Cadence Liberate MX Memory Characterization
Version 13.1

Cedar Xu
Lead Application Engineer (CIC)
July 24th, 2014

cadence®

Course Agenda

1. About This Course
2. Cadence Online Help
3. Cadence Characterization Product Overview
4. Introduction to Memory Characterization
5. Library Format Overview
6. Introduction to Liberate MX
7. The benefits of Using Liberate MX and Spectre XPS together
8. The Flows of Liberate MX
9. The Liberate MX Full Custom Flow
10. The Liberate MX stand custom flow
11. The Liberate MX Vendor instance Re-characterization flow
12. Model Generation
13. The Liberate MX Validation Flow
14. Running Liberate MX
15. Analyzing and Debugging Output
16. Appendix: ViVA Overview

Module 1: About This Course

Course Prerequisites

- Before taking this course, you need to have
  - A working knowledge of Liberty Syntax
  - A basic knowledge of Tcl programming
  - An understanding of transistor level simulation and syntax
  - A basic knowledge of Memory (SRAM) operations and circuits

Course Objectives

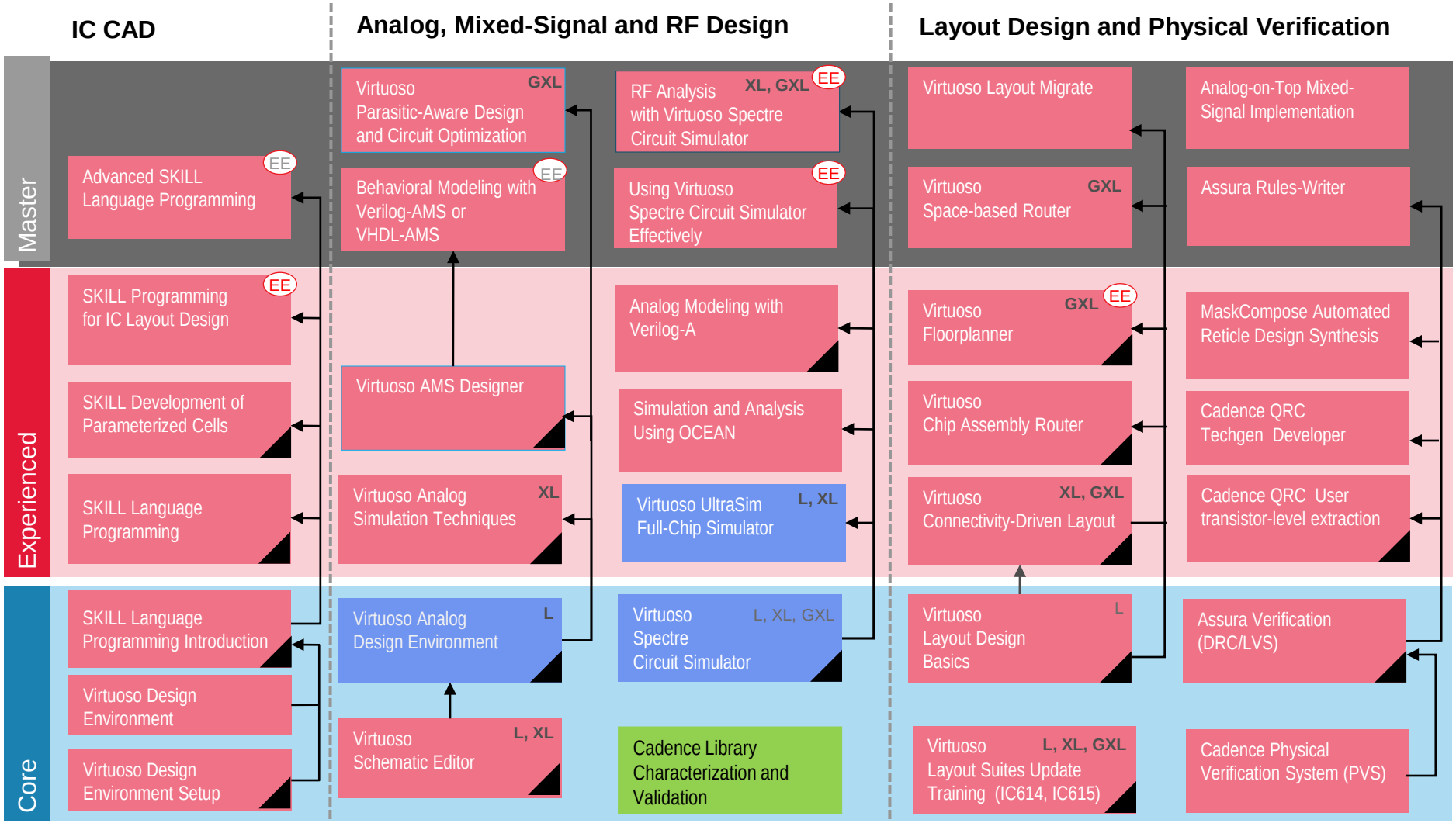
- In this course, you will
  - Understand the objectives and theory of memory characterization
  - Characterize memory instances for timing, power, and leakage
  - Create Liberty, Verilog, VHDL, and datasheet library views 
  - Learn the flows needed for re-characterization of third party memory IP
  - Understand how to validate that the characterized numbers are functional

Module 2: Cadence Online Help

Getting Help

- There are four ways to get help.
  - Use Cadence Help to access tool-specific information.
    - `cdnshelp`
  - Access Cadence Online Support for documents and support.
    - http://support.cadence.com
  - Go to the Cadence Community web site to stay current with tips and tricks.
    - http://www.cadence.com/community/forums
    - http://www.cadence.com/community/blogs
    - http://www.cadence.com/cdnlive
  - In browser… “altos”…select wiki … select support … FAQ
  - Send email to china_crc@cadence.com

Learning Map for
Custom Design with Virtuoso® and Assura® Technology



Some course titles may vary. Please refer to your regional catalog for exact titles and course datasheets.

L, XL, and GXL denote the tiers of Cadence products.

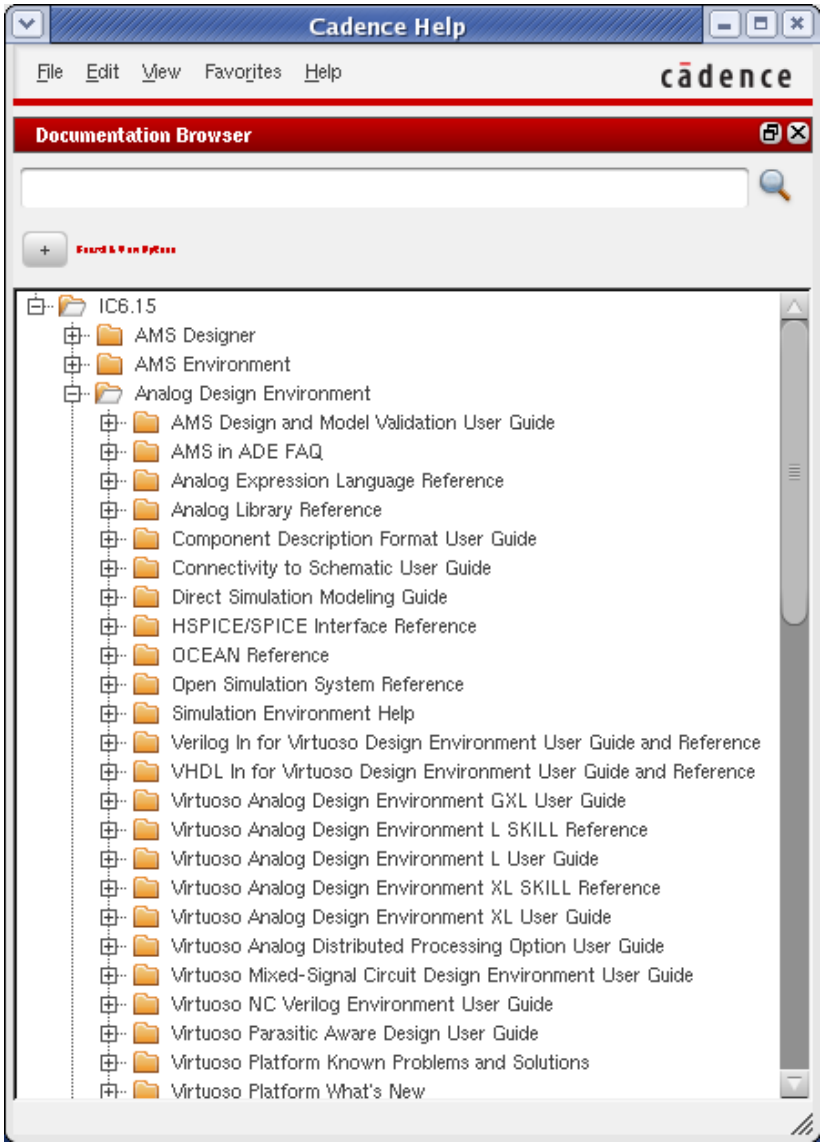
◢ Also available as an Internet Learning Series course (online)

EE This is an Engineer Explorer Course.

cadence®

Cadence Help

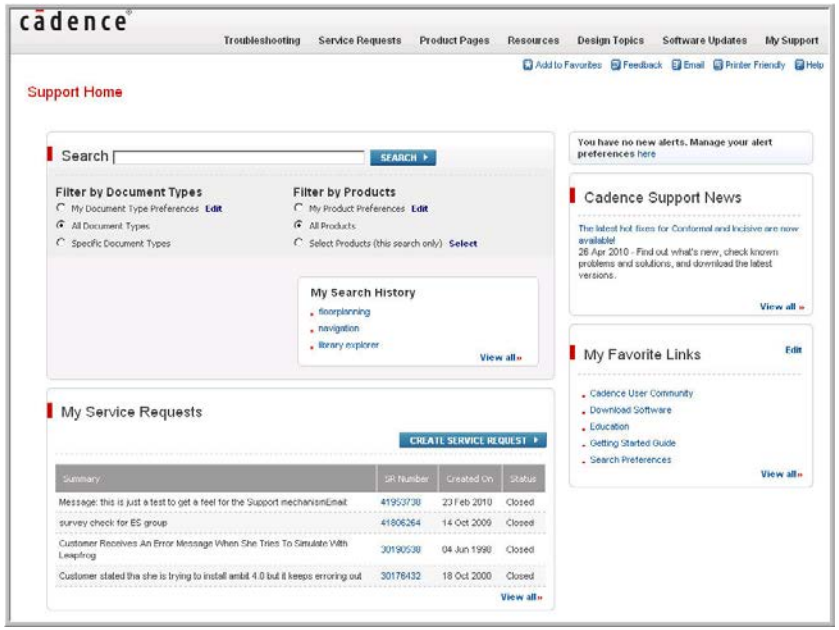
- Cadence Help gives you access to the Cadence online product documentation system.
- Documentation for each product is included automatically when you install the product. Documents are available in both HTML and PDF format.
- The Library window lets you access documents by product family, product name, or type of document.
- You can access Cadence Help from
  - The graphical user interface
    - The Help menu in windows
    - The Help button on forms
  - The command line: **cdnshelp**
  - The Cadence Online Support system (if you have a license agreement)



cadence

Cadence Online Support (COS)

- Cadence Online Support (http://support.cadence.com) is a web site that gives you access to support resources, including
  - An extensive knowledge base with
    - User guides
    - Reference manuals
    - Design topics
    - Frequently asked questions
    - Known problems and solutions
    - White papers
    - Application notes
  - Software updates for Cadence products
  - Access to Cadence customer support engineers
- Register now and take advantage of the many benefits of Cadence Online Support. If you do not have an active account, send email to *COS_Registration@cadence.com* and one will be set up for you.



To view the demo of Cadence Online Support, click one of the links below:

Online (iLS or Virtual)

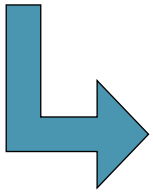
Classroom

Using Cadence Online Support

Access Cadence Online Support

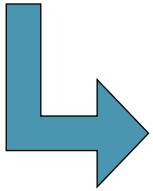
support.cadence.com

If you don't find a solution at the online support site...



Submit Service Request

From the online support site, fill out the Service Request Creation form to...

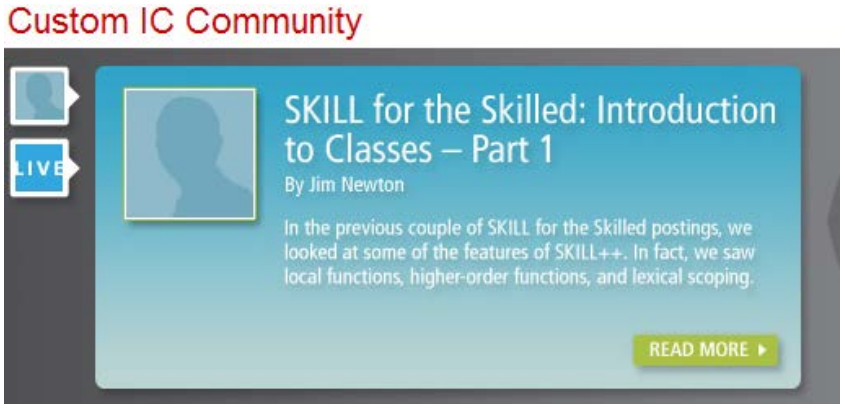


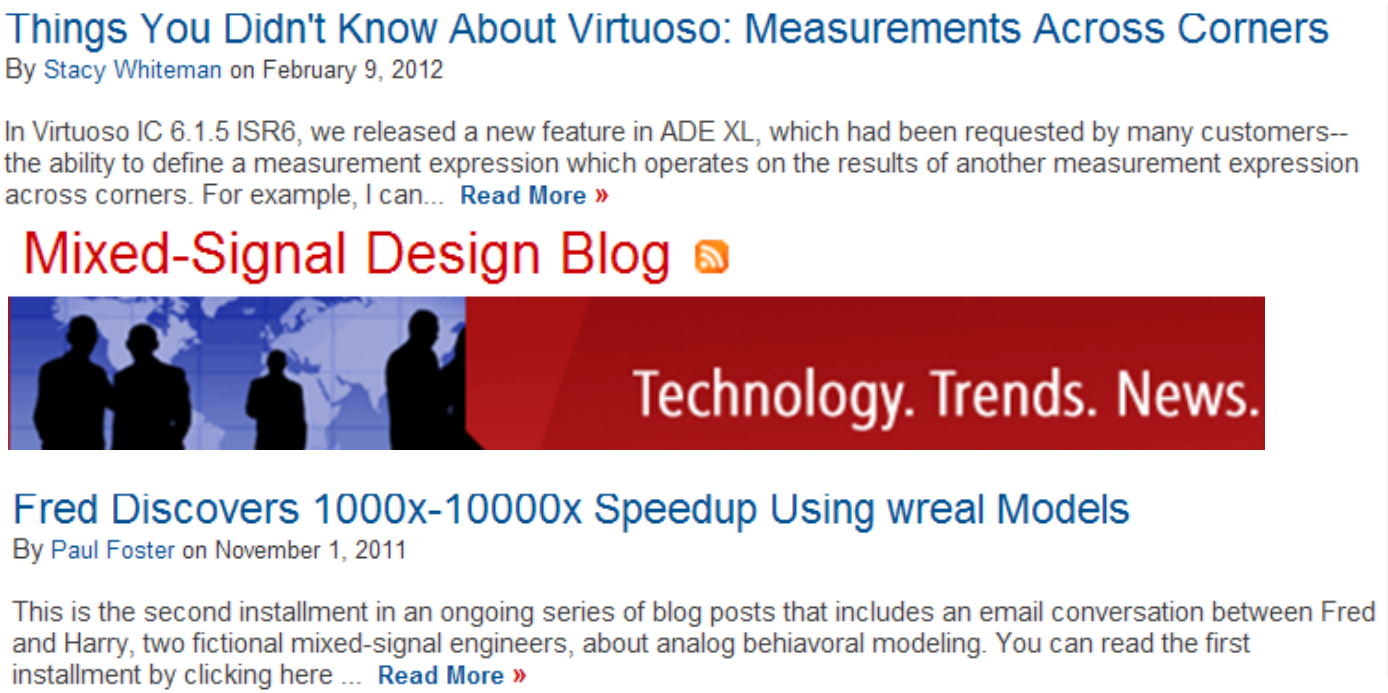
Receive Customer Support

If your problem requires more than customer support, then it is escalated to R&D for a solution.

Cadence Online Communities

- Stay connected by visiting resources, such as blogs and forums.
  1. Go to: http://www.cadence.com/community
  2. Select your area of interest.





cadence

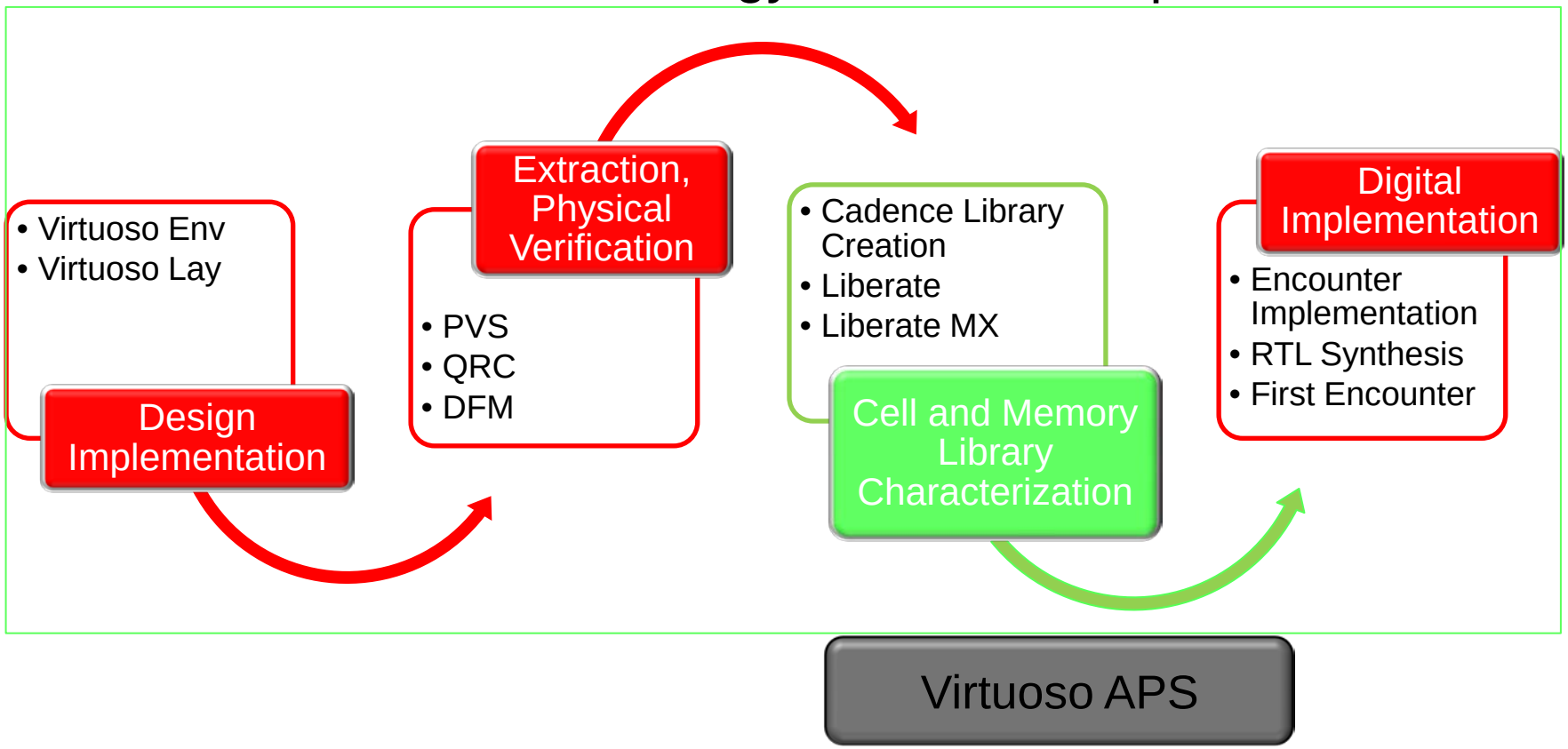
Module 3: Cadence Characterization Product Overview

Module Objectives

- In this module, you
  - Become familiar with the Characterization Product Offerings

Silicon Realization Completes the Flow

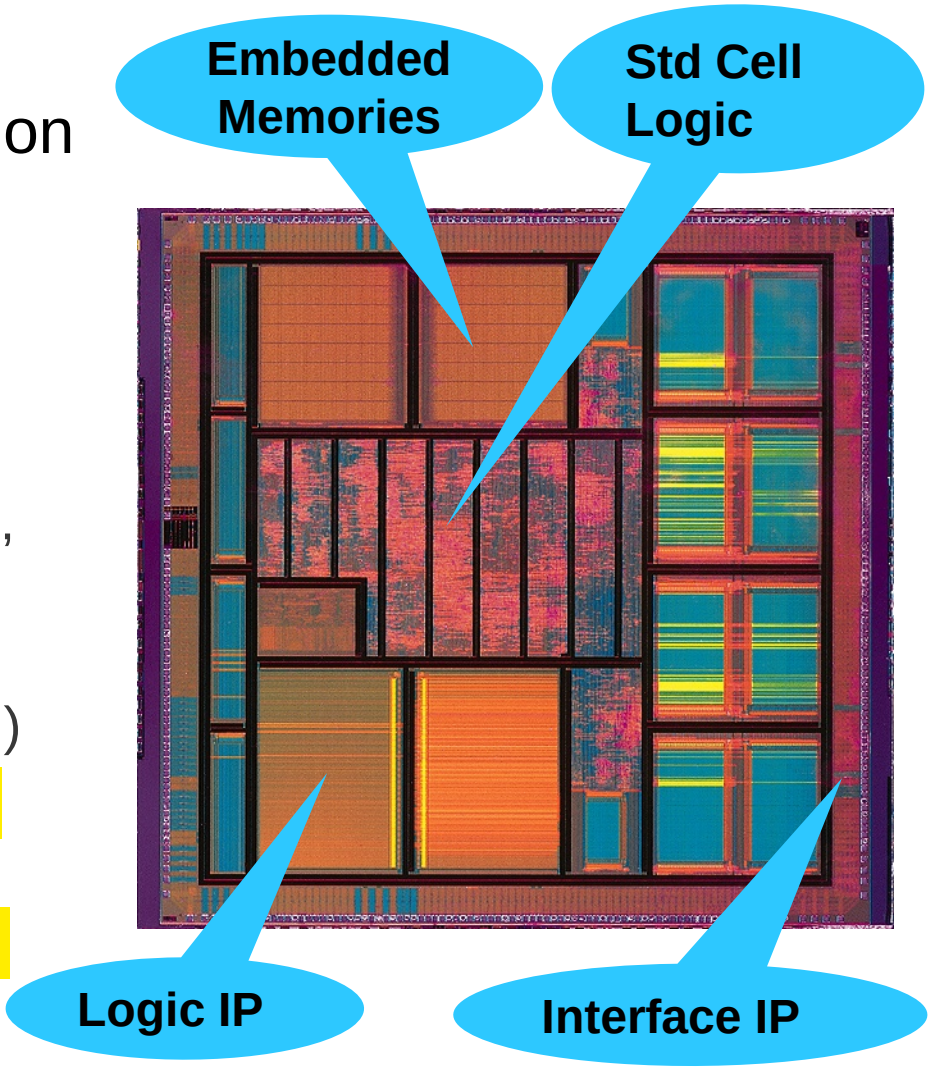
- Foundry certified PDKs start the process.
- Silicon Realization technology is used for implementation.



cadence

Cadence: Characterization Vision

- The Cadence Characterization vision includes “Highly efficient and automated electrical view creation and validation for all IP blocks.”
  - Logic and I/O cells (GPIO, PCI, SSTL, PECL etc.)
  - Embedded Memory (SRAM, ROM, Register files, CAM, etc.)
  - Custom digital blocks (custom cells, datapath, cores etc.)
  - Interface IP and analog blocks (USB, Serdes, DDR, etc)



cadence

Characterization Technology Portfolio

Liberate

New

Virtuoso Liberate	Virtuoso Liberate MX	Virtuoso Liberate LV	Virtuoso Variety	Virtuoso Liberate AMS
Standard Cells and Complex I/Os	**Memory and Custom Blocks**	**Library Validation**	**Process Variation Modeling**	**Mixed Signal Characterization**
• Ultra-fast library characterization • Advanced timing, power and noise models • CCS, ECSM, NLDM, NLPM, CCSN, ECSMN	• Unique “dynamic partitioning” technology for optimal runtime • Timing constraints and current source models for timing and noise	• Comprehensive validation system • Library function equivalence and data consistency checking • Revision analysis • Timing and power correlation	• Generates libraries that can be used by multiple SSTA tools • Local and global process variation • AOCV tables	• One step .lib generation for timing, power, leakage and noise

Inside View: Patented technology for generating and optimizing characterization stimulus

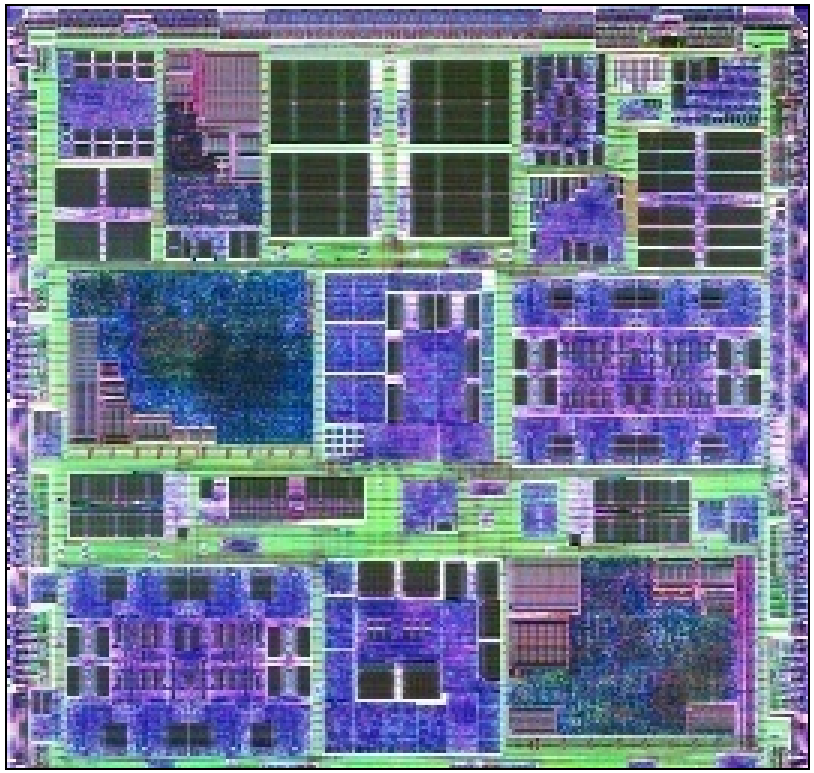
Spectre API Integration: 2-3x Improved throughput over command-line use model

Module 4: Introduction to Memory Characterization

Module Objectives

- In this module, you
  - Understand standard memory architecture
  - Understand what arcs are needed in a memory lib file and how they need to be measured

Why is Memory Characterization Important?



Processor with over 300 Memory instances

- Designs now have many distinct memory instances that need an abstract electrical models to enable the SoC Design flow –> synthesis, Place & Route, STA etc
- Memories instances consume much of the on-chip power hence are often subject to voltage scaling invalidating pre-existing models created by a memory compiler
- 40nm or below requires characterization with True-Spice accuracy to model transistor stress and coupling and support for advanced ECSM/CCS models

Who has Memories that need to be Characterized?

- The customer will possess a memory design that will be used as an IP block within a chip
  - Some memories are purchased from a library vendor (Virage, ARM, TSMC, etc)
    - The customer may require more PVTs than is included from the vendor
    - The customer may have limited knowledge of the internal workings of the memories
  - Others are developed in house
    - The customer will have knowledge of the internal workings of the design

Memory Architectures

- Liberate MX is a flexible tool that has successfully characterized a wide range of embedded memory designs
  - Single Port SRAM
  - Dual Port SRAM
  - Pseudo Dual Port SRAM
  - ROM
  - CAM/TCAM
  - CAMRAM
  - FIFO

Memory Architectures

- Due to its flexibility, Liberate MX is expected to be able to characterize other memories and macro blocks. However, since there is a lack of experience with these types, any evaluation will be more complex than an evaluation using the previously mentioned architectures.
  - DRAM
  - Non Volatile Memory
  - Large Digital Blocks

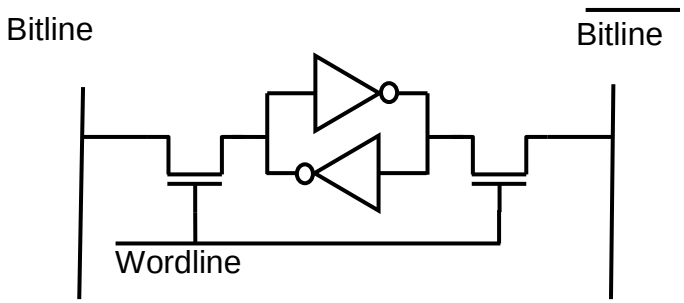
Memory Types

- Single port SRAM (single port bitcell)
  - Each operation will either read or write to one address location
- Dual port SRAM (dual port bitcell)
  - Each port is independently clocked and has its own address and read or write operation selected
- ROM (rom bitcell)
  - Each operation will read from an address location
- Pseudo Dual port SRAM (single port bitcell)
  - There is only one clock for both ports and a cycle consists of a read and a write to separately specified addresses
- CAM (single port cell with additional logic)
  - Write operation is similar to SRAM. Match operation consists of searching for a particular content and returning the address
- CAMRAM (CAM cells and SRAM cells)
  - Similar to CAM. Instead of returning address, reads that address from an SRAM

Memory Bit Cells

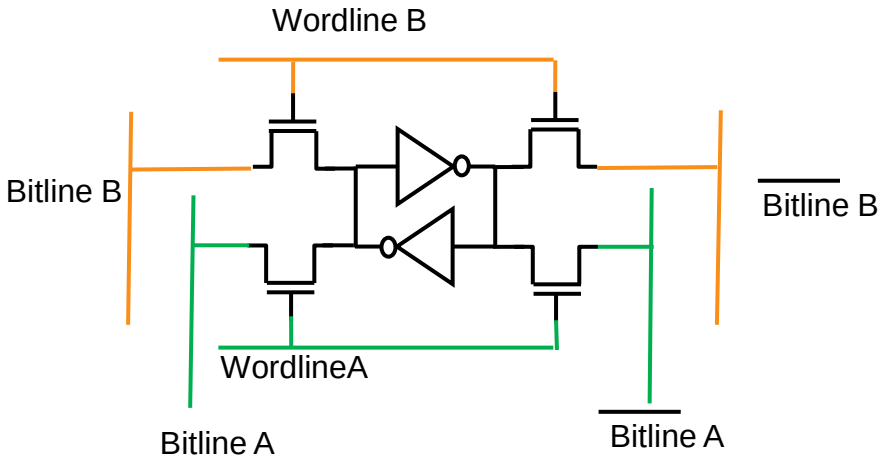
- The bit cell is the primary storage cell of the memory
- 6T (transistor) bitcells are used for single ports and pseudo dual ports

set_var mx_corecell "single_port"



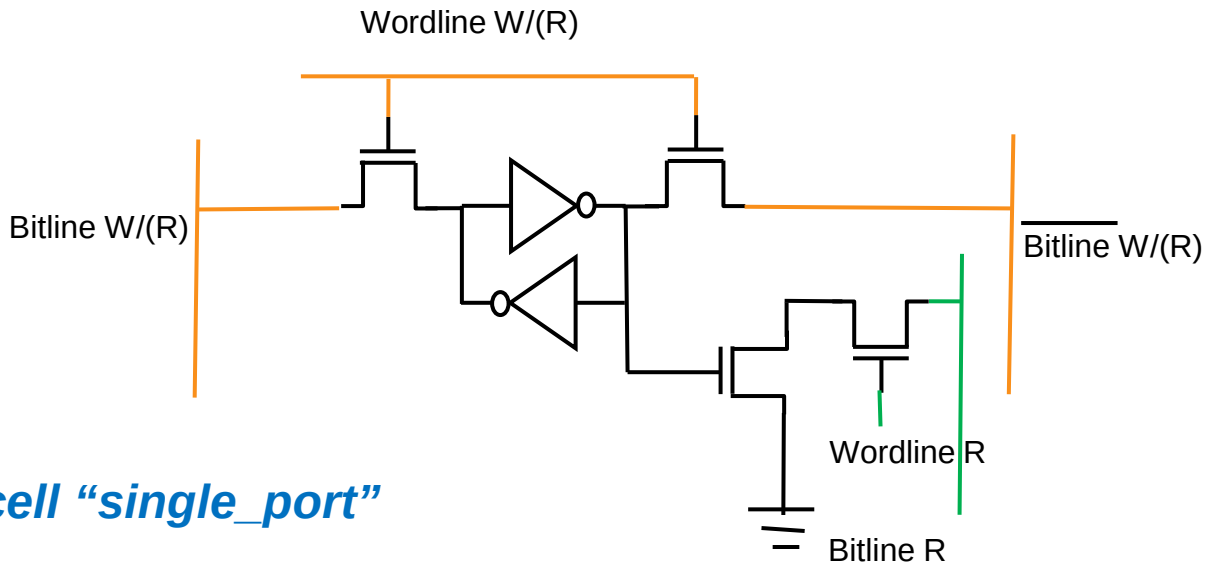
- 8T bitcells are used for dual ports where both ports read and write

set_var mx_corecell "dual_port"



Memory Bit Cells

- A different 8T bitcell may be used for dual ports with dedicated read and write ports



set_var mx_corecell "single_port"

- ROM, CAM, multiport and some custom memories use other bitcell designs

Memory clocking schemes

- **Self timed**
  - Will generate an internal clock signal from the rising edge of clock
  - The internal clock will control the start and end of the wordline and bitline transitions
  - Internal clock timing might be programmable
  - Internal clock termination will be a result of an internal self time loop
    - Might use dummy bitcells to mimic the delay in the actual operation
- **Level triggered**
  - Wordline and bitline start on the rising edge of clock, and terminate after the falling edge of clock

Memory size description

- **Memory configurations are referred to by several physical parameters**
  - **Word count**
    - The total number of addresses in the memory
  - **Bit count**
    - The number of data bits (IOs)
  - **Column Mux**
    - Physical option to describe the number of columns above each bit
    - Controls the aspect ratio of the instance
  - **Banks**
    - Number of separate vertical banks in the instance

Address Decoding

- **The address bus is composed of three parts**
  - **The Row Address**
    - Controls the selection of the wordline
    - Number of row addresses = log2 (number of rows)
  - **The Column Address**
    - Controls which bitline is selected
    - For each data bit, there will be a column mux that selects one column for write/read
    - Number of column addresses = log2 (column mux)
  - **The Bank Address**
    - Provides a level of mux between multiple banks of an instance.
    - Could control either bitline, wordline or both.

Address Space of the Memory Instance

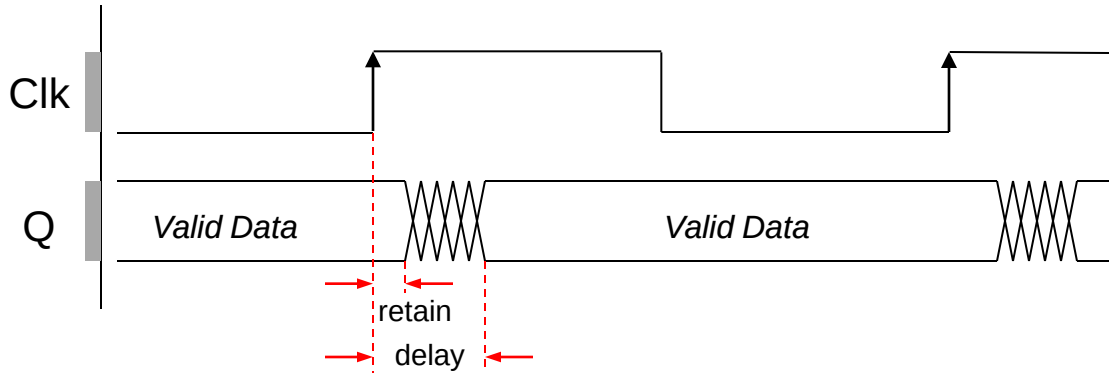
- If the memory has a 2^x number of rows, then the address = 1111…11 exists and it can be used to access the memory
- If the memory does not have a 2^x number of rows, then the top address location will need to be determined based on the address space
- Example:
  - Memory has 12 rows (4 row address bits)
  - Memory has column mux of 4 (2 column address bits)
  - The row address is the most significant portion of the address bus [5:2] and the column address is the lower part [1:0]
  - In this case, 111111 is not a valid address in the memory. Possible worst case addresses are 101111 or 011111

What is memory characterization?

- The SoC implementation flow requires a .lib file containing electrical information as follows
  - Pin capacitance on input pins
  - Setup and hold for input pins constrained by a clock
    - Includes data for pin and clock slews
  - Delay and retain from clock to output pins
    - Includes data for clock slews and output loads
  - Hidden power (power consumed when input switching doesn't cause the output to switch)
  - Leakage
  - Noise (immunity on inputs, holding strength on outputs)

Measuring Delay and Retain

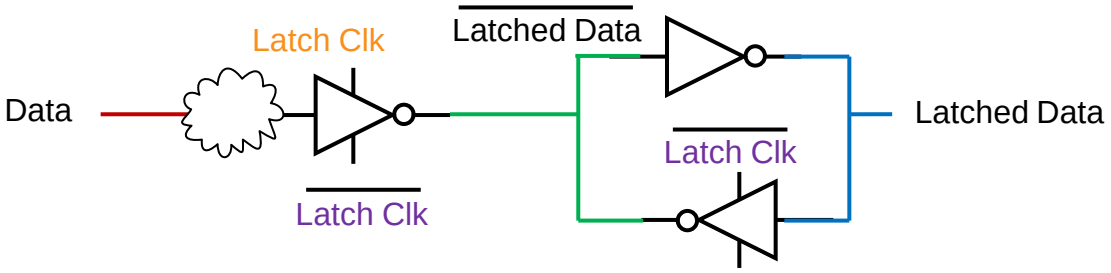
- **Delay** is the time that the new data is available on the data out bus after the clock rises.
  - Measured as the worst case time for data out to switch after clock
- **Retain** is the time that the previous data is still available on the data out bus after the next clock rises.
  - Measured as the best case time for data out to switch after clock
- In order to measure both delay and retain, it is important to simulate both best and worst case paths

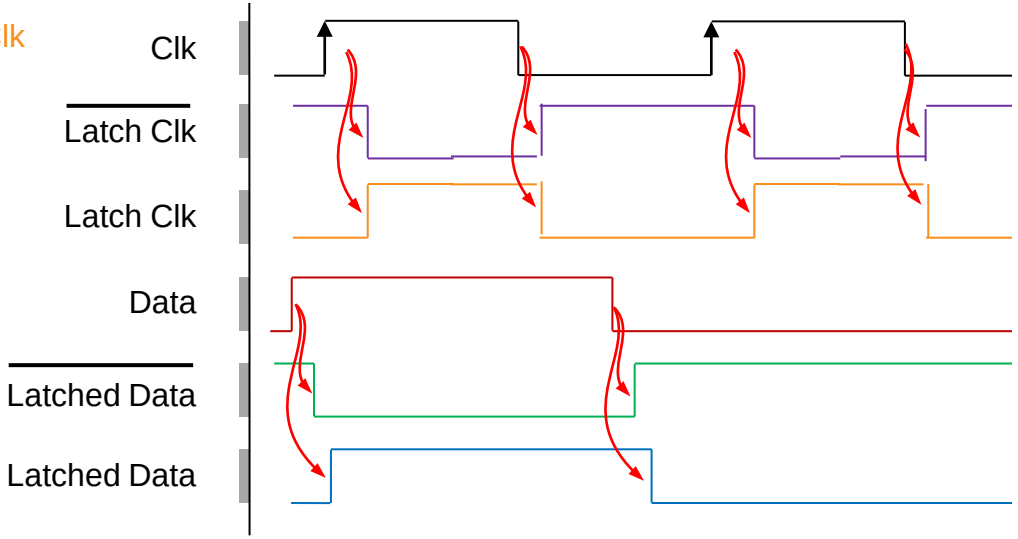


Virtual Rails (Optional)

- Virtual Rails allow for an internal power supply that might be controlled by external pins
- This is usually done for leakage current reduction
- i.e. “vdd_array”
- There may be timing constraints associated with entering and exiting power down modes for the virtual rails

Measuring Setup & Hold – Level Triggered Case





Setup Time = Max (Data -> Latched_Data, Data -> $\overline{\text{Latched\_Data}}$) – Min(Clk -> Latch_Clk, Clk -> $\overline{\text{Latch\_Clk}}$)

Hold Time = Max (Clk -> Latch_Clk, Clk -> $\overline{\text{Latch\_Clk}}$) – Min(Data -> Latched_Data, Data -> $\overline{\text{Latched\_Data}}$)

Measuring timing (delay/retain, setup/hold, etc)

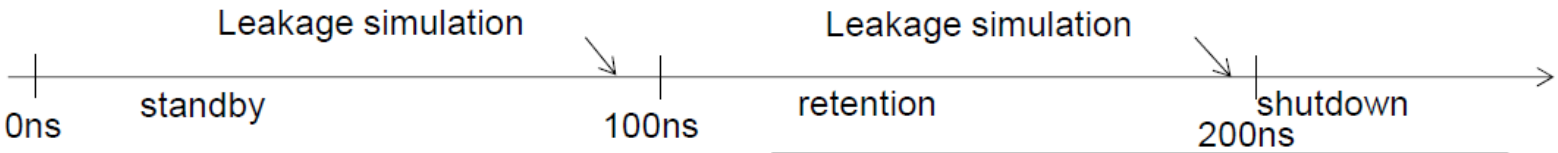
- **XPS command with full instance netlist** for partitioning
  - “spectre -64 +spice +xps +cktpreset=sram”
- **APS command with partition netlist** for timing characterization
  - “spectre -64 +spice +aps”

Measuring Hidden and Output Power

- **Hidden power** is characterized by performing the following operations on the memory
  - Write cycle (power is assigned to clock)
  - Read cycle (power is assigned to clock)
  - Toggle input pins
  - To get the dynamic component, static current is subtracted
- **Output power** is defined as the power consumption of the output switching
  - It is usually measured as the difference between a read operation where output switches and the same operation where output is stable.
- **No partition**
- **XPS command** for power characterization
  - “spectre -64 +spice +xps=s3 +cktpreset=sram_pwr”

Measuring Leakage power

- **No partition**
- **XPS command** for power characterization
  - "spectre -64 +spice +xps +cktpreset=sram_pwr"
- **LEAKI option of XPS** will interrupt the transient simulation and conduct a DC analysis to determine leakage of v-source being probed.
  - .options leaki_times=[99n 199n]

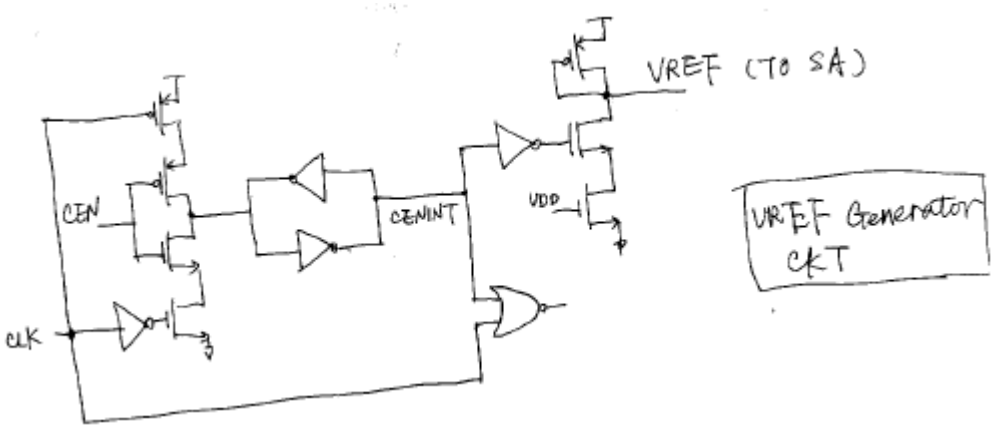


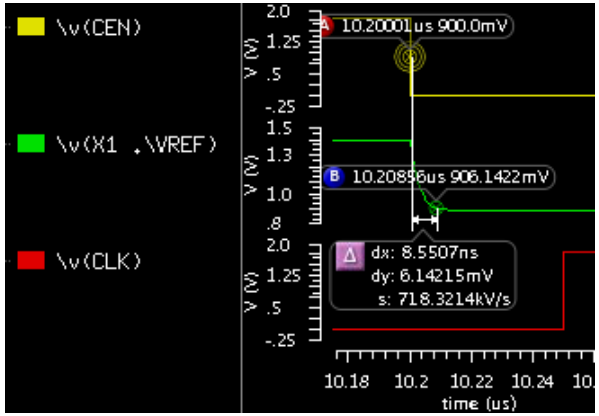
- **'*.leaki' file reports all leakage result**

```
i(VVDDAR) : t= 9.9e-8     val= -1.2377e-10
i(VVDDPR) : t= 9.9e-8     val= -8.725e-10
i(VVDDAR) : t= 1.99e-7      val= -9.7988e-11
i(VVDDPR) : t= 1.99e-7      val= -8.1215e-11
```

Constraints for Power Modes

- Entering and exiting power saving modes can affect the voltage level of virtual supply nodes
  - Setup and hold time for entering power mode should guarantee that the previous normal operation will not be corrupted by the power mode
  - Setup time for exiting power mode should guarantee that the circuit is fully ready for the next normal operation
    - Virtual rails should be pulled back up to full rail
    - This timing is not fully automated inside Liberate MX and will need to be supplemented with **define_measure**





Constraints for BIST Modes

- The BIST pin will generally control an input mux selecting between regular and test mode inputs.
- These inputs will usually be latched after the mux
- To measure setup and hold times for BIST, the regular and test pins should be held in opposite states and the BIST pin should be used to control the propagation of the inputs to their corresponding probe locations

When Conditions

- Typically a Memory lib file will have multiple “when” conditions for several arcs
  - Leakage current will be measured for multiple power saving modes
  - Delay will be measured for multiple timer programming settings
  - Dynamic currents will be measured for multiple values of write enable and timer programming
  - there could be other “when” states, depending on design,

```
leakage_power () {
  value : 55906.2;
  when : "!en&!banken&dead";
  related_pg_pin : vdd0;
}
leakage_power () {
  value : 55906.2;
  when : "!en&!banken&dead";
  related_pg_pin : vss0;
}
leakage_power () {
  value : 66487;
  when : "en&banken&!dead";
  related_pg_pin : vdd0;
}
leakage_power () {
  value : 66487;
  when : "en&banken&!dead";
  related_pg_pin : vss0;
}
```

```
pin (ph1) {
  clock : true;
  direction : input;
  related_ground_pin : vss0;
  related_power_pin : vdd0;
  max_transition : 0.24;
  capacitance : 0.00500156;
  rise_capacitance : 0.00480985;
  rise_capacitance_range (0.00428306, 0.00480985);
  fall_capacitance : 0.00500156;
  fall_capacitance_range (0.00442092, 0.00500156);
  internal_power () {
    when : "wren&en&banken&!dead";
    related_pg_pin : vdd0;
    rise_power (passive_power_template_5x1) {
      index_1 ("0.01, 0.025, 0.06, 0.12, 0.24");
      values ( \
        "1.84382, 1.84382, 1.84382, 1.84382, 1.84382" \
      );
    }
    fall_power (passive_power_template_5x1) {
      index_1 ("0.01, 0.025, 0.06, 0.12, 0.24");
      values ( \
        "3.27853, 3.27853, 3.27853, 3.27853, 3.27853" \
      );
    }
  }
```

Memory Compilers

Many memory instances are built from memory compilers

- Memory Compilers assemble the leafcells of the memory instance in order to create instances of a user selected size.
- Memory Compilers may provide estimations of timing and power data. This data is usually interpolated from the nearest simulated points.

Module 5:
Liberty Format Overview

Measurements made during characterization

Delay

- Non-linear Delay Model (NLDM)
- Composite Current Source (CCS - Synopsys)
- Effective Current Source Model (ECSM - Cadence)

Timing Constraints

- Setup & Hold
- Recovery & Removal
- Minimum Pulse Width

Power

- Switching
- Hidden
- Leakage

Pin Capacitance

- NLDM
- CCS Receiver Capacitance
- ECSM Capacitance

Measurements made during characterization (cont.)

Signal Integrity Models

- CCSN
- ECSMN

IBIS Models

- Power and Ground Clamp Current
- Pull-up & Pull-down IV Curves
- Rising and Falling Waveforms

Electromigration Models

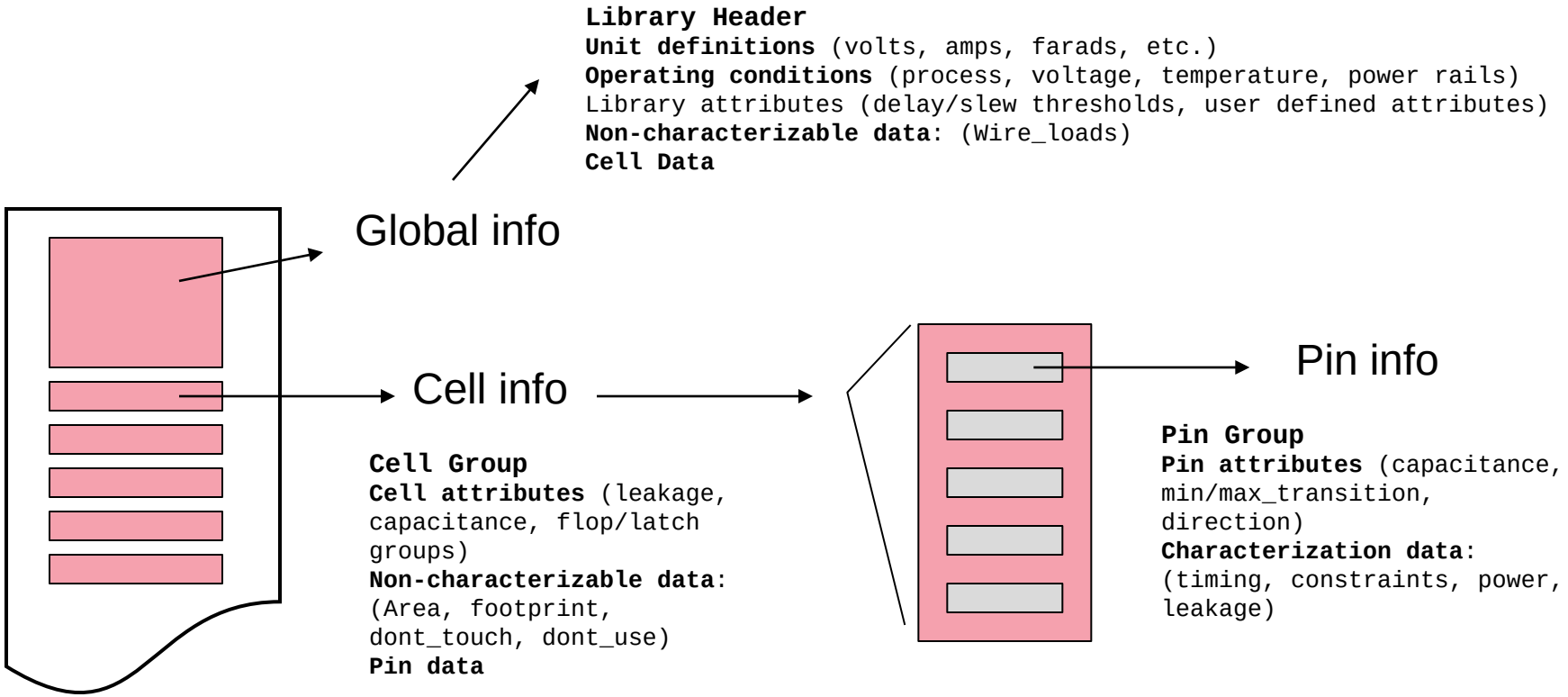
- Electromigration Maximum Toggle Rate

Not covered in this training

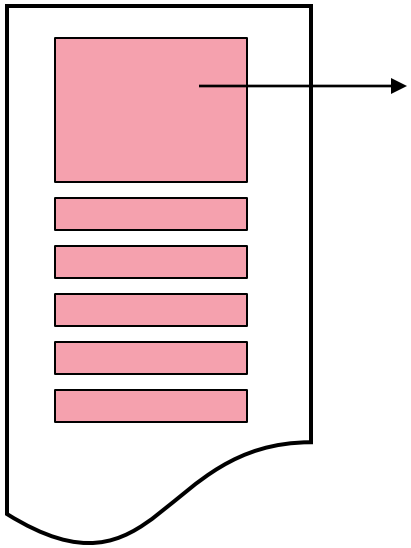
Quick Look at a Liberty Library

- Library Header
  - Library attributes
    - Voltage, current, capacitance, and time units definition
    - Slew/delay Thresholds
    - Operating conditions – process, voltage, temperature settings
    - Table templates for characterized data
- Cell definitions
  - Area, footprint, leakage attributes
  - Flip flop and latch groups (function definition)
  - Pin Definitions
    - Capacitance
    - Min/max_transition, min/max_capacitance
    - Direction (input, output, bidi)
    - Delay, slew, power, constraint data
    - Pin logic function

Birds-eye view of Liberty file



Example Liberty Header

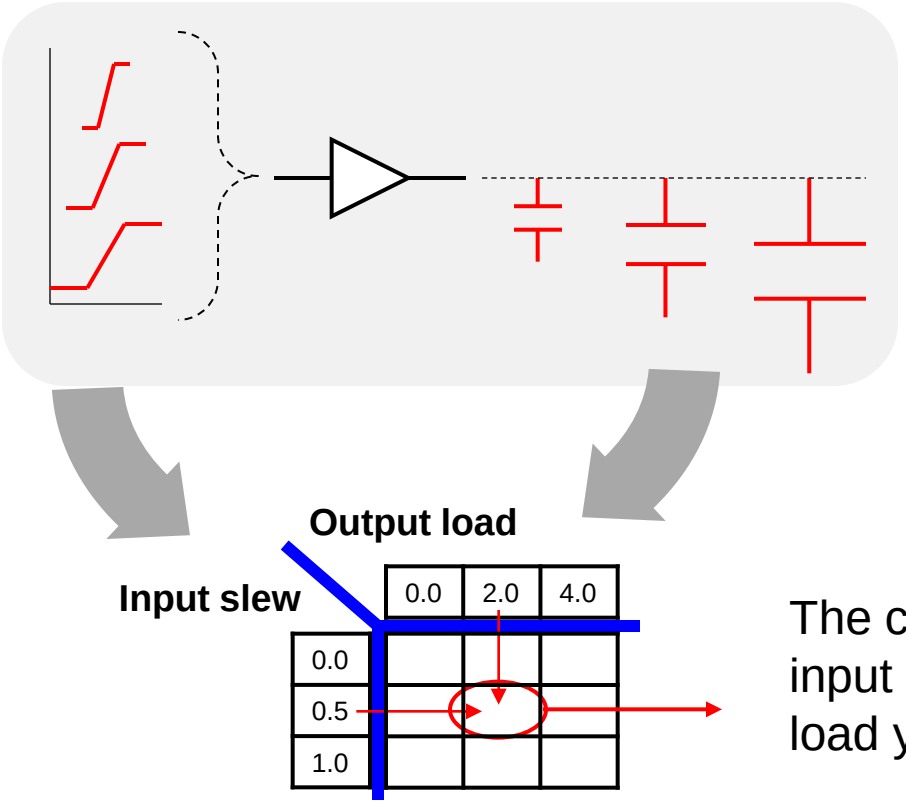


Global settings

```
library (acme_180_ccs) {
  delay_model : table_lookup;
  capacitive_load_unit (1,pf);
  current_unit : "1mA";
  leakage_power_unit : "1nW";
  pulling_resistance_unit : "1kohm";
  time_unit : "1ns";
  voltage_unit : "1V";
  nom_process : 1;
  nom_temperature : 125;
  nom_voltage : 1.08
  input_threshold_pct_fall : 50;
  input_threshold_pct_rise : 50;
  output_threshold_pct_fall : 50;
  output_threshold_pct_rise : 50;
  slew_lower_threshold_pct_fall : 20;
  slew_lower_threshold_pct_rise : 20;
  slew_upper_threshold_pct_fall : 80;
  slew_upper_threshold_pct_rise : 80;

  operating_conditions (PVT_108V_125C) {
    process : 1;
    temperature : 125;
    voltage : 1.08;
  }
lu_table_template (constraint_template_3x3) {
    variable_1 : constrained_pin_transition;
    variable_2 : related_pin_transition;
    index_1 ("0.100, 0.500, 1.0");
    index_2 ("0.100, 0.500, 1.0");
  }
  lu_table_template (delay_template_7x7) {
    variable_1 : input_net_transition;
    variable_2 : total_output_net_capacitance;
    index_1 (“0.100, 0.500, 1.00");
    index_2 ("0.025, 0.100, 0.20");
  }
```

Understanding look-up tables in Liberty

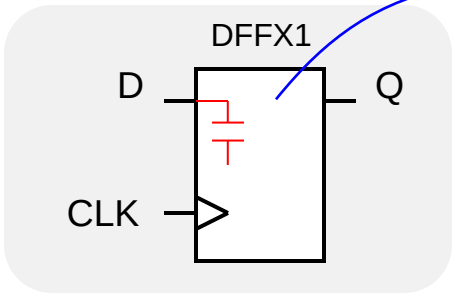
Series of input slews

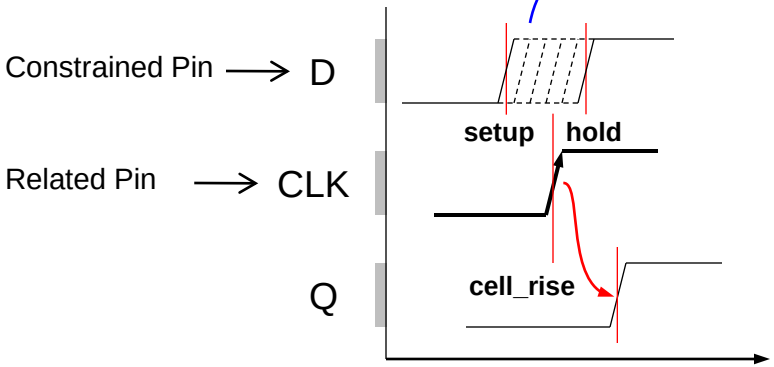


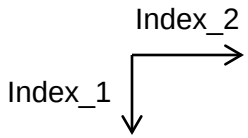
Series of output loads

The combination of an input slew and an output load yields a delay value.

Cell & Library entries – Timing and Constraints





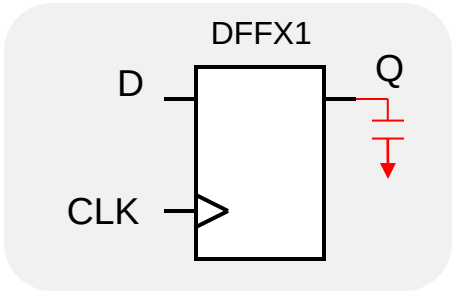


.lib

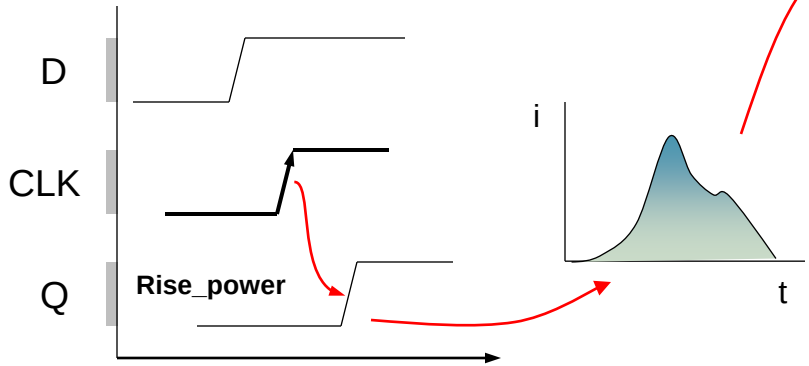
Partial entry

```
cell (DFFX1) {
  pin (D) {
    direction : input;
    max_transition : 1.5;

    capacitance : 0.00740386;
    rise_capacitance : 0.00740386;
    rise_capacitance_range (0.0062143, 0.00740386);
    fall_capacitance : 0.00583898;
    fall_capacitance_range (0.00572062, 0.00583898);

    timing () {
      related_pin : "CK";
      timing_type : setup_rising;
      rise_constraint (constraint_template_3x3) {
        index_1 ("0.25, 0.75, 1.5");
        index_2 ("0.25, 0.75, 1.5");
        values ( \
          "0.185383, 0.200818, 0.245979", \
          "0.301927, 0.312063, 0.358151", \
          "0.42611, 0.432496, 0.479517" \
        );
      }

  pin (Q) {
    direction : output;
    function : "IQ";
    max_capacitance : 0.6;
    timing () {
      related_pin : "CK";
      timing_sense : non_unate;
      timing_type : rising_edge;
      cell_rise (delay_template_5x5) {
        index_1 ("0.25, 0.5, 0.75, 1.25, 1.5");
        index_2 ("0.015, 0.05, 0.15, 0.3, 0.6");
        values ( \
          "0.517821, 0.652322, 1.02595, 1.58443, 2.70065", \
          "0.549747, 0.684235, 1.05788, 1.61636, 2.73261", \
          "0.567106, 0.701596, 1.07521, 1.63373, 2.75004", \
          "0.581625, 0.716122, 1.08978, 1.64826, 2.76458", \
          "0.582743, 0.717242, 1.0909, 1.6494, 2.76569" \
        );
      }
```

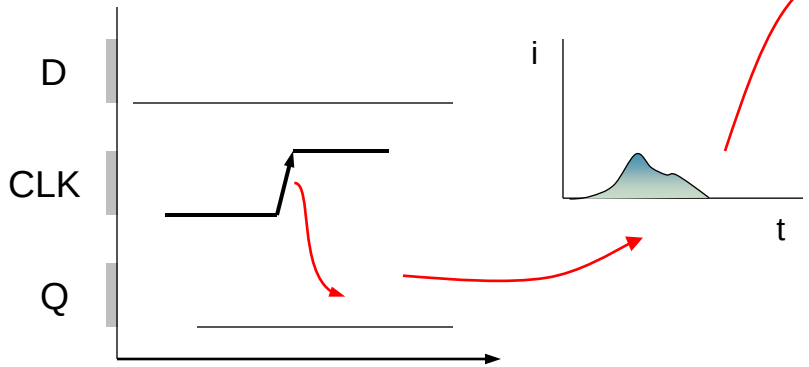
Cell & Library entries – Power



Switching Power →



Hidden Power →



.lib

```
cell (DFFX1) {                                   Partial entry

  pin (Q) {
    internal_power () {
      related_pin : "CK";
      rise_power (power_template_5x5) {
        index_1 ("0.25, 0.5, 0.75, 1.25, 1.5");
        index_2 ("0.015, 0.05, 0.15, 0.3, 0.6");
      values ( \
        "0.517821, 0.652322, 1.02595, 1.58443, 2.70065", \
        "0.549747, 0.684235, 1.05788, 1.61636, 2.73261", \
        "0.567106, 0.701596, 1.07521, 1.63373, 2.75004", \
        "0.581625, 0.716122, 1.08978, 1.64826, 2.76458", \
        "0.582743, 0.717242, 1.0909, 1.6494, 2.76569" \
      );
    }

  pin (CLK) {
    direction : input;
    internal_power () {
      rise_power (passive_power_template_7x1) {
        index_1 ("0.008, 0.02, 0.06, 0.12, 0.26, 0.54, 1.0");
        values ( \
          "0.024, 0.026, 0.027, 0.028, 0.030, 0.032, 0.035" \
        );
      }
```

Module 6:
Introduction to Liberate MX

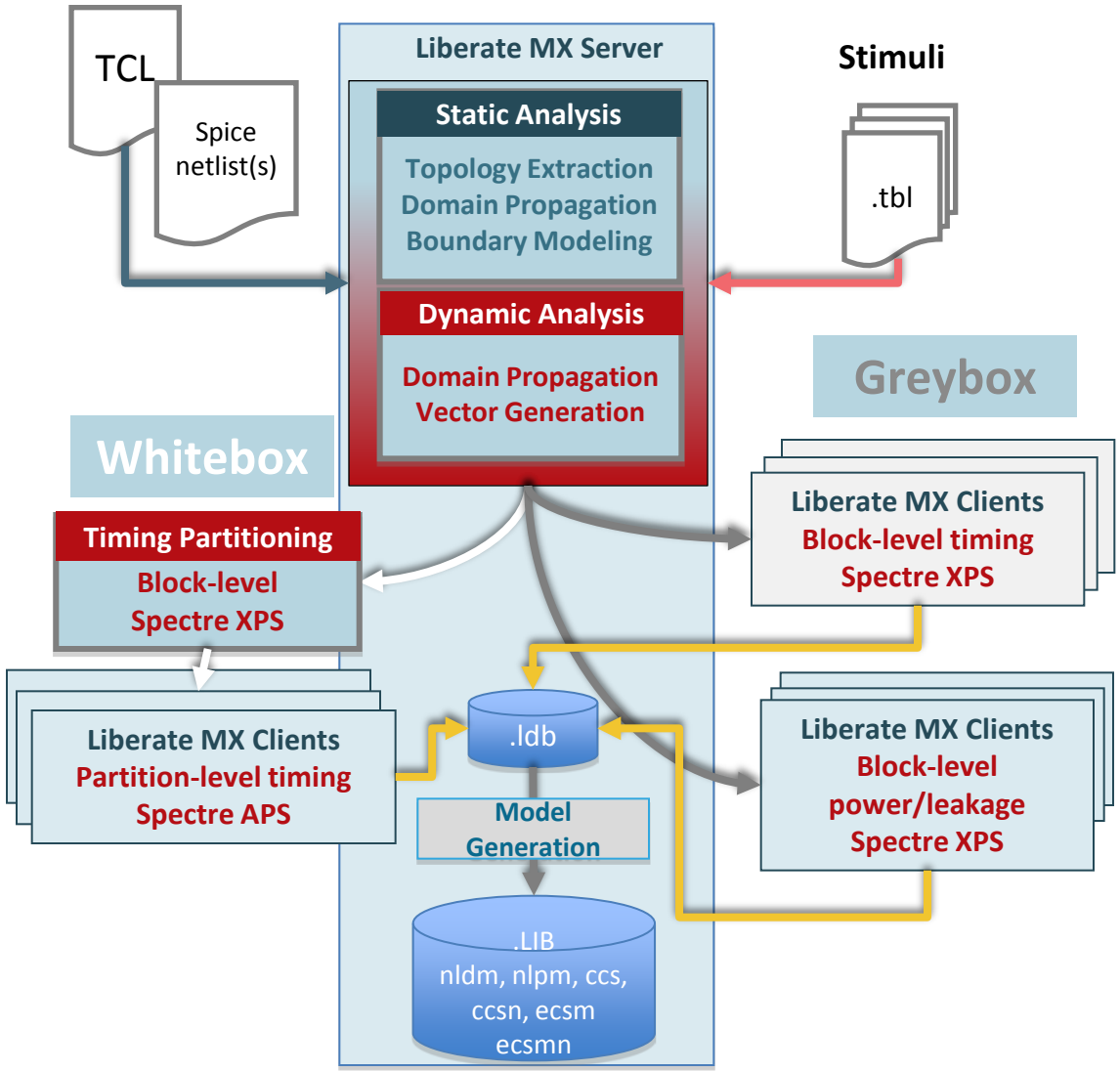
Module Objectives

- In this module, you
  - Will be introduced to the method and flow of Liberate MX

Virtuoso Liberate MX

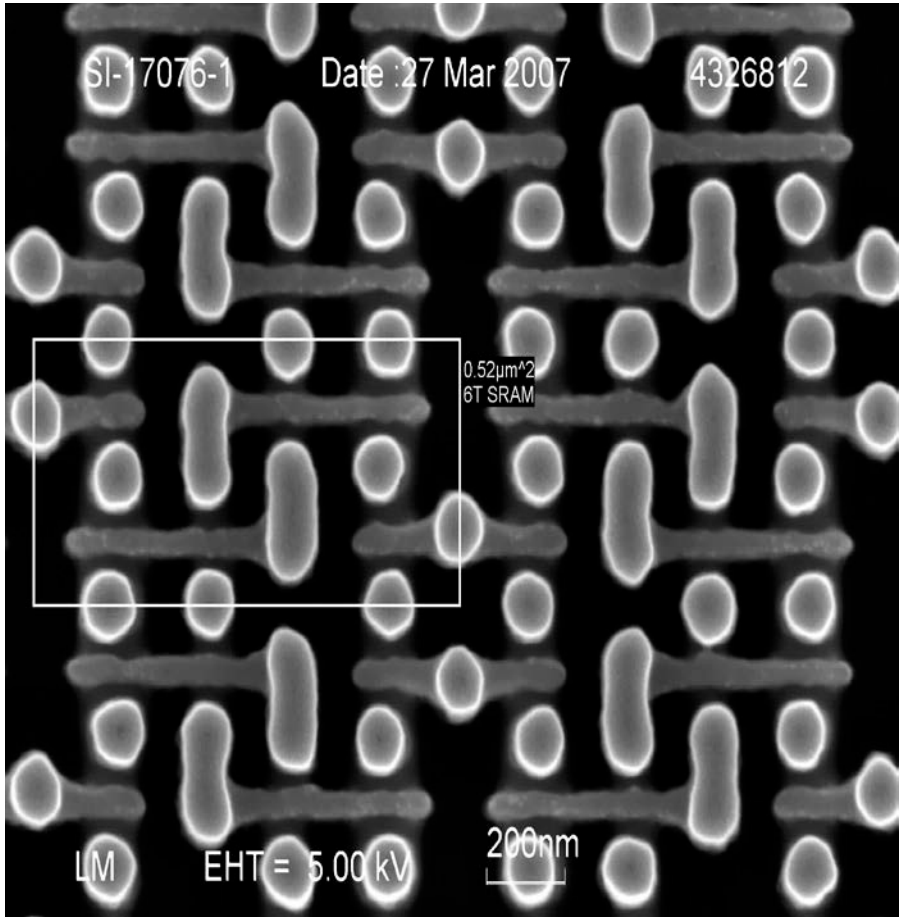
Cadence Memory Characterization Solution

- "Inside View"
  - Automatic identification of probes on key circuit elements using a mix of dynamic and static analysis
- "Dynamic Partitioning"
  - Creates run for each path and including all the necessary "active" side-paths
- Spectre XPS Fast-Spice simulator integration
  - Order of magnitude speed-up over existing solutions
- Accurate SPICE runs
  - Performed independently on decomposed partitions
  - Assembles results representing timing behavior of the original, larger instance



Liberate MX Features

- Automatic probing via static partitioning, clock propagation and topology recognition
- Block level fast SPICE - e.g. Spectre XPS for dynamic partitioning and power characterization.
- Dynamic partitions – delay, constraints - characterized with real SPICE –e.g. Spectre/APS
- Timing, power and noise models including CCS, ECSM.
- Custom measurements API –e.g for race margins



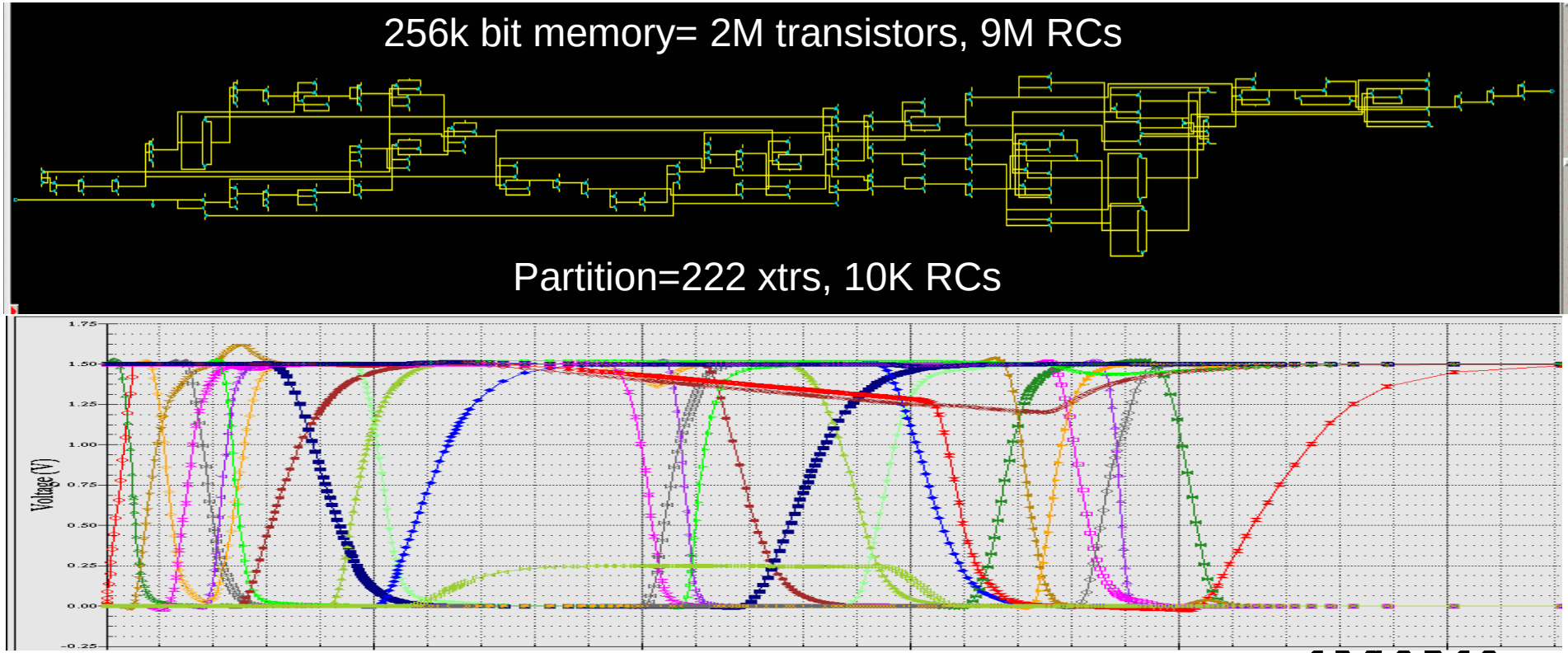
cadence

Clock Propagation & Automatic probing

- Partitioning via **xtr level traversal** and **topology independent feedback analysis** for latch / flop recognition on non-array logic
- **Core nodes and bit-lines identification** via pattern matching inside arrays – pattern specification through simple TCL commands.
- **Clock tree** propagation and identification
- Auto-probing used for constraint characterization

Liberate MX Features – Dynamic Partitioning

- Uses single run of Fast-Spice (one slew, one load) to simulate circuit activity and derive best and worst case paths
- Includes side path circuitry based on the simulation activity, yielding partitions with typically <1K transistors
- Accurate characterization for all slews and loads can be performed using True Spice



cadence

Liberate MX Features – Dynamic Partitioning

Examples

40 nm Low Power 256X8

- 84 unique partitions generated
- Average 357 xtrs per partition, Max 1140 xtrs, min 10 xtrs

Arc	Type	Xtr count
CLK-> Q	Delay	1140
DM ->Q R	Delay	32
DM ->Q F	Delay	76
AWT->Q	Delay	73
CLK->DM	Hold	705
CLK	MPW	423

32nm Low Power 1028X18

- 159 unique partitions generated
- Average 480 xtrs per partition, Max 959 xtrs, min 13 xtrs

Arc	Type	Xtr count
CLK -> Q	Retain	910
CLK->A	Setup	742
CLK->A	Hold	230
CLK	Min Period	959
CLK	MPW	146
A	Pin Cap	25

Sample Memory Types and Runtimes

Memory Type	Technology Node	size	Runtime (hrs – 7x7)
Single Port, split rail, low power SRAM	40 CMOS	128x32	0.56
Single Port, multi rail, low power SRAM	40 CMOS	128x32	0.57
		16384x64	1.75
Pseudo Dual Port (2RW), SRAM	40 CMOS	640x47	0.76
CAM	40 CMOS	128x37	0.61
ROM	40 CMOS	4096x128	0.42
OTP ROM	40 CMOS	128x36	0.60
Dual Port, low power, Register File	40 CMOS	32x8	0.50
Single Port SRAM, multi rail, advanced power mgmt, redundancy/repair, DFT, Scan, Retention Till Access	28 CMOS	256x16	1.2 (0.15 char)
		1024x128	3.16 (0.2 char)
Dual Port (2R2W), low power Register File	28 CMOS	512x32	1.4 (0.18 char)

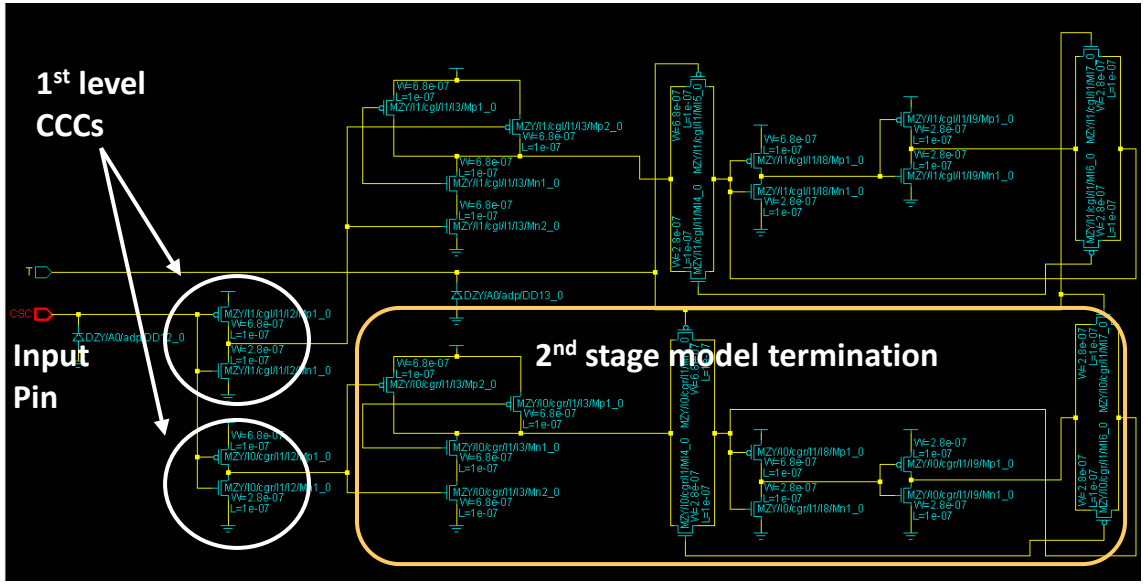
Liberate MX Accuracy

- Comparison of running a full custom memory instance using full spice (golden reference) , versus Liberate MX (Fast-Spice for “dynamic partitioning”, True-Spice for characterization)

Arc	Type	MX Value (ps)	Spice Value (ps)	Diff%
CLK->Q F	delay	470.3	474.1	-0.81%
CLK->Q R	delay	469.2	474.3	-1.09%
CLK->Q F	retain	458.7	465.2	-1.42%
CLK->Q R	retain	459.8	466.2	-1.39%
CEN F	setup	91.8	90.8	1.09%
CEN R	setup	94.8	95.5	-0.74%
CEN F	hold	99.7	100.5	-0.80%
CEN R	hold	61.8	61.7	0.16%

Memory CCS Models

- Output current/voltage waveform captured with accurate Spice simulation of *clock-to-output* partition
- Input capacitance, input noise model and output noise model characterized with boundary circuitry

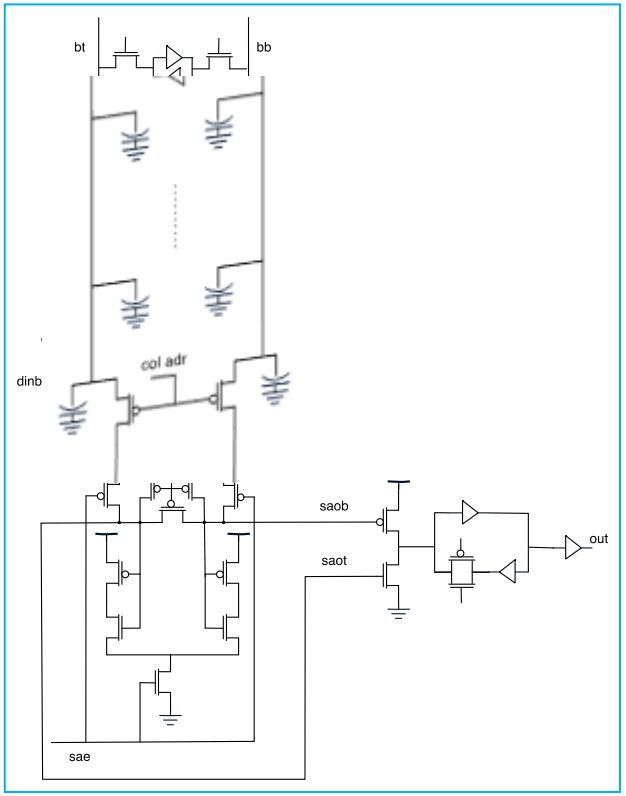


Input pin partition

- Current source noise: dc current, output voltage, propagated noise
- Current source capacitance table

Equivalent cap/load calculation : Side Circuit Inclusion

- Required for accurate SI analysis
  - Must account for dynamic effects such as AC coupling, charge sharing, proximity effects
- Determine RC aggressors based on coupling and activity
  - User control of "activity" level and "coupling" threshold
- Account for non-active elements (off)
  - MX uses “True-SPICE” simulation to calculate equivalent capacitance loads



Capacitive loads for inactive elements

Modeling with Simulation Vs Static Timing

Simulation With Vectors

- True Spice accuracy
- Dynamic power and leakage
- Current models for timing, noise
- Easy to setup with compiler templates; moderate setup for custom memories
- Easier to verify – reuse of existing test-benches
- Fast run-time that can be fully distributed, scalable with block size
- All data is “real”

Static Analysis Without Vectors

- ± 10% of True Spice
- Timing only
- Significant user configuration effort (internal latches identification, clock propagation, timing constraints)
- Pre-characterization needed for non-digital components
- Long run-times, increases non-linearly with memory size
- False paths leading to pessimistic models

Modeling with Liberate MX Vs Manual Methods

Liberate MX

- Liberate MX automatically creates partitions based on dynamic activity
- Liberate MX does not need user intervention to create partitions
- Liberate MX can automatically run in full instance (no partitioning mode) through a variable setting
- Liberate MX automatically identifies all probe locations and selects the worst case

Manual Methods

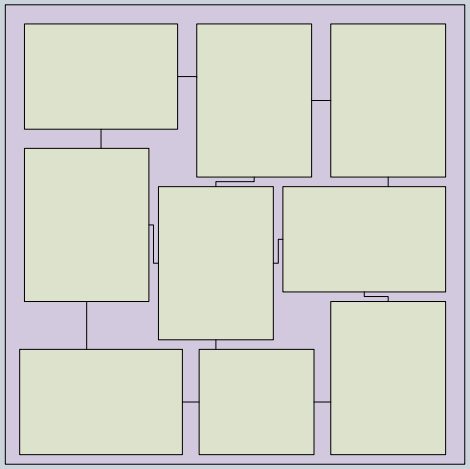
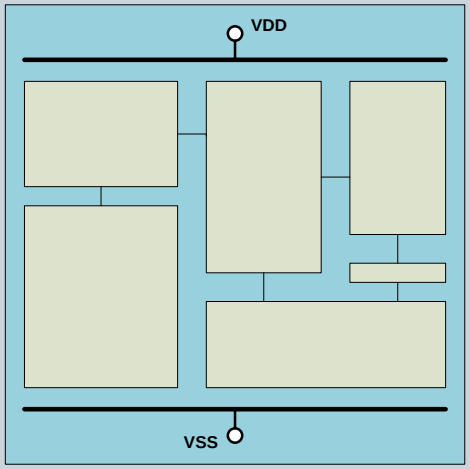
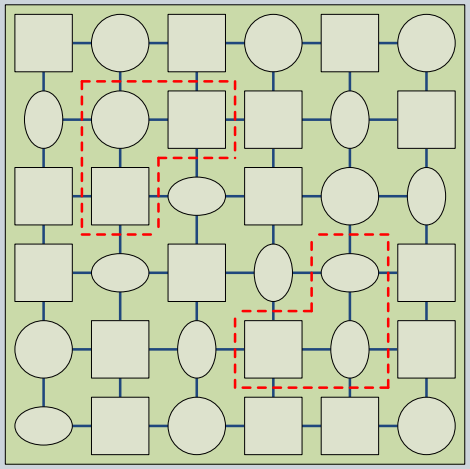
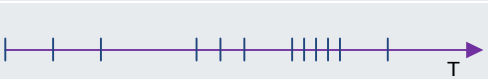
- Manually created critical paths are error prone
- Critical Paths are time consuming to create
- Difficult to verify critical paths because it requires supporting 2 parallel methodologies
- User must locate all relevant probing locations

Module 7: The benefits of using Liberate MX and Spectre XPS together

Module Objectives

- In this module, you
  - Will be introduced to the Spectre XPS simulator
  - Will see the benefits of running Liberate MX and Spectre XPS together

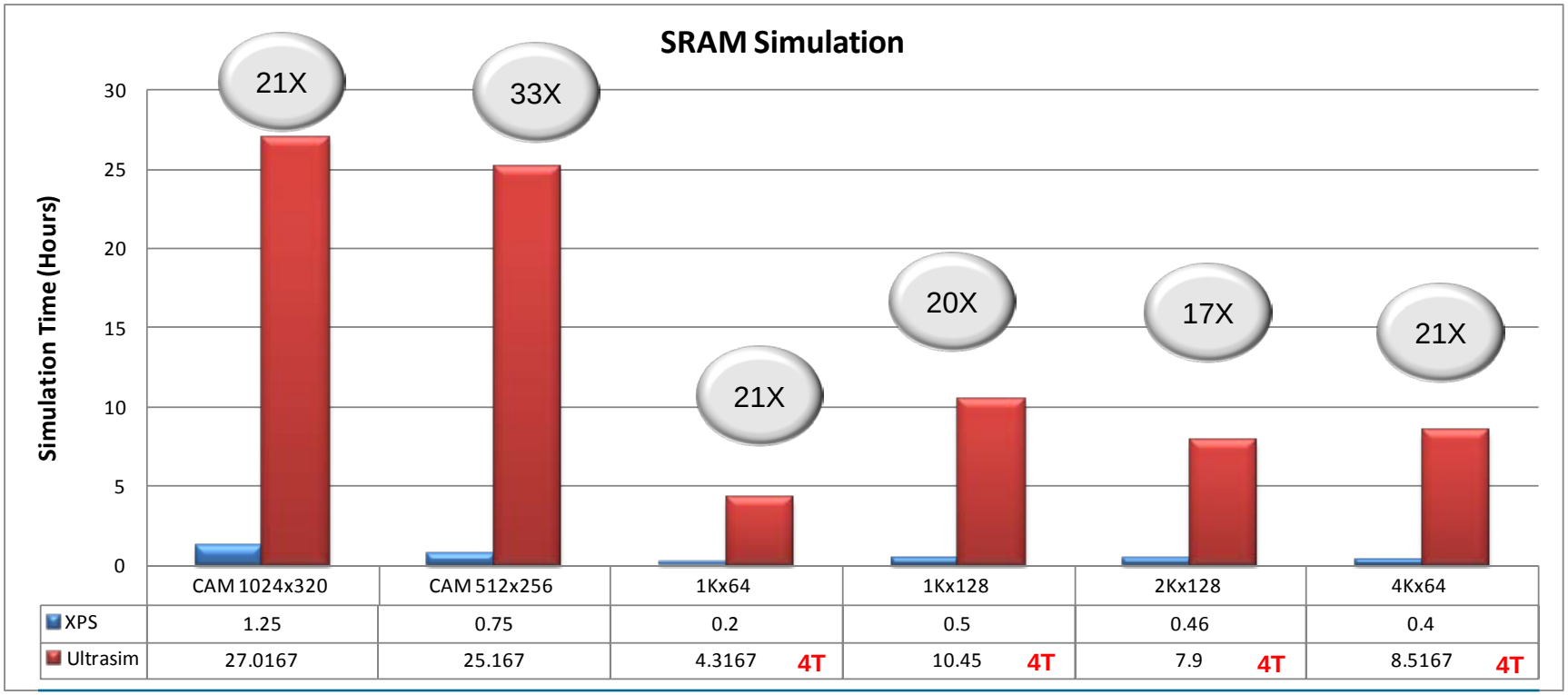
Spectre XPS for Power Gating, Parasitics
A Technology Comparison

APS	Ultrasim	XPS
		
		
• Precise matrix partitions, parallelized	• ~~Channel connected partitions~~ Power gate, voltage regulators	• Arbitrarily small partitions
• ~~Entire circuit solved together~~ Slowdown as circuit size increases	• Each partition solved individually	• Cluster of partitions solved together
• Single-rate, limited multi-rate possible	• Event-driven multi-rate	• Event-driven multi-rate
• Precise analytic device model	• Table model	• Efficient table model
• Flat simulation	• ~~Isomorphic and hierarchic simulation~~ Post-layout parasitics	• Flat simulation

cadence

Spectre® XPS
Performance

28nm SRAM Flat Simulation (CPU Time)

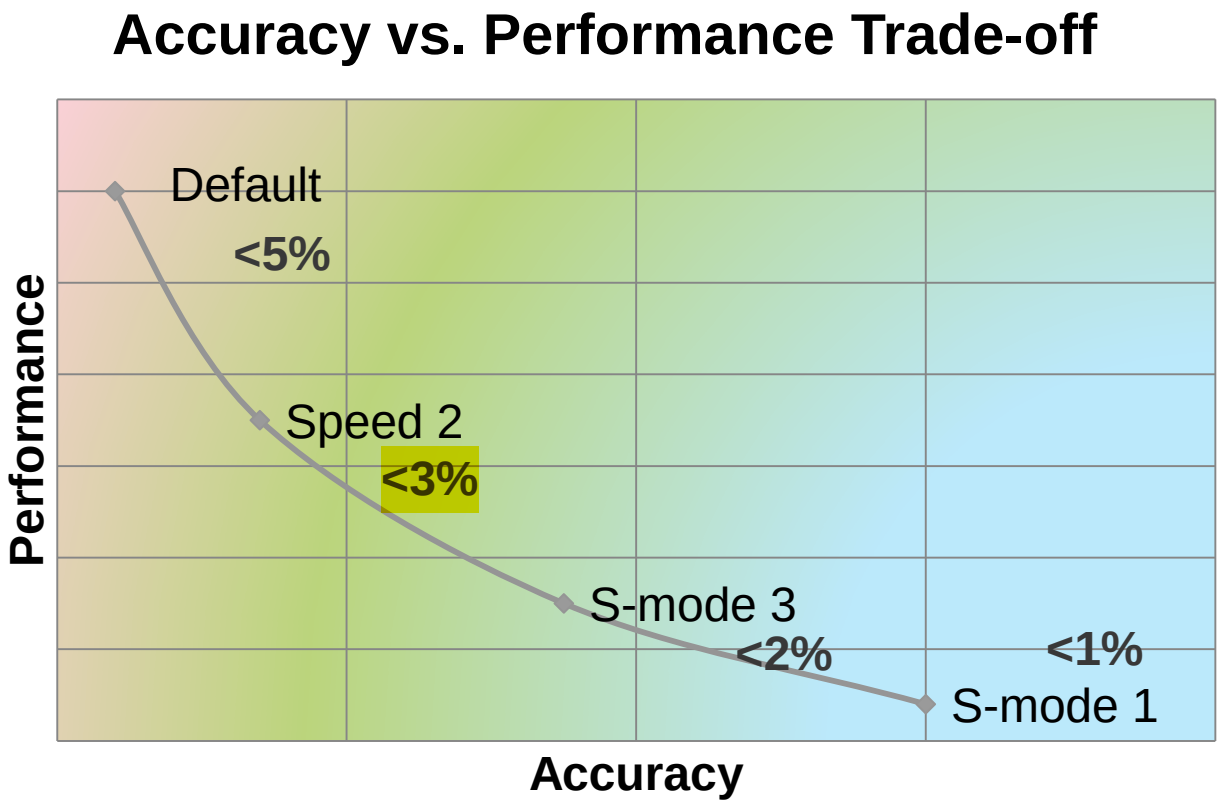


	CAM 1024x320	CAM 512x256	1Kx64	1Kx128	2Kx128	4Kx64
XPS	1.25	0.75	0.2	0.5	0.46	0.4
Ultrasim	27.0167	25.167	4.3167 4T	10.45 4T	7.9 4T	8.5167 4T

MOSFETs	**8.3M**	**3.5M**	**0.69M**	**1.3M**	**1.8M**	**1.7M**
Rs	**38.5M**	**17.5M**	**3.1M**	**5.95M**	**9.9M**	**9.8M**
Cs	**205M**	**87M**	**10.7M**	**20.6M**	**30.2M**	**30M**
MEM	**29GB**	**13GB**	**3.2GB**	**4.8GB**	**5.3GB**	**5.3GB**

cadence®

Spectre XPS
Scalability across performance and accuracy



Recommended stand-alone Use Models for SRAM

Timing: spectre -64 +spice +xps +speed=2 +cktpreset=sram

Power: spectre -64 +spice +xps=s3 +cktpreset=sram_pwr

Memory Characterization Throughput

Liberate MX + Spectre XPS Provides Greatest Performance

28nm Memory	Size		Runtime hrs (10 CPUs)		Speedup
	Xtrs	RCs	MX + XPS	MX + Other	
128KB SRAM	1.1M	22M	5:31:10	42:38:10	**7.7x**
256KB RegFile	1.5M	28M	6:30:15	40:01:10	**6.2x**
128KB CAM	3.1M	90M	12:50:20	N/A	**N/A**
1MB L2 Cache	7M	38M	3:48:10	115:40:50	**31.3x**

Module 8: The Flows of Liberate MX

Usage Models for Liberate MX

Depending on the type of memory design being characterized, there are multiple usage models for Liberate MX.

1. **Automated standard instance flow:** The design is of standard functionality [EX: single port SRAM, dual port SRAM, Reg files etc]
   - Pins: Clock, Address, Data in, Chip Enable, Write Enable, Bit write, Data out
   - Write is performed by enabling the memory, putting in write mode, and rising the clock
   - Read is performed by enabling the memory, putting in read mode and rising the clock
2. **Automated vendor re-characterization flow:** User is using memories from a third party vendor. This user might be adding an uncharacterized pvt or verifying the compiler accuracy.
3. **Full custom flow:** User is characterizing a custom design
   - The design has custom functionality
     - ❑write-bypass behavior of output pin
     - ❑Pre-charge glitch @ output pin
     - ❑others

Module 9: The Liberate MX Full Custom Flow

Module Objectives

- In this module, you
  - Will be introduced to the Liberate MX full custom flow
  - Will be able to setup and run an instance using the full custom flow

Setting up Liberate MX for the Custom Instance

- Liberate MX needs the following files
  - **Netlist** (usually extracted)
  - Spice **Models**
  - **mx.tcl**
    - top level options
    - specifies included file locations
    - contains characterization commands and variables
    - Table definitions
  - **template.tcl**
    - Definition of the instance to be characterized
    - Thresholds
    - Slews and loads
    - Arcs to be characterized
  - **Table files**
    - Simulation vectors (truth tables) for partitioning

Setting up Liberate MX: mx.tcl

- The following information is declared in the mx.tcl file:
  - Corner and Rail information
  - Simulators to be used (partitioning, leakage, partition)
  - Global fastspice settings
  - Probing control options
  - Macro Netlist
  - Models
  - Bitcell information
  - Debug options
  - Macro Characterization command
  - Library database and Liberty file export

Setting up Liberate MX: mx.tcl

- Defining the Operating conditions
  - ***set_vdd*** : used to define a supply node
  - ***set_gnd*** : used to define a ground node

```
set vdd_array 0.9
set vdd_logic 1.0

set gnd 0.0

set_vdd vdda $vdd_array
set_vdd vddp $vdd_logic
set_gnd vssx  $gnd
```

Setting up Liberate MX: mx.tcl

- Defining the Operating conditions
  - ***set_operating_condition*** : defines the primary supply voltage and temperature

```
set_operating_condition -voltage $vdd_logic -temp -30
```

- This defines the default voltage for the instance and for all input and output pins

Setting up Liberate MX: mx.tcl

- Overriding the voltage level for input and output pins:
- Needed for instances with multiple power supplies

set_pin_vdd

-supply_name <name>	Name of the supply that drives this pin.
{cell_names}	List of cells
<pin_name>	Name of pin
<vdd_value>	Power supply value

- This defines the voltage level for the specified pin and overrides the voltage defined in the set_operating_condition statement

Setting up Liberate MX: mx.tcl

- Setting Virtual Rails
  - Defining virtual rails allows MX to more efficiently static partition the instance
  - It is important to declare the virtual nodes, however, it is not important to properly define their voltage levels. This information will be derived from the simulations.
    - ***set_gnd –virtual*** `v_gnd $gnd`
    - ***set_vdd –virtual*** `v_vdd $vdd_logic`
      - i.e. ***`set_vdd -virtual -no_model -ignore_power`*** `vdda 1.05`
  - Bias nodes such as VREF shared in different blocks should be defined as “Virtual Rails”
  - If the virtual rail is used in a partition, MX will automatically handle this properly. However, for measurements that are on the virtual rail itself, partitioning should be disabled.

Setting up Liberate MX: mx.tcl

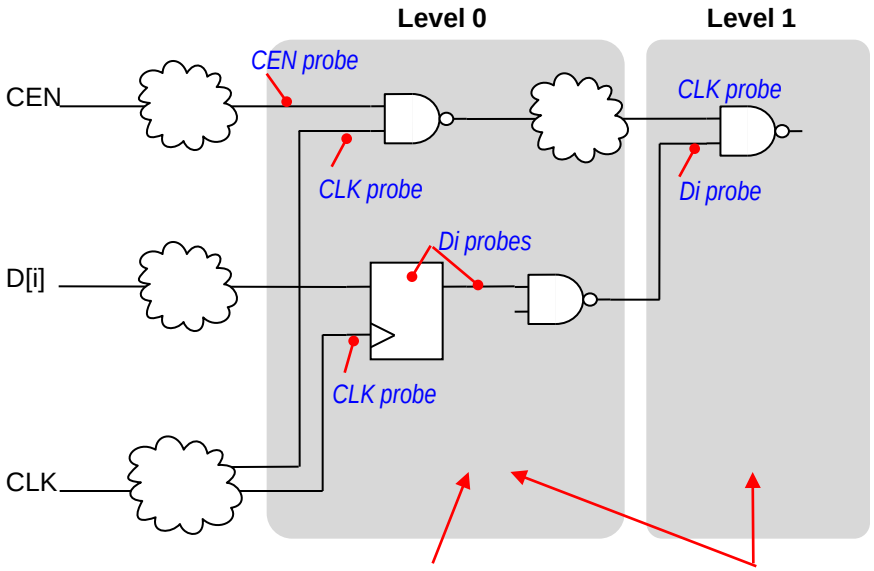
- In a fully extracted netlist, it can sometimes be difficult to identify the proper name of an internal switched rail. To have the tool list nodes that appear as virtual rail candidates, you can:
  - set variable ***mx_find_virtual_rails*** to 1 in your main script
  - run MX, analyze the reported nodes and add them to your main script
  - stop execution and re-run MX

```
### when you ask MX to find virtual powers and grounds:
set_var mx_find_virtual_rails 1

### ... switched power nodes candidates are reported on the output as in ...
(MX-info) - Partitioning - static - start ...
(MX-info) - Finding virtual rails ...         wall clock time +=    0 sec
(MX-info) - AA/ar_vdd:2 - 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
   AA/ar_vdd:2 $vdd ($gnd)' to your script
(MX-info) - Partitioned   21351/22271 xtrs    wall clock time +=    1 sec
(MX-info) - Merging loops ...
```

- **Note**: The nodes reported by this step as possible virtual are not guaranteed to be virtual. All reported nodes should be checked before adding to the –virtual list.

Setting up Liberate MX: mx.tcl

Auto Probing Level



Hold constraint probes placed here for
mx_autoprobing_**hold**_level = **0**

Setup constraint probes placed here for
mx_autoprobing_**setup**_level = **1**

Setting up Liberate MX: mx.tcl

- Settings for Automatic “Constraint” Probing
  - ***mx_autoprobing_hold_level***

    Controls the number of levels of intersection to traverse for locating of hold time probes. Default is 0. This means that only the first level of intersection of pin and related (e.g. data and clock) will lead to probes. The 0 level is usually the input latch

  - ***mx_autoprobing_setup_level***

    Controls the number of levels of intersection to traverse for locating of setup time probes. Default is 1. This means that the first and second level of intersection of pin and related will lead to probes. The 0 level is usually the input latch, while the 1 level is usually a later intersection with clock.

  - Can be set on the global level or inside the ***-attribute {altos_autoprobing_level <value>}*** statement to ***define_arc*** command.

Setting up Liberate MX: mx.tcl

- Appropriate Levels for setup and hold
  - A passing setup time will ensure that the new data can successfully pass through the first stage latch. Therefore, the setup level is typically set to 1
  - A passing hold time will ensure that the latch will not be corrupted by the new value of the signal. In a passing hold, the signal will usually not propagate beyond the first stage. Therefore, the hold level is typically set to 0
  - Different circuit structures can necessitate the changing of these values

Setting up Liberate MX: mx.tcl

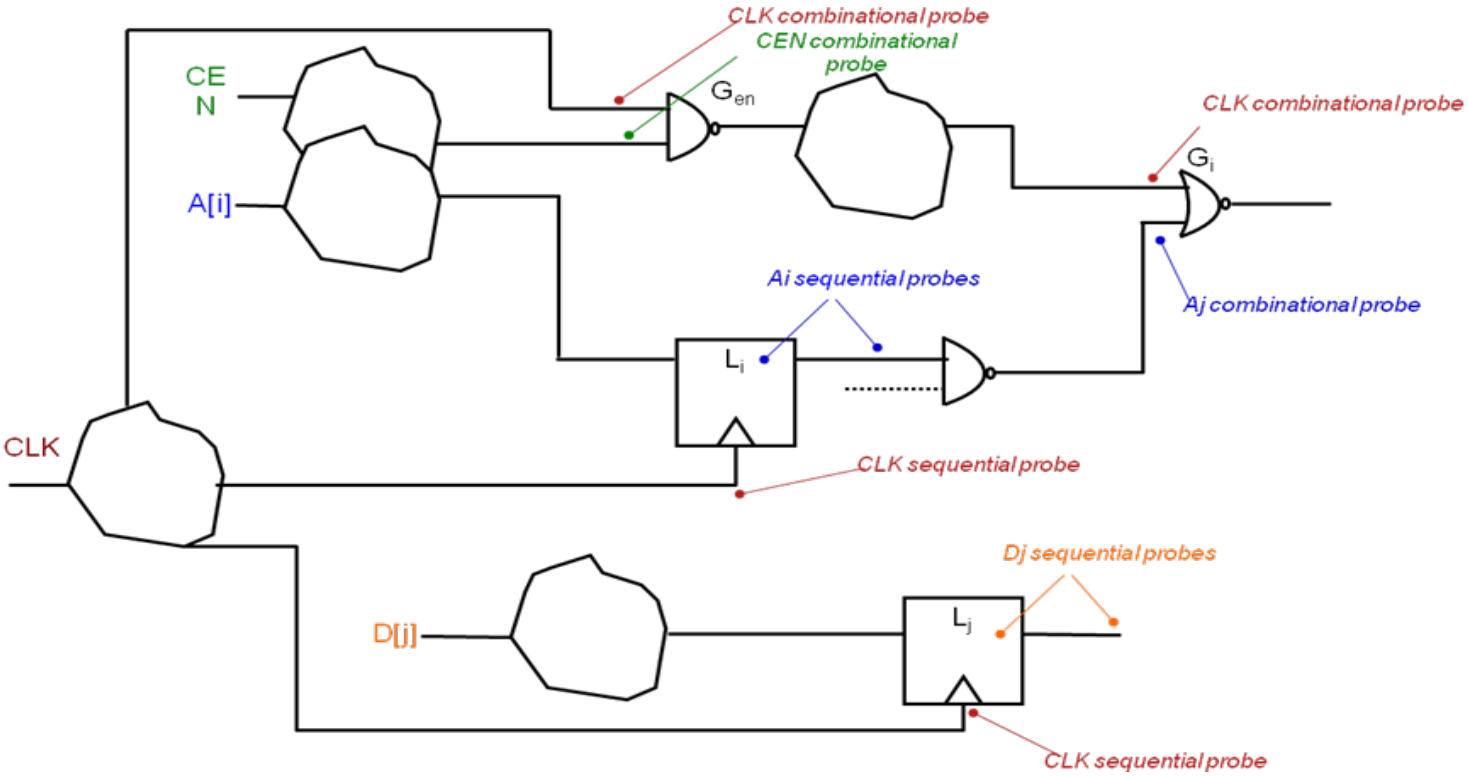
- Variables for Automatic Constraint Probing
  mx_setup_seq (default: all)
  mx_hold_seq (default: all)
  mx_setup_comb (default: all)
  mx_hold_comb (default: all)

- Control the probing locations to be considered for setup and hold.

- Allowed settings are “all”, “none” or a list of input pin names.

Setting up Liberate MX: mx.tcl

Sequential and Combinational Probing



Setting up Liberate MX: mx.tcl

Reading in the Netlist

- Use the ***read_spice*** command
- Pass a list of models and netlists to the command
- Netlist can be
  - pre- or post-layout
  - Fully extracted or only partially extracted
  - Hollowed-out or complete
- Only requirement is that a .subckt definition exists for the macro and that its pinout is consistent with the IO list described in ***define_cell*** command
- ".include" commands in the netlist may be lost – unless being replaced into model file
- Add command ***set_var mx_xps_inc_str ".inc"*** if the netlist is NOT in DSPF format
- Add command ***set_var spice_delimiter "string"***, where "***string***" could be "***/***" (default), "***.***", "***|***", etc

Setting up Liberate MX: mx.tcl

Reading in the Models

- It is only necessary to run ***read_spice*** on the model file if there are devices in the netlist that are not defined with ***define_leafcell***.
- Execution will be slower if models are read in when it is not needed
- Use the ***read_spice*** command
- Pass a list of models to the command
  - Can be the same command used for the netlist
- Models are normally saved in an include file (i.e. “model.inc” file)

```
### provide a model file
model.inc:
   .lib '<path>/usage.l' SSusage
   .lib '<path>/45.l' SS_diode
   .lib '<path>/45.l' SS_sram
   .inc ‘<path>/comp_extnet_rc0.include'
```

Setting up Liberate MX: mx.tcl

- Advanced Control for reading spice model and netlist
  - ***extsim_model_include*** → *".inc" call for spice model in **fast-spice** deck*
    - This line will tell liberate not to flatten the model, to just include it. Processing the model is not necessary if using an external simulator and can take a lot of time for a memory instance.
  - ***extsim_deck_include*** → *".inc" call for netlist in **full-spice** deck*
    - Will include the deck directly rather than writing it into the .sp file.
- Recommended Settings
  - ***extsim_model_include***, ***extsim_deck_include***: always use

The *define_leafcell* command

- This command defines the stopping point for netlist flattening.
- This command is required when using ***extsim_model_include*** and has no effect without it.
- Supports identification of various types such as MOSFET, diode, resistor and capacitor devices.
  - 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
```

Example *of model & netlist including*

```
set_var extsim_model_include /lan/csv/xpsrd_testout2/sram/include_tt_fix
set_var extsim_deck_include 1
define_leafcell -type nmos -pin_position {0 1 2 3} {nchpd_sr_mac nchpd_sr nchpg_sr_mac nchpg_sr nch_mac nch_lvt_mac nch_hvt_mac}
define_leafcell -type pmos -pin_position {0 1 2 3} {pchpu_sr_mac pchpu_sr pchpu_dpsr_mac pchpu_dpsr pch_mac pch_lvt_mac pch_hvt_mac}
define_leafcell -type diode -pins {0 1} {ndio}
read_spice {/lan/csv/xpsrd_testout2/sram/include_tt_fix /lan/csv/xpsrd_testout2/sram/rc/sram.typical.90c.ckt_addr6_0_fix}
```

Setting up Liberate MX: mx.tcl

Setup fast spice simulator path

Specify the external fast spice simulator of choice, and parameters

The following spice engines are supported today:

Cadence

- **Spectre XPS** *set_var fastsim_cmd “<path>/spectre”*
- UltraSim

Multiple fastspice simulators can be specified.

NOTE: when multiple engines are specified, each table file must contain a fastsim line to identify which simulator should be used with which table.

Setup full spice simulator path

Specify the external full spice simulator of choice, and parameters

Cadence

- **Spectre APS** *set_var extsim_cmd “<path>/spectre”*
  set_var extsim_cmd_option "+aps +spice -64"

Setting up Liberate MX: mx.tcl

Reuse of Fast Spice Data

set_var mx_fastsim_reuse 1

The setting of this variable controls whether the fast spice simulation results from previous run will be reused. Liberate MX will look for fastspice data in $TMPDIR/mx_reuse/mx_fastsim . The reuse of the fastspice results can greatly speed up debugging of a Liberate MX setup, however it should be used carefully.

DO NOT reuse fastsim results if following:

- Stimuli (tables) Changed
- Netlist has changed
- The fastspice options have changed
- The simulation interval has changed

Note : MX does some consistency checking but still it is recommended to user to remove data manually if any of the above mentioned changed

Setting up Liberate MX: mx.tcl

- **MX Simulation Interval**
  - The mx_simulation_interval is the duration of time for each line of the expanded table
  - The simulation interval determines the amount of time after an input event that the resulting internal events will be attributed to that event.
  - Command: ***set_var mx_simulation_interval*** ***<value>***
    - Set the value of this parameter to at least twice the value of the maximum delay that needs to be measured during the run. This is a global attribute and is applied to all tables and arcs
    - There are two other methods of specifying the simulation interval
      - Table-based, using ***simulation_interval <value>***.
      - Arc-based: using an ***-attribute {altos_mx_simulation_interval <value>}*** to ***define_arc***.
    - Multiple definitions have the precedence:
      1) Arc-based (highest)
      2) Table-based
      3) Global i.e. set_var (lowest)

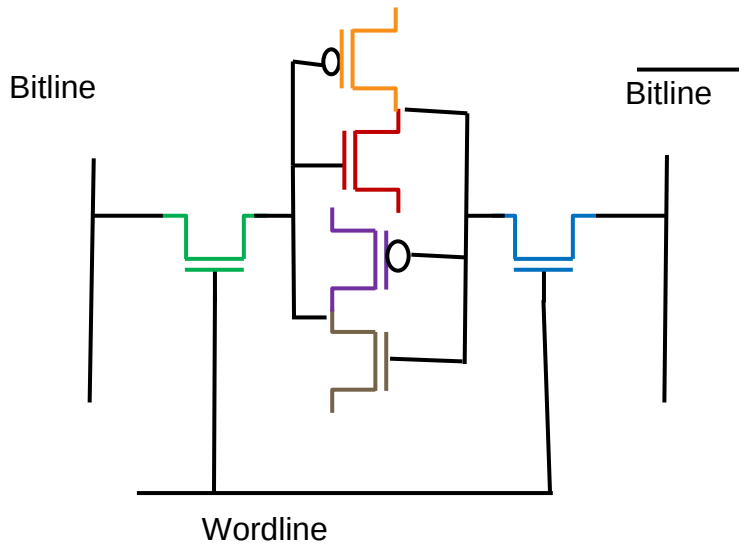
Setting up Liberate MX: mx.tcl

- Specify array core cells
  - Identification of memory array core cells, bit lines and word lines helps MX in speeding up both static and dynamic partitioning - but it is not a mandatory step
  - Memory core cell identification can be switched off with variable:
    set_var mx_find_memcores 0
  - By default MX looks for a 6T cell - with or without switched footer / header
  - For the more common cases, variable ***mx_corecell*** can also be used. Valid values are "***single_port***" (default, 6T), "***dual_port***" (8T) and rom
  - use option ***-mxcore*** to ***define_cell*** command to specify a description (pattern) of the core cell.
  - Ex: `-mxcore <file>`

Setting up Liberate MX: mx.tcl

Create a Custom Bit Cell File

```
mxcore  <name>{
        storage {
                sram {
                        device {
                                model:nmos
                                side:1
                                ctrl:core
                                use:storage
                        }
                        device {
                                model:pmos
                                side:1
                                ctrl:core
                                use:storage
                        }
                        device {
                                model:nmos
                                side:0
                                ctrl:core
                                use:storage
                        }
                        device {
                                model:pmos
                                side:0
                                ctrl:core
                                use:storage
                        }
                }
        }
        port {
                writeread {
                        device {
                                model:nmos
                                side:0
                                ctrl:word
                                use:pass
                        }
                        device {
                                model:nmos
                                side:1
                                ctrl:word
                                use:pass
                        }
                }
        }
```



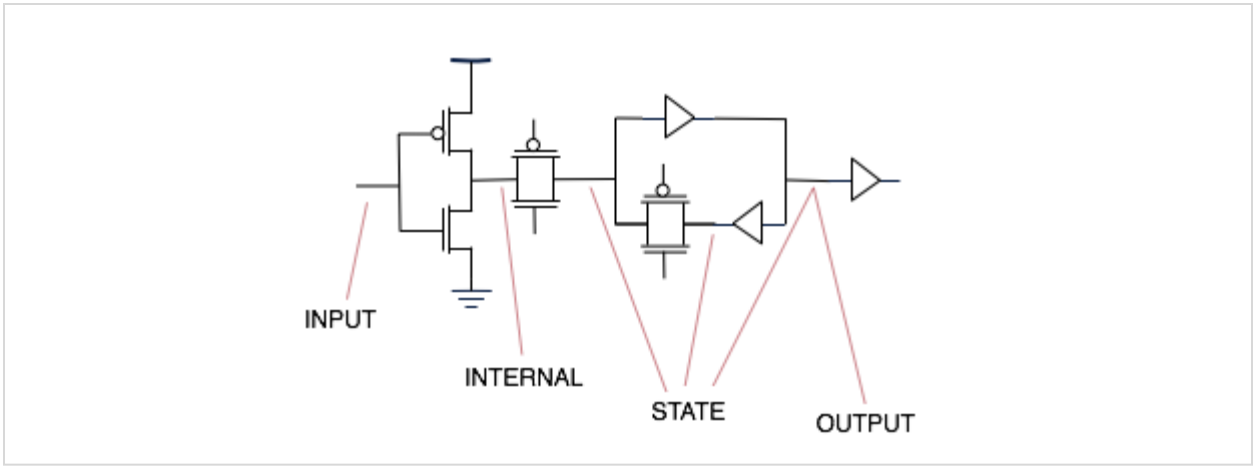
Setting up Liberate MX: mx.tcl

Setup automatic probing

Main commands to control how automatic probing is done are:

mx_seq_probing - Default value is "***input internal state output***"

valid values are ***input*** , ***internal***, ***state***, ***output***- or any combination of them. Identifies which node candidates should be considered in sequential logic per diagram below of typical latch input.



Setting up Liberate MX: mx.tcl

Constraints

- Default setting for calculating constraints is difference of delays
  - Delay from Data to Data probe is measured
  - Delay from Clock to Clock probe is measured
  - The difference of the delays is the setup or hold time
  - Faster run time
  - Usually more pessimistic
- Bisection search is also available
  - A search is performed where the setup or hold time given to the data signal is varied in order to determine the time which leads to a failure
  - Partitions are created from the fast spice simulation using difference of delays
  - The bisection search is performed on the partition

Setting up Liberate MX: mx.tcl

- During the partitioning and auto-probing phase, worst case constraint arcs are identified using path-delay differences method.
- The full spice characterization phase uses either bisection or method path_delay differences method depending on the value of ***mx_bisection*** parameter.

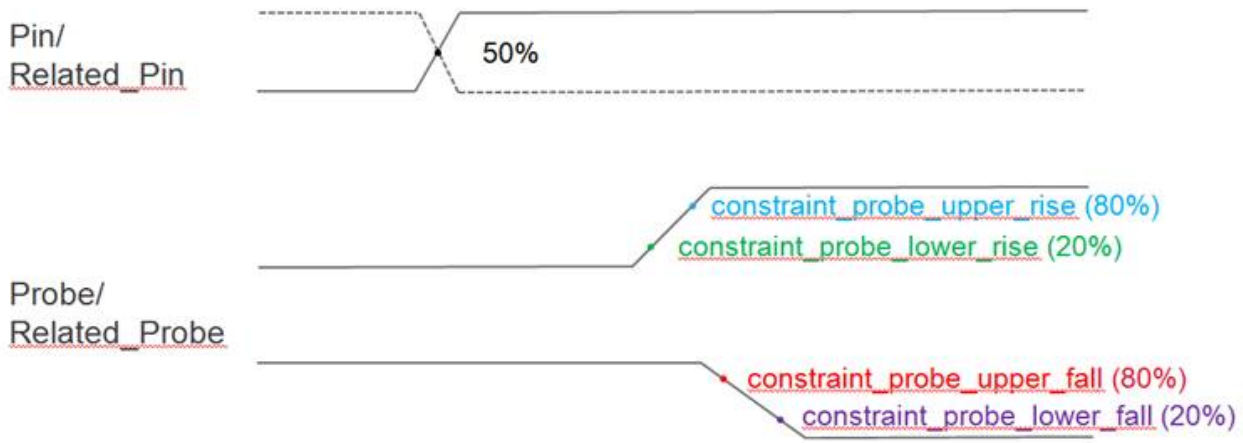
```
set_var mx_bisection  < 1 | 0 >
```

- **0**: Path-delay differences method. (default)
- **1**: Bisection method.
  - Automatic probes are usually pushed at the outputs of slave stages
    - command "***set_var mx_hold_swap_clock_data 0***" is required
- If automatic bisection is not used, the user can specify bisection probes explicitly by using ***-bis_probe {}*** option to ***dfine_arc -attribute {altos_mx bisection 1}*** command.

Setting up Liberate MX: mx.tcl

MX Variables for constraint probing thresholds setting

threshold point of a constraint measurement	for setup measurement max(Signal pin) -min(CLK pin)	for hold measurement max(CLK pin) - min(Signal pin)
constraint_probe_lower_rise	internal probe on CLK path (related_pin)	internal probe on Signal path (pin)
constraint_probe_upper_rise	internal probe on Signal path (pin)	internal probe on CLK path (related_pin)
constraint_probe_lower_fall	internal probe on Signal path (pin)	internal probe on CLK path (related_pin)
constraint_probe_upper_fall	internal probe on CLK path (related_pin)	internal probe on Signal path (pin)
50%	Signal pin	Signal pin
50%	CLK related_pin	CLK related_pin



cadence

Setting up Liberate MX: mx.tcl

Commands for margins / adjustments
before characterization, set variables:
mx_constraint_ocv_factor -> % factor by which to correct constraint measurement
mx_delay_ocv_factor -> % factor by which to correct delay measurement
mx_pathdelay_data_margin -> % margin to add to data path for setup calculation
mx_pathdelay_clock_margin -> % margin to add to clock path for hold calculation
before modeling, use command: add_margin
liberate_mx > add_margin -help
Usage information:

Var/FlagName	Type	Value	Help
-type	string		delay, trans, cap, constraint, leakage, power, hidden, setup, hold, recovery, removal or mpw; default apply margin to all types
-cells	list		cell list
-pin	list		list pin
-related	list		list related pins
-when	string	(*)	state dependent
-abs	float	(0)	Amount of margin to add, default 0.0 (no margin)
-rel	float	(0)	Relative amount of margin to add, e.g. use 0.05 for 5%, default 0.0 (0%)

ALAPI
complete TCL command API to access / manipulate / write any group in LDB

Setting up Liberate MX: mx.tcl

define_table {table_files} <cellname>

- Allows vector table files to be specified without using the ***-mxtable*** option to the ***define_cell*** command.
- Example:

  define_table {<path>/delay.tbl <path>/constraint.tbl} sram

Setting up Liberate MX: mx.tcl

char_macro is the command to run the characterization after all other commands and variables have been set

Examples:

Spectre XPS for dynamic partitioning with 5 threads and Spectre APS for characterization (partition) runs with 8 threads. Generate CCS timing and CCS noise models

char_macro –partsim spectre -partthread **5** ***–charsim spectre -charthread*** **8** ***-ccs -ccsn***

char_macro -extsim [list spectre] -thread **5** ***-char_params [list -extsim spectre -thread*** **8** ***-ccs -ccsn]***

Setting up Liberate MX: Template

read_library <reference.lib>

write_template –verbose template.tcl

- **This command sequence is executed at the *liberate_mx* prompt**
- **This command sequence will automatically define:**
  - templates to be used in the characterization process
    - ***define_template*** commands
  - indexes to be used for each pin / bus
    - ***define_index*** commands
  - thresholds for slews
  - cell, with IOs, template to use -> ***define_cell*** command
  - arcs and WHEN conditions -> ***define_arc*** commands

Setting up Liberate MX: Template

- Defining Template (slew and load ranges)

```
define_template -type delay -index_1 {0.007 0.0875 0.2625 }  -index_2 {0.0 0.108 0.216
0.432 0.864}  delay_template_3x5

define_template -type constraint -index_1 {0.007 0.0875 0.525 } -index_2 {0.007 0.0875
0.2625 } constraint_template_3x3

define_template -type power -index_1 {0.007 0.0875 0.2625 }  -index_2 {0.0 0.108 0.216
0.432 0.864} power_template_3x5
```

- Defining Macro Cell

```
define_cell \
       -clock { clk } \
       -input { we add[6:0] din[31:0] psm_vdd[1:0] tm[4:0] cs ovstb pt } \
       -output { dout[31:0] } \
       -delay delay_template_3x5 \
       -power power_template_3x5 \
       -constraint constraint_template_3x3  \
       SRAM
```

- **Supported Bus notations**
- **D[MSB:LSB] → D[15]…D[0]**
- **A<MSB:LSB> → A<15>…A<0>**
- **CS(MSB:LSB) → CS(15)…CS(0)**
- **D_MSB:LSB → D_15 …. D_0**
- **A|MSB:LSB| → A15 … A0**

Setting up Liberate MX: Template

- Optional definitions

#Measurements 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

slew threshold of index values, lookup table values

slew_derate_of_library = (70 - 30)/(90 - 10) = 40/80 = 0.5

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

threshold of output slew measurement

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

threshold of delay measurement & lookup table

... ...
... ...

NOTE - Thresholds can be defined on an arc base as well for delay/retain arcs - via the following options to ***define_arc*** command:
-slew_threshold,
-pin_probe_threshold,
-related_probe_threshold,
-delay_threshold,

Setting up Liberate MX: Template

mx_create_if_uda

< 0 | 1 > Tells MX to only characterize User Defined Arcs and do not add arcs based on circuit activity. Default: 0

1: MX will not generate extra-arcs from simulation results if those arcs are not defined in template.tcl.

0: MX will generate extra arcs even if they are not defined.

Setting up Liberate MX: Template

User arc definitions

User defined arcs can be specified for the following:

Specify WHEN conditions so to separate timing in to different modes:

```
define_arc -pin dout[0] -related_pin clk -when "!we & cs" -pin_dir R -related_pin_dir R
define_arc -pin dout[0] -related_pin clk -when "!we & cs" -pin_dir F -related_pin_dir R
define_arc -pin dout[0] -related_pin clk -when "we & cs" -pin_dir R -related_pin_dir R
define_arc -pin dout[0] -related_pin clk -when "we & cs" -pin_dir F -related_pin_dir R
```

```
foreach type { setup hold } {
  foreach dir { R F } {
       define_arc \
             -type $type \
             -pin add\[9:0\] \
             -related_pin clk \
             -when "we & cs" \
             -pin_dir $dir \
             -related_pin_dir R \
             $inst
  }
}
```

Setting up Liberate MX: Template

Specify equivalence classes among bus bits so to instruct the tool on whether/how to perform min/max bundling inside the bus. For example:

For bus add setup rising, find worst case across all bits of bus add and use that for partitioning / characterization

```
define_arc -type setup -pin add[6:0] -related_pin clk -pin_dir R -related_pin_dir R $cell
```

or equivalently:

```
define_arc -type setup -pin add[0] -related_pin clk -pin_dir R -related_pin_dir R $cell

define_arc -type setup -pin add[1] -related_pin clk -pin_dir R -related_pin_dir R $cell
… …
define_arc -type setup -pin add[6] -related_pin clk -pin_dir R -related_pin_dir R $cell
```

For bus add setup rising, find worst case across all column decoder bits, but keep separate from row decoder bits for partitioning / characterization. Keep MSB also separate

```
define_arc -type setup -pin add[1:0] -related_pin clk -pin_dir R -related_pin_dir R $cell

define_arc -type setup -pin add[5:2] -related_pin clk -pin_dir R -related_pin_dir R $cell

define_arc -type setup -pin add[6:6] -related_pin clk -pin_dir R -related_pin_dir R $cell
```

For bus add setup rising, partition and characterize all bits separately

```
define_arc -type setup -pin add[0:0] -related_pin clk -pin_dir R -related_pin_dir R $cell

define_arc -type setup -pin add[1:1] -related_pin clk -pin_dir R -related_pin_dir R $cell
… …
define_arc -type setup -pin add[6:6] -related_pin clk -pin_dir R -related_pin_dir R $cell
```

Increasing the number of arcs defined will result in a runtime increase.

Setting up Liberate MX: Template

Defining probes in ***define_arc***

-probe {*names****}*** List of names of nodes to monitor for setup/hold characterization

-probe_dir <R | F> Transition direction of probe pin(s)

-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)

The use of ***-probe*** and ***-related_probe*** adds probes to the list of considered probes. The worst case of the MX determined probes and the ***define_arc*** declared probes will be used in the partition simulations.

In order to force the probes specified by ***-probe*** and ***-related_probe*** to be the only probes considered, the ***-force*** option must be added to the ***define_arc*** command.

Setting up Liberate MX: Template

Removing probes from consideration in **define_arc**

-attribute {autoprobing_skip_rel_probe "name0 … name N"}

-attribute {autoprobing_skip_rel_probe_regexp "regexp0 … regexpN"}

-attribute {autoprobing_skip_probe "name0 … nameN"}

-attribute {autoprobing_skip_probe_regexp "regexp0 … regexpN"}

These attributes should be defined to prevent Liberate MX from using an undesired probe point. This is usually a work around to correct for some undesired behavior.

Setting up Liberate MX: Template

Other define_arc Examples

```
#arctype delay & retain

define_arc \
    -related_pin_dir R -pin_dir R \
    -related_pin {ph1} \
    -pin {dout[71:0]} \
    sram

define_arc \
    -related_pin_dir R -pin_dir F \
    -related_pin {ph1} \
    -pin {dout[71:0]} \
    sram

define_arc \
    -type retain \
    -related_pin_dir R -pin_dir R \
    -related_pin {ph1} \
    -pin {dout[71:0]} \
    sram

define_arc \
    -type retain \
    -related_pin_dir R -pin_dir F \
    -related_pin {ph1} \
    -pin {dout[71:0]} \
    sram
```

```
#arctype power

define_arc \
    -when "wren & en & banken & !dead" \
    -type power \
    -pin_dir R \
    -pin {ph1} \
    sram

define_arc \
    -when "wren & en & banken & !dead" \
    -type power \
    -pin_dir F \
    -pin {ph1} \
    sram

define_arc \
    -type power \
    -pin_dir R \
    -pin {dout[71:0]} \
    sram

define_arc \
    -type power \
    -pin_dir F \
    -pin {dout[71:0]} \
    sram
```

```
#arctype leakage

define_leakage \
    -when "!en & !banken & !dead" \
    sram

define_leakage \
    -when "!en & !banken & dead" \
    sram

define_leakage \
    -when "en & banken & !dead" \
    sram

define_leakage \
    -when "en & banken & dead" \
    sram
```

Setting up Liberate MX: Template

Creating Equations of automatically probed arcs with ***define_arc***

- Use the ***-name*** option to ***define_arc*** to assign a name to an arc
- Use the ***-equation*** option to ***define_arc*** to use the data from the named arc to calculate another arc

Example:

```
define_arc \
    -type mpw \
    -name CLK_MPW_H \
    -pin_dir R  \
    -related_pin {CLK} \
    -pin {CLK} \
    Instance_name
```

```
define_arc \
    -type min_period \
    -equation “2*CLK_MPW_H” \
    -related_pin_dir R  \
    -pin_dir R  \
    -related_pin {CLK} \
    -pin {CLK} \
    Instance_name
```

Setting up Liberate MX: Template

define_measure

User defined measurements

- User can specify any measurement which falls outside what is automatically covered by the tool through the ***define_measure*** command.
- The command has been developed to provide and API between MX and any 3rd party compiler environment and allows the user to specify a measurement between any (input, output, or internal) nodes and apply the same dynamic partitioning algorithm used for the other metrics like delay and constraint.
  - The ***define_measure*** command is relatively flexible can be used to implement a wide variety of spice measurements. However, the measurement types supported are not exhaustive.

Setting up Liberate MX: define_measure

Example

```
define_measure \
    -name      t_cen_probe \
    -trig      {CEN} \
    -trig_dir  fall \
    -trig_val  [expr $vdd * 0.5] \
    -targ      {CEN_PROBE:0} \
    -targ_type "delay" \
    -targ_dir  rise \
    -targ_val  [expr $vdd * 0.9] \
    instance_name

define_measure \
    -name      cen_f_setup \
    -keep      max \
    -duration  0.5 \
    -trig      {CEN} \
    -trig_dir  fall \
    -trig_val  [expr $vdd * 0.5] \
    -targ_type "delay" \
    -equations {"t_cen_probe -
   t_clk_probe"} \
    instance_name
```

```
define_measure \
    -name      t_clk_probe \
    -trig      {CLK} \
    -trig_dir  rise \
    -trig_val  [expr $vdd * 0.5] \
    -targ {CLK_PROBE_0} \
    -targ_type "delay" \
    -targ_dir  rise \
    -targ_val  [expr $vdd * 0.2] \
    instance_name

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

Setting up Liberate MX: define_measure

Options to *define_measure*

-keep string (none)	Keep the measure value in LDB, options are [none|min|max]
-name string ()	Measurement name (required)
-duration double (1)	Measurement duration in number of cycles, starting at this trigger (for the equation only)
-trig list ()	Trigger signal name (required)
-trig_dir string ()	Trigger signal direction [rise|fall] (required)
-trig_val float (-100)	Trigger voltage (required)
-trig_d string ()	Delay trigger after named measure or number of cycles
-trig_e string (first)	Trigger edge specifier within the duration window
-trig_s string ()	Shift trigger crossing time by result of named measure or number specified (in sec.)
-trig_start list ()	Start trigger signal name (when specifying a window for a voltage measurement)
-trig_start_dir string ()	Start trigger signal direction [rise|fall]
-trig_start_val float (-100)	Start trigger voltage
-trig_start_d string ()	Delay start trigger after named measure or number of cycles
-trig_start_e string (first)	Start trigger edge specifier within the duration window
-trig_start_s string ()	Shift start trigger crossing time by result of named measure or number specified (in sec.)
-trig_end list()	End trigger signal name (when specifying a window for a voltage measurement)
-trig_end_dir string ()	End trigger signal direction [rise|fall]
-trig_end_val float (-100)	End trigger voltage
-trig_end_d string ()	Delay End trigger after named measure or number of cycles
-trig_end_e string (last)	End trigger edge specifier within the duration window
-trig_end_s string ()	Shift End trigger crossing time by result of named measure or number specified (in sec.)

Setting up Liberate MX: define_measure

Options to *define_measure*

-targ list () One or two target signal name(s)

-targ_type string (delay) Target measurement type [delay|voltage]

-targ_dir string () Target signal (or difference) direction

-targ_val float (-100) Target voltage

-targ_d string () Delay target after named measure or number of cycles

-targ_e string (last) Target edge specifier within the duration window

-targ_s string () Shift targ crossing time by result of named measure or number specified (in sec.)

-failed_val string () Use this value for failed measurements

-max_val string () Use this value to set a max value for the measurement. Measurement will be set to the failed value if the result is > than the value specified for max. By default, no max is considered

-min_val string () Use this value to set a min value for the measurement. Measurement will be set to the failed value if the result is < than the value specified for min. By default, no min is considered

-equations list () Set of related equations consisting of measure names in Spice syntax

-when string () When to evaluate the measure

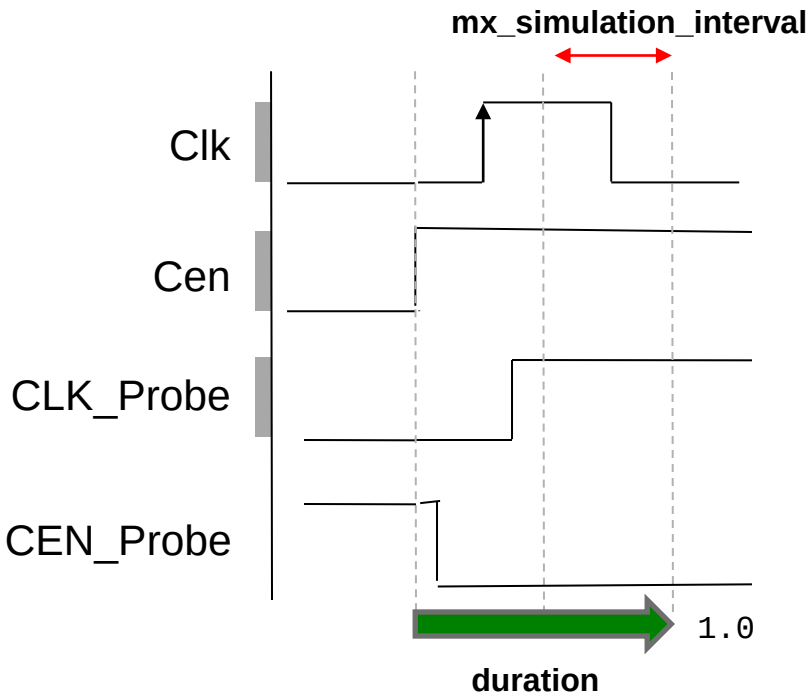
cellNames list () List of cell names

Setting up Liberate MX: define_measure

- The ***-duration*** flag to ***define_measure*** defines the interval of time for the equation to be evaluated
- If ***-duration*** is ***1***, That corresponds to a measurement window of ***2*mx_simulation_interval***
- For ***measurements***, the ***trig*** and the ***targ*** need to both occur within the ***duration*** of the trig event
- For ***equations***, the ***trig*** event starts the ***interval*** that all elements of the ***equation*** must evaluate within. The ***trig*** and ***targ*** events of each of the ***measurements*** in the ***equation*** must evaluate during the ***equation duration*** or they will be considered failed.

Define Measure: Duration

```
table  cycle1
pins   Clk   Cen   Q
vec0   L     L     X
vec2   R     1     M
endtable
```




```
define_measure \
    -name      t_clk_probe \
    -trig      {CLK} \
    -trig_dir  rise \
    -trig_val  [expr $vdd * 0.5] \
    -targ {CLK_PROBE_0} \
    -targ_type "delay" \
    -targ_dir  rise \
    -targ_val  [expr $vdd * 0.2] \
    instance_name

define_measure \
    -name      t_cen_probe \
    -trig      {CEN} \
    -trig_dir  rise \
    -trig_val  [expr $vdd * 0.5] \
    -targ      {CEN_PROBE:0} \
    -targ_type "delay" \
    -targ_dir  fall\
    -targ_val  [expr $vdd * 0.2] \
    instance_name

define_measure \
    -name      cen_f_setup \
    -keep      max \
    -duration 1.0 \
    -trig      {CEN} \
    -trig_dir  rise \
    -trig_val  [expr $vdd * 0.5] \
    -targ_type "delay" \
    -equations {"t_cen_probe - t_clk_probe"} \
    instance_name
```

Setting up Liberate MX: tables

- The table file contains the stimulus information for the fast spice simulation
  - Specifies which arc types are to be measured
  - Specifies the fastspice simulator and options
  - Specifies any include files for the fastspice simulation
  - Contains the input vectors and measurement times
- Note: It is common practice to separate the tables into individual files for different measurement types such as ***timing (delay retain)***, ***timing (setup)***, ***timing (hold)***, ***power***, ***leakage***, ***measure*** *etc*

Setting up Liberate MX: tables

- **Syntax for Stimulus/Tables**

arctypes <arc_type>

<fastsim_option> ***# optionally specify a fastsim simulator for this truth table***
<comment_string>
table *<table_id>*
pins ***<bus_id> <bus_id> ... <bus_id>***
<cycle_id> <value> <value> ... <value>
<cycle_id> <value> <value> ... <value>
<cycle_id> <value> <value> ... <value>

:::
endtable

where:
<arc_type> : <arc_type_string>
Vectors specified in table file will be used for <arc_type_string> > characterization
<arc_type_string> : one or more of *delay, retain, setup, hold, minperiod, mpw, power, leakage, measure, timing*
timing = delay retain setup hold
If <arc_type> not specified, default is = timing measure

table, pins, endtable are mandatory keywords
<table_id> : <string>
<cycle_id> : <string>
<bus_id> : <string>
<value> : *R F ? 0 1 B C D - X H L A P N onehot onecold*

Setting up Liberate MX: tables

<fastsim_option> : *fastsim, fastsim_reuse, fastsim_deck_include, fastsim_deck, fastsim_auto_ic*

fastsim <fastsim_identifier>
<fastsim_identifier> : *xps ultrasim*
Specifies what fastspice engine will be used to simulate the table file. Overwrites using -*extsim* option of *char_macro* command

fastsim_reuse
When present, instructs MX to look for previous fastsim results on this table file. fastsim results are stored for each run inside directory ./mx_fastsim - with name **<cellname>.<tablename>.alwf** This setting will override the setting in mx.tcl.

Setting up Liberate MX: Tables

fastsim_deck_include <netlist_path_name>
<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 in to database through the read_spice command. This allows for a table specific netlist to be run.

fastsim_deck <fastsim_deck_string>
<fastsim_deck_string> : a valid (for the engine specified in fastsim) text that will be included (as is) to the deck to be simulated in fast spice

fastsim_auto_ic <value>
<value> : one of 1, true, on, 0, false, off
Instructs MX to add (not add) .ic statements to bit olines and core nodes - as recognized in the static topology recognition step. It overwrites global variable mx_fastsim_auto_ic for the table file it is specified in.

If not specified, default value is given by value of global mx_fastsim_auto_ic (default is true)

If the user would like to provide their own initialization, it is recommended to disable the automatic initialization.

Example Table File

Bus size independent - reuse for all sizes

Measure CEB constraints

table	128X16M4F										
pins	CLK	CEB	WEB	BWEB	A	D	RTSEL	WTSEL	PD	BIST	Q
select	R	0	?	?	?	?	0x1	0x1	L	L	X
deselect	R	1	?	?	?	?	0x1	0x1	L	L	C
select	R	0	?	?	?	?	0x1	0x1	L	L	C
write0	R	L	L	L	L	L	0x1	0x1	L	L	X
write1	R	L	L	L	1	1	0x1	0x1	L	L	C
write0	R	L	L	L	0	0	0x1	0x1	L	L	C
read0	R	L	1	1	L	L	0x1	0x1	L	L	C
read1	R	L	H	H	H	L	0x1	0x1	L	L	D
read0	R	L	H	H	L	L	0x1	0x1	L	L	D
endtable											
table	128X16M4F_M										
pins	CLK	CEBM	WEBM	BWEBM	AM	DM	RTSEL	WTSEL	PD	BIST	Q
select	R	0	?	?	?	?	0x1	0x1	L	H	X
deselect	R	1	?	?	?	?	0x1	0x1	L	H	C
select	R	0	?	?	?	?	0x1	0x1	L	H	C
write0	R	L	L	L	L	L	0x1	0x1	L	H	X
write1	R	L	L	L	1	1	0x1	0x1	L	H	C
write0	R	L	L	L	0	0	0x1	0x1	L	H	C
read0	R	L	1	1	L	L	0x1	0x1	L	H	C
read1	R	L	H	H	H	L	0x1	0x1	L	H	D
read0	R	L	H	H	L	L	0x1	0x1	L	H	D
endtable											

Input Mnemonic	Meaning
R	0->1
F	1->0
H	11..11
L	00..00
1	Prev->11…1
0	Prev->00…0
?	??..?

Output Mnemonic	Meaning
C	Measure Constraint
D	Measure Delay
J	Measure Power
X	DONOT Measure

Measure CLK->Q delay retain

cadence®

Setting up Liberate MX: table

The Arc Evaluation Specifiers

- Specified under the **output** pin name in the table
- If specified as ***X*** then no arcs will be evaluated on this line (overrides all other specifications)
- If specified as ***D*** then delay and retain can be measured to the pin or bus that the D is under
- If specified as ***C*** then constraints will be evaluated on the line regardless of the pin it is under
- If specified as ***J*** then dynamic power will be evaluated on the line.
  - Unless the line is defined as "***output_power***" the pin that ***J*** falls under will not matter and the power will be attributed to the switching inputs.
  - In the case of "***output_power***", the measurement will be of the switching power of the pin that ***J*** falls under
- If specified as ***W*** then leakage will be measured
- If specified as ***M*** then ***min period***, ***mpw*** or ***define_measure*** can be evaluated regardless of the pin it falls under

Setting up Liberate MX: table

The expansion of the lines of the table

- Clock pins as defined in the ***define_cell*** command are handled differently than the other input pins.

- If a Clock pin contains an R or an F on the line of the table, the vector line is expanded into 2 lines.

  R => 0 1

  F => 1 0

- If there is a measurement on the line, the measurement interval will be applied to the second expanded line
- For the other signals on the line, the following conversions will be made

  L => 0 0

  H => 1 1

  0 => 1 0

  1 => 0 1

Setting up Liberate MX: table

The expansion of the lines of the table

Examples

```
table example
pins    CLK    ADR    Q
vec1     L      L     X
vec2     R      1     C
vec3     R      0     C
endtable
```

Will expand to:

```
table example
pins    CLK    ADR    Q
vec1     0      0     X
vec2     0      0     X
vec2     1      1     C
vec3     0      1     X
vec3     1      0     C
endtable
```

Because there are transitions of ADR and CLK on the same line as the C, we can measure setup and hold of ADR against CLK.

Setting up Liberate MX: table

The expansion of the lines of the table

Examples

```
table example
pins    CLK    ADR    Q
vec1     L      L     X
vec2     R      H     C
vec3     R      L     C
endtable
```

Will expand to:

```
table example
pins    CLK    ADR    Q
vec1     0      0     X
vec2     0      1     X
vec2     1      1     C
vec3     0      0     X
vec3     1      0     C
endtable
```

Because there are no transitions of ADR on the same line as the C, we can not measure setup and hold of ADR against CLK in this table.

Setting up Liberate MX: table

Multiple clocks

- When multiple clocks are present in a table, they should not both be set to R on the same line
- Use the expanded format instead when transitioning multiple clocks

Examples

```
table example
pins    CLKA    CLKB   ADR    Q
vec1     0       0      0     X
vec2     0       0      1     X
vec2     1       1      1     C
vec3     0       0      0     X
vec3     1       1      0     C
Endtable
```

Setting up Liberate MX: Table

- **Syntax for Stimulus/Tables**

Inputs: R F ? 0 1 B H L A P N Z onehot onecold
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
? → dont care; expanded to ??...?? and tied to 00...00 independently from any expansion it's involved in
1 → one; expands in to 00...00->11...11 when on the same line with an edge
0 → zero; expands in to 11...11->00...0 when on the same line with an edge
B → binary; expands in to 00...00->11...11->11...11->00...00 when on the same line with an edge.
A → alternate; expands in to 010...->101...->101...->010... when on the same line with an edge.
L →low; tied to 00...00 independently from any expansion it's involved in
H → high; tied to 11...11 independently from any expansion it's involved in
P N → positive (negative) pulse; expands the table to generate a 0->1->0 (1->0->1) sequence while keeping the other values on the same line untouched
onehot onecold → expands the table to generate a 00...01->00...10->...->01...00->10...00 (11...10->11...01->...->10...11->01...11) sequence while keeping the other values on the same line untouched

Outputs: X C D B J W SETUP HOLD
X → do not measure constraints, do not measure delay
C→ measure constraints only
D → measure delay only
J → measure dynamic power only
W → measure leakage only
B → measure all types as defined in the arctypes line
SETUP → measure only setup and not hold
HOLD → measure only hold and not setup

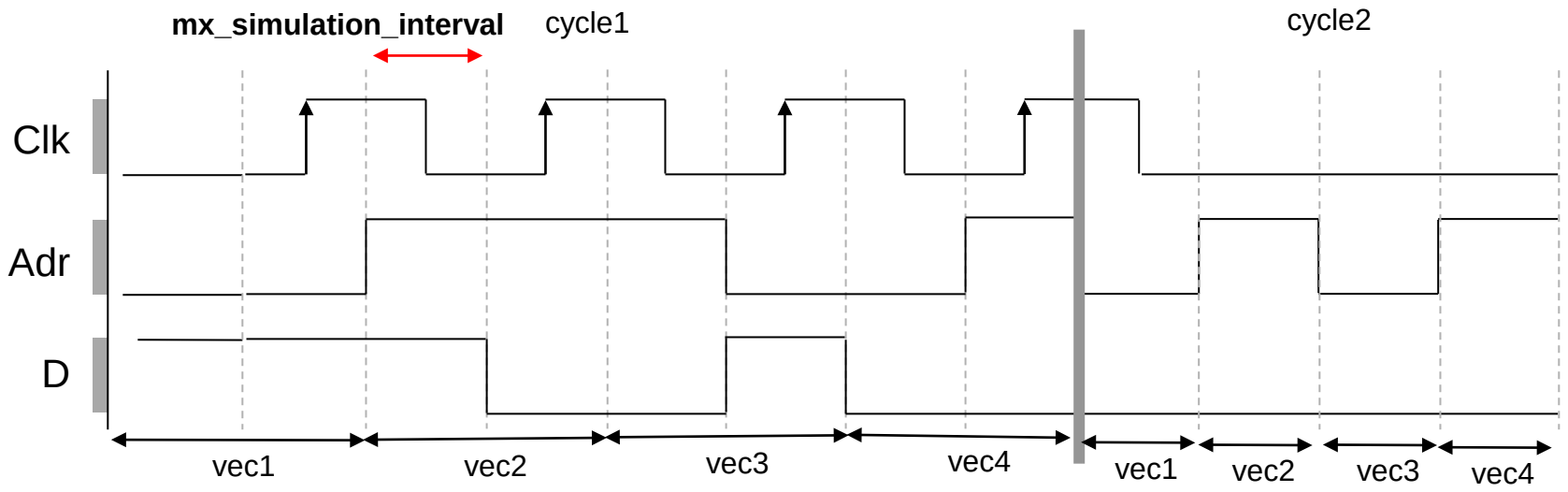
Setting up Liberate MX: Table

- The vectors are generated from the table file with an offset given to pins defined as clock pins.
  - *Transitions on clock pins occur at a time of ½ of the vector interval later than the other pins*

Example:

```
table  cycle1
pins   Clk   Adr   D
vec1   R     L     H
vec2   R     H     0
vec3   R     0     1
vec4   R     1     L
endtable
```

```
table  cycle2
pins   Adr
vec1   L
vec2   H
vec3   0
vec4   1
endtable
```



cadence

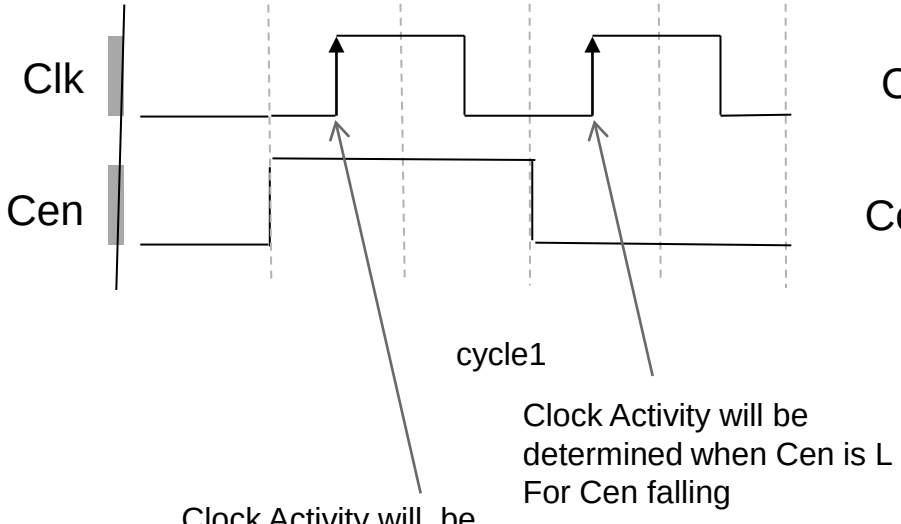
Vectors for Setup and Hold Time

- For all constraint tables, the full instance simulation will be run twice, once for setup and once for hold.
- In the setup run, the signal will switch before the related clock pin. The name of this table will be the name of the original constraint table.
- In the hold run, the signal will switch after the related clock pin. The name of this table will be ***zzz_<original_name>_auto_hold.tbl***
- This is necessary because of the side conditions needed to determine the clock activity.
  - For setup, the clock propagation comes on the later state of the signal (high for rising, low for falling).
  - For hold, the clock propagation comes on the earlier state of the signal (low for rising, high for falling).

Vectors for Setup and Hold Time (13.1 isr1)

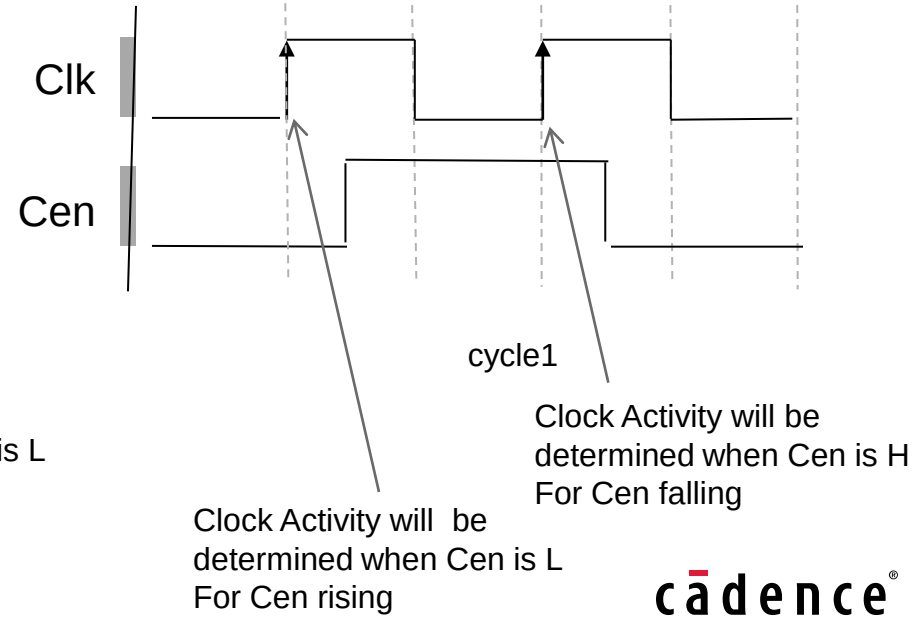
- Setup Time Vector

```
table  cycle1
pins   Clk   Cen   Q
vec1   R     1     C
vec2   R     0     C
endtable
```



- Hold Time Vector

```
table  cycle1
pins   Clk   Cen   Q
vec1   R     1     C
vec2   R     0     C
endtable
```



Setting up Liberate MX: Table

Power and Leakage Characterization

- Power and leakage characterization are performed using a fast spice simulator, run at the block level. A separate table is usually provided for internal power and one for leakage.

- Table-specific netlist, simulation interval, and initialization can be specified for power and leakage tables.

Setting up Liberate MX: Table

Example: calculate leakage in active and inactive mode. At least 100 ns after switching clk before actually taking the measurement. Use proper fastspice options for leakage to ensure accurate data.

```
arctypes leakage
fastsim xps
fastsim_cmd_option -64 +spice +xps +cktpreset=sram_pwr

table leakage
pins  clk cs q
set   R   L  X
meas  H   L  W
set   R   H  X
meas  H   H  W
endtable
```

Setting up Liberate MX: Table

FSDB to Table Flow

If the customer provides an FSDB file containing waveforms from a spice run, it is possible to translate those waveforms into a table file

Usage:

define_cell must be specified

```
fsdb2tbl <file_name>.fsdb –refs <reference signals> <file_name>.tbl
```

Default is to use the defined clock pins as reference for grid. This can be overwritten by defining the reference signals

Setting up Liberate MX: Table

- For dynamic power, make sure that the clock has measurements on both the rising and falling edges

```
table power
pins      clk cs q
Clk_rise   R  H  J
Clk_fall   F  H  J
Endtable
```

Or

```
table power
pins      clk cs q
Clk_low    0  H  X
Clk_high   1  H  J
Clk_low    0  H  J
Endtable
```

Taking the Maximum or Minimum of Multiple Measurements

- Sometimes it is necessary to take the maximum or minimum of multiple ***define_arc***'s and use that value for the final lib value of an arc
- Could be multiple –measure's or a mix of –measure and automatic probing

```
-attribute [list altos_mx_bundle clkrd_minperiod::max]
```

- In this example, the "max" specifies to take the maximum of all bundled arcs.
- The "clkrd_minperiod" ties all of the needed arcs together for determining the final value

Debugging of define_measure

- Allows the user to debug the measurement values for ***define_meaure*** statements
- Results from the full instance run are stored in ***measure.rpt.fastsim***
  - Contains all measurements, even if they are not chosen for partitions
- Results from the final partitions are in ***measure.rpt***
  - Contains only measurements that are chosen for the final partitions
  - Contains all slew or load values for the measurement

Measure.rpt and Measure.rpt.fastsim

measure.rpt.fastsim

margin name	eq(0)
mpw_clk_in_h_0	-1.687586333e-10
mpw_clk_in_h_1	1.750013911e-10
mpw_clk_in_h_2	5.664509770e-10
mpw_clk_in_h_3	3.976923436e-10
mpw_clk_in_l_4	-1.544223013e-10
mpw_clk_in_l_5	-5.258155565e-10
mpw_clk_in_l_6	2.005585762e-10
mpw_clk_in_l_7	-3.252692507e-10
altos_minp_latch_spacing_2	4.510260142e-10
altos_minp_latch_2	3.347472557e-10

measure.rpt

margin name	eq(0)
cycle_time	6.037769729e-10
	6.036560141e-10
	6.612139725e-10
mpw_clk_in_h_2	5.436660011e-10
	6.090900007e-10
	7.698739979e-10
mpw_clk_in_l_6	1.947259992e-10
	2.500029905e-10
	3.590369946e-10

Measuring Switching Power

- Liberate MX uses the full instance to simulate for power
- Switching power is measured by monitoring i(vdd), i(gnd) for each specified primary rail.
- Contribution is distributed equally across switching inputs in a particular simulation interval, and as detailed by command
  - Note: for accuracy of final power numbers, it is not recommended to switch multiple inputs at the same time because the power contribution of each event will not be able to be isolated
- Command: ***mx_power_assign*** *<default:clock>*
  - all | clock | input | output <list of pins>
  - Specifies how to distribute internal power among switching pins.
  - Selecting **all** is equivalent to input and output.
  - The default setting of **clock**, MX assigns all switching power to clock, independently from other inputs switching.
  - Selecting **input** distributes internal power contribution equally among switching input and clock pins.
  - Selecting **output** calculates the output switching power and assigns it to the output.

Measuring Switching Power

- Output power is calculated as the power consumed on the output circuitry (i.e. from SA output all the way to primary outputs) for a read operation.
- We can derive this component as the difference between the read power with output stable and the read power with output switching
- To model output power properly, user specifies a set of vectors in the table where output power should be measured, using cycle identifier keyword ***output_power***

Example table for output power

```
table power_Q
pins          CLK   A   WENB   Q
init          R     L   H      X
output_power  R     H   H      J
output_power  R     H   H      J
output_power  R     L   H      J
output_power  R     L   H      J
endtable
```

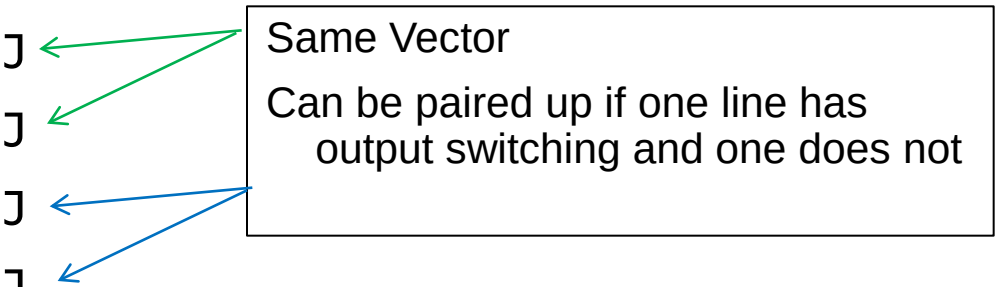
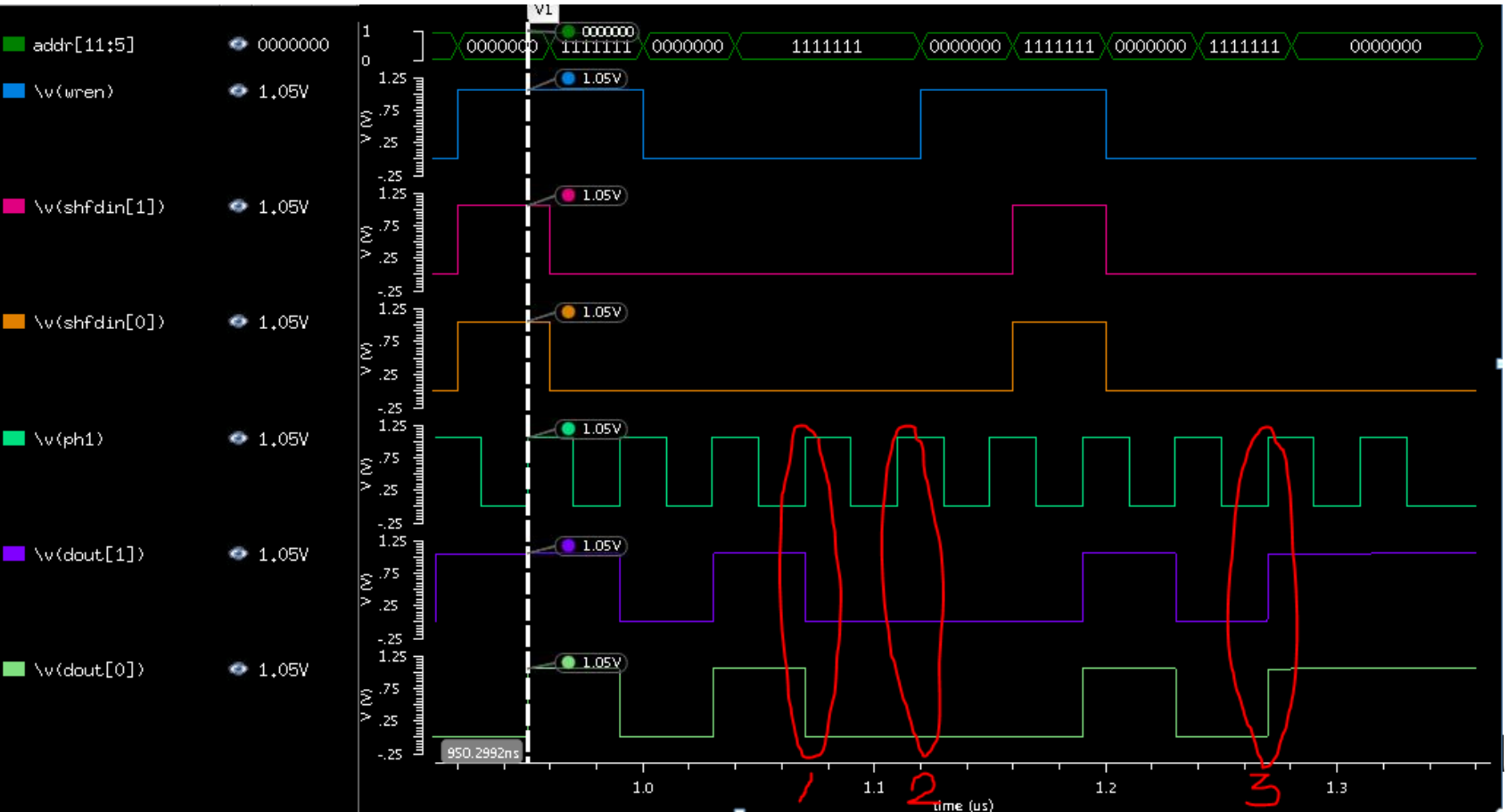


table	power_dout:								good for negtive glitch (reset to "0") on sequential "1" at output pin
pins	ph1	addr	shfdin	wren	byte	en	banken	dout	# comment
W1_AL	R	L	H	H	H	H	H	X	# All bit of AL at “1”
W0_AH	R	H	L	H	H	H	H	X	# All bit of AH at “0”
R1_AL	R	L	L	L	L	H	H	X	# Output initialization
output_power	R	H	L	L	L	H	H	J	# clock rising hidden power + dout **falling** power
output_power	R	H	L	L	L	H	H	J	# clock rising hidden power + dout **keeping** power(=0)
W0_AL	R	L	L	H	H	H	H	X	# All bit of AL at “0”
W1_AH	R	H	H	H	H	H	H	X	# All bit of AH at “H”
R0_AL	R	L	L	L	L	H	H	X	# Output initialization again
output_power	R	H	L	L	L	H	H	J	#clock rising hidden power + dout **rising** power
endtable									



ence®

How to Write the Table File

Step 1

- Write the arctypes and fastspice lines in the file

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

OR

```
arctypes power
fastsim spectre
fastsim_cmd_option -64 +spice +xps=s3 +cktpreset=sram_pwr
```

OR

```
arctypes leakage
fastsim spectre
fastsim_cmd_option -64 +spice +xps +cktpreset=sram_pwr
```

How to Write the Table File

Step 2

- Write a dummy cycle for the first cycle

```
table dummy_line
pins         CLK   DATA   ADR   Q
dummy         L     L      L    X
endtable
```

- Make sure that the measurement is set to “X” for this line
- The name “dummy_line” is the name of the table. It can be any name, but it must be unique in the table file.
- The name “dummy” is a comment string for that line. It is required, but its only purpose is as a comment.

How to Write the Table File

Step 3

- Write a table to measure setup and hold

```
table ADR_setup_hold
pins        CLK  DATA  ADR  WE  Q
Init         L    L     L   L   X
ADR_rise     R    L     1   L   C
ADR_fall     R    L     0   L   C
endtable
```

- Make sure that the line before a “1” is either a “L” or a “0” to get a transition
- Put a “C” under the output pins to measure constraints
- Set all side conditions to satisfy when conditions
- Include multiple side conditions, if necessary, to ensure simulation of the worst case

How to Write the Table File

Step 4

- Write a table to measure delay for a read operation

```
table Write_Read
pins        CLK   DATA   ADR   WE   Q
Write_0_A1   R     L      H    H    X
Write_1_A0   R     H      L    H    X
Read_0_A1    R     H      H    L    X
Read_1_A0    R     H      L    L    D
Read_0_A1    R     H      H    L    D
endtable
```

- Get the functionality from either a datasheet, waveform file, or by asking the circuit designer.
- Write the memory first to initialize the contents
- The first read is necessary to initialize the Q pin and ensure a transition for the later reads
- Setup read cycles to read both high and low from near and far addresses
- For write-bypass design, measurement on Q should be set to “D” for write-bypass delay measure

How to Write the Table File

Step 5

- Write a table to measure clk power for a write and read operation

```
table Write_Read_Power
pins        CLK  ADR  DATA  WE  Q
Write_5_A1   R    H   0x5   H   X
Write_A_A0   R    L   0xA   H   X
Write_3_A1   R    H   0x3   H   J
Write_3_A1   F    H   0x3   H   J
Write_6_A0   R    L   0x6   H   J
Write_6_A0   F    L   0x6   H   J
Read_3_A1    R    H   0x6   L   X
Read_3_A1    R    H   0x6   L   J
Read_3_A1    F    H   0x6   L   J
endtable
```

- Get the functionality from either a datasheet, waveform file, or by asking the circuit designer.
- Write the memory first to initialize the contents
- Write the memory again with half the bits flipping to measure current
- Read from the memory to initialize the output to a known state
- Read from the memory again with no outputs switching to measure current

How to Write the Table File

Step 6

- Write a table to measure power for input pins switching

```
table ADR_Power
pins        CLK   DATA   ADR   WE   Q
Init         L     L      L    L    X
ADR_rise     L     L      1    L    J
ADR_fall     L     L      0    L    J
endtable
```

- Only switch 1 pin on each line to isolate contribution
- Set side conditions to satisfy when conditions

How to Write the Table File

Step 7

- Write a table to measure leakage current

```
table Leakage
pins        CLK   DATA   ADR   WE   Q
Init         R     L      L    L    X
Measure      L     L      L    L    W
endtable
```

- First line initializes all latches
- Set side conditions to satisfy when conditions
- Repeat for multiple when conditions

Module 10: The Liberate MX Standard Custom Flow

Module Objectives

- In this module, you
  - Will be introduced to the Liberate MX define_memory command
  - Will see an example of the define_memory flow
  - Will be able to setup and run an instance using the standard custom flow

Automated Standard Instance Flow

- User must provide pin names for the following
  - Clock
  - Address
  - Data in
  - Data out
  - Write enable (must specify direction for write)
  - Bit write (optional, must specify direction for masked)
  - Chip Enable (must specify direction for enabled)

153

mx.tcl examples: Standard Instance

```
define_memory \
    -foundry {SMIC} \
    -netlist sram.spf \
    -models include_ss \
    -ref_lib ref_lib/sram.lib \
    -clock CLK \
    -address A \
    -data_in D \
    -data_out Q \
    -chip_enable {CEN L} \
    -write_enable {WEN L} \
    -port_suffix {A B} \
    -rail {VDD 1.08 VSS 0} \
    -global_voltage 1.08 \
    -temp 125 \
    -debug 1 \
    sram

char_memory
```

```
define_memory \
    -foundry {SMIC} \
    -netlist sram.spf \
    -models include_ss \
    -template [pwd]/template.tcl \
    -clock CLK \
    -address A \
    -data_in D \
    -data_out Q \
    -chip_enable {CEN L} \
    -write_enable {WEN L} \
    -port_suffix {A B} \
    -rail {VDD 1.08 VSS 0} \
    -global_voltage 1.08 \
    -temp 125 \
    -debug 1 \
    sram

char_memory
```

mx.tcl examples: Standard Instance (Cont.)

```
define_memory \
  -mx_setting [pwd]/mx_options_rom.tcl \
  -netlist [pwd]/rom.typical.90c.ckt_cedar \
  -models [pwd]/include_tt \
  -design {rom} \
  -clock {eeph1} \
  -address {addr} \
  -data_out {romdata} \
  -chip_enable {clken H} \
  -xps_smode_power yes \
  -temp 90 \
  -rail {vdd0 1.05 vss0 0} \
  -global_voltage 1.05 \
  -slews {0.008 0.023 0.038 0.068 0.128} \
  -loads {0.012968 0.017968 0.027968 0.047968} \
  -qloads {0.012968 0.017968 0.027968 0.047968} \
  {rom}
#set ::create_compiler_template_only true
char_memory
```

Define_memory flow: Standard Instance Flow

In order to use the define memory standard instance flow, the instance must meet certain criteria

- Must be an SRAM or ROM based design. This flow is not appropriate for CAM
- Operations must complete within one clock cycle.
- Must write to address when in write mode and clock is triggered
- Must read from address when in read mode and clock is triggered

The following design types are not supported:

- Memory with unlatched inputs
- Pre-decoded Addresses
- One Hot addresses
- Pipeline mode (Here the current output is from a read during previous cycle)
- Output glitch

Define_memory flow: Pin Names

- The names of the pins will be assembled from the following:
  - Any needed prefix for test
  - Base name for the pin
  - Any needed suffix for port
  - Bus information from the template or reference lib

Define_memory flow: spice inputs

- Providing the netlist and models
  - *define_memory –netlist* <netlist_name>
    - Can be extracted or pre-layout. Accuracy of lib will depend on accuracy of the netlist.
    - Should have .subckt for the full instance
    - Power supplies in the .subckt line
  - ***define_memory –models*** <model_file>
    - File contains the statements to use the models
    - Ex: .lib <modelfile> SS

Define_memory flow: Pin Names

- The names of the pins will be assembled from the following:
  - Any needed prefix for test
  - Base name for the pin
  - Any needed suffix for port
  - Bus information from the template or reference lib

Define_memory flow: PVT

- Defining the Power supplies and temperature
  - ***define_memory –voltage*** { <supply1> <supply1_value> <supply2> <supply2_value> …}
    - Define all power supplies for the instance with their values
  - ***define_memory –temp*** <temperature>
    - Define the temperature to run the characterization

Define_memory flow: pins

- Defining the pins and their functions
  - ***define_memory –clock*** <base name of clock>
  - ***define_memory –address*** <base name of address>
  - ***define_memory –data_in*** <base name of data in>
  - ***define_memory –data_out*** <base name of data out>
  - ***define_memory –read_timer*** <base name of read timer>
    - All values will be used to read delay timing
  - ***define_memory –read_timer_enable*** <base name of read timer enable>

Define_memory flow: control pins

- Defining the pins and their functions
  - ***define_memory –chip_enable*** {<chip enable base name> <chip enable active state>}
    - Base name of the chip enable pins followed by the value for chip is active
  - ***define_memory –write_enable*** {<write enable base name> <write enable active state>}
    - Base name of the write enable pins followed by the value for write mode
  - ***define_memory –bit_write*** {<bit write enable base name> <bit write enable active state>}
    - Base name of the bit write enable pins followed by the value for bit masked mode
  - ***define_memory –bypass_enable*** {<bypass enable base name> <bypass active state>}
    - Base name of the bypass enable pins followed by the value for memory is in bypass mode

Define_memory flow: control pins

- Defining the pins and their functions
  - ***define_memory –scan_enable*** {<scan enable base name> <scan enable active state>}
    - Base name of the scan enable pins followed by the value for scan is active
  - ***define_memory –scan_in*** <base scan in name>
  - ***define_memory –scan_out*** <base scan out name>

Note: It is assumed that if scan in is toggled and the memory is clocked in scan mode, that eventually the scan in data will appear on scan out.

Define_memory flow: control pins

- Defining the pins and their functions
  - ***define_memory –sleep*** {<sleep base name> <sleep active state>}
    - Base name of the sleep pins followed by the value for in sleep mode
  - ***define_memory –virtual_rails*** {<virt_rail_1> <voltage_1> <virt_rail_2> <voltage_2> …}
    - List of virtual rails, each followed by their normal voltage level
    - Must be the exact hierarchical name of the virtual rail

Define_memory flow: simulators and other options

Specify the simulators to be used

- ***define_memory –part_spice*** <simulator>
  - Specifies the simulator to be used for the full instance simulation portion
- ***define_memory –char_spice*** <simulator>
  - Specifies the simulator to be used for the partition simulation portion
- ***define_memory –debug***
  - Controls the level of debug information to be provided to the user
- ***define_memory –rcdb***
  - Invokes the rcdb database management for large instances
  - Requires spf format
- ***define_memory –greybox***
  - Generates a lib file directly from the full instance simulation results
  - Used for debug and flow testing
  - Can be used with aps as fastspice simulator to get full spice accuracy

Define_memory flow: other included files

- Some information is passed from other files
  - ***define_memory –template*** <file>
  - ***define_memory –ref_lib*** <file>

    - Do not need to provide both template and reference lib

Define_memory flow: other included files

- Some information is passed from other files
  - ***define_memory –mx_setting*** <file>
    - Sources in a file with other mx variables defined
    - Sourced right before the ***char_macro*** command
    - Values in this file will overwrite any internally defined values
    - Any necessary ***define_leafcell*** statements can go here

Define_memory flow

- Supplementing Tables
  - ***define_memory –additional_tables*** {<table>} …

Allows the user to supplement the generated vectors with custom vectors
- Useful for cases where there is custom functionality that is not handled by the default vectors
- Allows the user to specify other vectors to be considered

Define_memory flow: Debug

- Enabling additional debug information
  - ***define_memory –debug 1 …***

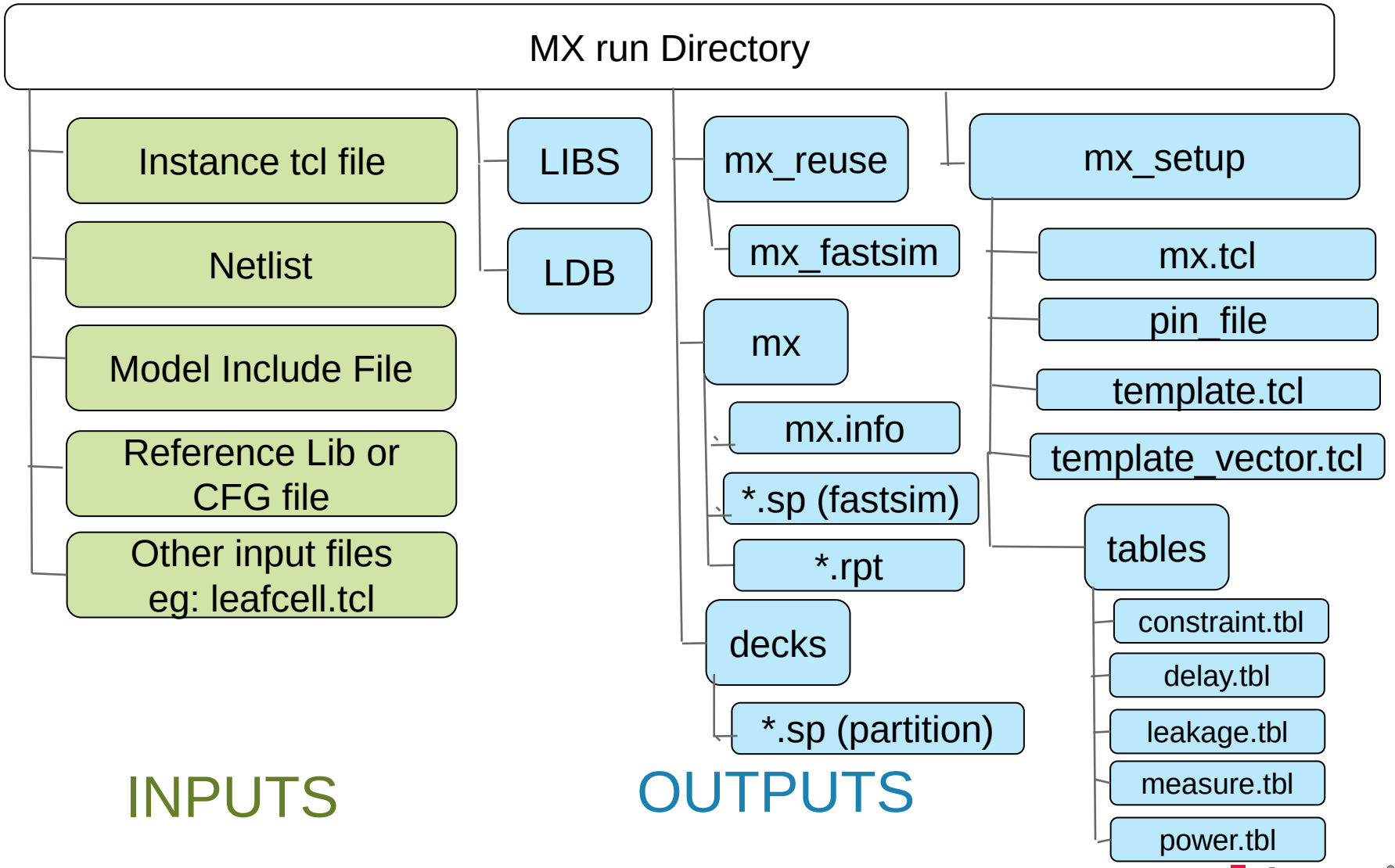
    - Print Verbose debug information
    - mx_spv_api is turned on
    - pin_file is generated showing the mapping of each input pin to its purpose

Module 10: Understanding the outputs of the define_memory flows

Module Objectives

- In this module, you
  - Will understand the output directory structure of the define_memory (standard instance and vendor re-characterization) flows
  - Will be able to debug and understand the generated files

Directory Structure



cadence

Module 11: The Liberate MX Vendor Instance Re-characterization Flow

Module Objectives

- In this module, you
  - Will be introduced to the Liberate MX define_memory command for re-characterizing third party vendor IP
  - Will understand how to setup an instance in this flow

Automated Vendor re-Characterization Flow

- Re-characterization of instances from TSMC, Virage (synopsys) and ARM is supported
- User provides the following
  - Reference Lib file
  - Netlist
  - Models and PVT
  - Virtual Rails (if applicable)
  - Foundry or leafcell definitions
- User does not need to provide table files, mxchar.tcl, template.tcl and most of the options and settings in mx.tcl
- Similar to the Standard Instance flow, but there are additional internal settings that are possible through knowledge of the vendor designs

Example: Virage Instance re-Characterization

```
define_memory \
    -netlist sram.spf \
    -models models_TT.inc \
    -ref_lib sram.lib \
    -vendor Virage \
    -global_voltage 0.9 \
    -temp 25 \
    -foundry TSMC \
    -xps_smode_power yes \
    sram

char_memory
```

Currently Supported Vendor Compilers

- The define_memory method does not to be customized for every memory compiler. It is expected that it will be able to run all designs in this table. The X's denote which compilers have already been tested.

		Architecture					
		1P SRAM	1P RF	2P SRAM	2P RF	pseudo 2P RF	ROM
TSMC	28nm	X	X	X			
	40/45nm				X		
	55/65nm						
	90nm						
	130nm						
	180nm						
Virage	28nm	X					
	40/45nm	X	X	X		X	X
	55/65nm						
	90nm						
	130nm						
	180nm						
ARM	28nm						
	40/45nm	X	X	X	X		X
	55/65nm						
	90nm						
	130nm						
	180nm						

Review the Directory Structure from the Demo Instance

Define_memory flow: Generated mx.tcl

- Contains all of the mx commands and options used in the characterization
- Specifies all include file locations
- Review from Demo

Define_memory flow: Generated template.tcl

- Contains instance specific information
  - ***define_cell*** command specifying the pinout of the instance
  - Thresholds for external pins
  - Arc definitions
- Provided to MX as a generated template or internally derived from a provided reference lib
- Review from Demo

Define_memory flow: Generated Tables

- Contain the vectors that will be used to run the full instance simulation
- Will usually contain the following files
  - Delay
    - For delay and retain
  - Constraints
    - For setup and hold
  - Measure
    - For clock period and mpw
  - Power
    - For dynamic power information
  - Leakage
    - For static current information
- Review from Demo

Define_memory flow: Final Outputs

- LDB
  - Contains the library database files
- LIBS
  - Contains the final generated lib files
- Review in Demo

Define_memory flow: Modifying the setup

- Edits can be made to the files in the mx_setup area
- The following command can be run to utilize the new edits for the run

```
liberate_mx mx_setup/mx.tcl
```

- Demonstrate in Demo

Module 12: Model Generation

Module Objectives

- In this module, you
  - Set the desired Library units
  - Create a Library (.lib) with user defined attributes and functions
  - Create Verilog® (.v) and VHDL (.vhdl) libraries
  - Generate datasheets

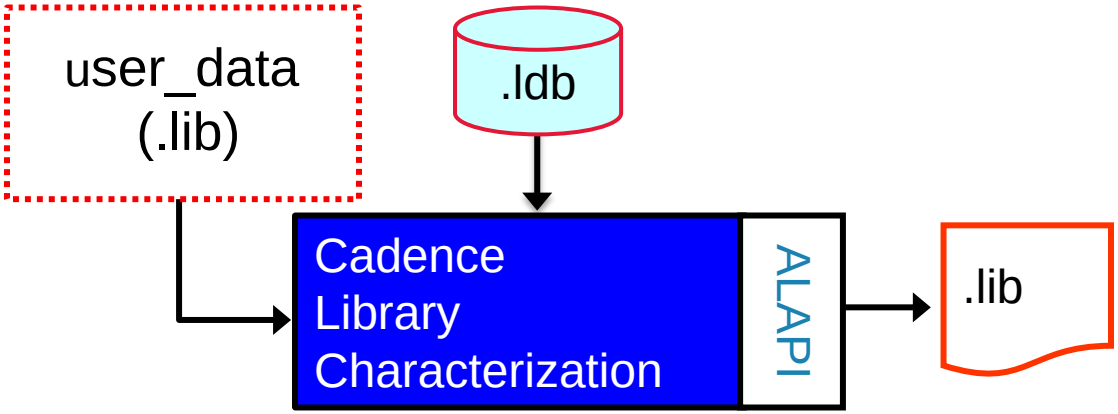
Generating Output Models

- Characterization should be run separately from model generation.
- Liberate utilizes the ALtos API (aka ALAPI) to generate models such as

  Liberty, Verilog, Vital/VHDL, and Datasheets
- Models are created using write_* commands.
- Customized models can be created using ALAPI. Some examples are available for reference in the directory $ALTOSHOME/etc (but do not modify these).

  ALAPI_* commands are documented in the ALAPI Manual (see ${ALTOSHOME}/docs.)

Creating Libraries (.lib)



- Liberate creates all libraries from the library database (.ldb) after characterization is complete.
- Liberate can generate libraries with different data formats from a single ldb dependent on the data characterized.
- Library Customization/augmentation is accomplished through the use of user_data file(s).

The *read_ldb* Command

- The ***write_ldb*** command will create a .ldb file. The .ldb:
  - Can be used to create models with previously characterized data.
  - Can be used to restart previously failed characterization runs.
  - Cells can be added to an existing .ldb
- The ***read_ldb*** command will read the .ldb into memory.
- The .ldb contains variable settings and may contain commands. Reading an ldb can change your settings by restoring previous ones!
- This command must be used before char_library

Syntax:

read_ldb

-check_driver_waveforms Checks both pre-driver and active driver-generated waveforms in ldb to verify that they have the correct slews

-remove **{list of cells}** Specify a list of cells to remove from the library database (Default: { })

<filename> A library database

Example 1:

read_ldb *typ_pvt.ldb.gz*

Example 2: Recharacterize 2 cells by first removing them.

read_ldb –remove *{invx1 dffx2} typ_pvt.ldb.gz*

The *set_units* Command

- Use the ***set_units*** command to define the units to be applied in the output library.
- Use the defined units globally for all cells and data in a single characterization run.

  Liberate and the liberty format does not support cell or arc-based units.
- The ***set_units*** command is not required for characterization but must be set prior to creating a library.
- The default units are:

Timing:	1 NanoSecond
Capacitance :	1 PicoFarad
Leakage :	1 NanoWatt
Current :	1 MilliAmpere

The *set_units* Syntax

- ***set_units***

-timing <1ns | 100ps | 10ps | 1ps>	Specify timing units
-capacitance <1pf | 100ff | 10ff | 1ff >	Specify capacitance units
-leakage_power <1mw | 1uw | 1nw | 1pw>	Specify leakage power units
-current <1a | 1ma | 1ua>	Specify the current units

- Example

*set_units *
*–timing 1pS *
*–capacitance 1fF *
*–leakage_power 1uW *
-current 1ua

- Note: Units may combine uppercase and lowercase…1uw and 1uW are both acceptable.

The *write_library* Command

- The ***write_library*** command creates Liberty format libraries.
- Supports all released formats of Liberty syntax from Synopsys and Cadence (NLM, ECSM and CCS).
  - The data for the desired output format must be in the *.ldb* file.
- Supports ***-user_data*** file to:
  - Augment output library format with non-characterized data
  - Add library, cell, and pin level attributes to the generated library.
    Example: *delay_model*, *wire_loads*, *area*, *clock_gating_integrated_cell*.
  - Override specific characterized data in output library
    Example “function”
  - Follows Liberty file syntax.
  - Can be created from a golden library using ***write_userdata_library***.

The *write_library* Syntax

write_library

-ccs	Include CCS data
-ccsn	Include CCSN (noise) data
-ccsp	Include CCSP (power) data
-ecsm	Include ECSM data
-ecsmn	Include ECSM noise data
-ecsmp	Include ECSM power data
-user_data <filename>	User provided library data
-filename <filename>	The output file name
<libname>	The library name.

Examples of *write_library*

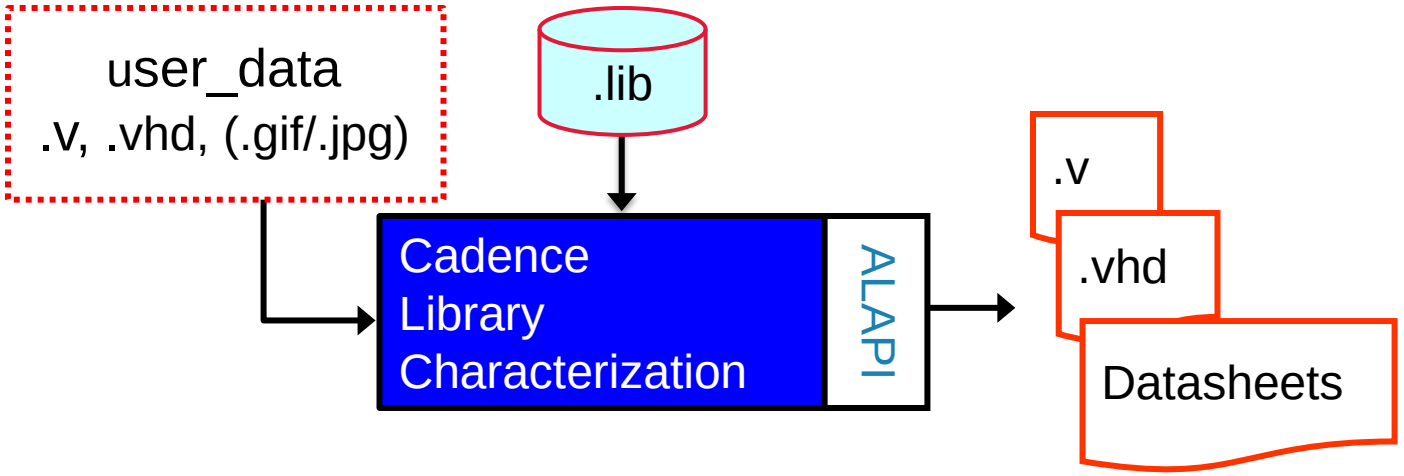
Example 1: Create a library in run directory for NLDM/NLPM.

write_library -filename *my_library.lib worst_case_lib*

Example 2: Create a library with NLDM/NLPM and ccs timing.

write_library -ccs \
-filename *${lib_dir}/wc_T125_v1p1.lib* \
wc_125C_1p1V

Verilog and VHDL Libraries



- Create Verilog/VHDL files from existing liberty (.lib) files.
  - Ensures .lib and .v / .vhdl are in-sync. Avoids potential mismatch between function, *sdf_cond*, etc. due to write_library post-processing.
- Customization of pre-existing .v and .vhdl files is supported using "***user_data***" files.

  user_data files provide verilog/Vital function, liberate will then add the appropriate timing (ex: specify block) to match the liberty model.
- Use ***write_verilog*** command to create Verilog models.
- Use ***write_vital*** command to create Vital models.

The *write_verilog* Syntax

write_verilog

-cells **{cell_names}**	List of cell names (Default: *all cells)*
-merge	Merge cell modules from user_data
-mux **<type>**	Type of primitives to create for MUX functions (Default: *basic logic primitives)*
-path **<path>**	The string used to denote a path. It must be either "=>", or "*>" (Default: "=>")
-sdf_version **<version>**	SDF version, must be 2.1, or 3.0 (Default: *2.1)*
-user_data **<filename>**	User provided Functional Verilog.
<verilog_filename>	Name of the output Verilog file

Example 1: Create a verilog library from an existing liberty library

```
read_library ${lib_dir}/wc_T125_v1p1.lib
write_verilog  -user_data ${lib_dir}/user_data.v \
 -sdf_version 3.0}  \
 ${lib_dir}/wc_T125_v1p1.v
```

The *write_vital* Syntax

write_vital

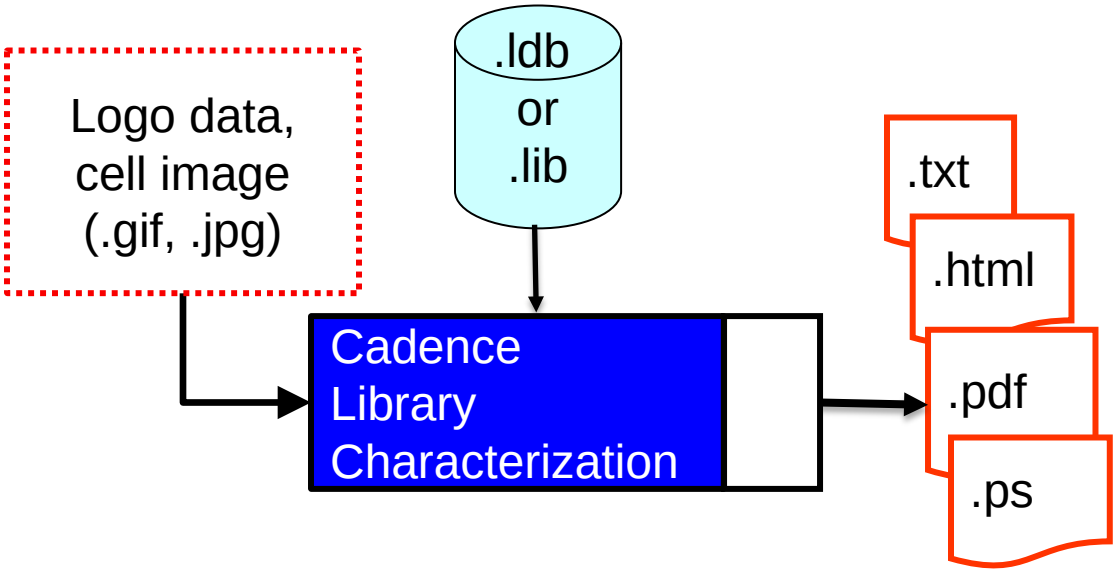
-cells **{cell_names}**	List of cell names (Default: *all cells)*
-merge	Merge cell modules from user_data
-sdf_version **<version>**	SDF version, must be 2.1, or 3.0 (Default: *2.1)*
-user_data **<filename>**	User provided Functional Verilog.
<verilog_filename>	Name of the output Verilog file

Example 1: Create a verilog library from an existing liberty library

```
read_library my.lib
# set timing check variables
set vital_timing_violation_format "Tviol_\$ip\\_\$rp\\_\$ref_edge"
set vital_recrem_violation_format "Rviol_\$ip\\_\$rp\\_\$ref_edge"
set vital_timing_info_format "Tinfo_\$ip\\_\$rp\\_\$ref_edge"
set vital_recrem_info_format "Rinfo_\$ip\\_\$rp\\_\$ref_edge"
# Output a Vital file
write_vital -user_data my_vital my.vhd
```

Note: For more controls, see the manual.

Creating Datasheets



- Datasheets are created from a .lib file.
- Output can be in text, HTML, PDF or postscript format.
- Graphics can be included in the form of *.gif* or *.jpg* files.
- Custom datasheets can be created by copying and altering the source code at $*ALTOSHOME/etc/datasheet.tcl*.

The *write_datasheet* Syntax

write_datasheet

- ***cells*** writes specified cells to the datasheet.
- ***conditional*** includes writing out all conditional arcs. (Default: output default groups only)
- ***dir*** **<dir_name>** HTML directory name
- ***filename*** **<file_name>** output a file name
- ***format*** **<text | html | pdf | ps>** Datasheet format. (Default: "*text")*
- ***groups*** **{ list }** list of groups to output
- ***include_indices*** includes indices in HTML reports.
- ***logo*** **<file_name>** Logo in .gif or .jpg format
- ***table_style*** specifies where to create the datasheet tables, from first-mid-last or min-avg-last. (Default: first-mid-last)

<library_name> datasheet library name and filename prefix

Examples of *write_datasheet*

Example 1: Write a text format datasheet to tt.txt

```
write_datasheet tt
```

Example 2: Write an HTML datasheet

```
write_datasheet \
    -format html \
    -dir tt_html \
    -logo liberate.gif \
    tt
```

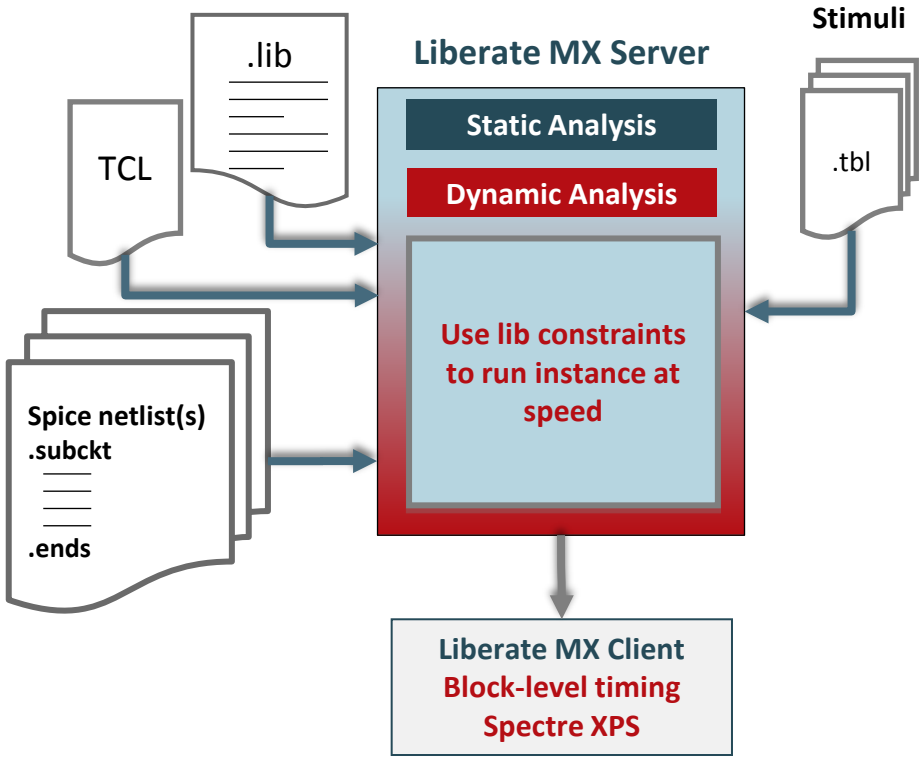
Module 13: The Liberate MX Validation Flow

Liberate MX Validation Flow

- The timing numbers that are reported in the lib file are used to run at speed simulation to prove functionality
- Vectors to be used for this test are provided by the user
- The user can review the generated waveforms to ensure that the desired functionality is maintained
- Simulations are run at the full instance level and there is no partitioning in this flow

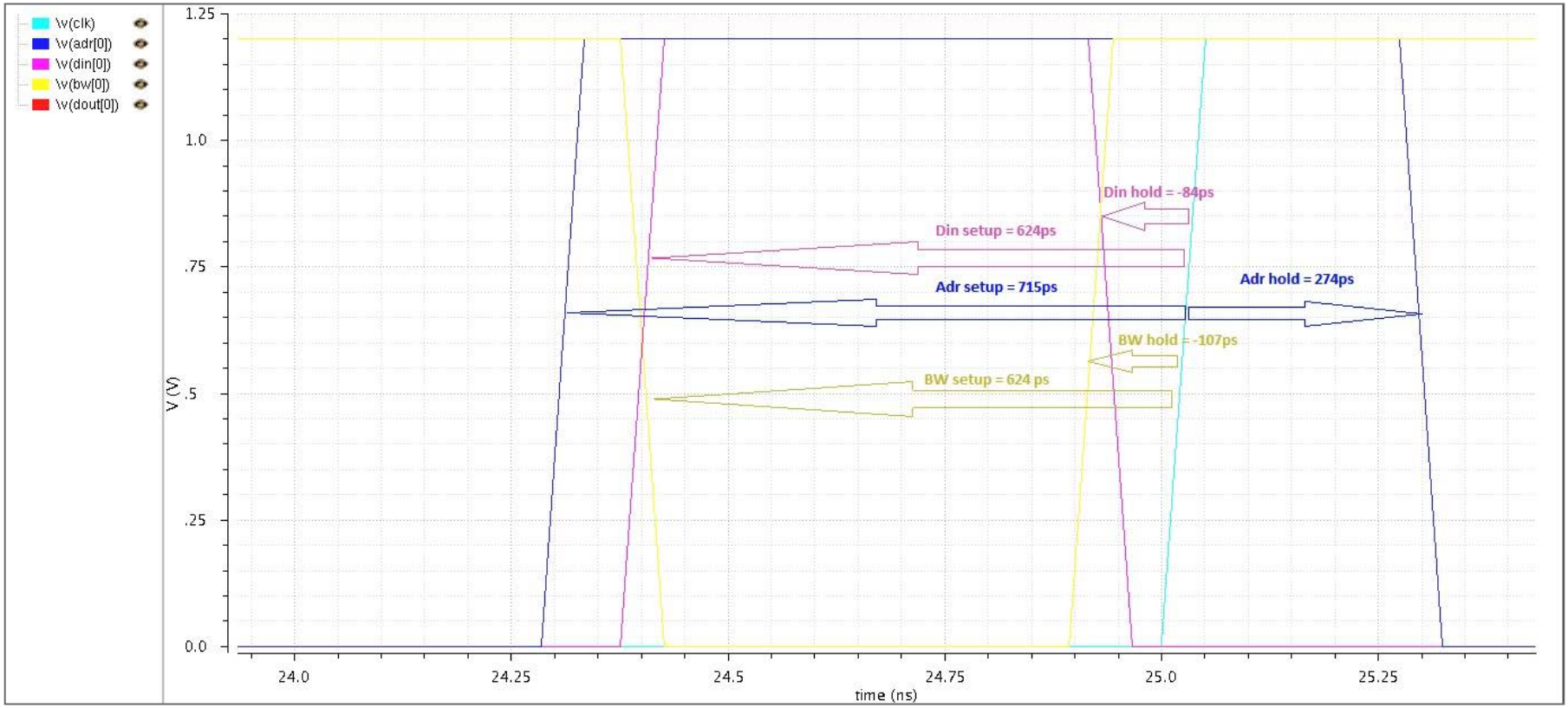
Liberate MX 13.1: Memory Library Validation

- Automated flow to validate memory library
- Run memory instance at speed using setup constraints from .lib
- Check memory functionality
  - Outputs toggling
  - Compare delays versus .lib
- Generate waveform for further debugging



cadence

Liberate MX Validation
Debugging Waveform



cadence

Liberate MX Validation setup

- The Liberate MX validation flow uses the same setup as the characterization with 3 exceptions
  - The template is written to include the timing values from the lib file
  - The run command is validate_macro instead of char_macro

Liberate MX Validation setup: template

- Generation of the template file for validation

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

- Will write the timing information into the template file

```
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
```

Liberate MX Validation setup: validate_macro

validate_macro

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

The circuit simulation program to be used for validation. Default: “xps"

-thread <number>

Specifies the maximum number of threads to be used on the host machine. Default: 0 (Let Liberate_MX decide).

Liberate MX Validation setup: tables

- Writing Tables for Validation
  - Tables should call spectre XPS/APS as simulator engine
  - Tables should be written to stress all arcs
  - Tables should be written in such a way as to cause a data out corruption if there is a violation of the constraints
  - For example, when exercising bist, the values should be opposite on regular and test to cause a behavior difference if there is a violation.

Liberate MX Validation setup: Example Table

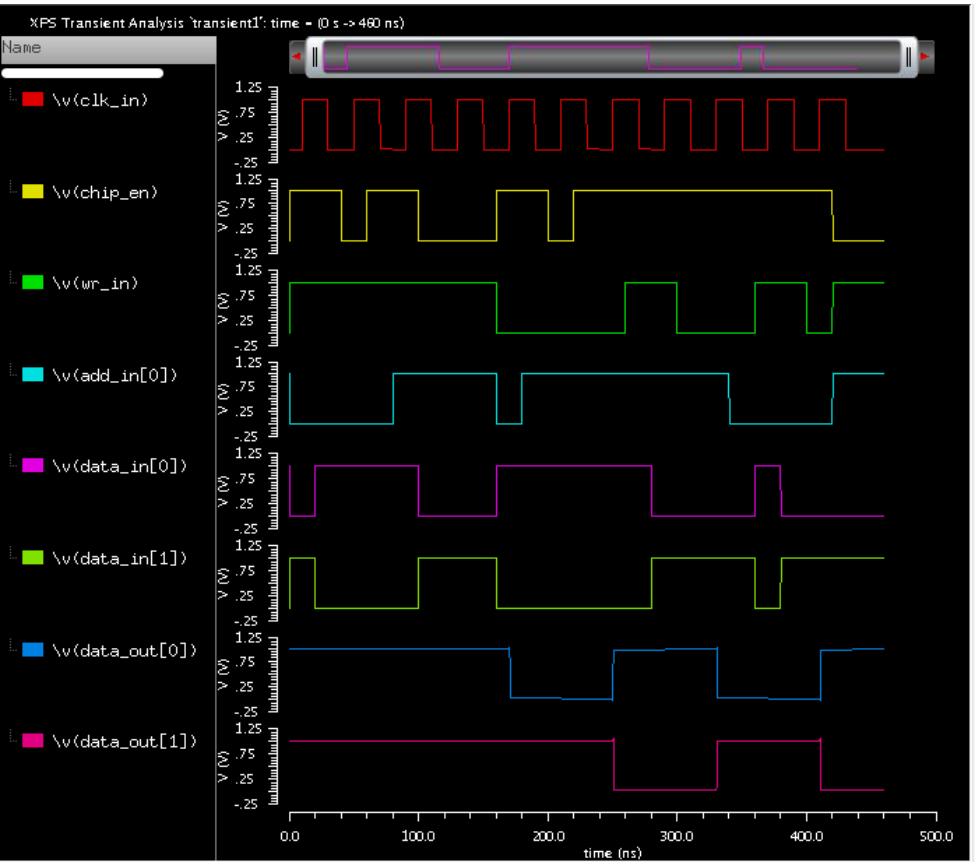
```
arctypes timing
fastsim spectre
fastsim_cmd_option +xps +spice +cktpreset=sram -format sst2 -O=4  -64
#fastsim_cmd_option -64 +spice +aps +mt=4 -f uwi -uwifmt sst2

table write
pins CLK A D    WEN CEN Q
W_A0 0   H 0xaa H   H   X
W_A0 1   L 0x55 L   L   D
W_A0 0   H 0xaa H   H   X
W_A1 0   L 0x55 H   H   X
W_A1 1   H 0xaa L   L   D
W_A1 0   L 0x55 H   H   X
endtable

table read_CLK_Q
pins CLK A WEN CEN Q
init 0   H L   H   X
init 1   L H   L   D
init 0   H L   H   X
R_A1 0   L L   H   X
R_A1 1   H H   L   D
R_A1 0   L L   H   X
R_A0 0   H L   H   X
R_A0 1   L H   L   D
R_A0 0   H L   H   X
endtable
```

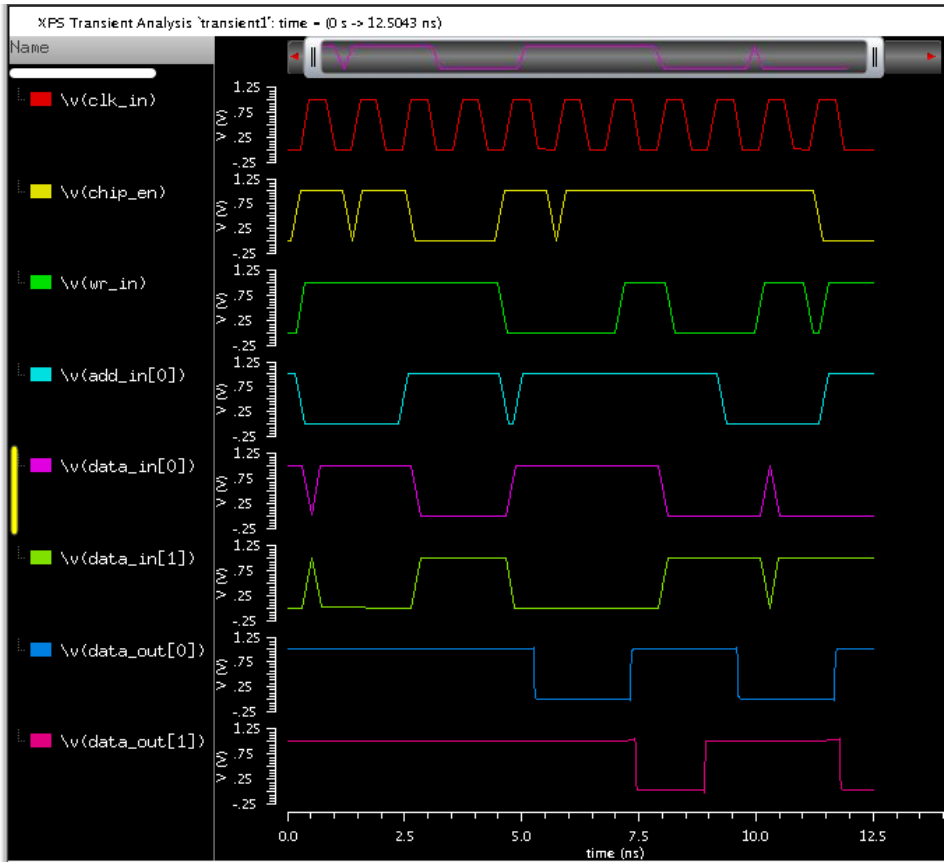
Liberate MX Validation Results

- The results of the validation run will be stored in the mx_reuse/mx_fastsim directory
  - mx_reuse/mx_fastsim/<inst>_validation_relax_0_/transient1.tran.trn
  - mx_reuse/mx_fastsim/<inst>_validation_stress_1_/transient1.tran.trn

- These files can be viewed in VIVA to determine if the required functionality is preserved

Validation Waveform





/tan/csv/xpsrd_testout2/xps/shujiexu/vas/altosmx_via/rom/template_test/rom_notmpl_cedar.tcl)

Module 14: Running Liberate MX

Running Liberate MX

- setenv ALTOS_64 1
- setenv ALTOSHOME < >
- setenv TMPDIR < >
- $ALTOSHOME/bin/liberate_mx mx.tcl |& tee mx.log

Note: All logging information will be produced to std error and std out. To preserve this information, it is necessary to redirect the output to a log file.

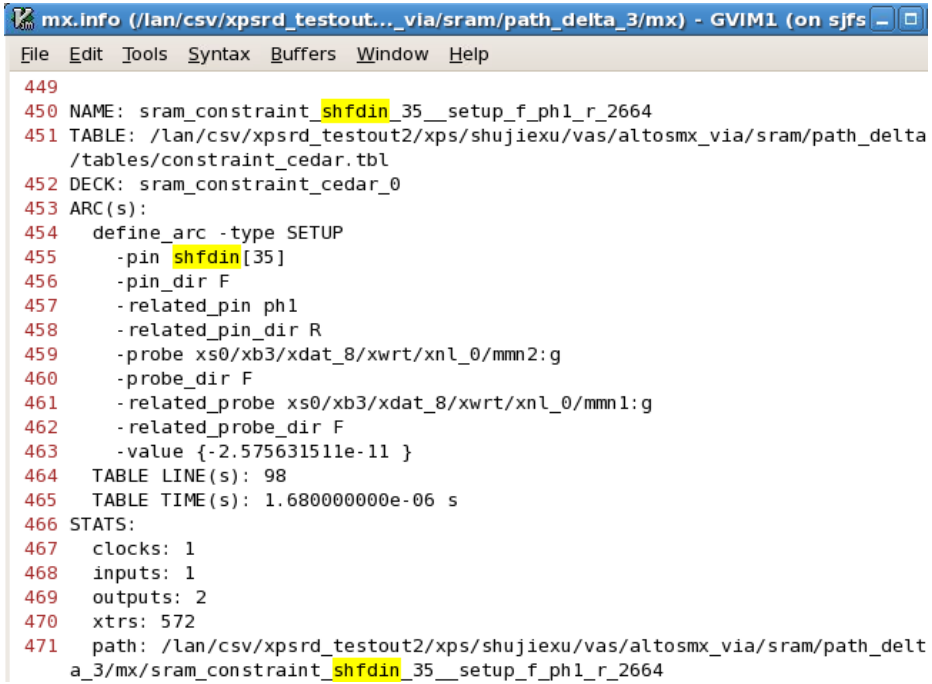
Module 15: Analyzing and Debugging Output

Lib File

- Review the Lib file produced in the demo run

mx.info

- Mx.info contains the information on the arcs that were characterized
  - Partition name
  - Probe point
  - Related probe point
  - Probe directions
  - Table and table line
  - Partition size
  - Values from fastspice simulation



- Review mx.info from the demo run

Fastspice Waveforms

- Waveforms for the fastspice full instance simulation can be viewed with standard waveform viewers

```
cd mx_reuse/mx_fastsim/<vector_set>
viva
```

Probe report

- A report can be created to summarize all considered probing locations for all measurements.
  - ***set_var mx_probes_report probes.rpt***

- Another report will be created with only the considered probes that are active in the fastspice. The name of this file will have an appended .red (probes.rpt.red)

Partitions

- Partition netlists are available in the decks directory
  - Path to partitions for a given arc are found in mx.info
  - Directory will contain
    - Spice deck
    - Spice output
    - Passed / failed file
- Show in Demo

True spice Waveforms

- In order to view waveforms for the full spice simulations
  - Untar the partition directory under decks
  - Remove the line "***save=nooutput***"
  - Add necessary print statements
  - Run spectre

Debug commands and options

- To report the result of automatic - and user specified - probing:
- ***mx_debug***
  - ***static*** Generates information on topology, such as channels connected, memory core cells, latches/combinational, and functional extraction.
  - ***clock*** Generates information on clock propagation.
  - ***pattern*** Generates information on mxtable expansion results.
  - ***dynamic*** Generates information on dynamic partitioning
  - ***activity*** Generates information on wire waveforms, as read back from fastspice.
  - ***arc*** Generates information on characterization arcs being identified during the partitioning process.

Debug commands and options

- For arcs using define_measure:

  set_debug measure 0x1

- If this is declared in the mx.tcl, will give information for each vector on the value of the measurement in the fast spice run
- If this is declared in the mxchar.tcl, will give information for each vector on the value of the measurement in the partition run

Define_measure debug example

```
(MDB) - evaluating measurement clkw_clkr_recovery in interval: [1.000000e-08:3.000000e-08]
(MDB) - user required: skipping index 1
(MDB) - evaluating measurement clkw_clkr_recovery in interval: [5.000000e-08:7.000000e-08]
(MDB) - skipping sim. index 5 for meaurement clkw_clkr_recovery as it's reserved for latch component of latch min period
(MDB) - skipping sim. index 6 for meaurement clkw_clkr_recovery as it's reserved for latch component of latch min period
(MDB) - evaluating measurement clkw_clkr_recovery in interval: [9.000000e-08:1.100000e-07]
(MDB) - skipping sim. index 9 for meaurement clkw_clkr_recovery as it's reserved for latch component of latch min period
(MDB) - skipping sim. index 10 for meaurement clkw_clkr_recovery as it's reserved for latch component of latch min period
(MDB) - skipping sim. index 11 for meaurement clkw_clkr_recovery as it's reserved for latch component of latch min period
(MDB) - evaluating measurement clkw_clkr_recovery in interval: [1.300000e-07:1.500000e-07]
(MDB) - user required: skipping index 13
(MDB) - evaluating measurement clkw_clkr_recovery in interval: [1.700000e-07:1.900000e-07]
(MDB) - checking evaluation of 'unmarked' measurement clkr_r_wlr_f_00 in interval starting at: 1.700000e-07
        ref0=1.700000e-07 simulation_interval=2.000000e-08 remainder(ref0, simulation_interval)=-1.000000e-08
(MDB) measure(clkr_r_wlr_f_00,1.70000000e-07,1.90000000e-07): trig=failed targ=failed - failed
(MDB) - checking evaluation of 'unmarked' measurement clkr_r_wlr_f_ff in interval starting at: 1.700000e-07
        ref0=1.700000e-07 simulation_interval=2.000000e-08 remainder(ref0, simulation_interval)=-1.000000e-08
(MDB) measure(clkr_r_wlr_f_ff,1.70000000e-07,1.90000000e-07): trig=failed targ=failed - failed
(MDB) - checking evaluation of 'unmarked' measurement clkw_r_wlw_r_00 in interval starting at: 1.700000e-07
        ref0=1.700000e-07 simulation_interval=2.000000e-08 remainder(ref0, simulation_interval)=-1.000000e-08
(MDB) measure(clkw_r_wlw_r_00,1.70000000e-07,1.90000000e-07): trig=1.70005000e-07 targ=failed - failed
(MDB) - checking evaluation of 'unmarked' measurement clkw_r_wlw_r_ff in interval starting at: 1.700000e-07
        ref0=1.700000e-07 simulation_interval=2.000000e-08 remainder(ref0, simulation_interval)=-1.000000e-08
(MDB) measure(clkw_r_wlw_r_ff,1.70000000e-07,1.90000000e-07): trig=1.70005000e-07 targ=1.70344369e-07 --> result: 3.39369356e-10
evaluate(clkw_clkr_recovery,1.70000000e-07,1.90000000e-07)=0.00000000e+00
(MDB) - evaluating measurement clkw_clkr_recovery in interval: [2.100000e-07:2.300000e-07]
```

Measurement name

Evaluation Time

Evaluation Result

Possible Issues

Static Partitioning takes forever

It should not take longer than few minutes to partition even the largest cases ...

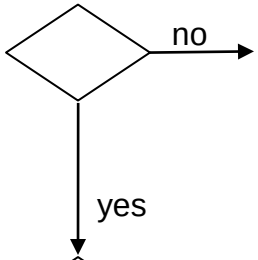
```
(MX-info) - Partitioning - static - start...
| (MX-info) - Partitioned   1815521/1818357 xtrs wall clock time +=   22 sec
| (MX-info) - Merging loops ...                 wall clock time +=    4 sec
| (MX-info) - Propagating constants ...         wall clock time +=    0 sec
| (MX-info) - Mem core cells found:  36 KB      wall clock time +=   14 sec
| (MX-info) - Getting clock trees ...           wall clock time +=    3 sec
| (MX-info) - ccc to gates ...                  wall clock time +=    9 sec
(MX-info) - Partitioning - static - ... end ___ wall clock time __   52 sec ___
```

Check you've properly defined rail nodes for the circuit - power, ground and virtual.

Possible Issues : Missing Delay Arc

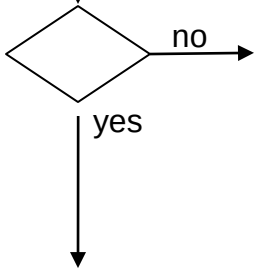
Check input patterns / truth tables

Valid sequence for the missing ARC? Not on the first line?



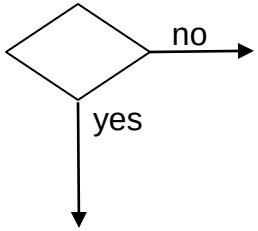
Add missing sequence and re-run.

Check wires waveforms.
Are pin/related indeed switching in the same simulation interval according to FastSPICE?



Check your table again / check your FastSPICE settings.

Check that expected delay < 0.5 * mx_simulation_interval. Is the window of observation is wide enough to capture the expected transitions?



Set mx_simulation_interval to at least twice the larges delay expected in the library and re-run (fastspice as well).

Contact Support

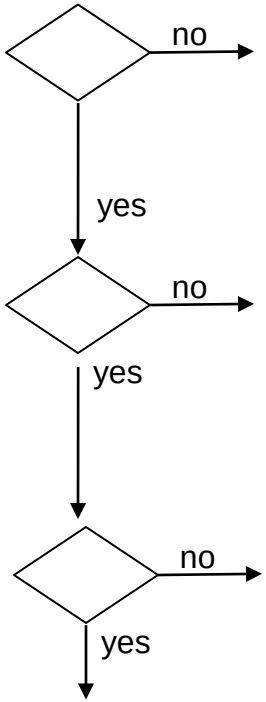
Possible Issues : Missing Constraint Arc

Check input patterns / truth tables

Do they contain a sequence for the missing ARC? Did you ensure that we are not measuring on the very first transition?

Check auto-probe identification. Did the tool find sequential / combinational logic on the intersection between data and clock trees?

Are the probes identified are indeed switching as expected?



Add missing sequence and re-run.

•Workaround: manually specify missing probes via define_arc command.
•Debug clock propagation.

Debug signal propagation

Contact Support

Possible issues: Missing Power Data

- Verify that the subckt pinout contains the power supply pins
- Verify that there are no global statements for the power supplies

Possible issues: Mismatching Library Values

- Matching to reference library
- Check the settings:
  - Spice models
  - Table indices
  - Slews and thresholds
  - Multipliers, library multipliers etc
- Use ***write_template*** to ensure MX uses the same settings in the golden/reference library

Correlating Data

Advanced Control Options

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

The greybox option can be used with a full spice simulator, such as APS, to prove partition accuracy. It can also be used with a fast spice simulator to generate a lib file for preliminary runs where ultimate accuracy is not required.

Module 16: ViVA Overview

Requirement on Server:

Linux RHEL 5.0 or above

Requirement on Tool Version:

IC616ISR3 and above (Edge Measurement feature)

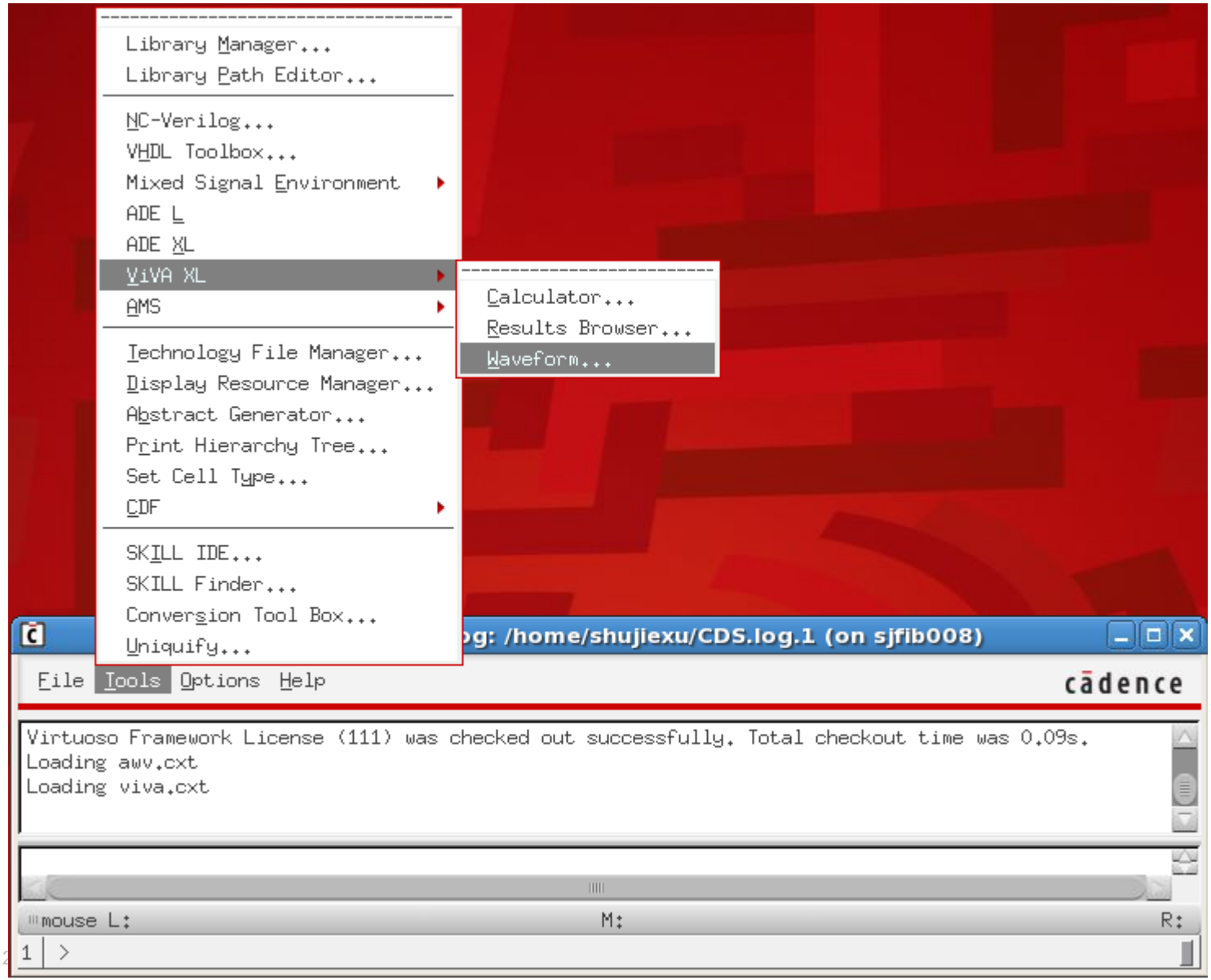
To enable EdgeMeasurement, set following environment variable in CIW or in .cdsinit file
envSetVal(“viva.rectGraph” “enableEdgeMeasurement” ‘string “true”)

Requirement on License:

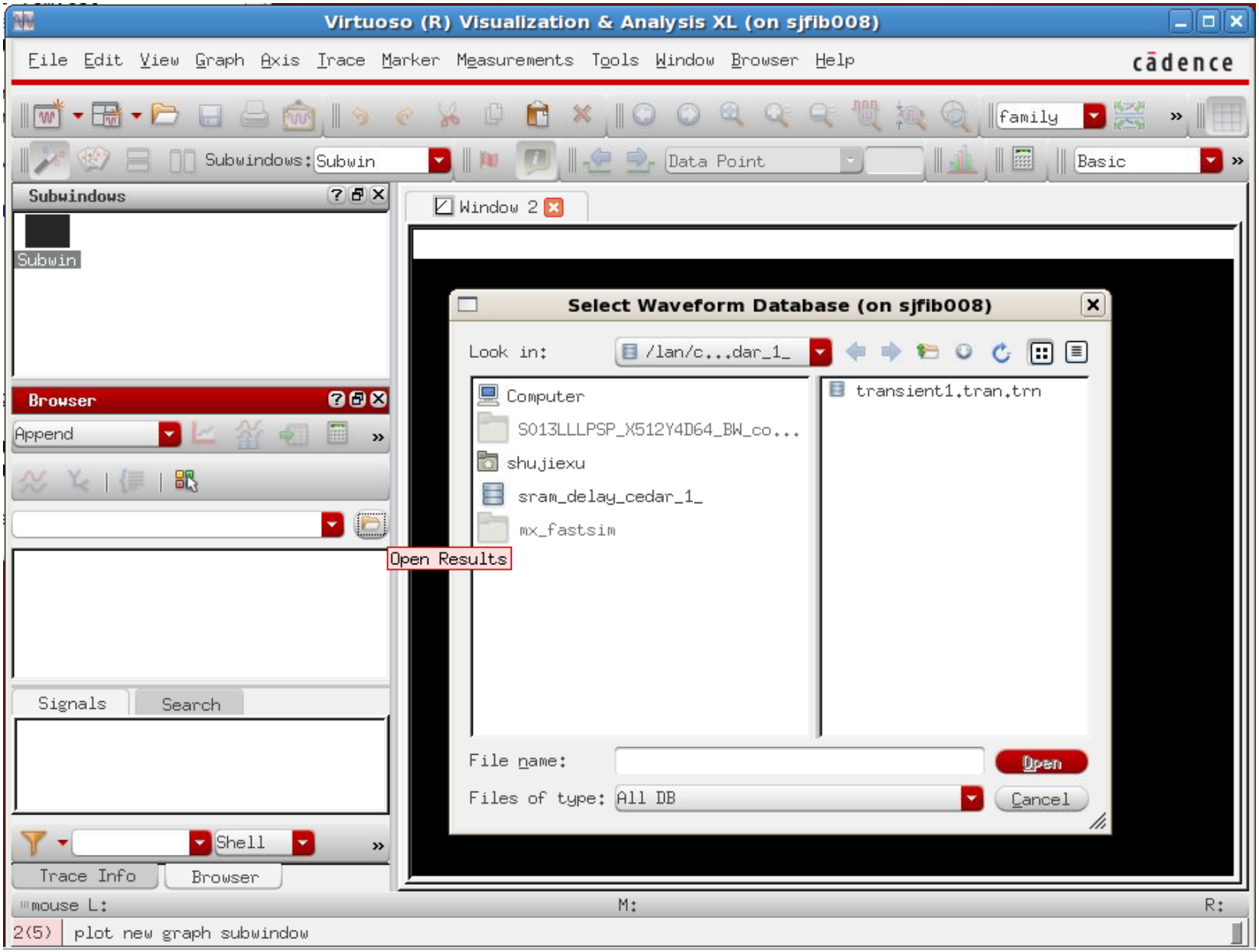
Product Id : 95255, [Version: 6.1.6]
Product Name: Virtuoso(R) Visualization & Analysis XL

.

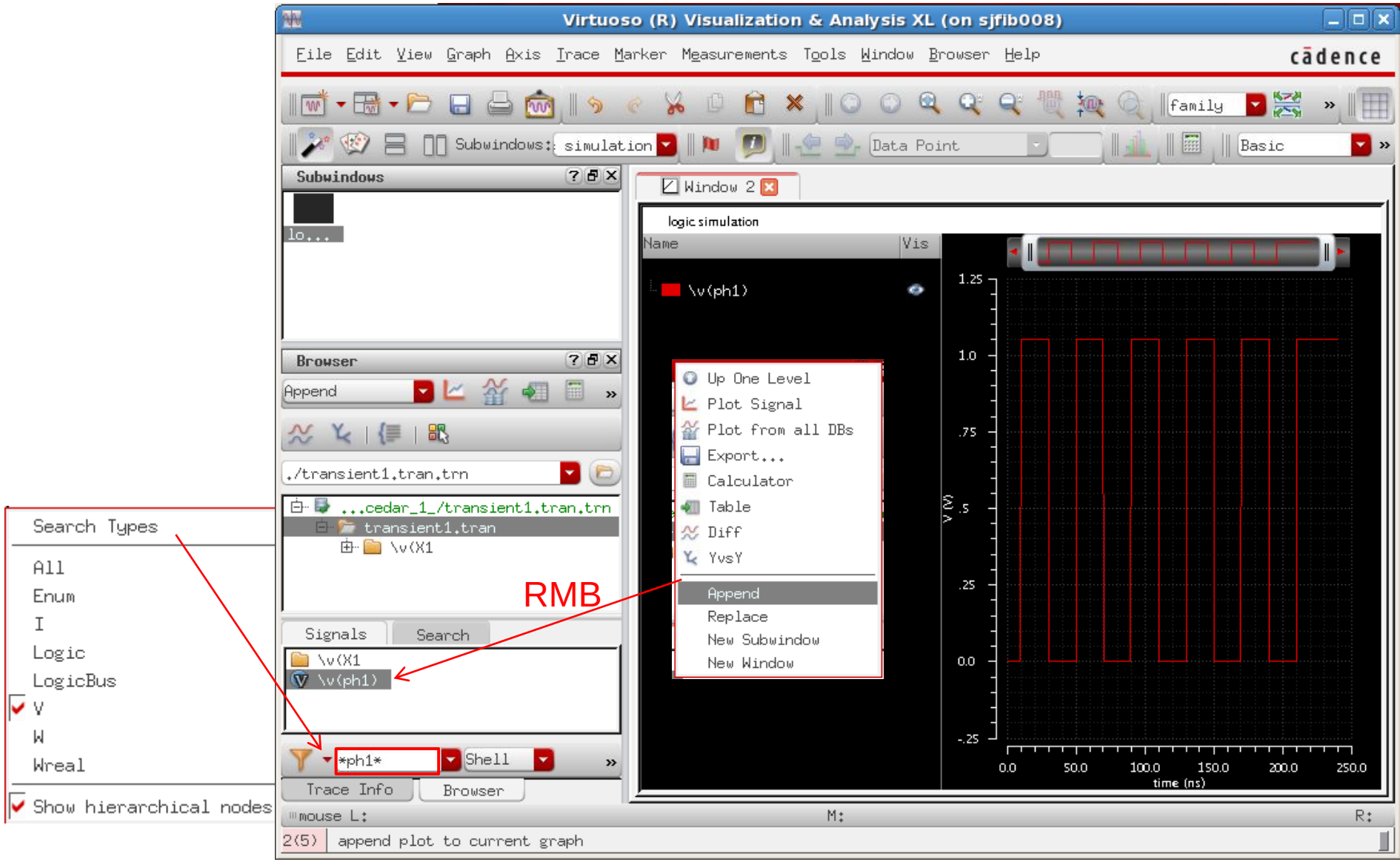
Launch ViVA XL through Virtuoso 61.6 CIW
OR with command “**viva -64 &**”



Open Results



Search Signal and Plot



cadence

Frequently Used Functions
zoom in / zoom out

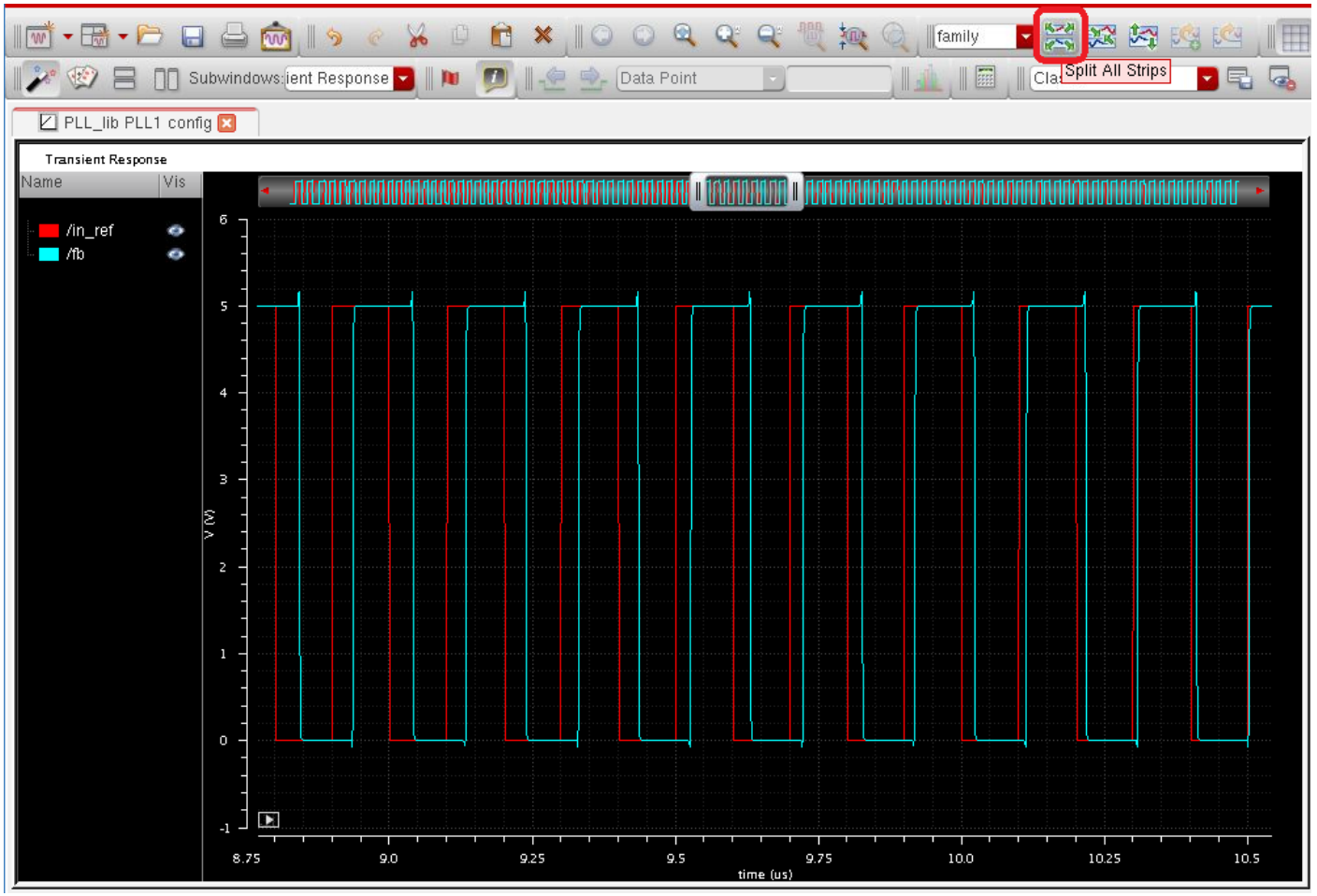
1. **Hold down the right mouse button (RMB) and drag to zoom in on any section of the graph.**
   Bindkey for Fit: f

2. **X-axis zoom in/zoom out:**
   shift + scroll mouse button

3. **Y-axis zoom in/zoom out:**
   ctrl + scroll mouse button

Frequently Used Functions

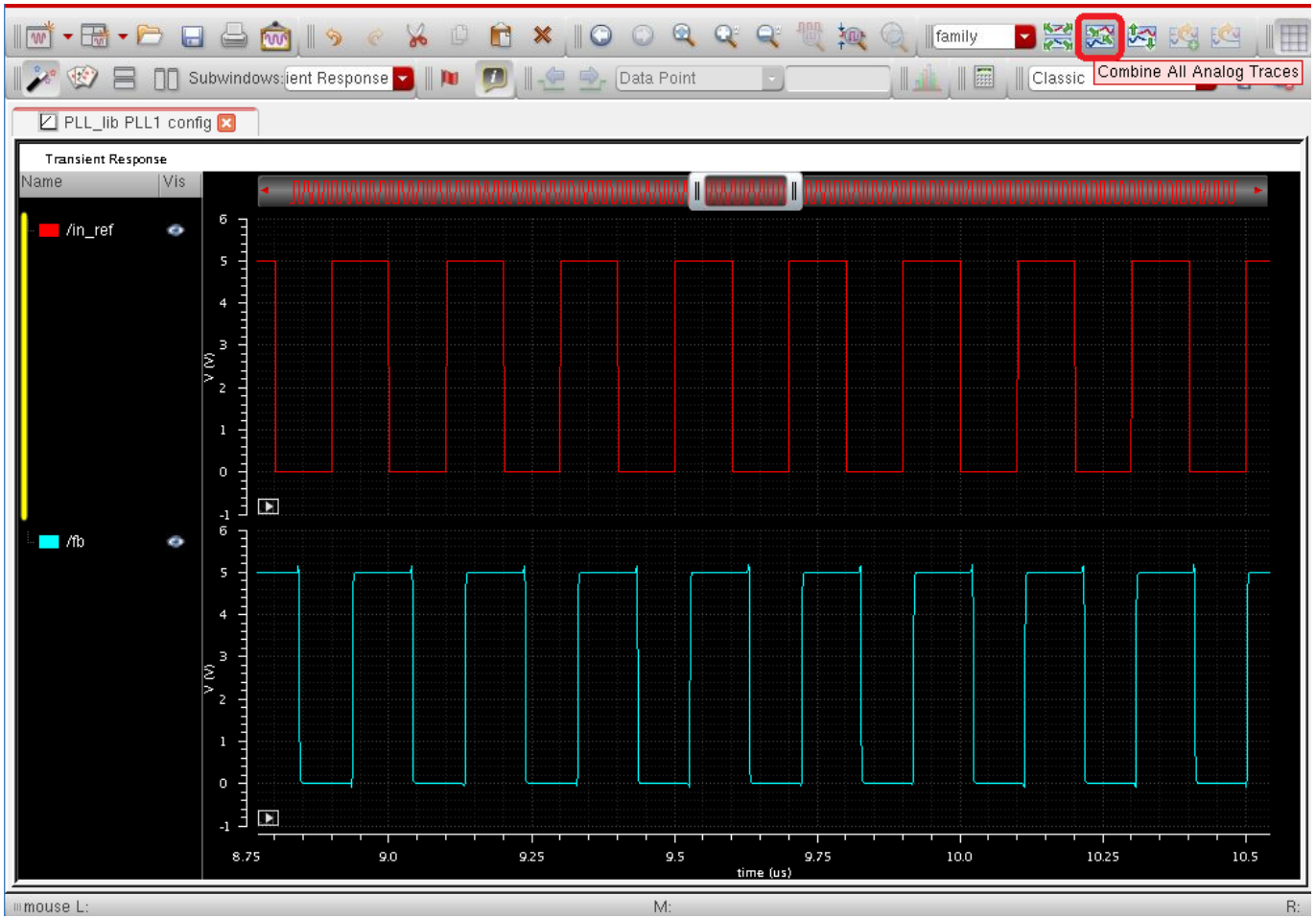
Split Strips



cadence

Frequently Used Functions

Combine All Analog Traces



cadence

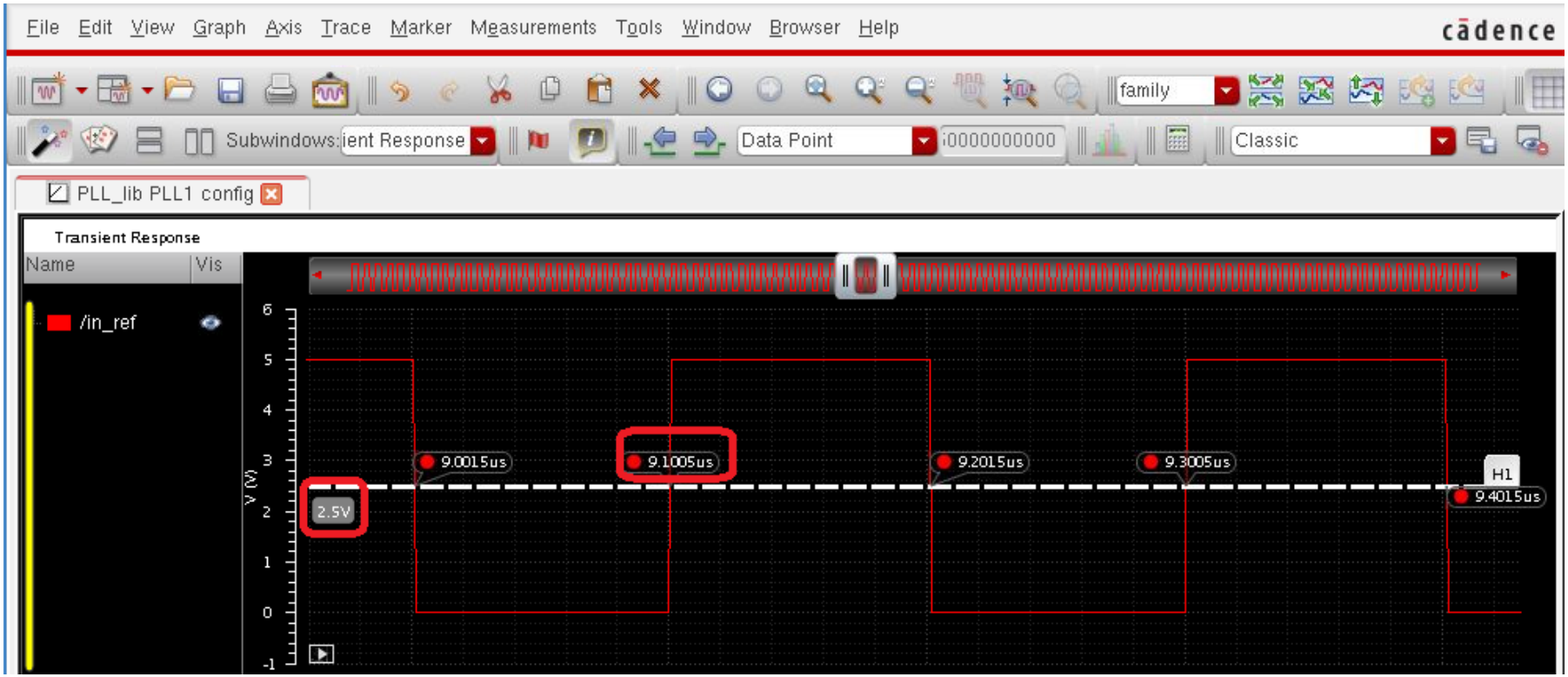
Frequently Used Functions
Point Marker

1. **Use bindkey m to add point marker;**

2. **Select 2 point markers, use bindkey d to get delta values between them.**

Frequently Used Functions
Horizontal Marker

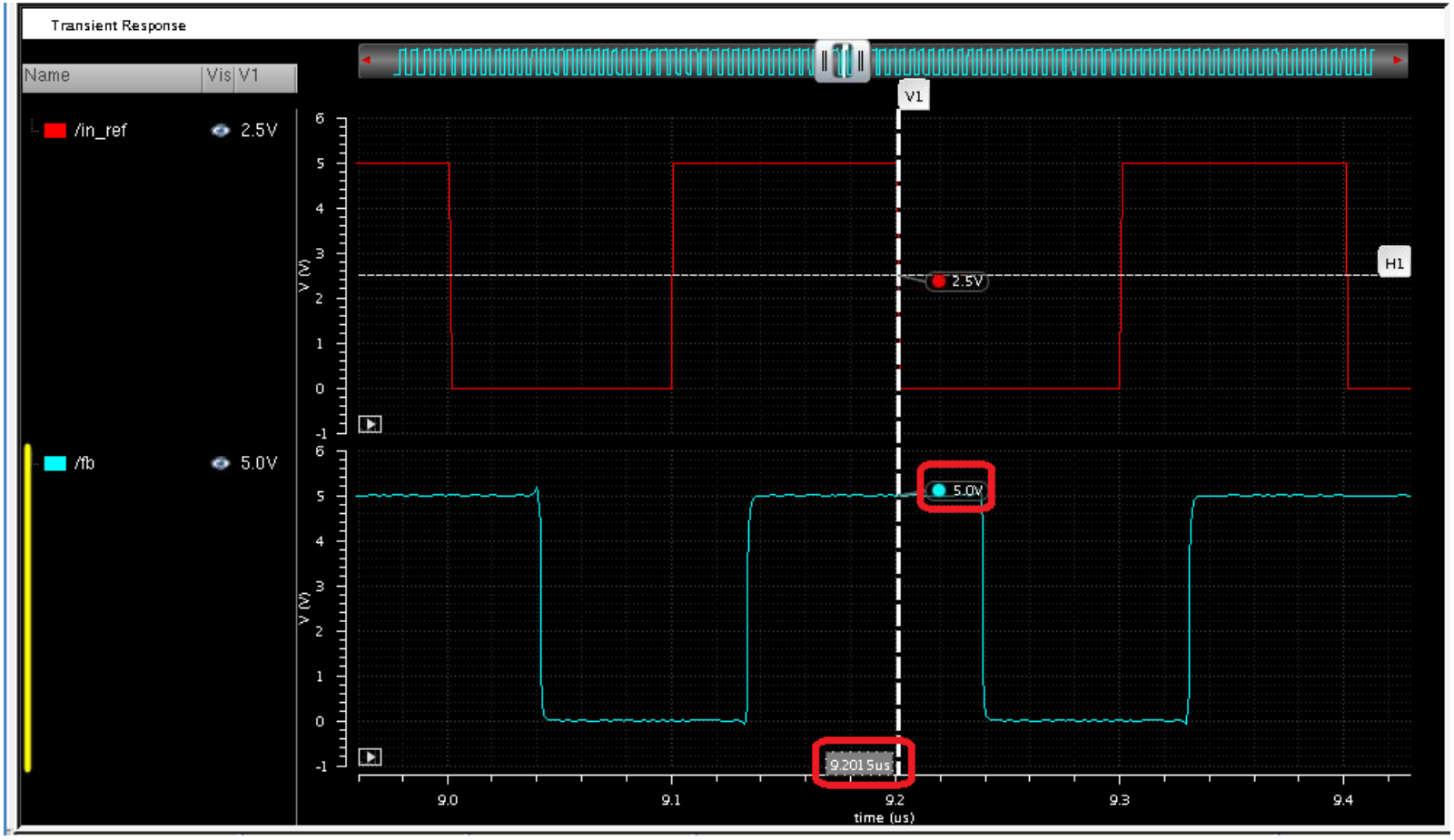
Use bindkey h to add horizontal marker



cadence

Frequently Used Functions
Vertical Marker

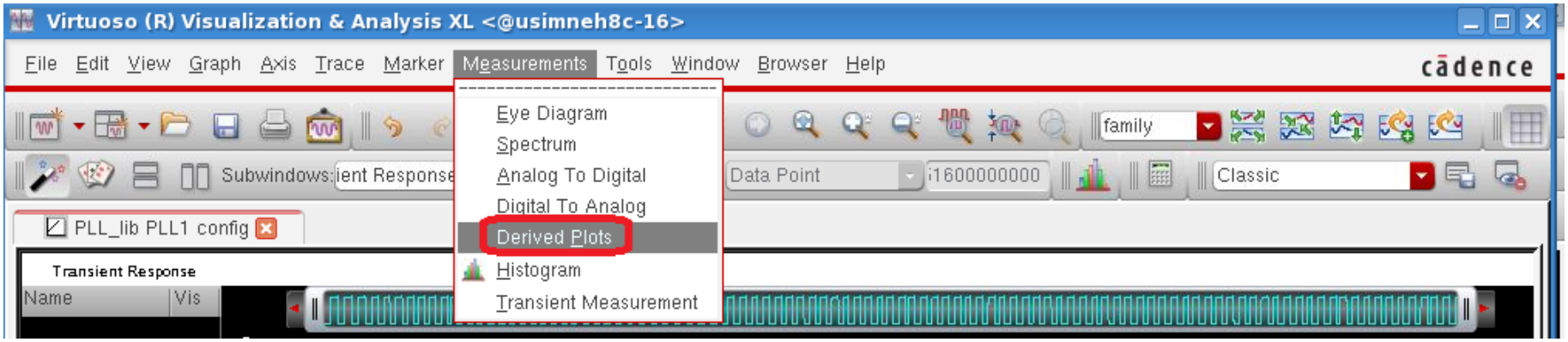
Use bindkey v to add vertical marker



cadence

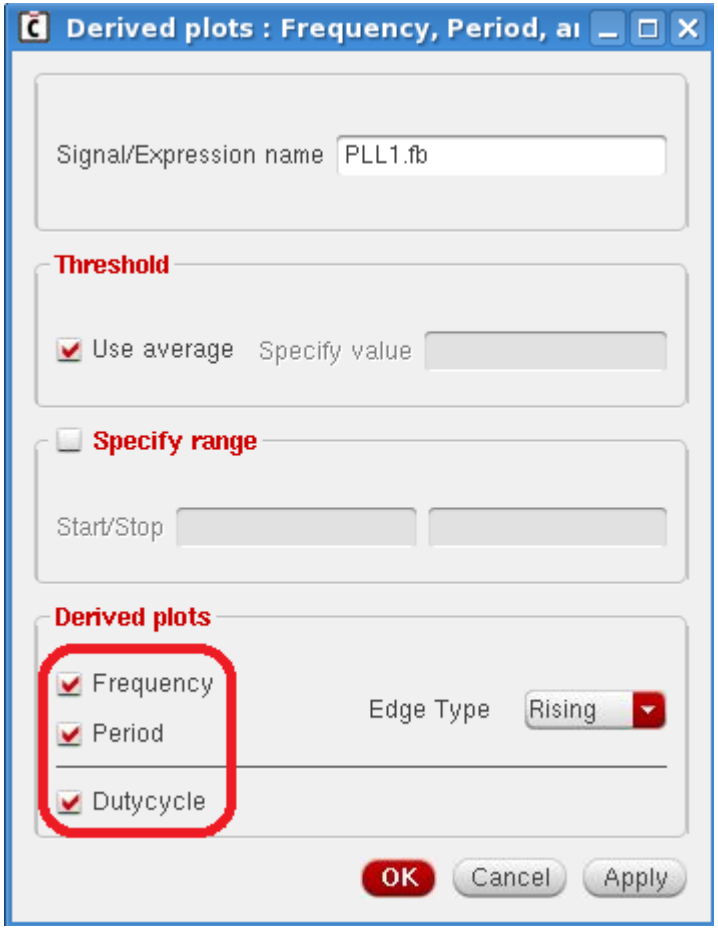
Frequently Used Functions
Frequency/Period/Duty Cycle

1. Select a trace, go to Measurements->Derived Plots



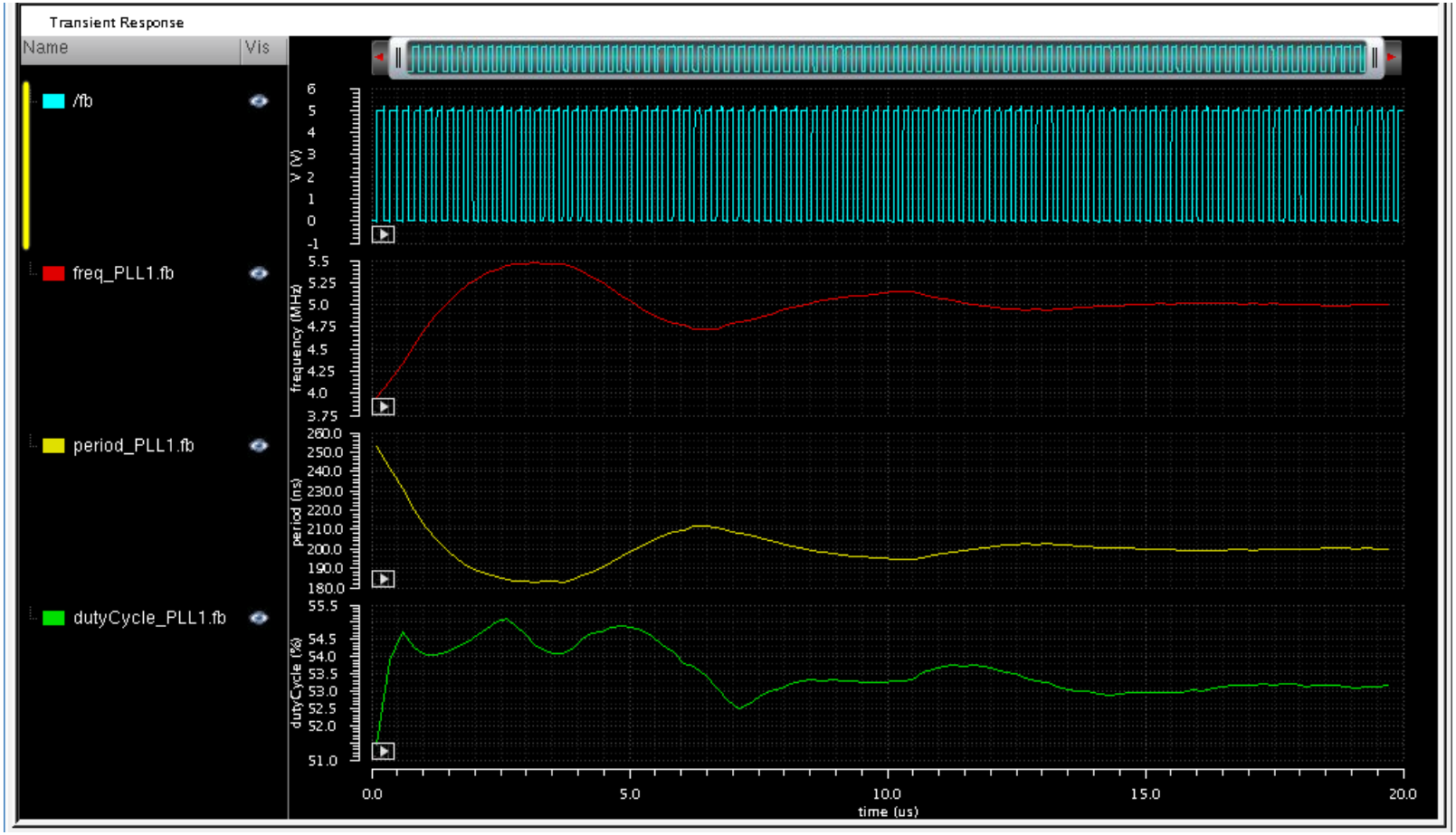
Frequently Used Functions
Frequency/Period/Duty Cycle

2. Can choose plot Frequency/Period/Dutycyle or all of them



Frequently Used Functions
Frequency/Period/Duty Cycle

3. Can get frequency/period/dutycycle vs. time for specified trace then



cadence

Frequently Used Functions

Bus

If multiple digital signals are selected, you can create a bus for them by choosing Trace->Bus->Create. Type a name for the bus, choose a radix and OK the form.

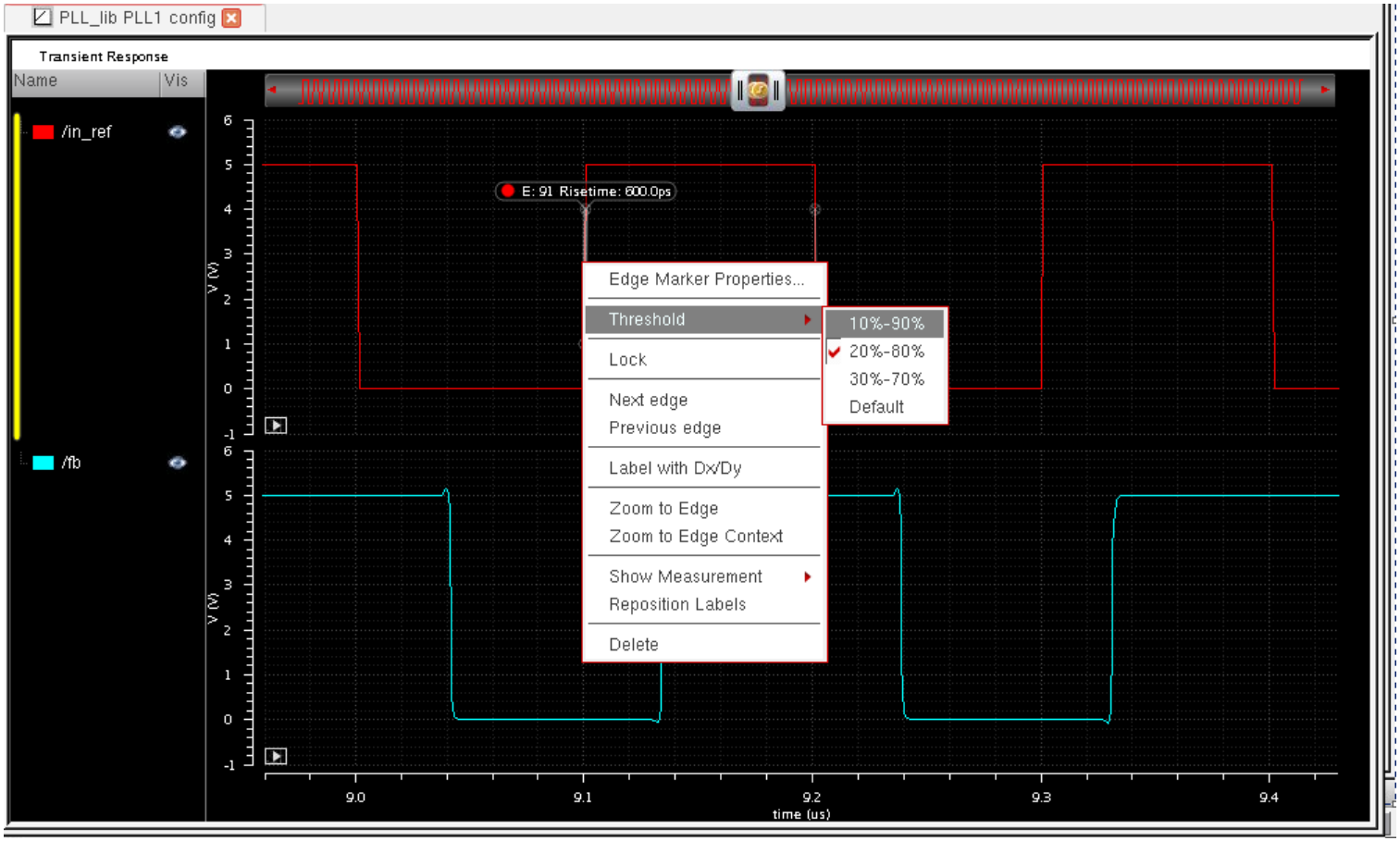
Frequently Used Functions (Edge Measurement feature)
Risetime/Falltime

1. Place mouse pointer on the edge to be measured, and press bindkey t,
or Right-click an edge in the Edge Browser and choose *Create Edge Marker*

if the edge selected is a rising edge, get risetime,
if the edge selected is a falling edge, get falltime.

Frequently Used Functions (Edge Measurement feature)
Risetime/Falltime

2. Select the edge marker added, RMB (right mouse button), choose Threshold, can change threshold of risetime and falltime, default is 20% - 80%

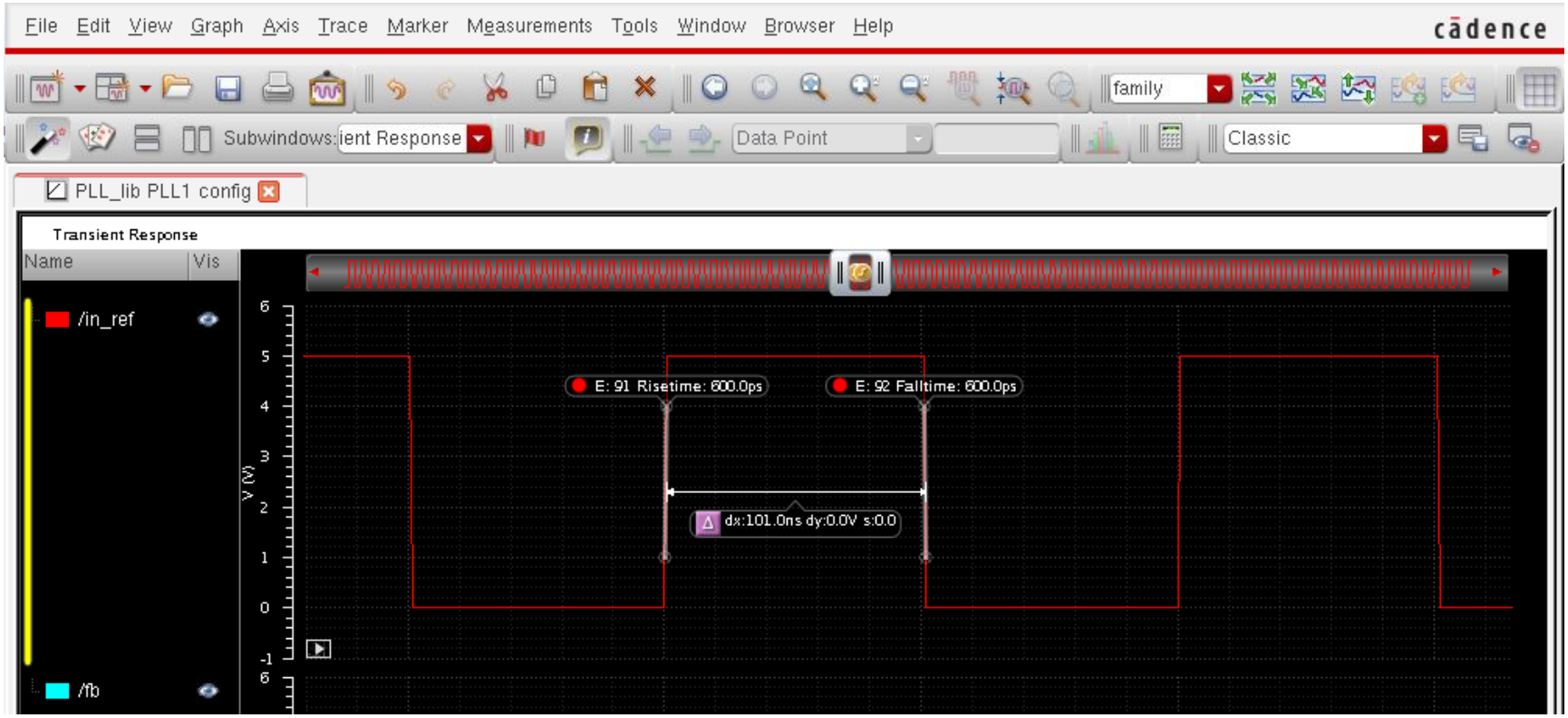


cadence

Frequently Used Functions (Edge Measurement feature)
Delay

Select 2 edge markers added previously,

Use bindkey d, can get delay between these 2 edges



cadence


cadence®