

Static Time Analysis

How to verify your chip timing

Ahmed Abdelazeem

Faculty of Engineering
Zagazig University

RTL2GDSII Flow, March 2022

Table of Contents

- 1 Introduction
- 2 Timing arcs and unateness:
- 3 TRANSMISSION GATE, D-LATCH, DFF, SETUP & HOLD
- 4 Timing Paths
- 5 Introduction to SDC Constraints
- 6 Analyzing a Timing Report

Table of Contents

- 1 Introduction
- 2 Timing arcs and unateness:
- 3 TRANSMISSION GATE, D-LATCH, DFF, SETUP & HOLD
- 4 Timing Paths
- 5 Introduction to SDC Constraints
- 6 Analyzing a Timing Report

What is Static Timing Analysis (STA)?

- **Static Timing Analysis (STA)** is a method for determining if a circuit meets timing constraints without having to simulate.

Static timing analysis:

- **Verifies Timing**
 - verifies timing between synchronous clocks
 - does not verify functionality

What is Static Timing Analysis (STA)?

- **Static Timing Analysis (STA)** is a method for determining if a circuit meets timing constraints without having to simulate.

Static timing analysis:

- **Verifies Timing**
 - verifies timing between synchronous clocks
 - does not verify functionality
- **Is Exhaustive**
 - uses formal, mathematical techniques instead of vectors
 - does not use dynamic logic simulation

What is Static Timing Analysis (STA)?

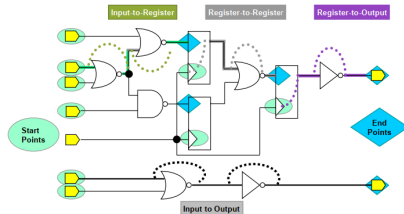
- **Static Timing Analysis (STA)** is a method for determining if a circuit meets timing constraints without having to simulate.

Static timing analysis:

- **Verifies Timing**
 - verifies timing between synchronous clocks
 - does not verify functionality
- **Is Exhaustive**
 - uses formal, mathematical techniques instead of vectors
 - does not use dynamic logic simulation
- **Is Fast**
 - significantly faster than gate level simulation
 - orders of magnitude Faster than Tx level (spice) simulation

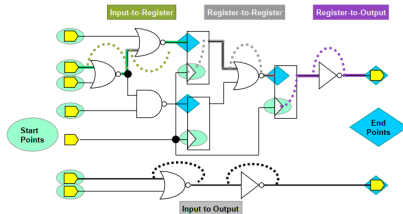
STA is Path-Based

- STA identifies timing paths within design for analysis

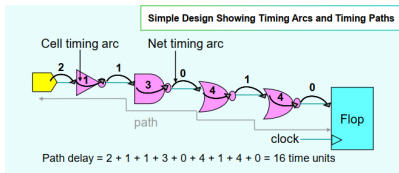


STA is Path-Based

- STA identifies timing paths within design for analysis



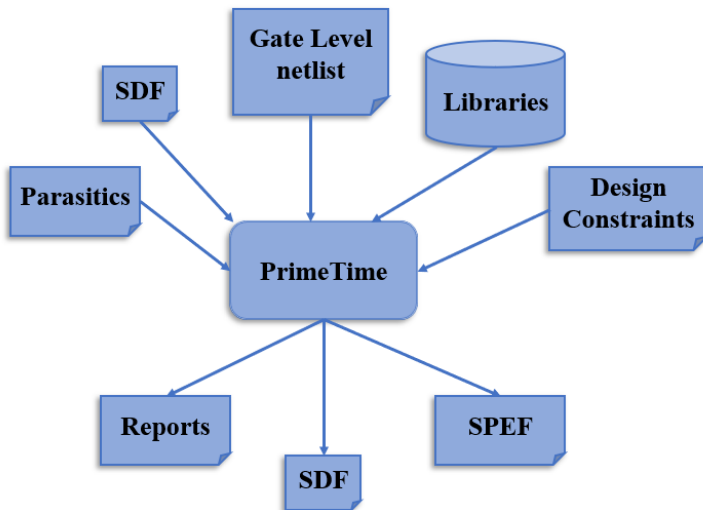
- Timing arcs within each path are calculated



STA is Constraint Driven

- STA in PT is constraint driven
- PT does not report a path, by default, that is not constrained for timing
- Incomplete or inaccurate constraints will lead to incorrect analysis and wasted runtime

PrimeTime Inputs and Outputs



Timing Analysis

Table of Contents

- 1 Introduction
- 2 Timing arcs and unateness:
- 3 TRANSMISSION GATE, D-LATCH, DFF, SETUP & HOLD
- 4 Timing Paths
- 5 Introduction to SDC Constraints
- 6 Analyzing a Timing Report

Timing arcs

Timing arcs

- means a path from each input to each output of the cell.

Timing arcs

Timing arcs

- means a path from each input to each output of the cell.
- Combinational Cells has Timing Arcs from each Input to each Output of the cell

Timing arcs

Timing arcs

- means a path from each input to each output of the cell.
- Combinational Cells has Timing Arcs from each Input to each Output of the cell
- Flip-flops have Timing Arcs from the Clock Input pin to Data Output Q pin (Propagation delay/ Delay Arc) and from Clock Input pin to Data Input D pin (setup, hold checks/ Constraint Arc)

Timing arcs

Timing arcs

- means a path from each input to each output of the cell.
- Combinational Cells has Timing Arcs from each Input to each Output of the cell
- Flip-flops have Timing Arcs from the Clock Input pin to Data Output Q pin (Propagation delay/ Delay Arc) and from Clock Input pin to Data Input D pin (setup, hold checks/ Constraint Arc)
- Latches have 2 timing arcs:
 - Clock pin to Output Q pin, when D is stable
 - Data D pin to Output Q pin when D changes (Latch is transparent)

Timing Unate

Unateness

- How Output changes for different types of transitions on Input

Timing Unate

Unateness

- How Output changes for different types of transitions on Input
- **Positive Unate** if Output Transition is same as Input Transition

Timing Unate

Unateness

- How Output changes for different types of transitions on Input
- **Positive Unate** if Output Transition is same as Input Transition
- **Negative Unate** if Output Transition opposite to Input Transition

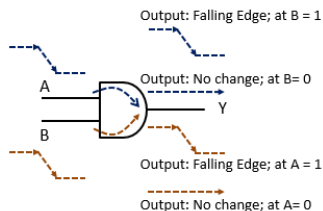
Timing Unate

Unateness

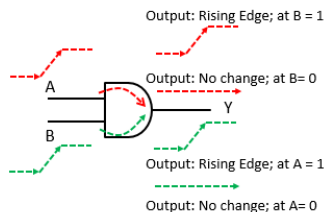
- How Output changes for different types of transitions on Input
- **Positive Unate** if Output Transition is same as Input Transition
- **Negative Unate** if Output Transition opposite to Input Transition
- **Non-Unate** if the Output Transition cannot be determined solely from the direction of change of an Input. It also depends upon the state of the other Inputs

Positive unate

- If a rising transition on the input gives the output to rise and falling transition on the input gives the output to fall i.e. there is no change in transition of input and output then that timing arc is called positive unate.



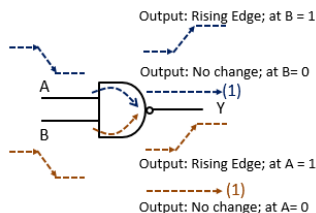
Timing Arc AND falling edge



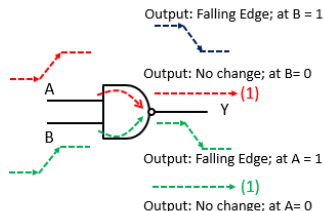
Timing Arc AND rising edge

Negative unate

- If a rising transition on the input gives the output to fall and falling transition on the input gives the output to rise i.e. there is a change in transition of input and output then that timing arc is called negative unate.



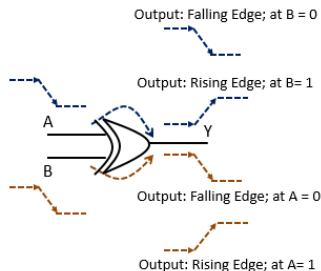
Timing Arc NAND falling edge



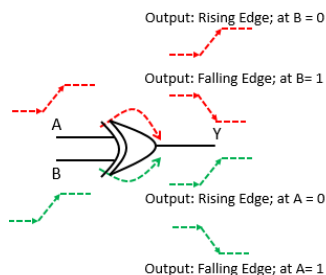
Timing Arc NAND rising edge

Non-unate

- The output transition cannot be determined by not only the direction of an input but also depends on the state of the other inputs.



Timing Arc XOR falling edge



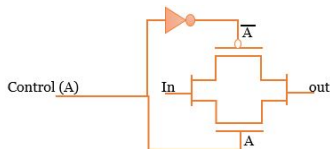
Timing Arc XOR rising edge

Table of Contents

- 1 Introduction
- 2 Timing arcs and unateness:
- 3 TRANSMISSION GATE, D-LATCH, DFF, SETUP & HOLD**
- 4 Timing Paths
- 5 Introduction to SDC Constraints
- 6 Analyzing a Timing Report

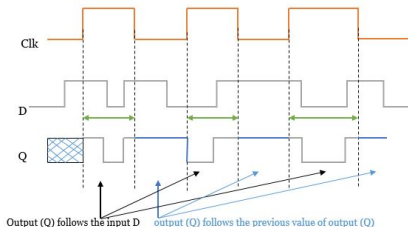
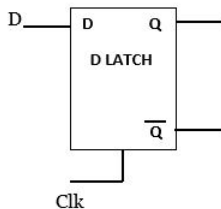
Transmission Gate

- The transmission gate is consists of a parallel connection of PMOS & NMOS.
- Two gate voltage of PMOS and NMOS are the complement of each other
- The effective resistance of the transmission gate is almost constant because of the parallel connection of PMOS and NMOS.
- It is a bidirectional circuit and it carries the current in either direction.



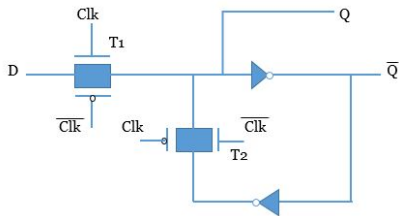
D Latch

- The **latch is a level-sensitive** device and it is transparent when the clock is high if it is a positive level-sensitive latch and when the clock is low it is called negative level-sensitive latch.
- In latch the output (Q) is dependent only on the level of the clock (Clk). In this latch D control the output (Q)

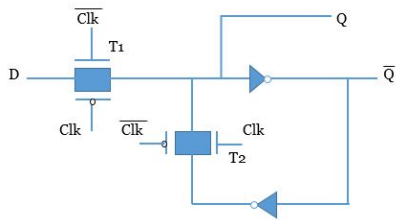


Positive/Negative D latch using transmission Gate

It consists of two transmission gates and two inverters.



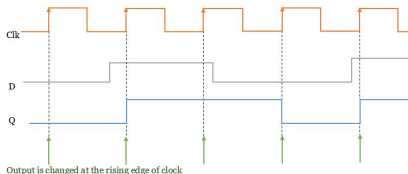
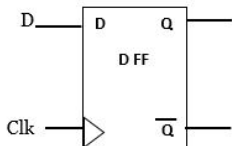
Positive D-Latch



Negative D-Latch

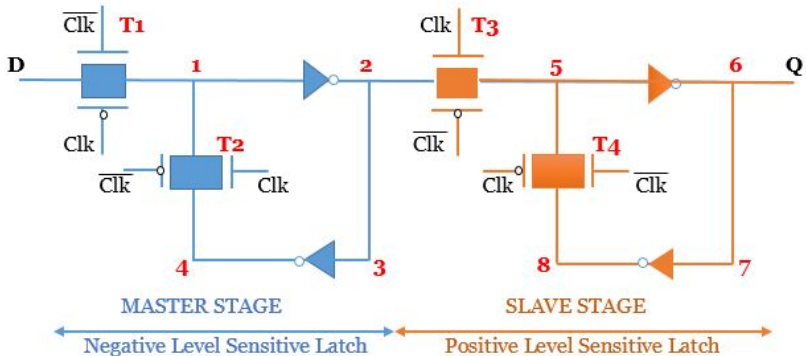
D Flip flop:

- A D flip flop is an **edge-triggered device** which means the output (Q) follows the input (D) only at the active edge (for positive rising edge) of the clock (for the positive edge-triggered) and retain the same value until the next rising edge i.e. output does not change between two rising edges, it should be changed only at the rising edge.



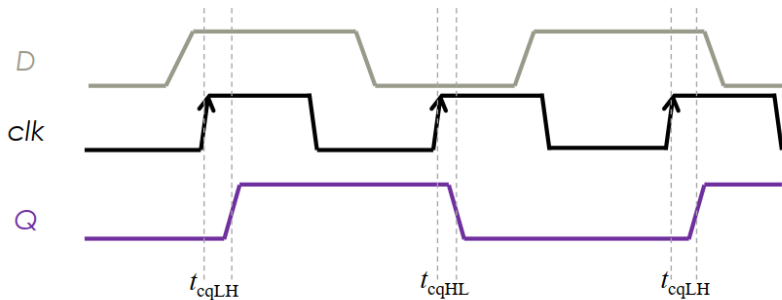
D Flip flop using a transmission gate

- It is a combination of negative level-sensitive latch and positive level-sensitive latch that giving an edge-sensitive device. Data is change only at the active edge of the clock.



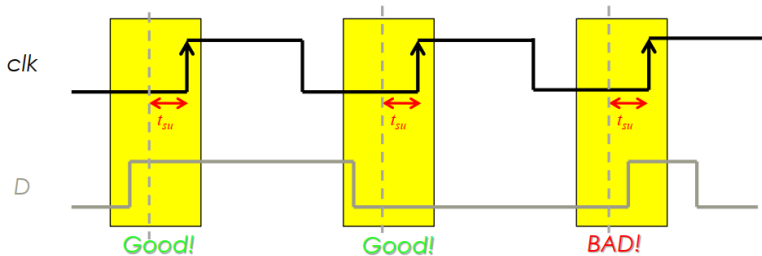
Timing Parameters, t_{cq}

- t_{cq} is the time from the clock edge until the data appears at the output
- The t_{cq} for rising and falling outputs is different



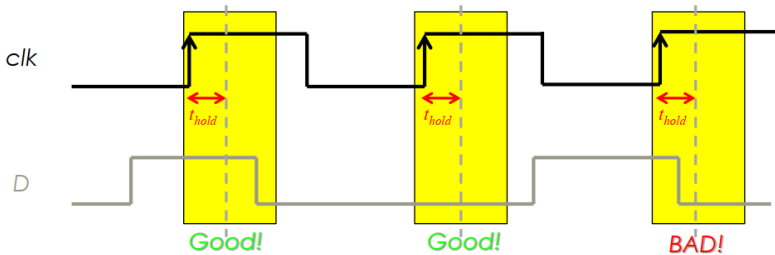
Timing Parameters, t_{setup}

- t_{setup} - Setup time is the time the data has to arrive **before the clock** to ensure correct sampling.



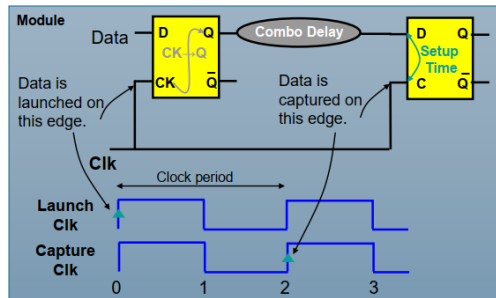
Timing Parameters, t_{hold}

- t_{hold} - Hold time is the time the data has to be stable **after the clock** to ensure correct sampling.



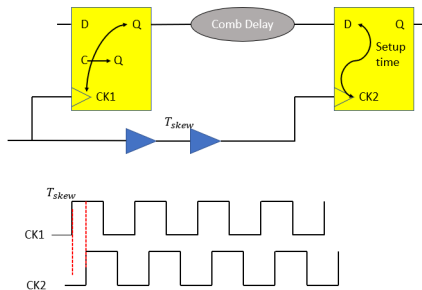
Understanding Launch and Capture Clock Edges

- **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



Setup (Max) Constraint

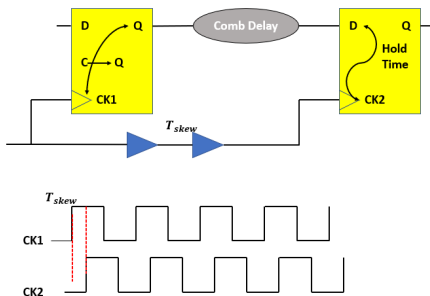
- Setup time is the duration of time that the synchronous input data must be stable before the triggering-edge of the clock.



$$T_{clk} + T_{skew} \geq T_{c2q} + T_{comb} + T_{setup}$$

Hold (Min) Constraint

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



$$T_{hold} + T_{skew} \leq T_{c2q} + T_{comb}$$

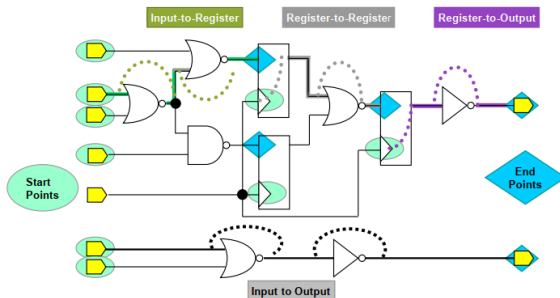
Table of Contents

- 1 Introduction
- 2 Timing arcs and unateness:
- 3 TRANSMISSION GATE, D-LATCH, DFF, SETUP & HOLD
- 4 Timing Paths**
- 5 Introduction to SDC Constraints
- 6 Analyzing a Timing Report

What Are Timing Paths?

Timing path

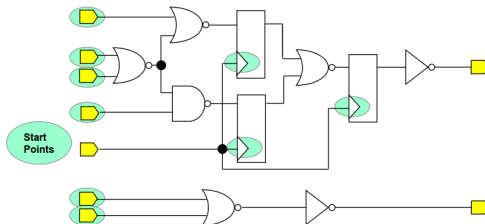
A timing path is a combination of all the timing arcs from a **start point** to an **end point**. You can break down timing paths into four simple types or categories.



Start Points

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

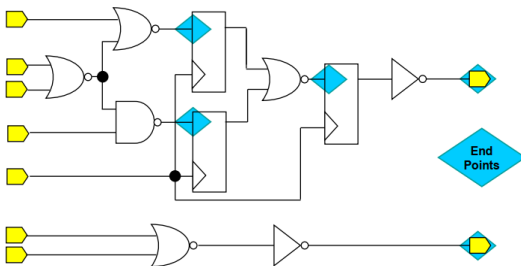
- The **input port** of a design (other than a clock port).
- The **clock pin** of a sequential cell.



End Points

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

- The **Output port** of a design .
- The **Data pin** of a sequential cell.



What Is Slack?

Slack

Required time is defined by the timing constraints like your clock period. Arrival time is when the signal actually arrives.

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

$\text{Slack} = \text{Required Time} - \text{Arrival Time (setup)}$

$\text{Slack} = \text{Arrival Time} - \text{Required Time (hold)}$

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

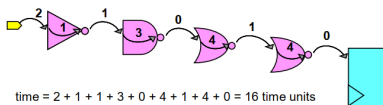
The calculating equation for slack is different for the different types of paths.

How Do You Time Timing Paths?

Delay

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

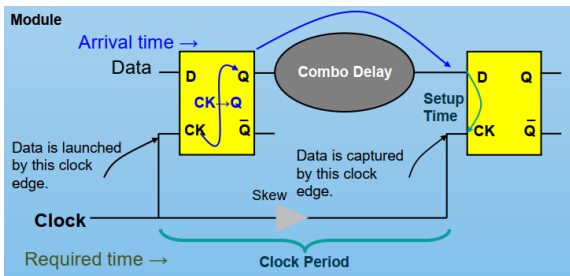
- Calculate all the cell delays.
 - Cell delay timing arcs are determined by the type of cell delay model used in the technology library
- Calculate all the net delays.
 - Net delay timing arcs are determined by the layout estimator or the wire-load model selected
- Add all the cell and net delays together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



Register-to-Register Setup Requirement

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

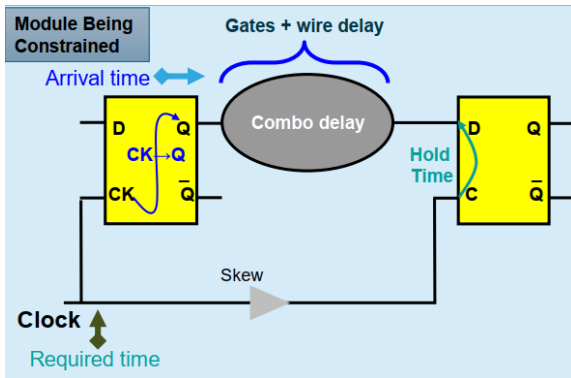
- $\text{Required time} = \text{Clock Period} + \text{Skew} - \text{Setup Time}$
- $\text{Arrival time} = t_{CK2Q} + \text{Combo delay}$
- $\text{Slack} = \text{Required time} - \text{Arrival time}$



Register-to-Register Hold Requirement

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

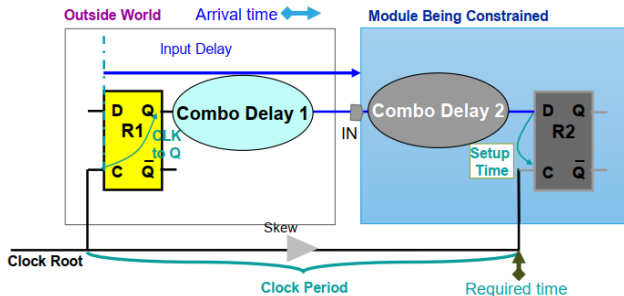
- Required time = Hold Time + Skew
- Arrival time = t_{CK2Q} + Combo delay
- Slack = Arrival time – Required time



Input-to-Register Setup Requirement

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

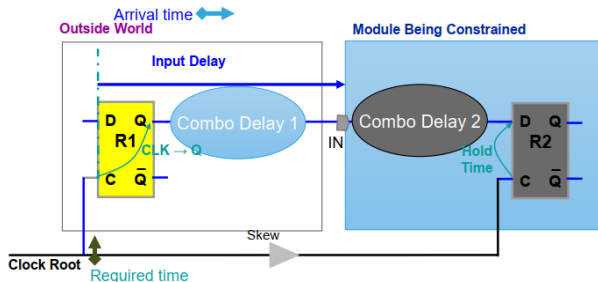
- Required time = Clock Period + Skew – Setup Time of R2
- Arrival time = t_{CK2Q} + Combo Delay 1 + Combo delay 2
 - Input_Delay = clk_to_q + combo1
- Slack = Required time – Arrival time



Input-to-Register Hold Requirement

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

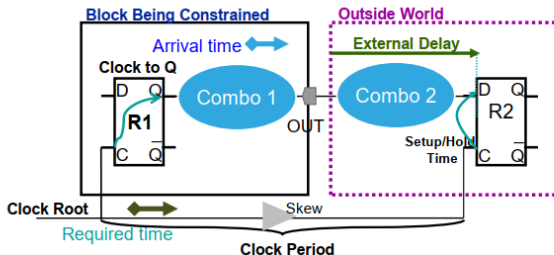
- Required time = Hold Time of R2 + Skew
- Arrival time = Input delay + Combo delay
 - Input_Delay = clk_to_q + combo1
- Slack = Arrival time – Required time



Register-to-Output Setup Requirement

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

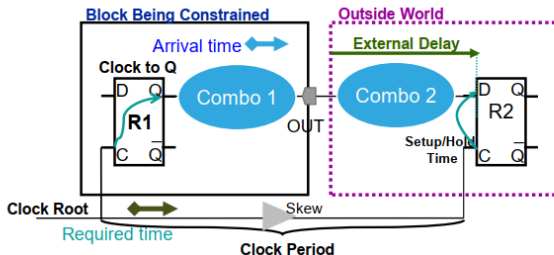
- $\text{Required time} = \text{Clock Period} + \text{Skew} - \text{Output_Delay}$
 - $\text{Output_Delay} = \text{combo_2_delay} + \text{setupR2}$
- $\text{Arrival time} = t_{CK2Q} + \text{Combo Delay 1}$
- $\text{Slack} = \text{Required time} - \text{Arrival time}$



Register-to-Output Hold Requirement

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

- Required time = Output Delay + Skew
 - $\text{output_delay2} = \text{combo_2_delay} - \text{holdR2}$
- Arrival time = t_{CK2Q} + Combo delay 1
- Slack = Arrival time – Required time



Input to Output Path Timing Requirement

- Combinational paths have no clock defined. Therefore, you need to use either a virtual clock or set max/min.
- If you define a virtual clock or a dummy clock, then slack is calculated as:
 - $\text{Slack} = \text{Clock Period} - \text{Input Delay} - \text{Output Delay} - \text{Combinational Delay}$
 - Remember that input delay and output delay are set with respect to a clock.
- You can also set max/min delays using the following syntax:
 - `set_max_delay`
 - `set_mix_delay`
- Then the slacks for the input to output paths are calculated as below.
 - Setup (late) slack = `max_delay - combo_delay`
 - Hold (early) slack = `min_delay - combo_delay`

Activity: Timing Paths

Here are the parameters for this activity:

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

What is the worst timing slack based on this illustration?

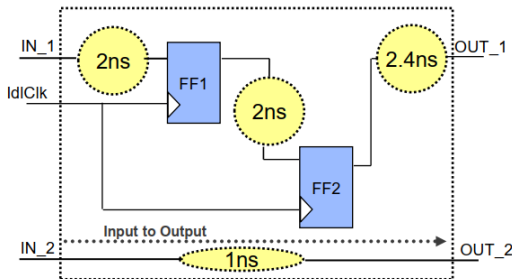


Table of Contents

- 1 Introduction
- 2 Timing arcs and unateness:
- 3 TRANSMISSION GATE, D-LATCH, DFF, SETUP & HOLD
- 4 Timing Paths
- 5 Introduction to SDC Constraints
- 6 Analyzing a Timing Report

What Are Constraints?

- Constraints provide specifications that the design must meet through optimization.
- Typical examples of constraints are:
 - 1 Clock constraints
 - 2 External constraints
 - 3 Power constraints
 - 4 Net Delay constraints
 - 5 Environmental constraints
 - 6 Design rules for manufacturing

Common SDC Constraints

1 Timing

- `create_clock`
- `create_generated_clock`
- `set_clock_latency`
- `set_clock_transition`
- `set_clock_uncertainty`
- `set_input_delay`
- `set_output_delay`

2 Exceptions

- `set_false_path`
- `set_multicycle_path`
- `set_max_delay`
- `set_mmin_delay`

3 Power

- `set_max_dynamic_power`

- `set_max_leakage_power`

4 Operating conditions

- `set_operating_conditions`

5 Wire-load models

- `set_wire_load_mode`
- `set_wire_load_model`

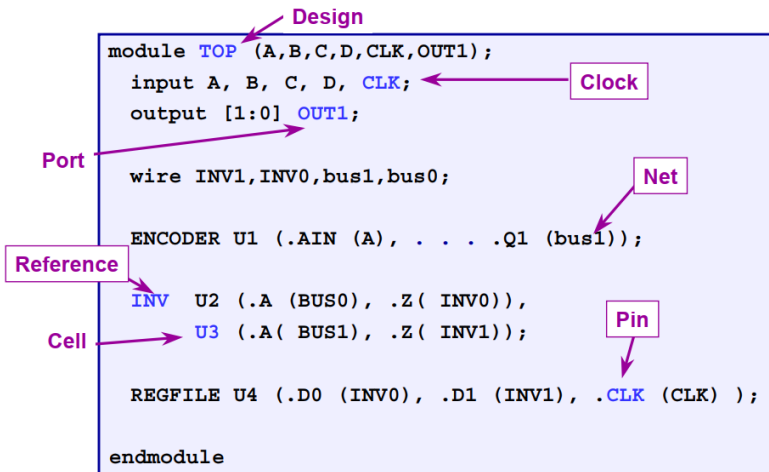
6 Environmental

- `set_driving_cell`
- `set_input_transition`
- `set_load`
- `set_fanout_load`

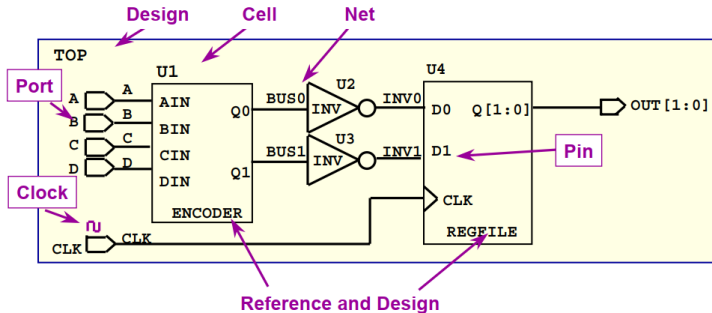
7 Design rules

- `set_max_capacitance`
- `set_max_fanout`
- `set_max_transition`

Design Objects



Design Objects



- **Designs:** {TOP, ENCODER, REGFILE}
- **References:** {ENCODER, REGFILE , INV}
- **Cells:** {U1, U2, U3, U4}

Design Objects [for all Synopsys tools]

- **Design:** A circuit description that performs some logical function.
- **Cell(Instants):** It is the instantiated name of the sub-design in the design.
- **Reference(Module):** The original design to which the cell or instance refers.
- **Port:** These are the primary inputs, outputs or IO's of the design.
- **Pin:** Pins are the input and output of cells (such as gates and flip-flops) within a design. The ports of a sub-design are pins within the parent design.
- **Net:** Nets are the wires that connect ports to pins and pins to each other.
- **Clock:** The port or pin that is identified as a clock source.

Timing Goals: Synchronous Designs

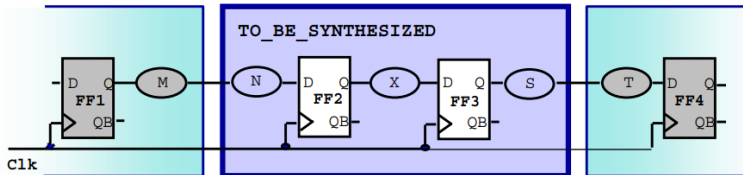
1 Synchronous Designs:

- Data arrives from a clocked device
- Data goes to a clocked device

2 Objective:

- Define the timing constraints for all paths within a design:
 - all input logic paths
 - the internal (register to register) paths, and
 - all output paths

Timing Goals: Synchronous Designs (cont)



Question

- What information must you provide to constrain all the register-to-register paths in your design?
- Does the duty cycle of your clock matter?

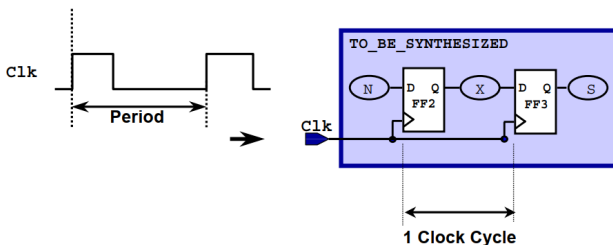
Defining a Clock

■ User MUST Define:

- Clock Source (port or pin)
- Clock Period

■ User may also define:

- Duty Cycle
- Clock Name
- Offset/Skew



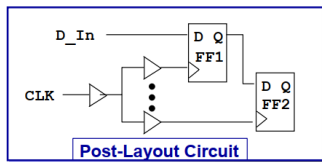
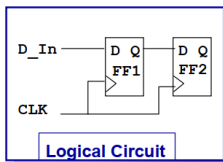
```
dc_shell>create_clock -name CLK -period 10 [get_ports Clk]
dc_shell>set_dont_touch_network [get_clocks Clk]
```

Default Clock Behavior

- Defining the clock in a single-clock design constrains all timing paths between registers for single-cycle, setup time
- By default the clock rises at 0ns and has a 50% duty cycle
- By default DC will not “buffer up” the clock network, even when connected to many clock/enable pins of flip-flops/latches
 - The clock network is treated as “ideal” - infinite drive capability
 - 1 Zero rise/fall transition times
 - 2 Zero skew
 - 3 Zero insertion delay or latency
 - Estimated skew, latency and transition times can, and should be modeled for a more accurate representation of clock behavior

Modeling Clock Trees

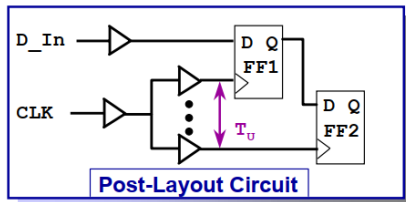
- Design Compiler is **NOT** used for synthesis of the clock tree
- Clock tree synthesis is usually done by **PnR tool**, based on physical placement data



Question

What design considerations need to be taken into account by the synthesis tool, prior to layout?

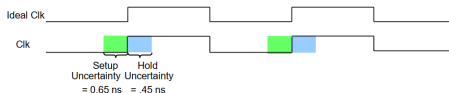
Modeling Uncertainty on Clock Edges



- To account for varying delays between the clock network branches (commonly called clock skew):

```
dc_shell>set_clock_uncertainty -setup 0.65 [get_clocks Clk]
```

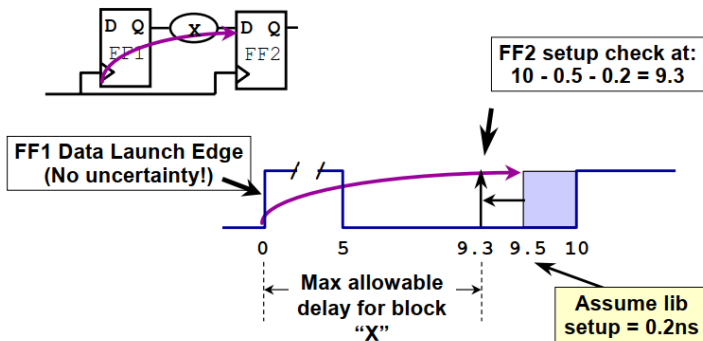
```
dc_shell>set_clock_uncertainty -hold 0.45 [get_clocks CLlk]
```



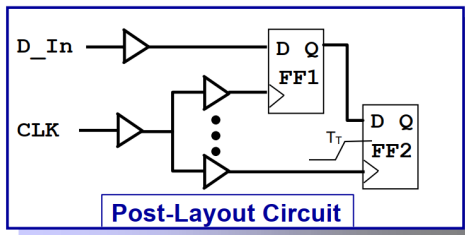
set_clock_uncertainty and Setup Timing

Example

```
dc_shell> create_clock -period 10 [get_ports CLK]
dc_shell> set_clock_uncertainty 0.5 [get_clocks CLK]
```



Setting Clock Transition



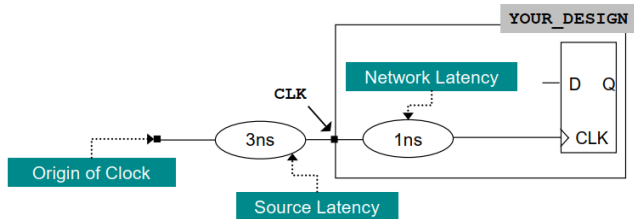
- Transition models the rise and fall times of the clock waveform at the register clock pins:

```
dc_shell>set_clock_transition 0.38 -rise [get_clocks Clk]
dc_shell>set_clock_transition 0.25 -fall [get_clocks Clk]
```

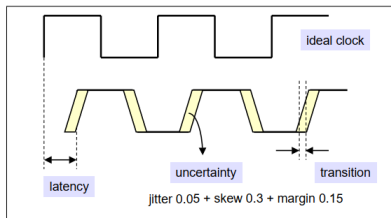
Modeling Latency or Insertion Delay

- **Network latency** models the average 'internal' delay from the create_clock port or pin to the register clock pins
- **Source latency** models the delay from the actual clock origin to the create_clock port or pin:

```
dc_shell>create_clock -period 10 [get_ports CLK]
dc_shell>set_clock_latency -source {max 3 [get_clocks CLK]}
dc_shell>set_clock_latency {max 1 [get_clocks CLK]}
```



Pre/Post Layout Clock



Synthesis Constraints

Example

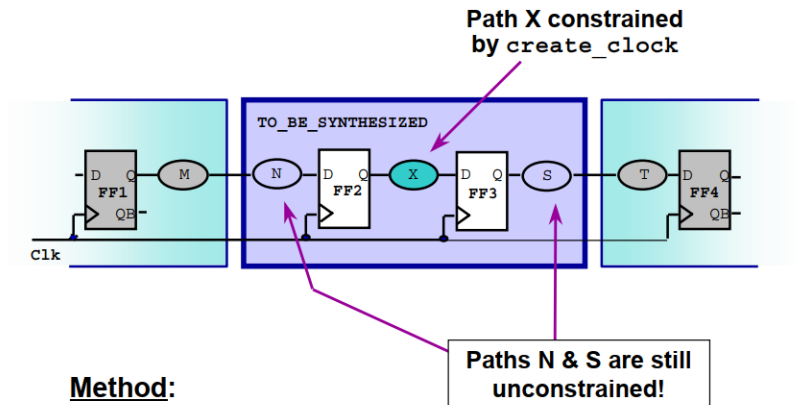
```
reset_design  
.....
```

Post-CTS STA Constraints

Example

```
reset_design  
.....
```

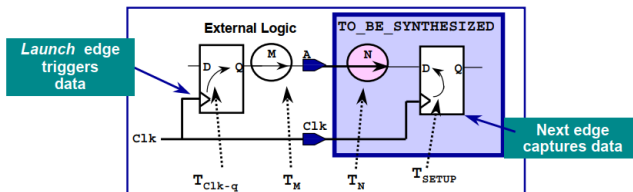
Timing Goals: Synchronous Designs, I/O



Method:

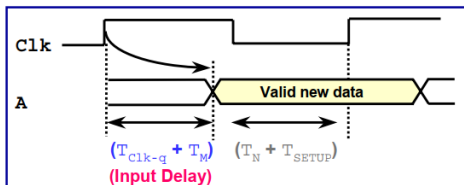
1. Define the clocks
2. Define the I/O timing *relative* to the clocks

Constraining Input Paths



Question

What information must you provide to constrain the input paths?



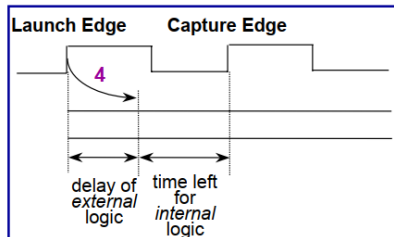
Constraining Input Paths in DC

```
dc_shell> set_input_delay -max 4 -clock Clk [get_ports A]
```

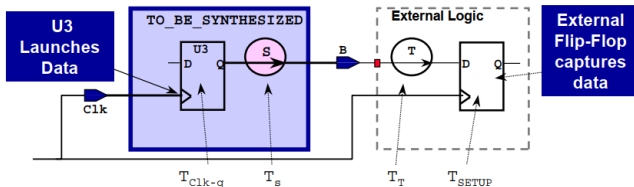
The **set_input_delay** command constrains input paths

You specify how much time is used by external logic...

DC calculates how much time is left for the internal logic

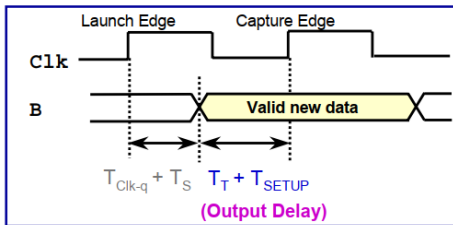


Constraining Input Paths



Question

What information must you provide to constrain the output paths?



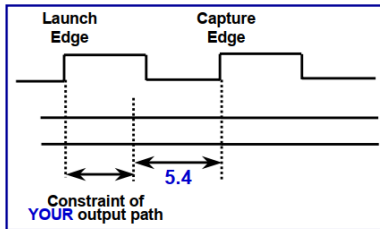
Constraining Output Paths in DC

```
dc_shell>set_output_delay -max 5.4 -clock Clk [get_ports B]
```

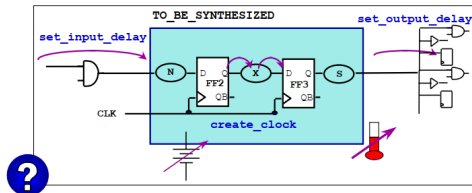
The **set_output_delay** command constrains input paths

You specify how much time is needed by external logic...

DC calculates how much time is left for internal logic



Factors Affecting Timing

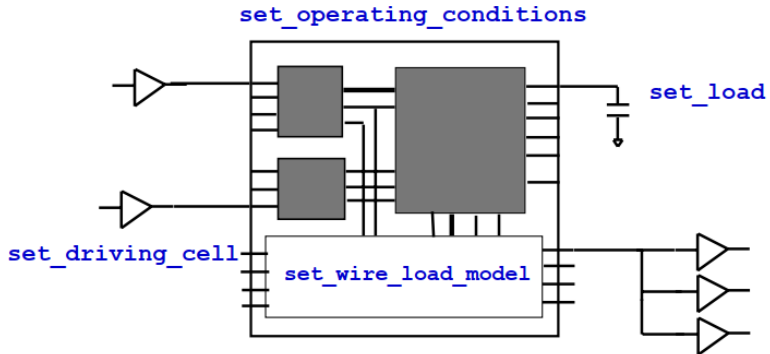


```
create_clock -period 2 [get_ports Clk]
set_input_delay -max 0.6 -clock Clk [get_ports A]
set_output_delay -max 0.8 -clock Clk [get_ports B]
```

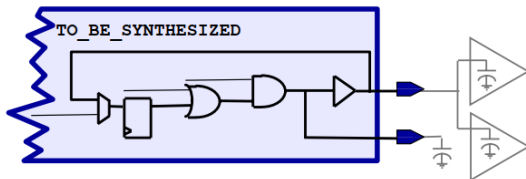
Note

The above constraints are required, but not sufficient for DC to accurately model and optimize all logic path delays. Need to take into account: Input drivers/transition times, Output loading, PVT corners and parasitic RCs

Describing Environmental Attributes



Effect of Output Capacitive Load



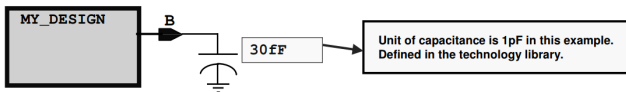
Note

- Capacitive loading on an output port affects the transition time, and thereby the cell delay, of the output driver.
- By default DC assumes zero capacitive loading on outputs. It is therefore important to accurately model capacitive loading on all outputs.

Modeling Output Capacitive Load: Example 1

Spec:

Chip-level: Maximum capacitive load on output port B = 30fF



```
set_load [expr 30.0/1000] [get_ports B]
```

Question

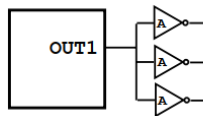
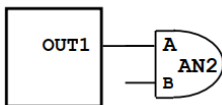
What if a specific capacitance value is not known, at ablock-level output port for example?

Modeling Output Capacitive Load: Example 2

Spec:

Block-level: Maximum load on output port B = 1

“AN2” gate load, or= 3 “inv1a0” gates

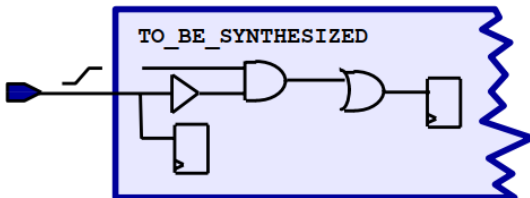


Answer

Use `load_oflib/cell/pinto` place the load of a gate from the technology library on the port:

```
set_load [load_ofmy_lib/AN2/A][get_ports OUT1]
set_load [expr[load_of my_lib/inv1a0/A] * 3] \
[get_ports OUT1]
```

Effect of Input Transition Time



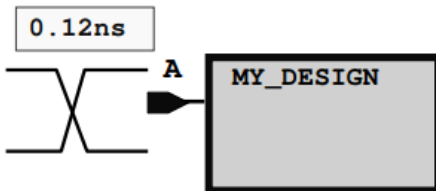
Note

- Rise and fall transition times on an input port affect the cell delay of the input gate.
- By default DC assumes zero transition times on inputs. It is therefore important to accurately model transition times on all inputs.

Modeling Input Transition: Example 1

Spec:

Chip-level: Maximum rise/fall input transition on input port A = 0.12ns



```
set_input_transition 0.12 [get_ports A]
```

Question

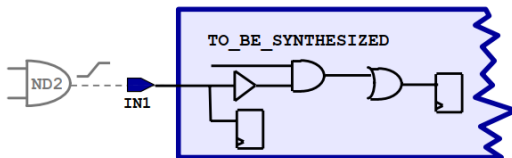
What if a specific transition time value is not known, at a block-level input port for example?

Modeling Input Transition: Example 2

Spec:

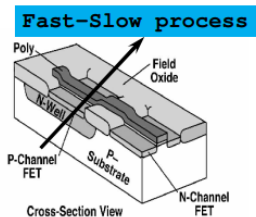
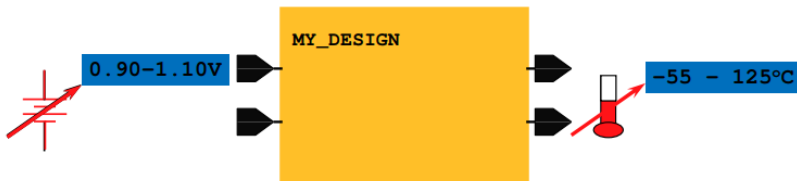
Block-level: Maximum load on output port B = 1

“AN2” gate load, or= 3 “inv1a0” gates



```
set_driving_cell -lib_cell and2a0 [get_ports IN1]
```

Modeling PVT Effects



Variations in Cell Delays

Library cells are usually characterized using “nominal” voltage and temperature:

nominal

```
nom_process : 1.0;
```

```
nom_temperature : 25.0;
```

```
nom_voltage : 1.8;
```

Question

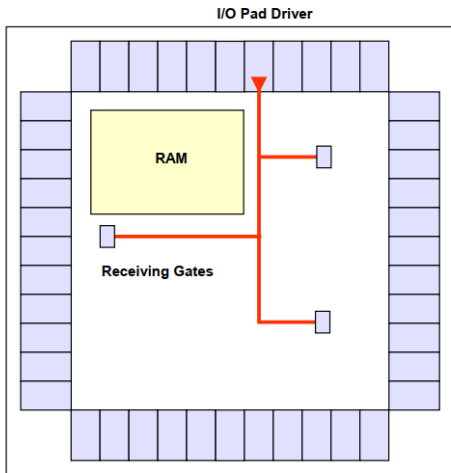
What if the circuit is to operate at a voltage and/or temperature OTHER than nominal?

Operating Conditions

- 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.
- ASIC vendors might deliver multiple technology libraries, defining:
 - best-case and worst-case operating conditions
 - optimistic and pessimistic WLM
 - minimum and maximum timing delay

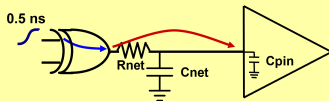
```
set_min_library ssc_core_slow \  
-min_version ssc_core_fast  
set_operating_conditions -max slow_125_1.62 \  
-min fast_0_1.98
```

Net delays



**Prior to layout, how
can the RC delay of
nets be estimated?**

Path Delays are Based on Cell + Net Delays



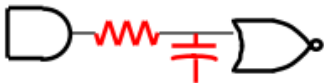
$$\text{Cell Delay} = f(\text{Input Transition Time}, C_{net} + C_{pin})$$

$$\text{Net Delay} = f(R_{net}, C_{net} + C_{pin})$$

Cell and net delays are both a function of parasitic RCs

Modeling Net RCs with Wire Load Models

- A wire load model is an **estimate** of a net's RC parasitics based on the net's fanout:
 - Models for various design sizes are supplied by your vendor
 - R/C values are average estimates based on data extracted from similar designs which were fabricated using this process



Wire Load Model Examples

What does this mean???

Extrapolation
slope

20 2946.37

Ask your library provider how to select the appropriate model!!

Specifying Wire Loads

- Manual model selection

```
set_wire_load_model -name 8000000
```

- Automatic model selection selects an appropriate wireload model during compile:

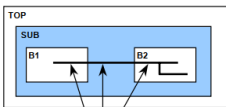
```
dc_shell> report_lib 90nm_com
```

Selection		Wire load name
min area	max area	
0.00	43478.00	140000
43478.00	86956.00	280000
86956.00	173913.00	560000
173913.00	347826.00	1000000
...		

Wireload Model Mode

- Specifies wire load model to use for nets that cross hierarchical boundaries.

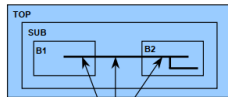
`mode = enclosed`



WLM_SUB

less pessimistic mode

`mode = top`



WLM_TOP

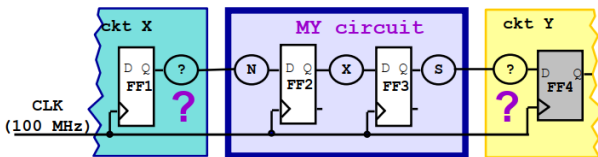
Example

```
set_wire_load_mode enclosed
```

Time Budgeting

Question

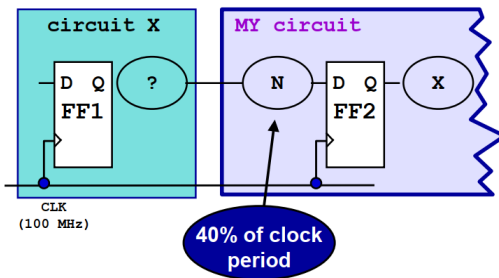
What if you don't know the delays on your inputs or the setup requirements of your outputs?



Answer

Create a Time Budget !

Time Budgeting (cont)

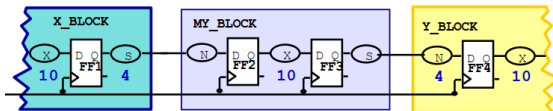


Note

Better to budget conservatively than to compile with paths unconstrained!

Time Budgeting Example

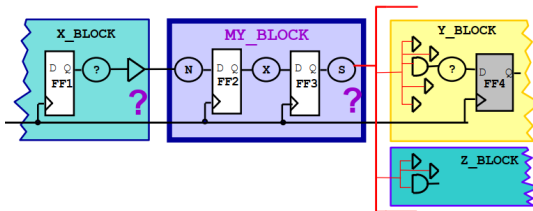
```
create_clock -period 10 [get_ports CLK]
set_dont_touch_network [get_clocks CLK]
set_input_delay -max 6 -clock CLK [all_inputs]
remove_input_delay [get_ports CLK]
set_output_delay -max 6 -clock CLK [all_outputs]
```



Load Budgeting

Question

What if, prior to compiling, the cells driving your inputs, and the loads on your outputs are not known?



Answer

Create a Load Budget!

Load Budgeting (cont)

- Assume a weak cell driving the inputs, to be conservative
- Limit the input capacitance of each input port
- Estimate the number of other major blocks your outputs may have to drive

Question

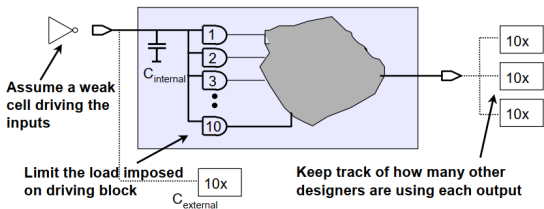
How do we limit the input capacitance of an input port?

A: Place restrictive design rules on our input ports

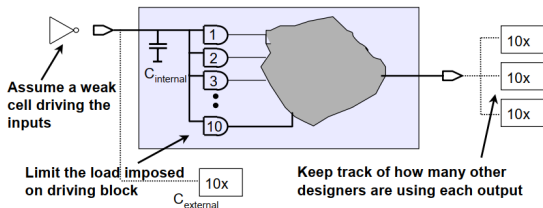
Load Budget Example

Example Specification:

- Inputs of any block shall present no more than the load of 10 “AND2” gates to their driving block
- Outputs of any blocks will only be allowed to connect to a maximum of 3 other blocks



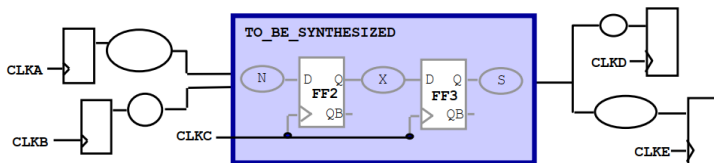
Load Budget Example (cont)



```

set_driving_cell -lib_cell inv1a0 [all_inputs]
remove_driving_cell [get_ports Clk]
set MAX_INPUT_LOAD [expr [load_of(tech_lib/and2a0/A) * 10]]
set_max_capacitance $MAX_INPUT_LOAD [all_inputs]
remove_attribute max_capacitance [get_ports Clk]
set_load [expr [$MAX_INPUT_LOAD * 3]] [all_outputs]
  
```

Asynchronous Multiple Clock Designs



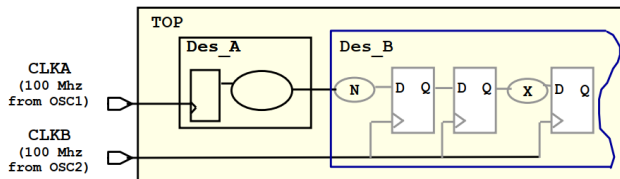
Question

What do you do if the design has asynchronous clock sources?

Synthesizing with Asynchronous Clocks

- It is your responsibility to account for the
 - Instantiate double-clocking, metastable-hard Flip-Flops
 - dual-port FIFO, etc
- Create clocks to constrain the paths within each clock domain
- You must also disable timing-based synthesis on any path which crosses an asynchronous clock boundary:
 - This will prevent DC from wasting time trying to get the asynchronous path to “meet timing”

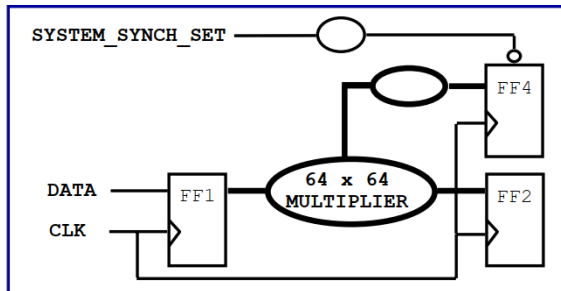
Example: Asynchronous Design Constraints



```
create_clock -period 2 [get_ports CLKA]
create_clock -period 2 [get_ports CLKB]
set_false_path -from [get_clocks CLKA] -to\
  [get_clocks CLKB]
set_false_path -from [get_clocks CLKB] -to\
  [get_clocks CLKA]
```

Multi-Cycle Behavior

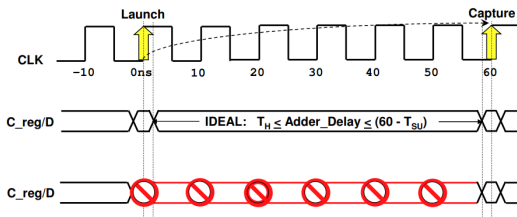
Situation: Not all paths operate at the target frequency of the design



- Choose one of the following options:
 - 1 Add pipeline stage(s) to divide the logic into single-cycle paths
 - 2 Ease off the single-cycle requirement: allow more clock cycles

Timing with Multi-cycle Constraints

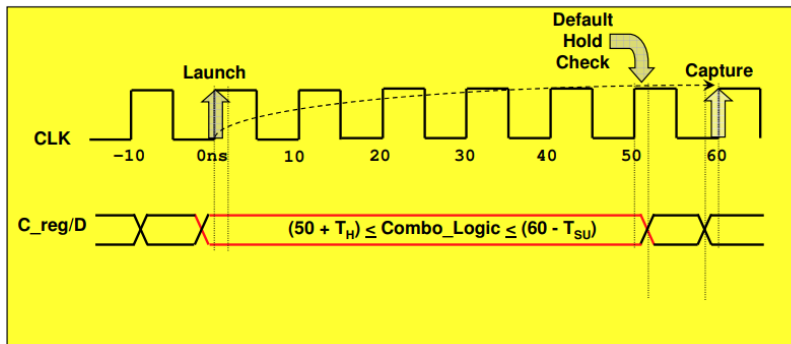
```
create_clock -period 10 [get_ports CLK]
set_multicycle_path {setup 6 -to FF4/D[*]}
```



DC assumes change could occur near any clock edge causing metastability!

Where does DC perform hold analysis?

Default Hold Check



Why is hold check performed at 50 ns?

Set the Proper Hold Constraint

```
set_multicycle_path -setup 6 -to FF4/D[*]  
set_multicycle_path -hold 5 -to FF4/D[*]
```

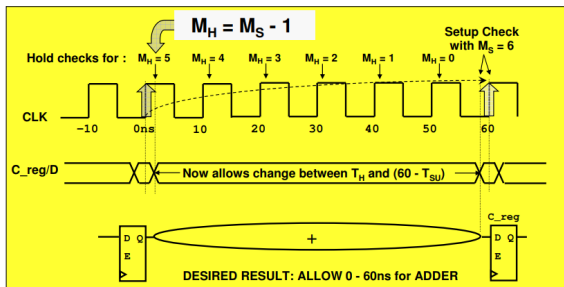


Table of Contents

- 1 Introduction
- 2 Timing arcs and unateness:
- 3 TRANSMISSION GATE, D-LATCH, DFF, SETUP & HOLD
- 4 Timing Paths
- 5 Introduction to SDC Constraints
- 6 Analyzing a Timing Report**

DesignCompiler Timing Reports

- Users will typically access DesignTime via the `report_timing` command
- The `report_timing` command
 - The design is broken down into individual timing paths
 - Each timing path is timed out **twice**;
 - 1 once for a rising edge input, and
 - 2 once with a falling edge input
 - The **critical path** (worst violator) for each clock group is found
 - A timing report for each clock group is echoed to the screen
- A DesignCompiler timing report has four major sections

Timing Report: Path Information Section

Report : timing

-path full

-delay max

-max_paths 1

Design : TT

Version: 2000.05

Date : Tue Aug 29 18:22:38 2000

Operating Conditions: **slow_125_1.62** Library: **ssc_core_slow**

Wire Load Model Mode: enclosed

Startpoint: **data1** (input port clocked by clk)

Endpoint: **u4** (rising edge-triggered flip-flop clocked by clk)

Path Group: **clk**

Path Type: **max**

Des/Clust/Port

Wire Load Model

Library

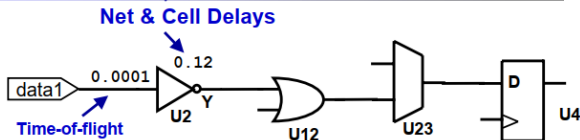
TT

5KGATES

ssc_core_slow

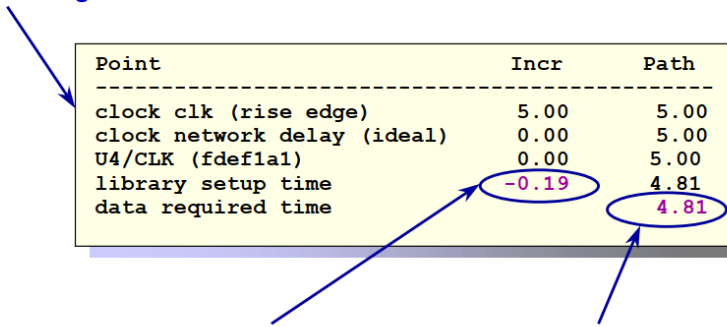
Timing Report: Path Delay Section

Point	Incr	Path	
-----	-----	-----	
clock clk (rise edge)	0.00	0.00	
clock network delay (ideal)	0.00	0.00	
input external delay	1.00	1.00	f
data1 (in)	0.00	1.00	f
u2/Y (inv1a1)	0.12	1.12	r
u12/Y (or2a1)	0.26	1.38	r
u23/Y (mx2d2)	0.23	1.61	f
u4/D (fdef1a1)	0.00	1.61	f
data arrival time		1.61	



Timing Report: Path Required Section

Clock Edge



Point	Incr	Path
-----	-----	-----
clock clk (rise edge)	5.00	5.00
clock network delay (ideal)	0.00	5.00
U4/CLK (fdef1a1)	0.00	5.00
library setup time	-0.19	4.81
data required time		4.81

**From the
Library**

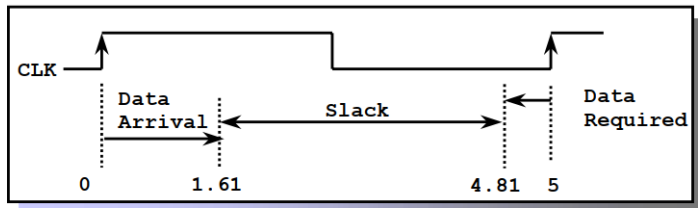
**Data must be valid
by this time**

Timing Report: Summary Section

data required time	4.81
data arrival time	-1.61
<hr/>	
slack (MET)	3.20

Either (MET) or (VIOLATED)

Timing margin (slack): negative indicates constraint violation



....

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ
وَمَا أُوتِيتُمْ مِنَ الْعِلْمِ إِلَّا قَلِيلًا