This document is for the sole use of Le Khanh Nam of Gmail

Basic Static Timing Analysis

Course Version 3.0

Lecture Manual **Revision 1.0**

cadence®

This document is for the sole use of Le Khanh Nam of Gmail

© 1990-2023 Cadence Design Systems, Inc. All rights reserved.

Printed in the United States of America.

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

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence trademarks, contact the corporate legal department at the address shown above or call 1-800-862-4522.

All other trademarks are the property of their respective holders.

Restricted Print Permission: This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

- The publication may be used solely for personal, informational, and noncommercial purposes;
- The publication may not be modified in any way;
- Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
- Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence customers in accordance with, a written agreement between Cadence and the customer.

Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

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

This document is for the sole use of Le Khanh Nam of Gmail

Table of Contents

Basic Static Timing Analysis

Module 1 About This Course 2

Module 2 Introduction to Static Timing Analysis 9

Module 3 Cell Delay 29

Module 4 Net Delay 38

Module 5 Clocks 51

Module 6 Timing Checks 68

Module 7 Timing Paths 86

Module 8 Introduction to SDC Constraints 102

Module 9 Analyzing a Timing Report 159

Lab 9-1 Setting Up the Database

Lab 9-2 Creating Constraints

Lab 9-3 Analyzing the Timing in a Design

Module 10 Next Steps 173

This document is for the sole use of Le Khanh Nam of Gmail

Basic Static Timing Analysis

Version 3.0

Revision **1.0**

Estimated time: **1 day**

cadence®

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module 1

About This Course

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Course Prerequisite

Before taking this course, you need to

- Have knowledge of ASIC design methodology

 © Cadence Design Systems, Inc. All rights reserved.

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Course Objectives

In this course, you

- Identify and apply timing arc information from a library, such as unateness, delays, and slew
- Identify cell delays from a library and calculate output slew degradation
- Use wireload information to calculate net delays
- Identify the properties of a clock, including period, edges, and slew, and calculate the duty cycle
- Apply setup and hold checks to diagnose design violations
- Identify timing path types to calculate slack values
- Set environmental constraints, clocks constraints, and path exceptions
- Constrain a design using SDC
- Analyze reports to identify timing problems

 © Cadence Design Systems, Inc. All rights reserved.

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Course Agenda

- Introduction to Static Timing Analysis
- Cell Delay
- Net Delay
- Clocks
- Timing Clocks
- Timing Paths
- Introduction to SDC Constraints
- Analyzing a Timing Report
  - Setting Up the Database
  - Creating Constraints
  - Analyzing the Timing in a Design

© Cadence Design Systems, Inc. All rights reserved.

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Software and Licenses

For the software and licenses used in the labs for this course, go to:

https://www.cadence.com/en_US/home/training/all-courses/84452.html

If there is additional information regarding the specific software, it is detailed in the lab document and/or the README file of the database provided with this course.

 © Cadence Design Systems, Inc. All rights reserved.

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Become Cadence Certified by Earning a Digital Badge

Digital badges indicate mastery in a certain technology or skill and give managers and potential employers a way to validate your expertise.

- Cadence Training Services offers digital badges for our popular training courses.
- Your digital badge can be added to your email signature or social media platforms like LinkedIn or Facebook.

Benefits of Cadence Certified Digital Badges

- Validate expertise
  - Expand career opportunities
- Professional credibility
  - Stand apart from your peers
- For more information, go to www.cadence.com/training or email es_digitalbadge@cadence.com.

How do I register to take the exam?

- Log in to our Learning Management System, click on the course in your transcript, and go to the Content tab to locate the exam.

How long will it take to complete the exam?

- Most exams take 45 to 90 minutes to complete. You may retake the exam multiple times to pass the exam.

How do I access and use the digital badge?

- After you pass the exam, you get a digital badge and instructions on how to place it on social media sites.

How is the digital badge validated?

- Credly validates the digital badge as issued to you by Cadence and includes the details of the criteria you completed to earn the badge.

© Cadence Design Systems, Inc. All rights reserved.

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Icons Used in This Class

Best Practice

Language/
Command Syntax

Concept/
Glossary

Frequently Asked
Questions/
Quiz

Error Message

Problem & Solution

Quick Reference

GUI and Command

How To

Lab List

Throughout this class, we use icons to draw your attention to certain kinds of information. Here are the icons we use, and what they mean.

© Cadence Design Systems, Inc. All rights reserved.

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module 2

Introduction to Static Timing Analysis

cadence

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objectives

In this module, you

- Identify where static timing analysis (STA) fits into the flow
- Identify timing library information for timing arcs in a design, such as:
  - Unateness, cell delays and net delays, and slew

 © Cadence Design Systems, Inc. All rights reserved.

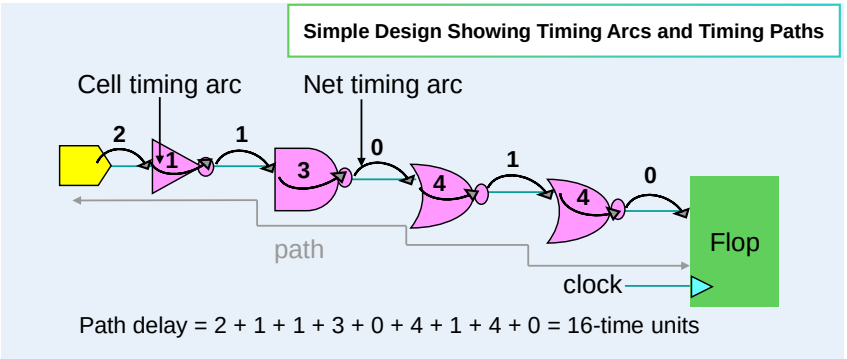
This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is Static Timing Analysis (STA)?

Static timing analysis (STA) is the process of adding the cell and net delays to obtain the path delays and comparing the path delays to the timing specification.



 © Cadence Design Systems, Inc. All rights reserved.

When you design a circuit, you need to make sure that it's working correctly. There are multiple ways of doing that. You can run a functional simulation, but that is only checking to see if your circuit will function correctly. You can run a SPICE Analysis, but that takes too long for each path because you have to time millions of paths. So, in Static Timing Analysis, we look at the circuit devoid of any time frames. And so, simply just add the cell delays and the net delays from the libraries or elsewhere, come up with the path delays, and then compare these path delays to a timing specification as given by your constraints.

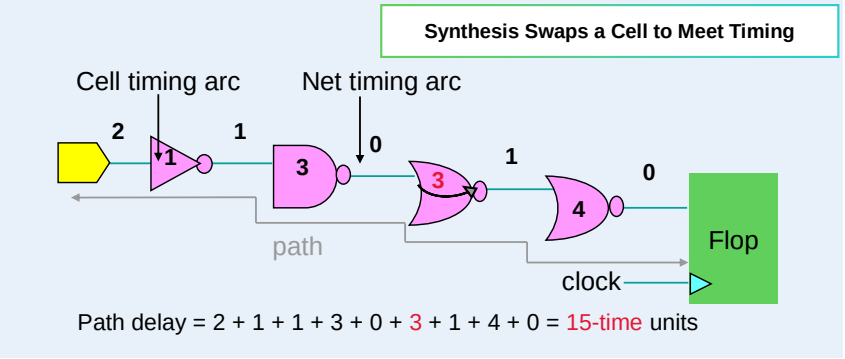
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is the Purpose of STA?

First, STA calculates the path delays for optimization tools. Then, based on the path delays, the optimization tools choose cells from the timing library to create a circuit that meets your timing requirements.

Second, STA analyzes the timing of a circuit to verify that the circuit works at the specified frequency.



 © Cadence Design Systems, Inc. All rights reserved.

Synthesis tools convert your RTL to gates. Place-and-route further optimizes your design. For these optimization tools to find the right cells for the right location, they rely on static timing analysis to calculate the path delays. Static timing analysis has to time millions of paths every hour to give the correct and accurate information to these optimization tools so that they can create a circuit that meets your timing requirements. Finally, the same static timing analysis is also used to analyze the timing of the circuit to make sure it works at the required frequency. This is done using more stringent requirements, such as those for signoff. Each of these tools, which are considered signoff tools, has different flavors or ways of doing their timing signoff with different capabilities.

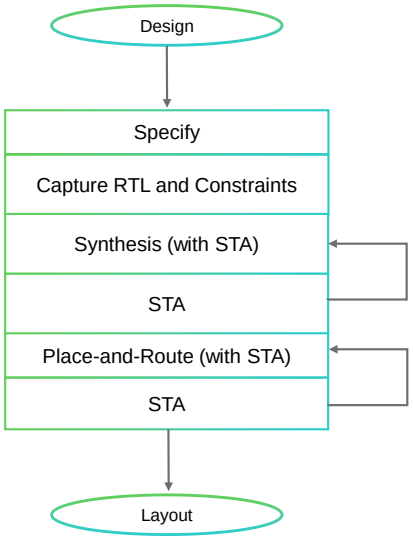
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

How Does STA Fit into a Design Flow?

You use STA within the synthesis, and you run it after synthesis and again after place-and-route.

- Specify circuit functionality and timing libraries.
- Capture RTL and write constraints.
- Convert RTL to gates (cells)1 using synthesis to meet the timing constraints.
- Run STA to verify the timing of synthesis results.
- Place cells in a layout and route the connections. Timing-driven placement and routing use STA to meet timing.
- Run final verification using signoff STA tools.



 © Cadence Design Systems, Inc. All rights reserved.

This is a typical flow for your design. First, we specify circuit functionality and timing libraries. You'll then capture your RTL and write constraints. If we provide all this input to the synthesis tool, it will then convert this RTL-to-gates using the libraries. While doing that, the synthesis tool will try to meet the timing constraints. This is the first time after we have used the synthesis tool, which uses STA internally to figure out what cells to use. Then, later on, at the end of the synthesis, we will run a static timing analysis to verify the timing of the synthesis results. When we are satisfied with that, we push it to place-and-route, where we again use static timing analysis to place the cells and route the design in such a way that it meets the timing requirements. The STA is used for this purpose to meet timing. Finally, we run the final verification using signoff tools to verify if your design meets your timing requirements, and then once we're satisfied, we push the design to layout.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Static Timing Analysis Tools

Timing tools in synthesis and place-and-route software include:

- Timing engine in the Genus™ Synthesis Solution
- Timing engine in Innovus™ and Tempus™ software

Timing verification tools (timing signoff tools) include:

- Tempus Timing Signoff Solution

 © Cadence Design Systems, Inc. All rights reserved.

cadence

The timing engines that are built-in synthesis and place-and-route software provide a fast way of selecting cells so that the synthesis and place-and-route software can optimize the paths for better timing. They all have this function of providing the timing information to these tools. You also have the timing verification tools that provide the final signoff after the design is synthesized, placed and routed. These timing verification tools typically sign off on all the different timing checks and constraints, whether the design meets the specifications or not. The timing tools or timing engines in synthesis and place-and-route often try to calibrate themselves to make sure that they correlate well with the timing verification tools.

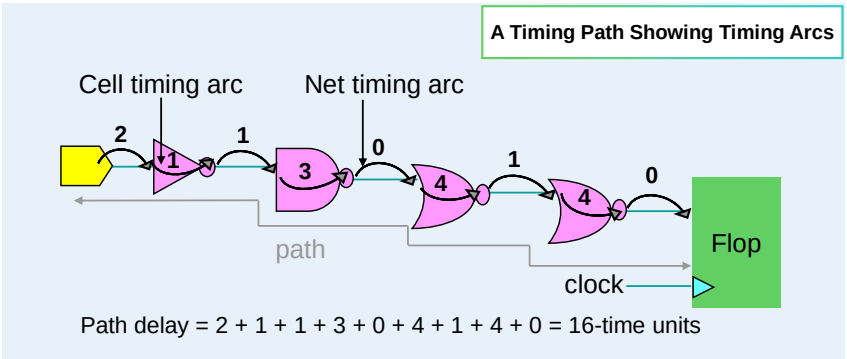
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Are Timing Paths?

The STA tool calculates the timing-path delays. The timing paths consist of two basic elements:

- Timing arcs in cells
- Timing arcs of nets



 © Cadence Design Systems, Inc. All rights reserved.

As we've seen before, the STA tool calculates the timing path delays. These timing paths mainly consist of two basic elements. One is the timing arc in the cells and the timing arcs of the nets. The goal of timing is to restrict the path delay to a certain amount as specified by the constraints. In this example, we have a flop that provides timing check data. So, the checks are also part of static timing analysis.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Are Timing Libraries?

Timing libraries provide the delays of the:

- Cells and
- Interconnects (nets)

The STA tool uses the delays of the nets and cells to calculate the path delays and verify the delays against the requirements.

The Liberty format (*.lib*) is the standard timing library in the industry.

 © Cadence Design Systems, Inc. All rights reserved.

Timing libraries provide delays of cells and interconnects, also known as nets and wires. Apart from this, they also provide power and skew information. The STA tool uses this information to calculate the path delays and then verify these path delays against the constraint requirements that are provided. The standard format that is widely used in the industry is the Liberty format, also known as ".LIB". There are other formats like ECSM or LBR that are used by different tools internally. Most tools accept the Liberty format or .LIB as an input.

© Cadence Design Systems, Inc. All rights reserved.

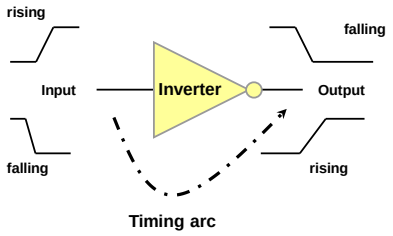
This document is for the sole use of Le Khanh Nam of Gmail

What Are Timing Arcs?

A timing arc is an imaginary arc that represents a single causal relationship.

If a change in an input causes a change in the output, it is known as a causal relationship.

Timing arcs provide a simple understanding of the structure and functionality of a gate.



Rising and falling timing arc delays across a gate are not always symmetric and are listed separately in a library.

 © Cadence Design Systems, Inc. All rights reserved.

What are timing arcs? If a change in the input is causing a change in the output, then we have a causal relationship. If the input in this inverter is rising and the output is falling, or when the input is falling, the output is rising. This dependency of the output on these inputs is called a *causal relationship*. Each of these imaginary arcs between the input and the output is called a timing arc. It's important to understand timing arcs so that you can determine the path delays correctly because not every arc is symmetric. The rising-to-falling arc is probably not the same as the falling-to-rising arc. We need to determine each arc based on how the output is actually behaving based on the inputs. All of this information is provided in the timing library, specifically on how to understand the purpose of each of these timing arcs.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Timing Arc Characteristics

The following are the characteristics of a timing arc:

- Unateness
- Slew
- Delay

All the above information for a timing arc is derived from the timing libraries.

 © Cadence Design Systems, Inc. All rights reserved.

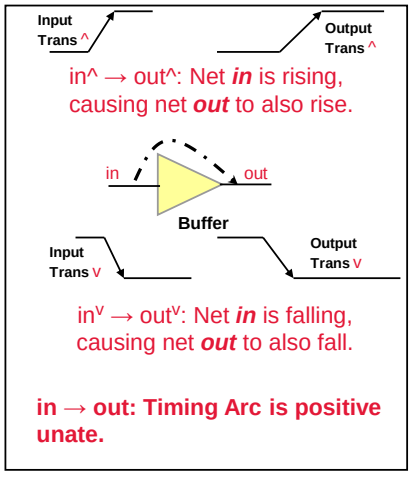
The following are the different characteristics of a timing arc: unateness, slew, and delay. Specific information for a timing arc can be obtained from the timing libraries. Let's look at what each of these means and their purpose.

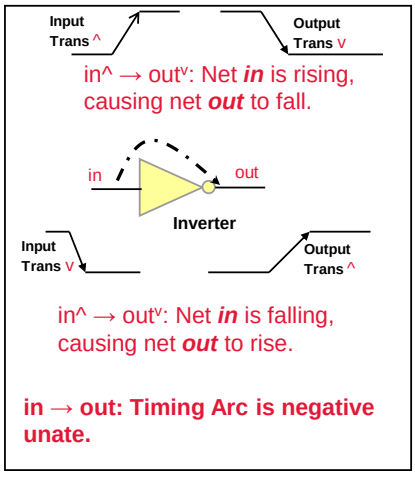
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Timing Arcs: Unateness

For a given timing arc, if the output signal transition (rise or fall) is the same as the input, then the timing arc is considered positive unate and vice versa. Sometimes the arc can be nonunate.





 © Cadence Design Systems, Inc. All rights reserved.

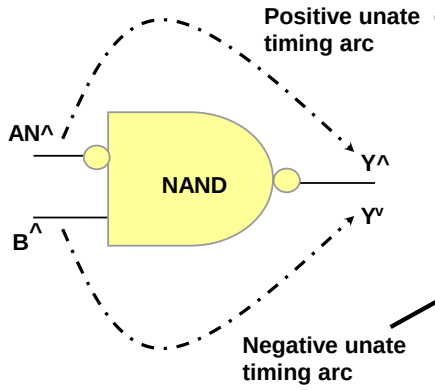
Unateness is a very important property of all timing arcs. Static timing analysis uses unateness extensively to determine how the timing information is calculated. If you look at the example of a buffer, the output is rising when the input is rising; the output is falling when the input is falling. So, the timing arc is a positive unate because the transition of the output is the same as the transition on the input. If you look at the inverter, when the input is rising, the output is falling. And when the input is falling, the output is rising. So, the transition is the opposite of the input transition. Therefore, this is considered a negative unate. Some arcs, of course, can be nonunate where there is no relationship between how the output is changing against the input. But most of the time, you can easily determine the unateness of a timing arc by looking at the truth table of that cell. So, for a buffer, you can say it's going from zero to one, and the output is changing from zero to one at the same time. So that gives you clues into how to determine the unateness of each individual timing arc in certain cells that are multiple inputs. We have to look at each timing arc to determine what the unateness of that timing arc is. For every input-to-output relationship, that is one arc.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Example of Timing Arcs in a Library

This diagram shows timing arcs within the NAND gate (cell) and the unateness of the arcs.




```
 pin(Y) {
    direction:output; capacitance:0.0;
    function : "(!(!AN B))";
timing() {
      related_pin : "AN";
      timing_sense : positive_unate;
…
 timing() {
      related_pin : "B";
      timing_sense : negative_unate;
…
      cell_rise(delay_template_7x7) {
      …
      }
      cell_fall(delay_template_7x7) {
      …
      }
```

 © Cadence Design Systems, Inc. All rights reserved.

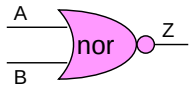
Every input-output combination forms a timing arc. Here you can see the different arcs of a NAND cell and their corresponding library timing sense attributes. The timing sense of a NAND gate for the AN pin is shown as positive unate and the B pin as negative unate. Each related pin under the timing section of the output pin represents a timing arc. The unateness of the timing arc is specified in the library by the timing sense attribute. When the net B is rising, net Y is falling. Therefore, the timing arc is negative unate. When net A is rising, net Y is rising. Therefore, the timing arc is positive unate.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Timing Arcs

1. How many timing arcs are there in this NOR cell?



2. What is the unateness of the arcs in the NOR cell?

In a minute, the answer will appear on the page.

Move the time slider to the end if you want to see the answer sooner.

Solutions are also in the Notes page.

 © Cadence Design Systems, Inc. All rights reserved.

Try the following activities to reinforce your learning.

Solution

A NOR gate has 2 (2C1) input-to-output combinations. Each arc has two valid transitions, rise and fall. Therefore, you have 4 timing arcs.

From the truth table of the NOR, you can see that:

- When either of the inputs is rising, the output is falling, and vice versa. Therefore, both the timing arcs of a NOR gate are negative unate.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Timing Arcs: Signal Transition and Slew

Transition time is the time that a signal takes to change states from low-to-high or high-to-low.

- Rise and fall transition times are the properties of a timing arc and are typically measured in nanoseconds.
- Rise and fall times can be either on the input signal or the output signal.
- **Slew** is the rate of transition measured in Volts per nanosecond.

STA tools measure input transition time using slew thresholds.

- Slew thresholds are defined in the library, and the input rise and fall times are calculated from the slew thresholds.

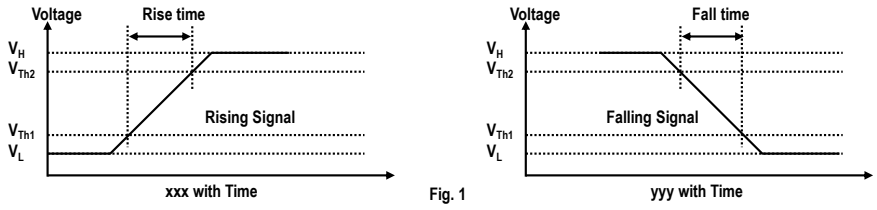


Fig. 1

***Note**: Some tools, including Cadence® tools, interchangeably refer to transition times as slew, depending on the context.*

 © Cadence Design Systems, Inc. All rights reserved.

All semiconductor devices take some time to switch between states. Transition time is the time that a signal takes to change states from low to high or high to low. The picture shows the signal rising from low to high and the signal falling from high to low. These rise and fall transition times are the properties of a timing arc. Usually, rise time and fall time are measured between 10% and 90%, or 20% and 80% of your input signal or your output signal, and are calculated as rise times or fall times. Technically, the rate of transition is measured in volts per nanosecond and is called slew. Transition times, such as the rise and fall times, are measured in terms of time, typically in nanoseconds. STA tools measure input and output transition time using slew thresholds. These slew thresholds are defined in the library, and the input rise and fall times are calculated from the slew thresholds.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

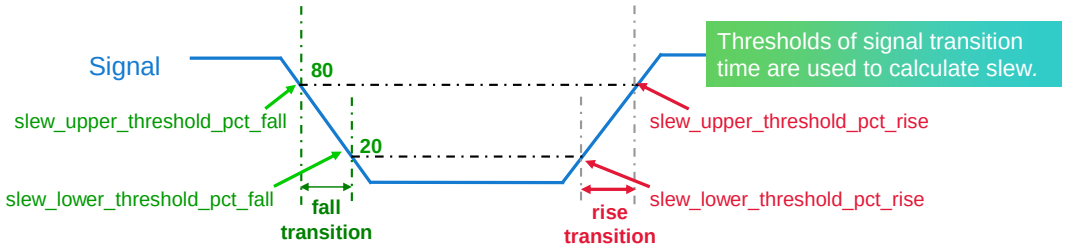
Transition: Rising and Falling Transition Times

STA tools use the slew threshold values from the library to calculate the transition times.

Example

STA tools calculate the rise time from the *slew_lower_threshold_pct_rise* of **20%** and the *slew_upper_threshold_pct_rise* of **80%**. Similarly, STA tools use the fall threshold values for fall time.

The graph shows transition time (slew) being measured from 80% to 20% of the falling signal, for the fall transition time (aka fall slew), and from 20% to 80% for the rise transition time (aka rise slew).



 © Cadence Design Systems, Inc. All rights reserved.

The transition time is the time it takes for a signal to rise or fall. The upper threshold value determines the actual time when a device turns on and stays on. The lower threshold value determines the time at which the device turns off and stays off.

In the illustration, the lower thresholds are 20%, and the upper thresholds are 80%. The rise transition is measured from 20% to 80% of the signal, and the fall transition is measured from 80% to 20%.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

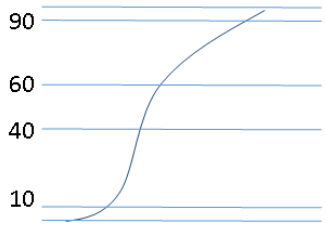
Example of Slew Thresholds

Slews are typically measured in a small voltage range (like 40-60) to capture a more linear portion of the transition.

Measured slews are scaled to the library slew thresholds.

Example (from .lib)

Slew is measured **0.05** at range **40-60** (**20**% swing). The library slew thresholds are **10-90** (**80**% swing), then this slew is recorded as **0.05 x 80/20 = 0.20**




```
/* Example .lib threshold definitions */
  slew_lower_threshold_pct_fall : 10.0;
  slew_upper_threshold_pct_fall : 90.0;
  slew_lower_threshold_pct_rise : 10.0;
  slew_upper_threshold_pct_rise : 90.0;
  input_threshold_pct_fall      : 50.0;
  input_threshold_pct_rise      : 50.0;
  output_threshold_pct_fall     : 50.0;
  output_threshold_pct_rise     : 50.0;
slew_measure_lower_threshold_pct_fall :40%
slew_measure_upper_threshold_pct_fall: 60%
slew_measure_lower_threshold_pct_rise: 40%
slew_measure_upper_threshold_pct_rise: 60%
```

 © Cadence Design Systems, Inc. All rights reserved.

Slews are typically measured in a small voltage percentage range, like 40 to 60, to capture a more linear portion of the transition. These measured slews are then scaled to the library slew thresholds. From the library, we have an example here where slew is measured as .05 at the 40 to 60 range. So, it's a 20 and swing. And the library slew thresholds are 10 to 90, which is an 80% swing. So, this rate of transition or slew is recorded as 0.05 times 80 over 20, which is 0.20.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Understanding Output Transition

STA tools get the output transition time from the library and use that value as the input transition to the next cell in the path.

Libraries list the input transition (slew) as **index_1**, and the output load as **index_2**.

SPICE tools use index_1 and index_2 to derive the output transition time. This output transition time is listed in the library as the **values**.

Example (from .lib)

The value for input rise transition is 0.05 and the value for output load is 0.021.

The value of output rise transition is taken from the *lookup table* (LUT) model as **0.293559**.

index_1 is rows
index_2 is columns

```
##Example .lib showing rise and fall output transitions.
rise_transition(delay_template_7x7) {
  index_1 ("0.05, 0.15, 0.6, 1.4, 2.3, 3.3,4.5");
  index_2 ("0.00035, 0.021, 0.0385, 0.084, 0.147, 0.231, 0.3115");
  values ( \
 "0.049669, 0.293559, 0.504914, 1.054467, 1.815376, 2.829936, 3.802228", \
 …
  "0.114690, 0.331215, 0.530978, 1.073969, 1.825687, 2.834624, 3.805160");
}
fall_transition(delay_template_7x7) {…}
```

LUT		Load			
		0.00035	**0.021**	0.0385	…
Slew	**0.05**	0.049669	**0.293559**	0.504914	…
	0.15	…	…	…	…
	…	…	…	…	…

 © Cadence Design Systems, Inc. All rights reserved.

Timing libraries capture everything in the form of table lookup, also known as lookup tables or LUTs. In this example, it represents what's in the library in the form of a table and is more easily readable. If you want to identify what the output transition is, just picture the table plotted as index 1 and index 2, where input slew is index 1, and load is index 2. You look up the output transition value for that corresponding slew and corresponding load.

This example shows that the output transition is plotted between the input transition and the output load. Based on the example for an input transition of 0.05, the value of the output load is 0.021. Based on the lookup table, the value is shown to be .293559. In these lookup tables, remember that index one is always the row, and index two is always the column. This way, you can always identify what the values are based on what is specified for index one and then index two.

The delays of the next stage input transition or slew depend on the input transition or slew of the previous stage. Thus, we calculate the output transition or the next stage input transition or slew values using these lookup tables.

To reiterate, the STA tools calculate the output transition times and use that value as the input transition to the next cell in the same path or next stage. This is how we propagate the delays and cells and find out the delays throughout the path from these lookup tables from one stage to the next and so on.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Output Slew Degradation

Slew at an output pin degrades as it travels to the next input pin, especially on a long net.

The lookup table for slew degradation is based on the following:

- Output pin transition (slew),
- Interconnect (net) delay.

Example (from .lib)

If the value for output slew is **1**, and net delay is **1**, then the value of degraded output slew would be **1.1** for rise and **1.2** for fall.

```
 /* Table Lookup for Slew */
lu_table_template(trans_deg) {
variable_1: output_pin_transition ;
variable_2 : connect_delay ;
                  index_1("0, 1");
                  index_2("0, 1");
  }
/* Slew Degradation Tables */
rise_transition_degradation(trans_deg) {
   values("0.0, 0.1", "1.0, 1.1");
  }
fall_transition_degradation(trans_deg) {
       values("0.0, 0.2", "1.0,  1.2");
  }
```

 © Cadence Design Systems, Inc. All rights reserved.

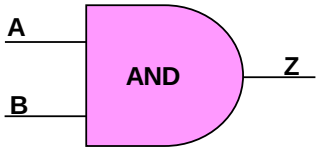
Because the input signal is constant, and the signal is traveling across many stages along the path, the slew at the output of one stage degrades before it gets to the next stage, and so on and so forth. As it travels along a long net, you'll see slew degradation. So as this slew degrades, the signal degrades as well. The delays get longer and longer based on the slews degrading. So, we need to capture all these slew degradations as we go along. There is also a lookup table for slew degradation, which uses the output pin transition and the interconnect delay or the net delay and calculates the degradation. You will see these degradation tables in your timing libraries. These kinds of calculations give you the overall timing through that path and how badly it will degrade based on these slew degradation tables.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Output Slew

Open the *test.lib* file from the *Database Download* module attachments and find the worst-case output transition of the AND2X1 cell, given that the value of the input net transition is 0.206, and the value of the output capacitance is 0.010112.



In a minute, the answer will appear on the page.

Move the time slider to the end, if you want to see the answer sooner.

Solutions are also in the Notes page.

 © Cadence Design Systems, Inc. All rights reserved.

Try the following activities to reinforce your learning.

Solution

In the library, you will find this LUT for the rise transition of the A → Z arc:

```
rise_transition(delay_template_7x7) {
  index_1 ("0.042, 0.066, 0.112, 0.206, 0.392, 0.764, 1.5");
  index_2 ("0.00079, 0.002054, 0.00474, 0.010112, 0.020856, 0.042186, 0.08532");
  values ( \
    "0.049964, 0.064096, 0.095169, 0.160213, 0.295482, 0.567142, 1.116929", \
    "0.050165, 0.064589, 0.095406, 0.160276, 0.295541, 0.567171, 1.116947", \
    "0.050550, 0.064846, 0.095540, 0.160350, 0.295551, 0.567165, 1.116943", \
    "0.052987, 0.067255, 0.096955, 0.160932, 0.295615, 0.567169, 1.116939", \
```

Therefore, for the index_1 of 0.206 and index_2 of 0.010112, the rise transition is 0.160932.

From the library, input slew and output load are the index_1 and index_2 from delay_template_7x7.

Then, under the transition tables for AND2X1, we find:

- The value of cell delay for A → Z rising = 0.160932,
- The value of cell delay for A → Z falling = 0.097039,
- The value of cell delay for B → Z rising = 0.160815,
- The value of cell delay for B → Z falling = 0.098912.

Therefore, the worst transition time would be the rise time of Z when A is rising.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Timing Arcs: Cell and Net Delay

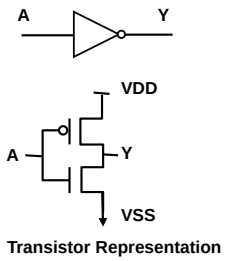
In addition to unateness and transition, timing arcs also have cell or net delay.

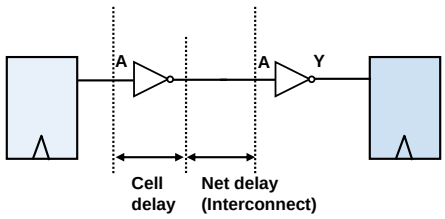
Cell delay

- Transistors within a cell take a certain amount of time to switch. Therefore, a change to the input of a cell takes time to cause a change to the output.

Net delay

- Net delay is the delay between the time a signal is first applied to a net and the time it takes to reach other devices connected to that net.





 © Cadence Design Systems, Inc. All rights reserved.

Depending on the process technology, different physical elements have different levels of contribution. Delays encountered in digital circuitry are composed of two main components cell delay and net delay. Each stage delay, whether it's cell delay or net delay, represents the time required to propagate a signal from the input of one stage to the input of the next. Historically, with process technologies above 90 nanometers, cell delay has been the major limiting factor in timing closure. However, at process technologies below 90 nanometers, net delays dominate the cell delays.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module 3

Cell Delay

In this module, let's look at how tools calculate cell delay.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objectives

In this module, you

- Identify cell delay information from the libraries
- Use Table Lookup (TLU) models to calculate cell delay

 © Cadence Design Systems, Inc. All rights reserved.

In this module, you'll learn how to identify the cell delay information that comes from the libraries, and how these table lookup models are used to calculate the cell delay.

© Cadence Design Systems, Inc. All rights reserved.

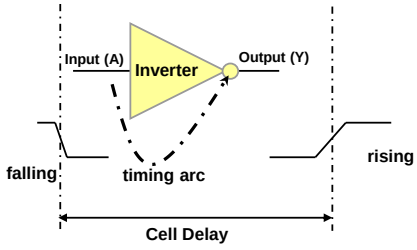
This document is for the sole use of Le Khanh Nam of Gmail

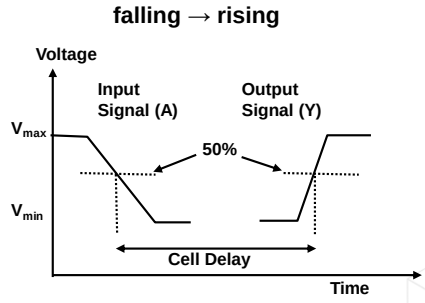
What Is Cell Delay?

Each cell has one or more timing arcs, and every timing arc has a propagation delay. *Propagation delay* is the time difference from the time when the input signal changes state to the time when the output signal changes state. This is commonly referred to as *cell delay*.

Delay through a cell is determined by:

- The intrinsic delay.
- The load that it is driving.
- The input transition, also known as input slew.





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

A gate or cell has one or more timing arcs, and each of these timing arcs has a propagation delay associated with them. This propagation delay through the cell is commonly known as cell delay.

We calculate cell delay based on the intrinsic delay of the cell, the load it's driving, and the input transition.

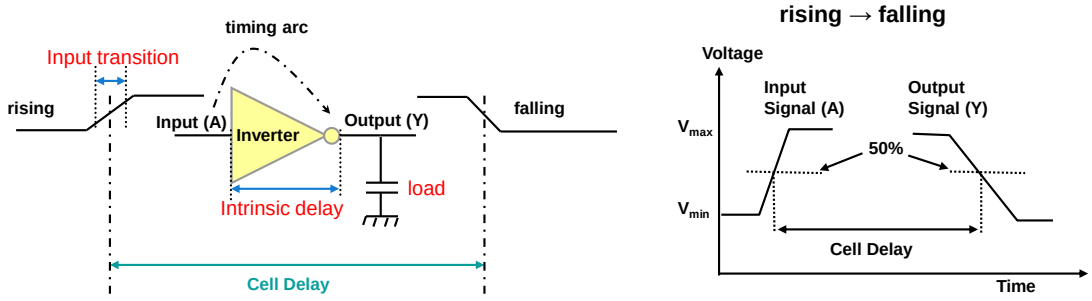
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is Cell Delay? (continued)

Types of cell delay include the following:

- *Transition* is the time it takes for the pin to change state from low-to-high or high-to-low.
- *Intrinsic delay* is defined as the delay in a cell when a signal with zero transition time is applied to the input pin, and the output pin does not have any load.



 © Cadence Design Systems, Inc. All rights reserved.

Propagation delay is the total delay between when the input signal changes state to when the output changes state. This is commonly known as cell delay. You can break down cell delay into transition times and intrinsic delay. Transition time is when the pin changes state from low to high or high to low. Intrinsic delay is the delay of the cell when there is no input transition or output load.

© Cadence Design Systems, Inc. All rights reserved.

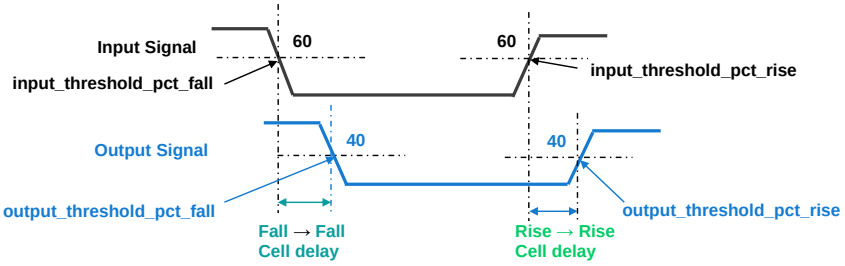
This document is for the sole use of Le Khanh Nam of Gmail

Cell Delay: Input and Output Thresholds

By default, STA tools measure the cell delay from 50% of the input signal to 50% of the output signal.

The library specifies the thresholds of the input and output signals to calculate the delay from input to output.

In this example, the rise and fall cell delays are measured from 60% of input to 40% of output.




```
/* Example .lib threshold definitions */
  input_threshold_pct_fall      : 60.0;
  input_threshold_pct_rise      : 60.0;
  output_threshold_pct_fall     : 40.0;
  output_threshold_pct_rise     : 40.0;
```

 © Cadence Design Systems, Inc. All rights reserved.

The propagation delays are measured from the input threshold percentage point to the output threshold percentage point. We commonly measure the cell delay from 50 percent of the input signal to 50 percent of the output signal. If you look at the libraries, these input threshold percentages and output threshold percentages are given for both rise and fall. We measure the delay from the input signal to the output signal based on these library percentage thresholds. In this example, we are measuring the cell delay from 60 percent of the input to 40 percent of the output because the input threshold is given at 60 percent, and the output threshold is at 40 percent.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Example of a Cell Model

The example on the right is an excerpt of the cell library model. The cell models include the following features:

- All the timing arcs, from the inputs to the cell outputs.
- The **output function**.
- The cell footprint that tells the place and route tools what the layout of the cell is.
- The cell delays.
- The output transition times.
- The cell power dissipation tables.

There are two standard models for LUT:

- 2-dimensional LUT, which uses input slew and output load.
- 3-dimensional LUT, which uses input slew, output load, and related output load.

```
/* Example cell model from a .lib*/
cell (NAND2BX1) {
  cell_footprint : nand2b;
  area : 6.7896;
  pin(AN) {
   direction:input;
   capacitance:0.001304;
  }
  pin(B) {
   direction:input; capacitance:0.002610;
  }
  pin(Y) {
   direction:output;
   capacitance:0.0;
   function : "(!(!AN B))";
timing() {
    related_pin : "AN";
    timing_sense : positive_unate;
…
 timing() {
    related_pin : "B";
    timing_sense : negative_unate;
```

You can use a different cell of the same cell_footprint, with different areas or timing without affecting the path's logic.

Output represented as a function of input pins.

Timing arcs within the cell show the unateness of the arcs.

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

You'll see a lot of different data for any cell inside the cell model of the library.

Each cell model in the library defines the cell footprint, the area of that cell, the various pins of the cell, the directions of the pins, the output function for each output pin, etc. Based on this information, library characterization tools can create, for example, cell delay models for you to use at the different slew and load variations. These tools can also determine transition times, power dissipation, yields, etc., based on different input variables. These different things are listed in the library as calculated values in a lookup table format.

There are two main models for cell delay in the Liberty libraries, the 2D lookup table and the 3D lookup table.

The 2D cell delay lookup table uses the input slew and output load as input variables to look up the values of the delays. The 3D cell delay lookup table uses the input slew, output load, and related output load.

If there are multiple outputs, for example, Y and Ybar, then to calculate the delay to Y, you use the output load as seen on pin Y and the related output load as seen on Ybar and vice versa for Ybar. The accuracy of the lookup tables depends on the accuracy of the simulations of such characterization tools. The values from these simulations are plugged into your Liberty Library models, which you will then use in your static timing analysis.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Example of a Cell Delay 2D LUT Model

The following cell delay table is a 2-dimensional LUT model using the input slew and output load to determine the delay through the cell.

Example (from .lib)

For a slew of **1.05** and a load of **0.3279**, from the 2D table, the cell delay equals **0.3643**.

Slew Axis \ Load Axis	0.0410	0.0820	0.1639	0.3279	0.6557	1.3115	2.6230
0.0500	("0.0435,	0.0599,	0.0897,	0.1364,	0.2185,	0.3959,	0.7750", \
0.3000	"0.0815,	0.1120,	0.1658,	0.2470,	0.3551,	0.5173,	0.8830", \
0.5500	"0.0964,	0.1331,	0.1983,	0.3000,	0.4480,	0.6635,	1.0099", \
0.8000	"0.0962,	0.1440,	0.2206,	0.3375,	0.5051,	0.7811,	1.1515", \
1.0500	"0.0981,	0.1515,	0.2349,	0.3643,	0.5526,	0.8605,	1.3025")

© Cadence Design Systems, Inc. All rights reserved.

This is an example of a 2D lookup table model. If you look at the .LIB, the library itself, the slew is plotted on the Y axis here, and the load is plotted on the X axis. For example, for the slew of 1.05 and for a load of .3279, it shows .3643. You would determine that based on the lookup table, and that's what the tool will also do.

© Cadence Design Systems, Inc. All rights reserved.

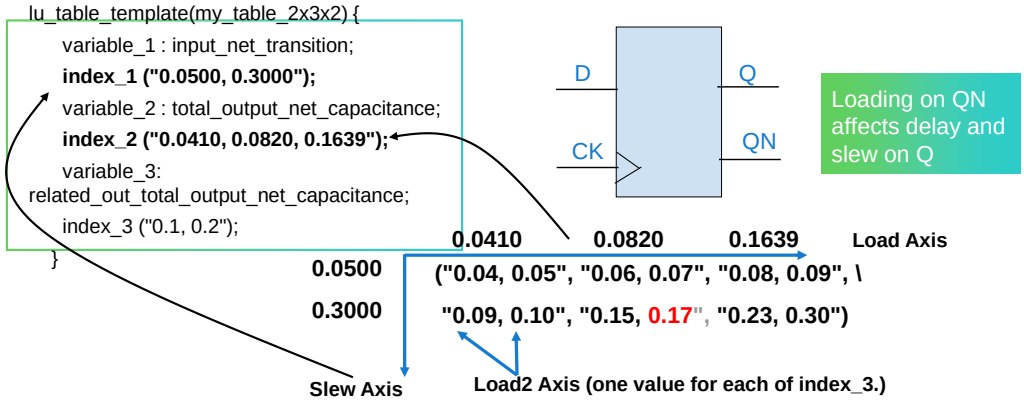
This document is for the sole use of Le Khanh Nam of Gmail

Example of a Cell Delay 3D LUT Model

The following is a 3-dimensional LUT model using the output load, related output load, and input slew to determine the delay through the cell.

Example (from .lib)

For a slew of 0.3, a load of 0.0820, and a related output load of 0.2 from the 3D table, the cell delay is **0.17**.



 © Cadence Design Systems, Inc. All rights reserved.

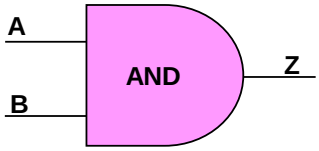
In a 3D lookup table model, you also have the related output load as seen by QN. This illustration shows the slew axis, the load axis, and load 2 axis. For this lookup table, the related output load has two values. Once you locate the slew and load axis, for the third axis, we use load 2, and thus calculate the delay based on load 2. If the delay doesn't fall on any of these load points, then the tool interpolates the values to arrive at your cell delay.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Cell Delay

Open the *test.lib* file and find the worst-case cell delay of the AND2X1 cell, given that the value of the input net slew is 0.206 and the value of the output capacitance is 0.010112.



 © Cadence Design Systems, Inc. All rights reserved.

Try the following activity to reinforce your learning.

Solution

In the library, the input slew and output load are the index_1 and index_2 from delay_template_7x7.

Then, under the cell delay for AND2X1, you will find this LUT in the library:

```
cell_rise(delay_template_7x7) {
  index_1 ("0.042, 0.066, 0.112, 0.206, 0.392, 0.764, 1.5");
  index_2 ("0.00079, 0.002054, 0.00474, 0.010112, 0.020856, 0.042186, 0.08532");
  values ( \
  "0.091522, 0.102488, 0.123201, 0.161183, 0.234593, 0.379090, 0.670558", \
  "0.096236, 0.107177, 0.127815, 0.165780, 0.239183, 0.383702, 0.675180", \
  "0.105184, 0.116100, 0.136686, 0.174547, 0.247873, 0.392410, 0.683938", \
  "0.119600, 0.130721, 0.151606, 0.189428, 0.262634, 0.407105, 0.698662", \
  "0.133882, 0.145795, 0.167463, 0.205899, 0.279294, 0.423465, 0.714886", \
```

Therefore, the value of cell delay for A → Z rising = 0.189428

Similarly, for the other timing arcs, you will find these values in their LUTs:

The value of cell delay for A → Z falling = 0.216009

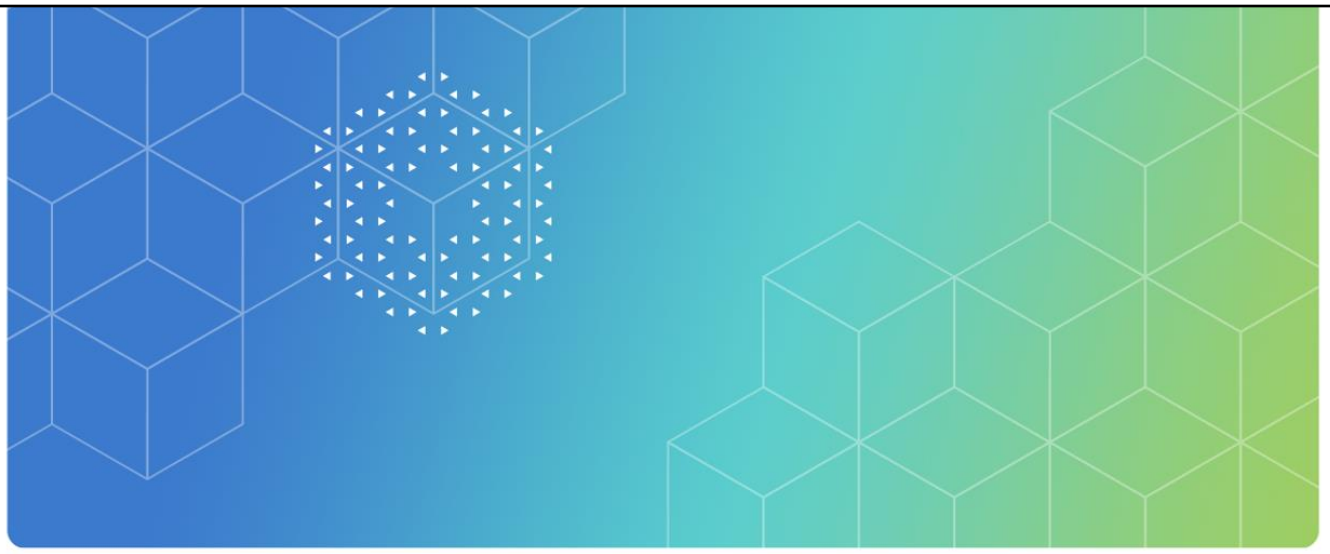
The value of cell delay for B → Z rising = 0.191553

The value of cell delay for B → Z falling = 0.233191

Therefore, the worst delay is through B → Z falling equals 0.233191

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail



Module 4

Net Delay

Let's look at how the tool calculates Net Delay.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objectives

In this module, you

- Apply wireload models to calculate net delays
- Identify the file formats used in backannotation

 © Cadence Design Systems, Inc. All rights reserved.

In this module, we'll look at what wireload models are and how they're used to calculate net delays. We'll also identify the different file formats that are used in the various tools that represent the wire delay information.

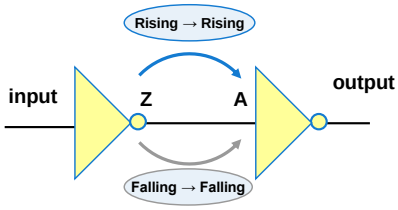
© Cadence Design Systems, Inc. All rights reserved.

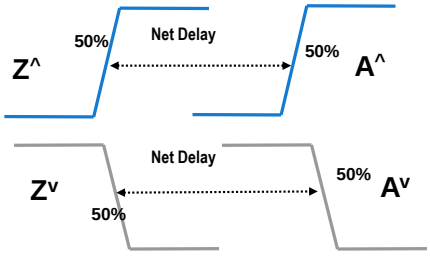
This document is for the sole use of Le Khanh Nam of Gmail

What Is Net Delay?

The interconnect provides the timing arc from pin to pin and has an inherent delay. This delay is because of the resistance and capacitance of the interconnect.

This interconnect delay is also known as wire delay or net delay.





All interconnect arcs are positively unate.

All net delays are usually calculated from 50% of the input to 50% of the output.

 © Cadence Design Systems, Inc. All rights reserved.

Your design layout uses different metal layers and vias to connect the various gates or cells in your design; thus, they are called interconnects or wires. The metal layers inherently have a delay because of the resistance and capacitance, i.e., the physical properties of the metal layers themselves. This inherent delay of the metal layers used to connect the cells is also known as interconnect delay, or wire delay, or simply net delay.

The net delays are typically calculated from 50 percent of the input to 50 percent of the output. It is important to notice is that all these interconnects are naturally positively unate.

The timing arcs of the nets connect the timing arcs of the cells to form a timing path. The timing of these timing paths is then verified against any given requirements to make sure your design works.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Wire-Delay Estimation

Physical Layout Estimation (PLE) in Genus™
PLE uses actual design and physical library information.

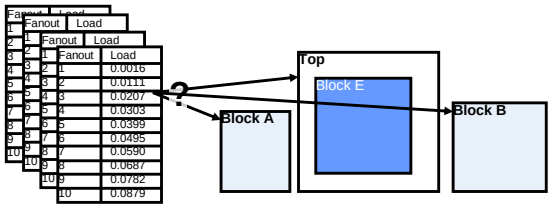
- Dynamically calculates wire delays for different logic structures in the design.
- Correlates better with place-and-route.
- Inputs
  - LEF library
  - Capacitance table
  - Floorplan

Wireload Models

- Wireload models are statistical.
- Wireloads are calculated based on the nearest calibrated area.
- Selection of appropriate wireload models for a design is tedious.
- Correlation is difficult even with custom wireload models.
- Inputs
  - Wireload models

Some Types of Routing

- Steiner route
- Trial route
- Global route
- Nanoroute



 © Cadence Design Systems, Inc. All rights reserved.

Routing tools like nanoroute that you see here are employed by the physical design tools to lay down the metal layers or the wires. Physical tools use routing and extraction to determine the actual wire delays by laying down the metal layers to connect the various cells in your design. But the front-end synthesis tools like Cadence Genus do not have that luxury. So, they need to rely on some form of wire delay estimation to accurately predict the path delays. Wire delay estimation can be done either through wireload models or a layout wire delay estimation tool like the physical layout estimation or PLE in Cadence Genus Synthesis Solution. The wireload models or PLE must accurately predict the wire delays to meet the requirements of your design. Therefore, they need to correlate to the actual routing performed by the physical design tools.

The foundries supply the wireload models along with the timing libraries, and you can plug the wireload models into your static timing analysis tool to calculate your net delays. Unfortunately, one of the cons of wireload models is that they are statistical, and they're calculated by the foundries based on the nearest enclosing block area. So, based on a given block size, the STA tools must predict what the actual wire length from one cell to another cell must be. This can be error-prone because not every cell in the design is placed at these estimated wire lengths. Furthermore, we use a guess for the fanouts and loads to estimate the maximum wire length, then calculate the corresponding resistance and capacitance of the estimated wire length, and finally the delay of the wire. This multi-step process can be very tedious and result in a very pessimistic view of the wire delays and thus the design we want to create. Thus, wire-load models can be challenging when it comes to correlation to the actual place-and-route tools.

Nowadays, with tools like PLE in Genus, when you provide the process technology files and the floorplan of your design, they try to calculate wire delays dynamically. The correlation with PLE is much better than the wire-load models and is much closer to the routing tools.

© Cadence Design Systems, Inc. All rights reserved.

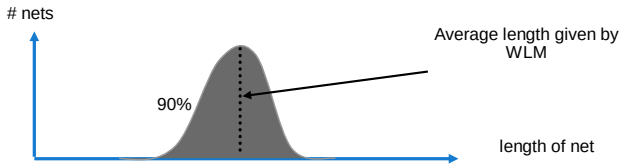
This document is for the sole use of Le Khanh Nam of Gmail

Estimating Net Delay with Wireload Models

The net resistance (R) and the net capacitance (C) are used to calculate the delay of the net.

$Delay_{net} = R_{net} * C_{net}$

Wire-load models (WLMs) provide synthesis tools with an estimation of net resistance (R) and capacitance (C) based on the length and fanout that represent 90% of the nets.



Poisson Distribution of *number of nets* versus *length* for a fixed size design, and fanout = 1

 © Cadence Design Systems, Inc. All rights reserved.

The wireload models are calculated based on the average statistical net length over many designs. A Poisson distribution, such as shown here, is used to determine the average net length for nets in a certain block. Average statistical net lengths thus determine the corresponding resistance and capacitance, and therefore you calculate the delay of that net. These net lengths that are statistically weighted make the wire loads pessimistic and, therefore, might be unreliable for some designs. But, because they provide a good estimate, you could use them as a good starting point for synthesis or even for critical path analysis right after the synthesis stage.

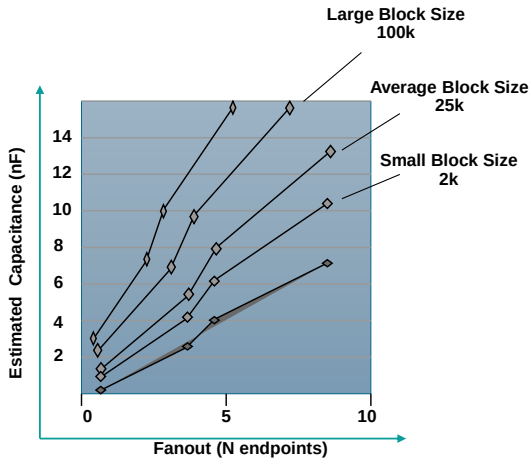
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Calculating Net Delay from Wireload Models

How does an STA tool calculate net delay from a WLM?

- Determines the area that a net fits in.
- From the wireload selection table, selects the WLM with a block area that just encloses the area of the net.
- Uses the WLM to estimate the wire length based on the fanout of the net or uses the wireload table to model capacitance and resistance more accurately.
- The capacitance multiplier (CM) and resistance multipliers (RM) are used to determine the actual RCs of the net using the following:
  - Cnet = CM*length
  - Rnet = RM *length
- The delay of the net is calculated using:
  - Delay = Rnet*Cnet
- You can use custom wire-load models generated from the place-and-route tools or use the statistical wireload models in the library provided by the vendor to calculate net delays.



For the same fanout, the bigger the block size, the bigger the estimate of average net length, and hence the value of the capacitance.

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

In the graph, notice that the larger block sizes and larger fanouts have larger capacitances, and thus larger delays. To reduce pessimism, you may choose to generate custom wireload models based on your design from certain tools; instead of using the generic wireload models provided by your library vendor. To minimize pessimism or to simplify the net delay calculation, you can choose one of the multiple modes that the timing tools might allow, such as the top mode, the enclosed mode, and the segmented mode. Depending on the mode that is being used, varying block sizes are selected for the wireload model selection table.

Therefore, to calculate net delay using the wireload model's method, you need various wireload models based on various block sizes.

The STA tool determines an area that a particular net fits into and, using the wireload selection table, selects the block area that encloses the area of the net. Then, you use the wireload model to obtain the average wire length for a given fanout and block size. Finally, use the capacitance multiplier and resistance multiplier to determine the delay of the net.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Example of a Wireload Selection Table

Select the WLM with a block area that just encloses the area of the net.

The STA tool uses the area-based wireload selection table to select the proper wireload model for your block or subdesign.

Example

If the block area is between **0** and **203** area units, the tool selects the **B0.5X0.5** wireload model.

```
/* Example .lib wire-load selection table*/
wire_load_selection() {
  wire_load_from_area( 0, 203, "B0.5X0.5");
  wire_load_from_area( 203, 813, "B1X1");
  wire_load_from_area( 813, 3252, "B2X2");
  wire_load_from_area( 3252, 7317, "B3X3");
  wire_load_from_area( 7317, 13008, "B4X4");
  wire_load_from_area( 13008, 20325, "B5X5");
  wire_load_from_area( 20325, 29268, "B6X6");
  wire_load_from_area( 29268, 39837, "B7X7");
  wire_load_from_area( 39837, 52032, "B8X8");
  wire_load_from_area( 52032, 65853, "B9X9");
  wire_load_from_area( 65853, 81300, "B10X10");
```

 © Cadence Design Systems, Inc. All rights reserved.

This example shows the wireload selection table. If the block area is between zero and 203, then the tool selects the B point five point five (B.5.5) wireload model.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Example of a Wireload Model

Use the wireload model to estimate wire length based on fanout.

Example

The resistance of a wire with **5** fanouts is **0.885 x 0.020929 = 0.0185221 kΩ**.

The capacitance of a wire with **5** fanouts is **0.885 x 0.14221 = 0.1259 pF**

Now the net delay is calculated as R*C, which would be **0.0185221*0.1259 = 0. 00233193239ns** or approximately **2ps**.

```
/* Example .lib wire-load table*/
 wire_load("B0.5X0.5") {
    resistance : 0.020929;
    capacitance : 0.142210;
    area : 3.255208;
    slope : 0.177000;
     fanout_length (1, 0.177000);
     fanout_length (2, 0.354000);
     fanout_length (3, 0.531000);
     fanout_length (4, 0.708000);
     fanout_length (5, 0.885000);
     fanout_length (6, 1.062000);
     fanout_length (7, 1.239000);
     fanout_length (8, 1.416000);
     fanout_length (9, 1.593000);
     fanout_length (10, 1.770000);
  }
```

 © Cadence Design Systems, Inc. All rights reserved.

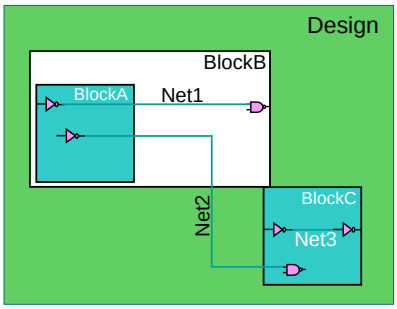
The actual wireload model looks similar to this, where you have these kinds of resistance multipliers and capacitance multipliers. And the RC is now calculated based on the capacitance of the wire and resistance of the wire.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Wireload Models

Open the *test.lib* file from the *Database Download* module and find the default net delay for a net in a block with an area of 1000 and a fanout of 5.



Open the *test.lib* file from your downloaded resources.

 © Cadence Design Systems, Inc. All rights reserved.

Try this activity to test your understanding of the concepts discussed.

Solution

In the test.lib file, you will find a default_wire_load: "TSMC13_Conservative“. In the TSMC13_Conservative wireload, for an area of 1 unit:

- The net length at 5 is 0.01007 from the table.
- The capacitance multiplier is 1.
- The capacitance for a fanout length of 5 is 0.01007*1.
- The resistance multiplier is 0.
- The resistance for a fanout length of 5 is 0.01007*0.
- Therefore, the delay of the net is R*C = 0.
- Side Note: To find fanout lengths not listed in the table, the slope is used.
- For example, in this case, the slope is 0.002993.
- So, the net length at fanout 15 would be calculated as
- Net length @ 10 + (slope) * (15-10)
- = 0.029938 + 0.002993*5

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is Backannotation of Net Delay?

After initially using WLMs to generate the delays of the nets, you can refine your estimates with real extracted data.

When the design is placed and routed, the extracted parasitic (RC) information is gathered and passed back to the synthesis tools either directly as parasitics or as delays from a delay calculator. This process is known as backannotation.

Backannotated information is available in the following file formats:

Format	Description
SPEF (standard parasitic exchange format)	For extracted parasitics
DSPF (detailed standard parasitic format)	For detailed extracted parasitics
RSPF (reduced standard parasitic format)	For less accurate extracted parasitics
SDF (standard delay format)	For delay information

 © Cadence Design Systems, Inc. All rights reserved.

Initially, when you start out with your synthesis, you're using the wireload models, but then when you get to your place-and-route, and it is done, you can now generate the extracted data, which is either in the form of a SPEF or an SDF. This extracted data can be passed back to your synthesis tools, which is called *backannotation*. You can send this information back to the synthesis to get more accurate timing data for your net delays. Synthesis can now do a better job of optimizing the rest of the design based on the net timing information.

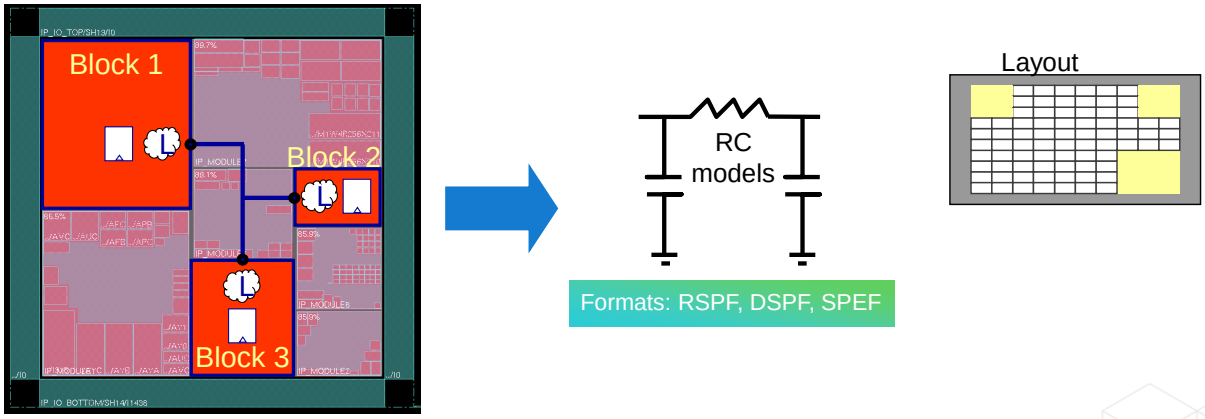
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Backannotation: Extraction

Extraction is the process of measuring the RC values after placement and routing.

- SPEF is the most widely used format.
- Cadence® Quantus™ QRC is a widely used extraction tool.



 © Cadence Design Systems, Inc. All rights reserved.

Extraction tools can create models for your RC network using some sort of compression or reduction mechanism to generate either 2D models, 3D models, or 2 1/2 D models of your layout. These RC models, when reduced, give you either the RSPF or DSPF or the more commonly used SPEF (Standard Parasitic Exchange Format). This process is called extraction. The SPEF, as mentioned here, is the industry-wide standard format. This SPEF is later converted into delays (Standard Delay Format – SDF) by the Delay Calculators. You can take either the SDF to feed back into your synthesis, or the SPEF can be used to feed back into your synthesis. This process is called backannotation.

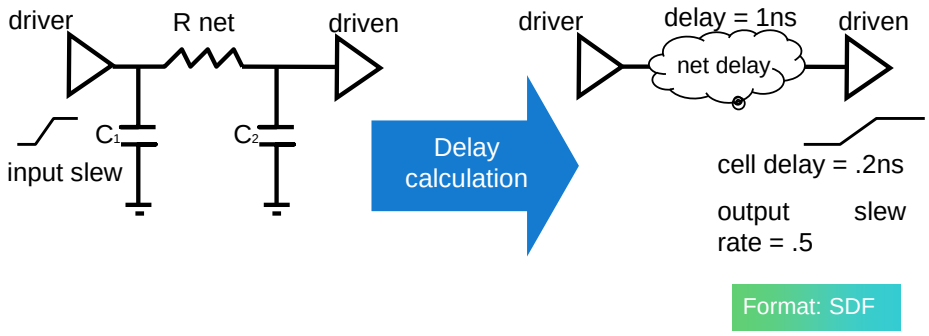
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Backannotation: Delay Calculation

Delay calculation is the process of determining the physically accurate net and cell delays from the RC values.

Nowadays, the timing tools have built-in delay calculators.



 © Cadence Design Systems, Inc. All rights reserved.

One of the steps of backannotation is delay calculation. If we are feeding the tools, and delays of the nets, then we need to have a delay calculator that does that portion of converting your RC network into a delay network. This process is called *delay calculation*. Usually, statistical analysis requires the net delays to be provided either through an SDF or the SPEF, then converted internally through their delay calculators, and then used to plug in the net delays when doing an optimization. Either way, these net delays are calculated, and so delay calculation plays an important piece in your whole backannotation process. The accuracy of your net delays depends on both the extraction tools as well as the delay calculation mechanism that is being used.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Backannotation: Summary of Formats

SDF

- Faster STA runtime because delays are provided
- Smaller file sizes
- Less accurate delay calculation by the STA tool
- Easier to debug timing issues

Detailed SPEF/SPEF/Reduced SPEF

- Slower STA runtime due to delayed calculation in the STA engine
- Larger file sizes
- More accurate delay calculation by the STA tool
- Harder to debug timing issues

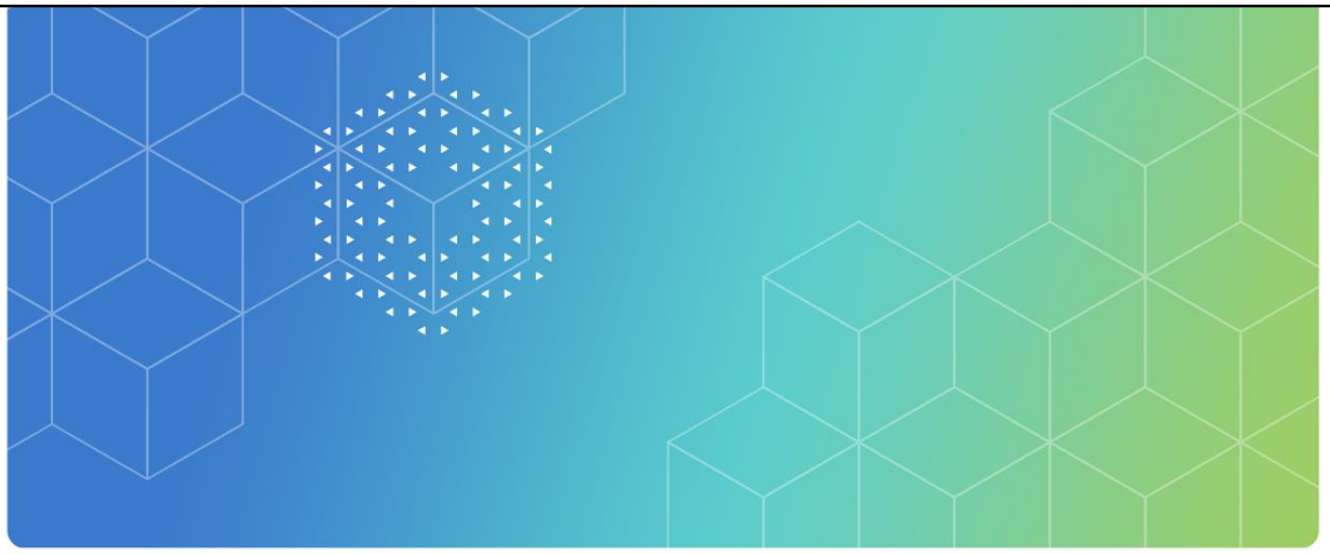
 © Cadence Design Systems, Inc. All rights reserved.

Different formats provide different levels of accuracy and runtimes. Some of the larger file sizes come with longer runtimes, and therefore it may take more time to generate these particular files. When you reduce them and make them simpler, it's faster to read into the tools.

The delay calculation is something that is happening in either the front end or the back end. You can start out with DSPF or with the SPEF and then bring that information directly into your synthesis tool and then let the synthesis tool do the delay calculation. Or you can do the calculation ahead of time and then bring in an SDF. It all depends on what formats are being used. You're trying to justify the runtime with the amount of accuracy that you prefer. So, you generate these kinds of correlation numbers that correlate between your front-end tools and your back-end tools even if you have to sacrifice some accuracy so that you can meet your design goals faster.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail



Module 5

Clocks

In this module, we will focus on clocks.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objectives

In this module, you

- Identify and apply clock period and pulse width
- Calculate the duty cycle of a clock
- Identify clock slew or clock transition
- Identify clock uncertainty
- Apply clock latency

 © Cadence Design Systems, Inc. All rights reserved.

In this module, let's identify what is a clock, how a clock is defined, what is the clock period, pulse width, duty cycle and all the different properties of a clock. We'll also look at how the clock is propagated in a design.

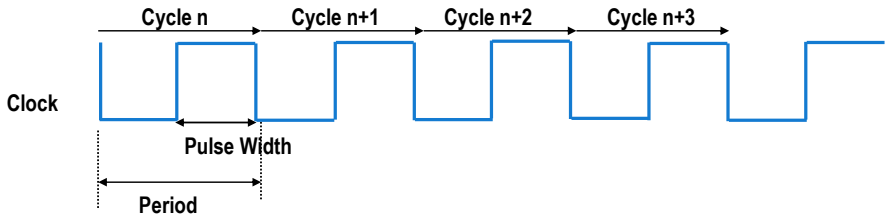
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is a Clock?

A clock is a periodic signal broadcast through a sequential system to synchronize data transfer between storage elements (flops).

After you calculate the path delays, you use the STA tool to time those delays in relation to a clock. The goal is to meet the timing of each path within one clock cycle or period.



Clocks are generated by a crystal oscillator outside of the chip or in a phase-locked loop (PLL) within a chip.

 © Cadence Design Systems, Inc. All rights reserved.

Clocks are periodic signals that are broadcast and therefore are generated either on the chip or outside of the chip. A crystal oscillator can be used to generate clocks outside of a chip. To generate it within the chip, you use phase-locked loops or PLLs. In digital circuits, clocks play a vital role because they trigger the storage of these memory devices and control your state machines.

If you have flops or latches in your design, you're using a clock to trigger the storage of data in those flops and latches. In Verilog language, anytime you see an always@(posedge clock) or always@(negedge clock), or always@(clock), you're creating a storage device. If a posedge clock captures the data into the storage device, then that device is a positive edge-triggered flop. Similarly, the negedge clock stores data into a negative edge-triggered flop. Each clock period has one positive edge and one negative edge. New data can arrive at each capturing clock edge. Therefore, the clock period determines what time constraint a flop has in order to safely store the data before the next data arrives.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

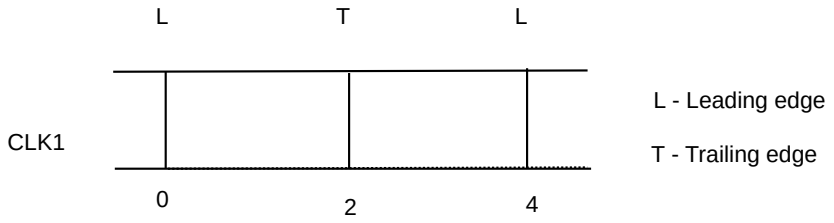
What Is an Ideal Clock?

An ideal clock models a clock signal in an ideal scenario and is plainly a signal that has a leading transition and a trailing transition, where:

- The trailing transition is opposite to the leading transition.
- The leading transition only indicates the first transition, which may be either a rising edge or a falling edge transition.

Example Ideal Clock Specification

```
Period = 4, Edges = {0 2}
```



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

Your design might have different kinds of flops and latches that use different clock edges to trigger the storage of data into the flop or latch. Therefore, it is helpful to idealize the clocks whenever you are unsure of whether the triggering edge is a rising edge or a falling edge. It depends on the type of device on how the clock is used. Therefore, you can idealize the clock with the leading edge as the initial edge and the trailing edge as the following edge. In the graphic above, CLK1 shows a leading edge at 0, which may be a rising or a falling edge. At the time unit 2, you have a trailing edge, and that could either be a rising or a falling edge, depending on how the clock is configured. You can use these ideal clocks for the purposes of our analyses whenever we are unsure of the clock properties.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Features of a Clock

The essential features of a clock include:

Clock period

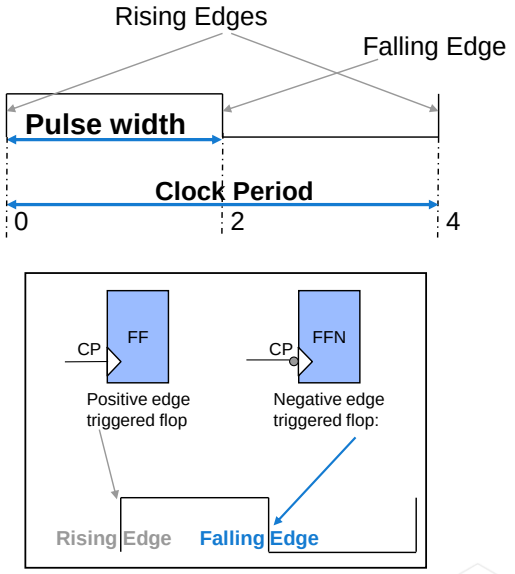
The clock period describes the interval at which the waveform repeats itself.

Rising and falling edges

The rising edge is when the clock signal goes from low to high, and vice versa for the falling edge.

Pulse width

A clock period is divided into times when the clock is high or low. Pulse-width is the amount of time that the clock is high (or low for pulse-width low).



 © Cadence Design Systems, Inc. All rights reserved.

The clock has multiple properties. One of them being the clock period, which defines at what interval the waveform repeats itself, and then you have the rising edges and the falling edges. The rising edge is when the clock is going from low to high, the falling edge is when the clock is going from high to low. The pulse width is determined based on how much the clock is high.

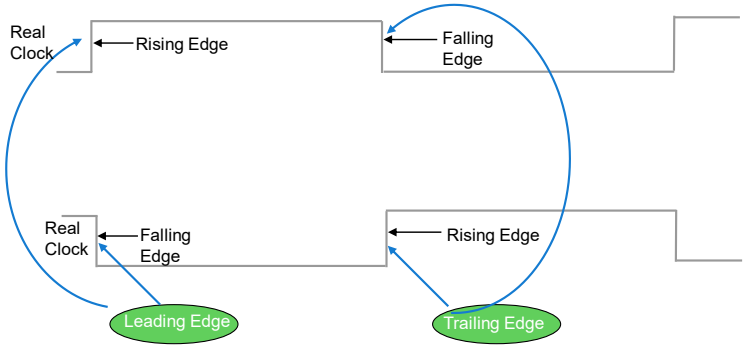
© Cadence Design Systems, Inc. All rights reserved.

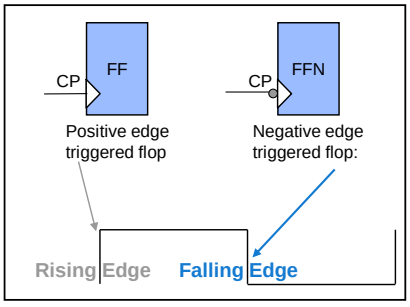
This document is for the sole use of Le Khanh Nam of Gmail

Clock Association

The polarity of the signal clock is defined when an input port is associated with the ideal clock.

These real clocks are positive if the rising edge is tied to the leading edge.





They are negative if the falling edge is tied to the leading edge.

 © Cadence Design Systems, Inc. All rights reserved.

When you associate those ideal clocks with an actual port, if the port is driving only positive edge-triggered flops, then a rising edge can be set for the initial (leading) edge of the clock. If the port is driving negative edge-triggered flops, then a falling edge can be the leading edge. If you have a mixed bag of positive and negative edge-triggered flops in your design, it is best to define separate clocks for the two types of flops. Otherwise, it requires a very careful analysis of the various launch and capture edges of the clock to determine the proper storage of data into the flops.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Understanding the Duty Cycle of a Clock

Duty cycle is defined as the ratio of active pulse width to the entire clock period.

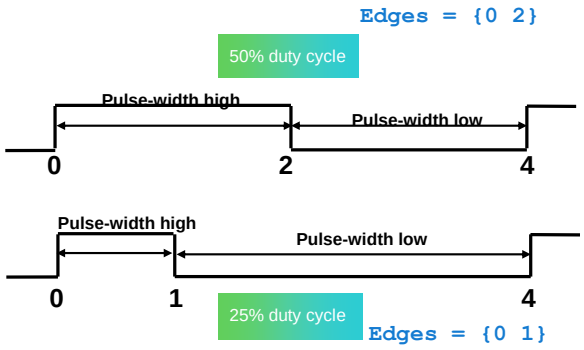
You do not need to define the duty cycle. It is inferred from the edge specification.

Example

For a clock with period 4 ns and an edge specification of {0 2}, the duty cycle is calculated as follows:

```
Clock Period = 4, Edges = {0 2} or {0 1}
Pulse-width high = 2-0 = 2
Duty Cycle = (Pulse width high)/(Clock Period) = 2/4 = 50%
```

Because the first edge specified is the rising edge, the portion of the clock cycle from 0 to 2 is pulse-width high.



 © Cadence Design Systems, Inc. All rights reserved.

In some designs, it is important to understand the duty cycle because when there is a transfer of data from positive edge-triggered to negative edge-triggered flops or vice versa, there is a reliance on the duty cycle. This is also important in the case of a latch. Whether the latch is transparent or opaque is determined by the duty cycle. In most designs, where there are just positive edge-triggered flops or just negative edge-triggered flops, the duty cycle doesn't play much of a role.

Let us look at the following examples. Here, a clock is defined by its period of 4 and the edges of 0 and 2. Duty cycle is calculated as (pulse width high) divided by (clock period), as shown in the top waveform, which is 2 minus 0 divided by 4. Therefore, we have a 50 percent duty cycle. On the bottom waveform, the edges are defined at 0 and 1. Therefore, the pulse width is high only for one unit, and so it's a 25 percent duty cycle.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Duty Cycle

What is the duty cycle of a clock with a period of 5 units, and a waveform represented by {1 3}?

In a minute, the answer will appear on the page.
Move the time slider to the end, if you want to see the answer sooner.
Solutions are also in the Notes page.

 © Cadence Design Systems, Inc. All rights reserved.

Try this activity to improve your understanding of the topics discussed.

Solution

Duty Cycle = (Pulse width high)/(Clock Period) = (Falling Edge – Rising Edge)/Period

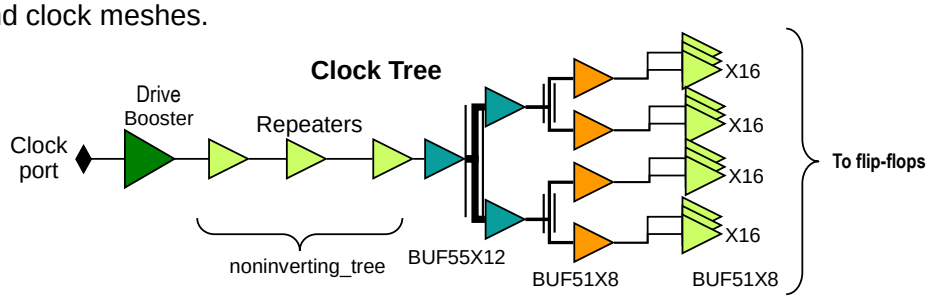
Therefore, Duty Cycle = (3 – 1)/5 = 40%

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Clock Propagation

Clocks traverse through the entire design to reach several flops and latches. Therefore, the signal is buffered to keep its integrity. These buffers form different structures. The most common forms are clock trees and clock meshes.



Ideal Mode: When there are no clock buffers in the design, you model the effect of these buffers for static timing analysis. This modeling stage is called the ideal mode.

Propagated Mode: When clock buffers are inserted into the design, the STA tools replace the modeled clock information with the actual clock information. This stage is called the propagated mode.

 © Cadence Design Systems, Inc. All rights reserved.

In any design, a clock triggers several flops and latches. These clocks are considered high fan-out nets, and they need to reach or traverse the entire design to reach these flops and latches. To keep the integrity of the clock signal, it needs to be constantly buffered, and these buffers can be in different structures. The most common forms are known as clock trees or clock meshes, and so you would see different types of buffers along the clock tree and or clock mesh. These are generally referred to as clock trees. You might start out with a drive booster and a bunch of repeaters, then different kinds of buffer strengths placed along the length of the tree traverses to reach the flops or latches. In the case where your design does not have any clocks typically during synthesis, the clock tree doesn't exist, so you need to mark the effect of a clock tree into your design. At this stage, during synthesis, it's known as the ideal mode. When the clock buffers are inserted into the design, it is known as the propagated mode, where the modeling that we have done is replaced with the actual clock information.

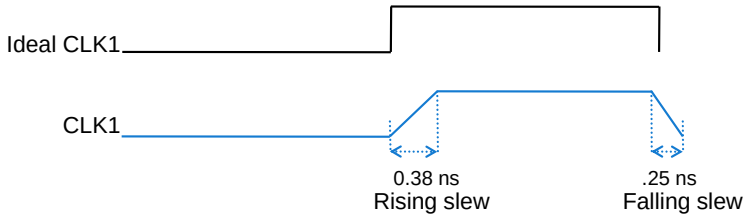
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Clock Slew (Transition)

Transition time is the time the signal takes to change states from active high to active low or vice versa. It is also known as *clock slew*.

- You need to set the clock slew in ideal mode.
- You remove the clock slew in propagated mode because clock slews are automatically calculated by the STA tools after clock buffers are added during place and route.



 © Cadence Design Systems, Inc. All rights reserved.

As we've seen before, the transition time is the time taken by a signal to change states. In the case of a clock, this is known as the clock slew, and clock slew is essential in ideal mode. We need to define the clock slew in ideal mode because ideal clocks typically don't have any rise and fall times; it’s instantaneous. Whereas once the clocks are inserted, then the STA tools will calculate the slews of the clocks through the buffers that are inserted.

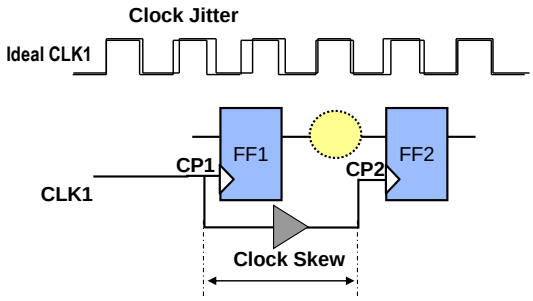
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Understanding Clock Uncertainty

From cycle to cycle, the period and duty-cycle can change slightly due to clock generation circuitry. This is known as *clock jitter*. The clock signal arrives at different components at different times because of clock trees and buffers. The difference in clock arrival times is known as *clock skew*.

Uncertainty in the arrival of your clock edges adds pessimism to your design requirements. Thus a combination of clock jitter and clock skew is used as a value for clock uncertainty.



Clock Uncertainty = Clock jitter + Worst clock skew (setup)

Clock Uncertainty = 0 + Best Clock Skew (Hold)

 © Cadence Design Systems, Inc. All rights reserved.

Because we're talking about real clocks, these clocks are generated by clock oscillators or PLLs. There is an uncertainty in the period and the duty cycle of the clock, because of the circuitry that is generating these clocks. This is considered as clock jitter. Each clock oscillator comes with a datasheet, in which the clock jitter is listed.

Your clock also has a slight variance in the clock arrivals at different leaf flops in the design, because of the clock trees and buffers in your clock path. Ideally, these clock trees are modeled to reduce variance of the clock arrival at the different leaf flops, but it is inevitable because it is a real design where data transfers between flops happen across various flops in the design. This variance in clock arrival times due to the clock trees is called clock skew. Clock uncertainty can be thought of as a combination of clock skew and clock jitter. The STA tools report clock uncertainty as calculated for all the different clocks in your design. You can use clock uncertainty to add pessimism to your design, especially for paths that start from one clock to another.

© Cadence Design Systems, Inc. All rights reserved.

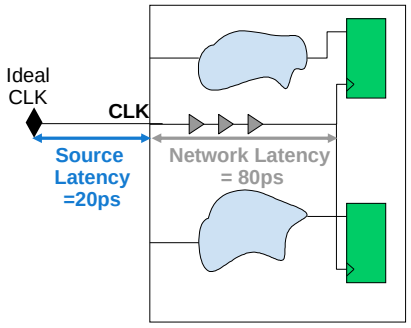
This document is for the sole use of Le Khanh Nam of Gmail

Modeling Clock Latency

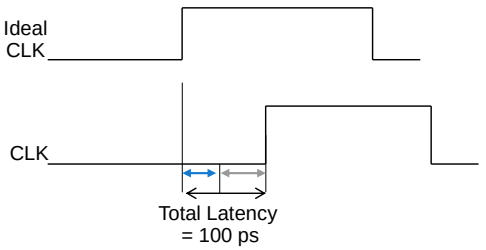
Clock latency is the time taken by a clock signal to propagate from the clock definition point to a register clock pin. Latency is also known as insertion delay.

There are two types of latency:

- Source Latency: Clock source to the clock port in the design.
- Network Latency: Clock port to the registers (clock tree delay).



Total Latency @ clock port/pin = Clock source latency + Clock network latency



 © Cadence Design Systems, Inc. All rights reserved.

You need to model the clock arrival times at different flops and latches in your design to determine the required time for that signal to reach the flop. Clock arrival times are modeled via clock latencies. You need to define two types of latencies, source latency, and network latency. Source latency is measured up to the point of entry of the clock into the design or block. Network latency is measured inside the design. So, any buffering you do inside your design is included in the network latency. Therefore, the total latency at any given flop or latch is the combination of source latency and network latency. In ideal mode, you model the effect of clock buffers using these clock latencies. In propagated mode, you can remove network latency. Clock latencies are commonly referred to as insertion delays.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Clock Latency

Find the total latency of a clock with these parameters:

- Maximum clock tree delay = 100 ps
- Maximum delay from the clock source to the clock port = 300 ps

In a minute, the answer will appear on the page.
Move the time slider to the end, if you want to see the answer sooner.
Solutions are also in the Notes page.

 © Cadence Design Systems, Inc. All rights reserved.

Take a minute to do this activity, to improve your understanding.

Solution

Clock network latency = Clock tree delay = 100 ps

Clock source latency = 300 ps

Total clock latency = Clock network latency + Clock source latency = 400 ps

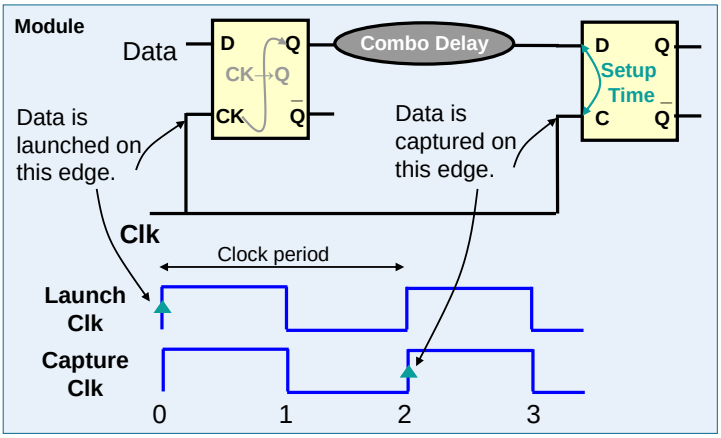
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Understanding Launch and Capture Clock Edges

The launch clock is the clock signal that triggers the data. The launch edge is the edge of the launch clock, which triggers the data into the initial flop.

The capture clock is the clock that captures the data. The capture edge is the edge of the capture clock, which allows the data to be stored in the receiving flop.



 © Cadence Design Systems, Inc. All rights reserved.

When a signal moves from one flop to another, the signal is considered launched from one flop and captured on another flop. The clock that launches the data is called the launch clock, and the edge of the clock that launches that data is called the launch edge. The capture clock is the one that captures the data, and the edge that captures that data is called the capture edge. In this design, there is only one clock that is both launching the data on one rising edge and capturing the data at a different rising edge. In this case, the difference between the capture edge and launch edge is one clock period. Because of the delays in the path, you cannot capture it instantaneously after it is launched. It takes one clock period for the data to be launched at one edge and captured at another edge on a different flop. The data can change after one clock period but not before it is captured. Therefore, one clock period is required for this design to safely store the data before it changes. There are also other requirements before a data signal is safely stored in the flop. You must determine whether the data has arrived at the right time and if it satisfied those requirements called setup and hold. We will talk about setup and hold later.

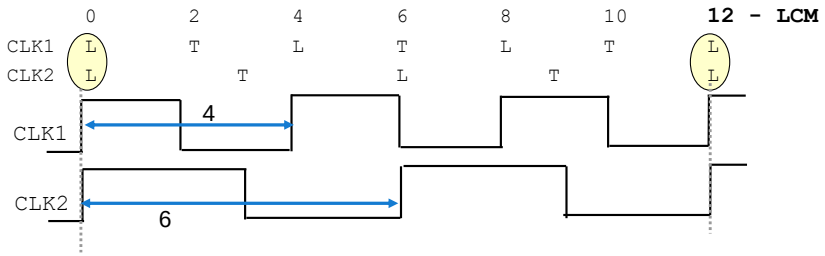
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Multiple Clocks

The worst-case launch and capture edges are not necessarily intuitive when there are multiple clocks. You need to follow some steps before you determine worst-case data transfer in multiple clock scenarios.

- Find the least common multiple (LCM) of the clock periods, also known as the base period.
- Align the leading edges of both clocks.
- Unroll the ideal clocks to find the most restrictive *non-zero delta* between the two edges of interest.



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

When you have multiple clocks in the design, and you have data transferring from one clock to another, the worst-case launch and capture edges are not intuitive. So, you'll need to follow certain steps to determine the worst-case data transfer.

First, you need to determine the least common multiple of these clock periods. In the example shown, the clock periods are 4 and 6, and they have a least common multiple (LCM) of 12.

Next, you align the leading edges. The leading edges of both clocks are aligned.

Next, you unroll the clock until the leading edges of both clocks meet. The LCM would be where the leading edges align again. In this example, as shown in yellow, the leading edges align at 0 and again at 12. The entire waveform would repeat itself in the same pattern, with the leading edges aligning again and again after every 12 units. It is easier to locate all the different launch and capture edges when aligned as such. Every transfer pattern can be replicated here because the entire waveforms would repeat itself.

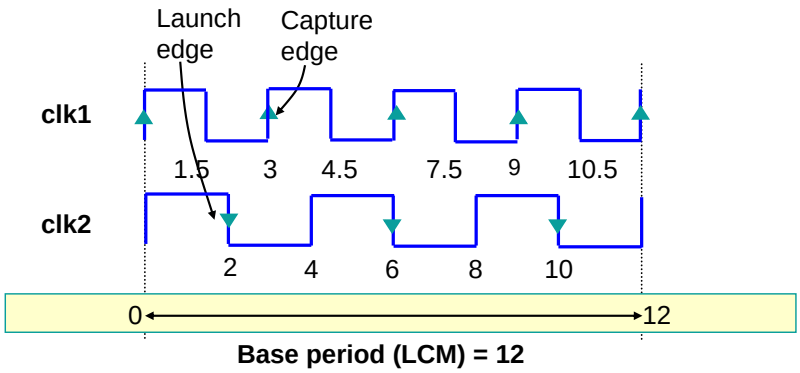
Finally, you determine the worst-case transfer by locating the most restrictive launch and capture edges in the unrolled waveforms. So now you can locate all the data transfers that can happen between the two clocks and find the most restrictive or the worst-case scenario.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Examples of Launch and Capture Edges

In a design with positive clocks clk1 {0 1.5} and clk2 {0 2}, with clock periods of 3 and 4 units respectively, the LCM or base period is **12**.



For **clk2 falling** to **clk1 rising** data transfer, the most restrictive is $6^{v} \rightarrow 6^{\wedge}$ or $2^{v} \rightarrow 3^{\wedge}$ but $6^{v} \rightarrow 6^{\wedge}$ is unrealistic, as it doesn't account for any data delays.

 © Cadence Design Systems, Inc. All rights reserved.

Let's look at an example with multiple clocks. If you have clocks with periods of 3 for clock1 and 4 for clock2. The LCM of the clock periods is 12.

First, align the leading edges of the clocks at 0. Since both are positive clocks, we align the rising edges, which also happen to be the leading edges at 0. Then we unroll the clocks up to 12 units.

Next, look at all the edges and identify all the clock2 falling launch to clock1 rising capture transfers. Notice clock2 falling edges at 2. 6, and 10. Notice clock1 rising edges at 0, 3, 6, 9, and 12. For setup checks, which we will talk about in detail later, you'll have to identify the next closest capture edge. For 2, it would be 3. For 6, it would be 6 or 9. For 10, it would be at 12.

Then, identify the worst-case scenario amongst them. You may be tempted to say clock2 is falling at 6 to clock1 rising at 6. But since it doesn't account for data delays, you can ignore that one. You can then say the next worst-case scenario is clock2 falling at 2 to and clock1 rising at 3.

The worst-case scenario and the corresponding launch and capture edges are shown in the graphic.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

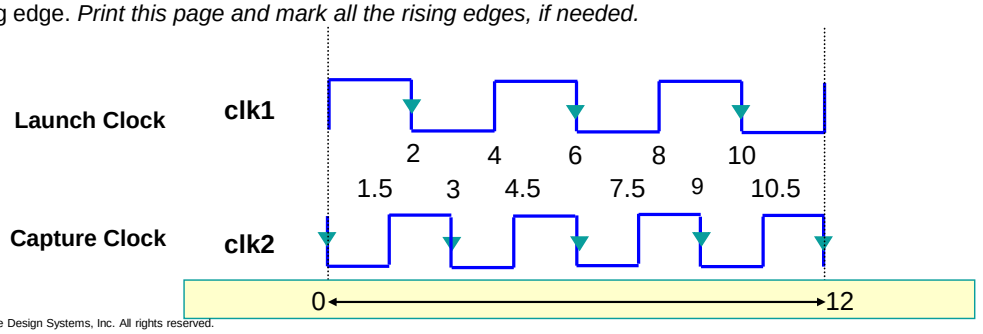
Activity: Launch and Capture Edges

The following are the parameters for this activity:

- The clock period of clk1 is 4ns.
- The clock period of clk2 is 3ns.
- The duty cycle of both clocks is 50%.

What is the worst-case launch and capture edges for data transfers from rising *clk1* to rising *clk2 for setup*?

Tip: Setup check uses the closest edge after launch for capture. Notice that the CLK2 is a negative clock, and the leading edge is a falling edge. *Print this page and mark all the rising edges, if needed.*



 © Cadence Design Systems, Inc. All rights reserved.

Try this activity to improve your understanding.

Solution

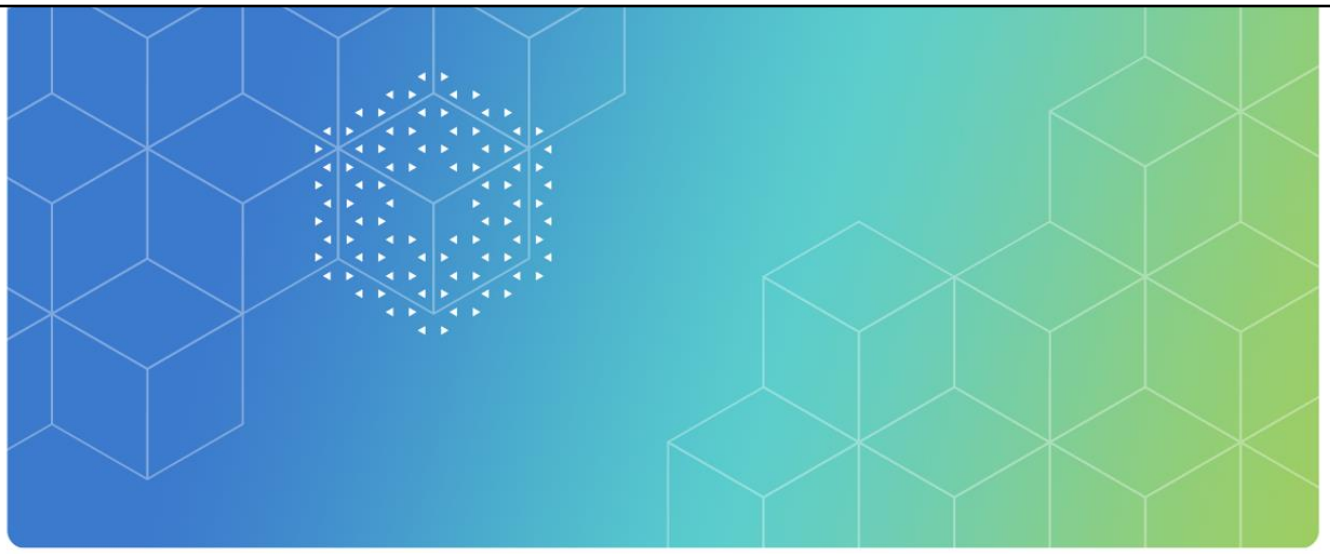
The setup check uses edge after the worst-case launch edge is 4, and the worst-case capture edge is 4.5.

Hold check uses the edge before the worst-case launch edge is 8, and the worst-case capture edge is 7.5.

We will talk about *setup* and *hold* later in the course.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail



Module 6

Timing Checks

Let's look at timing checks.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objectives

In this module, you

- Identify the setup and hold information from libraries
- Identify how setup time affects flops
- Apply setup checks
- Apply hold checks
- Identify phase shift

 © Cadence Design Systems, Inc. All rights reserved.

In this module, let's identify what the setup and hold checks imply, how it affects the flops, and how information is stored in the flops. Then we will identify how they are stored in the libraries for the STA tools to use.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Are Flip-Flops?

A flip-flop is a storage device that is edge-triggered. It is also commonly referred to as a flop or a register.

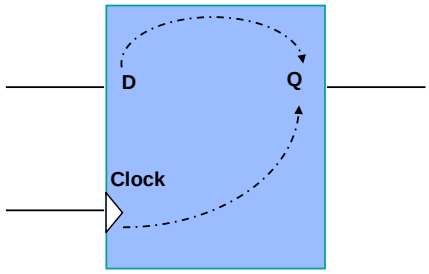
In a flop, the active clock “edge” triggers the input to the output. Therefore, flop timing is calculated from the clock edge, and the **clock-to-Q** delay is used for flop timing.

There are two essential timing arcs in the flip-flop:

- D-to-Q
- Clock-to-Q

The important timing parameters of a flop are:

- Clock-to-Q delay
- Setup time
- Hold time



Flip-Flop

 © Cadence Design Systems, Inc. All rights reserved.

A flop or flip-flop or register, as it is commonly known, is one of the common storage elements, and it is edge-triggered. If we look at the timing arcs in the flop, you have the D to Q timing arc, and then you also have the Clock to Q timing arc. One thing to remember is that in an edge-triggered flop, the input data is propagated to the output only when the clock triggers the data to go from D to Q output. So, all the timing information that is calculated is from that clock edge. When you look at the timing calculations, the Clock to Q delay of the Clock to Q timing arc is what is used in timing. STA tools will use that Clock to Q delay. Whenever you look at timing reports, the delays are calculated from the clock edge. So, the important timing parameters in a flop are the Clock to Q delay. We also have setup and hold time requirements. We'll talk about those in detail over the next few slides.

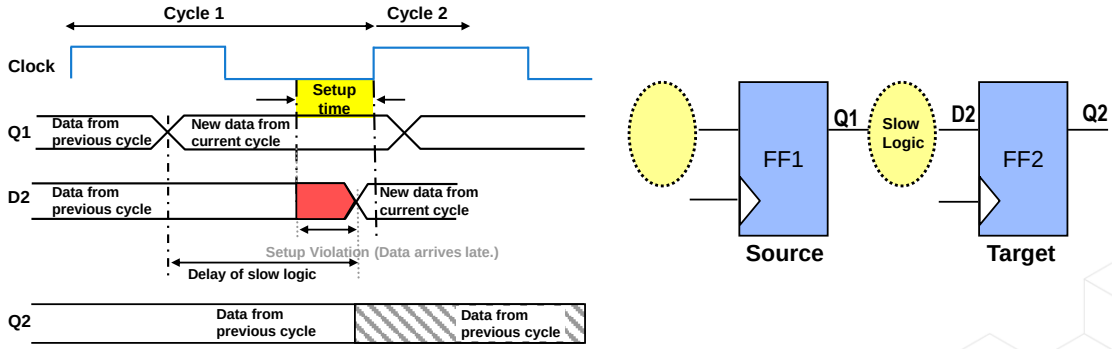
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Understanding Setup Time

Setup time is the duration of time that the synchronous input data must be stable before the triggering edge of the clock.

- If setup time is not met, data is not stored in the flop.
- Setup violations can be remedied by either slowing down the clock (increasing the period) or by decreasing the delay of the data path (slow logic).



 © Cadence Design Systems, Inc. All rights reserved.

For the data to be safely stored inside a flop, the data must be stable for it to trigger the saving of the data into the flop. If the data is not stable and changes during the setup time, the data is not stored in the flop. If you look at the graphic below, there is some slow logic, and the data is going from Q1 to D2. The data is being transferred from FF1 to FF2. If you look at the D2 signal, it changes before cycle2 of the clock starts. The new data is ready to be saved into FF2 at the rising edge of clock cycle2. But the D2 signal just changed value right before the clock edge without meeting the setup time requirement. Therefore, the data will not be stored in FF2.

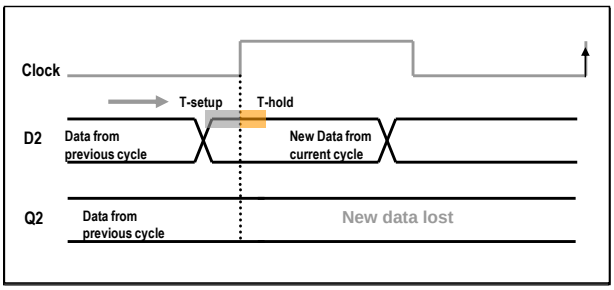
The reason for this is the slow logic. Obviously, Q1 and D2 are different signals, and even though they have similar waveforms, there is a delay between them. The Q1 signal takes some time through the slow logic to arrive at D2. Q1 data from the previous cycle changed before cycle2 begins because the rising edge for the clock in cycle1 triggered new data into FF1, thus changing Q1. And now the new data is already in Q1. The old data barely arrived at D2 and was not yet saved into the target flop FF2 before changing into the new data. Thus, by not meeting the setup time of the flop, it caused a setup violation.

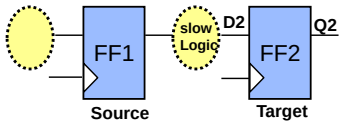
Therefore, setup time can be defined as the duration of time that the input data must be stable before the triggering edge of the clock. Setup violations can be remedied by either slowing down the clock (by increasing the period) or by decreasing the delay of the data path (or slow logic).

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setup Time Violations: Slow Data





What happens when D2 data is slow to arrive?

- Setup time is violated
- New data is lost

 © Cadence Design Systems, Inc. All rights reserved.

A positive edge-triggered D-flop stores and holds the old data until the next positive edge of the clock arrives. In this case, you have two flops, with the data launching from the FF1 flop and captured by the FF2 flop. At this point, in the FF2 target flop, when the data from the previous cycle arrives at D2, it can enter the flop even though the clock is low but is not fully stored because of the delay inside the flop and it waits for the rising edge to store the data inside the flop.

What happens when D2 is slow to arrive?

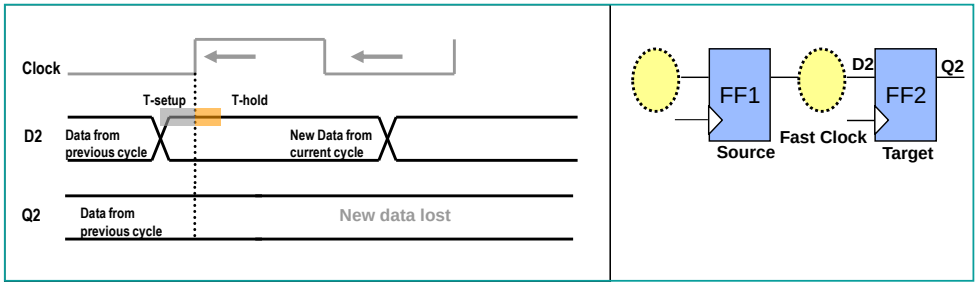
If the data from the FF1 flop takes too long to get to FF2, then by the time FF2 is ready to store the data from the previous cycle, the data starts changing into the new data from the current cycle. Because the new data possibly conflicts with the not-fully-stored old data, it creates a race condition between the old data already inside the flop that is not yet fully stored and the new data just arriving at its input. Therefore, the stored output of the capture flop is unknown.

Therefore, when your data is slow to arrive and the setup time is violated, the new data is considered lost or unknown. To avoid that situation, the data must arrive in time for the minimum setup check must be met and for the data to be stored correctly in the capture flop.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setup Time Violations: Fast Clock



What happens when you speed up the clock?

- Setup time is violated
- New data is lost

 © Cadence Design Systems, Inc. All rights reserved.

When you have a really fast clock or when you speed up your clock, your setup time is violated because now the data becomes slower compared to your clock. The new data is actually lost. For good functioning circuitry, your clock cannot be too fast compared to your data.

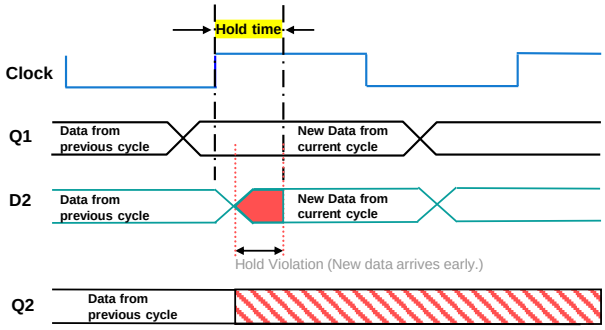
© Cadence Design Systems, Inc. All rights reserved.

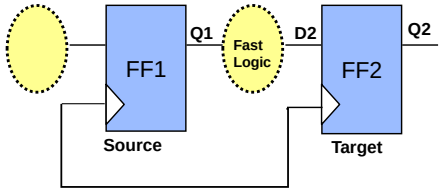
This document is for the sole use of Le Khanh Nam of Gmail

Understanding Hold Times

Hold time is the duration that the synchronous input (D) must be stable after the triggering edge of the clock.

The data delay must be greater than the hold time of the target flop plus the clock uncertainty between the flops for the data to be safely stored in the target flop.





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

For your data to be safely stored, the other requirement is your hold time. The hold time is the duration that the D input must be stable after the triggering edge of the clock. So, the data must be stable. The data delay must be greater than the hold time of the target flop plus any clock uncertainty between the flops. This means your data cannot be too fast compared to the clock. You can correct the hold violations by providing some data delay. You can either decrease your clock uncertainty in the design, or you can also add delay cells. You can reduce the amount of skew between the clock trees by using better clock tree generation. If you constrain your clock trees properly, there won't be any hold violations because that minimum data delay requirement is now met. If we look at this illustration, the D2 input is switching when the old data is being captured into the second flop. Therefore, the data is not captured safely into the new flop.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Hold Time Violations: Fast Data Change

What happens when D2 data starts changing immediately?

- Hold time is violated
- New data is lost even though it is caught within clock edge

 © Cadence Design Systems, Inc. All rights reserved.

What happens when the data changes immediately? You'll have a hold time violation. And even though it is caught within the edge, the data is still lost because the flop didn't even fully absorb the previous data into the flop before the data changed a second time. For the data to be safely stored in the flop, the new data must not arrive so fast that the flop doesn't even absorb the data from the previous cycle. Therefore, the hold checks are done at the same edge as the launch edge rather than doing it at the capture edge, like in the case of a setup check. If the data doesn't change faster than the hold check, it will most likely be safely captured into the flop unless there is a huge skew between the flops. This is why it is important to invest effort into building well-balanced clock trees.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Library Setup and Hold Checks

This illustration shows a library cell and its setup and hold lookup tables.

Setup from a Table Lookup

Hold from a Table Lookup

```
cell (DFFQX2) {
  cell_footprint : dffq;
  area : 27.1584;
  pin(D) {
    direction : input;
    capacitance : 0.001172;
  timing() {
      related_pin : "CK";
      timing_type : setup_rising;
      rise_constraint(setup_template_3x3) {
        index_1 ("0.042, 0.392, 1.5");
        index_2 ("0.042, 0.392, 0.764");
        values ( \
          "0.085938, 0.054688, 0.054688", \
          "0.125000, 0.085938, 0.078125", \
          "0.125000, 0.070312, 0.070312");  }
  }
    timing() {
      related_pin : "CK";
      timing_type : hold_rising;
      rise_constraint(hold_template_3x3) {
        index_1 ("0.042, 0.392, 1.5");
        index_2 ("0.042, 0.392, 0.764");
        values ( \
          "-0.046875, -0.023438, -0.031250", \
          "-0.078125, -0.054688, -0.054688", \
          "-0.062500, -0.031250, -0.031250"); }
  }
```

 © Cadence Design Systems, Inc. All rights reserved.

The setup and hold checks are stored in your libraries, the .libs, in the form of a lookup table. The lookup table contains the setup and hold values that are calibrated at several values of load and skew for each of the flops. The setup is shown as a positive value. The signal timing to the flop has to meet or exceed that setup value, which is done as a check at the capture flop during timing analysis. For the hold, the values are shown as a negative value. Both setup and hold have to be met for any particular capture flop to safely save the data into the flop.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Timing Checks

The period of the only clock in a design is 100 ps. The minimum setup of all the flops in the design is 150 ps.

The design is constantly not meeting timing during static timing analysis.

What seems to be the problem?

 © Cadence Design Systems, Inc. All rights reserved.

Take a minute to try this activity.

Solution

The minimum setup is 150 ps, but the clock period is only 100 ps. You need to increase the clock period to meet at least the setup of the flops, if nothing else.

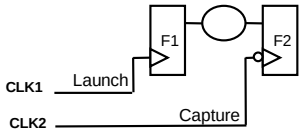
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

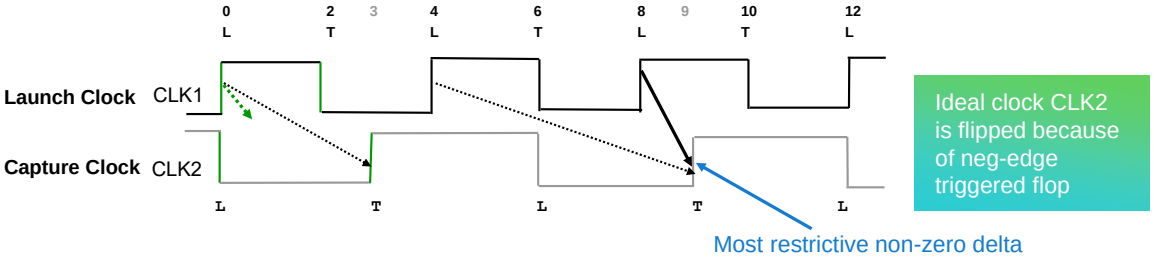
Multiple Clock Domains: Setup Check

Triggered by leading edge of CLK1

Checked with trailing edge of CLK2



Late Relationship



- Path starting with a leading edge of CLK1 and ending with a trailing edge of CLK2
- Time zero launch to capture: leading 0.0, trailing 3.0
- Most restrictive launch to capture: leading 8.0, trailing 9.0 (Positive for late)

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

When you have multiple clocks in the design, look at the capture clock and the capture flop during your setup checks. Even though there is CLK1 and CLK2, CLK2 is the one that determines the safe storage of the data in the F2 here. Make sure that when the setup checks are run, the setup check is happening on F2 and not F1 because there are different edges where the launch and capture can happen. In the case of multiple clock domains, we do need to look at the most restrictive capture and launch combination. If we look at the time 0 launch, the capture will occur at the trailing 3. Whereas if we look at the leading edge 8 launch, the capture will happen on CLK2 at the trailing 9. This seems to be the most restrictive or worst-case combination. We have to look at that worst-case combination of launch and capture to make sure that the design meets the timing or the setup check for that particular restrictive capture and launch combination.

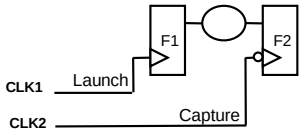
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

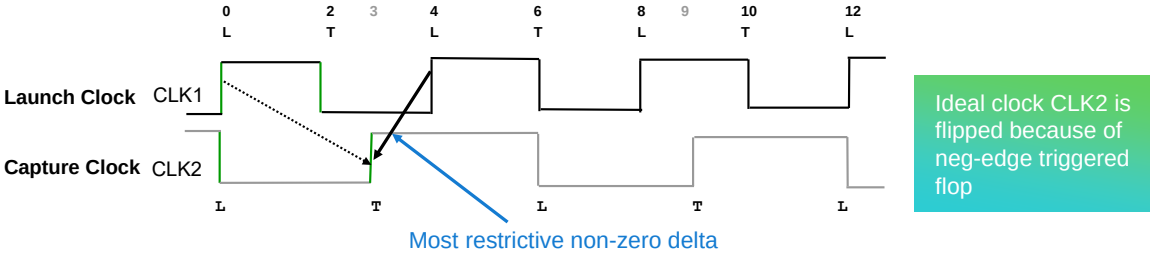
Multiple Clock Domains: Hold Check

Triggered by leading edge of CLK1

Checked with trailing edge of CLK2



Early Relationship



- Path starting with a leading edge of CLK1 and ending with a trailing edge of CLK2
- Time zero launch to capture: leading 0.0, trailing 3.0
- Most restrictive launch to capture: leading 4.0, trailing 3.0 (Negative for early)

 © Cadence Design Systems, Inc. All rights reserved.

The hold check is a little more complicated. We have to understand that whatever edge is launching the data, if the next edge is not launching the data again before the previous one is saved, then you're okay. So, if you launch at 0 and at the trailing edge of 3, if the data has not completed saving into the F2, then you don't want the new launch at 4 to give you new data. You would want it to wait until the data is safely stored. If we look at the launch 4, the trailing edge at T3 still hasn't completed saving. That is necessarily the most restrictive edge, and therefore the hold requirement is done at T3 for the hold check. So, if we look at the most restrictive launch to capture its 4 and 3, and the data is safely stored, then you're okay. Otherwise, you're failing the hold check.

© Cadence Design Systems, Inc. All rights reserved.

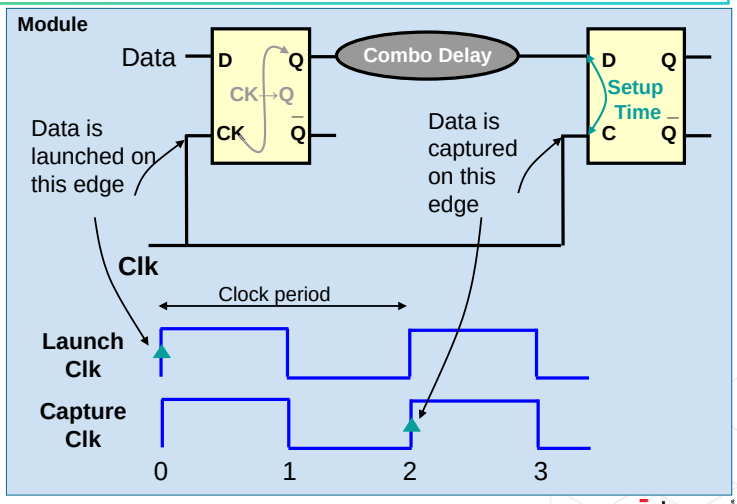
This document is for the sole use of Le Khanh Nam of Gmail

Phase Shift

Phase shift is the least amount of time difference between the capture clock edge and the launch clock edge.

Some STA tools use an amount equal to phase shift as the required time for the capture clock signal to safely store the data in the register.

In single-clock designs, the setup phase shift is one clock period, and the hold phase shift is zero.



 © Cadence Design Systems, Inc. All rights reserved.

In timing reports, some tools report the difference in the most-restrictive launch and capture edges as phase shift. This can be helpful in calculating the time required for the data to arrive at the capture flop. Phase shift is a concept that helps create better timing reports when you have separate clocks for launch and capture. But, even in the case of using the same clock for both launch and capture, as shown in this example here, some tools separately report a phase shift of one clock period and then include the phase shift of one clock period in the required time calculation for setup.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Basics: Phase Shift

- Timing analysis starts at time zero.
  - The start and end times for paths are referenced from the first clock edges from time zero, even if the most restrictive edges are at different times.
- Phase shift is the delay adjustment used so that the first clock edges represent the most restrictive timing edges.
- Phase shift is based on the ideal clock edges, not on the real signal edges.
- **Late (setup) phase shift:** Unroll the ideal clocks to find the most restrictive non-zero delta between the two edges of interest.
  - The capture edge occurs after the launch edge.
- **Early (hold) phase shift:** Unroll the ideal clocks to find the most restrictive non-zero or delta between the two edges of interest.
  - The capture edge occurs before or equal to the launch edge.

 © Cadence Design Systems, Inc. All rights reserved.

For multiple clocks, we must understand the basics of phase shift. Let's look at how phase shift works. Phase shift is based on ideal clock edges and not on the real signal edges. First identify the launch clock and capture clocks. Then start your timing analysis at time zero. Identify the worst-case launch and capture after unrolling the clocks. Most likely, these most-restrictive launch and capture edges are not in the first cycle. But you can visualize that worst-case scenario, like it happens in the very first clock cycle by shifting the capture clock until the same restrictive transfer also happens in the first cycle. Then you record the amount of shifting you did to the capture clock as phase shift. Thus, the phase shift can also be defined as the delay adjustment used so that the first clock cycle edges represent the most restrictive timing edges. For setup or late phase shift, the capture edge occurs after the launch edge, so we must unroll the ideal clocks, try to find out the most restrictive non-zero delta and shift it to the first cycle. Whereas with the hold, the capture edge happens before or equal to the launch edge, but the same unrolling happens, and we must figure out what the most restrictive non-zero delta is, then we shift it to the first cycle.

© Cadence Design Systems, Inc. All rights reserved.

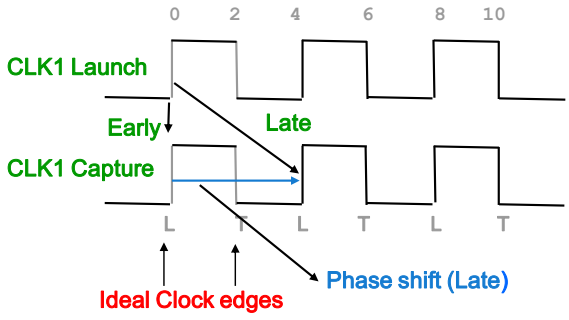
This document is for the sole use of Le Khanh Nam of Gmail

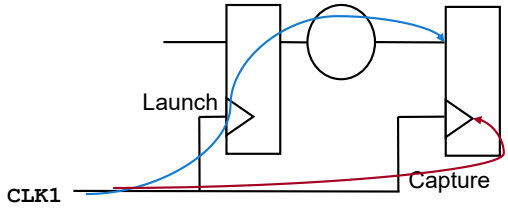
Calculating Phase Shift

- Phase shift represents the delay adjustment being used for the projection of the most restrictive non-zero delta time into the first clock cycles of the unrolled clocks. The following equation is valid for all phase shift calculations (late and early, setup and hold, respectively):

$$(CE - LE) - (C1 - L1)$$

- Where CE represents the capturing clock edge, LE represents the launching clock edge, C1 represents the corresponding capture edge of the first cycle, and L1 represents the corresponding launch edge of the first cycle. (See notes)





 © Cadence Design Systems, Inc. All rights reserved.

There is a simple equation that works to calculate phase shift for both setup and hold. The way it works is to identify the most restrictive launch and capture edges. Then identify the first capture and launch edges from your first cycle. Once you know those four edges, you plug the time units at which they occur into the equation (CE minus LE) minus (C1 minus L1), giving you your phase shift value.

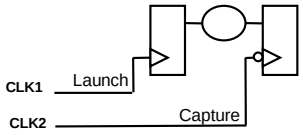
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

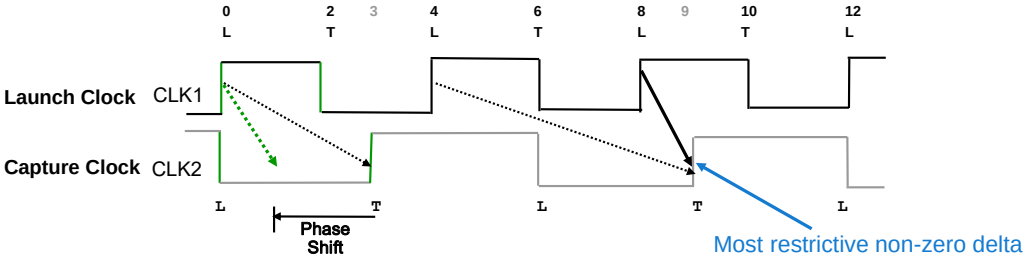
Multiple Clock Domains: Phase Shift for Setup

Triggered by leading edge of CLK1

Checked with trailing edge of CLK2



Late Relationship



- Path starting with a leading edge of CLK1 and ending with a trailing edge of CLK2
- Time zero launch to capture: leading 0.0, trailing 3.0
- Most restrictive launch to capture: leading 8.0, trailing 9.0 (Positive for late)
- Phase shift is ((9.0 - 8.0) - (3.0 - 0.0)) = -2.00

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

Let's look at an example of phase shift calculation for set up in a multiple clock domain scenario. If we unroll a 4ns clock and a 3ns clock, the LCM (least common multiple) is 12. For a negative edge triggered flop, what we're doing instead of using the CLK2 as-is, we are reversing the phase of CLK2. Notice that the leading edge of CLK2 is a falling edge. Looking at ideal clocks here, we align the leading edge to the leading edge after unrolling. Then we can find and determine that the most restrictive non-zero delta would be at 8 launches and 9 capture.

In the first clock cycle, if you choose the launch edge at 0, then the first trailing edge at 3ns is the capture edge. This has a difference of 3ns. Notice that the launch edge is a leading edge and a rising edge of CLK1. Notice that the capture edge is a trailing edge and a rising edge of capture clock CLK2. If you choose a launch edge of 8ns from the launch clock CLK1, then choose the capture at 9ns. This is the most restrictive launch to capture because the launch to capture has the smallest difference of 1ns. If you want, go ahead and verify all other launch to capture scenarios to confirm this conclusion. Let's pretend that this same scenario is happening in the first clock cycle. To do this, move the capture clock waveform so that the most restrictive scenario happens in the first clock cycle. In this case, we can move the CLK2 waveform to the left by 2 units, and we can see the most restrictive scenario is in the first clock cycle. These 2 units are considered phase shift because we are moving the waveform by 2 units. So, if you arrive at the phase shift via calculation, it is also the same value of -2. Therefore, this -2ns is the value for phase shift for setup.

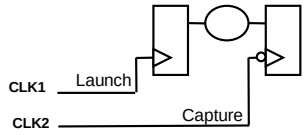
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

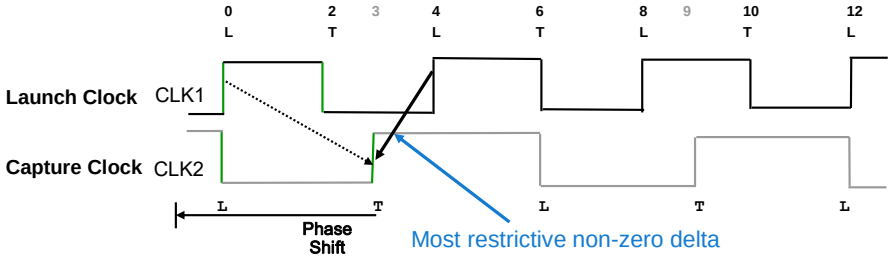
Multiple Clock Domains: Phase Shift for Hold

Triggered by leading edge of CLK1

Checked with trailing edge of CLK2



Early Relationship



- Path starting with a leading edge of CLK1 and ending with a trailing edge of CLK2
- Time zero launch to capture: leading 0.0, trailing 3.0
- Most restrictive launch to capture: leading 4.0, trailing 3.0 (Negative for early)
- Phase shift is ((3.0 - 4.0) - (3.0 - 0.0)) = -4.00

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

cadence

In this multi-clock scenario example, we have the 4ns and 3ns clocks for the launch and capture. In this case, after unrolling the clocks, the most restrictive non-zero delta is shown at 4ns launch and 3ns capture. And for the first clock cycle, for a launch at 0, the capture is also at 3. Thus, the phase shift is calculated as -4ns. And this value of -4ns is used as a phase shift in the timing reports.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

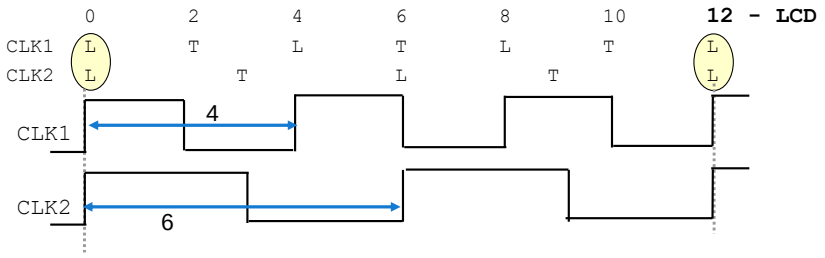
Activity: Phase Shift

The parameters for this activity include the following:

- The clock period of clock1 is 4ns.
- The clock period of clock2 is 6ns.
- The duty cycle of both clocks is 50%.

What is the minimum phase shift (late) required in a clock-to-clock transfer for data transfers from rising *clock1* to rising *clock2*?

Hint: Determine the base period and unroll clocks to determine the minimum shift required.



 © Cadence Design Systems, Inc. All rights reserved.

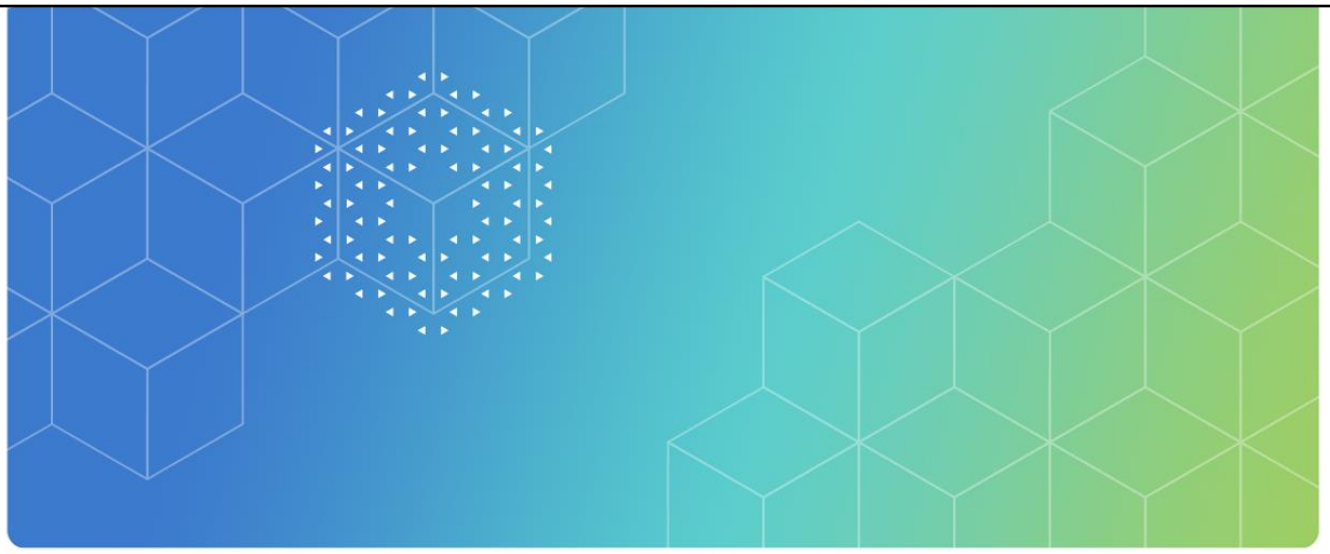
Try this activity to improve your understanding of the topics discussed.

Solution

The minimum phase shift for rising clock1 to rising clock2 transfer is 2ns.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail



Module 7

Timing Paths

In this module, let's look at timing paths.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objectives

In this module, you

- Identify the types of timing paths
- Calculate slack for every path type
- Determine the worst timing path in a small circuit

 © Cadence Design Systems, Inc. All rights reserved.

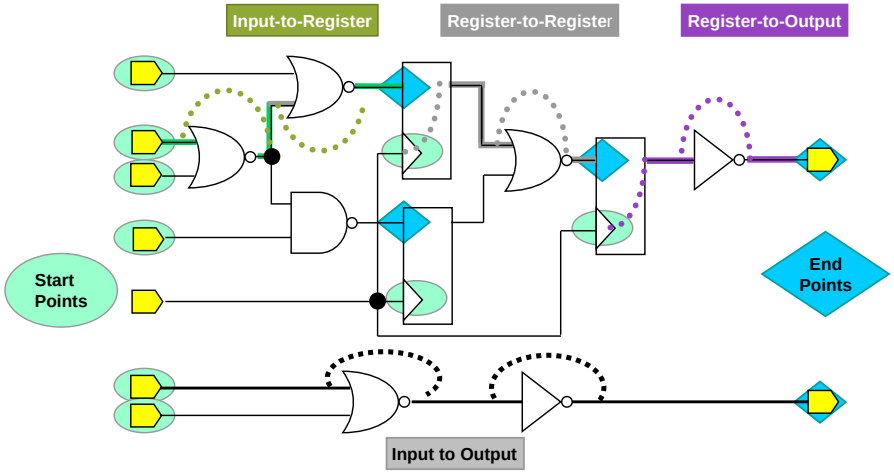
In this module, let's identify the types of timing paths and calculate the slack for every timing path. As an example, we will take a small circuit to determine the worst timing path.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Are Timing Paths?

A timing path is a combination of all the timing arcs from a start point to an endpoint. You can break down timing paths into four simple types or categories.



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

A timing path is a combination of all the timing arcs from the start point to the endpoint. For timing analysis, the start points to end points combinations are classified into four path types. Each of these path types has different requirements and different analysis styles. In the case of input to register, you need to have the input delay at the port, also known as the data arrival time at the port. For the register to output, you need to have the external output delay of that port, also known as output delay. For input-to-output paths, they need budgeting because there is no clock there. You can use virtual clocks for those input-to-output paths, or you can set path delay restrictions like max delay and min delay. The register-to-register paths use a clock at both the input side as well as the output side. You need those clock definitions to define the requirements for a register-to-register path. As you can see in the picture, there might be multiple paths between the same start point and end points or to the same endpoint. Each path is timed separately by the STA tool to determine the worst path. Then it is also used to verify the setup and hold requirements for any capture flops in the path.

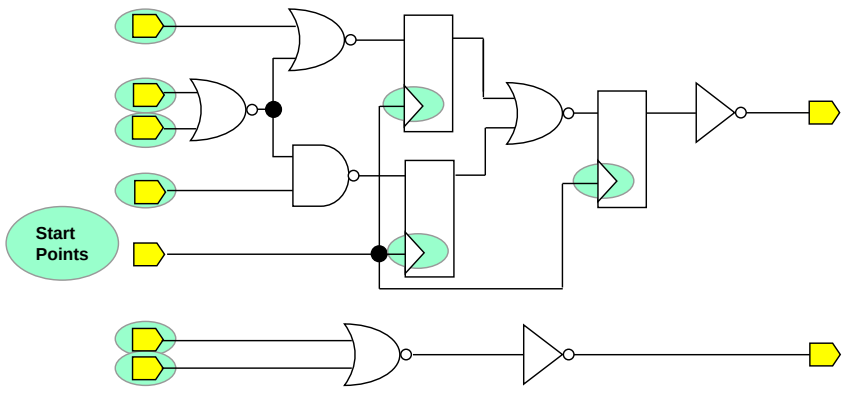
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Start Points

There are two types of start points in a timing path:

- The input port of a design (other than a clock port).
- The clock pin of a sequential cell.



 © Cadence Design Systems, Inc. All rights reserved.

You have two types of start points in a timing path. All the input ports of your design other than a clock port are considered as a start point. The clock pins of sequential cells (either flops or latches) are also considered as a start point.

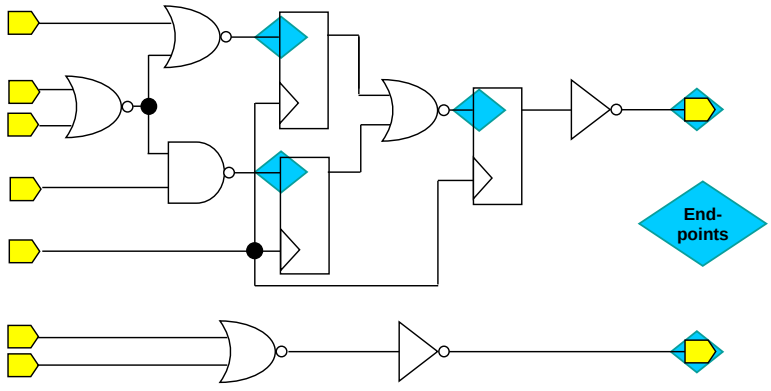
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Endpoints

There are only two types of endpoints:

- The data input pin of a sequential cell, or
- The output port of a design.



 © Cadence Design Systems, Inc. All rights reserved.

There are two types of endpoints. The data input of a sequential cell (either flops or latches) is considered an endpoint. The output port of the design is also considered an endpoint.

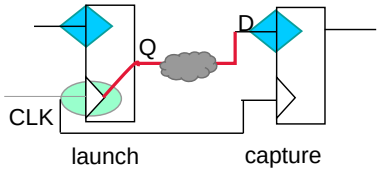
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Importance of the Clock in Timing Paths

Timing paths are relative to a clock (synchronous).

- The data arrives at a start point relative to a clock.
- The data is captured at an endpoint relative to a clock.
- The clock signal resets the time at the register, starting a new cycle.



What happens at a register?

- If the data arrives just before the clock, it is propagated through the register to the next stage.
- The new data now has a new clock cycle during which the data can propagate to the next register.
- The register resets the timing of the path at the clock edges.
- Therefore, STA tools break all timing paths at the registers so that each timing path has **one clock cycle** or **one clock period** as the timing goal.

The timing goal in STA tools is commonly known as Required time.

 © Cadence Design Systems, Inc. All rights reserved.

To simplify timing analysis and to meet the timing of all the paths, the static timing analysis tools break the timing paths at the registers or flops and make one clock period as the timing goal for each of these timing paths. The timing paths are broken down relative to a clock. Meaning they're all considered synchronous. Let's say the data arrives at the start point relative to a clock. That's the launch clock. The data is captured at an endpoint relative to another clock, which is the capture clock. The clock signal resets the time at the register and starts a new cycle. If the data meets both the setup and the hold requirements, it is propagated through the register to the next stage. The new data now has a new clock cycle during which it can propagate to the next register. Thus, at the flop, the timing of the path is reset at the clock edges.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is Slack?

Required time is defined by the timing constraints like your clock period.

Arrival time is when the signal arrives.

Slack is defined as the difference between the required time and the arrival time of a signal at the endpoint.

```
Slack = Required Time - Arrival Time (setup)
Slack = Arrival Time - Required Time (hold)
```

The synthesis engine uses the slack value to identify the paths for further optimization to meet timing or to reduce area.

The calculating equation for slack is different for the different types of paths.

Slack is included in all timing reports. A positive slack indicates that the path met your timing constraint requirement, and a negative slack indicates that the path did not meet your requirement.

A design is considered to work successfully if all the paths in the design meet timing, i.e., if none of the paths have a negative slack.

 © Cadence Design Systems, Inc. All rights reserved.

Arrival time is usually calculated from the start point or the launch clock. The required time is based on the capture clock. For setup, the difference between the required time and arrival time (Required Time minus Arrival Time) is called *slack*. For hold, the slack is usually calculated as Arrival Time minus Required Time. These slack values are included in all timing reports. A path timing is reported as met or violated based on whether the slack is positive or negative. If you're meeting timing, it's because your slack is positive. If you're not meeting timing or violating, it's because your slack is negative. You must not have any violations for your entire design to meet timing.

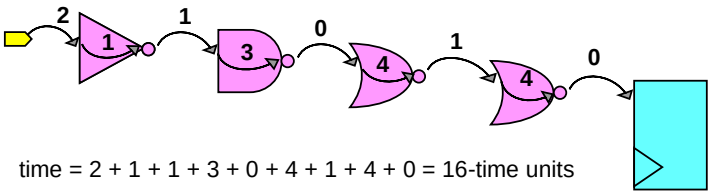
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

How Do You Time Timing Paths?

To calculate the delay of a path, simply add the delays of all the timing arcs in the path.

1. Calculate all the cell delays.
   - Cell delay timing arcs are determined by the type of cell delay model used in the technology library.
2. Calculate all the net delays.
   - Net delay timing arcs are determined by the layout estimator, or the wire-load model selected.
3. Add all the cell and net delays together to arrive at the path delay.



time = 2 + 1 + 1 + 3 + 0 + 4 + 1 + 4 + 0 = 16-time units

 © Cadence Design Systems, Inc. All rights reserved.

How are these timing paths timed? Simply add up all the delays of the timing arcs, and the path gives you the delay of the entire path. The total path delay is then verified against the timing requirement of that path to determine whether that path is passing or failing. The cell delay timing arcs are taken from the library, and the net delay arcs are from the wire load models or the physical wire delay estimation. STA tools do consider the effect of unateness on the timing arcs. The libraries have information for both rising as well as falling. The worst case of both the rising and falling is used in the timing reports. But for the following few slides, let's ignore the effect of unateness, uncertainties, etc., and assume the design is a single clock design in the ideal mode.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Register-to-Register Setup Requirement

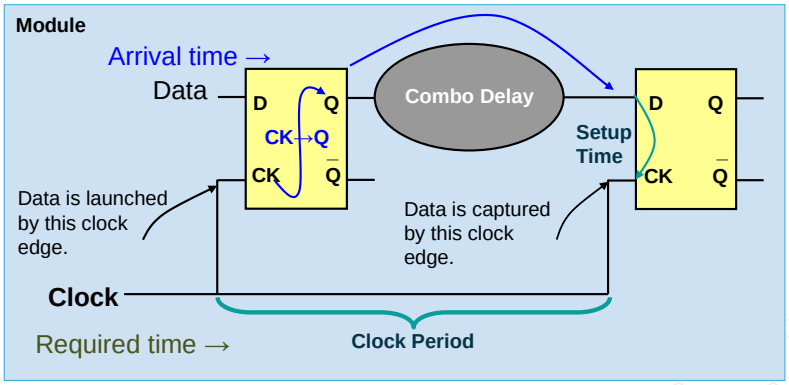
The following equation is the setup requirement for the register-to-register path:

clk_to_q + combo_delay ≤ clock_period – reg_setup

Required time = Clock Period – Setup Time

Arrival time = $t_{CK \rightarrow Q}$ + Combo delay

Slack = Required time – Arrival time



 © Cadence Design Systems, Inc. All rights reserved.

For a register-to-register path, the requirement is that your Clock to Q delay plus any combinational delay must be less than or equal to your clock period minus any setup time. If we translate that into the required time and arrival time slack calculations, the required time is the clock period minus setup time. Arrival time is your Clock to Q delay plus any combinational delay. For register-to-register paths, Arrival times are calculated from the start point or the launch clock, and required times are typically calculated from your capture clock. Some tools might move the setup into the arrival time calculation. But the idea is the same. You're just looking at the arrival times and the required times as two different sets of data and then arriving at the slack calculation based on these same equations, which is the required time minus arrival time for setup and vice versa for hold. As expected, with ideal clocks, there is no clock delay or uncertainties recorded in these equations. But for propagated clocks, clock delays, latencies, and uncertainties become part of either the arrival time or the required time, depending on where they fit, whether it's the launch clock or the capture clock.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Register-to-Register Hold Requirement

The following equation is the hold requirement for the register-to-register path:

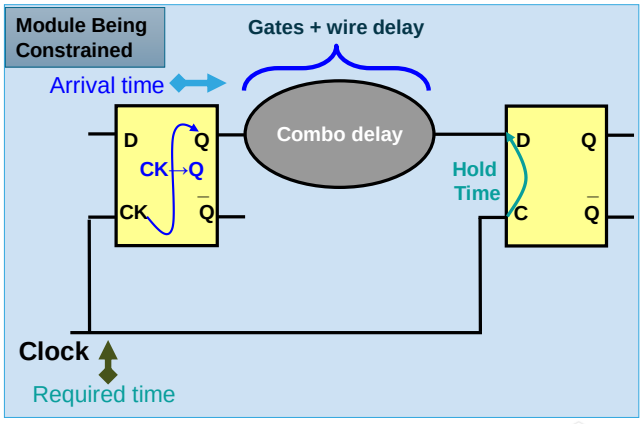
clk_to_q + combo_delay ≥ hold_check(0) + reg_hold

Required time = Hold Check (0) + Hold Time

Note: Default hold check is at 0.

Arrival time = $t_{CK \rightarrow Q}$ + Combo delay

Slack = Arrival time – Required time



 © Cadence Design Systems, Inc. All rights reserved.

The default hold check is done at the launch clock edge (or, in this case, at time 0). Therefore, your Required time is where the hold check is done, plus your hold time.

The Arrival time is Clock to Q delay plus any combinational delay.

Slack is the Arrival time minus the Required time.

Your design should be able to meet the hold requirement if your hold times are small enough and if there are no huge skews between the different flops.

© Cadence Design Systems, Inc. All rights reserved.

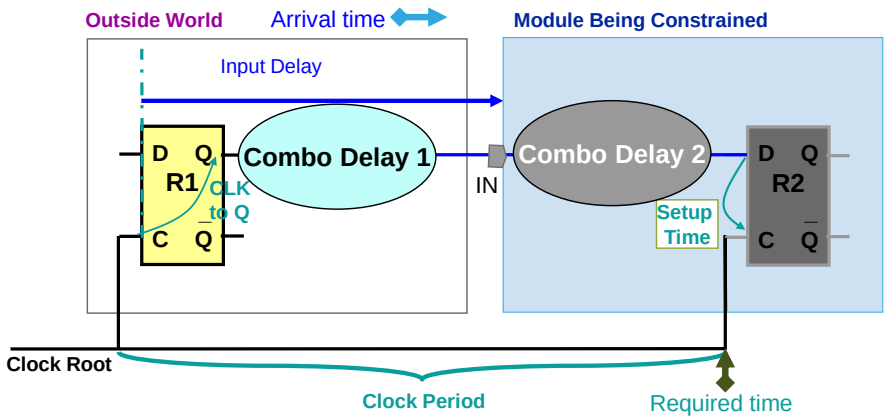
This document is for the sole use of Le Khanh Nam of Gmail

Input-to-Register Setup Requirement

Required time = (Clock Period) – (Setup time of R2)

Arrival time = ($t_{CK \rightarrow Q}$ of R1 + Combo Delay 1) + Combo Delay 2

Slack = Required time – Arrival time



 © Cadence Design Systems, Inc. All rights reserved.

To determine an input-to-register type of setup requirement, we need to know when the input started relative to our clock. Unfortunately, there is no visible launch-clock like what we saw in the register-to-register path. In a real design scenario, for chip-level design, these input port delays must be provided relative to a clock. For a block or module inside a larger design, chip-level constraints are budgeted or extracted to create the block-level constraints. Therefore, in this case, we can pretend there is a launch clock that is outside of your module or your design that is the same as your design clock. This pretends world has an R1 flop and a Combo 1 delay. The Clock to Q of the R1 flop plus the Combo Delay 1 is lumped into what is known as an input delay.

Therefore, the Arrival time is noted simply as the input delay of the port plus Combo Delay 2.

The Required time is calculated as the clock period minus the setup time for your R2 flop.

All setup slack calculation equations are the same. Slack equals required time minus arrival time.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

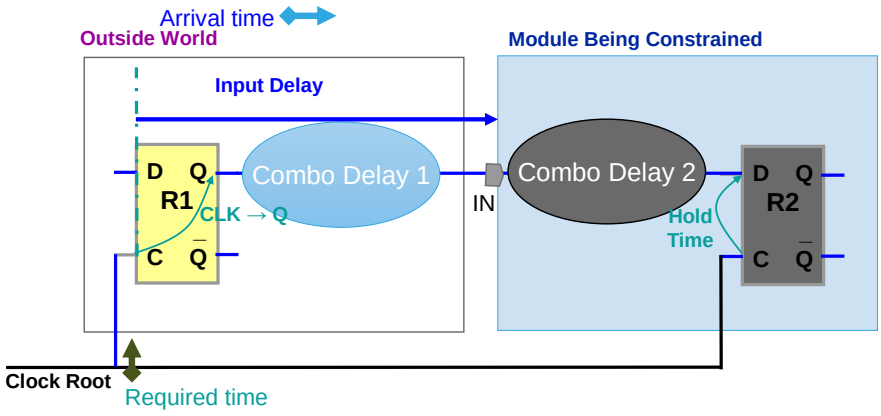
Input-to-Register Hold Requirement

Required time = Hold Check (0) + Hold time of R2

Arrival time = Input delay + Combo Delay 2

Where Input Delay = tclk-q of R1 + Combo Delay 1

Slack = Arrival time – Required time



 © Cadence Design Systems, Inc. All rights reserved.

By now, you might have noticed that because of the addition of an external flop, this case is just like the register-to-register path. You now have multiple combinational delays instead of just one. For hold, the same required time applies, which is the hold check of 0, and the hold time of the flop that is being verified or checked or is capturing the data. Arrival time is the input delay plus Combo Delay 2. All hold slack calculation equations are the same. Slack equals arrival time minus required time.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Register-to-Output Path Timing Requirement

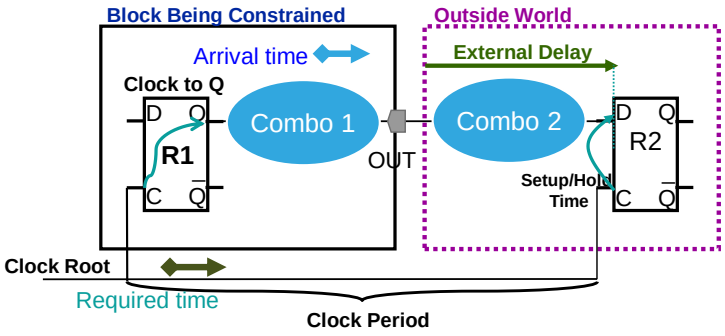
For a register-to-output path:

Setup slack = clock_period – (clk_to_q + combo_1_delay + output_delay1)

Where: `output_delay1 = combo_2_delay + setupR2`

Hold Slack = Hold check (0) – (clk_to_q + Combo1 Delay + output_delay2)

Where: `output_delay2 = combo_2_delay - holdR2`



 © Cadence Design Systems, Inc. All rights reserved.

For the purposes of register-to-output paths, we again make up an outside world and determine an external output delay. Pretend that there is another flop that is being driven outside using the same clock and come up with the external output delay. Then this case also becomes like the register-to-register scenario. Thus far, you might have noticed that in all these path types, arrival time calculations are through the data path, and all required time calculations are through the clock path. Notice that the blue is the arrival time calculation, and the green shows the required time calculation.

For setup, the output delay can be registered as Combo 2 delay plus the setup time of the R2 flop. Required time can simply be one clock period. Arrival time is Clock-to-Q plus Combo 1 Delay plus the Output delay.

For hold, the output delay can be registered as Combo 2 delay minus the hold time of the R2 flop. If you notice, we subtracted the hold time from the output delay rather than adding it to the Required time. But it still achieves the same results as a register-to-register path scenario if you make the required time for the hold to be simply the hold check at the launch clock time of 0. The arrival time for hold is the Clock-to-Q plus Combo Delay 1 plus output delay.

All external delays must therefore be defined in your constraints. We'll look at constraints later.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Input-to-Output Path Timing Requirement

Combinational paths have no clock defined. Therefore, you need to use either a virtual clock or set path delays to constrain input-to-output paths.

- If you define a virtual clock or a dummy clock, then slack is calculated as:
  Slack = Clock Period – Input Delay – Output Delay – Combinational Delay
  - Remember that input delay and output delay are set with respect to a clock.

- Then, the slacks for the input-to-output paths are calculated as:
  Setup (late) slack = path_delay_late – combo_delay
  Hold (early) slack = path_delay_early – combo_delay

 © Cadence Design Systems, Inc. All rights reserved.

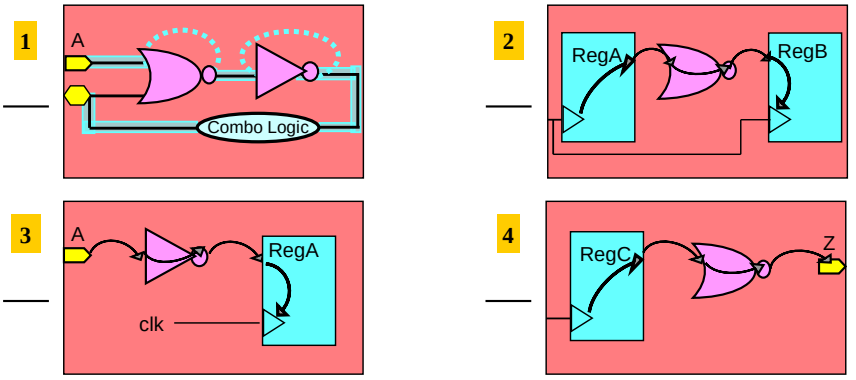
For input-to-output paths, you can set an input delay or output delay with respect to a clock, but because these paths don't have any clock defined, you need to use a virtual clock. This virtual clock or dummy clock can then define the slack for your combinational paths. You subtract your input delay, output delay, and combinational delay within that design from your clock period to come up with the slack. You can also constrain such input to output paths using a path delay constraint.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Timing Paths Types

Match the path type in each of the following illustrations to the list below.



a) Combinational **b)** Register-to-Output **c)** Register-to-Register **d)** Input-to-Register

 © Cadence Design Systems, Inc. All rights reserved.

Try the following activity to improve your understanding of the topics discussed.

Solution

1. a
2. c
3. d
4. b

© Cadence Design Systems, Inc. All rights reserved.

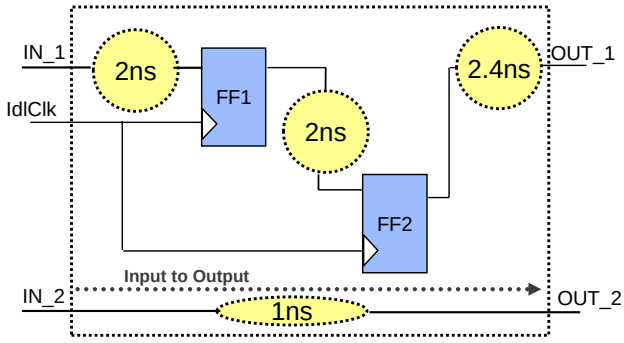
This document is for the sole use of Le Khanh Nam of Gmail

Activity: Timing Paths

The parameters for this activity include the following:

- The clock-to-Q delay of each flop is 0.5ns.
- The setup of each flop is 0.5ns.
- All input and output delays are 0.5ns.
- The clock period of Clk is 4ns.

What is the worst timing slack based on this illustration?



 © Cadence Design Systems, Inc. All rights reserved.

Try the following activity to improve your understanding of the topics discussed.

Solution for Calculating Slack

For the FF2/CLK→OUT1 path, the Slack = 0.6 ns. This path has the worst slack.

Explanation

The slack of the paths is calculated as shown here:

- Slack = [required time] – [arrival time]

The required times of all the paths are calculated as shown here:

- IN1→FF1/D required time = 4 [clock period] – 0.5 [setup of Q(FF1)] = 3.5ns
- FF1/CLK→FF2/D required time = 4 [clock period] – 0.5 [setup of Q(FF1)] = 3.5ns
- FF2/CLK→OUT1 required time = 4 [clock period] = 4ns
- IN2→OUT2 required time = 4 [clock period] = 4ns

The arrival times of all the paths are calculated as shown here:

- IN1→FF1/D arrival time = 0.5 [input delay] + 2 [comb delay] = 2.5ns
- FF1/CLK→FF2/D arrival time = 0.5 [clk→Q(FF1)] + 2 [comb delay] = 2.5ns
- FF2/CLK→OUT1 arrival time = 0.5 [clk→Q(FF2)] + 2.4 [comb delay] + 0.5 [output delay] = 3.4ns
- IN2→OUT2 arrival time = 1 [comb delay] + 0.5 [input delay] + 0.5 [output delay] = 2ns

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module 8

Introduction to SDC Constraints

Let's look at SDC constraints.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objective

In this module, you

- Identify constraints on each type of design object

 © Cadence Design Systems, Inc. All rights reserved.

In this module, we'll identify what design objects are, and the constraints on the different types of design objects. Then, we'll look at design level constraints, environmental constraints, and all the different types of constraints that are in an SDC format.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Are Constraints?

Constraints provide specifications that the design must meet through optimization.

Typical examples of constraints are:

- Clock constraints
- External constraints
- Power constraints
- Net Delay constraints
- Environmental constraints
- Design rules for manufacturing

 © Cadence Design Systems, Inc. All rights reserved.

Constraints provide all the specifications that a design must meet through optimization. You can have constraints for clocks, external drivers and loads, power, nets, the design environment, and constraints for design rules for manufacturing. You can use these constraints to optimize and, later, verify the design. For this course, we will stick to the timing-related constraints.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is Standard Design Constraints (SDC) Format?

The Standard Design Constraints (SDC) format is the standard for writing constraints in the industry.

STA tools also have their own style of writing constraints, which conform to Tcl syntax.

SDC Format Equivalence

As you look at examples of constraints, you might notice slight differences in format.

The following list shows the three equivalent ways to create SDC constraints:

- Manually written constraint
  create_clock -period 100 -waveform {0 50} clk
- Generated constraints from other tools:
  create_clock -period 100 -waveform {0 50} [get_ports {clk}]
  create_clock -period 100.000000 -waveform {0.000000 50.000000} [get_ports {clk}]

All three of the above formats are functionally equivalent. Some tools might prefer one way to another.

 © Cadence Design Systems, Inc. All rights reserved.

The SDC constraints are the standard format for writing constraints in the design industry. STA tools have their own style of writing constraints. Each tool dumps out its own type of Tcl syntax constraints. But at the same time, all of these writing variations or styles that you see are considered equivalent for the purposes of writing your constraints. When you write out your constraints, you might write it out in a simple format. But something that is generated from a tool might use different floating-point numbers or specify get_ports to identify ports. All these different variations are functionally equivalent. It's just that some tools prefer one way or the other.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Common SDC Constraints

Timing

- create_clock
- create_generated_clock
- set_clock_latency
- set_clock_transition
- set_disable_timing
- set_propagated_clock
- set_clock_uncertainty
- set_input_delay
- set_output_delay

Exceptions

- set_false_path
- set_max_delay
- set_multicycle_path

Power

- set_max_dynamic_power
- set_max_leakage_power

Operating conditions

- set_operating_conditions

Wire-load models

- set_wire_load_mode
- set_wire_load_model
- set_wire_load_selection_group

Environmental

- set_drive
- set_driving_cell
- set_load
- set_fanout_load
- set_input_transition
- set_port_fanout_number

Design rules

- set_max_capacitance
- set_max_fanout
- set_max_transition

 © Cadence Design Systems, Inc. All rights reserved.

If we were to group the SDC constraints, you would group them into operating conditions or constraints that were environmentally related, your design rules or power related and timing related, and finally, wireload models-related constraints. So, all of these different constraints are specified to achieve the common goal, which is to meet the timing for your design.

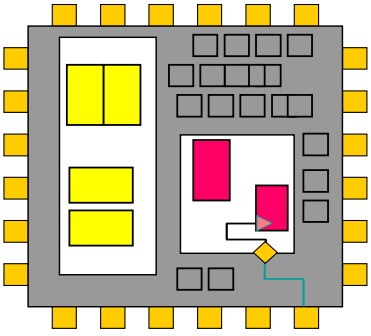
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Applying Constraints to Design Objects

You apply certain constraints to design objects to affect different parts of the design.

The table shows several design objects and the commands to get a list of these objects.



Object	Command	Description
Design	current_design	Design is a container for cells or is the entire circuit.
Cell or Block	get_cells	Cell is an instance of a design or is a library component.
Port	get_ports all_inputs all_outputs	A port is a signal entry point or exit point to a design.
Pin	get_pins	A pin is a hierarchical port of a design, port of an instance, or a port of a library cell.
Clock	get_clocks all_clocks	A clock is a port or a pin that drives sequential cells.
Net	get_nets	A net is an interconnect between cell pins and design ports.

 © Cadence Design Systems, Inc. All rights reserved.

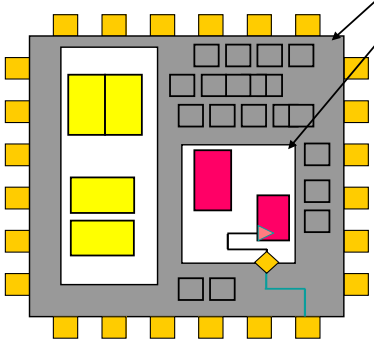
In order to set the constraints, let's categorize the design into different design objects. You might have your design, which is the container for the entire circuit; you might have cells or blocks or have ports, which is an entry or exit point into the design, and your pins, which are necessarily the ones that are around your blocks for an instance or a hierarchical port or design or a sub design. Clocks are the sequential cell drivers. Nets are the wires or the interconnect between the cells and the design ports.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Design Object: Chip or Design

Operating conditions affect the chip or a block.



Object	Command	Description
Design	current_design	Design is a container for cells or is the entire circuit.
Cell or Block	get_cells	Cell is an instance of a design or is a library component.
Port	get_ports all_inputs all_outputs	A port is a signal entry point or exit point to a design.
Pin	get_pins	A pin is a hierarchical port of a design, port of an instance, or a port of a library cell.
Clock	get_clocks all_clocks	A clock is a port or a pin that drives sequential cells.
Net	get_nets	A net is an interconnect between cell pins and design ports.

 © Cadence Design Systems, Inc. All rights reserved.

cadence

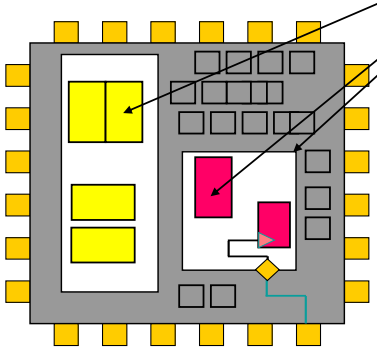
Your operating conditions are the ones that affect the entire chip, or if you're constraining a block in this case, then those that affect the entire block.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Design Object: Cell or Block

Apply power constraints and wireload models to the cell or block object.



Object	Command	Description
Design	current_design	Design is a container for cells or is the entire circuit.
Cell or Block	get_cells	Cell is an instance of a design or is a library component.
Port	get_ports all_inputs all_outputs	A port is a signal entry point or exit point to a design.
Pin	get_pins	A pin is a hierarchical port of a design, port of an instance, or a port of a library cell.
Clock	get_clocks all_clocks	A clock is a port or a pin that drives sequential cells.
Net	get_nets	A net is an interconnect between cell pins and design ports.

 © Cadence Design Systems, Inc. All rights reserved.

cadence

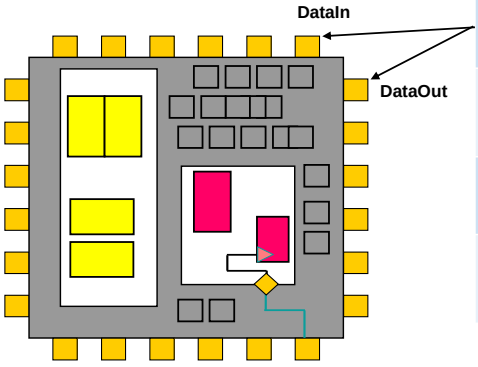
You would apply constraints like power constrains or wireload models to the cell or the block object.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Design Object: Port

Apply timing constraints and environmental constraints to a port object.



Object	Command	Description
Design	current_design	Design is a container for cells or is the entire circuit.
Cell or Block	get_cells	Cell is an instance of a design or is a library component.
Port	get_ports all_inputs all_outputs	A port is a signal entry point or exit point to a design.
Pin	get_pins	A pin is a hierarchical port of a design, port of an instance, or a port of a library cell.
Clock	get_clocks all_clocks	A clock is a port or a pin that drives sequential cells.
Net	get_nets	A net is an interconnect between cell pins and design ports.

 © Cadence Design Systems, Inc. All rights reserved.

To tell the tools what the surrounding environment looks like, you might apply environmental constraints to a port object, including set_drive or constraints like that. To find and use a port object, you can use a get_ports command.

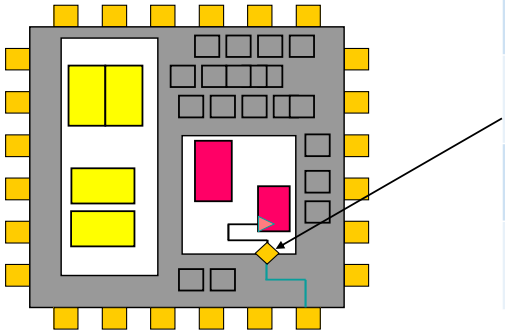
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Design Object: Pin

Apply timing constraints and environmental constraints to a port object.

Pins are like ports of subdesigns.



Object	Command	Description
Design	current_design	Design is a container for cells or is the entire circuit.
Cell or Block	get_cells	Cell is an instance of a design or is a library component.
Port	get_ports all_inputs all_outputs	A port is a signal entry point or exit point to a design.
Pin	get_pins	A pin is a hierarchical port of a design, port of an instance, or a port of a library cell.
Clock	get_clocks all_clocks	A clock is a port or a pin that drives sequential cells.
Net	get_nets	A net is an interconnect between cell pins and design ports.

 © Cadence Design Systems, Inc. All rights reserved.

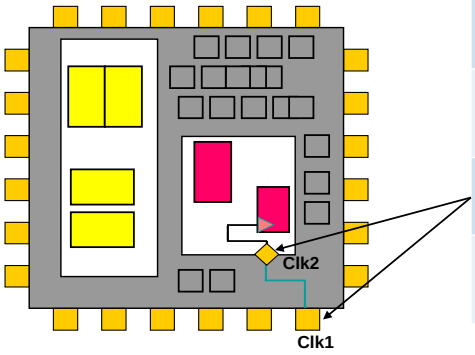
Pins are like ports of your subdesign or inputs and outputs to a cell. So, any constraints that need to be applied at the hierarchical port level of your design would be applied at the pin level.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Design Object: Clock

Apply clock-related timing constraints to the clock object.



Object	Command	Description
Design	current_design	Design is a container for cells or is the entire circuit.
Cell or Block	get_cells	Cell is an instance of a design or is a library component.
Port	get_ports all_inputs all_outputs	A port is a signal entry point or exit point to a design.
Pin	get_pins	A pin is a hierarchical port of a design, port of an instance, or a port of a library cell.
Clock	get_clocks all_clocks	A clock is a port or a pin that drives sequential cells.
Net	get_nets	A net is an interconnect between cell pins and design ports.

 © Cadence Design Systems, Inc. All rights reserved.

If a port is defined as a clock, then that becomes a clock object. So, you would define period, waveform or anything related to a clock on a clock object.

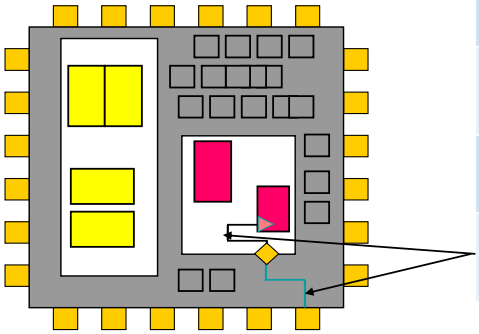
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Design Object: Net

Apply timing constraints to the net objects.

Wireload models affect the delays of these net objects.



Object	Command	Description
Design	current_design	Design is a container for cells or is the entire circuit.
Cell or Block	get_cells	Cell is an instance of a design or is a library component.
Port	get_ports all_inputs all_outputs	A port is a signal entry point or exit point to a design.
Pin	get_pins	A pin is a hierarchical port of a design, port of an instance, or a port of a library cell.
Clock	get_clocks all_clocks	A clock is a port or a pin that drives sequential cells.
Net	get_nets	A net is an interconnect between cell pins and design ports.

 © Cadence Design Systems, Inc. All rights reserved.

cadence

You can apply timing constraints to net objects. Wireload models are typically for the purposes of calculating the delays of these net objects, so you can get the net object and identify what the delay on that particular net is.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Design Rule Constraints

Design rule checks (DRC) are checks that must be met before the foundry can manufacture the design. The constraints for DRC checks are simply known as design rule constraints.

The chosen manufacturer's technology or process determines the checks.

DRC checks such as max_capacitance and max_transition are part of the library supplied by the foundry.

Design rule constraints can be applied at the design level.

- You cannot relax these constraints because they are supplied by the foundry to meet certain specifications.
- A more restrictive setting can be applied to specific pins if you choose.

The following commands set these design rule constraints in SDC:

- set_max_capacitance <design | output pins>
- set_max_fanout <design | output pins>
- set_max_transition <design | input pins>

 © Cadence Design Systems, Inc. All rights reserved.

cadence

The design rule constraints are applied at the design level. These design rule constraints must be met so that your design can be manufactured. These rules come from the fabrication process, and you must set these design rule constraints according to the fabrication process.

© Cadence Design Systems, Inc. All rights reserved.

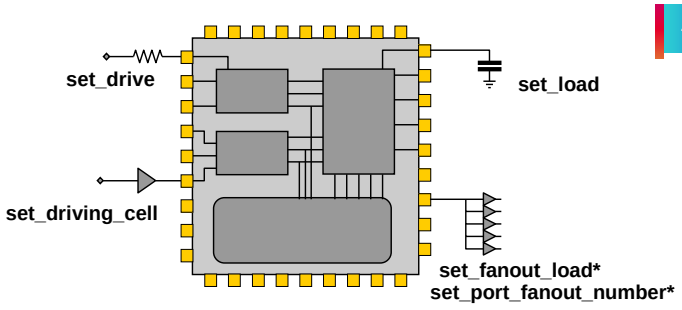
This document is for the sole use of Le Khanh Nam of Gmail

Setting Environmental Constraints

The environment of the chip has an influence on chip performance. Environmental constraints are applied at the port level.

You need to model the environment in terms of constraints:

1. Set the driving cell that drives the input of the chip
2. Set the load that must be driven by the pins
3. Set input delays
4. Set output delays



 © Cadence Design Systems, Inc. All rights reserved.

The environmental constraints are typically applied at the port level. The environment of the chip obviously has an influence on your chip performance, and you need to model the environment in terms of the constraints. A driving cell is something that drives the input of the chip, and the load is what must be driven by the output pins. Both must be accounted for when you're designing your chip. For example, how much load can an output drive? Or what is the driving cell that is driving your input pins? Both must be specified for your design to work properly.

© Cadence Design Systems, Inc. All rights reserved.

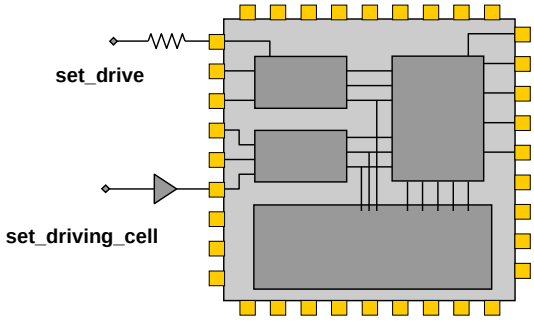
This document is for the sole use of Le Khanh Nam of Gmail

Step 1: Setting the Driving Cell

Use the *set_driving_cell* command to define the cell that drives the input port.

```
set_driving_cell -lib_cell BUFX8_TAX0  -library xlite_core [get_ports
   {ipmitxbfr_rdata[3]}]
```

The input drive of a pin is the strength of incoming signal, or its capacity to drive the input and its fanout.



 © Cadence Design Systems, Inc. All rights reserved.

If you know what specific cell is driving the input of a pin, then you can set that using the set-driving-cell command. By defining the driving cell, you're defining what the strength of the incoming signal is, or its capacity to drive the input and the fanout of that input pin. So, you can use the set-driving-cell command to tell the STA tool to use this SDC constraint to determine the driving cell of that input pin.

© Cadence Design Systems, Inc. All rights reserved.

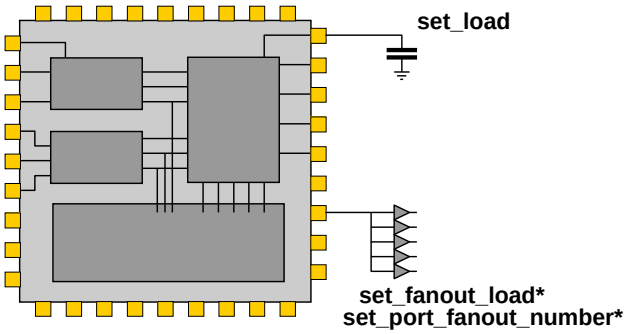
This document is for the sole use of Le Khanh Nam of Gmail

Step 2: Setting Output Load

The *set_load* command lets you define the net capacitance load that the output pin "sees."

```
set_load -pin_load 0.2 [get_ports {decalf_hdrmem_ld}]
```

The load of the output port is the amount of capacitance that the port has to drive. It is the total capacitance as seen by the output port. If multiple pins are driven by the output port, then it is a summation of all the capacitances of individual endpoints.



 © Cadence Design Systems, Inc. All rights reserved.

You can specify the same information as shown above using the following commands together:

The set_port_fanout_number command lets you specify the fanout of the output port. This command is used in conjunction with the set_fanout_load command.

- set_port_fanout_number 5 [all_outputs]

The set_fanout_load command lets you specify the value of the input capacitance of a cell driven by the output port.

- set_fanout_load 100x100 -library slow [get_ports CELLX1/A*]

The set_port_fanout_number and set_fanout_load are also used as design rule constraints for the ports.

You can define what the load of an output port is, which is the total amount of capacitance that the port has to drive using the set-load command. Or you can also specify the set-fanout-load and set-port-fanout-number, a combination of these two to determine the load per fanout and how much fanout it has. Both of these in combination can determine your total load that this particular output pin sees. Look at the notes to see examples of port fanout number and fanout load values and examples.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

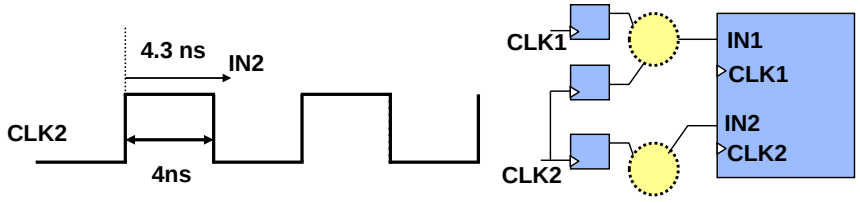
Step 3: Setting Input Delay

Use the set_input_delay command to set the arrival time on a port relative to a clock.

```
set_input_delay delay [-clock clock][-add_delay] port_pin_list
```

```
set_input_delay 4.3 -rise -clock CLK2 {IN2}
```

The command in the example implies that the input signal arrives at the *IN2* port, 4.3ns after the active edge of the *CLK2* clock.



 © Cadence Design Systems, Inc. All rights reserved.

You can define the environment of a chip using the set-input-delay and set-output-delay commands. Using the set-input-delay command, set the arrival time on a port relative to a clock. The set-input-delay command specifies a delay value and whether it’s rises or fall, and what clock it's relative to. In this example, the input signal arrives at the IN2 with respect to the CLK2 clock, and it arrives 4.3ns after the active edge of CLK2. For bidirectional ports, you can specify the path delays for both input and output modes.

© Cadence Design Systems, Inc. All rights reserved.

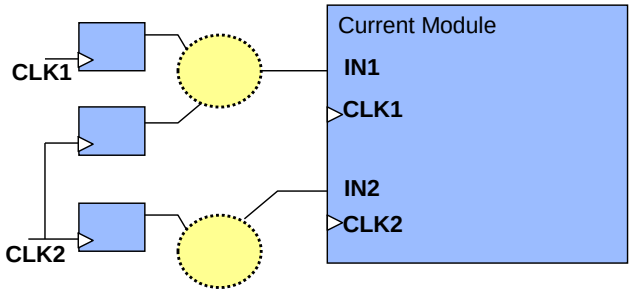
This document is for the sole use of Le Khanh Nam of Gmail

Setting the Input Delay on Ports with Multi-Clock Relationships

Use *-add_delay* to capture multiclock delay relationships.

```
set_input_delay 2.7 -clock CLK1 -add_delay { IN1 }
set_input_delay 4.2 -clock CLK2 -add_delay { IN1 }
```

The command in this multiclock example specifies an input delay for IN1 of 2.7ns relative to the CLK1 clock and 4.2ns relative to CLK2.



 © Cadence Design Systems, Inc. All rights reserved.

For multiple clocks scenarios where a particular input has to go to multiple clocks, then you would provide that relationship using the -add-delay option. If you want to specify that the first example command shows the input delay in relation to clock1, and the next one is the input delay in relation to clock2. When you specify the -add-delay option for each of the clocks, then that will give the multi-clock scenario that you need when that same input is going to different clocks in the design.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

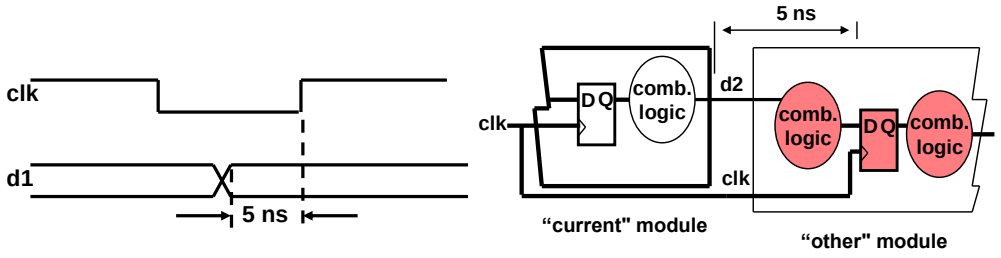
Step 4: Setting Output Delay

Use the *set_output_delay* command to set the output path delay values for the design.

```
set_output_delay delay [-clock clock][-add_delay] port_pin_list
```

```
set_output_delay 5 -rise -clock clk { d1 }
```

The command in the example sets an expected output delay of **5ns** relative to clock *clk*.



 © Cadence Design Systems, Inc. All rights reserved.

Set the output delay using the set-output-delay command for all the output ports in your design. This will ensure that all the registered output paths will meet the timing specified. The tool uses this value to constrain the output ports. The set-input-delay and set-output-delay are used at a block level in place of the set-driving-cell and set-load commands when you're unsure or budgeting the constraints based on the modules in the design.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Operating Conditions

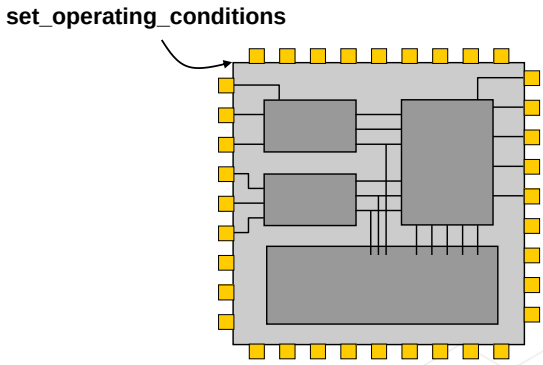
```
set_operating_conditions
```

To switch between the operating conditions, use this command.

The timing analysis tool resets itself when you switch between operating conditions and creates different timing data based on the new library information.

Operating conditions are applied at the design level. Nowadays, the technology library is created for a specific set of operating conditions. Thus, specifying the library is probably sufficient.

```
set_operating_conditions -name "WCCOM" -
  library "slowtech"
```



 © Cadence Design Systems, Inc. All rights reserved.

Operating conditions are something that you would apply at the design level. We have specific library files created for a specific set of operating conditions. So, you can specify just the library, and that in itself is sufficient. But if you are reading in multiple libraries and you want to use a specific set of operating conditions, then you should use the set-operating-conditions command to set that operating condition to use as your default for timing analysis.

As the design process varies, you might have to change this operating condition often to get different timing data. A corner is one specific set of processes, voltage and temperature. For multi-mode multi-corner designs, you'll have a particular design being verified for different modes and different corners. If you are verifying your timing across multiple corners, then you will have to specify your operating conditions according to the corner that you're currently verifying.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Wireload Mode: Top

To set the wireload mode to the *top*, use:

```
set_wire_load_mode top
```

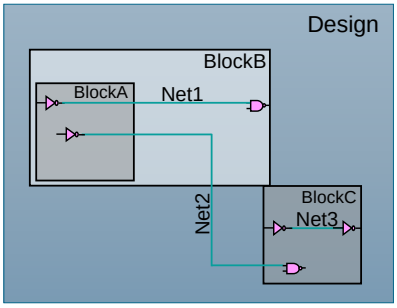
The wireload mode is applied at the design level.

- You can choose *top*, *enclosed*, or *segmented* modes.

In *top* mode, all the net delays are based on the entire design area.

If the default wireload model in the library is based on a zero WLM, all the net delays will be 0. This approach is used in synthesis to check constraint sanity.

Optional in newer tools



In **top** mode, all the nets in the design use the *Large_wlm* model.

 © Cadence Design Systems, Inc. All rights reserved.

Wireload modes determine how the net delay calculation happens. In the case of the top mode, all the net delays in the entire design use the top model. In the case where you have a wireload model that is considered as large, and if it encompasses the entire area of the design, then the STA tool uses Large to calculate the net delays throughout the design. Obviously, this is very pessimistic. Use this only for conservative estimates. The opposite end is using a zero-wireload model. If you just want to see if the design will meet timing even without any wire delays, then you could use the zero wireload model option. To force a particular subdesign to use a different wireload model than the one from the top, you can specify the set-wire-load-model command for that particular subdesign.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Wireload Mode: Enclosed

To set the wire-load mode to *enclosed*, use:

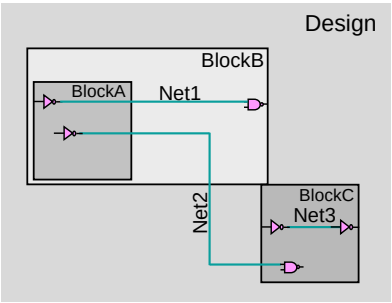
```
set_wire_load_mode enclosed
```

In *enclosed* mode, net delays are based on different block sizes rather than the entire design area.

In this illustration, the wire-load models for the blocks are chosen automatically from the library as follows:

- Nets enclosed in Design, use *Large_wlm*.
- Nets enclosed in BlockB, use *Avg_wlm*.
- Nets enclosed in BlockA and BlockC, use *Small_wlm*.

Optional in newer tools



In **enclosed** mode:
Net1 uses *Avg_wlm*
Net2 uses *Large_wlm*
Net3 uses *Small_wlm*

 © Cadence Design Systems, Inc. All rights reserved.

Another wireload mode is the enclosed mode. Let's say your entire design is 8000 units, and the average block is 4000 units. Block B would be encompassed under 4000 units, whereas the entire design would take 8000 units. Then for the enclosed mode, Block B would choose the 4000 units because it can be enclosed in the 4000-unit model. But for the rest of the design nets, the tool will use the larger one, which is the 8000-unit wireload model. Now let’s extend the same logic to another example. If there are Blocks A and C, they will fall under the category of small WLM of, let's say, 1000 units. Then you would have all the nets under Block A and Block B use the small WLM, which falls under the 1000 units. So, from that example scenario, if we look at multiple nets in this design, Net1 uses whatever encompasses that particular Net1, which is Block B, and uses average WLM. For Net2, the entire design encompasses it. Neither Block B nor Block C can contain that particular Net2. So Net2 must use large WLM, and Net3, because it is enclosed in Block C, must use the small WLM.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Wireload Mode: Segmented

To set the wire-load mode to *segmented*, use:

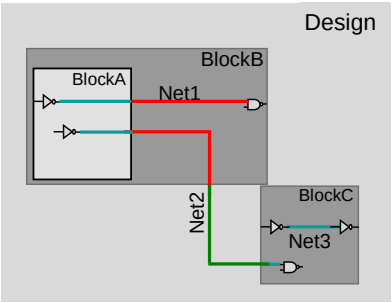
set_wire_load_mode segmented

In *segmented* mode, the tool adds the delays from each net segment to get the total net delay.

In this illustration, the wireload models for the blocks are chosen automatically from the library as follows:

- Nets segments in Design, use *Large_wlm*.
- Nets segments in BlockB, use *Avg_wlm*.
- Nets segments in BlockA and BlockC, use *Small_wlm*.

Optional in newer tools



In **segmented** mode:
Segment of Net1 in BlockA uses Small_wlm
Segment of Net1 in BlockB uses Avg_wlm
All of Net3 uses Small_wlm
BlockA/Net2 uses Small_wlm
BlockB/Net2 uses Avg_wlm
Design/Net2 uses Large_wlm
BlockC/Net2 uses Small_wlm

 © Cadence Design Systems, Inc. All rights reserved.

In the segmented wireload mode, the tool breaks down the entire design net into small chunks. Depending on what encloses that particular chunk of the net, that model is used for that particular piece of the net. If you look at the bottom right, the segmented mode shows the different segments of the net and what wireload models are used. In the case of Net3, all of Net3 is encompassed in Block C and uses a small WLM. But if you break down Net2, Block A Net2 will use small WLM, Block B Net2 will use average WLM, and the design Net2 uses the large WLM. Block C Net2 uses the small WLM. So, you're breaking down the nets into smaller pieces. This sounds like the most logical choice, but because the tool varies in how it does the optimization, the computation is intensive, and it runs very slowly. Also, if there are nets that fall under critical paths, then the wireload model often changes because of the block that encompasses that particular net. Thus, the models and the delays the tool generates for these nets become unreliable. Wireload models can be used for timing verification. But, in general, they are not as reliable for optimization, and segmented is specifically not a recommended model for net delays.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Wireload Models

To force a WLM on a block, use:

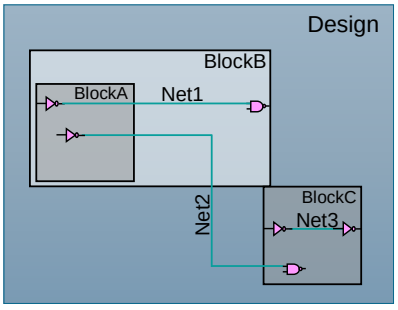
```
set_wire_load_model
```

Use these set_wire_load_model commands for the design example on the right, chosen based on the size of the block.

- `set_wire_load_model -name Large_wlm {Design}`
- `set_wire_load_model -name Avg_wlm [get_cell {BlockB}]`
- `set_wire_load_model -name Small_wlm [get_cell {BlockA BlockC}]`

Optional in newer tools

Wireload models are applied at the block or design level. The STA tool automatically chooses the default wireload model (WLM) from the library to determine the net delays.



 © Cadence Design Systems, Inc. All rights reserved.

Wireload models are applied during synthesis to determine the wire delays. These wireload models are categorized based on area. So, when there are no wireload models specified, the tool automatically selects the default from the library. If you want to specify the model for a block, you can specify the set-wire-load-model command. So, you usually have a wireload mode and wireload model, and both of these, in combination, determine what wireload model to use for a particular net. When nothing is specified, it goes based on what the defaults are from the library.

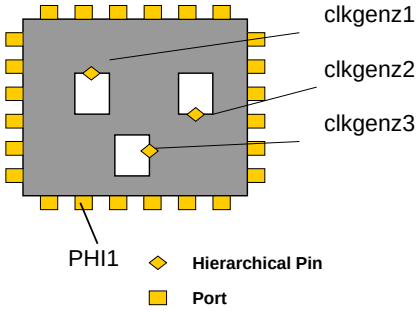
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

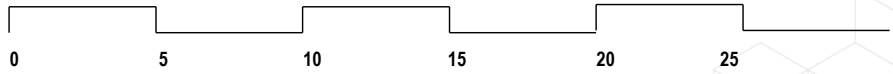
Creating a Clock

To define a clock signal and specify the ports or pins that are connected to the clock, use:

```
create_clock -name -period -waveform {rising_edge  falling_edge} \
     <port/pin>
```




```
create_clock -name "clock1" -period 10 -
  waveform {0.0 5.0} {PHI1}
create_clock -name "clock2" -period 20 -
  waveform  {0.0 10.0} {clkgenz1 clkgenz2
  clkgenz3}
```



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

The create-clock command creates a clock object with an ideal signal. It specifies the period, waveform and the ports or pins that are connected to the clock source. The duty cycle can be derived from the period and the waveform, defined by the edges.

In the first example shown above, the command defines an ideal clock, which is connected to a port called PHI1. The ideal clock is named clock1 and has a period of 10ns, a leading edge at 0, and a trailing edge at 5. So, the duty cycle is 50%.

For the second example, the period is 20; the ideal clock is defined as clock2; it is set on pins named clockgenz1, clockgenz2, and clockgenz3.

This clock has a period of 20 and a duty cycle of 50% because the waveform is defined as {0 10}, so the leading edge is at 0, and the trailing edge is at 10.

These three clocks, which could be generated clocks, could have also been defined using the create-generated-clock command, and then any modifications on the source clock are inherited by the generated clocks.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Clock Transition

Clock transition is also known as clock slew. It is set on an ideal clock using the *set_clock_transition* command.

```
set_clock_transition [-rise] [-fall] [-min] [-max] \
  transition_value clock_list
```

```
set_clock_transition 0.38 -rise [get_clocks clk1]
set_clock_transition 0.25 -fall [get_clocks clk1]
```

 © Cadence Design Systems, Inc. All rights reserved.

Just like all other signals, clocks also have rise and fall times. The fanout of the clock is much greater than those of most other signals. The output slew degrades as the clock signal propagates through these cells. STA tools keep track of these maximum transition times on all signals, including the clock. After creating the ideal clock, you need to refine it with specifics.

So, if you want to model the real nature of a clock, you need to add the transition times, uncertainty, and latency to the ideal clock. In this example, we are specifying a slew of 0.38ns on the rising edge of the ideal clock and a slew of 0.25ns on the falling edge. Notice that get-clocks command is calling the clock object and the clock object will indirectly call the related ports. As we are defining things on the clock object, all ports that are related to clk1 will all get the same slew, uncertainty, and any other values you set on the clock object.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

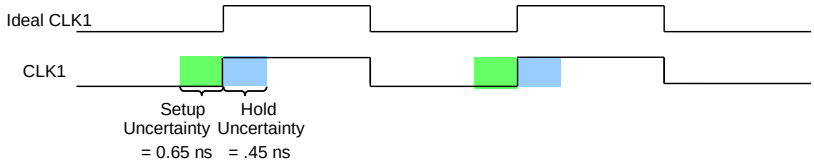
Setting Clock Uncertainty

Use the *set_clock_uncertainty* command to account for additional margin for setup and hold.

```
set_clock_uncertainty [-from from_clock] [-to to_clock] [-rise] [-fall] \
  [-setup] [-hold] uncertainty [clock_objects]
```

```
create_clock -name CLK1 -period 10 -waveform { 0 4} [get_ports CLK1]
set_clock_uncertainty -setup 0.65 [get_clocks CLK1]
set_clock_uncertainty -hold 0.45 [get_clocks CLK1]
```

The above commands specify a setup uncertainty of 0.65ns and a hold uncertainty of 0.45ns on the CLK1 clock.



 © Cadence Design Systems, Inc. All rights reserved.

You can specify clock uncertainty using the set-clock-uncertainty command, which has options for setup as well as hold. You can define separate uncertainties for setup and hold, or you can just set a clock uncertainty for both. In this example, we are setting different uncertainties for setup and hold. In the ideal mode, it would go by these values for setup and hold uncertainties, and skew is added in there. Whereas in the propagated mode, you would want to set uncertainty only to the jitter value rather than including both skew as well as the jitter.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Clock Latency

Use the *set_clock_latency* command to define clock latency.

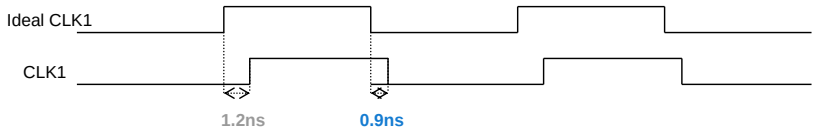
```
set_clock_latency [-source] [-clock clock_list] latency obj_list
   [-min ] [-max] [-rise ] [-fall] [-early] [-late] [-clock_gate]
```

In addition to clock slew and clock uncertainty, *clock latency* is set on the clock port to account for the delays in the clock signal from the clock definition point to a register clock pin.

Example

```
set_clock_latency 1.2 -rise [get_clocks CLK1]
```

```
set_clock_latency 0.9 -fall [get_clocks CLK1]
```



 © Cadence Design Systems, Inc. All rights reserved.

You can define both the source latency as well as the network latency using the set-clock-latency command. When you specify rise or fall, these would include the rise and fall transition times as well. Without those, you would just model the latency for when there is no transition time included.

The source latency is specified using the -source option. When you don't specify the -source option, you're modeling the network latency.

In this example graphic showing the ideal clock versus the actual clock, the visible shift in the clock waveform is caused by the latencies.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

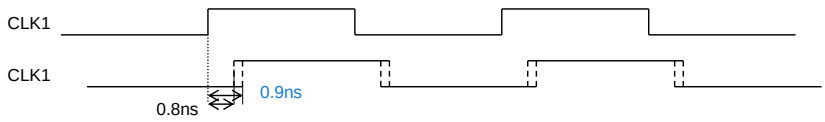
Setting Clock Latency: Hold and Setup

Use *-early,* and *-late* clock latency settings for hold and setup checks, respectively.

```
set_clock_latency 0.8 -source -early [get_clocks CLK1]
```

```
set_clock_latency 0.9 -source -late [get_clocks CLK1]
```

The example commands above define a *clock source latency* of 0.8 for hold and 0.9 for setup.



 © Cadence Design Systems, Inc. All rights reserved.

You can define different clock latencies for hold and setup using the -early and -late options, where -early relates to hold and -late relates to setup.

Therefore, when you define clock latency with the -source, and -early options, it implies that you're setting the latency for hold and that it is the source latency.

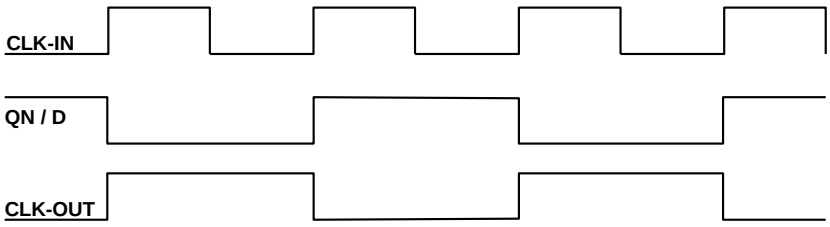
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Generated Clocks

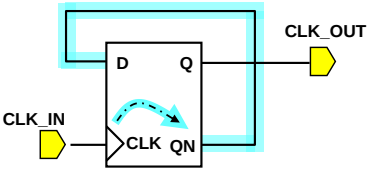
A generated clock is a clock that can inherit the properties of a parent clock.

- Use generated clocks to model clock dividers and multipliers to create a new clock from a clock source.
- Generated clocks provide for an actual relation than having to unroll the clocks to find their base clock period.



These waveforms are assuming there is no CLK → Q delay

A divide-by-two clock in a design



 © Cadence Design Systems, Inc. All rights reserved.

Generated clocks are helpful when you want to model a particular clock and inherit all the properties of the parent clock. Any uncertainties that you need to inherit, you can inherit from the master or a parent clock. The other advantage of generated clocks is that when you want a one-third, which must use floating-point numbers, the calculations are compute-intensive, and the timing reports can get messy due to the calculations and approximations used. In the case of a generated clock, implying that it's one-third of a master clock, the relational information makes it easier to unroll the clocks as and when needed.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

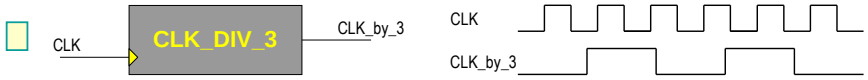
Creating Generated Clocks

To create a clock object generated within the block or design for either multiply-by or divide-by clocks, use:

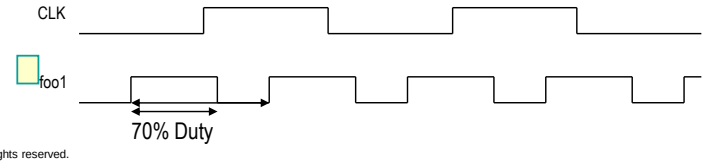
```
create_generated_clock  [-name] -source master_pin [-edges] [-divide_by factor]
      [-multiply_by factor] [-master_clock] source_objects
```

You can also generate clocks from the edges of the source clock.

create_generated_clock -divide_by 3 -source CLK [get_pins CLK_by_3]



create_generated_clock -multiply_by 2 -duty_cycle 70 -source CLK [get_pins foo1]



 © Cadence Design Systems, Inc. All rights reserved.

A generated clock is a fraction or a multiple of the source clock. As we discussed earlier, the purpose is to inherit the properties of the source clock. Because you can use it to model the outputs of multipliers and clock dividers, the create-generated-clock command is what you use to model such clocks. It helps with defining the relationship between the clocks, especially when these clocks have data talking to each other or signals talking to each other. Instead of having to unroll the clocks to define their base period, you already have an established relationship, and all the floating-point calculations become easier. In the first command, the generated period is one-third, and the one at the bottom shows that it's twice the period. You can vary the duty cycles, and you can provide different edges, and so on. This helps in calculating the phase shift easily and verifying any data transfers that happen between these clocks.

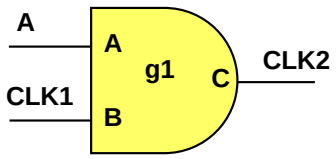
© Cadence Design Systems, Inc. All rights reserved.

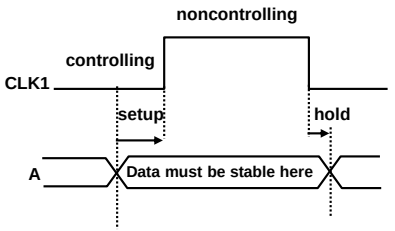
This document is for the sole use of Le Khanh Nam of Gmail

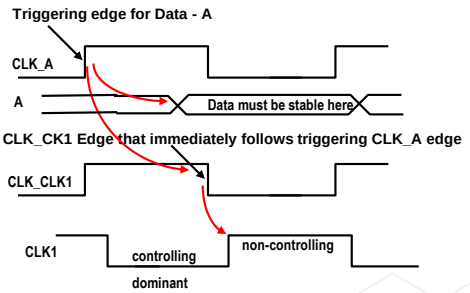
Gated Clocks

Clock gating occurs when a data signal is used to control a clock signal.

Clock clipping is a possible effect of clock gating and happens when the data signal dominates the output of the clock gate, and the clock is lost.







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

Gated clocks are quite common and can help reduce dynamic power dissipation. Especially when you have a lot of flops in your design, you want to be able to control the switching of the flops.

You can turn off the clock to those flops and save on dynamic power for your design. In the case where you have gated clocks, the data signal, which is the A signal in this example, is the enable that enables or disables the clock that goes to clock2. clock2 is the one that is driving the flops. The signal enable can be synchronous, or it can be an asynchronous signal.

Clock clipping happens when the data signal itself dominates the output of the clock gate, and your original clock is lost. To prevent clock clipping, which is a side effect of clock gating, this enable signal must be synchronous. The synchronicity can be defined as either that the clock that drives A is the same as clock1 or the root of clock1. Or it can also be an integer multiple of the clock root of clock1. In either case, clock clipping will not occur.

In the case where your input A is at logic one, then clock1 propagates to the output clock2. It should be the same as clock1; when a zero, the output is also zero. Thus, clock2 is now a gated clock signal that is controlled by the enable signal A. When you run the setup and hold checks for this gated instance, the data must be stable. Meaning the enable signal must be stable. It should pass the setup and hold checks for that gated clock. This gate must have the setup and hold checks defined in the library. Only then you can use this clock-gated element G1 in your design. You can also perform certain clock gating checks along with the setup and hold checks. One of these checks is to ensure that clock clipping does not occur. So, the clock clipping, as we've defined, is when the data signal dominates the output of the clock gate. To ensure that this doesn't happen, you use a dominant value for the clock input. Whenever you're using an AND gate or NAND gate, the dominant value is zero. If you use an OR gate, then the dominant value is one.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Clock Gating Checks

To add clock gating checks (setup or hold) on pins of any design object that gate a clock signal, use:

```
set_clock_gating_check
```

The set_clock_gating_check command ensures that the controlling data signals are stable before the clock is active.

This example sets a clock gating check on the clock CK1.

```
set_clock_gating_check -setup 0.4 -hold 0.2
  [get_clocks CK1]
```

This example sets a clock gating check on all the clock gating cells that are AND1.

```
set_clock_gating_check -setup 0.5 [get_cells
  AND1]
```

 © Cadence Design Systems, Inc. All rights reserved.

cadence

You can use the set-clock-gating-check command to set checks on the pins of any design object that gates a clock signal.

In the examples shown, a clock gating check of 0.4 for setup and 0.2 for hold is used.

You must look for all the clock gating elements and ensure that these clock gating checks pass.

Some of these clock gating checks are built into the library elements of these cells. Most timing analysis tools can extract these setup and hold checks from the libraries themselves and make sure that these checks are performed automatically. In those cases, you might not need to set any additional clock gating checks.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Asynchronous Clocks

Asynchronous clocks are clocks that are usually generated from different clock sources.

Asynchronous clocks have the following properties:

- No common base period.

  Because they have no common base period, STA tools assume that the phase shift between asynchronous clocks is zero.

- Data transfer between asynchronous clocks is unreliable.

  In most designs, where there is no transfer of data between asynchronous clocks, functional false paths are set between those clocks. If there is data transfer between those clocks, then there must be checks in place to verify the integrity of the data.

 © Cadence Design Systems, Inc. All rights reserved.

When you have multiple clocks in a design, you can verify any transfer of data between these two clock domains easily when those clocks are synchronous. But when they're asynchronous, it's possible that (by asynchronous, we mean the clocks are generated from different clock sources) there is no common base period or there is no common LCM. Because there is no common base period, the tools assume that the phase shift between these asynchronous clocks is zero. Any data transfer that happens between asynchronous clocks is unreliable. So, you need to define false paths or functional false paths between these clocks so that there is no data transfer between these asynchronous clocks because they cannot be verified. Some designs have synchronizers to synchronize asynchronous clocks. But that is outside the scope of this course.

© Cadence Design Systems, Inc. All rights reserved.

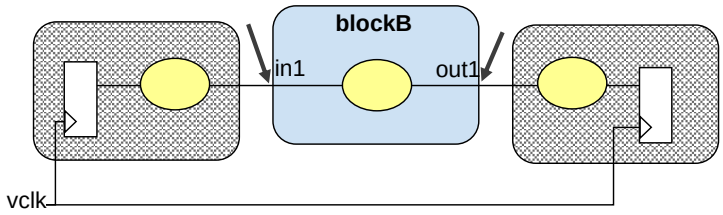
This document is for the sole use of Le Khanh Nam of Gmail

What Are Virtual Clocks?

Use virtual clocks when you are designing a combinational path that is relative to a clock outside of your design or when there is no actual clock source in your design.

Assume that the block to be synthesized and timed is *blockB*.

- You can define a virtual clock with the *create_clock* command, but you do not bind it to a port.
- The clock signal, *vclk*, represents a virtual clock.
- The input delay and output delay are relative to the virtual clock.



 © Cadence Design Systems, Inc. All rights reserved.

Virtual clocks can be used to constrain combinational paths and make them meet certain requirements. If there is no actual clock source in your design, to make that block or combinational path meet a certain requirement, you can use these virtual clocks. In this example, if Block B is to be synthesized and timed, then you can create a virtual clock and set the input and output delay relative to that virtual clock.

Because these virtual clocks do not have a port association in your design, you can just use the create-clock command without binding it to any port or pin.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Are Path Exceptions?

When you must ask the STA tool to create exceptions to the normal timing calculations, you do that via *path exceptions*.

These are the most common path exceptions used in the timing tools:

- Multicycle paths
- False paths
- Disabling timing arcs
- Case analysis
- Path delay limits

 © Cadence Design Systems, Inc. All rights reserved.

Because the STA tool creates exceptions to the normal timing calculations for all the following items, they're called path exceptions. Multicycle paths are purposely additional cycles allowed for a certain path. False paths are something that you don't want the tool to time. We will look at each of these in more detail in the following slides.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Example SDC File

```
create_clock -period 7 -waveform {0 3.5} [get_ports {clk_fpci66m}]
create_clock -period 14 -waveform {3.1 10.1} [get_ports {clk_ref25m}]
create_generated_clock -name GCLK1 [get_pins {pci_if_top1/pci_datapath1/master_addr_cnt1/pci64_xfer_reg}] \
        -source [get_ports {clk_ref25m}] -edges {1 2 3}
set_clock_transition 4 [get_clocks clk_pci]
set_case_analysis  0 [get_ports TE1]
set_multicycle_path  -from [get_cells subblk/data*] -to [get_cells subblk/c_reg*] -hold 1
set_multicycle_path -through [get_nets pci_am_block/mul*] -setup 2
set_false_path -from [get_cells pci_am_block/r_gpc_reg] -to [get_cells pci_am_block/gpx_reg]
set_clock_latency 4 [get_clocks {clk_ref25m}]
set_clock_latency -source 1 [get_clocks GCLK1]

set_input_delay 2.5 -clock "clk_ref25m" [get_ports {ipmitxbfr_rdata[0]}]
set_output_delay 3 -clock "clk_pci" [get_ports {decalfipmi_fllen_z}]
set_max_fanout 2 [get_ports {ipmitxbfr_rdata[4]}]
set_driving_cell -lib_cell BUFX8_TAX0  -library xlite_core [get_ports {ipmitxbfr_rdata[3]}]
set_wire_load_model  -name "pci_block_wl" -library "xlite_core“
set_load -pin_load 0.2 [get_ports {decalf_hdrmem_ld}]
```

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

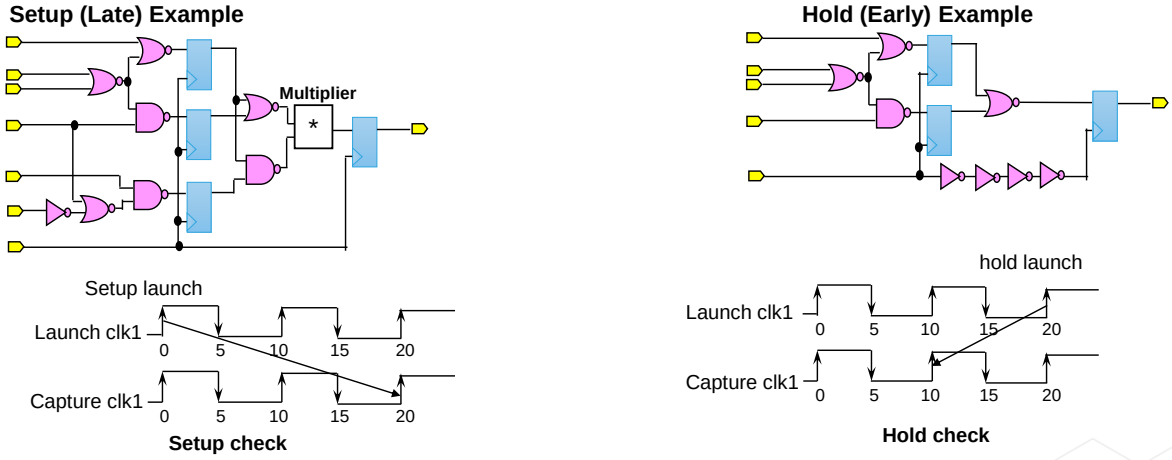
This example SDC file shows how SDC constraints are put together for a sample design. The clocks are defined, then a few path exceptions in blue, and then the input and output delays are shown in this example SDC file.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Are Multicycle Paths?

By default, static timing tools assume all timing paths will meet timing successfully in one cycle. Those paths that require more than one clock period for execution are called multicycle paths.



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

As you understand from the previous topics we've discussed, by default, STA tools assume that all timing paths are to meet timing in one clock cycle when you have a single clock design. Now, there are paths that require more than one clock cycle, and such paths are called multicycle paths. Some multipliers are pipelined to one cycle, but when the output of the multiplier does not require pipelining, it must be considered a long path, and a multicycle path must be defined through the path. The rest of the design must be considerate of this known long path and use its output only after certain cycles. Then it becomes a known multicycle path. For such multicycle paths, you don't want the synthesis tool to waste any resources, trying to figure out the best way to optimize this path to fit into one cycle. Therefore, you make an exception. You tell the tool that this is a known long path and that it is a multicycle path so that it can skip optimizing it for one clock cycle. In the setup example on the left, we tell the optimization and timing tools that this path is a multicycle path and that it will require two cycles instead of one for the setup check to happen.

You also need to tell the tools when you have a hold multicycle path. Here the clock path is longer than the data path. Therefore, in this example, you verify the hold not at the launch edge but at the edge before that.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Multicycle Paths

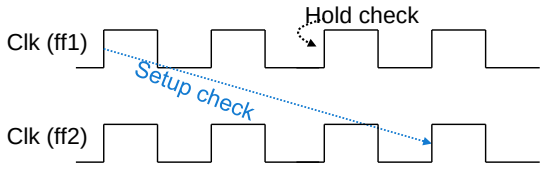
The *set_multicycle_path* command specifies that designated timing paths in the current design meet setup and hold checks in multiple clock cycles.

```
set_multicycle_path [-setup | -hold] <value> [-start | -end] \
  [-from <list>][-through <list>] [-to <list>]
```

The following command tells the tool to check for a setup violation three cycles after the launch clock edge. All paths between ff1 and ff2 are 3-cycle paths.

```
set_multicycle_path -setup 3 -from {ff1} -to {ff2}
```

In some tools, if this setup multicycle command is issued alone, you then need to reset the hold if you don't want a hold multicycle.



 © Cadence Design Systems, Inc. All rights reserved.

You can use the set-multicycle-path command to set multiple cycles for a particular path. In the examples shown, we are setting a setup multicycle path. In some tools, setting the setup multicycle path might set hold as well. The example shown increases setup check by 2 more cycles, but also moves the hold check by -2 cycles.

Thus, this command sets a hold multicycle constraint on the path. Therefore, in paths that do not have a hold multicycle, such as paths in a multiplier, you need to reset the hold. These tools will now check the hold, one cycle before the setup check. If you don't want that to happen, to only modify the setup multicycle, let the tool know by resetting the hold multicycle.

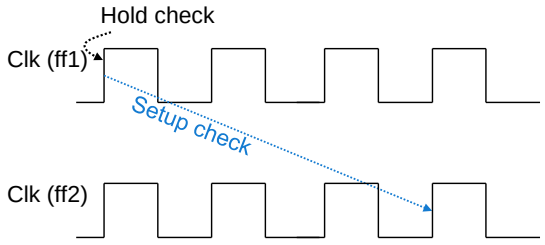
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting a Multicycle Path: Resetting Hold

When you want to set a multicycle path for setup only but not for hold, use both of these commands together:

```
set_multicycle_path -setup 3 -from {ff1} -to {ff2}
set_multicycle_path -hold 2 -from {ff1} -to {ff2}
```



Make sure to check tool-specific behaviors for these path exceptions.

The commands set all paths between ff1 and ff2 to 3-cycle paths for setup, but the hold check is now moved back to the launch edge of the clock.

By setting the hold, in effect, you are canceling the hold multicycle path that is set by the *set_multicycle_path -setup* command.

 © Cadence Design Systems, Inc. All rights reserved.

As mentioned before, using a setup multicycle path also changes the hold check. By default, most tools check the hold one cycle before the setup check.

To specify a setup only multicycle, you might want to reset the hold multicycle. To reset the hold for the multicycle path, specify a hold-multicycle-path along with the setup multicycle path. The example shows how to reset the hold check back to the launch edge of the start clock. In this case, setting the hold multicycle path, cancels the effect of setup check modifications to the hold from the multicycle path command.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Multicycle Paths for Multiple Clocks

When a multicycle path spans two clocks, use the *-start* or *-end* options with the *set_multicycle_path* command to tell the STA tool which clock the check will be applied to.

Example

This multifrequency example shows a 20 ns clock to 10 ns clock data transfer with a multicycle path of 2 cycles from ff1/CP to ff2/D.

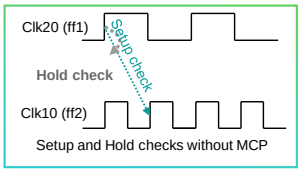
```
set_multicycle_path -start -setup 2 -from ff1/CP -to ff2/D
```

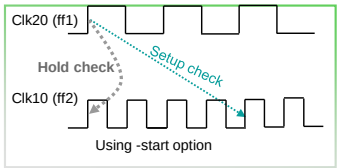
```
set_multicycle_path -hold 1 -from ff1/CP -to ff2/D
```

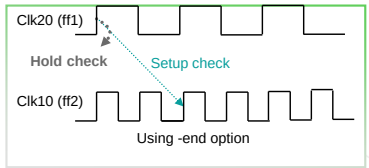
The following commands apply the constraints in the ending clock:

```
set_multicycle_path -setup 2 -from ff1/CP -to ff2/D
```

```
set_multicycle_path -end -hold 1 -from ff1/CP -to ff2/D
```







 © Cadence Design Systems, Inc. All rights reserved.

For multiple clocks, you will use the -start and -end options along with the set-multicycle-path command. By default, the setup checks are always relative to the end clock or the capture clock, and the hold checks are relative to the start clock. In the left-most graphic, there is no multicycle path constraint. Here, the setup check is done at the first cycle after the launch clock, and the hold check will be done at the launch clock itself or the same edge.

The middle graphic shows the case for the first command example-set, where we have a multicycle path constraint with the -start option, meaning that the tool must use 2 cycles of the start clock. Therefore, the setup check is done at 2 cycles of the start clock or at 4 cycles of capture clock.

But if you use the -end option as in the second command example-set, it will use two cycles of the end clock, and the setup check is done after two cycles of clock10. The right-most graphic shows that scenario.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

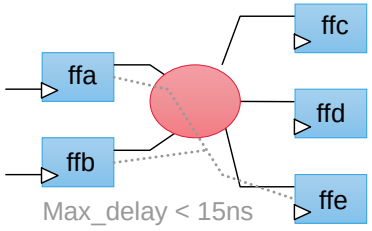
Setting Maximum Path Delay

The set_max_delay command specifies maximum delay for paths.

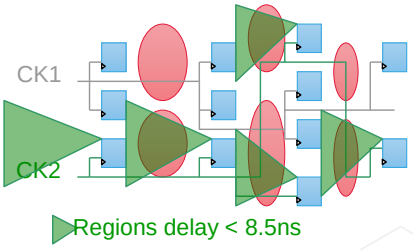
```
set_max_delay [-rise] [-fall] [-from from_list] \
  [-to to_list] [-through through_list] delay_value
```

The synthesis tool optimizes the path to meet the requirement set by the max delay command. The STA tool uses the max delay in its calculation in ideal mode but not in propagated mode.

set_max_delay 15.0 -from {ffa ffb} -to {ffe}



set_max_delay 8.5 -to [get_clocks CK2]



 © Cadence Design Systems, Inc. All rights reserved.

You can set the maximum delay to optimize the delays along the path to meet timing requirements.

The example command on the left requires all paths to FFE have a path delay of less than 15ns.

The command on the right requires all paths to the endpoints to clock 2 have a path delay of less than 8.5ns.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Minimum Path Delay

The *set_min_delay* command specifies a minimum delay for timing paths.

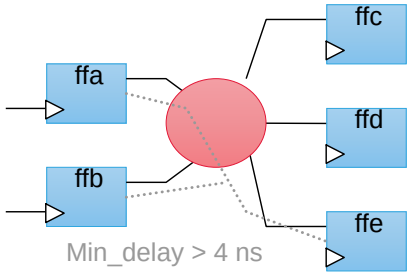
set_min_delay

This command forces all paths from *ffa or ffb to ffe* to have a minimum path delay of 4ns for the purpose of timing analysis.

```
set_min_delay 4 -from {ffa ffb} -to {ffe}
```

It is used in two cases:

- To set a target minimum delay for output ports in a combinational design.
- To override the default single cycle timing for paths, where *set_multicycle_path* is insufficient.



 © Cadence Design Systems, Inc. All rights reserved.

The min delay command can set the target minimum delay for the output ports in a combinational design.

You can also use the set-min-delay command to override the default single-cycle timing where the multicycle path is insufficient.

The example command requires that all paths from FFA and FFB to FFE have a minimum delay of 4ns.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Setting Case Analysis

To set constant or transitional values on a port or a pin, use:

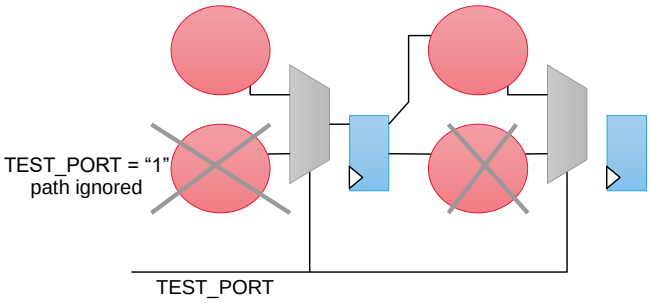
```
set_case_analysis
```

This command tells the tool to ignore all timing paths when the TEST_PORT is 1.

```
set_case_analysis 0 TEST_PORT
```

The specified constants or transitional values do not alter the netlist.

The value set by the *set_case_analysis* command has priority over the logical value from the netlist.



 © Cadence Design Systems, Inc. All rights reserved.

The set-case-analysis constraint is propagated throughout the netlist, where the timing tool behaves as if that constant has deleted or optimized that piece of the design. The actual constants are propagated through the netlist. The netlist is not altered, but the timing tool behaves as though the netlist has been altered. In the example, when you set the case analysis to zero on the test port signal, the timing tool will ignore all the paths for when the test port is one. All the crossed-out logic doesn't exist according to the timing tool. A good use of the case analysis is to prevent large transition times on high fanout synchronous nets. The set-false-path command will not prevent transition times from propagating, whereas setting case-analysis will. It is better to set the case analysis on such nets.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

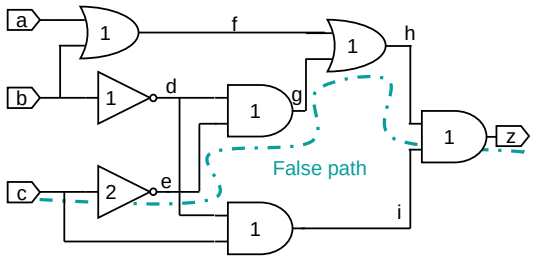
What Are False Paths?

A false path is a path that can never be timed properly in the actual circuit. These false paths are those that are logically not feasible, or timing them is not feasible.

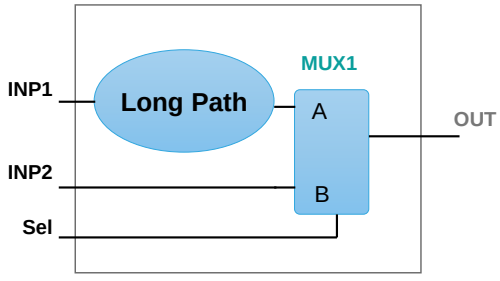
You need to tell the synthesis tool that the LONG path in this illustration must be set as a false path if it is never realized in your design.

For example, paths from an asynchronous reset port are usually set as false paths.

It is important in static timing analysis to avoid analyzing false paths to save on optimization time.



Example of Functional False Paths



 © Cadence Design Systems, Inc. All rights reserved.

False paths are paths that either cannot or should not be analyzed by the STA tool.

It is important to eliminate false paths so that the tool does not spend cycles trying to analyze them.

Some Cadence software automatically flags these paths as false, removing the need for in-depth analysis of such paths by the logic designer. However, you need to run these tools in order to identify those paths before you run static timing analysis on your design.

For example, in the functional false path illustration, the path that goes from C->E->G->H->Z is the longest path. But after you do a bit of analysis, you can determine that C rising to Z falling and C falling to Z rising paths that have a delay of 5 units are false paths and can never be realized. The path with a delay of 4 units becomes the real critical path. The STA tool, the synthesis tool, and the place-and-route tool know that the critical path that is identified immediately is not the real critical path. Rather, the one before or below that with the 4-unit delay is the real critical path and is essential to avoid wasting the tool's time and resources to find a better fit for it to work on other paths.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

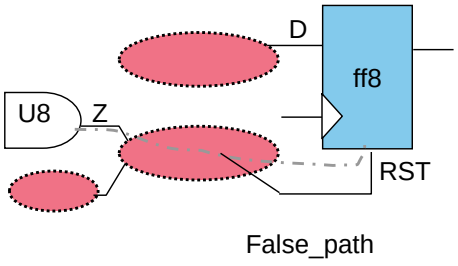
Setting False Paths

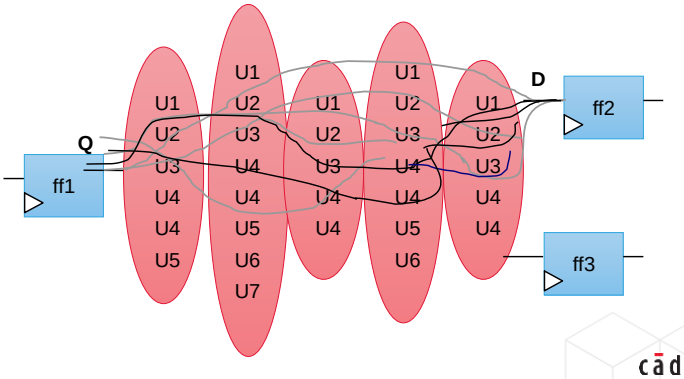
The *set_false_path* command marks the start point and end point pairs as false timing paths.

```
set_false_path [-setup] [-hold] [-rise] [-fall] \
  [-from from_list] [-to to_list] [-through through_list]
```

```
set_false_path -from U8/Z -to ff8/RST
```

```
set_false_path -from ff1/Q -through \
{U1/Z U2/Z} -through {U3/Z U4/C} \
-to ff2/D
```





 © Cadence Design Systems, Inc. All rights reserved.

The set-false-path command can mark the paths in your design that are specified using the start point and end point pairs as false paths. Use a through-list to set specific paths as false paths.

In the example command on the left, we're declaring that the path from U8/Z to ff8/reset is a false path. Such false paths are thus not timed. They are ignored by the tool and save you time.

Thus, it is important to specify false paths very carefully because multiple paths may be affected.

In the example on the right, ff1 to ff2 have more paths than what you might want to set as false paths. You might accidentally disable all the paths from ff1/Q to ff2/D when you only wanted to set false paths from ff1 through U1 or through U2. Thus, make sure that your false paths are set appropriately.

The timing tool will ignore these paths from either optimization or timing. False paths do take precedence over multicycle paths. You can also specify other settings like max delay or min delay, which can override a false path command. To avoid these kinds of precedence issues, look at your tools and see what their default settings are and how to avoid these pitfalls.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Disabling Timing Arcs

To disable the timing arcs of an instance or of an interconnect, use:

```
set_disable_timing
```

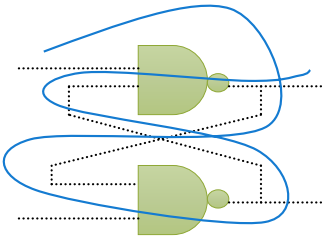
This command differs from the *set_false_path* command as follows:

- The *set_disable_timing* command disables a timing arc so that neither arrival times nor constants propagate through it.
- The *set_false_path* command disables only the arrival time information. Constants are not affected by this command.

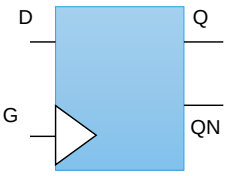
How do I make a latch transparent?

```
set_disable_timing -from latch1/G -to
latch1/QN
```

Use to break combinational loops.



Latch



 © Cadence Design Systems, Inc. All rights reserved.

If you don't want to use a specific timing arc of a particular cell because you either want to improve the timing or you just don't care to use that specific arc in that cell, you can choose to use the set-disable-timing command, instead of the set-false-path command. The set-disable-timing command disables the timing arc so that neither the arrival times nor the constants propagate through it. Whereas with the set-false-path command, it only disables the arrival time information, but constants are not affected. So, any constants that you set might propagate through that timing arc.

In designs with combinational loops, it is common to break the combinational loops using the set-disable-timing command because STA tools cannot time combinational loops.

In order to make a latch transparent, you can use the set-disable-timing command. When you do that, the path in this example, from latch1/G to latch1/QN, is disabled. Only the D to QN timing arc is transparent. The tool propagates the value of D to QN regardless of the value of G. This will allow the latch to act as a buffer. With this set-disable-timing command, you are asking the timing or optimization tool to pretend as if a timing arc doesn't exist even though the timing arc exists in the library cell.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Identifying Design Objects

Which of the following are design objects?

1. Timing Constraints
2. Pin
3. Clock
4. Design
5. Timing arc
6. Net
7. Block
8. Library files
9. Clock jitter
10. Wire-load models

In a minute, the answer will appear on the page. Move the time slider to the end, if you want to see the answer sooner.

 © Cadence Design Systems, Inc. All rights reserved.

Try the following activity to improve your understanding of the topics discussed.

Solution

2, 3, 4, 6, 7

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Matching Design Objects to Constraints

Match the objects on the left to the best related constraints on the right.

1. Pin	A. Uncertainty and latency
2. Net	B. Not a design object
3. Port	C. Power constraints
4. Design	D. Wire-load models
5. Timing arc	E. Environmental constraints
6. Clock	F. Timing constraints
7. Cell or Block	G. Operating conditions

In a minute, the answer will appear on the page. Move the time slider to the end, if you want to see the answer sooner.

 © Cadence Design Systems, Inc. All rights reserved.

Try the following activity to improve your understanding of the topics discussed.

Solution

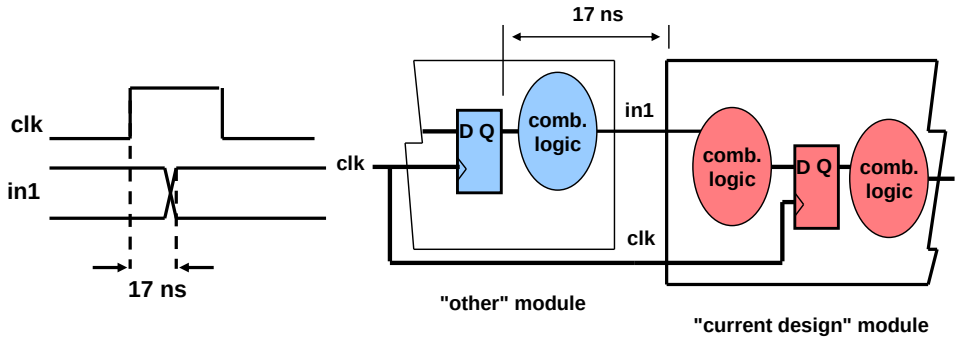
1. F, 2. D, 3. E, 4. G, 5. B, 6. A, 7. C

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Setting Input Delay

Specify the constraint for an input delay of **17ns** at **in1** relative to the **clk** clock.



 © Cadence Design Systems, Inc. All rights reserved.

Try the following activity to reinforce what you just learned.

Solution

```
set_input_delay 17 -clock clk { in1 }
```

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Creating a Clock

Problem

A design has a clock *clk1*, with a period of 10ns, and a duty cycle of 40%.

1. What is the command to set the **clk1** clock with the above specification?

   create_clock _____

2. Is this waveform correct for **clk1**?

 © Cadence Design Systems, Inc. All rights reserved.

Take a minute to try out this activity.

Solution

1. create_clock -name clk1 -period 10 -waveform {0.0 4.0}

2. Yes, because the rising and falling edges are not defined in the problem statement.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Clock Latency

This timing report from Cadence® Genus® Synthesis Solution reports a setup check.

```
 Pin              Type    Fanout Load Slew      Delay    Arrival
                                (fF)  (ps)       (ps)      (ps)
g206/A    <<<                                     +0    22068
g206/Y           xor2      1    20.7    58      +146    22214 R
mul_21_16/Z[7]
in4r_reg_7/D    fflopd                           +0    22214
in4r_reg_7/CK   setup 20000                     +100    22314 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clock)     capture                                1000 R

-------------------------------------------------------------
Timing slack : -1314ps (TIMING VIOLATION)
Start-point  : in4[1]
End-point    : in4r_reg_7/D
```

Are the following statements true for this timing report? Why or why not?
Early source latency = 12000 ps; early network latency = 8000 ps;
Late source latency = 10000 ps; late network latency = 9000 ps.

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

cadence

Try the following activity to improve your understanding of the topics discussed.

Solution

No. The timing report is for a setup check. The sum of late source and late network latencies equals the value of latency specified above.

Setup checks are turned on by default in most tools. Hold analysis is usually done separately after the clock tree is built.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

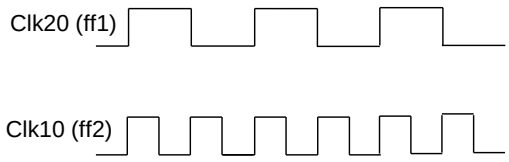
Activity: Setting Multicycle Paths

1. For this illustration, if the following command is set, what is the number of **Clk10** cycles for the setup check?

```
set_multicycle_path -start -setup 2 -from { ff1 } -to { ff2 }
```

2. If the following commands is also set, what is the number of **Clk10** cycles for the hold check?

```
set_multicycle_path -hold 1 -from { ff1 } -to { ff2 }
```



 © Cadence Design Systems, Inc. All rights reserved.

Do the following activity to reinforce the topics just discussed.

Solution

1. 4 cycles

2. 0 cycles

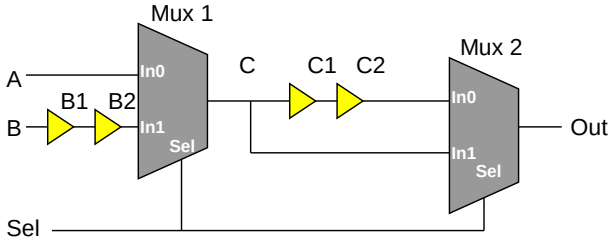
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Identifying a False Path

There are four timing paths through Mux 1 and Mux 2.

- Path 1: A – C – C1 – C2 – Out
- Path 2: A - C – Out (through In1 of Mux 2)
- Path 3: B - B1 – B2 – C – C1 – C2 – Out
- Path 4: B – B1 – B2 – C – Out (through In1 of Mux 2)



Can you identify the false paths?

Tip: The *Sel* signal controls the output of the multiplexers.

 © Cadence Design Systems, Inc. All rights reserved.

Try this activity to reinforce the topics just discussed.

Solution

The select signal, Sel, drives both the multiplexers.

This design can have only two possible active timing paths at any given time:

- When Sel = 0, path 1 is active.
- When Sel = 1, path 4 is active.

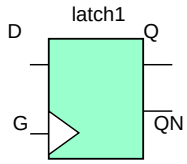
Thus, path 2 and 3 are false paths.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Disabling Timing Arcs

How would you make both output pins latch1/Q and latch1/QN transparent?



 © Cadence Design Systems, Inc. All rights reserved.

Try this activity to reinforce the topics just discussed.

Solution

The simplest solution is to use the following command:

```
set_disable_timing -from latch1/G
```

When a latch is transparent, it behaves like a combinational gate, and it passes constants through the gate. When a latch is opaque, nothing can propagate through it.

© Cadence Design Systems, Inc. All rights reserved.

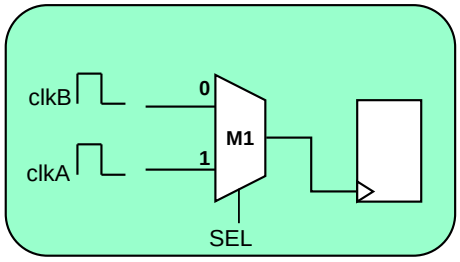
This document is for the sole use of Le Khanh Nam of Gmail

Activity: Setting Case Analysis

In the MUXed clock example below, assume that you do not want to use *clkA* for STA.

1. What is the command to use?

   set_case_analysis _____ _____



2. Are MUXed clocks okay to use?

 © Cadence Design Systems, Inc. All rights reserved.

Do the following activity to reinforce the topics just discussed.

Solution

1. Use set_case_analysis to set 0 or 1 on the select signal SEL, depending on whether the clkA signal feeds the 1 input or the 0 input of the MUX.

   set_case_analysis 0 M1/SEL

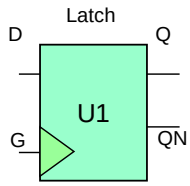
2. Yes. Muxed clocks are okay to use as long as you specify the case analysis.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Activity: Setting Another Case Analysis

You can use *set_case_analysis* command to make a latch transparent.



Specify the command to use for this example:

set_case_analysis _____ _____

 © Cadence Design Systems, Inc. All rights reserved.

Do the following activity to reinforce the topics just discussed.

Solution

Set 1 on the enable signal U1/G as in this example:

set_case_analysis 1 U1/G

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module 9

Analyzing a Timing Report

Let's look at how to analyze a timing report.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Module Objectives

In this module, you

- Identify some timing analysis strategies
- Identify the essential parts of a timing report
- Analyze timing reports

 © Cadence Design Systems, Inc. All rights reserved.

In this module, we'll identify timing analysis strategies and identify the essential pieces of a timing report. Then, we'll look at how to analyze some example timing reports.

© Cadence Design Systems, Inc. All rights reserved.

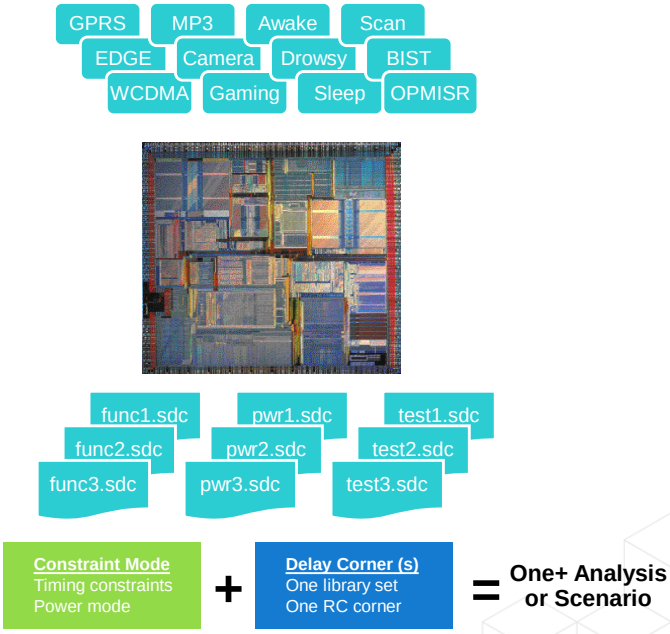
This document is for the sole use of Le Khanh Nam of Gmail

What Are Timing Analysis Modes?

Each of the multiple modes and multiple corners in your design can be analyzed for timing in various forms depending on the resources and time you have to perform the task.

The following are the most commonly used timing analysis modes.

- **Single Analysis Mode**
  - One delay corner: Commonly used by synthesis tools in initial runs
- **Best Case/Worst Case Analysis Mode or BC/WC Mode**
  - Two extreme delay corners: Commonly used for design feasibility analysis
- **On-Chip Variation Analysis Mode or Min/Max Analysis Mode**
  - Two delay corners: Used for timing signoff



 © Cadence Design Systems, Inc. All rights reserved.

You might know that chips today have multiple standards, functionalities, and power profiles, as well as different test modes and multiple constraint sets.

In today's designs, the analysis must be done concurrently on all modes and corners.

So how do you satisfy all the constraints simultaneously and implement the chip while satisfying all these modes? Some tools offer multimode multi-corner analysis, starting with the basic building blocks of one mode and one delay corner.

Each analysis view or scenario is often referred to as can be analyzed using the three modes described here.

There are three types of analysis modes that are commonly used in static timing analysis tools. The single analysis mode, the best-case worst-case analysis mode, and the on-chip variation (OCV) mode.

The choice of analysis mode influences where the timing information is coming from, how that information is used for delay calculation, where the timing constraints are coming from, and how they're interpreted. The on-chip variation is the most commonly used one for signoff analysis.

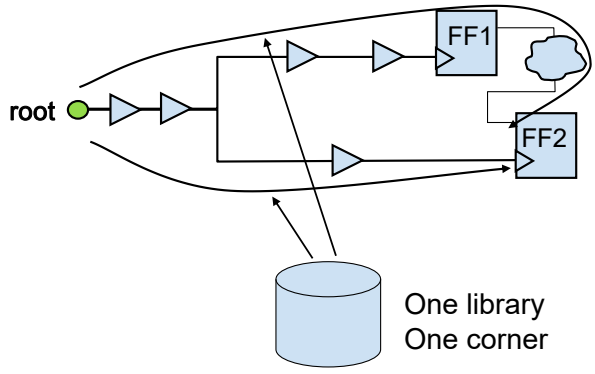
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Single Analysis Mode

In the single analysis mode, the same library is used for both clock paths. This is the most simplistic type of analysis because it does not consider any on- or off-chip variation.

- A single operating condition is used to scale delay values.
- Single mode setup and hold clock latencies will be identical unless the clock tree has reconvergent logic.



 © Cadence Design Systems, Inc. All rights reserved.

The single mode is essentially the max-only delay mode. If you're using the max-only library, that is the only library where the delay information is going to come from. Only the constraints related to max or unspecified are the ones that are used. If you are going to use a single delay calculation path, the slowest slews are the only ones that are propagated. This is all for setup. Even when you switch to hold, the timing analysis will still be done with the max data and constraints. When the delays are taken for these buffers in this design, only the max library delays are the ones that are used.

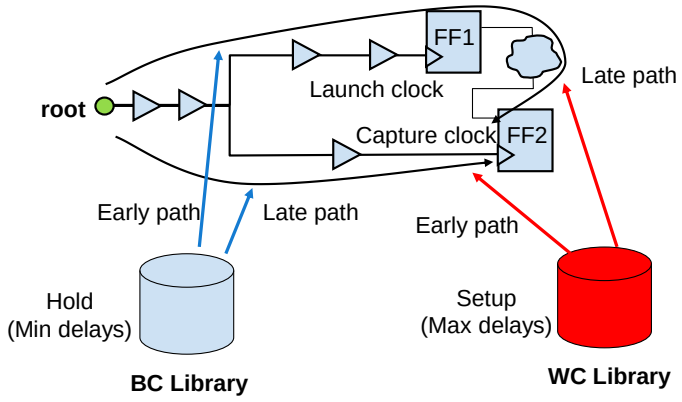
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Best-Case Worst-Case Analysis Mode

Using both BC and WC libraries together is one way to model off-chip variation of delays due to external temperature and voltage variations.

- Analyzes off-chip variation.
- Operating conditions are two extremes.



 © Cadence Design Systems, Inc. All rights reserved.

Using both best-case and worst-case libraries together is one way to model off-chip variation of delays due to external temperature and voltage variations.

Max libraries are normally used for max delays, and min libraries are used for min delay calculation.

From the graphic, the early and late paths for each analysis are calculated from different libraries creating a worst-case condition that is very pessimistic.

The setup calculation can be made worse by using the late path calculation from the worst-case library and the early path calculation from the best-case library.

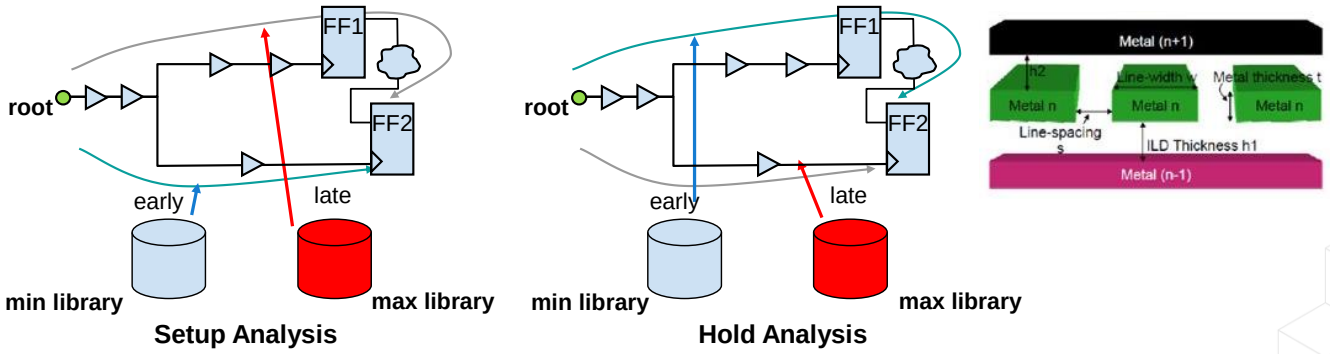
Depending on how these libraries are calibrated, this could create a very unrealistic scenario and therefore this analysis mode is seldom used.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

On-Chip Variation Analysis Mode

- Changes in physical parameters, such as variation in the thickness of layers, cause delay variations on the chip for the same cell placed in different locations. OCV models such variation.
- In this OCV analysis mode, the software calculates the delay for one path based on maximum operation conditions while calculating the delay for another path based on the minimum operating conditions for setup and hold checks.



 © Cadence Design Systems, Inc. All rights reserved.

Process variation causes physical parameters like thickness, width, or spacing between layers across the chip to vary and therefore cause delay variations. Therefore, it is common to have two buffers using a similar library cell to have slightly different delays when one of the buffers is in one part of the chip and another elsewhere because of process variation. In the on-chip variation mode, you take the delay values for each buffer from different libraries. This variation in the delays of the buffers from different regions of the chip is captured via these early and late libraries because the libraries are characterized accordingly and, therefore, can be used for these purposes. In the graphic, the early values are from the min or early library, and the late values are from the max or late library. Setup analysis uses the “late or max” library for the launch clock path and uses the “min or early” library to calculate capture clock path delays. Hold analysis uses the min library for the launch clock path and the max library for the capture clock path. This is how OCV models delay variation across the chip.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

What Is Statistical OCV (SOCV)?

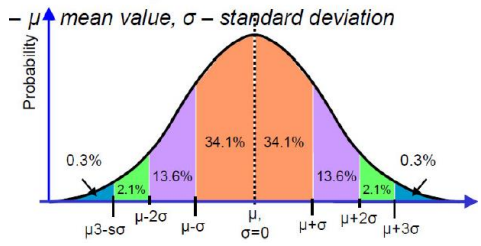
SOCV is a variation modeling technique that computes the impact of local process variations on the delay and slew of each instance in the design at a given global variation corner. SOCV propagates the sigma of arrival and required times through the timing graph and then computes statistical characteristics of slack at all timing pins.

SOCV solves some shortcomings of OCV analysis:

- Inefficiency at ultralow voltage operation ($V_{operation} \sim V_{threshold}$)
- Cell timing dependency on slews and loads

SOCV analysis requires variation libraries in the form of a Liberty Variation Format (LVF) library.

LVF includes arc-level absolute variations of cell delays, output transitions, and timing checks as functions of input slew and load.



 © Cadence Design Systems, Inc. All rights reserved.

The regular OCV analysis is inefficient at ultra-low voltage operations because of the inability to model local slews and load variations. The depth of the path plays a role in the accurate modeling of the delays along the path. A simple two-input two-cell path has a different slew and load at each point along the path than a path with N number of cells of varying inputs. SOCV or Statistical OCV can model delay variations based on local slews and process variations. SOCV can also provide slews and loads needed to time the cells at those voltages. To do this, SOCV analysis requires process variation libraries known as Liberty Variation Format or LVF. LVF includes timing-arc-level absolute variations of cell delays, output transitions, and timing checks as functions of inputs slew and load to better model the local data at each cell. SOCV analysis uses LVF data to propagate the sigma of arrival and required times through the timing graph and then computes the slack as statistical sigma and mean values at all the timing pins, therefore giving you a more accurate picture of your timing information.

© Cadence Design Systems, Inc. All rights reserved.

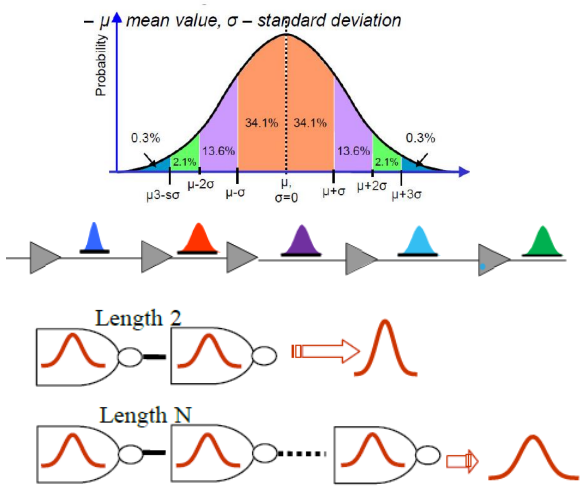
This document is for the sole use of Le Khanh Nam of Gmail

Liberty Variation Format (LVF)

```
cell (cell_name) {
ocv_derate_distance_group: ocv_derate_group_name;
...
pin | bus | bundle (name) {
direction: input | output;

timing() {
  ...
   ocv_sigma_cell_rise(delay_lu_template_name){
     sigma_type: early | late | early_and_late;
     index_1 ("float, ..., float");
     index_2 ("float, ..., float");
     values ( "float, ..., float", \
         ..., \
         "float, ..., float");
  }

   ocv_sigma_cell_fall(delay_lu_template_name){
      sigma_type: early | late | early_and_late;
      index_1 ("float, ..., float");
      index_2 ("float, ..., float");
         values ( "float, ..., float", \
         ..., \
        "float, ..., float");
     }
    ...
  } /* end of timing */
  ...
  } /* end of pin */
...
```



(Statistical calculations happen during path tracing.)

 © Cadence Design Systems, Inc. All rights reserved.

This example library cell shows the structure of the Liberty Variation Format for a cell. The graphic on the right shows the distribution considered for SOCV. Notice the smoothening and the sharp changes in the waveforms at each cell as the signal proceeds to different cells during the path. The liberty variation format or the LVF considers the OCV sigma cell rise and fall parameters to model cell delays, output transitions, and timing checks as functions of actual input slew and loads of the cell in the path. All the timing-arc-level waveforms of cell delays, output transition, and timing checks are in the form of a lookup table. The graphic also shows that the cell delay etc., waveforms will be different for two cells versus N number of cells.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

How to Read Timing Reports

Look for essential information in your timing report that can help you debug timing issues.

A typical timing report is divided into three essential sections:

- Slack
- Required Time
- Arrival Time

Verify the Slack (Summary)

The slack section lists the type of check, the start and end points of the path, and the slack for the path. After identifying the type of check, focus first on the paths with a negative slack, i.e., paths that are violating.

Verify the Required Time

The required time section lists the endpoint and captures clock arrival time, including clock uncertainties and latencies and any path exceptions. Make sure that the required time calculation matches expectations from your constraints.

Verify the Arrival Time

The arrival time section includes the start point and the delays along the arrival path, like the cell delays and the net delays, etc. Look for unexpected large delays along the path.

 © Cadence Design Systems, Inc. All rights reserved.

cadence

Typically, any timing report is broken down into the arrival time calculation, the required time calculation and slack. You'll see the arrival time, which includes the start point, any input delay, cell delays, and so on, up to the endpoint. The required time calculation typically shows your capture clock and any uncertainties that are subtracted from the required time. The slack analysis, which shows whether the timing is violated or met, also shows if there is any additional information that is pertinent for your design or path.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Innovus™: Setup Check Report

```
Path 1: MET Setup Check with Pin u55/CK
Endpoint:   u55/D (^) checked with leading edge of 'CLK_W_4'
Beginpoint: u99/Q (v) triggered by leading edge of 'CLK_W_4'
Path Groups: {CLK_W_4}
 Other End Arrival Time          0.150
- Setup                     0.176
+ Phase Shift                10.000
= Required Time               9.974
- Arrival Time              0.827
= Slack Time                 9.148
Clock Rise Edge                     0.000
+ Clock Network Latency (Prop)            0.150
= Beginpoint Arrival Time                 0.150
-------------------------------------------------------------------------
Load   Slew   Delay   Arrival  Cell   Arc          Pin
                Time
-------------------------------------------------------------------------
0.013  0.116  -      0.150    -     CK ^         u99/CK
0.027  0.086  0.195  0.345    DFF   CK ^ -> Q v   u99/Q ->
1.006  1.140  0.482  0.827    INV   A v -> Y ^    u4/Y
1.006  1.140  0.000  0.827    DFF   D ^          u55/D
-------------------------------------------------------------------------
```

Type of check

Required time

Slack

Arrival time

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

The example report shown here is a report generated from the Innovus Implementation System. The report shows that the path has met the timing. The setup check has passed. It also shows the end point and the beginning point. The start point is also known as *begin point*. Both the begin and end points are triggered with the leading edge of CLK-W-4.

The tool has some built-in grouping of paths. The report shows a path grouping called CLK-W-4. Then we see the other end arrival time, which is when the capture clock arrives at the flop. Any network clock buffer delays are included in the other end arrival time. The required time calculation shows the phase shift and setup. Phase shift is one clock period because it's the same clock. The required time calculation shows the other end arrival time minus setup plus phase shift.

Then we move on to the arrival time calculation. This calculation is done via a table at the bottom of the report. It adds delays through all the cells in the path. You bring that value at the end of the table back into the slack calculation by putting the arrival time value there. If you have a launch clock, like this example begin point does; you list the launch clock information. As noted, the launch clock is a propagated clock. There are buffers in the design with a delay of 0.15. That delay is added as the begin point arrival time, which is where the launch clock starts. If you notice closely, you can see the timing path, with the delays starting from the rising edge of the launch clock through the rising clock-to-falling q, then the inverter, up to the point of the D pin of the flop that is the endpoint. The D pin of the flop shows that it is a rising edge. The report adds the delays for all the timing arcs for the cells in the timing path. The delay calculation at each timing arc includes the load, the slew, and the unateness of the cell. The report utilizes all this information and arrives at the conclusion that the timing has been met.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Innovus: Hold Check Report

```
Path 1: VIOLATED Hold Check with Pin seg1/u3/CK
Endpoint: seg1/u3/D (v) checked with leading edge of 'CLK_W_1'
Beginpoint: IN1    (v) triggered by leading edge of '@'
Path Groups: {CLK_W_1}
Other End Arrival Time        0.682
+ Hold                       -0.034
+ Phase Shift                 0.000
- Cycle Adjustment           -20.000
= Required Time              20.648
Arrival Time                  0.117
Slack Time                   -20.531
Clock Rise Edge                    0.000
+ Input Delay                      0.000
+ Drive Adjustment                -0.014
= Beginpoint Arrival Time         -0.014
 ------------------------------------------------------------------------ --------
Load   Slew   Delay   Arrival   Cell     Arc          Pin
              Time
 ------------------------------------------------------------------------ -------
0.013  0.029  -      -0.014   -        IN1 v        IN1
0.013  0.057  0.054  0.040    CLKBUF   A v -> Y v   seg1/u1/Y
0.003  0.050  0.077  0.117    CLKBUF   A v -> Y v   seg1/u2/Y
0.003  0.050  0.000  0.117    DFF      D v          seg1/u3/D
 ------------------------------------------------------------------------ -------
Applied exceptions:
 ---------------------------------- ---------------
From              Early          Late
 ---------------------------------- ---------------
IN1               -              cycles 2
 ---------------------------------- ---------------
```

Type of check

Other end arrival time

Required time

Slack

Arrival time

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

This report here shows a hold check in the Innovus Implementation System.

Path 1, in this case, has violated the hold check. The tool reported where that hold check was violated and that the endpoint was checked with the leading edge of CLK-W-1. The beginning point is IN1. So, this is an input to the flop path. The report also shows the other end arrival time value. This is important, especially when you don't have the full launch clock information. The other end arrival time is the capture clock latency, that is, the delay through the buffers to the other end, which is the capture clock. In this case, the tool shows multicycle paths via the cycle adjustment, and that cycle adjustment is shown as minus 20. The required time is a combination of the other end arrival time plus your hold check plus a phase shift of zero, plus the multicycle path. The required time is given as 20.648.

The arrival time is 0.117 through the path, that is, through all the cells in the path. The launch clock starts at -0.014. That is the beginning point arrival time.

If we look at the arrival time, it's 0.117, but the required time is 20.648, and the slack calculation is done as Arrival time minus Required time. Because it's a negative value, the slack is listed as violated.

So, this path violates the hold check. It is possible that the cycle adjustment was incorrectly set. Analyze the timing reports carefully to make sure that the constraints have been properly applied and that the tool is reporting the path correctly. Identify problems like these and try to resolve them before certifying the design.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Tempus™: Timing Report

```
Path 1: MET Setup Check with Pin ff2/CK
Endpoint:   ff2/D (v) checked with leading edge of 'CK'
Beginpoint: ff1/Q (^) triggered by leading edge of 'CK'
Other End Arrival Time          12000
- Setup                           304
- Uncertainty                    1000
= Required Time                 10696
- Arrival Time                   2373
= Slack Time                     8323
    Clock Rise Edge                          0
    + Clock Network Latency (Ideal)       2000
    = Beginpoint Arrival Time             2000
-----------------------------------------------------------------------
          Pin      Edge   Cell     Fanout Load Slew Delay  Arrival Required
                                     (fF)  (ps)  (ps) Time(ps) Time(ps)
-----------------------------------------------------------------------
       ff1/CK      ^                              0          2000    10323
       ff1/Q       ^      DFFHQX1        1  7.1  160  +310     2310    10633
       b2/A        ^                                +0      2310    10633
       b2/Y        v      INVX1          1  5.7   88   +63     2373    10696
       ff2/D  <<< v       DFFHQX1                     +0      2373    10696
-----------------------------------------------------------------------
Cost Group   : 'CK' (path_group 'CK')
```

Required time

Slack

Arrival time

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

In the example, it is a Tempus Signoff timing report. It may look slightly different from some of the other reports, but all the necessary information is structured in similar ways as you noticed in previous timing reports. Therefore, it is important to identify all the same buckets, the arrival time calculation, the required time calculation, and the slack calculation. In this report, notice what is the start point and the endpoint. What are they called in this report? Also, identify where the slack is. Where is the required time calculation? Look for clock uncertainty right there. Where is the breakdown of arrival time or the arrival time calculation? How are the cells and arcs represented? Notice that rising arcs are identified with a ^ symbol and falling arcs with a V symbol. Look for the delay through each of the cells. Does any cell delay stand out? How does the slew and load of that cell look? What type of cell is it?

Identifying all such information and breaking down a timing report into smaller manageable pieces can help debug the reports easier.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Tempus Report: Effect of Constraints

```
Path 1: VIOLATED Setup Check with Pin DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/\p_reg[31] /CP
Endpoint:   DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/\p_reg[31] /D              (v)
checked with  leading edge of 'm_clk'
Beginpoint: DTMF_INST/TDSP_CORE_INST/DATA_BUS_MACH_INST/\data_out_reg[0] /Q (v)
triggered by trailing edge of 'm_clk'
Path Groups:  {C2C}
Other End Arrival Time          0.000
- Setup                         0.143
+ Phase Shift                   5.500
+ Cycle Adjustment              5.500
= Required Time                10.857
- Arrival Time                 10.858
= Slack Time                   -0.001
     Clock Fall Edge                2.750
    = Beginpoint Arrival Time      2.750
    +-------------------------------------------------------------------------------+
    |                    Instance                     |      Arc      |     Cell     | Delay | Arrival | Required |
    |-------------------------------------------------+---------------+--------------+|
    | DTMF_INST/TEST_CONTROL_INST/MCLK_ROOT            | CN v          |              |       |   2.750 |    2.749 |
    ...

    | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/\p_reg[31]  | D v           | EDFQLVTD1    | 0.001 |  10.858 |   10.857 |
    +-------------------------------------------------------------------------------+
```

- Compute clock (or checking signal) arrival at checking element (= Required time)
- Subtract setup time
- Add phase shift from launch to capture clock event (1 clock cycle of 5.5ns)
- Add another cycle (multicycle path setting from SDC)

 © Cadence Design Systems, Inc. All rights reserved.

By changing the constraints, you end up changing the timing reports. In this case, we added a cycle adjustment of 5.5ns, which is code for multicycle paths. By adding this additional cycle, your timing required time has gone up. The path is still violating, but not by much. Your slack is almost zero, mainly because of the addition of a multicycle path in the constraints, as nothing else has changed. Therefore, it is important that you identify the effect of the constraints and that any incorrect or inconsistent constraints are not what is causing your design to meet or violate the timing. Incorrect constraints can lead to design failures.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Labs

Lab 9-1 Setting Up the Database

Lab 9-2 Creating Constraints

- In this lab, you set the environmental constraints. You edit a constraints.sdc file and add constraints according to a design specification. After editing the file, you compile the design and generate a lint report to see if the constraints are acceptable to the compiler.
- In this lab, you set the timing constraints, including clocks, input delays, and output delays.

Lab 9-3 Analyzing the Timing in a Design

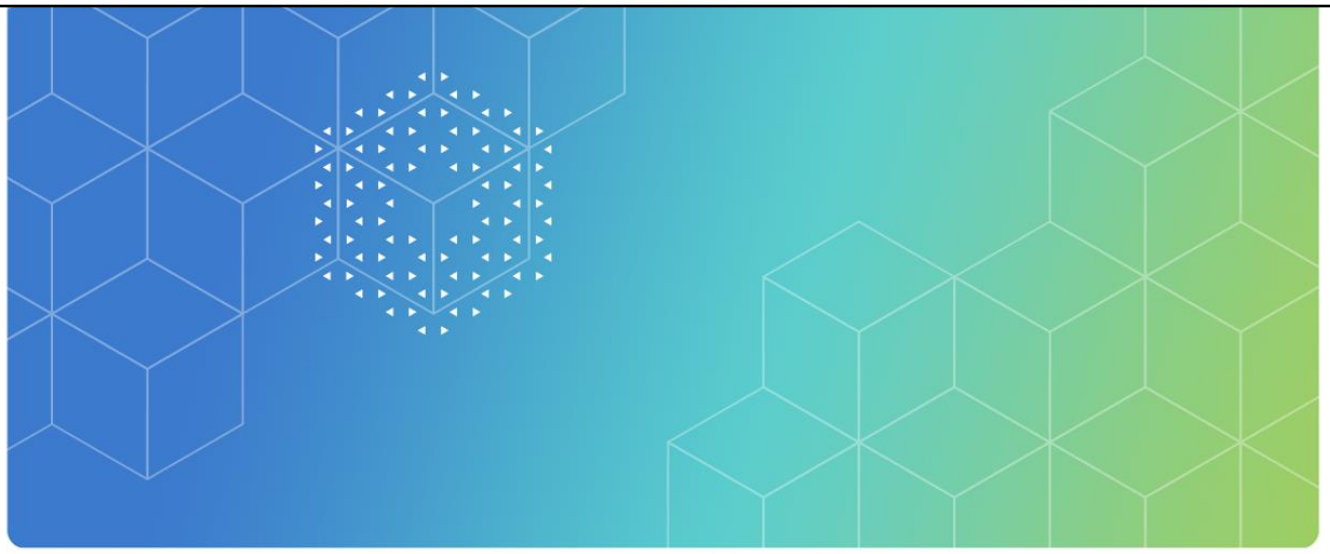
- In this lab, you use the information in a timing report to debug a design that is not meeting timing.

 © Cadence Design Systems, Inc. All rights reserved.

In these labs, you're asked to look at the timing report and debug the design that is not meeting timing.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail



Module 10

Next Steps

Let's look at how to analyze a timing report.

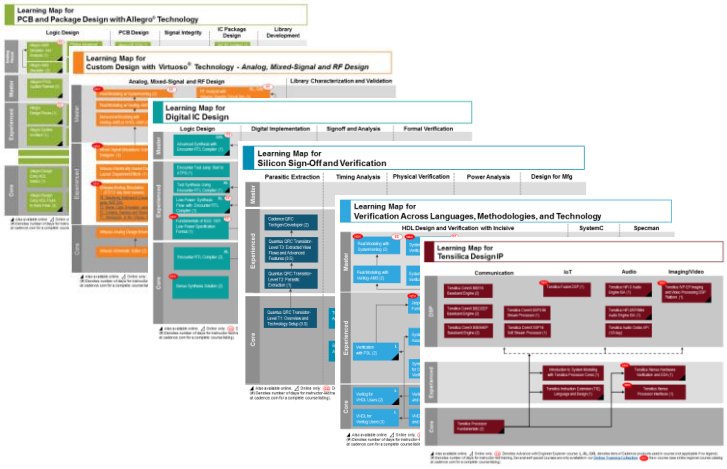
© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Learning Maps

Cadence® Training Services learning maps provide a comprehensive visual overview of the learning opportunities for Cadence customers.

Click here to see all our courses in each technology area and the recommended order in which to take them.



 © Cadence Design Systems, Inc. All rights reserved.

cadence

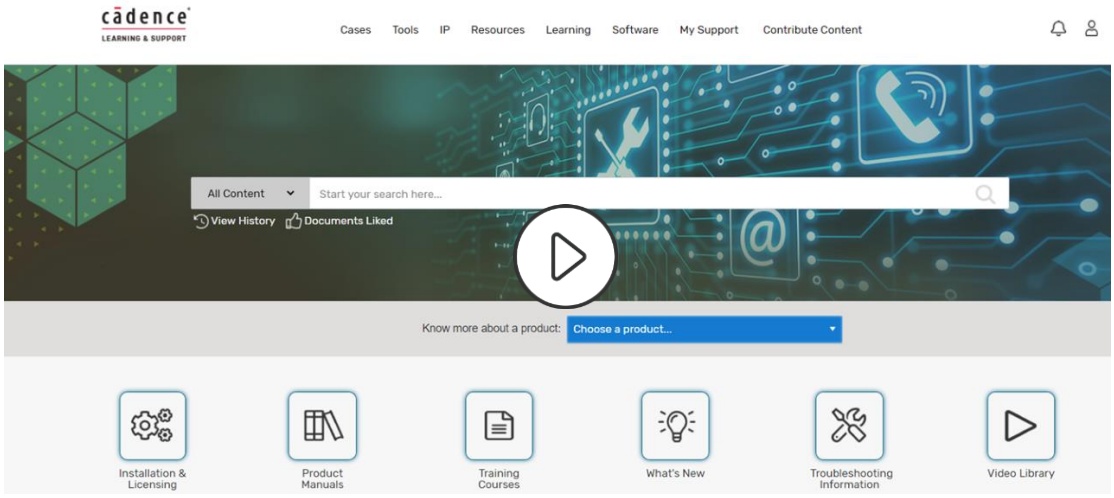
Go here to view the learning maps:

http://www.cadence.com/Training/Pages/learning_maps.aspx

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Cadence Learning and Support



Cadence Support now includes over 2000 product/language/methodology videos ("Training Bytes")!

 © Cadence Design Systems, Inc. All rights reserved.

Click here to view the demo of COS.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

Wrap Up

- Complete Post Assessment, if provided
- Complete the Course Evaluation
- Get a Certificate of Course Completion

Thank you!

 © Cadence Design Systems, Inc. All rights reserved.

cadence

This page does not contain notes.

© Cadence Design Systems, Inc. All rights reserved.

This document is for the sole use of Le Khanh Nam of Gmail

cadence®

© Cadence Design Systems, Inc. All rights reserved worldwide. Cadence, the Cadence logo, and the other Cadence marks found at https://www.cadence.com/go/trademarks are trademarks or registered trademarks of Cadence Design Systems, Inc. Accellera and SystemC are trademarks of Accellera Systems Initiative Inc. All Arm products are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All MIPI specifications are registered trademarks or service marks owned by MIPI Alliance. All PCI-SIG specifications are registered trademarks or trademarks of PCI-SIG. All other trademarks are the property of their respective owners.

This page does not contain notes

© Cadence Design Systems, Inc. All rights reserved.