Synopsys, Inc.
690 East Middlefield Road
Mountain View, CA 94043 USA

CCS Timing Library Characterization Guidelines

Version 3.4, October 2016

Abstract

This document describes CCS Timing library requirements and key items to consider during library characterization. The intended audience of this application note is library developers who characterize libraries that will be used with post-layout RC delay calculation in the Synopsys Galaxy Sign-Off Platform.

The information described in this document is valid for Library Compiler version X-2005.09-SP3 and after.

Synopsys, Inc.
690 East Middlefield Road
Mountain View, CA 94043 USA

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys company and certain product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Revision	Comments
1.2	Original CCS Timing release.
2.0	• Updated cover page, Section 2.1, Simulation Autostop and Full Transition, and Section 7, Library Compiler Checks for CCS Timing Libraries, on tolerance of full rail voltage check from 2% to 5% for Library Compiler X-2005.09-SP3. • Updated Introduction section to provide background • Added new Section 3.1, Receiver Model Library Requirements, detailing library transition time index requirements for receiver models. Sections 3.1-3.4 from previous revision incremented by 1. • Updated Section 6.2, Input Characterization Waveforms, on recommendations for input transition time to minimize Miller effect at startup. • Updated Section 6.7, Deriving Capacitances for the Receiver Model, with information on truncating pre-transition data points for the Synopsys script used to derive C1 and C2. Some material moved to Section 3.1.
2.1	• Added new Section 3.2, C1 and C2 Calculation for Receiver Models, describing requirements for calculating C1 and C2 capacitance acquisition. Sections 3.2-3.5 from previous revision incremented by 1. • Added details about C1 and C2 characterization methods to section 6.7, which describes characterization methods to ensure good results for C1 and C2 values.
2.2	• Added guidelines for characterization index value selection • Changed segmentation_tolerance in HSPICE options to 0.005, from 0.01 • Added table of contents
3.1	• Added detailed explanation of driver and receiver algorithms • Added details on implementation of CCS script algorithms
4.0	• Added new section on “CCS Timing Liberty Extension (June 2016)”

Copyright Notice and Proprietary Information **2**
1 Introduction **5**
1.1 THE CCS SOLUTION FOR TODAY AND TOMORROW 5
1.2 CCS TIMING MODEL 6
1.3 CHARACTERIZATION FOR CCS TIMING 6
2 CCS Timing Driver Model Requirements **8**
2.1 SIMULATION AUTOSTOP AND FULL TRANSITION 8
2.2 ZERO CURRENT VALUES 13
2.3 CURRENT POLARITY 14
2.4 PEAK CURRENT 16
2.5 CURRENT VECTOR VARIABLES 17
2.6 NEGATIVE REFERENCE_TIME 18
2.7 IDENTICAL REFERENCE_TIME FOR INPUT TRANSITION TIME WITH VARYING LOADS 19
3 CCS Timing Receiver Model Requirements **20**
3.1 RECEIVER MODEL LIBRARY REQUIREMENTS 21
3.2 C1 AND C2 CALCULATION FOR RECEIVER MODELS 23
3.3 RECEIVER MODELS WITH RISE AND FALL TABLES 24
3.4 RECEIVER MODELS WITH SINGLE-EDGE TABLES 25
3.5 ARC-BASED RECEIVER MODELS WITHOUT LOAD DEPENDENCY FOR A SINGLE EDGE 26
3.6 ARC-BASED OR PIN-BASED RECEIVER MODEL 27
4 Requirements for CCS Timing Voltage and Temperature Scaling Libraries **29**
4.1 STRUCTURAL REQUIREMENTS BETWEEN CCS TIMING LIBRARIES 29
4.2 CHARACTERIZATION LOADS BETWEEN CCS TIMING LIBRARIES 29
4.3 LIBRARY VOLTAGE 29
5 CCS Timing General Library Requirements **30**
5.1 DELAY AND TRANSITION TIME TRIP POINTS 30
5.2 ACCURACY OF LIBRARY 31
5.3 RANGE OF INDICES 31
5.4 NUMBER OF SIGNIFICANT DIGITS 31
6 Library Characterization for Timing Analysis **33**
6.1 CIRCUIT SIMULATOR SETTINGS 33
6.1.1 Control of Simulation Time Step ***34***
6.1.2 Simulation Speed ***34***
6.1.3 Simulation Algorithm ***35***
6.2 INPUT CHARACTERIZATION WAVEFORMS 35
6.3 SYNOPSYS PREDRIVER WAVEFORM ALGORITHM 38
6.4 DELAY AND TRANSITION TIME TRIP POINTS 39
6.5 OPERATING RANGE OF CELLS 39
6.6 SELECTING CHARACTERIZATION POINTS 41
6.6.1 Geometric Spacing and Over-Sampling ***42***
6.6.2 Plotting of Interpolation Errors ***42***
6.7 SEGMENTING THE CURRENT WAVEFORM 42
6.8 DERIVING CAPACITANCES FOR THE RECEIVER MODEL 46
6.9 OTHER CONSIDERATIONS 47
6.9.1 Simultaneous Switching ***47***
6.9.2 Merging Timing Arcs ***48***
6.9.3 Three-Dimensional Timing Arcs ***48***
6.9.4 Level Shifters ***49***
6.9.5 Cells with Unbuffered Inputs ***49***
7 Library Compiler Checks for CCS Timing Libraries **49**
8 CCS Timing Liberty Extension (June 2016) **50**
9 References **51**

1 Introduction

The Composite Current Source (CCS) modeling technology is the first in the Electronic Design Automation industry to deliver a complete open-source current based modeling solution for timing, noise and power. Along with the available parsers, characterization/validation tools, and guidelines, this open-source Liberty™ modeling format enables efficient characterization for cell library creators. For IC designers, the CCS modeling technology enables comprehensive nanometer design analysis and optimization for the first time. Designers can reduce design margins and speed design closure by eliminating iterations.

The widely available Non-linear delay and power models (NLDM/NLPMs) have long been the method used to abstract delay and output transition time characteristics of cells for usage in timing flows. At process geometries of 90-nm and below, many new effects cannot be accurately modeled using this approach. Some of these modeling challenges include:

- High Impedance Interconnect
- Miller effect
- Dynamic IR-drop
- Multi-voltage, and Dynamic Voltage and Frequency Scaling (DVFS)
- Driver weakening
- Temperature inversion
- Increasing variations

To make matters worse, some of these effects are inter-dependent between timing, noise and power. For example timing and slew rates affect power which impacts IR-drop which in turn changes timing. Also signal integrity can impact power which in-turn impacts IR-drop and timing.

1.1 The CCS Solution for today and tomorrow

With the advent of smaller nanometer leading to the issues discussed above, the composite current source (CCS) approach of modeling cell behavior has been developed to address the new effects of very deep submicron processes. The CCS modeling technology consists of a single Open-Source Liberty™ library that covers current-based modeling for timing, noise and power along with all the necessary tools and guidelines for accurate library generation and validation.

CCS models include data specifically targeted at characterizing a cell's timing, noise, and power behavior. Because CCS is current-based, it enables both temperature and voltage scaling of all cell behavior. This scaling capability can

be applied to timing, noise and power and is more sophisticated than conventional interpolation, thus improving accuracy. The ability to scale accurately for voltage and temperature reduces the number of library corners that must be characterized and greatly simplifies low power design (e.g. multi-voltage and DVFS designs) flows in addition to enabling new capabilities such as IR-Drop delay analysis.

The Composite Current Source modeling technology addresses today's existing and emerging design requirements—the physical effects of nanometer designs as well as the needs of design strategies such as multiple-voltage domains. In one model, this open-source Liberty format combines the cell data needed to support timing, noise, and power analyses that are efficient yet accurate because they begin with current values that are characterized for the relevant nanometer dependencies. Because the open CCS format is extensible, these models constitute a foundation that can be enhanced as needed to meet future requirements such as variation-aware or statistical timing analysis.

1.2 CCS Timing Model

Timing modeling with CCS is composed of a driver model and a receiver model. The driver model is a time- and voltage-dependent current source. Because this current source's drive resistance is essentially infinite, the model provides high accuracy even when the drive resistance is much lower than the interconnect impedance.

The CCS receiver model consists of two capacitances C1 and C2, allowing dynamic adjustment of capacitance during the transition. The capacitance values can also be dependent on input slew, output load and state of the cell.

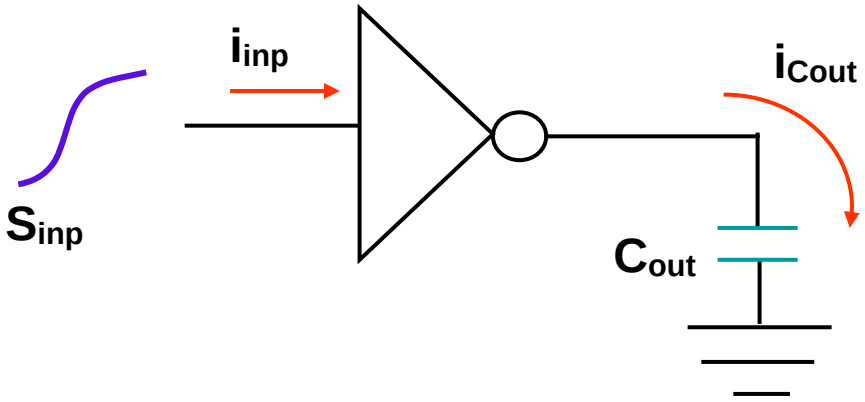
The cell response (output voltage waveform) for a characterized input slew and output load can be easily determined by integrating the current waveform. When the cell is driving a detailed RC network, data from multiple characterized current waveforms (for different output load values) are used to determine the current dynamically during the transition. Sophisticated scaling of the current waveform enables high accuracy calculation between characterized values of input slew, output load, Vdd and Temperature.

Using these capabilities, CCS timing models achieve accuracy within 2% of SPICE. At the same time, the models are simple enough to enable very fast delay calculations for highly complex nanometer designs.

1.3 Characterization for CCS Timing

Characterizing a cell for CCS models is similar to characterizing nonlinear delay models (NLDMs). The overall characterization process for timing, noise, and power is faster, primarily because CCS enables two orders of magnitude (100X) faster noise characterization runtime than NLDM.

For CSS characterization, an input stimulus is chosen to produce a specific input slew time (S_{inp}) ; a load capacitance (C_{out}) is connected to the output pin; and a circuit simulation is run in the same way as for NLDM. But instead of measuring voltage thresholds at the output pin, current is measured through the load capacitor and into the input pin. The current through C_{out} is used for the driver model, and the current into the input pin is used to determine the receiver model.



These characterization experiments are repeated for a table of different S_{inp} and C_{out} combinations. The current through C_{out} is saved for every circuit simulation timestep and then reduced to a much smaller set of current and time (i, t) points. The points are chosen such that $V_{out}(t)$ can be accurately reproduced for every timestep during the transition

The current and voltage at the input pin are saved and then used to determine C1 and C2 values such that gate-level delay calculation can closely match times to the delay threshold and to the second slew threshold at the input pin.
So to summarize, the composite current source (CCS) approach of modeling cell behavior has been developed to address the new effects of very deep submicron processes. Characterization for NLDMs and CCS Timing can be done simultaneously, allowing easy CCS Timing library data extraction from the simulation results. In this application note, we will focus on characterization of the CCS driver and receiver models for accurate post-layout RC delay calculation in the Synopsys Galaxy Sign-Off Platform. Each of the following topics will be discussed:

- CCS Timing library requirements, including driver model, receiver model, scaling, and general library requirements
- Key items to consider in library characterization for meeting CCS Timing requirements, including current waveform segmentation and deriving receiver model capacitances
- CCS Timing library screening checks that are in X-2005.09 Library Compiler

The intended audience of this application note is library developers who characterize libraries that will be used with post-layout RC delay calculation in the Synopsys Galaxy Sign-Off Platform. It is recommended that the reader be familiar with the following CCS Timing collateral:

- CCS Timing White Paper
- CCS Timing Liberty Syntax
- CCS Timing Library Validation Guidelines
- CCS Timing Library Correlation

2 CCS Timing Driver Model Requirements

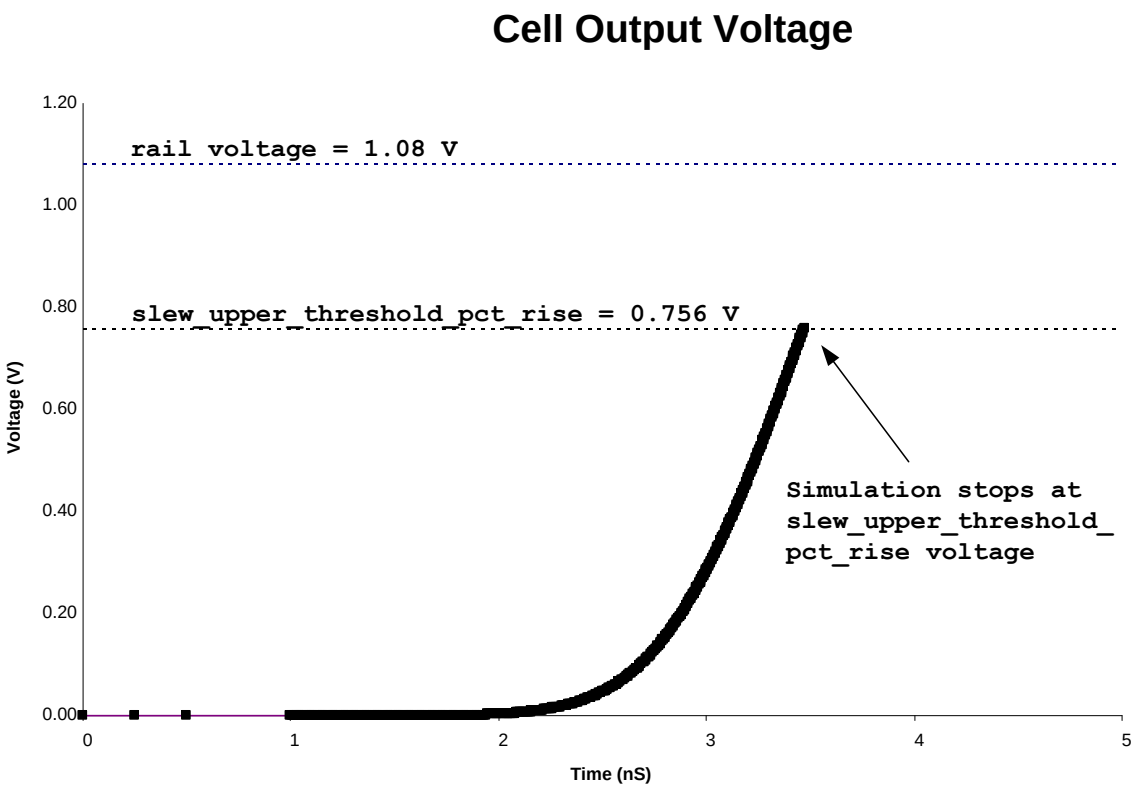
In this section, we will describe the CCS Timing driver model requirements for current vectors in the library. For all cases, Library Compiler X-2005.09 screens for these conditions and issues an error, warning, or informational message.

2.1 Simulation Autostop and Full Transition

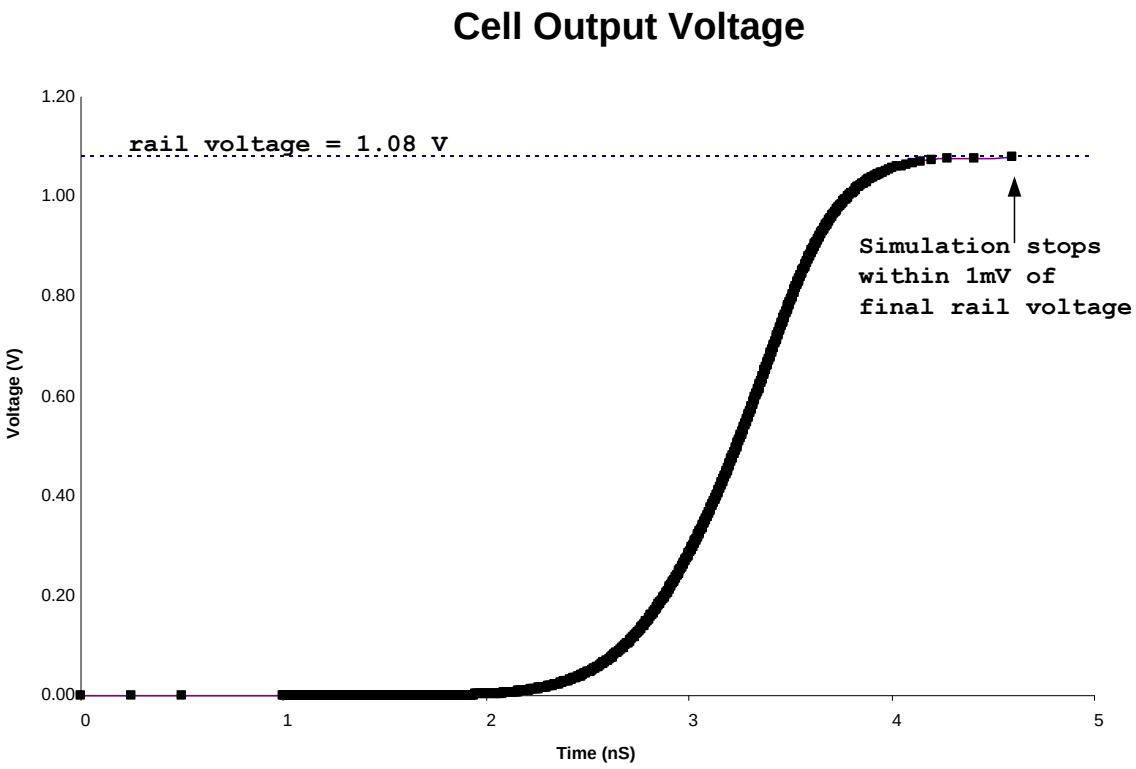
For CCS Timing, the recommended characterization autostop value is when the output pin voltage is within 1mV of rail. Speeding characterization by using setting simulation autostop to the second transition time threshold will cut off the current measurement before it is completed. Figure 1 shows simulation results for the output voltage of a cell that a) has an autostop value at the second transition time threshold (70% of 1.08V or 0.756V) and b) has an autostop value within 1mV of the final rail voltage of 1.08 V.

Figure 1: Simulation Result with Autostop Feature

a) BAD: Autostop set to slew_upper_threshold_pct_rise value of 70% of Vdd


Cell Output Voltage
rail voltage = 1.08 V
slew_upper_threshold_pct_rise = 0.756 V
Simulation stops at slew_upper_threshold_pct_rise voltage
Voltage (V)
Time (nS)
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0
1
2
3
4
5

b) GOOD: Autostop set to Vdd – 1mV



To sanity check current vector data, the current can be integrated using the characterization capacitance to obtain the voltage. The voltage should reach within 5% of the final rail voltage. Specifically, the formula below should be used to integrate the current.

$$Vout2 = Vout1 + \left(\frac{0.5}{Cchar}\right) * (Iout2 + Iout1) * (Time2 - Time1)$$

The final voltage should meet the criteria below.

For a rising transition, $abs(Vout - Vdd) <$ Vmargin

For a falling transition, $abs(Vout - Vss) <$ Vmargin

where Vmargin = $0.05 * (Vdd - Vss)$

Beginning with the X-2005.09-SP3 version of Library Compiler, the check for final rail voltage will be changed from 2% to 5%. See Example 1.

Example 1: Library Compiler Warning Message for Final Rail Voltage

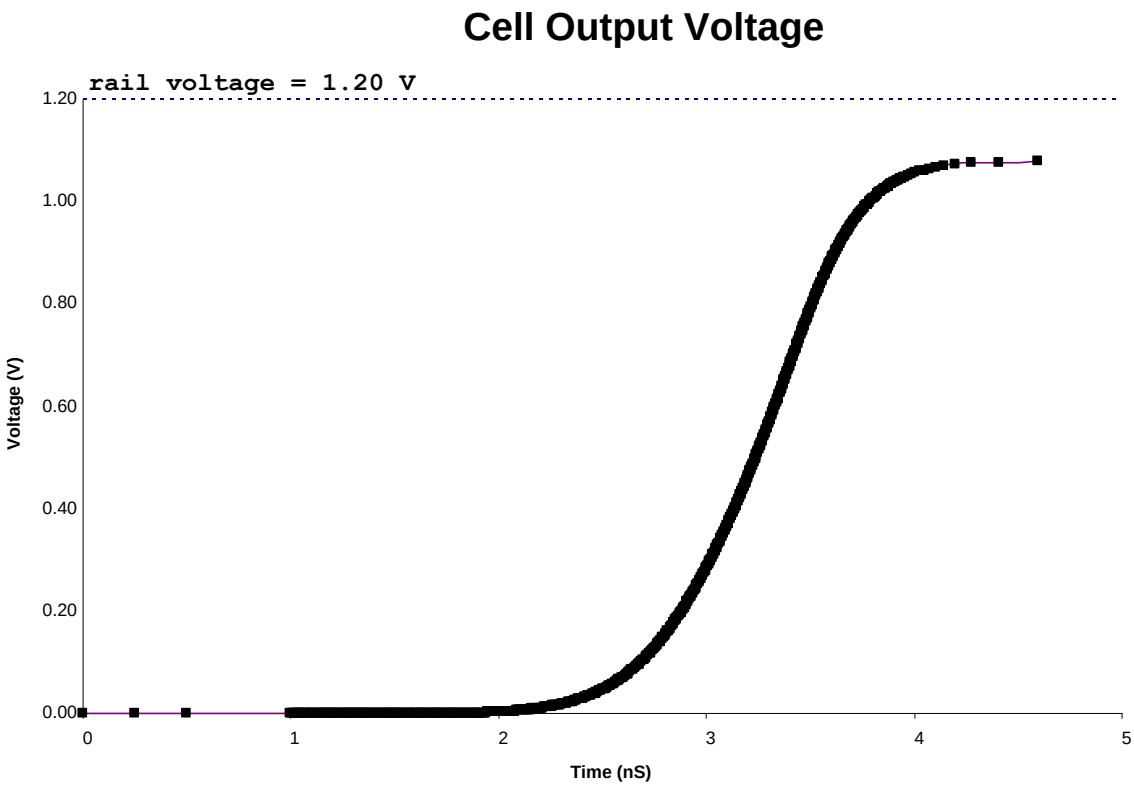
```
Warning: Line 358, The final signal voltage of the vector
is 1.08329, which does not reach within the 5 percent of
the rail voltage 1.02. (LBDB-668)
```

In some cases, the cell output may not be able to achieve within 5% of the final rail voltage. As a minimal requirement, the cell should reach the voltage specified by the second transition time threshold. Cells that do not meet this criteria should be reviewed by the library developer. The X-2005.09 version of Library Compiler will issue an error message for this condition. See Example 2. Figure 2 shows a cell that does not reach full rail voltage on its output.

Example 2: Library Compiler Error Message for Second Transition Time Threshold

```
Error: Line 358, The final signal voltage of the vector
is 0.700329, which does not reach beyond the 2nd slew
threshold voltage, which is 0.714000. (LBDB-669)
```

Figure 2: Cell Output Voltage Does Not Reach Full Rail



Since the voltage waveform should at least cross the voltage specified by the second transition time threshold, current vectors with zero current should not be included in the library. See Example 3.

Example 3: Current Vector with Zero Current

```
vector ( CCS_fall ) {
      reference_time : 2.172803e+00 ;
      index_1 ( "0.879602" ) ;
      index_2 ( "0.102702" ) ;
      index_3 ( "1.795857e+00, 2.147012e+00,
2.411707e+00, 2.641051e+00, 3.069397e+00, 3.223209e+00,
3.320412e+00, 3.351926e+00, 3.403433e+00, 3.494074e+00,
3.662858e+00, 3.763292e+00, 3.899033e+00" ) ;
      values ( "0.000000e+00, 0.000000e+00, 0.000000e+00,
0.000000e+00, 0.000000e+00, 0.000000e+00, 0.000000e+00,
```

```
0.000000e+00, 0.000000e+00, 0.000000e+00, 0.000000e+00,
0.000000e+00, 0.000000e+00" ) ;
```

2.2 Zero Current Values

The current vector should not contain zero current values. PrimeTime assumes that the initial current is zero before the first time value in the current vector and the final current is assumed to be the last value stored in the current vector (a small value). Example 4 shows a current vector that stores an initial and final current value of zero. Example 5 shows a current vector that contains a zero value as one of its current values.

Example 4: Current Vector with Initial and Final Zero Current

```
vector ( CCS_fall ) {
      reference_time : 2.172803e+00 ;
      index_1 ( "0.879602" ) ;
      index_2 ( "0.102702" ) ;
      index_3 ( "1.795857e+00, 2.147012e+00,
2.411707e+00, 2.641051e+00, 3.069397e+00, 3.223209e+00,
3.320412e+00, 3.351926e+00, 3.403433e+00, 3.494074e+00,
3.662858e+00, 3.763292e+00, 3.899033e+00" ) ;
      values ( "0.000000e+00, -5.749690e-03, -1.797790e-
02, -4.025600e-02, -1.030730e-01, -1.222390e-01, -
1.298390e-01, -1.305350e-01, -1.280700e-01, -1.078560e-
01, -4.595000e-02, -2.170950e-02, 0.000000e+00" ) ;
```

Example 5: Current Vector with Zero Current

```
vector ( CCS_fall ) {
      reference_time : 2.172803e+00 ;
      index_1 ( "0.879602" ) ;
      index_2 ( "0.102702" ) ;
      index_3 ( "1.795857e+00, 2.147012e+00,
2.411707e+00, 2.641051e+00, 3.069397e+00, 3.223209e+00,
3.320412e+00, 3.351926e+00, 3.403433e+00, 3.494074e+00,
3.662858e+00, 3.763292e+00, 3.899033e+00" ) ;
```

```
      values ( "-6.393840e-03, -5.749690e-03, -1.797790e-
02, -4.025600e-02, -1.030730e-01, -1.222390e-01, -
1.298390e-01, -1.305350e-01, -1.280700e-01, -1.078560e-
01, -4.595000e-02, 0.000000e+00, -2.170950e-02" ) ;
```

Library Compiler X-2005.09 will issue an error message if there is a zero value in the current vector. See Example 6.

Example 6: Library Compiler Error Message for Zero Value in Current Vector

```
Error: Line 381, The 'values' attribute of the 'vector'
has a value '0.000000', which is equal to '0.000000'.
(LBDB-664)
```

2.3 Current Polarity

For the `output_current_rise` group, the current values must be positive in the `values` section of the current vector. Similarly, for the `output_current_fall` group, the current values must be negative.

The PMOS devices in a cell with rising output will cause the output current of the cell to be positive. Similarly, NMOS devices sink current and will cause the output current of a cell to be negative. In Library Compiler X-2005.09, an error will be issued for current vectors of this type and the library will not compile. See Example 7.

Example 7: Library Compiler Error Message for Mixed Polarities in Current Vector

```
Error: Line 75, The 'values' attribute  of the 'vector'
has a value '-0.196476', which should be positive for the
output_current_rise vector. (LBDB-664)
```

Example 8 shows that the second current vector in the `output_current_rise` section has negative current values, which does not meet the criteria for current polarity.

Example 8: Incorrect Current Polarity in Current Vector

```
output_current_rise () {
  vector ( CCS_rise ) {
      reference_time : 1.018468e+00 ;
      index_1 ( "0.0100" ) ;
      index_2 ( "0.104" ) ;
      index_3 ( "1.026000e+00, 1.030780e+00,
1.030994e+00, 1.034013e+00, 1.213171e+00, 1.316816e+00,
1.447324e+00, 1.584940e+00, 1.715004e+00, 1.868917e+00,
2.044892e+00, 2.377380e+00" ) ;
      values ( "1.277399e-01, 2.053170e-01, 2.381160e-01,
2.377790e-01, 2.142830e-01, 1.894730e-01, 1.310250e-01,
7.119820e-02, 3.582050e-02, 1.474390e-02, 5.060350e-03,
7.094180e-04" ) ;
    }
  vector ( CCS_rise ) {
      reference_time : 1.018468e+00 ;
      index_1 ( "0.0100" ) ;
      index_2 ( "0.465" ) ;
      index_3 ( "1.026178e+00, 1.030994e+00,
1.035269e+00, 1.800915e+00, 2.275368e+00, 2.837299e+00,
3.435500e+00, 4.019846e+00, 4.659704e+00, 5.573401e+00,
6.505548e+00" ) ;
      values ( "1.341709e-01, 2.441400e-01, 2.440120e-01,
2.213040e-01, -1.964760e-01, -1.396400e-01, 7.716020e-02,
```

```
3.794630e-02, 1.600170e-02, 5.013170e-03, 1.570870e-03" )
;
    }
```

2.4 Peak Current

The first current value in the current vector should not be the peak of the current waveform, nor should the last value in the current vector be the peak of the current waveform. For rising outputs, the peak value is the most positive value in the current vector, and for falling outputs, the peak value is the most negative value in the current vector. Possible causes for the peak current as the first point in the current vector are characterization transition times that are less than the minimum transition time for the process (see Section 6.4) or simulation issues, such as large simulation time steps or accuracy settings. Library Compiler X-2005.09 will issue an error message for this condition. See Example 9. Example 10 and Figure 3 illustrate a current vector with these characteristics.

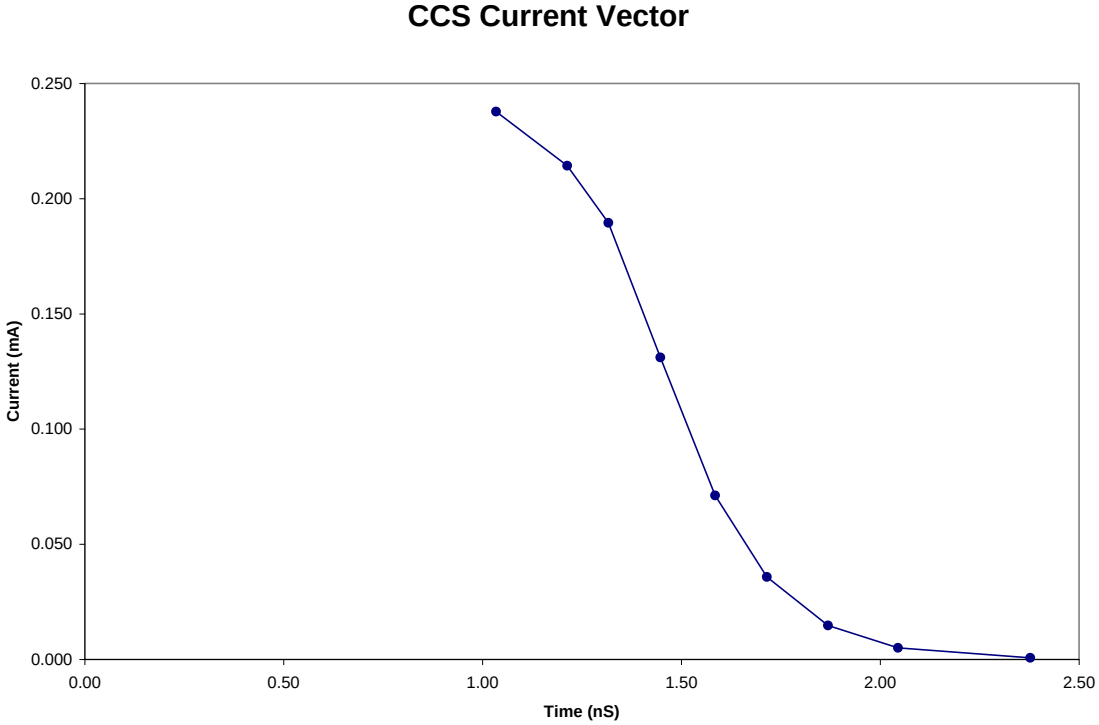
Example 9: Library Compiler Error Message for Peak Current as Initial Point in Current Vector

```
Error: Line 95, The first value of the 'values' attribute
of the 'vector' is the peak value. (LBDB-665)
```

Example 10: Current Vector with Peak Current as Initial Point

```
output_current_rise () {
  vector ("CCS_rise") {
   reference_time : 4.015000e+00;
   index_1("0.024000");
   index_2("0.07900");
index_3("4.760000e+00,4.760459e+00,4.801981e+00,5.302611e
+00,5.818013e+00,6.289269e+00,6.992352e+00,7.785586e+00")
;
values("0.963412e+00,0.062298e+00,0.060605e+00,0.051152e+
00,0.030261e+00,0.011958e+00,0.002145e+00,0.000454e+00");
  }
```

Figure 3: Peak Current as the First Point in a Current Vector



2.5 Current Vector Variables

The template specifying composite current source driver models must always have `time` as the value for `variable_3`. The `input_net_transition` and `total_output_net_capacitance` values can be assigned to either `variable_1` or `variable_2`. See Example 11. If `variable_3` does not have the value `time`, Library Compiler X-2005.09 will issue an error message for this condition. See Example 12.

Example 11: Correct CCS Driver Model Templates

```
output_current_template ( CCS_rise ) {
  variable_1 : input_net_transition ;
  variable_2 : total_output_net_capacitance ;
  variable_3 : time ;

output_current_template ( CCS_fall ) {
```

```
variable_1 : total_output_net_capacitance ;
variable_2 : input_net_transition ;
variable_3 : time ;
```

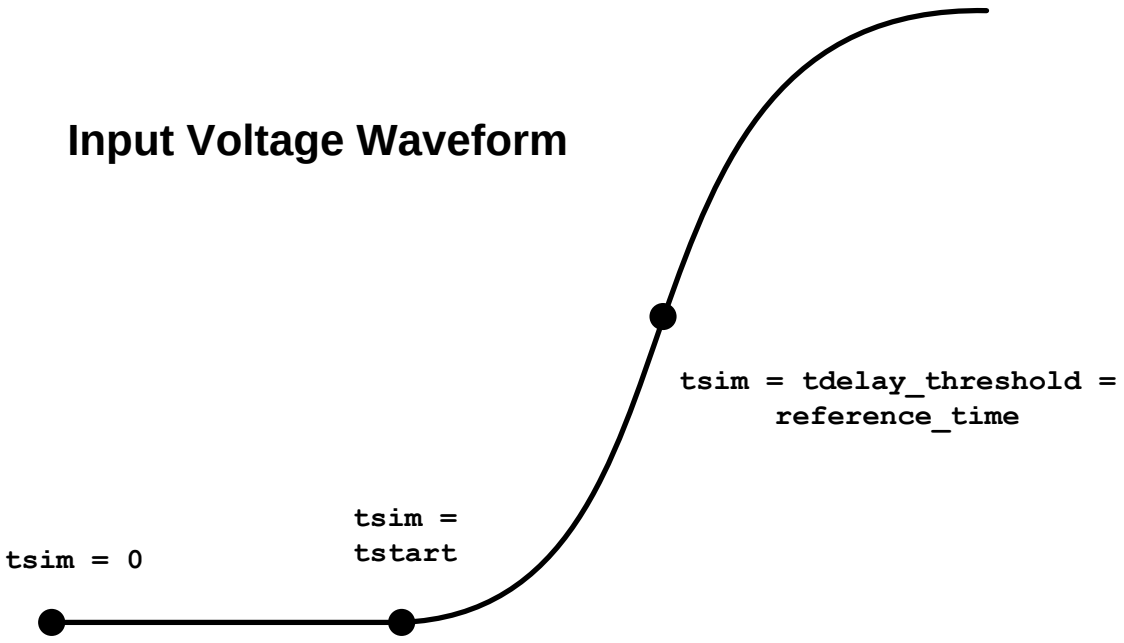
Example 12: Library Compiler Error Message for `variable_3` with `input_net_transition` Value

```
Error: Line 198, The index of 'input_net_transition' can
define only one value. (LBDB-653)
```

2.6 Negative `reference_time`

The `reference_time` attribute is the time at which the input voltage characterization waveform crosses the input delay threshold. The start time for a CCS Timing characterization is the time at which the voltage transition on the input to the cell starts. This precedes the `reference_time`. The `reference_time` is used to calculate the cell delay in CCS delay calculation. See Figure 4 for an example of reference_time.

Figure 4: `reference_time` Definition



Since `reference_time` is related to physical circuit behavior, Library Compiler X-2005.09 will issue an error if a negative `reference_time` is used in the library. See Example 13.

Example 13: Library Compiler Error Message for Negative `reference_time`

```
Error : Line 491, The 'reference_time' of the 'vector' is
-0.419265, which should be positive. (LBDB-667)
```

2.7 Identical `reference_time` for Input Transition Time with Varying Loads

The `reference_time` attribute must be identical for the same input transition time and edge (either rise or fall) across all capacitive loads. This ensures that the input characterization waveform is consistent across all characterization loads. Library Compiler will flag this as a compilation error if this condition is not met. Example 14 shows that for an input transition time of 20pS, the `reference_time` for a load of 0.104pF is 1.018468nS and is 1.018469nS for a load of 0.465pF. Example 15 shows the error message in Library Compiler for this condition.

Example 14: Inconsistent `reference_time`s for Varying Loads with Identical Input Transition Time

```
output_current_rise () {
  vector ( CCS_rise ) {
      reference_time : 1.018468e+00 ;
      index_1 ( "0.0200" ) ;
      index_2 ( "0.104" ) ;
      index_3 ( "1.026000e+00, 1.030780e+00,
1.030994e+00, 1.034013e+00, 1.213171e+00, 1.316816e+00,
1.447324e+00, 1.584940e+00, 1.715004e+00, 1.868917e+00,
2.044892e+00, 2.377380e+00" ) ;
      values ( "1.277399e-01, 2.053170e-01, 2.381160e-01,
2.377790e-01, 2.142830e-01, 1.894730e-01, 1.310250e-01,
7.119820e-02, 3.582050e-02, 1.474390e-02, 5.060350e-03,
7.094180e-04" ) ;
    }
  vector ( CCS_rise ) {
      reference_time : 1.018469e+00 ;
      index_1 ( "0.0200" ) ;
      index_2 ( "0.465" ) ;
      index_3 ( "1.026178e+00, 1.030994e+00,
1.035269e+00, 1.800915e+00, 2.275368e+00, 2.837299e+00,
3.435500e+00, 4.019846e+00, 4.659704e+00, 5.573401e+00,
6.505548e+00" ) ;
      values ( "1.341709e-01, 2.441400e-01, 2.440120e-01,
2.213040e-01, 1.964760e-01, 1.396400e-01, 7.716020e-02,
3.794630e-02, 1.600170e-02, 5.013170e-03, 1.570870e-03" )
;
    }
```

Example 15: Inconsistent `reference_time` Error Message in Library Compiler

```
Error: Line 115, The values of 'reference_time' and
'input_net_transition'are not consistent in group
'output_current_rise'. (LBDB-654)
```

3 CCS Timing Receiver Model Requirements

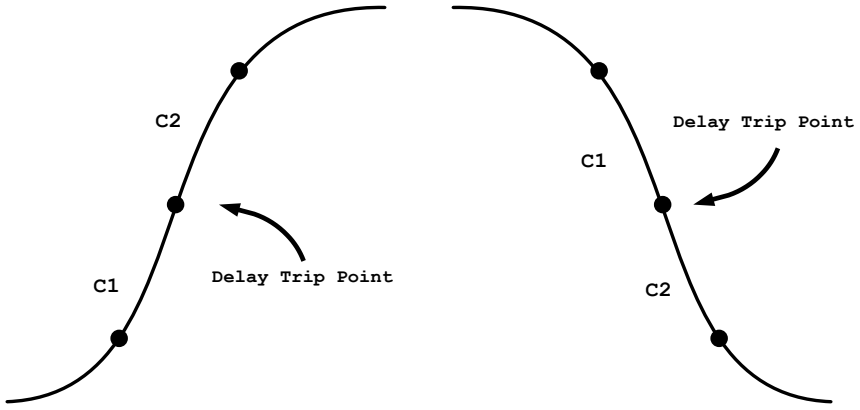
There are multiple ways to create composite current source receiver models, for example, pin-based and arc-based. It is recommended that a full receiver model with rise and fall information should always be generated. The following sections will discuss the requirements for receiver model library tables as well as receiver model characterization considerations. These are discussed

because characterizing the full receiver model for some cells may be problematic. For these situations, additional syntax for the receiver model is supported. The method chosen can impact the quality of results.

3.1 Receiver Model Library Requirements

The CCS Timing receiver model uses two capacitance values to model the variation of cell input capacitance during the input signal transition. For Primetime delay calculation, the C1 value is used before the delay trip point has been reached in the transition and the C2 value is used after the delay trip point has been reached. This is true for both rise and fall transitions. See Figure 5.

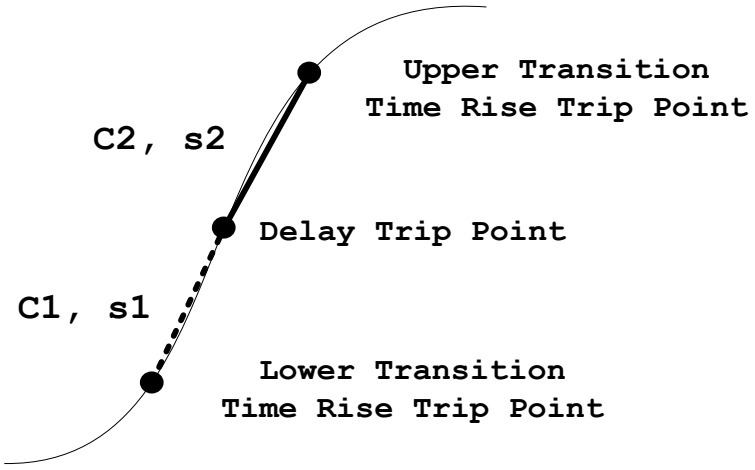
Figure 5: Receiver Model Capacitances



The `receiver_capacitance*_rise` and `receiver_capacitance*_fall` library syntax always follows the transition occurring on the input of the cell. The C1 capacitance value is recommended to be chosen such that the current produces a voltage that matches the value at the input delay trip point of the cell. Similarly, the C2 capacitance value is recommended to be chosen such that the current produces a voltage that matches the second slew trip point of the cell.

Since the C1 and C2 values are dependent on a portion of the input waveform that is smaller than that covered by the library transition time, the receiver model tables should reflect this. Receiver model transition time indexes should use the partial transition time of the input waveform and extrapolate these values to the range of the library transition time. Figure 6 shows an input characterization waveform that has different input transition time values for C1 and C2.

Figure 6: Characterization Waveform with Different Transition Time Values for C1 and C2



To derive the final library transition time index for the C1 and C2 tables, do the following.

1. Add measure statements to the characterization simulation deck to measure `s1` and `s2`, the transition times corresponding to the part of the waveform used for C1 and C2. See Examples 16 and 17.

Example 16: Measuring s1 and s2 for Rising Input Waveform

```
.meas tran s1r trig v(in) val='slew_lower_thresh_voltage_rise' rise=1
             targ v(in) val='input_delay_thresh_voltage_rise'  rise=1
.meas tran s2r trig v(in) val='input_delay_thresh_voltage_rise' rise=1
               targ v(in) val='slew_upper_thresh_voltage_rise'  rise=1
```

Example 17: Measuring s1 and s2 for Falling Input Waveform

```
.meas tran s1f trig v(in) val='slew_upper_thresh_voltage_fall' fall=1
             targ v(in) val='input_delay_thresh_voltage_fall'  fall=1
.meas tran s2f trig v(in) val='input_delay_thresh_voltage_fall' fall=1
               targ v(in) val='slew_lower_thresh_voltage_fall'  fall=1
```

2. Extrapolate the measured s1 and s2 values to the full library transition time range using the formulas below. This ensures that all receiver model transition time indexes are across the defined library transition time trip points.

`receiver_capacitance1_rise` transition time index =

$$s1r * \frac{(slew\_upper\_threshold\_pct\_rise - slew\_lower\_threshold\_pct\_rise)}{(lib\_slew\_derate) * (input\_threshold\_pct\_rise - slew\_lower\_threshold\_pct\_rise)}$$

`receiver_capacitance2_rise` transition time index =

$$s2r * \frac{(slew\_upper\_threshold\_pct\_rise - slew\_lower\_threshold\_pct\_rise)}{(lib\_slew\_derate) * (slew\_upper\_threshold\_pct\_rise - input\_threshold\_pct\_rise)}$$

`receiver_capacitance1_fall` transition time index =

$$s1f * \frac{(slew\_upper\_threshold\_pct\_fall - slew\_lower\_threshold\_pct\_fall)}{(lib\_slew\_derate) * (slew\_upper\_threshold\_pct\_fall - input\_threshold\_pct\_fall)}$$

`receiver_capacitance2_fall` transition time index =

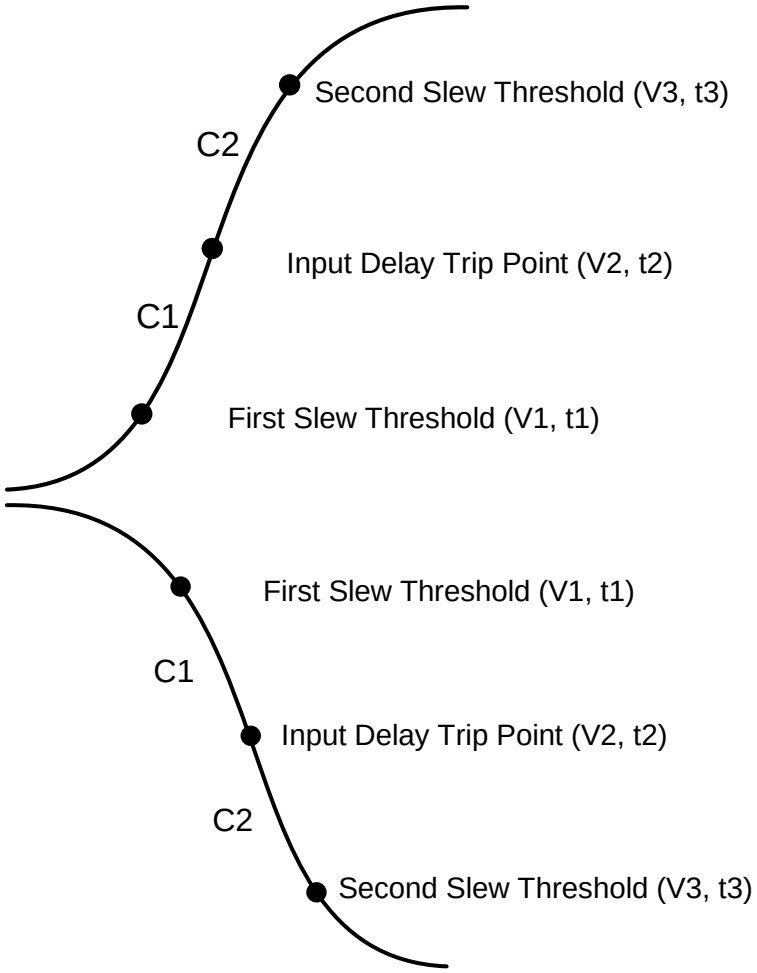
$$s2f * \frac{(slew\_upper\_threshold\_pct\_fall - slew\_lower\_threshold\_pct\_fall)}{(lib\_slew\_derate) * (input\_threshold\_pct\_fall - slew\_lower\_threshold\_pct\_fall)}$$

Note that if ramps are used for input characterization waveforms, the receiver model transition time indexes will be identical to the driver model transition time indexes.

3.2 C1 and C2 Calculation for Receiver Models

For optimal CCS receiver accuracy, The value of C_1 in Figure 7 must be chosen to ensure that the point (V_2,t_2) is exactly matched. The value of C_2 must be chosen to ensure that the point (V_3,t_3) is exactly matched. The recommended characterization methods for C1 and C2 are described in Section 6.7. The Synopsys CCS characterization “jumpstart” kit characterization methods are detailed there.

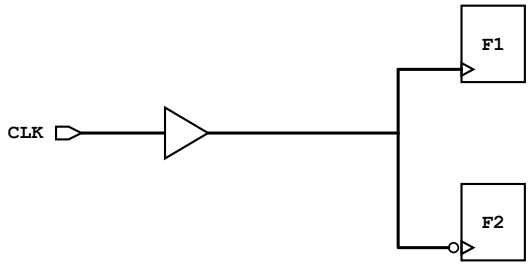
Figure 7: Input Waveforms for rising and falling slews



3.3 Receiver Models with Rise and Fall Tables

Cells with library attributes of `timing_type : rising_edge` or `timing_type : falling_edge` (rising- or falling-edge triggered flip-flops) are recommended to have receiver models with both `receiver_capacitance*_rise` and `receiver_capacitance*_fall` information. This is typically the CLK pin of a register cell. The CLK pin of a register can be a side load on a clock network with rising and falling edge triggered flip-flops, see Figure 8 below.

Figure 8: Rising- and Falling-Edge DFFs on a Clock Network



If a cell is a side load on a net and is missing the rise or fall capacitance tables, PrimeTime will use the NLDM lumped pin capacitance instead of the CCS Timing receiver model. This is why you should provide both rise and fall receiver model tables.

The same consideration should be given to half unate arcs such as `combinational_rise` and `combinational_fall` arcs, as well as `preset` and `clear` timing arcs.

3.4 Receiver Models with Single-Edge Tables

In some cases, a net may only have one transition of interest, or the receiver model edge for when the cell is a side load may not be an important design consideration. For example, all sequential cells in a design may have positive unate clear arcs. In this case, receiver model tables for rise will not be of interest. For half unate timing arcs, receiver models of this type will be compiled and used in PrimeTime. If a pin has missing receiver model information for an edge, the NLDM lumped pin capacitance will used instead. Library Compiler will issue a warning for single edge receiver models of half unate timing arcs. See Example 18. A receiver model with rise and fall information is required for full unate timing arcs. Library Compiler will issue an error message if an edge is missing. See Example 19.

Example 18: Library Compiler Warning Message for Single Edge Receiver Model of Half Unate Timing Arc

```
Warning: Line 923, The timing arc does not have full
receiver modeling information. (LBDB-673)
```

Example 19: Library Compiler Error Message for Single Edge Receiver Model of Full Unate Timing Arc

```
Error: Line 148, The 'receiver_capacitance1_rise' is
missing for this timing arc. (LBDB-267)
```

3.5 Arc-Based Receiver Models Without Load Dependency for a Single Edge

Some cells may have load dependency for one edge of the receiver model, but not the other. For example, the fall receiver model for the CLK pin of a rising edge flip-flop will have little load dependency. In this case, one capacitance index can be used in the arc-based receiver model. See Example 20.

Example 20: Arc-Based Receiver Model without Fall Load Dependency

```
receiver_capacitance1_rise ( rcvr_rise_1 ) {
    values ( \
      "1.947000e-03, 1.959000e-03, 1.988000e-03,
2.016000e-03, 2.034000e-03",\
      "1.960000e-03, 1.969000e-03, 1.994000e-03,
2.020000e-03, 2.037000e-03",\
      "1.992000e-03, 1.997000e-03, 2.012000e-03,
2.030000e-03, 2.043000e-03",\
      "2.031000e-03, 2.032000e-03, 2.035000e-03,
2.043000e-03, 2.051000e-03",\
      "2.185000e-03, 2.184000e-03, 2.179000e-03,
2.171000e-03, 2.161000e-03" \
    ) ;
}
  receiver_capacitance2_rise ( rcvr_rise_2 ) {
    values ( \
      "2.251000e-03, 2.223000e-03, 2.169000e-03,
2.129000e-03, 2.107000e-03",\
      "2.287000e-03, 2.255000e-03, 2.190000e-03,
2.141000e-03, 2.114000e-03",\
      "2.381000e-03, 2.340000e-03, 2.253000e-03,
2.182000e-03, 2.138000e-03",\
      "2.536000e-03, 2.474000e-03, 2.349000e-03,
2.247000e-03, 2.182000e-03",\
      "2.782000e-03, 2.682000e-03, 2.488000e-03,
2.337000e-03, 2.243000e-03" \
    ) ;
receiver_capacitance1_fall ( rcvr_fall_1 ) {
index_1 ( "0.065, 0.110, 0.400, 0.900, 1.50” ) ;
index_2 ( "0.000" ) ;
    values ( \
      "1.882000e-03”,\
      “1.887000e-03”,\
       “1.901000e-03”,\
```

```
        "1.916000e-03",\
        "1.927000e-03" \
    ) ;
}
receiver_capacitance2_fall ( rcvr_fall_2 ) {
index_1 ( "0.065, 0.110, 0.400, 0.900, 1.50" ) ;
index_2 ( "0.000" ) ;
    values ( \
      "2.189000e-03",\
      "2.169000e-03",\
      "2.129000e-03",\
      "2.100000e-03",\
      "2.084000e-03" \
    ) ;
}
```

3.6 Arc-Based or Pin-Based Receiver Model

If both rise and fall tables of an arc-based receiver model show little load dependency, a pin-based receiver model should be used. This type of behavior would be expected for multistage cells and the CLK pin of sequential cells. See Examples 21 and 22 for an arc-based receiver model that has been converted to a pin-based receiver model.

Example 21: Arc-Based Receiver Model with No Load Dependency

```
lu_table_template ( rcvr_rise_1 ) {
  variable_1 : input_net_transition ;
  variable_2 : total_output_net_capacitance ;
  index_1 ( "0.065, 0.110, 0.400, 0.900, 1.50" ) ;
  index_2 ( "0.002, 0.010, 0.100, 0.500, 0.700" ) ;
}
pin ( Z ) {
timing (  ) {
     related_pin  : "A" ;
     timing_sense : negative_unate ;


receiver_capacitance1_rise ( rcvr_rise_1 ) {
```

```
    values ( \
      "1.947000e-03, 1.947000e-03, 1.947000e-03,
1.947000e-03, 1.947000e-03",\
      "1.960000e-03, 1.960000e-03, 1.960000e-03,
1.960000e-03, 1.960000e-03",\
      "1.992000e-03, 1.992000e-03, 1.992000e-03,
1.992000e-03, 1.992000e-03",\
      "2.031000e-03, 2.031000e-03, 2.031000e-03,
2.031000e-03, 2.031000e-03",\
      "2.185000e-03, 2.184000e-03, 2.184000e-03,
2.185000e-03, 2.185000e-03" \
    ) ;
} /* end of arc-based receiver_capacitance1_rise */
```

Example 22: Corresponding Pin-Based Receiver Model

```
lu_table_template ( rcvr_rise_1 ) {
  variable_1 : input_net_transition ;
  index_1 ( "0.065, 0.110, 0.400, 0.900, 1.50" ) ;
}

pin ( A ) {
   receiver_capacitance() {
      receiver_capacitance1_rise ( rcvr_rise_1 ) {
         values ( \
               "1.947000e-03",\
               "1.960000e-03",\
               "1.992000e-03",\
               "2.031000e-03",\
               "2.185000e-03" \
         ) ;
      } /* end of pin-based receiver_capacitance1_rise */
        receiver_capacitance2_rise() {
. . . .
      } /* end of pin-based receiver_capacitance2_rise */
        receiver_capacitance1_fall() {
. . . .
      } /* end of pin-based receiver_capacitance1_fall */
        receiver_capacitance2_fall() {
. . . .
      } /* end of pin-based receiver_capacitance2_fall */
   } /* end of pin-based receiver model */
```

4 Requirements for CCS Timing Voltage and Temperature Scaling Libraries

During the process of creating the scaling relationships, PrimeTime runs consistency checking between the libraries in each scaling group. There are some requirements that the libraries need to satisfy, in order for scaling to work correctly

4.1 Structural Requirements Between CCS Timing Libraries

The libraries must be structurally identical. For example, the cell names, pin names, and order of timing arcs should be the same. Ensure that the set of arcs and pins is identical across the members of the scaling library groups. The positions of the arcs and pins within the libraries must be the same as well. The number of output-capacitance indices for all timing arcs should be identical.

If libraries are not structurally identical, scaling relationships will not be created and warning message SLG-202 is issued by Primetime:

```
Warning: Completion of scaling library groups failed.
(SLG-202)
```

4.2 Characterization Loads Between CCS Timing Libraries

The output-capacitance indices used for Composite Current-Source (CCS) driver data must have identical values across the libraries as well. This is not a requirement for CCS timing receiver models or constraints. If this requirement is not satisifed, the following warning will be issued by Primetime:

```
Warning: The members of a scaling library group have
significantly different output-capacitance indices; this
can adversely affect scaling accuracy. (SLG-203)
```

4.3 Library Voltage

The nominal voltage and the voltage specified in the default operating condition of the library must be identical. See Example 23.

Example 23: Nominal Voltage and Operating Condition Voltage Identical in the Library

```
library (EXAMPLE) {
        nom_voltage : 1.050000;
        operating_conditions (MAX) {
                process : 1.000000;
```

```
        temperature : 125.000000;
        voltage : 1.050000;
    }
default_operating_conditions : MAX ;
```

If an operating condition is not defined in the library, Library Compiler X-2005.09 will issue a warning message, create an operating condition, and specify it as the default operating condition that matches the nominal voltage in the library, as shown in Example 24.

Example 24: Missing Default Operating Condition Warning Message in Library Compiler

```
Warning: Line 13, The default_operating_conditions is not
defined. operating_conditions "nom_pvt" is created and
set as the default_operating_conditions. (LBDB-662)
```

5 CCS Timing General Library Requirements

In this section, we will describe general guidelines for creating accurate CCS Timing libraries.

5.1 Delay and Transition Time Trip Points

Unlike NLDM, CCS Timing accuracy is not impacted by output transition time and delay trip point selection for a given input transition time. This is because CCS can calculate the voltage waveform for the entire transition, making the result independent of trip points. Thus, selecting transition time trip points by correlating cell output transition time and delay is not required. However, selection of transition time trip points should take into account accuracy for transition time propagation down a timing path. Usually the best correlation between input transition time and output behavior is obtained when the transition time is measured in the linear region about the logic threshold. Unlike NLDM, CCS will tolerate asymmetrical transition time trip points chosen within 5% of the delay trip points. A library with 25%-55% transition time trip points for rise and 75%-45% transition time trip points for fall with rise and fall delay trip points of 50% is not recommended for NLDM, but is acceptable for CCS-Timing with adequate library validation.

The delay and transition time trip points used during characterization should match those specified in the library. Table 1 shows the variables used to specify trip points in the library header.

Table 1: Library Variables for Delay and Transition Time Trip Points

Delay	Transition Time
input_threshold_pct_rise	slew_upper_threshold_pct_rise
output_threshold_pct_rise	slew_lower_threshold_pct_rise
input_threshold_pct_fall	slew_upper_threshold_pct_fall
output_threshold_pct_fall	slew_lower_threshold_pct_fall

If a library contains transition times extrapolated beyond the characterization range, the slew_derate_from_library variable should be set in the header of the library For a library characterized with 30%-70% trip points and extrapolated to 10%-90% trip points, the slew derate is calculated using the formula below.

$$slew\_derate\_from\_library = \frac{simulation\_trip\_pt\_range}{extrapolated\_trip\_pt\_range} = \frac{(0.7-0.3)}{(0.9-0.1)} = 0.5$$

5.2 Accuracy of Library

The library should have sufficient indices, and the selection of the indices should be such that the interpolation error compared to the reference circuit simulator is within acceptable tolerance. For more details on this topic, please see the CCS Timing Library Correlation document.

5.3 Range of Indices

The range of indices for input transition time and output capacitance must expand across the entire range of allowable values when the cell is used in a design. Make sure that the minimum input transition time and output capacitances extend to minimum values allowed in designs. If the library will be used for signal integrity analysis, some consideration should be given to extending the characterization range. Cells should not be used outside the characterization range, as this may cause large differences in delay and output transition time compared to the reference circuit simulator. Extrapolations outside the table should be avoided.

5.4 Number of Significant Digits

Current vector data can be represented with scientific notation in the reference_time attribute, index_3 variable, and values attributes. This is not essential, but will preserve digits in the current waveform with a small load. It is recommended to use at least 6 significant digits. Example 25 shows a current vector using scientific notation.

Example 25: Current Vector with Scientific Notation and 6 Significant Digits

```
vector ( CCS_fall ) {
      reference_time : 1.012923e+00 ;
      index_1 ( "0.009692" ) ;
      index_2 ( "0.102702" ) ;
      index_3 ( "1.028487e+00, 1.032531e+00,
1.034390e+00, 1.156875e+00, 1.269003e+00, 1.353348e+00,
1.529358e+00, 1.627232e+00, 1.750080e+00, 2.095409e+00" )
;
      values ( "-1.710798e-01, -2.676370e-01, -2.663900e-
01, -2.564110e-01, -2.358110e-01, -1.983590e-01, -
7.114770e-02, -2.992160e-02, -8.843480e-03, -5.037530e-
05" ) ;
    }
```

For the receiver model, the capacitance values in the table should have a minimum of four non-zero significant digits. Example 26 illustrates a receiver model table that meets this criteria. If the library does not meet the guideline for significant digits, Library Compiler X-2005.09 will output an informational message as shown in Example 27.

Example 26: Receiver Model with 4 Non-Zero Significant Digits

```
receiver_capacitance1_fall ( CCS_fall_1 ) {
    values ( \
      "0.003689, 0.003810, 0.003819", \
      "0.004035, 0.003851, 0.003829", \
      "0.005041, 0.003921, 0.003855" \
    ) ;
  }
```

Example 27: Library Compiler Informational Message for Significant Digits

```
Information: Line 10934, The 'vector' has 'values'
0.000958, which has less than 4 significant digits.
(LBDB-670)
```

6 Library Characterization for Timing Analysis

During library characterization and generation, there are many factors that must be considered to ensure accurate timing information in the library. This section discusses the following key items.

- Circuit simulator settings
- Input characterization waveforms
- Delay and transition time trip points
- Operating range of cells
- Selecting characterization points
- Segmentation tolerance
- Receiver modeling
- Other considerations

6.1 Circuit Simulator Settings

During the characterization process, a reference or “golden” circuit simulator such as HSPICE is used to simulate the cell. It is important to simulate with appropriate settings to achieve the best accuracy. In general, the accuracy settings for the spice simulator need to be tighter than what is traditionally used for NLDM library characterization.

Using the HSPICE option “accurate” does a good job of achieving the needed accuracy but will significantly degrade the simulator performance. This should be avoided unless other less costly options do not achieve the desired accuracy. Use of the HSPICE setting, “.option runlvl=5” achieves a good accuracy vs. performance tradeoff. Improper accuracy settings typically result in too few sample points or simulation noise in the CCS current waveform. Simulation noise usually appears as oscillations in either the first 10% of the waveform or the last 30% of the waveform. This can be viewed by importing the output of HSPICE simulation into a graphic post processing tool such as CosmosScope. If tightening of the simulation accuracy does not correct this situation, then the problem may be caused by the process model or voltage or temperature settings. This will need to be corrected to obtain a valid CCS waveform. Additional information on setting accuracy options for HSPICE can be found in the HSPICE Simulation and Analysis User Guide. For simulators other than Hspice, the same method can be used to obtain optimal settings for accuracy and performance.

6.1.1 Control of Simulation Time Step

For HSPICE, dynamic time step controls may be used. For more details, please see reference [3]. This method ensures that enough points are captured when voltages in the circuit are changing quickly and reduces the number of points when voltages in the circuit change slowly. This can be achieved by setting a large simulation time step in the **.tran** statement and setting the **absvar** and **relvar** variables in the **.option** command to control the simulation time steps. Example 28 shows the simulation commands that should be included in an HSPICE deck to enable dynamic time stepping. Segmentation tolerance will be discussed in Section 6.6.

Example 28: Using Adaptive Time Stepping in HSPICE

```
.param segmentation_tolerance=0.005
.param vswing='vdd-vss'
.param relative='segmentation_tolerance/vswing'
.option accurate absvar=segmentation_tolerance
relvar=relative
.option runlvl=5
.tran 10ps 1.0u
```

If the simulator used for library characterization does not have adaptive time step control settings like **absvar** and **relvar**, small enough fixed time steps should be used such that the maximum voltage change per time step is less than the segmentation tolerance.

6.1.2 Simulation Speed

If the **.tran** simulation setting in Example 28 is used, **autostop** should be included in the Spice deck to minimize simulation runtime, otherwise a total time of 1 microsecond will be simulated. The **autostop** feature terminates the simulation after all **.meas** statements have been completed. If characterizing for CCS, **.meas** statements within 1mV of the final rail voltage on the input and output of the cell should be added to the Spice deck. Example 29 shows the **autostop** and **.meas** statements for an inverter with a rising output. If the cell has previously been characterized for NLDM, **autostop** may already be set at the second transition time threshold and should be changed when characterizing for CCS timing. See Section 2.1 for more details.

Example 29: Using `autostop` to Speed Simulation

```
.option autostop
.meas tran tstopi  when v( input)='vss+0.001'     fall=1
.meas tran tstopo  when v( output)='vdd-0.001'     rise=1
```

6.1.3 Simulation Algorithm

Simulator transient analysis has several methods for numerical integration. The **trapezoidal** algorithm should be used for library characterization. It provides the highest accuracy and fastest simulation time. The **gear** method should be avoided, since it is targeted for inductive or lossy circuits and is not as accurate. The trapezoidal method is the default numerical integration algorithm in HSPICE.

6.2 Input Characterization Waveforms

In an actual design, the waveform seen at a cell's input pin arrives from any cell in the library through a complex RC network. All of the input waveforms along a design path will differ from the pre-set stimulus waveform used for characterization. It is necessary to create a stimulus waveform that minimizes the error introduced by this difference.

In order to reduce potential optimism introduced by characterization waveforms, min libraries used for min analysis should be characterized using the fastest waveform shape or the linear ramp method. Max libraries used for max analysis should be characterized using waveform shapes that resemble those in an actual design. Traditional library characterization has typically used the output of another cell, known as an "active driver" to generate the waveform shapes. The active driver method is not recommended for several reasons.

The first reason is that there are many different driver/load cell combinations in a design. Choosing a specific driver improves accuracy for receiver cells connected to that cell at the expense of accuracy for receiver cells connected to other driver cells. Empirical results with many libraries has shown that overall correlation accuracy is greater with the Synopsys predriver waveform shown below than with active driver cells, especially for worst-case results.

The second reason is that the use of an active driver can introduce errors depending upon the method chosen to generate a desired output slew. These methods include:

- Sweep output capacitance for fixed input slew
- Sweep input slew for fixed output capacitance
- Sweep both

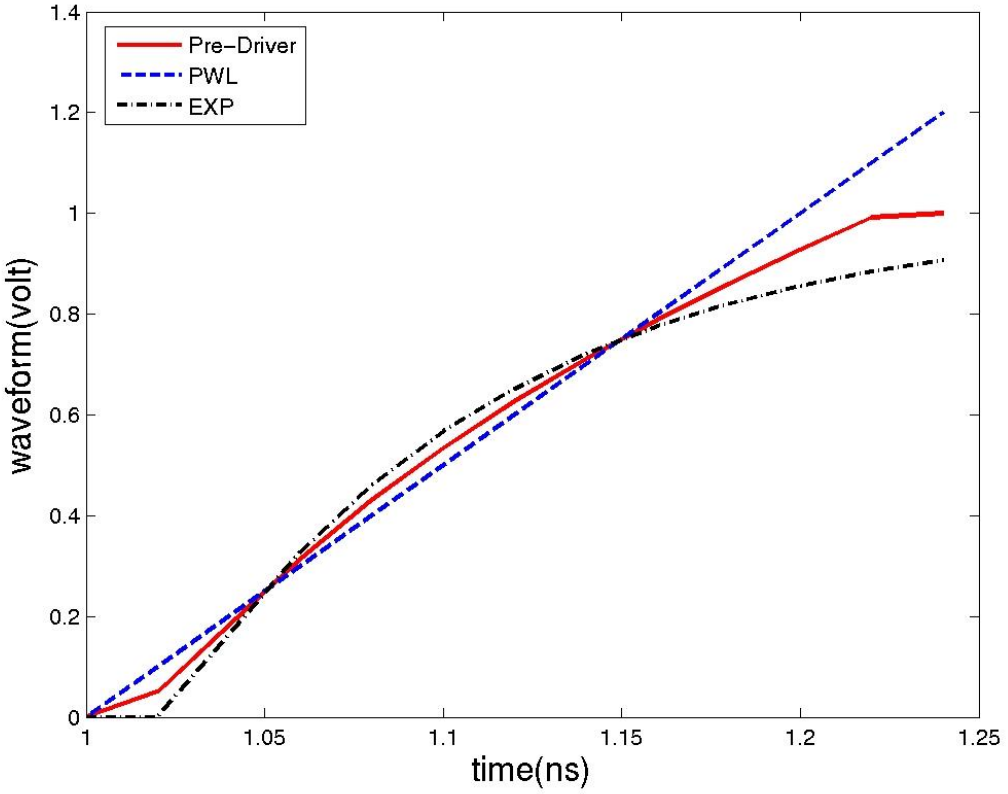
The shape of the output waveform is different in the above cases even if the slew values are exactly the same. Picking one method may minimize the error for some specific cell instances in characterization and static timing analysis, but it may worsen the error for other instances. For a design testcase with homogeneous cells and/or networks, using an active driver specific to the testcase can give better results, but that approach may be meet sign-off quality for static timing analysis under all conditions.

The best choice for a pre-driver waveform that minimizes error for designs should be based upon empirical results on many different technologies libraries and designs. The Synopsys pre-driver waveform method does attempts to place the introduced error between the two extremes of:

- Fast input slew with no RC network effect
- Slow input slew with significant RC network effect

Figure 9 compares the Synopsys pre-driver waveform against a fast input slew and a slow input slew. The Synopsys pre-driver waveform method strives to minimize the maximum error possible over all driver, load, and net scenarios. The CCS Timing characterization scripts provided by Synopsys generate characterization waveforms using the Synopsys pre-driver waveform method.

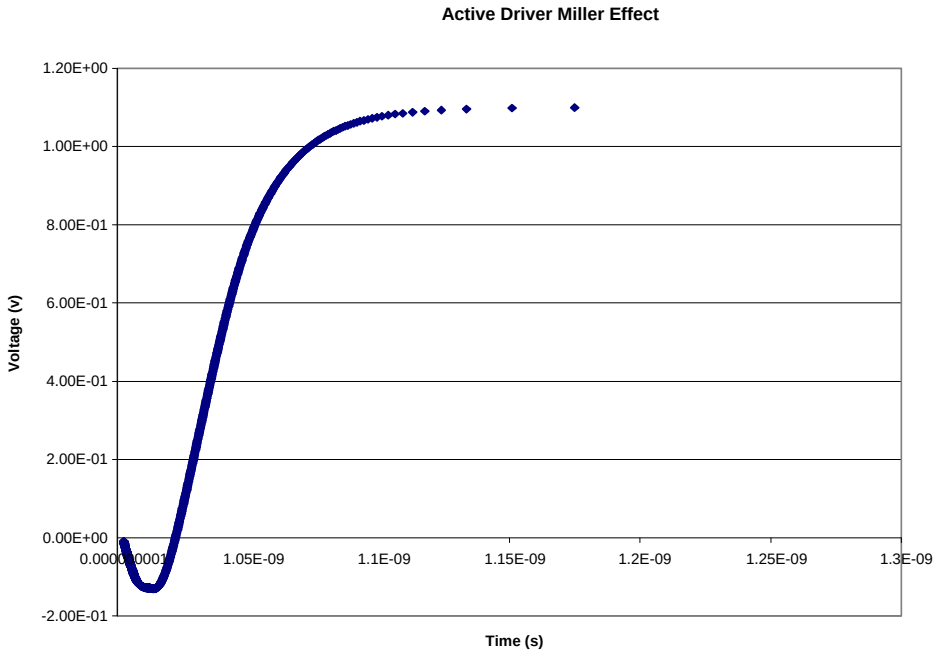
Figure 9: Synopsys Pre-Driver Waveform



The input characterization waveforms should remain at a fixed voltage for a short period of time before transitioning. This allows the circuit to stabilize in its DC operating condition. The transition should not start at t=0.

If the active driver method is chosen for library characterization, the reasons against using active drivers should be carefully considered. In addition, if the method for active driver characterization sweeps output capacitance for a fixed input slew, the input slew should be chosen to minimize Miller effect at simulation startup. Figure 10 shows the output of an active driver with a Miller effect of ~10% of the full rail voltage. This is an excessive amount and can impact library accuracy because the effect is propagated to the cell being characterized. When the cell's output current is integrated, the voltage is assumed to be the same polarity as the current. During the Miller glitch, current and voltage have opposite polarities, and this information is not included in the library and is a source of correlation error.

Figure 10: Output of Active Driver with Large Miller Effect



6.3 Synopsys predriver waveform algorithm

The Synopsys **"CCS jumpstart scripts"** include a predriver waveform generator. It produces a SPICE syntax linear combination of PWL and exponential waveform:

v_ramp(t) = Vdd * (t-tstart)
v_step(t) = Vdd * (1 - exp(-(t-tstart)/RC))
v_predriver(t) = 0.5 * (v_ramp(t) + v_step(t))

The predriver waveform may be demonstrated by using this HSPICE netlist:

```
* Get the pre-driver waveform for input slew = 1ns
.param inslew=1n
.param vdd = 1.0v, vlow=0.2, vhigh=0.8
.param tl=`1n+0.2n`, th=`1n+0.8n`
.param pct_low=0.2, pct_high=0.8
* ramp input
V1 1 0 PWL (1n 0, '1n + inslew' vdd)
* step-response
.param rc=`(th-tl)/log((1.0-pct_low)/(1.0-pct_high))`
.param tx=`tl+rc*(log(1.0-pct_low))`
V2 2 0 EXP ( 0 1 tx rc 20n rc)
* mix of ramp and step-response * 0.5 * (v1 + v2)
E3 3 0 poly(2) 2 0 1 0 0 0.5 0.5
.option post=1
.tran 1ps 20n
```

```
.plot
.end
```

6.4 Delay and Transition Time Trip Points

For CCS Timing, selection of transition time trip points will impact the transition time value that is reported and used for delay calculation in the next stage. During library characterization, the current waveform is captured for the full rail-to-rail transition of the output, and during delay calculation PrimeTime calculates the voltage transition at the driver and receiver(s). The input transition time to the next stage is measured from the waveform at each receiver using the transition time trip points specified in the library.

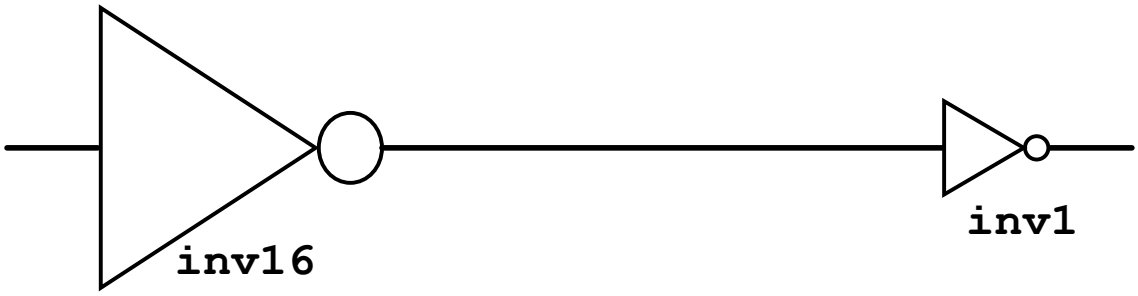
6.5 Operating Range of Cells

The operating range of cells should be considered when you determine the minimum and maximum values of indices in the table. The methods described here are an example of how to determine the operating range of a cell. Library developers may use different methods.

Typically, the operating range is constrained by the electrical rules for the library and process. This is how the operating range may be determined:

- The minimum transition time value can be determined by the output transition time of the inverter with the maximum size (small Rd) driving the minimum size inverter (large Rd) with no wire capacitance. See Figure 11.

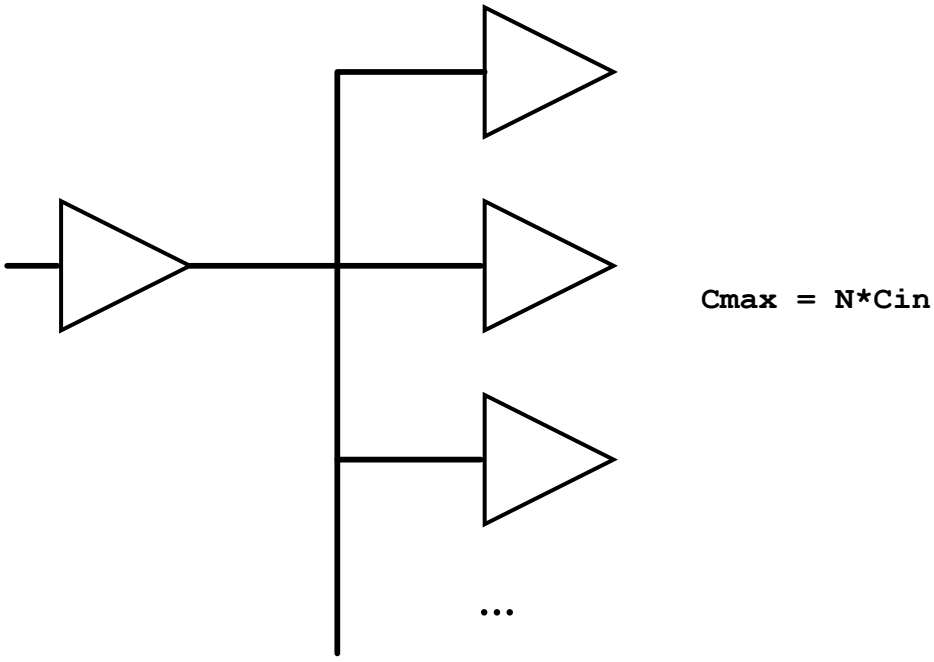
Figure 11: Determining Library Minimum Transition Time



- The minimum output capacitance can be determined by the load of the minimum size inverter. This typically entails evaluating the inverter with the smallest transistor sizes (width and length), and therefore smallest value of gate capacitance at the input port.

- The maximum transition time and maximum output capacitance are interdependent. The maximum transition time is dependent on process technology, design rules, and design applications. A cell's maximum capacitance can be determined by its maximum fan-out load without exceeding the maximum transition time. See Figure 12.

Figure 12: Cell Driving Its Maximum Load, Cmax



Important note: You should set **max_transition** and **max_capacitance** attributes in the library to make sure that cells are not used outside of the characterization range.

6.6 Selecting Characterization Points

The operating range sets the minimum and maximum indices in the table. Once the operating range has been determined, the next step is to select the intermediate characterization points. The following decisions must be made:

- How many indices are needed in the table?
- What are the values of the indices in the table?

As a general rule-of-thumb, Synopsys recommends starting with a 7 by 7 table as the minimum table size. The decision on intermediate indices should be made such that the interpolation errors are minimized. Library characterization experts have developed different techniques to determine the indices. In general, selecting indices that are not equally spaced can minimize interpolation errors. Typically, in regions where the input transition time is small (fast) and the output load is small (light), delay and output transition time tend to increase non-linearly with increasing input transition time and output load. Thus, in these regions, more characterization points are needed to minimize interpolation

errors and to achieve better correlation with reference circuit simulators. We will discuss two different methods for determining indices.

6.6.1 Geometric Spacing and Over-Sampling

One technique uses a number generator program to generate the indices. Given the minimum and maximum operating range, a number generator program is used to generate the indices using geometric spacing, such as a distribution of equal size logarithmic steps. This approach ensures that in regions with small values of input transition times and output capacitance values, the indices are placed closer together in order to minimize interpolation errors. The technique also over-samples by generating tables from 5 by 5 indices, up to 21 by 31 indices.

Data for each table (delay and transition time) are plotted and evaluated to determine non-linear regions in the table. Based on this, a smaller number of indices are selected; while maintaining the requirements for minimizing interpolation errors.

For more information on the above technique, please see reference number [1].

6.6.2 Plotting of Interpolation Errors

Another technique also generates a set of indices that are not equally spaced for minimizing interpolation errors. However, at the start of the cell characterization, indices are equally spaced. For each quadrant of the table, a matrix of 10 by 10 equally spaced points are simulated using the reference simulator. The interpolation errors are measured and plotted. Based on these plots, one may observe that in regions where input transition time and/or output capacitances are small, the interpolation errors tend to be larger compared to the other regions of the table. One may apply automation to re-position the indices in the table while reducing interpolation errors.

For more information on the above technique, please see reference number [2].

6.7 Segmenting the Current Waveform

The output current waveform from the simulation will typically have hundreds of points. To reduce library size, a minimal number of points from the current waveform should be saved. However, there is a trade-off between CCS library accuracy and number of points saved.

The Synopsys **“CCS jumpstart scripts”** include a module that reduces the detailed waveform to a shortened piecewise-linear version. This process is called segmentation. The script uses a segmentation tolerance to determine the

accuracy of the segmentation. The number of points saved from the current waveform increases as the required accuracy is increased.

Here is a brief summary of how the **"CCS jumpstart scripts"** segmentation algorithm in "CCS_Segment.pm" operates:

For each timing arc:

1. Obtain i(t), V(t) waveforms from HSPICE simulation

2. Traverse from end time to start time to select starting time for current measurement

- Skip all initial voltage glitches of reverse polarity

3. Traverse from start time to end time to select stop time for current measurement

- Use HSPICE autostop to end simulation when voltage reaches to rail
- To support cases where output does not reach rail, select a simulation time that is at least double the second slew threshold time

4. Find peak current value

5. Modify first current value to allow Voltage to exactly match at first saved time point

- Calculate extrapolated first (time, current) sample point
- Ensure that time is positive, and integrated voltage matches actual

6. Modify first current value to ensure that it is not the peak current

- Accurate peak current sampling is critical for scaling in Primetime
- Library Compiler will flag an error if waveform is monotonic

7. Traverse all remaining current values, and select only the remaining i(t) values necessary to assure that

| V(actual) – V(ccs) | **<** ccs_segment_tolerance,

Use trapezoidal integration to calculate the CCS voltage

$$\Delta V(t_n) = \sum_{tstart}^{t_n} \frac{[I(t_n) + I(t_{n-1})]}{2} \frac{(t_n - t_{n-1})}{C}$$

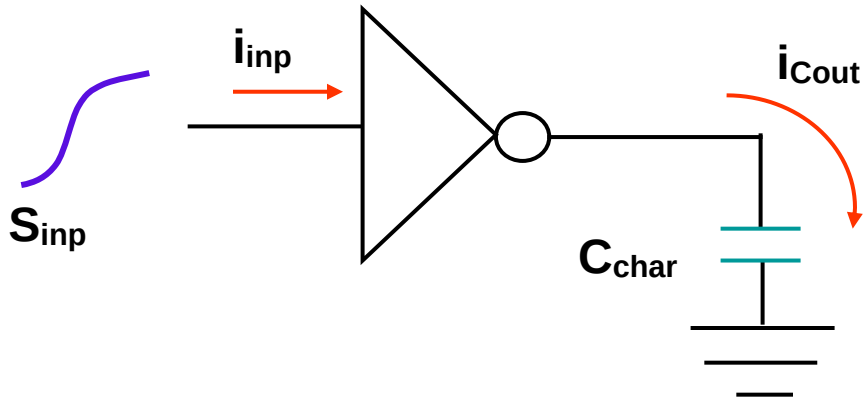
8. Write the optimized i(t) vector into the characterization database

9. Perform Quality of Results test on the segmented waveform

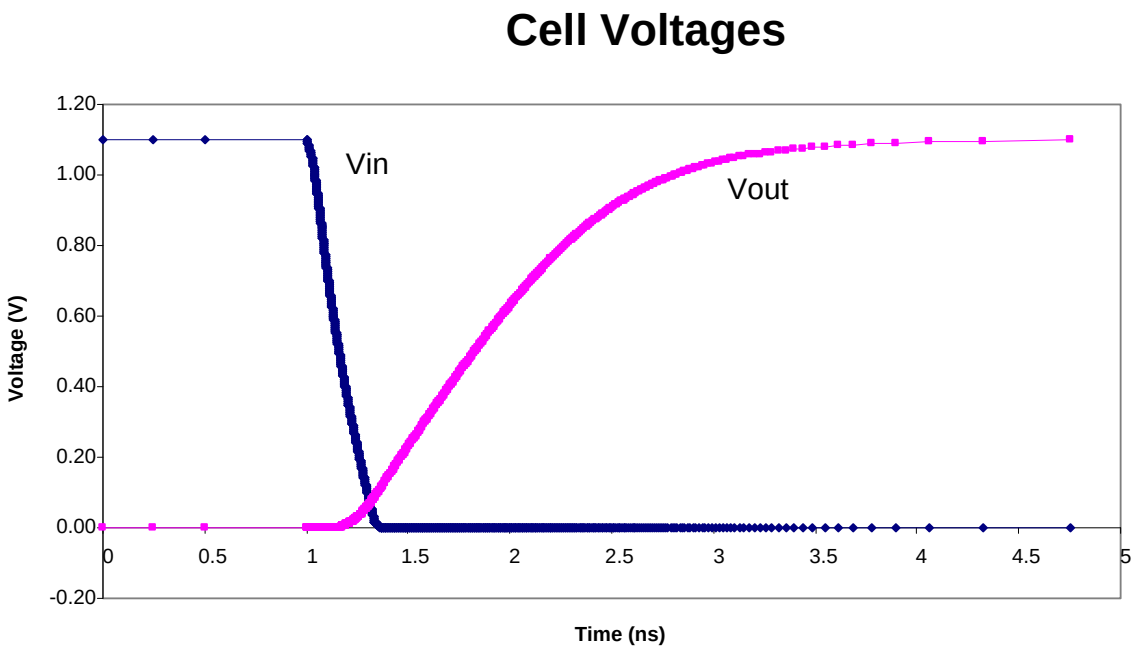
Figure 13 shows the characterization schematic, the resulting simulation waveforms, and the segmented current that is included in the library.

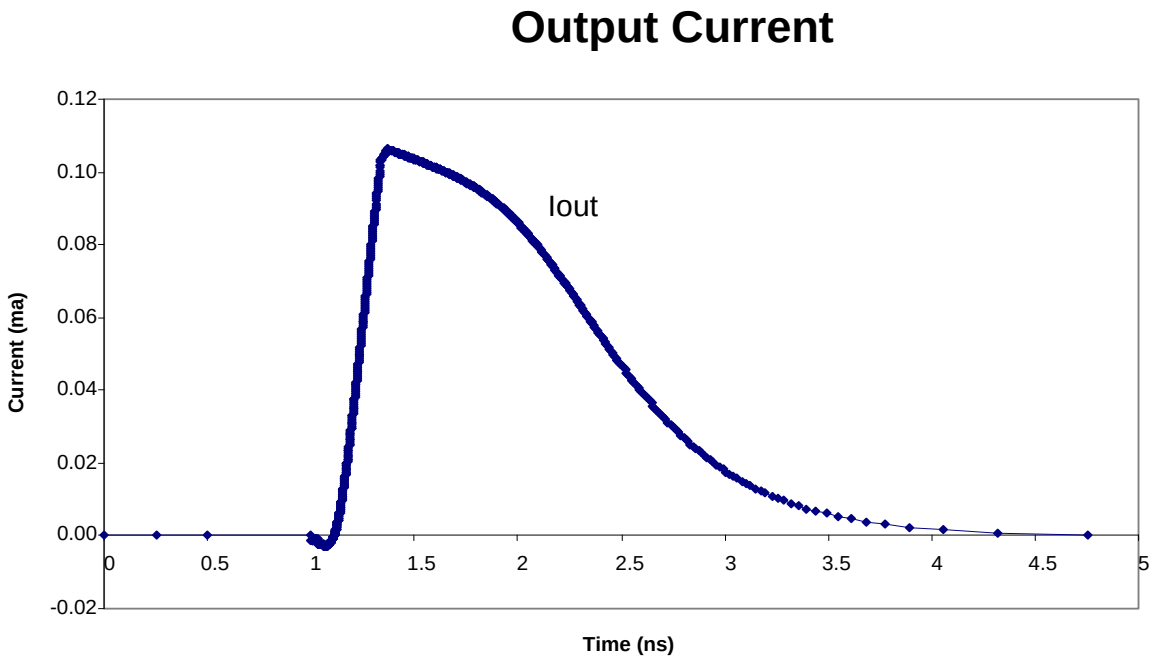
Figure 13: Segmenting the Output Current

a) Characterization Schematic

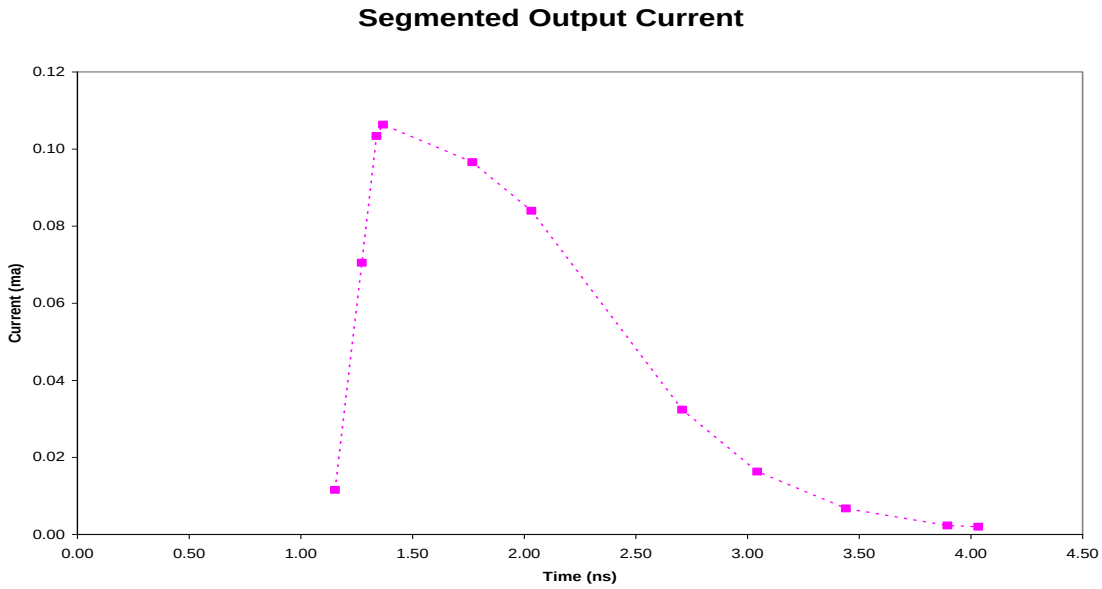


b) b) Simulation Results





c) Segmented waveform results



The quality of the segmented current is checked by integrating it, and the resulting voltage waveform is compared to the simulation voltage waveform. Points are selected from the current waveform such that the voltage difference is within the specified segmentation tolerance. The simulation time step must be

set such that the maximum voltage change per time step is less than the segmentation tolerance. Circuit simulators such as HSPICE allow adaptive time step control such that voltage change does not exceed the segmentation tolerance. If the simulator used for library characterization does not have adaptive time step control, small enough fixed time steps should be used to meet this criterion. For more information, refer to section 6.1.1.

6.8 Deriving Capacitances for the Receiver Model

The CCS receiver model uses two capacitance values to model the cell loading during the transition. For optimal CCS receiver accuracy, the value calculated as C_1 should be chosen to ensure that the point (V_2,t_2) is exactly matched. The value calculated as C_2 should be chosen to ensure that the point (V_3,t_3) is exactly matched.
The Synopsys **"CCS jumpstart scripts"** includes a module that calculates optimal values for each receiver capacitance. Here is how the module "CCS_Solve.pm" operates:

For each rising timing arc:

1. Find C1 "seed value" below delay threshold

$$C_{1\text{seed}} = \frac{\int_{t1}^{t2} I(t)dt}{V_2}$$

t1 = starting time , t2 =time at delay threshold V2

2. Find C2 "seed value" above delay threshold

$$C_{2\text{seed}} = \frac{\int_{t2}^{t3} I(t)dt}{(Vdd - V_2)}$$

t2 = time at delay threshold V2, t3 = ending time

3. Calculate optimal C_1 and C_2 values.

- Brent algorithm is used to minimize curve fit error over a specific waveform range
- The default waveform range is a single point (V_2, t_2) for C_1

C_1 = CCS_Solve::brent ($C_{1\text{seed}}$, …, (V_2, t_2));

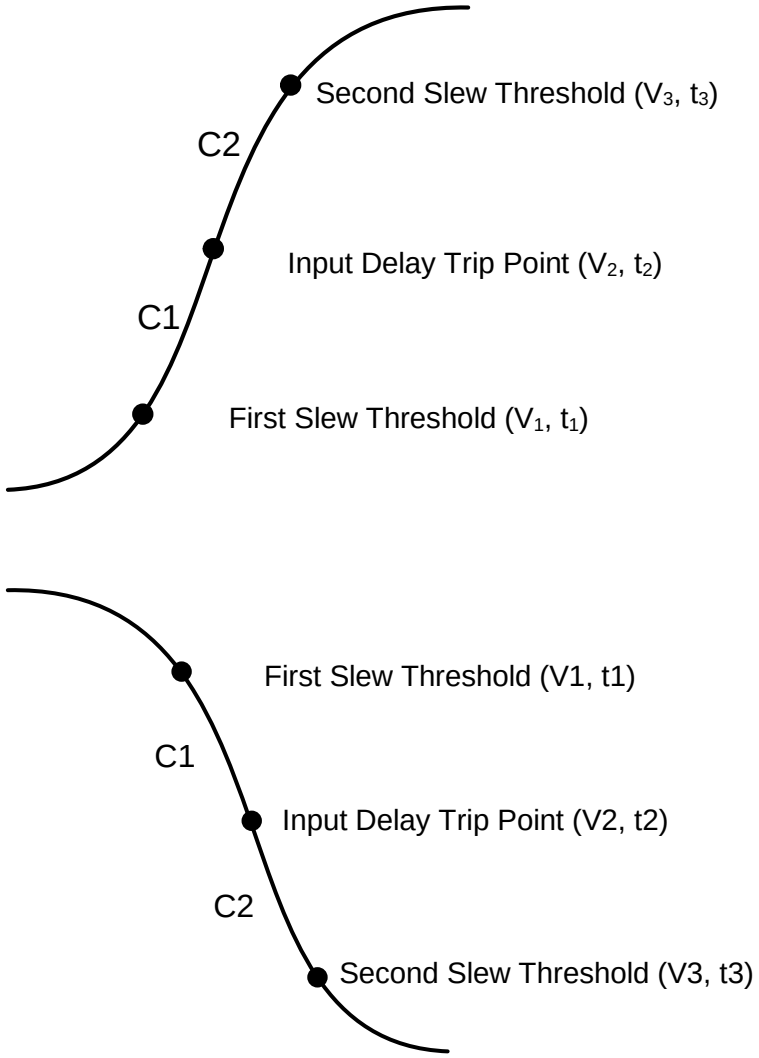
- The default waveform range is a two points (V_2, t_2) and (V_3, t_3,) for C_2

C_2 = CCS_Solve::brent ($C_{2\text{seed}}$, …, (V_2, t_2), (V_3, t_3,));

4. Test receiver values by calculating delay and slew values and comparing to actual delay and slew values.

5. Store results into characterization database in Liberty syntax

Figure 14: Characterization Waveform for rising and falling slews



.

6.9 Other Considerations

6.9.1 Simultaneous Switching

Cells with simultaneously switching inputs are not recommended for gate-level timing analysis. Libraries characterized using simultaneously switching inputs should not be used for static timing analysis.

6.9.2 Merging Timing Arcs

Merging related timing arcs to produce a single worst-case arc may be interpreted by CCS as not physically possible data and is not recommended. Merged CCS current data may generate RC-104 warning messages. A sample message is shown below in Example 30.

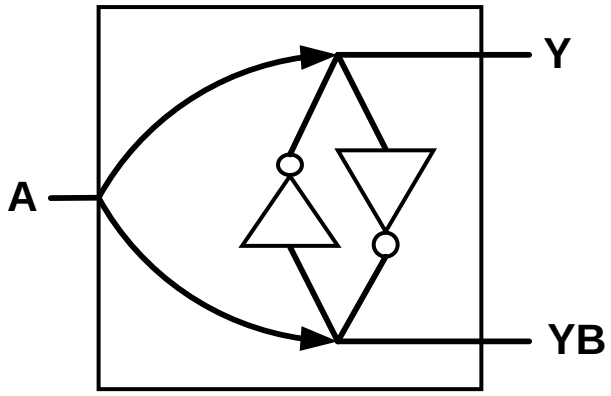
Example 30: PrimeTime RC-104 Warning Message

```
Warning: Failed to compute the cell timing arc
   (design/INV) INV/A-->Z (max-falling negative_unate)
   because there is a problem with the driver's CCS data.
   [r/f inp_slew = 0.034/0.034, out_cap = 0.107  (lib
units)]
```

6.9.3 Three-Dimensional Timing Arcs

Timing arcs with delay and transition time dependent on input transition time, output load of the arc, and additional external loading on a related cell output are not recommended for RC delay calculation. CCS Timing does not support three-dimensional timing arcs. In Figure 15, the timing arc from A to Y is dependent on the load at YB. Similarly, the timing arc from A to YB is dependent on the load at Y.

Figure 15: Cell with Inter-Dependent Three-Dimensional Timing Arcs



6.9.4 Level Shifters

A cell can be characterized with the input rails different from the output rails. This is supported for CCS Timing.

6.9.5 Cells with Unbuffered Inputs

If the cell is multistage, a pin-based receiver model is recommended for CCS to capture the pass gate behavior for each input transition. An arc-based receiver model can be used for a single stage. A fully indexed receiver model table can be used to capture the load dependency of the transmission gate when it is open. When it is closed, a receiver model with a single capacitance index can be used, since there is no load dependency. See Section 3.4 for more information.

7 Library Compiler Checks for CCS Timing Libraries

Below is a summary of the CCS Timing checks implemented in the X-2005.09 version of Library Compiler.

CCS Library Requirement	Check	Section	LC Message
Transition Check	Integrate the current waveform and check that the voltage is within 5% of the final rail voltage. Warn if this criteria is not met, and error out if voltage fails to reach second transition time threshold.	2.1	LBDB-668 LBDB-669

Current Polarity	Check that the current polarity is correct for each timing arc.	2.3	LBDB-664
Zero Current	Check that the current vector does not contain zero current for the entire current vector. Check that the current vector does not contain zero current values.	2.1 2.2	LBDB-664
Peak Current	Check that the first or last current in the current vector is not the peak current.	2.4	LBDB-665
Library Voltage	Check that the nom_voltage and the default operating condition voltage in the library are identical.	4.3	LBDB-662
Identical reference_time	Check that the reference_time attribute is identical for a fixed transition time and varying loads.	2.7	LBDB-654
Negative reference_time	Check that the reference_time attribute is non-negative for each current vector.	2.6	LBDB-667
Number of significant digits	Check whether receiver model table values and current vectors have enough significant digits whether using scientific or decimal notation.	5.4	LBDB-670
Current Vector Variables	Check that `variable_3` in an `output_current_template` is always `time`.	2.5	LBDB-653

8 CCS Timing Liberty Extension (June 2016)

To further improve accuracy of timing analysis, the receiver capacitance modeling syntax was extended to support multiple segments. In the multisegment receiver capacitance model, the voltage rise or fall at a cell input node is divided into multiple (typically, more than two) mutually exclusive segments, as shown by the dotted lines in the following figure. The receiver capacitance values are characterized for each voltage segment. Using multiple segments better represents the nonlinearity of the net receiver capacitance at the node.

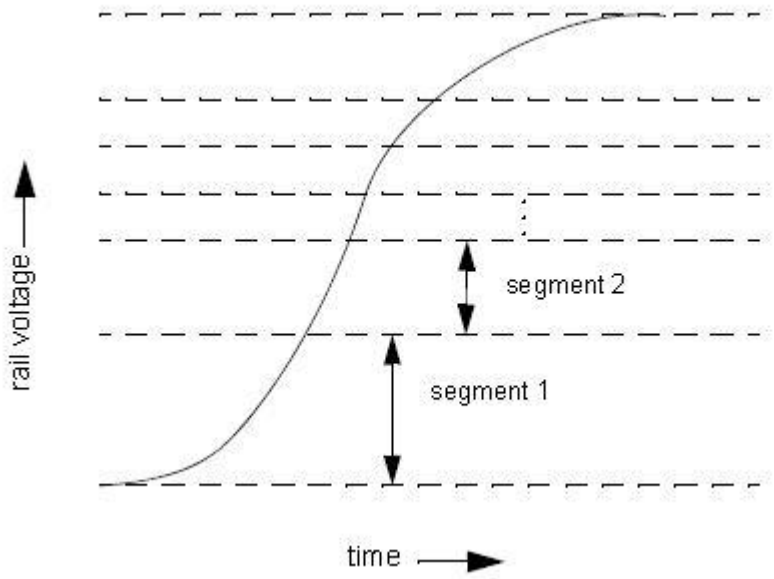


Figure 16: Multisegment Receiver Capacitance Model

For each segment, use the new `receiver_capacitance_rise` and `receiver_capacitance_fall` groups to store the receiver capacitance values as lookup tables. To specify the segment, use the new `segment` attribute in these groups. You can define these groups in the pin-level `receiver_capacitance` and `timing` groups. This model uses the same input slew in the `input_net_transition` (`index_1`) values of the lookup tables across all the segments, irrespective of different waveform shapes in each segment. The input slew is defined using the library-level `slew_derate_from_library` attribute.

9 References

[1] "Using PathMill® to Characterize Library Cells", Hai Vo-Ba, System VLSI Technology Operation, Hewlett Packard, Synopsys Users' Group Conference, 2001.

[2] "Constructing Better Non-Linear Delay Models (NLDM) to Improve Timing Closure", Steven E. Start, Erik N. Comparini, American Microsystems, Inc., Synopsys User s' Group Conference, 2001.

[3] "Simulation Speed and Accuracy", Pages 327-330, HSPICE Simulation and User Guide, Version W-2005.03, March 2005, from Synopsys, Inc.