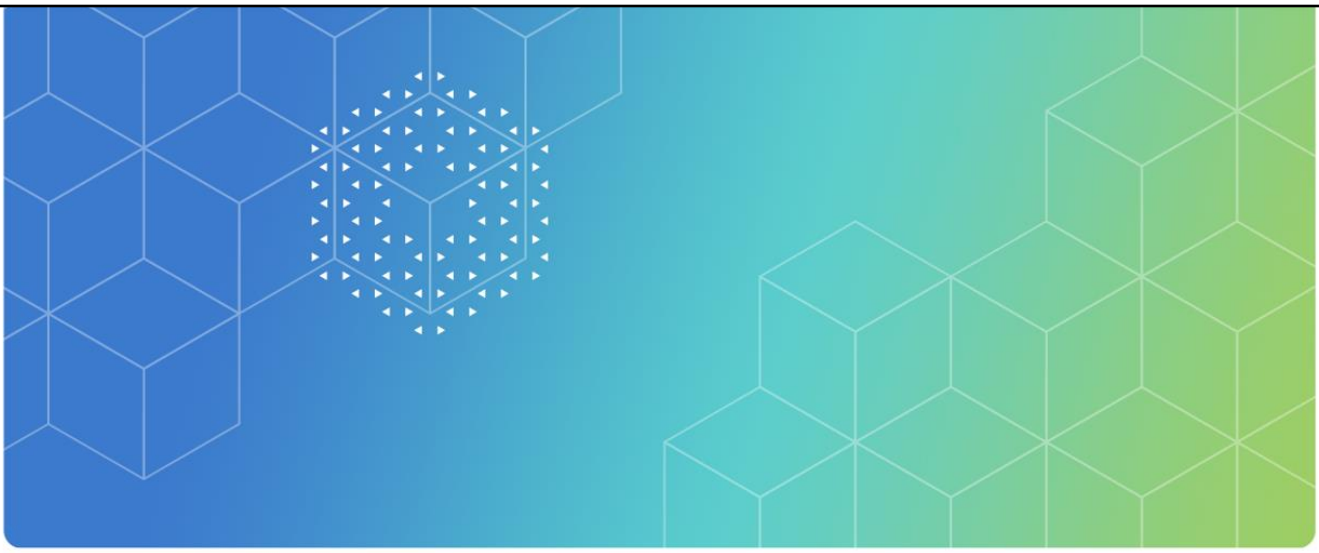
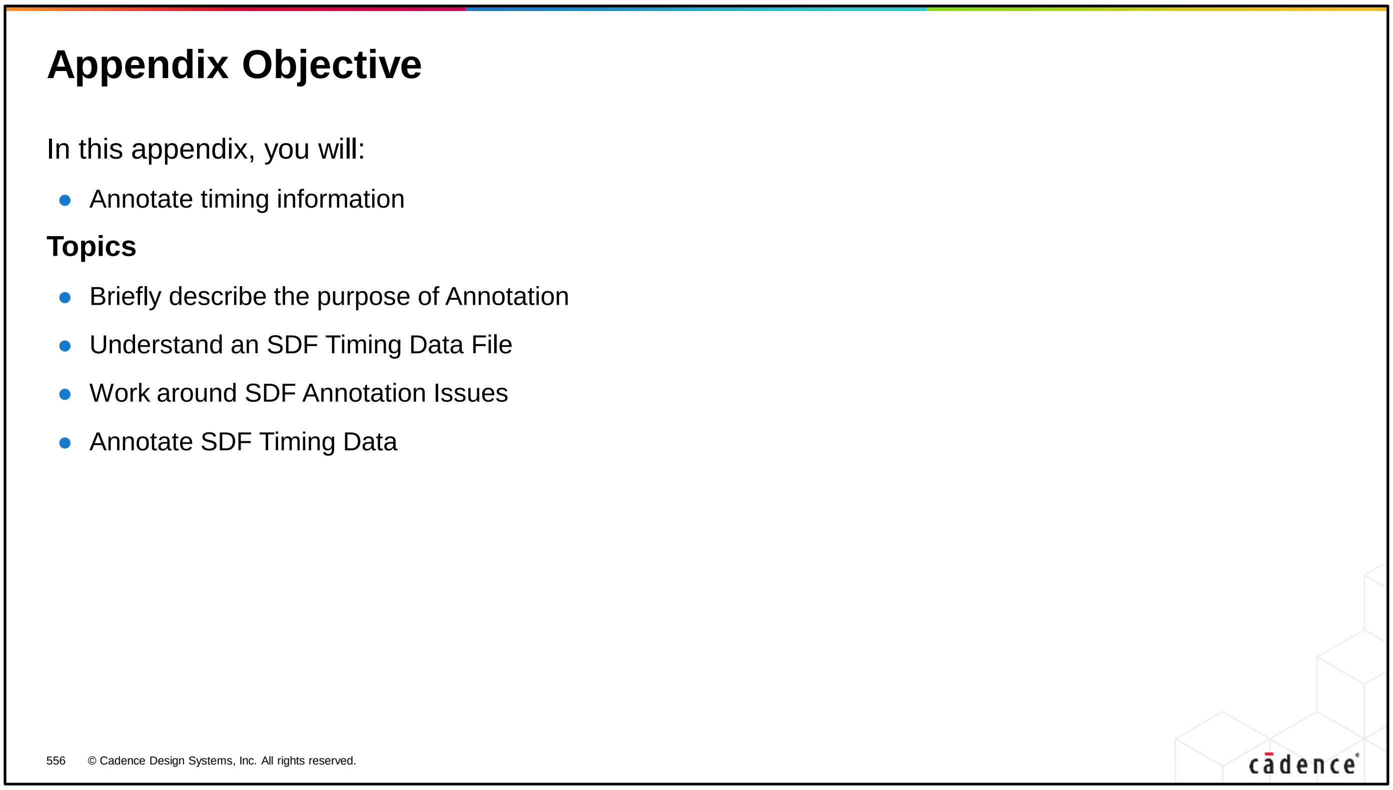


Appendix C

SDF Annotation Overview

cādence

This page does not contain notes.



Appendix Objective

In this appendix, you will:

- Annotate timing information

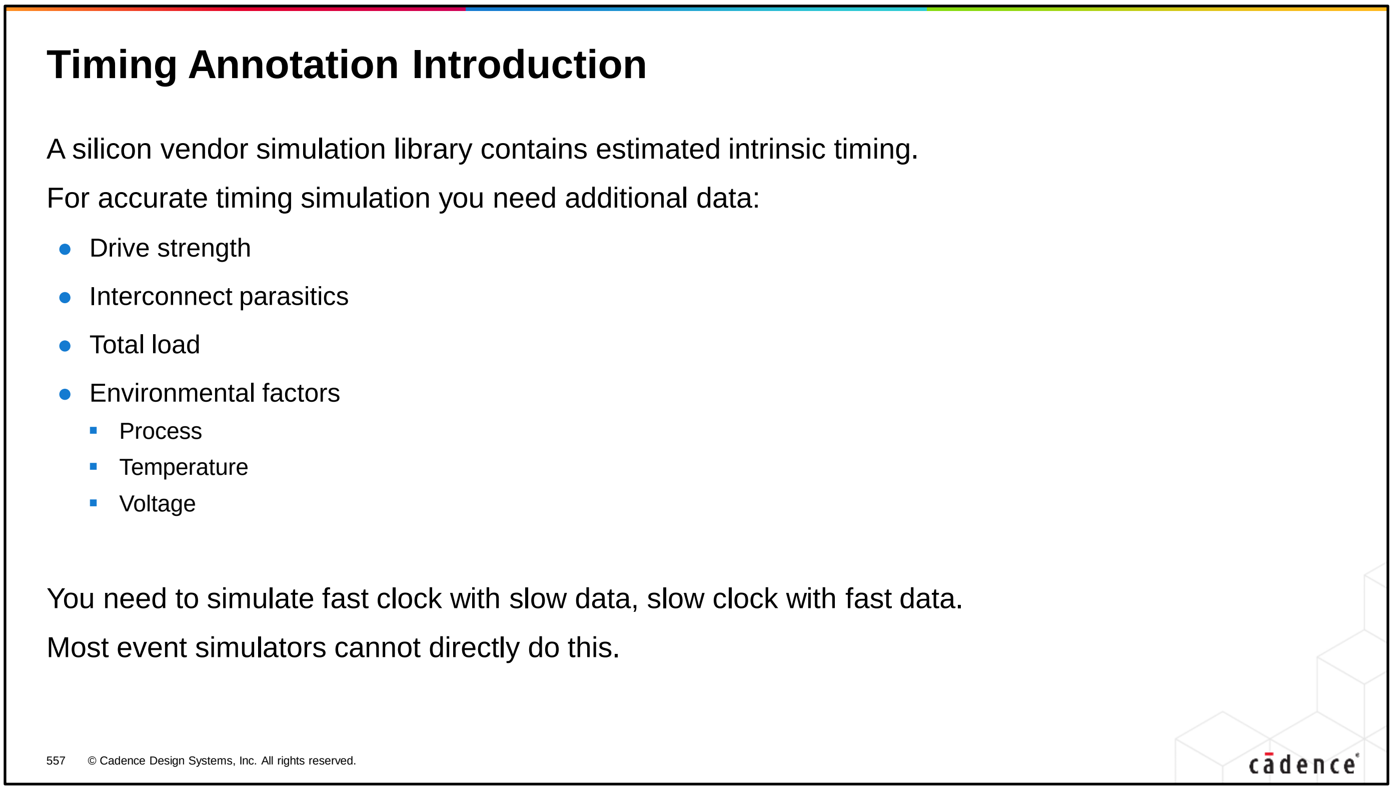
Topics

- Briefly describe the purpose of Annotation
- Understand an SDF Timing Data File
- Work around SDF Annotation Issues
- Annotate SDF Timing Data

This module communicates to you the knowledge needed to annotate timing data to your Verilog designs and to modify the Standard Delay Format (SDF) file for debugging purposes.

This module briefly describes the purpose of annotation, explains some of the SDF keywords, discusses some issues you may encounter, and demonstrates the annotation process.

The IEEE standard 1497-2001 “SDF for the Electronic Design Process” defines the SDF version 4.0.



Timing Annotation Introduction

A silicon vendor simulation library contains estimated intrinsic timing.

For accurate timing simulation you need additional data:

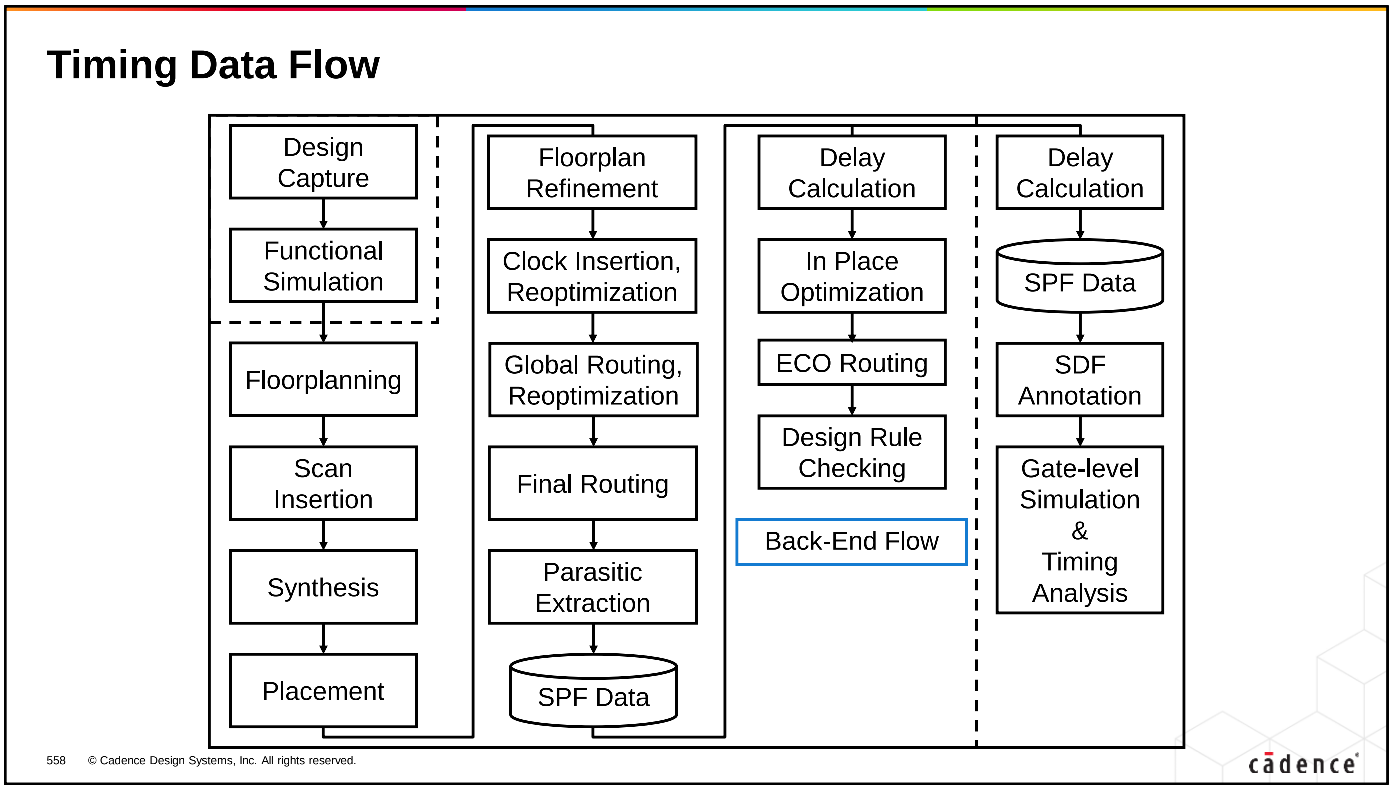
- Drive strength
- Interconnect parasitics
- Total load
- Environmental factors
  - Process
  - Temperature
  - Voltage

You need to simulate fast clock with slow data, slow clock with fast data.

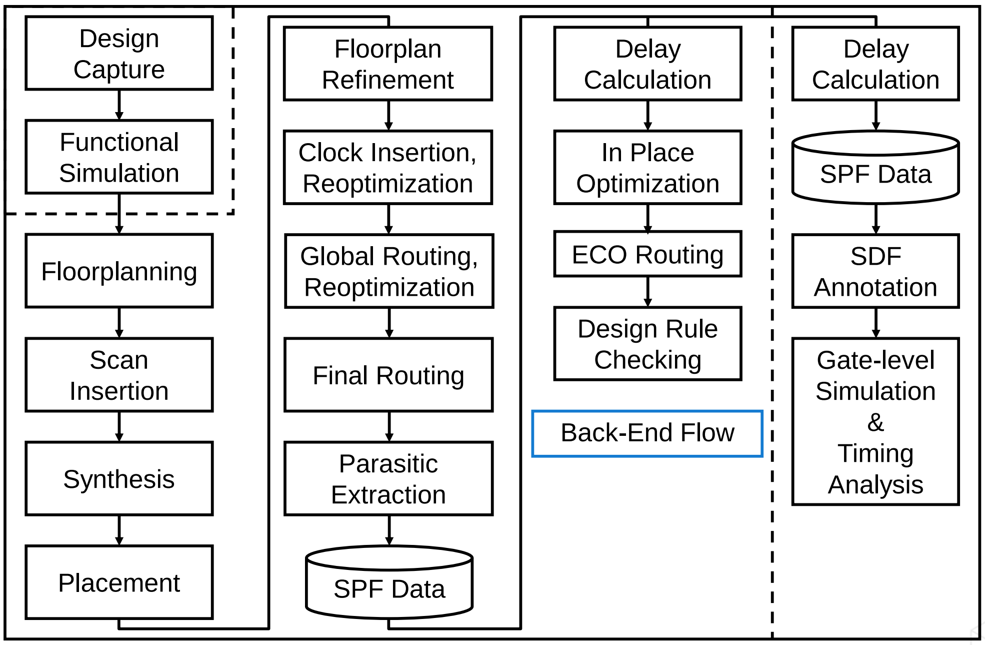
Most event simulators cannot directly do this.

A silicon vendor simulation library typically contains estimated intrinsic timing. For accurate timing simulation you need additional data, especially for the interconnect delay. Delay calculators provide this data using a standard delay format (SDF).

The hardware description languages have constructs to specify minimum, typical, and maximum timing information, but simulators can use only one of these timing sets during a simulation session. You can prepare and annotate a SDF file containing fast clock times and slow data times for setup checks, or slow clock times and fast data times for hold checks, and run the simulation with the new timing data. This will provide results that are more accurate, but still not as accurate as a static timing analysis tool.

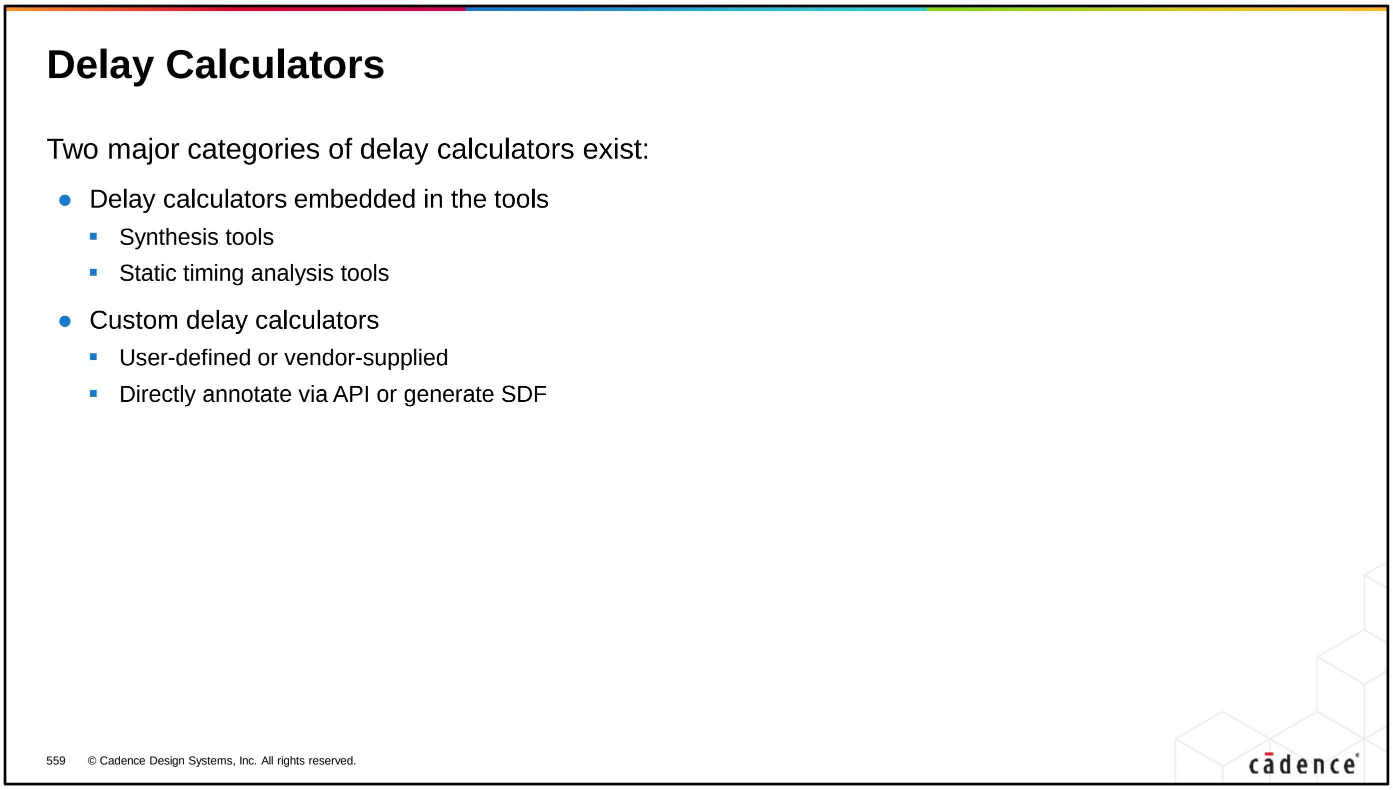


Timing Data Flow



The typical back-end flow now tightly integrates synthesis, layout, and route tools. These tools interchange data using proprietary database formats, and generate SDF data only for analysis tools outside of the flow.

A delay calculator requires a structural representation of the design. It can generate estimated delays based only on design connectivity and hierarchy, or detailed delays using parasitic information extracted by layout tools. The reduced parasitics may be in the Standard Parasitics Format (SPF).



Delay Calculators

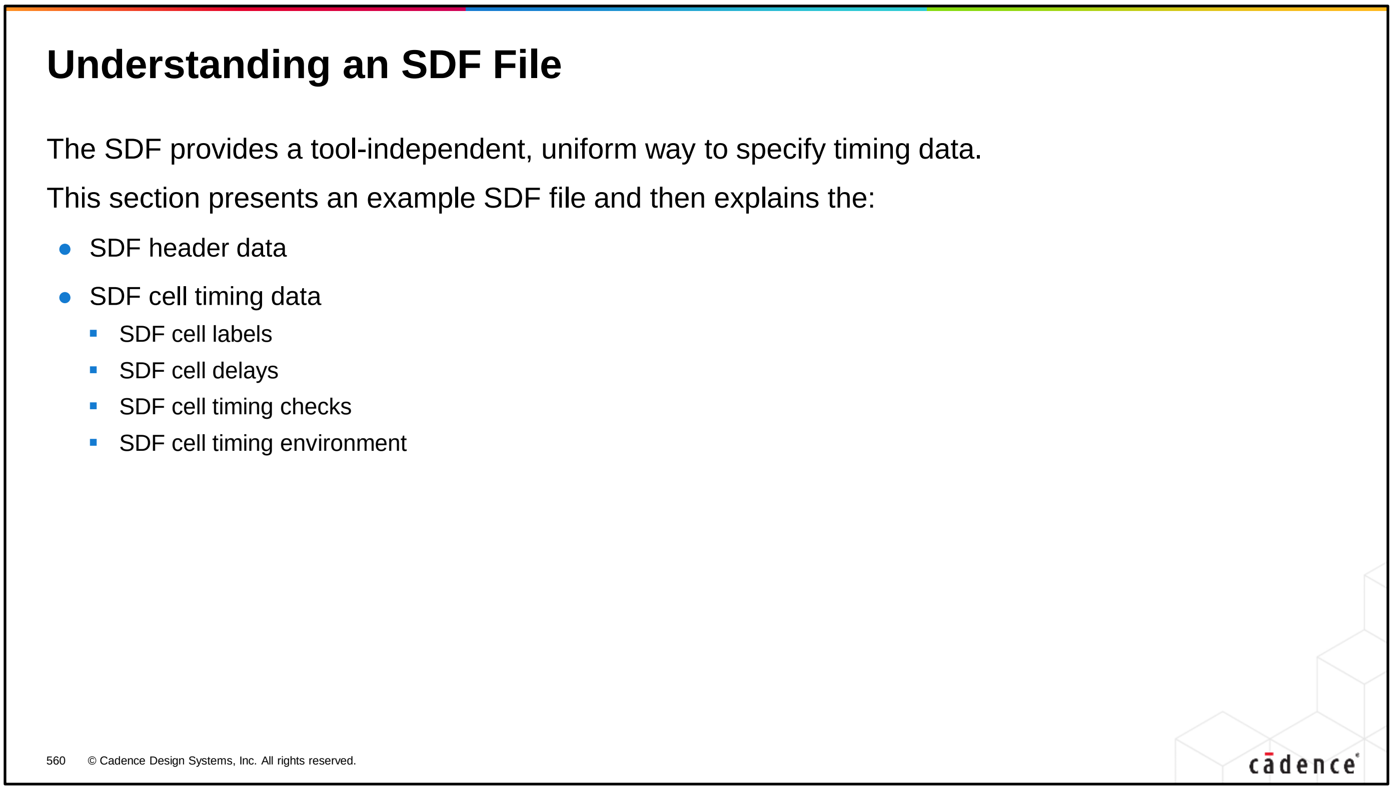
Two major categories of delay calculators exist:

- Delay calculators embedded in the tools
  - Synthesis tools
  - Static timing analysis tools
- Custom delay calculators
  - User-defined or vendor-supplied
  - Directly annotate via API or generate SDF

559 © Cadence Design Systems, Inc. All rights reserved.

You can write a custom delay calculator. You must choose an appropriate delay equation.

Most ASIC vendors provide a delay calculator based on their manufacturing process. They can write these in C and use the Application Programming Interface (API) to directly annotate the design during simulation. They can also write standalone programs that generate SDF for the built-in timing data annotator.



Understanding an SDF File

The SDF provides a tool-independent, uniform way to specify timing data.

This section presents an example SDF file and then explains the:

- SDF header data
- SDF cell timing data
  - SDF cell labels
  - SDF cell delays
  - SDF cell timing checks
  - SDF cell timing environment

560 © Cadence Design Systems, Inc. All rights reserved.

cādence

This section presents an example SDF file and then explains each section of the file. It individually explains the label, delay, timing check, and timing environment clauses of the cell timing data. Some of these clauses have sub-clauses, thus forming a hierarchy of cell timing data.

Example SDF File

```
(DELAYFILE
 (SDFVERSION "4.0")
 (DESIGN "system")
 (DATE "Sun Feb 22 14:10:03 EST 2009")
 (VENDOR "Cadence")
 (PROGRAM "delay_calc")
 (VERSION "08.20-p001")
 (DIVIDER /) /*hierarchical divider */
 (VOLTAGE 4.5:5.0:5.5)
 (PROCESS "worst")
 (TIMESCALE 1ns) /* delay time units */
 (CELL (CELLTYPE "system") (INSTANCE block_1) /* top-level blocks */
  (DELAY (ABSOLUTE
    (INTERCONNECT D1/z P3/i (.155::.155) (.130::.130)))))
 (CELL (CELLTYPE "INV") (INSTANCE *) /* all instances of "INV" */
  (DELAY (INCREMENT
    (IOPATH i z (.345::.348) (.325::.329)))))
 (CELL (CELLTYPE "OR2") (INSTANCE B1/C1) /* this instance of "OR2" */
  (DELAY (ABSOLUTE
    (IOPATH i1 z (.300::.300) (.325::.325))
    (IOPATH i2 z (.300::.300) (.325::.325)))))
)
```

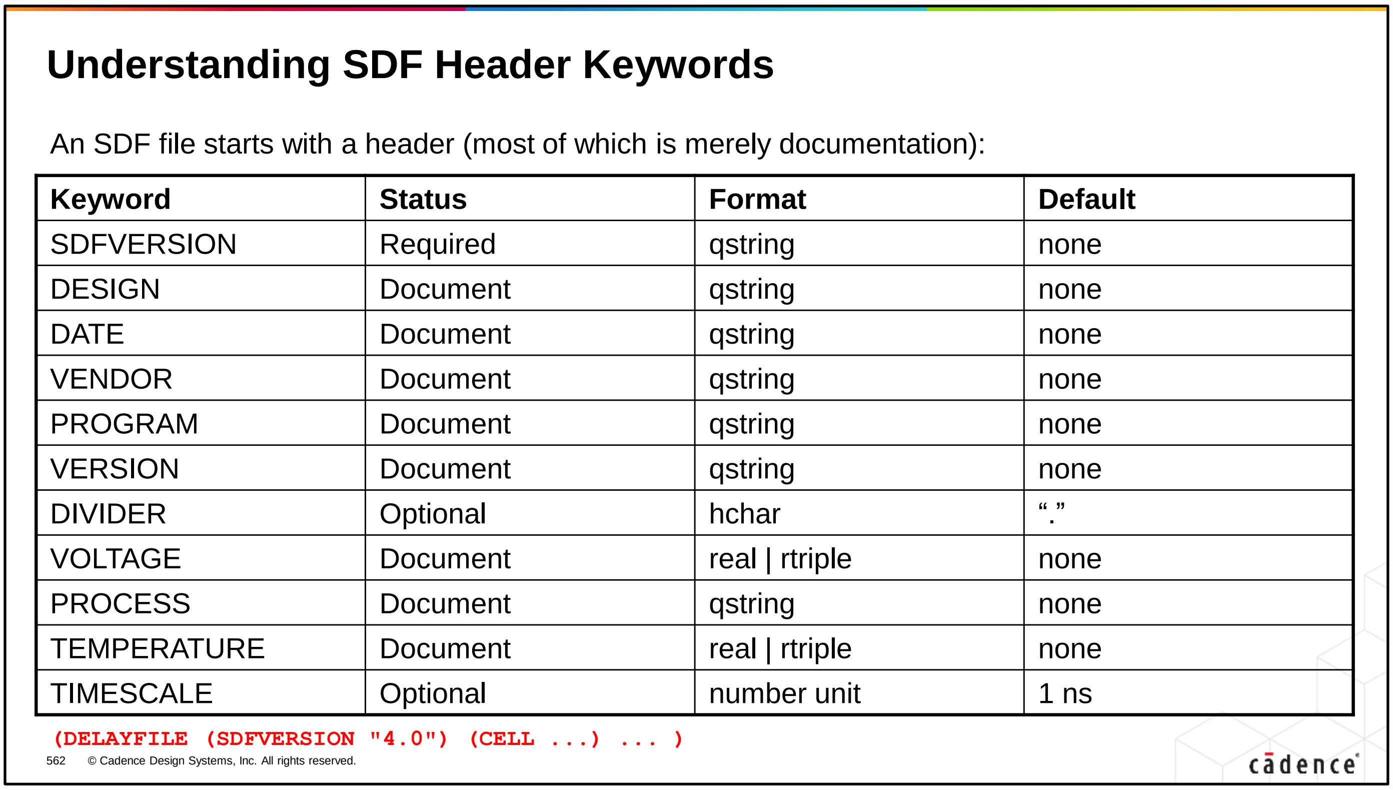
The SDF header specifies SDF file configuration information in proper order.

Cell references may be individual instances or type references.

Cell delays can be absolute or incremental, and can be conditional.

Cell delays may be single values or min:typ:max triples. If you omit a value from a delay triple, that delay will default to the delay specified in the cell timing model. The example illustrates this omission.

The SDF can also specify timing checks, which can be conditional, and timing constraints, which simulators ignore.



Understanding SDF Header Keywords

An SDF file starts with a header (most of which is merely documentation):

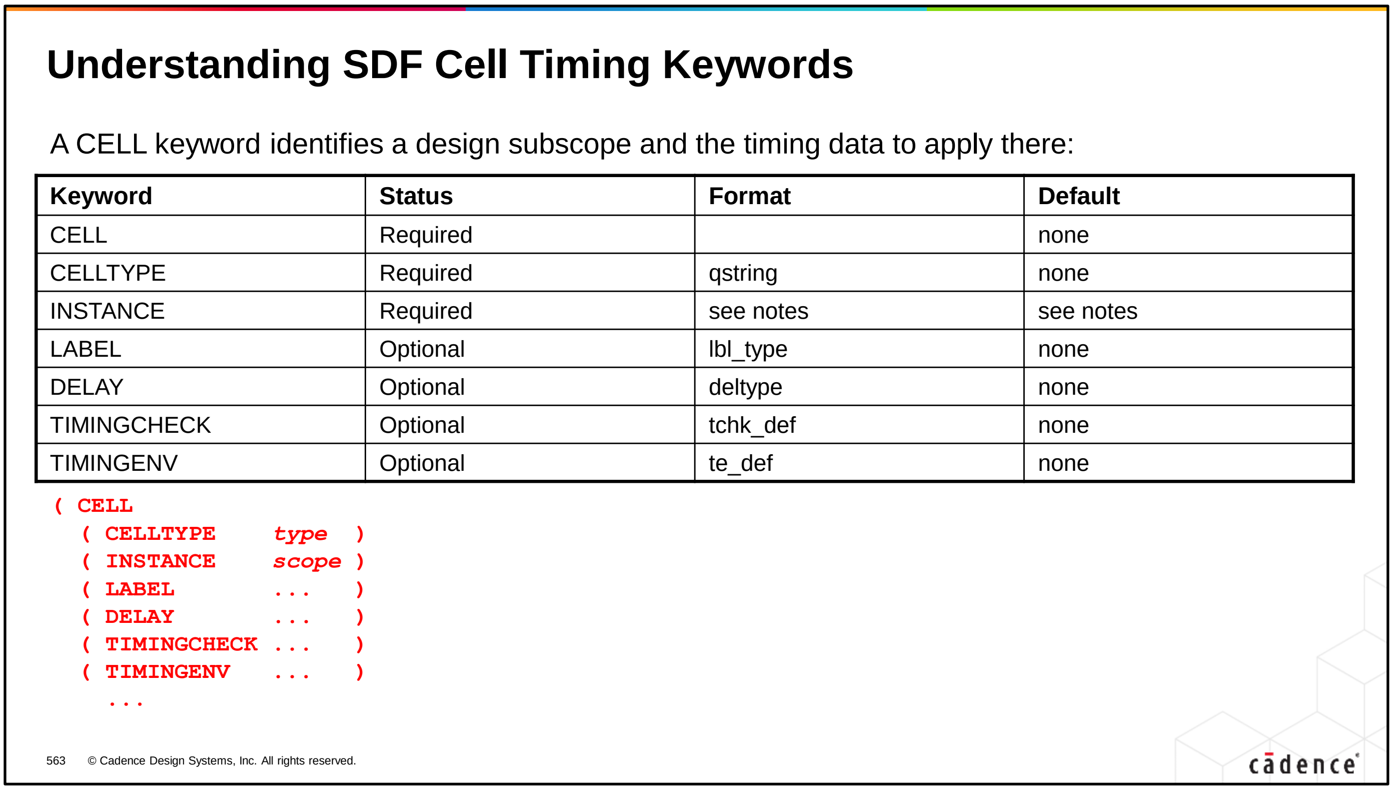
Keyword	Status	Format	Default
SDFVERSION	Required	qstring	none
DESIGN	Document	qstring	none
DATE	Document	qstring	none
VENDOR	Document	qstring	none
PROGRAM	Document	qstring	none
VERSION	Document	qstring	none
DIVIDER	Optional	hchar	“.”
VOLTAGE	Document	real | rtriple	none
PROCESS	Document	qstring	none
TEMPERATURE	Document	real | rtriple	none
TIMESCALE	Optional	number unit	1 ns

```
(DELAYFILE (SDFVERSION "4.0") (CELL ...) ... )
```

The SDFVERSION header keyword is required.

The DIVIDER defaults to the period (“.”) character and the TIMESCALE defaults to 1ns, so those header keywords are optional.

The remainder of the header is purely for documentation purposes.



Understanding SDF Cell Timing Keywords

A CELL keyword identifies a design subscope and the timing data to apply there:

Keyword	Status	Format	Default
CELL	Required		none
CELLTYPE	Required	qstring	none
INSTANCE	Required	see notes	see notes
LABEL	Optional	lbl_type	none
DELAY	Optional	deltype	none
TIMINGCHECK	Optional	tchk_def	none
TIMINGENV	Optional	te_def	none

```
( CELL
  ( CELLTYPE    type  )
  ( INSTANCE    scope )
  ( LABEL       ...   )
  ( DELAY       ...   )
  ( TIMINGCHECK ...   )
  ( TIMINGENV   ...   )
    ...
```

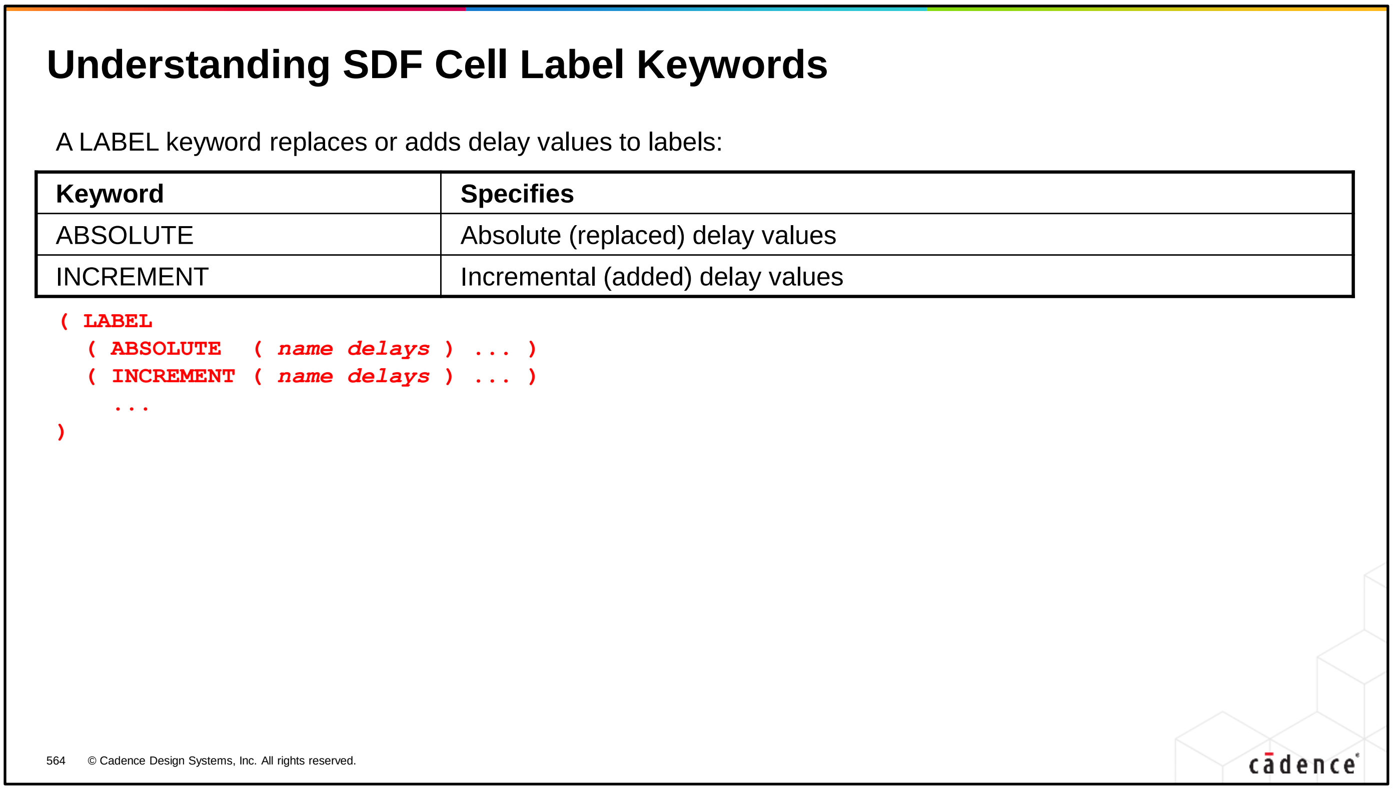
A CELL keyword identifies a design subscope and the timing data to apply there.

The CELLTYPE clause is required. Annotation tools must by default report following cell instances that do not match the specified cell type.

The INSTANCE clause is required, and either an asterisk or hierarchical identifier may follow it:

- The hierarchical identifier is relative to the scope at which you instruct the annotator to perform the annotation. Legal identifier characters are alphanumerics, the dollar sign ($), and the underscore (_). Any other identifier character will be individually escaped with a backslash (\) character. Note that this escape mechanism is not identical to the Verilog escape mechanism.
- The asterisk (*) wildcard character applies the timing data to all cells of the cell type.
- Lack of a hierarchical identifier, or existence of a wildcard, applies the timing data to the scope at which you instruct the annotator to perform the annotation.

Following slides describe the LABEL, DELAY, TIMINGCHECK, and TIMINGENV clauses.



Understanding SDF Cell Label Keywords

A LABEL keyword replaces or adds delay values to labels:

Keyword	Specifies
ABSOLUTE	Absolute (replaced) delay values
INCREMENT	Incremental (added) delay values

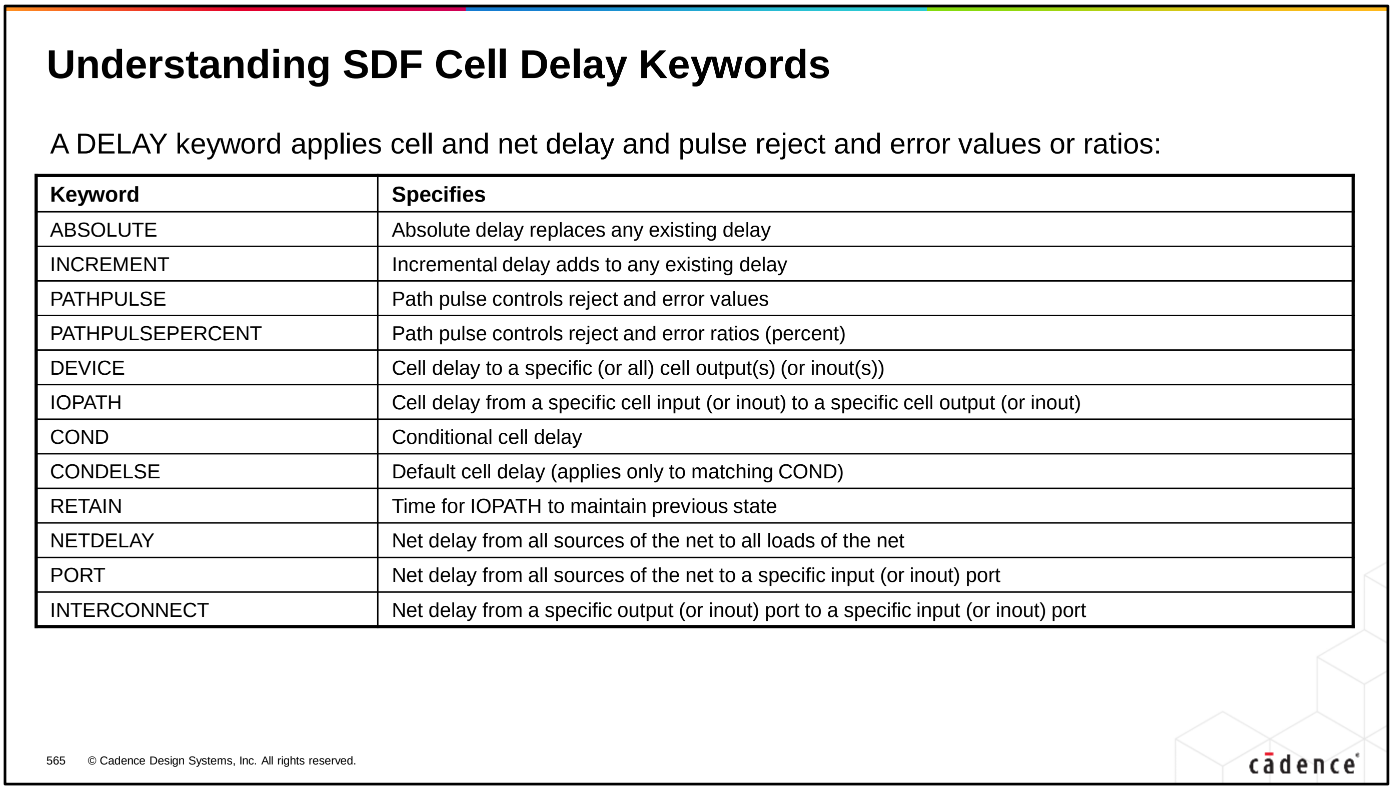
```
( LABEL
  ( ABSOLUTE  ( name delays ) ... )
  ( INCREMENT ( name delays ) ... )
    ...
)
```

A LABEL clause replaces or adds delay values to labels. A label is a Verilog specify block parameter (specparam).

- The ABSOLUTE sub-clause specifies a new value for the label.
- The INCREMENT sub-clause specifies a value to add to the previous value of the label.

You can specify 1, 2, 3, 6, or 12 delay values. The delay values apply to the twelve edges of the four-value (01ZX) logic system, with varying degrees of granularity. Each delay value may be a single real number or a triple of real numbers specifying unique minimum, typical, and maximum values. You may omit up to two of the numbers in a triple.

Later slides provide examples of delay values.



Understanding SDF Cell Delay Keywords

A DELAY keyword applies cell and net delay and pulse reject and error values or ratios:

Keyword	Specifies
ABSOLUTE	Absolute delay replaces any existing delay
INCREMENT	Incremental delay adds to any existing delay
PATHPULSE	Path pulse controls reject and error values
PATHPULSEPERCENT	Path pulse controls reject and error ratios (percent)
DEVICE	Cell delay to a specific (or all) cell output(s) (or inout(s))
IOPATH	Cell delay from a specific cell input (or inout) to a specific cell output (or inout)
COND	Conditional cell delay
CONDELSE	Default cell delay (applies only to matching COND)
RETAIN	Time for IOPATH to maintain previous state
NETDELAY	Net delay from all sources of the net to all loads of the net
PORT	Net delay from all sources of the net to a specific input (or inout) port
INTERCONNECT	Net delay from a specific output (or inout) port to a specific input (or inout) port

A DELAY clause applies cell and net delay and pulse reject and error values or ratios.

- The ABSOLUTE sub-clause specifies a new value for the delay.
- The INCREMENT sub-clause specifies a value to add to the previous value of the delay.
- The PATHPULSE sub-clause specifies cell path pulse rejection, and optionally error, limits. The annotator by default sets the pulse rejection limit to the delay. A pulse having a duration lower than this limit is rejected. This is known as an “inertial” delay model. You can set the rejection limit lower, and you can optionally set an error limit that is between the rejection limit and the delay. A pulse having a duration of at least the rejection limit, but lower than the error limit, filters the transition to the unknown value at the output port. A rejection and error limit of zero passes all pulses regardless of width. This is known as a “transport” delay model.
- The PATHPULSEPERCENT sub-clause specifies cell path pulse rejection, and optionally error, limits as a percent of the delay. This is useful for providing estimated path pulse controls on a global basis that you can later override on a per-cell basis for individual cells.

You annotate cell delay with either the DEVICE or the IOPATH sub-clause:

- The DEVICE clause specifies the intrinsic delay of a cell, gate, or port instance.
- The IOPATH clause specifies the delay from an instance input to an instance output. You can make IOPATH delays conditional upon the state of some other input port, and for IOPATH delays you can specify a data retention time that is less than the delay. The output port transitions to the unknown state for the duration between the retention time and the delay time.

You annotate interconnect delay with either the NETDELAY, PORT or INTERCONNECT sub-clause:

- The NETDELAY clause specifies delay for all paths on the net. This clause is seldom used.
- The PORT clause specifies delay for all paths terminating at a specified cell input port.
- The INTERCONNECT clause specifies port-to-port interconnect delay.

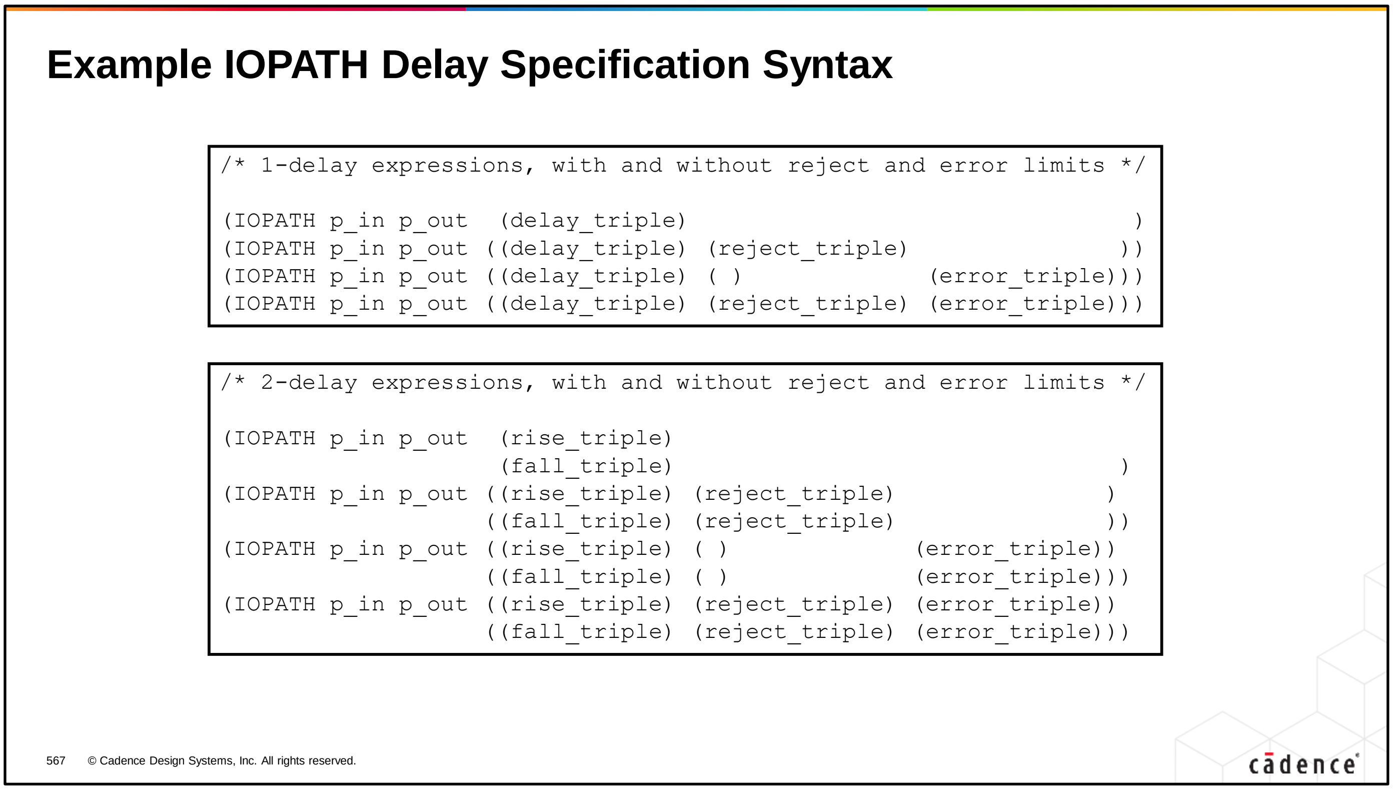
Annotating Cell Delay

In the SDF file, annotate cell delay with:

- The DEVICE keyword to specify the intrinsic delay of a cell, gate, or port instance:
  - This specifies min:typ:max delay for rise, fall, and turn off edges:
    ```
    ( DEVICE U1   (7:8:9) (1:2:3) (4:5:6)   )
    ```
  - This specifies min:typ:max delay, reject, and error for all edges:
    ```
    ( DEVICE U1 ( (7:8:9) (1:2:3) (4:5:6) ) )
    ```
- The IOPATH keyword to specify the delay from an instance input to an instance output:
  - This specifies min:typ:max delay for rise, fall, and turn off edges:
    ```
    ( IOPATH INA OUTA   (7:8:9) (1:2:3) (4:5:6)   )
    ```
  - This specifies min:typ:max delay, reject, and error for all edges:
    ```
    ( IOPATH INA OUTA ( (7:8:9) (1:2:3) (4:5:6) ) )
    ```

You annotate cell delay with either the DEVICE clause or the IOPATH clause:

- The DEVICE clause specifies the intrinsic delay of a cell, gate, or port instance.
  - The 1st short example specifies min:typ:max intrinsic delay for rise, fall, and turn off edges.
  - The 2nd short example specifies min:typ:max intrinsic delay and reject and error limits for all edges. Due to an additional set of parentheses, this is actually only one delay value, so applies to all edges. The three triples are the delay and the reject and error limits.
- The IOPATH clause specifies the delay from an instance input to an instance output.
  - The 3rd short example specifies min:typ:max I/O path delay for rise, fall, and turn off edges.
  - The 4th short example specifies min:typ:max I/O path delay and reject and error limits for all edges.



Example IOPATH Delay Specification Syntax

```
/* 1-delay expressions, with and without reject and error limits */

(IOPATH p_in p_out  (delay_triple)                                  )
(IOPATH p_in p_out ((delay_triple) (reject_triple)                 ))
(IOPATH p_in p_out ((delay_triple) ( )             (error_triple)))
(IOPATH p_in p_out ((delay_triple) (reject_triple) (error_triple)))
```

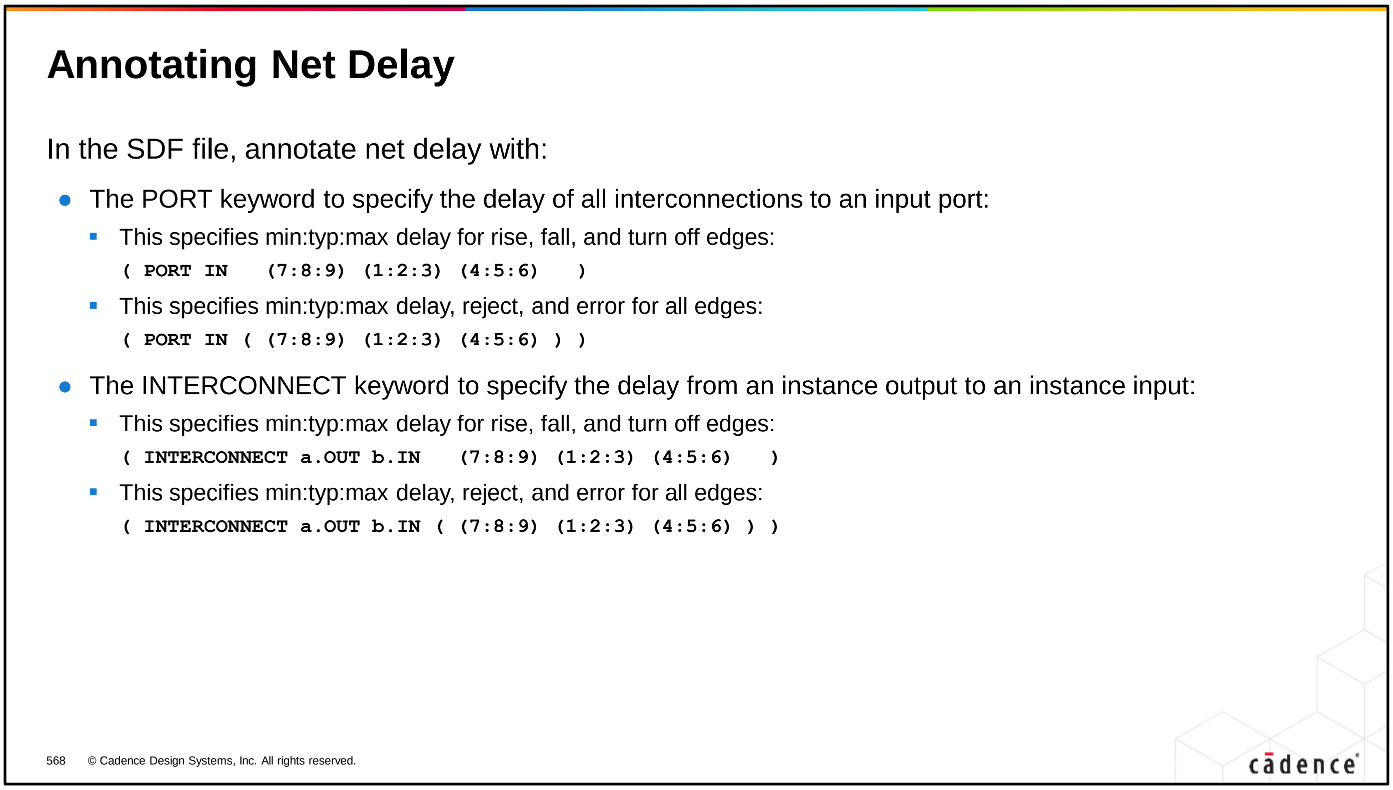
```
/* 2-delay expressions, with and without reject and error limits */

(IOPATH p_in p_out  (rise_triple)
                    (fall_triple)                                 )
(IOPATH p_in p_out ((rise_triple) (reject_triple)               )
                   ((fall_triple) (reject_triple)               ))
(IOPATH p_in p_out ((rise_triple) ( )             (error_triple))
                   ((fall_triple) ( )             (error_triple)))
(IOPATH p_in p_out ((rise_triple) (reject_triple) (error_triple))
                   ((fall_triple) (reject_triple) (error_triple)))
```

567 © Cadence Design Systems, Inc. All rights reserved.

Here is some example IOPATH delay specification syntax:

- The first is of one-delay expressions, with and without reject and error limits. As these are all just single delay values, they apply to all transitions. Note that if you want to annotate the error limit without annotating the reject limit, then you need to use a placeholder for the reject limit.
- The second is of two-delay expressions, again with and without reject and error limits. The rise delay applies to all potentially positive transitions and the fall delay applies to all potentially negative transitions. Other transitions use the worst-case delays. That is, transitions from unknown to high impedance use the maximum of the two delays and transitions from high impedance to unknown use the minimum of the two delays.



Annotating Net Delay

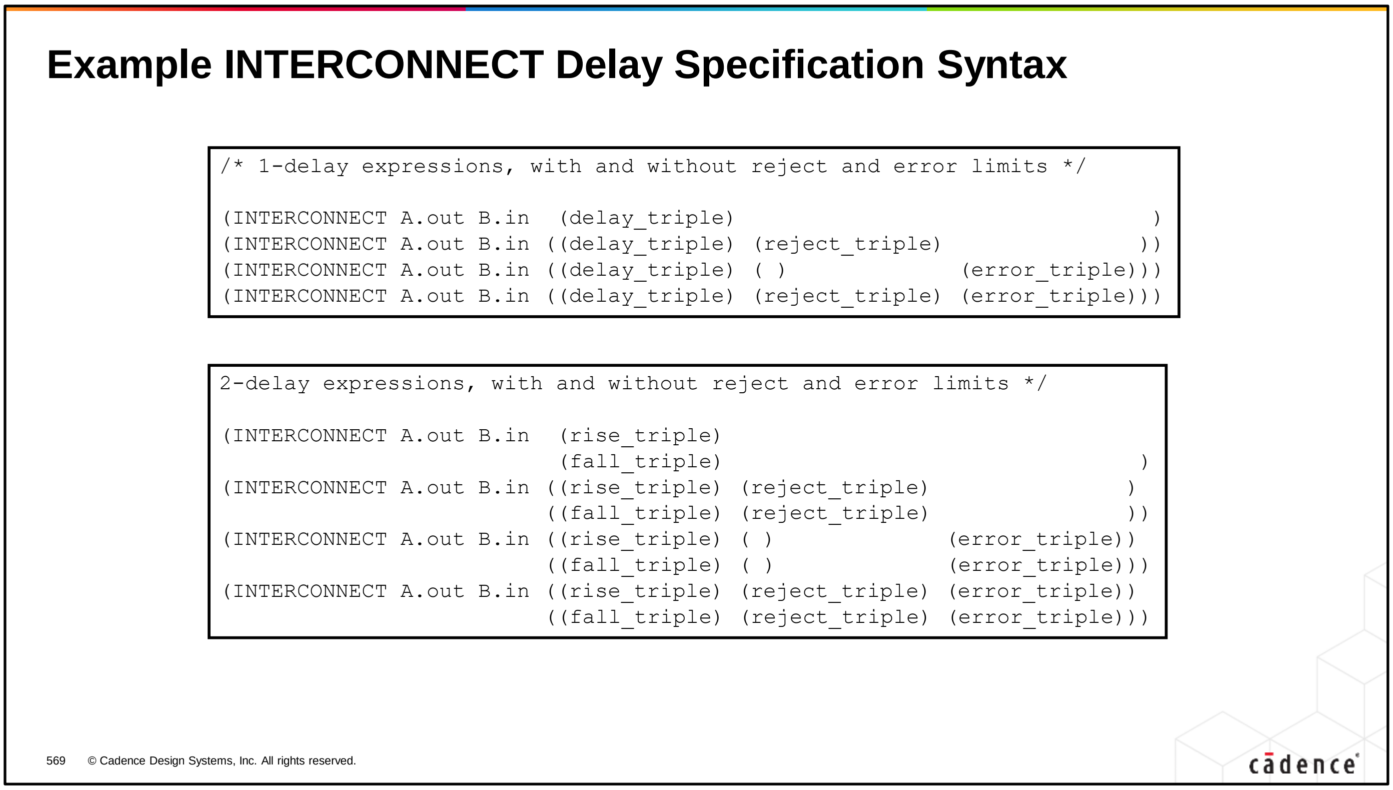
In the SDF file, annotate net delay with:

- The PORT keyword to specify the delay of all interconnections to an input port:
  - This specifies min:typ:max delay for rise, fall, and turn off edges:
    ```
    ( PORT IN   (7:8:9) (1:2:3) (4:5:6)   )
    ```
  - This specifies min:typ:max delay, reject, and error for all edges:
    ```
    ( PORT IN ( (7:8:9) (1:2:3) (4:5:6) ) )
    ```
- The INTERCONNECT keyword to specify the delay from an instance output to an instance input:
  - This specifies min:typ:max delay for rise, fall, and turn off edges:
    ```
    ( INTERCONNECT a.OUT b.IN   (7:8:9) (1:2:3) (4:5:6)   )
    ```
  - This specifies min:typ:max delay, reject, and error for all edges:
    ```
    ( INTERCONNECT a.OUT b.IN ( (7:8:9) (1:2:3) (4:5:6) ) )
    ```

568 © Cadence Design Systems, Inc. All rights reserved.

You annotate interconnect delay with either the PORT clause or the INTERCONNECT clause:

- The PORT clause specifies the delay of all interconnections to an input port.
  - The 1st short example specifies min:typ:max delay for rise, fall, and turn off edges.
  - The 2nd short example specifies min:typ:max delay, and reject and error limits, for all edges. Due to an additional set of parentheses, this is actually only one delay value, so applies to all edges. The three triples are the delay and the reject and error limits.
- The INTERCONNECT clause specifies the delay from an instance output to an instance input.
  - The 3rd short example specifies min:typ:max delay for rise, fall, and turn off edges.
  - The 4th short example specifies min:typ:max delay, and reject and error limits, for all edges.



Example INTERCONNECT Delay Specification Syntax

```
/* 1-delay expressions, with and without reject and error limits */

(INTERCONNECT A.out B.in  (delay_triple)                                 )
(INTERCONNECT A.out B.in ((delay_triple) (reject_triple)                ))
(INTERCONNECT A.out B.in ((delay_triple) ( )             (error_triple)))
(INTERCONNECT A.out B.in ((delay_triple) (reject_triple) (error_triple)))
```

```
2-delay expressions, with and without reject and error limits */

(INTERCONNECT A.out B.in  (rise_triple)
                          (fall_triple)                                 )
(INTERCONNECT A.out B.in ((rise_triple) (reject_triple)                )
                         ((fall_triple) (reject_triple)                ))
(INTERCONNECT A.out B.in ((rise_triple) ( )             (error_triple))
                         ((fall_triple) ( )             (error_triple)))
(INTERCONNECT A.out B.in ((rise_triple) (reject_triple) (error_triple))
                         ((fall_triple) (reject_triple) (error_triple)))
```

Here is some example INTERCONNECT delay specification syntax:

- The first is of one-delay expressions, with and without reject and error limits. As these are all just single delay values, they apply to all transitions. Note that if you want to annotate the error limit without annotating the reject limit, then you need to use a placeholder for the reject limit.
- The second is of two-delay expressions, again with and without reject and error limits. The rise delay applies to all potentially positive transitions and the fall delay applies to all potentially negative transitions. Other transitions use the worst-case delays. That is, transitions from unknown to high impedance use the maximum of the two delays and transitions from high impedance to unknown use the minimum of the two delays.

Understanding SDF Cell Timing Check Keywords

A TIMINGCHECK keyword specifies minimum or maximum limits for the time between transitions of two signals or two transitions of the same signal:

Keyword	Stamp	Check	Value	Limit
SETUP	port 1	port 2	Positive	Min
HOLD	port 2	port 1	Positive	Min
SETUPHOLD	port 1,2	port 2,1	Signed	Min
RECOVERY	port 1	port 2	Positive	Min
REMOVAL	port 2	port 1	Positive	Min
RECREM	port 1,2	port 2,1	Signed	Min
SKEW	port 1,2	port 2,1	Signed	Max
BIDIRECTSKEW	port 1,2	port 2,1	Positive	Min
WIDTH	port	port	Positive	Min
PERIOD	port	port	Positive	Min
NOCHANGE	port 2,1	port 1,2	Signed	Min

A TIMINGCHECK clause specifies minimum or maximum limits for the time between transitions of two signals or two transitions of the same signal. All of these checks do approximately the same thing – they record the time of a transition, and then at a later transition of the same or a different signal, report any violation of the limit.

Only the SETUPHOLD, RECREM, SKEW, and NOCHANGE checks accept negative values. The SETUPHOLD check is a combination of SETUP and HOLD checks where one or the other check may use a negative value, provided that the sum of the two numbers is not less than zero. Likewise, the RECREM check is a combination of RECOVERY and REMOVAL checks.

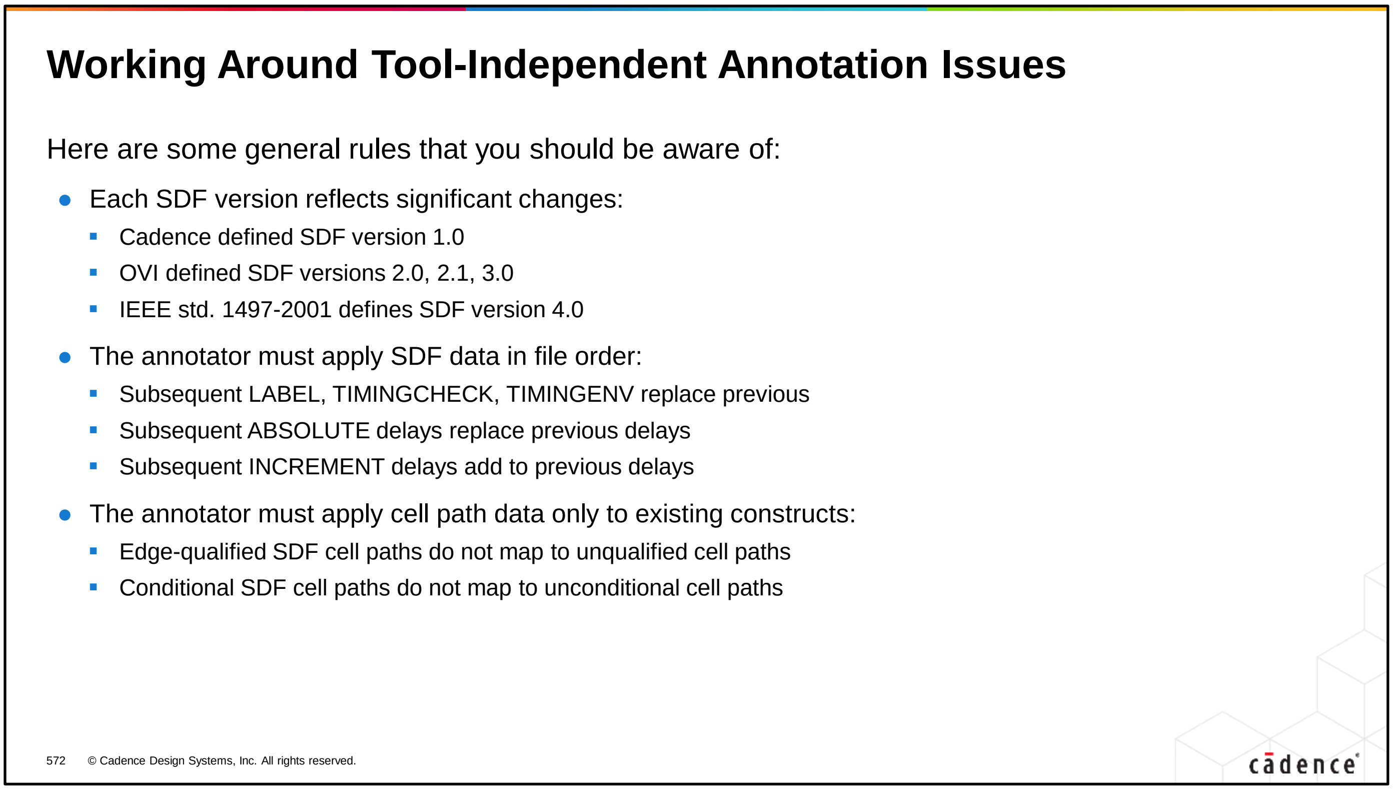
The SKEW check has a maximum limit and all other checks have minimum limits.

Understanding SDF Cell Timing Environment Keywords

A TIMINGENV keyword associates constraint values with critical paths in the design and provides information about the timing environment in which the circuit will operate. Constructs in this subclause are used for forward annotation to design implementation tools.

Keyword	What It Specifies
PATHCONSTRAINT	Max path delay
PERIODCONSTRAINT	Max clock period
SUM	Max sum of multiple path delays
DIFF	Max difference between two path delays
SKEWCONSTRAINT	Max clock skew
ARRIVAL	Input port arrival time
DEPARTURE	Output port departure time
SLACK	Input port available slack
WAVEFORM	Clock waveform

A TIMINGENV clause associates constraint values with critical paths in the design and provides information about the timing environment in which the circuit will operate. Constructs in this subclause are used for forward annotation to design implementation tools. As they are not used for simulation, you will seldom see them.



Working Around Tool-Independent Annotation Issues

Here are some general rules that you should be aware of:

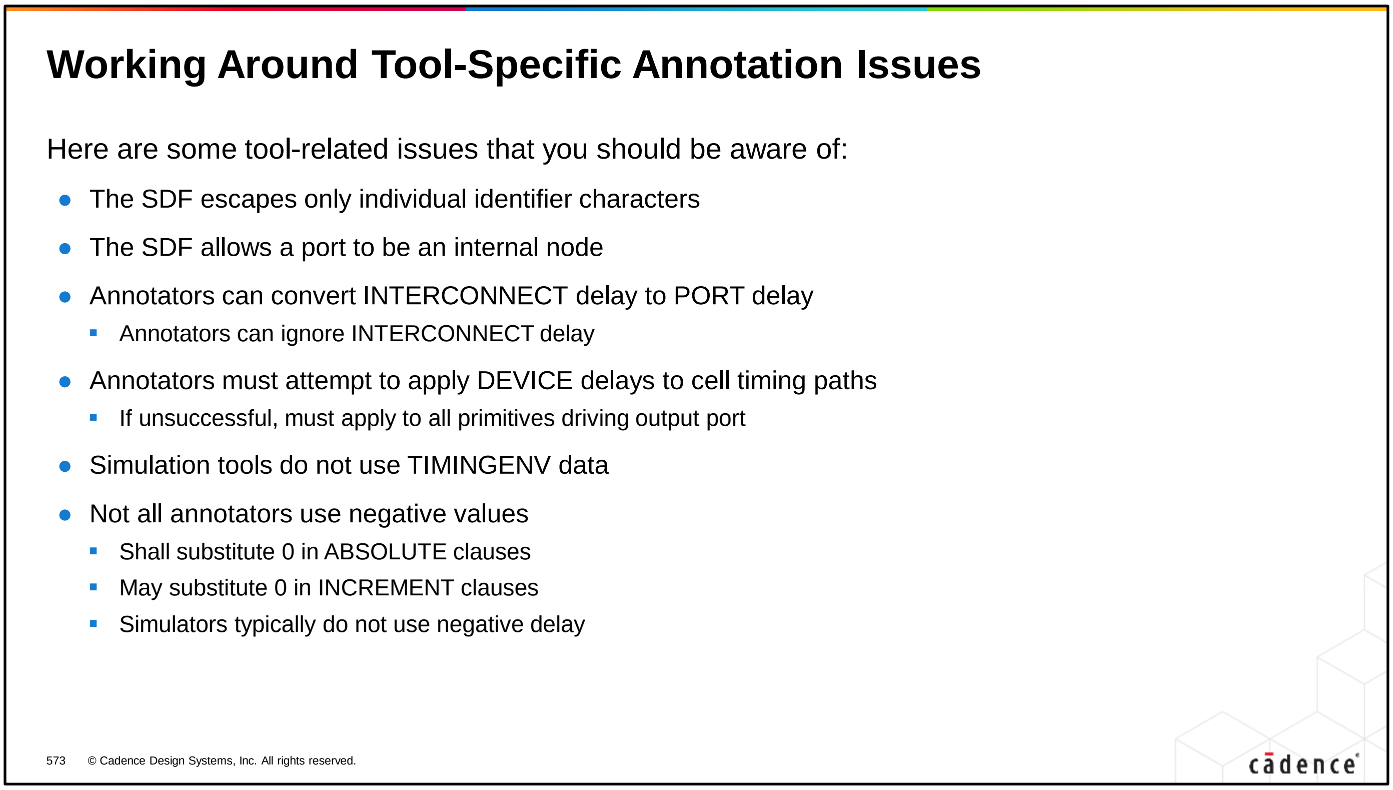
- Each SDF version reflects significant changes:
  - Cadence defined SDF version 1.0
  - OVI defined SDF versions 2.0, 2.1, 3.0
  - IEEE std. 1497-2001 defines SDF version 4.0
- The annotator must apply SDF data in file order:
  - Subsequent LABEL, TIMINGCHECK, TIMINGENV replace previous
  - Subsequent ABSOLUTE delays replace previous delays
  - Subsequent INCREMENT delays add to previous delays
- The annotator must apply cell path data only to existing constructs:
  - Edge-qualified SDF cell paths do not map to unqualified cell paths
  - Conditional SDF cell paths do not map to unconditional cell paths

572 © Cadence Design Systems, Inc. All rights reserved.

The IEEE Std. 1497-2001 “SDF for the Electronic Design Process” defines the SDF version 4.0 and requires the SDF file to specify the SDF version. Open Verilog International (OVI) defined previous SDF versions and permits the annotator to by default assume the SDF version 1.0. Each SDF version contains significant differences that this training material does not re-document.

The annotator must apply SDF timing data in file order. Subsequent data targeting a previously annotated construct replaces the previously annotated data with one exception. This exception is that incremental delay data adds to the previously annotated data.

The annotator must apply cell path timing data only to existing constructs, and cannot only partially annotate the existing construct. You can annotate unqualified or unconditional data to qualified or conditional paths, but cannot annotate qualified or conditional data to unqualified or unconditional paths.



Working Around Tool-Specific Annotation Issues

Here are some tool-related issues that you should be aware of:

- The SDF escapes only individual identifier characters
- The SDF allows a port to be an internal node
- Annotators can convert INTERCONNECT delay to PORT delay
  - Annotators can ignore INTERCONNECT delay
- Annotators must attempt to apply DEVICE delays to cell timing paths
  - If unsuccessful, must apply to all primitives driving output port
- Simulation tools do not use TIMINGENV data
- Not all annotators use negative values
  - Shall substitute 0 in ABSOLUTE clauses
  - May substitute 0 in INCREMENT clauses
  - Simulators typically do not use negative delay

573 © Cadence Design Systems, Inc. All rights reserved.

cādence

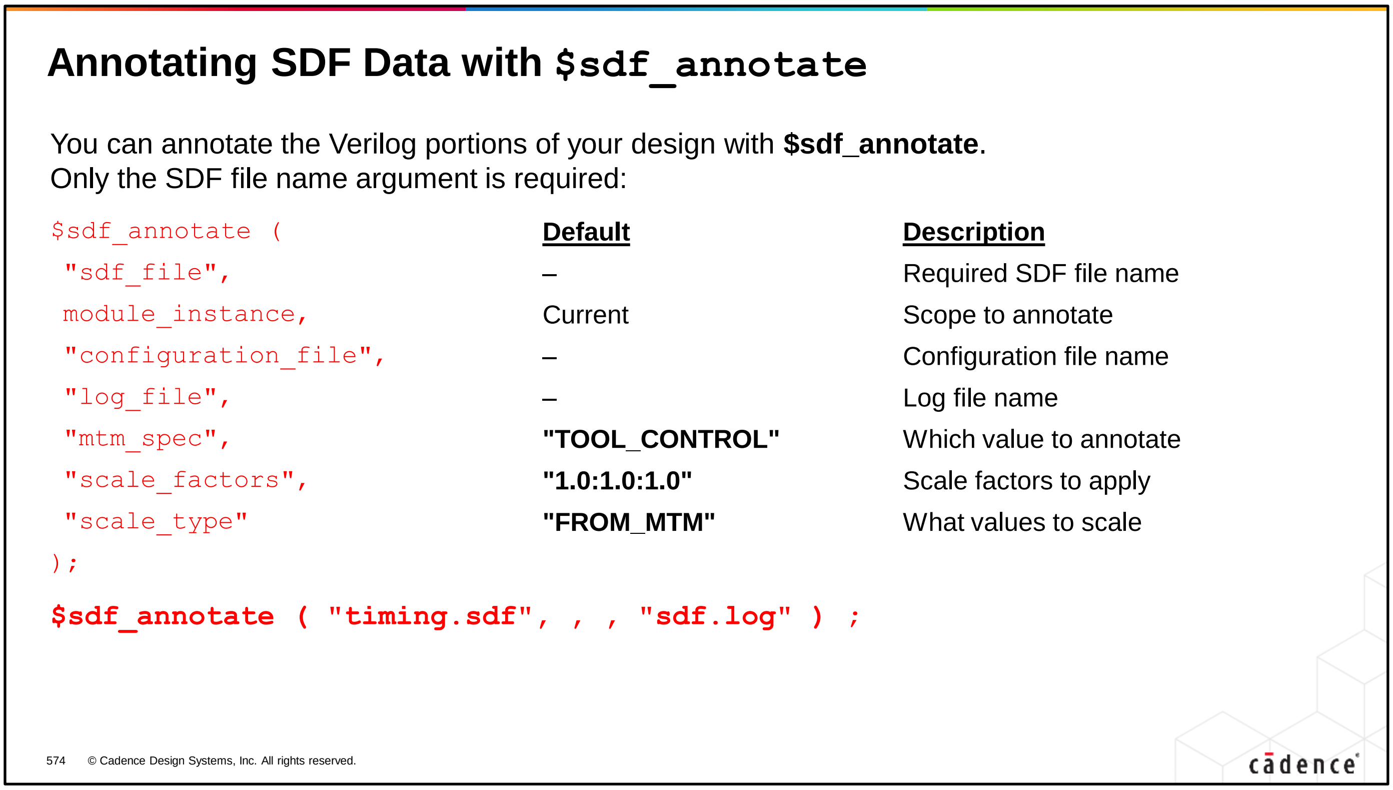
The SDF escapes individual illegal identifier characters with a preceding backslash (“\”) character. Other EDA tools escape the entire identifier. Verilog escapes otherwise illegal identifiers with a leading backslash character and a trailing whitespace (space, tab, or newline) character.

The SDF allows a port used for timing purposes to actually be an internal node that is not a connector at a design unit boundary. SDF files referencing such internal nodes might not be portable between EDA tools.

Annotators unable to annotate individual source-to-load interconnect pairs may choose to convert INTERCONNECT delay to PORT delay, and may also choose to simply ignore INTERCONNECT delay.

Annotators must attempt to apply DEVICE delays to cell timing paths from all input or inout ports to a specified or all output or inout port(s). If no such path exists for a given applicable output, the annotator must apply the delay to all primitives driving that output from any level in the cell hierarchy.

The TIMINGENV keyword provides timing environment and constraint data for forward annotation to design implementation tools. Simulators do not use those constraints.



Annotating SDF Data with `$sdf_annotate`

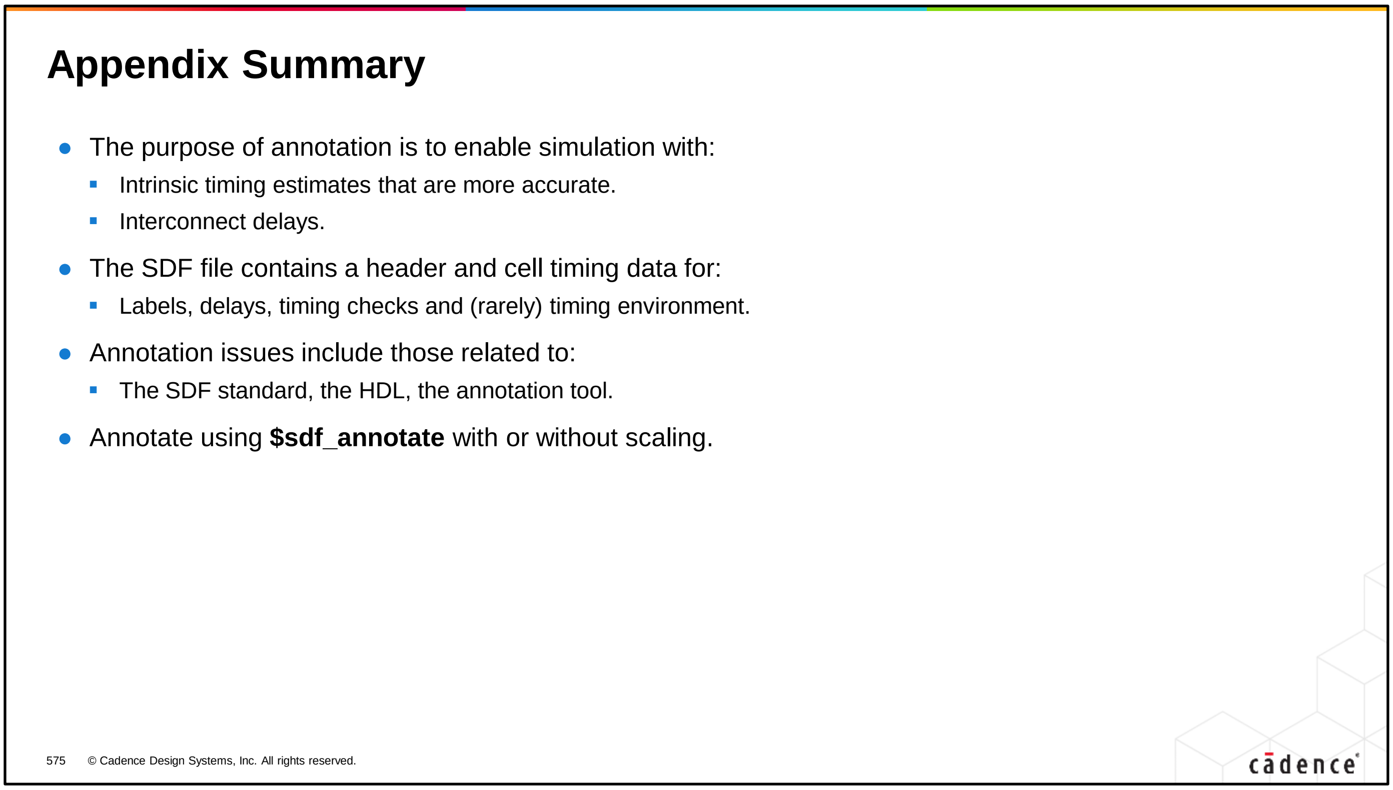
You can annotate the Verilog portions of your design with **$sdf_annotate**.
Only the SDF file name argument is required:

`$sdf_annotate (`	**Default**	**Description**
`"sdf_file",`	–	Required SDF file name
`module_instance,`	Current	Scope to annotate
`"configuration_file",`	–	Configuration file name
`"log_file",`	–	Log file name
`"mtm_spec",`	**"TOOL_CONTROL"**	Which value to annotate
`"scale_factors",`	**"1.0:1.0:1.0"**	Scale factors to apply
`"scale_type"`	**"FROM_MTM"**	What values to scale
`);`		

```
$sdf_annotate ( "timing.sdf", , , "sdf.log" ) ;
```

You can annotate the Verilog portions of your design with the $sdf_annotate built-in system task. Only the SDF file name argument is required. To provide later arguments, you need to at least provide commas as argument placeholders. Arguments you provide here override those in the configuration file.

- The sdf_file argument is the SDF file name as a character string literal or a register vector containing the equivalent ASCII string value.
- The module_instance optional argument specifies the scope in which to annotate the SDF information. The simulator by default annotates the scope of the $sdf_annotate system task call.
- The configuration_file optional argument is the configuration file name. With this vendor-defined file you can presumably more precisely control the annotation. Neither the IEEE Std 1364-2001 Verilog nor the IEEE Std 1497-2001 SDF defines this file.
- The log_file optional argument is the log file name.
- The scale_type optional argument specifies which SDF file timing values to use as the basis for scaling and subsequent annotation. The “FROM_MTM” value means to use the minimum, typical and maximum values. You can alternatively select “FROM_MINIMUM”, “FROM_TYPICAL” or “FROM_MAXIMUM”.
- The scale_factors optional argument specifies the scale factors for generating new minimum, typical, and maximum timing values. The default is to not scale them.
- The mtm_spec optional argument specifies which new timing value to annotate. The default is to annotate the values selected upon simulator invocation, but you can alternatively annotate just the “MINIMUM”, “TYPICAL” or “MAXIMUM” values.



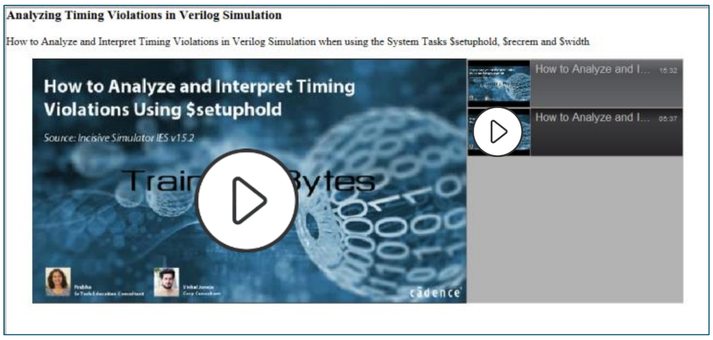
Appendix Summary

- The purpose of annotation is to enable simulation with:
  - Intrinsic timing estimates that are more accurate.
  - Interconnect delays.
- The SDF file contains a header and cell timing data for:
  - Labels, delays, timing checks and (rarely) timing environment.
- Annotation issues include those related to:
  - The SDF standard, the HDL, the annotation tool.
- Annotate using **$sdf_annotate** with or without scaling.

This module communicated to you the knowledge you need to annotate timing data to your Verilog designs and to modify the SDF file for debugging purposes.

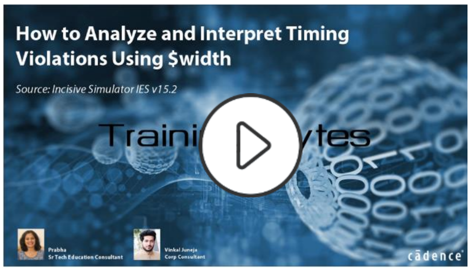
This module briefly described the purpose of annotation, explained some of the SDF keywords, discussed some issues you may encounter, and demonstrated the annotation process.

Videos on SDF Annotation in COS


Analyzing Timing Violations in Verilog Simulation

How to Analyze and Interpret Timing Violations in Verilog Simulation when using the System Tasks $setuphold, $recrem and $width

How to Analyze and Interpret Timing Violations Using $setuphold
Source: Incisive Simulator IES v15.2
How to Analyze and I... 15:32
How to Analyze and I... 05:37


How to Analyze and Interpret Timing Violations Using $width
Source: Incisive Simulator IES v15.2

There are 3 videos posted in support.cadence.com (COS) related to sdf annotation.

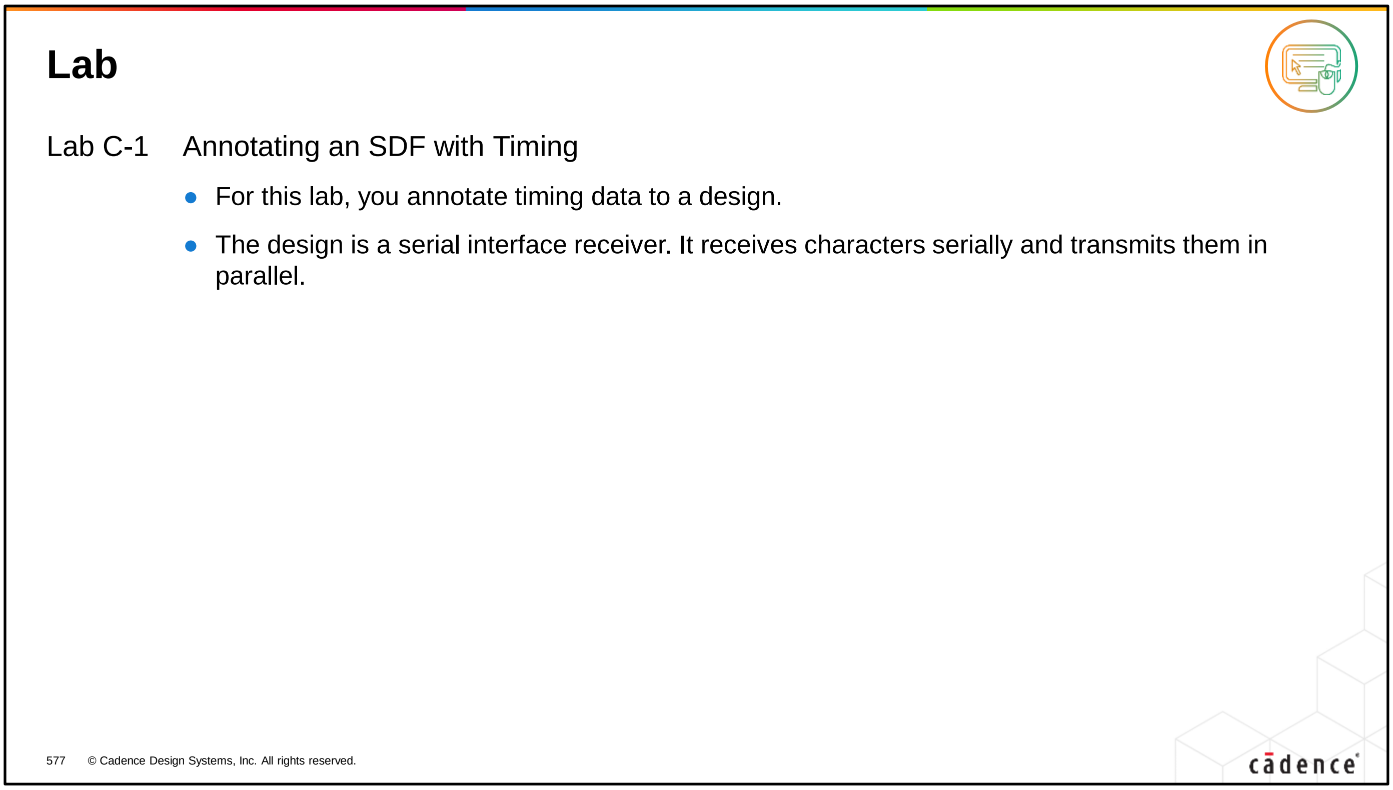
“How to Analyze and Interpret Timing Violations in Verilog Simulation when using System Tasks $setuphold, $recrem and $width.”

These videos explain simple real issues as examples using waveforms.

How to Analyze and Interpret Timing Violations Using $setuphold

How to Analyze and Interpret Timing Violations Using $width

How to Analyze and Interpret Timing Violations Using $recrem



Lab

Lab C-1 Annotating an SDF with Timing

- For this lab, you annotate timing data to a design.
- The design is a serial interface receiver. It receives characters serially and transmits them in parallel.

577 © Cadence Design Systems, Inc. All rights reserved.

cādence

Your objective for this lab is to annotate timing information.

For this lab, you annotate timing data to a design.