obtained by running SPICE simulations on the cell under a range of conditions.

### Timing Arcs

A timing arc is a construct used to represent a single causal (change in input causes change in output) relationship. These arcs form the building blocks for STA.

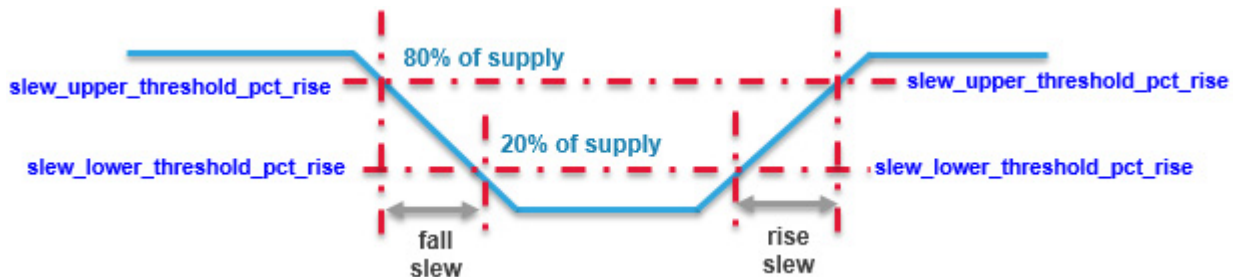
## Transition Time

It is defined as the time it takes for a signal to change states between two specific levels. Rise and Fall transitions times, as shown in the figure below, are properties of a timing arc.



## Slew

As slew rate is the rate of change, slew is typically measured in terms of transition time. Thresholds of signal transition times are used to measure slew. In the figure below, the threshold setting specifies that the falling slew is the difference between time points when the falling edge reaches 80% and 20% of supply value.



The slew thresholds are typically chosen to correspond to the part of the waveform that is linear. In newer technologies, the timing libraries will have these thresholds set to 30% and 70% of supply.

## Cell Delay

Propagation delay through a cell is commonly known as cell delay. The slew of the input waveform and the load connected to the cell influence the delay value. The delay values are characterized for different values of input slew and output load. Typical delay measurements are done from 50% of input signal to 50% of output signal.

## Related Pin

`Related_pin` attribute defines pin(s) that represent the beginning point of the timing arc. This attribute is required in all timing groups.



## Timing Arc Types

Following types of timing arc can be used:

❑ **Combinational timing arcs:**

These are used to describe the timing arcs for combinational element. The timing arc will be attached to an output pin and the related\_pin will be an input or an output. Types of combinational timing arcs:

- combinational
- combinational\_rise
- combinational\_fall
- three\_state\_disable
- three\_state\_disable\_rise
- three\_state\_disable\_fall
- three\_state\_enable
- three\_state\_enable\_rise
- three\_state\_enable\_fall

❑ **Sequential timing arcs:**

These describe timing arcs for sequential elements. It can be a delay arc (if it describes relation between clock transition to data output i.e. input to output) or a constraint arc (if it describes relation between clock transition and data input i.e. input to input). Sequential timing arcs can be one of the following:

- Edge-sensitive (rising\_edge or falling\_edge)
- Preset or clear
- Setup or hold (setup\_rising, setup\_falling, hold\_rising, or hold\_falling)
- Nonsequential setup or hold (non\_seq\_setup\_rising, non\_seq\_setup\_falling, non\_seq\_hold\_rising, non\_seq\_hold\_falling)
- Recovery or removal (recovery\_rising, recovery\_falling, removal\_rising, or removal\_falling)
- No change (nochange\_high\_high, nochange\_high\_low, nochange\_low\_high, nochange\_low\_low)

## Timing Sense

This attribute is used in the library to specify unateness in the .lib file.

## Setup and Hold

These are synchronous timing checks that ensure proper propagation of data through sequential cells. Setup time is the duration for which input data must be stable before the triggering edge of clock. Hold time is the duration for which the synchronous input should be stable after the triggering edge of clock.

