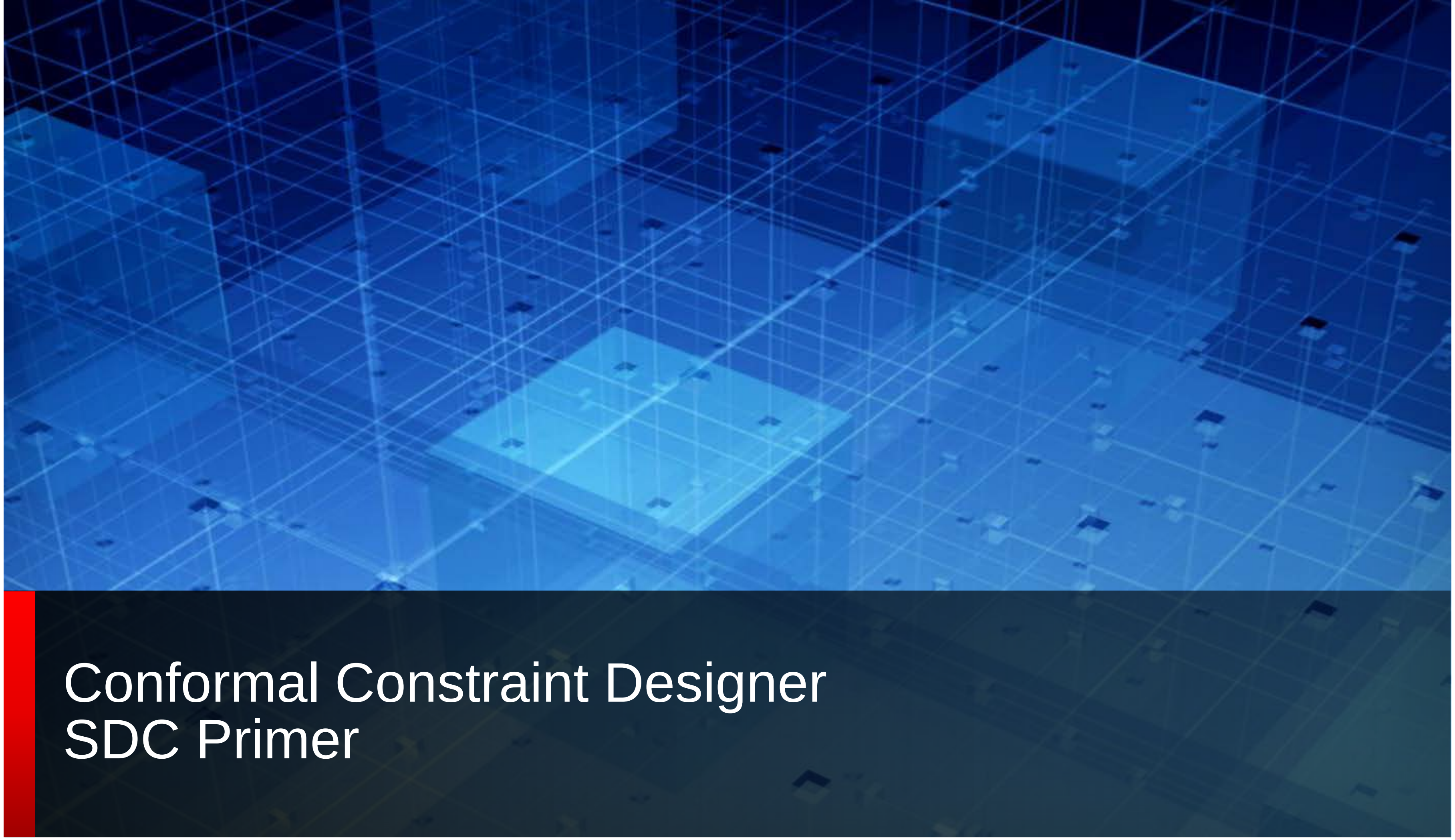


Conformal Constraint Designer
SDC Primer

Design Challenges

Front end designers, chip integrators, and the P&R team who receives timing constraints, are all impacted by the **quality, completeness and correctness** of timing constraints

Main challenges that compel teams to check timing constraints

1. Design complexity
   - Larger designs
   - Hierarchical design style
   - IP reuse
   - **This leads to exploding number of timing constraints**

Design Challenges

Main challenges that compel teams to check timing constraints (Cont'd)

2. Productivity through leverage of IP and design techniques
   - IP from other internal groups or from 3rd party vendors
   - Design now has more complex and unfamiliar constraints
   - **Multiple design teams make managing timing constraints more complex, greater potential to make mistakes**

3. Business demands
   - Tighter schedule; Do more with less
   - Solve problems earlier in design process, at RTL
   - **Teams face tremendous time-to-market pressure: Designs need to work first time**

SDC Constraints

❖ Major types of timing constraints
- Clock modeling
  - Clock period
  - Clock uncertainty
  - Clock latency
- IO Delays
- Timing exceptions
  - False path
  - Multicycle path
  - Min/Max Delay

Design Objects

❖Accessing different parts of the design

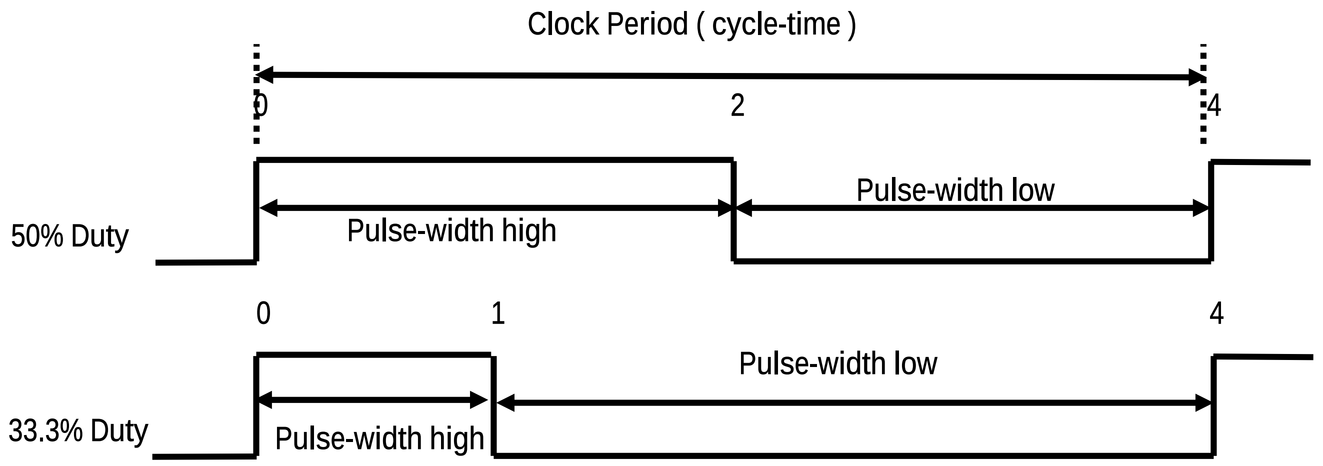
Design	current_design	A container for cells. A block.
Clock	get_clocks all_clocks	A clock in a design All clocks in a design
Port	get_ports all_inputs all_outputs	An entry point to or exit point from a design All entry points to a design All exit points from a design
Cell	get_cells	An instance of a design or library cell
Pin	get_pins	An instance of a design port or library cell pin
Net	get_nets	A connection between cell pins and design ports.
Lib_cell	get_lib_cells	A primitive logic element

Clock Modeling & Constraining

- Clock Modelling
  - Period, Duty Cycle, Source
  - Jitter, Uncertainty, and Skew
  - Source & Network insertion delay, aka Latency
  - create_generated_clock
  - set_clock_transition

Start with Clock Period and Duty Cycle

- ❖ Clocks are Periodic Waveforms
- ❖ Clock period (a.k.a cycle-time)
- ❖ Duty Cycle = ratio = pulse width high time / pulse width low time



cadence®

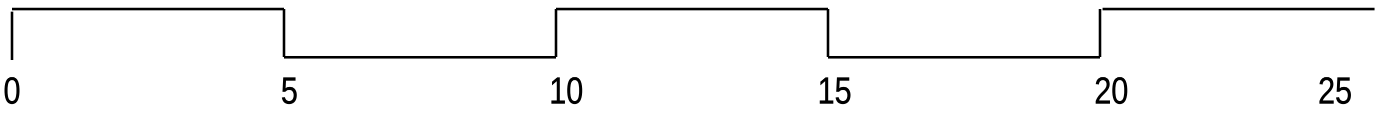
create_clock

- ❖ Defines a Clock Object:
- ❖ - period, source
- ❖ - name & waveform optional

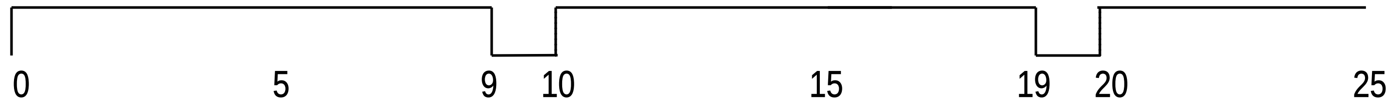
```
create_clock -period period_value
             [-name clock_name]
             [-waveform edge_list]
             [-add]
             [ source_objects]
```

source of signal

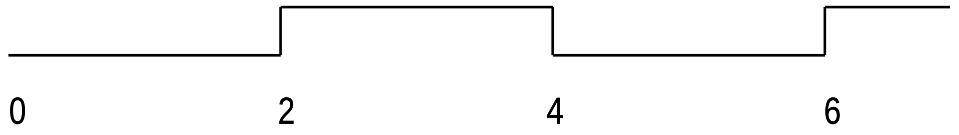
```
create_clock –name “PHI1” –period 10 –waveform {0.0 5.0} [get_ports clk]
```

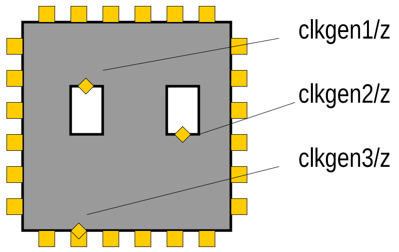


```
create_clock –name “clk10” –period 10 –waveform {0.0 9.0} [get_pins U1/clkout]
```



```
create_clock -name "clk" -period 4 -waveform {2.0 4.0} {clkg1/Z clkg2/Z clkg3/Z}
```





cadence®

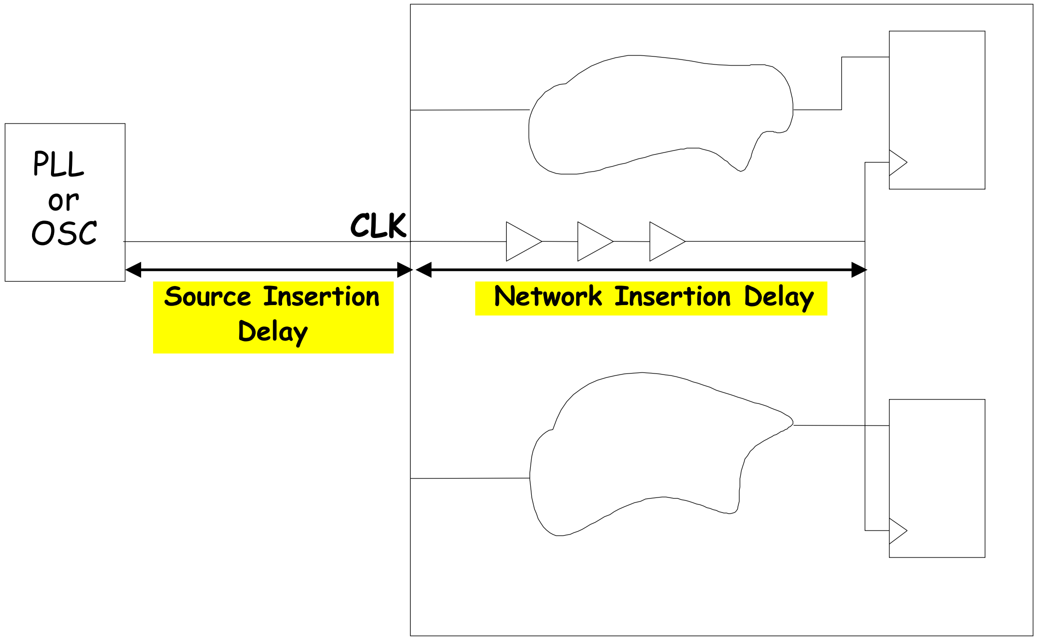
Clock Insertion Delay (a.k.a. clock latency)

- Delay from clock source to beginning of clock tree:

  Clock Source Insertion Delay = set_clock_latency –source

- Delay of clock distribution tree within design:

  Clock Network Insertion Delay (clock tree delay) =set_clock_latency

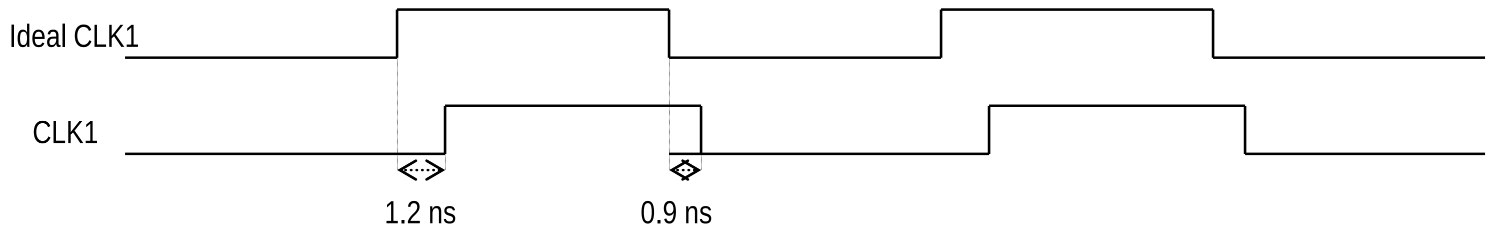


set_clock_latency

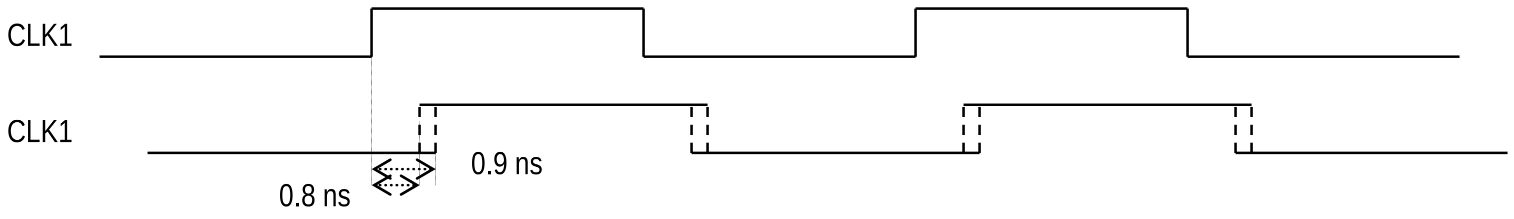
- Also called insertion delay, defines time it takes a clk signal to propagate from the clk definition point to a reg clk pin.
- Most tools assume ideal clocking, which means clocks have a specified network latency of zero by default

```
set_clock_latency [-rise] [-fall]
[-min] [-max] [-source] [-late]
[-early] delay
object_list
```

```
create_clock –period 10 –waveform {0 5} [get_ports CLK1]
set_clock_latency 1.2 -rise [get_clocks CLK1]
set_clock_latency 0.9 -fall [get_clocks CLK1]
```



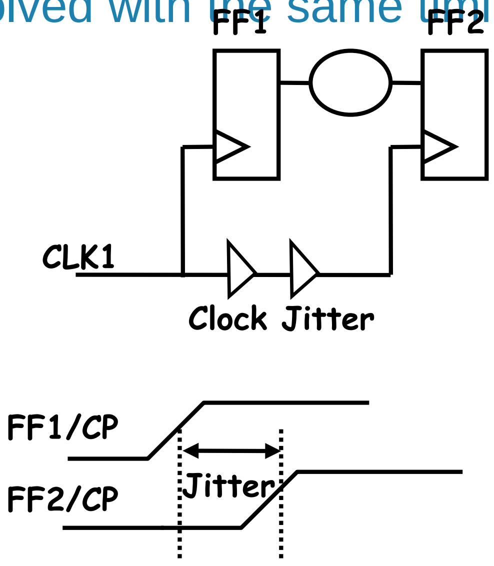
```
set_clock_latency 0.8 -source -early [get_clocks CLK1]
set_clock_latency 0.9 -source -late [get_clocks CLK1]
```

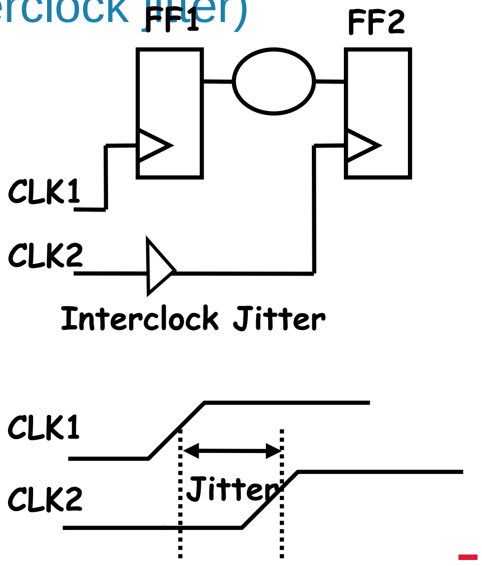


cadence®

Clock Uncertainty (a.k.a. Clock Jitter)

- From cycle to cycle, the period and duty-cycle of a clock can vary slightly due to clock generation and/or clock distribution circuitry (clock tree)
- i.e. the clock signal does NOT arrive simultaneously at all flops in a design
- Clock uncertainty can be used to describe an interval of time which represents the different arrival times of the clock for different branches of the same clock tree (a.k.a. clock jitter)
- Clock uncertainty can also be used to describe an interval of time which represents the different arrival times of different clocks that are involved with the same timing path (a.k.a. interclock jitter)



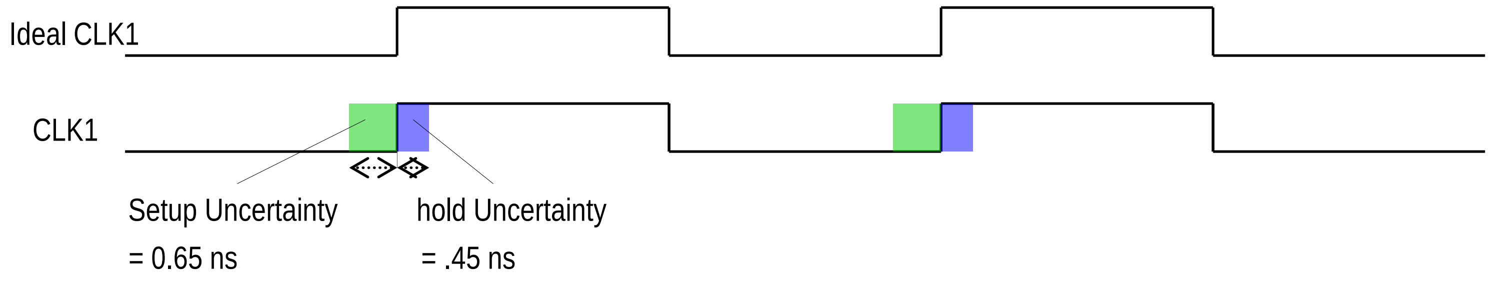


set_clock_uncertainty

- Specifies uncertainty (skew) of clock networks
  - Simple uncertainty specifies setup/hold uncertainties to all paths to the endpoint.
  - Inter-clock uncertainty specifies skew between various clock domains.
- Set uncertainty to worst skew expected to the endpoint or between the clock domains.
- You can also increase the value to account for additional margin for setup/hold time checks

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

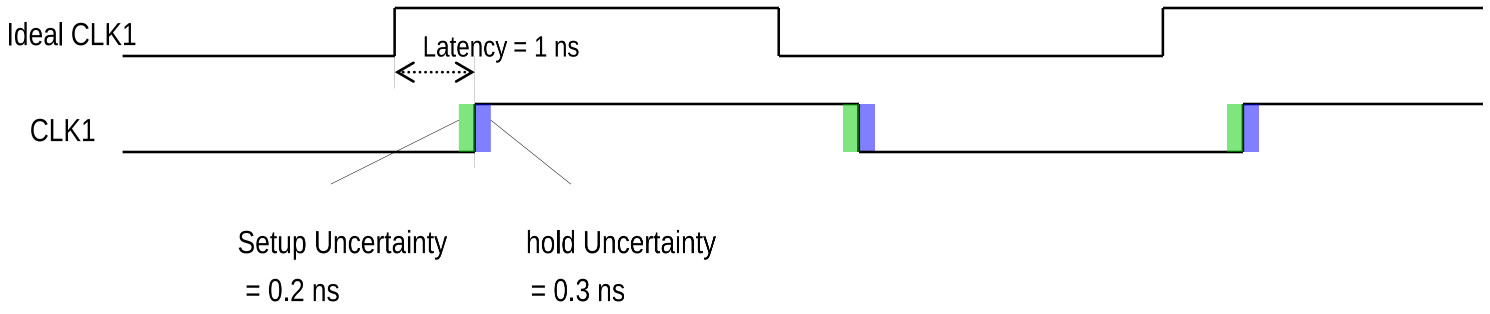
```
set_clock_uncertainty -setup 0.65 [get_clocks CLK]
set_clock_uncertainty -hold 0.45 [get_clocks CLK]
```



Define both uncertainty & latency for clocks

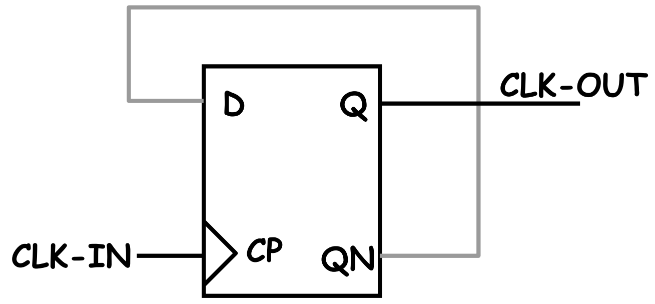
- Describe an ideal clock with a period of 10
  - waveform of {0 5}
  - Rise & Fall clock latency of 1ns
  - hold uncertainty of 0.3, and setup uncertainty of 0.2

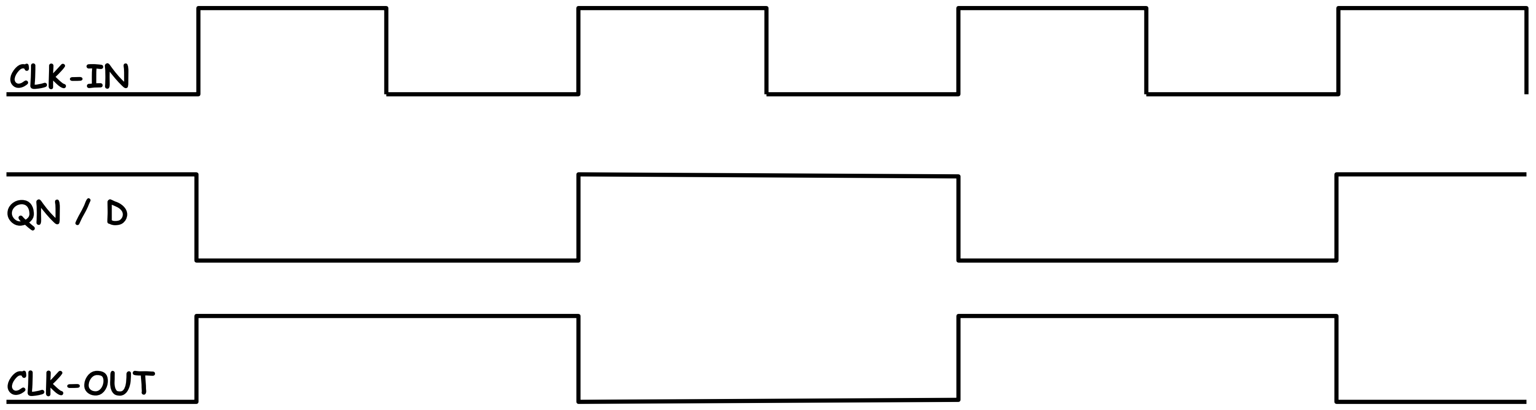
```
create_clock –name “CLK1” –period 10 –waveform {0.0 5.0} [get_ports clk]
set_clock_latency 1.0 -rise [get_clocks CLK1]
set_clock_latency 1.0 -fall [get_clocks CLK1]

set_clock_uncertainty -setup 0.2 [get_clocks CLK1]
set_clock_uncertainty -hold 0.3 [get_clocks CLK1]
```



How to Create One Clock From Another

- A divide-by clock in a design
- Can be modeled by either create_clock or create_generated_clock



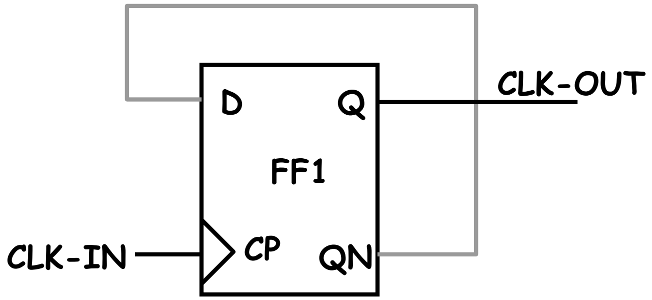


These waveforms are assuming there is no CP -> Q delay

Understanding Generated Clocks

- This is a capability to model synchronous clock dividers / multipliers relative to a master clock
- Thus, when if the master clock changes – so do all of the generated clocks
- The example clock can be modeled by the following command:

```
create_generated_clock -source FF1/CP -divide_by 2 -name CLK-OUT [get_pins FF1/Q]
```

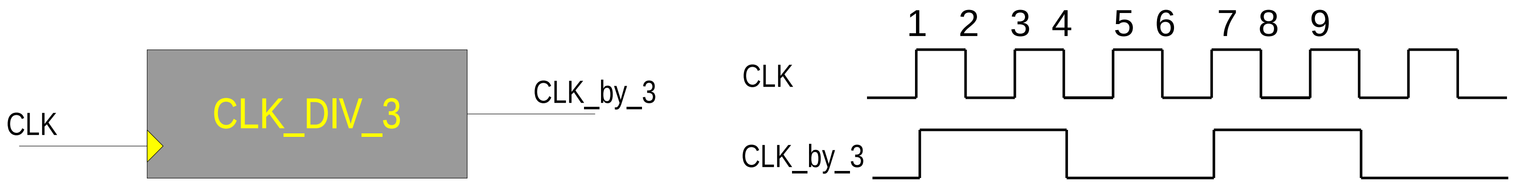


create_generated_clock

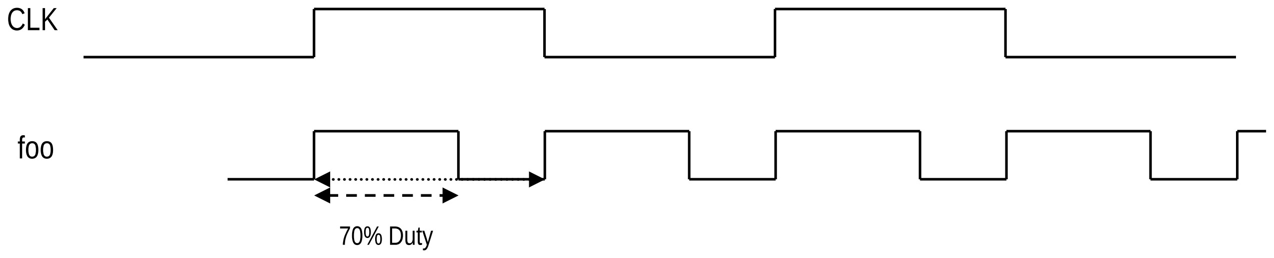
- Creates a clock object
- Apply to a pin of an instance, overrides create_clock
- Tied to source clock: whenever the source clock changes, the generated clock changes
- Use -divide_by or -multiply_by for multiples of 2
- Use –edges with 3 edges of the source clock to describe the waveform of the new generated clock

```
create_generated_clock  [-name clock_name]
-source master_pin [-edges edge_list]
[-divide_by factor] [-multiply_by factor]
[-duty_cycle percent] [-invert]
[-edge_shift shift_list] [-add]
[-master_clock clock] source_objects
```

```
create_generated_clock –edges {1 4 7} -source U1/CLK –name c3 [get_pins U1/CLK_by_3]
```



```
create_generated_clock -multiply_by 2 -duty_cycle 70 -source u1/CLK [get_pins u1/foo]
```



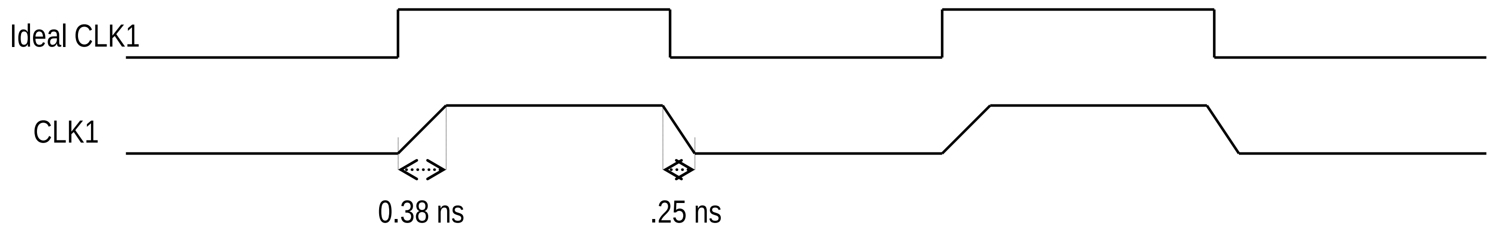
cadence®

set_clock_transition (aka. slew)

- ❖ Command overrides the transition times on flip-flop clk pin
- ❖ Useful for pre-layout when clk trees are incomplete
- ❖ Use only with ideal clocks.
- ❖ For propagated clocks, the calculated transition times are used.
- ❖ If a clock transition is not specified for an ideal clock, the transition time is calculated as it is for other pins in the design.

```
set_clock_transition
[-rise] [-fall]
[-min] [-max]
transition
clock_list
```

```
set_clock_transition 0.38 -rise [get_clocks CLK1]
set_clock_transition 0.25 -fall [get_clocks CLK1]
```



Clock Modeling Summary

- create_clock to define period, duty cycle, source of clock
- set_clock_latency to define insertion delays
- set_clock_uncertainty to define clock jitter (aka skew)
- set_clock_transition to define rise/fall transition times of clock edges
- create_generated_clock to define clocks relative to a master clock
- Use –name option – it simplifies use model of addressing clock objects

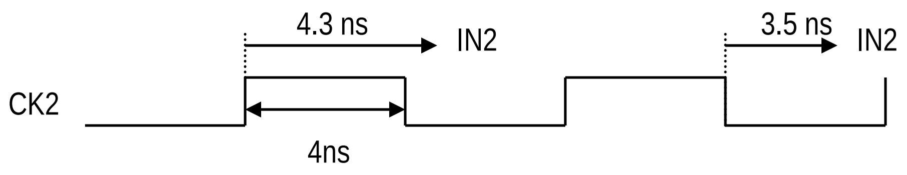
set_input_delay – 1/2

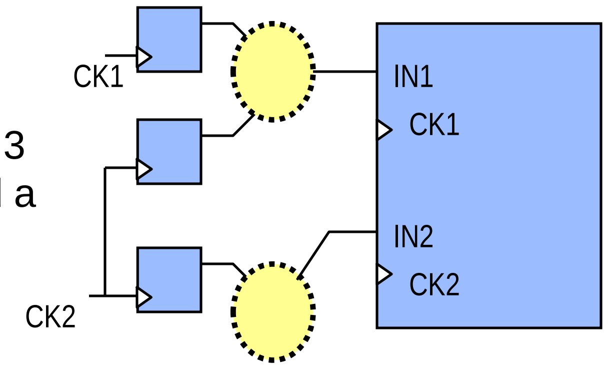
- Defines the arrival time relative to a clock.
- For bidirectional ports, one can specify the path delays for both input and output modes
- Use –add_delay to capture multi-clock delay relations

set_input_delay [-clock clock_name] [-clock _fall] [-level_sensitive] [-rise] [-fall] [-max] [-min] [-add_delay] [-network_latency _included] [-source_latency _included] delay_value port_pin_list

```
set_input_delay 4.3 -rise -clock CK2 {IN2}
set_input_delay 3.5 -fall -clock CK2 {IN2}
```

- Command assumes that a rising signal on IN2 can occur 4.3 time units after the rising edge of clock 8ns period CK2 and a falling signal has a delay of 3.5 units to reach IN2

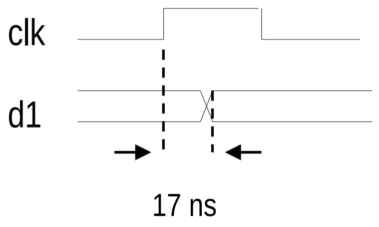


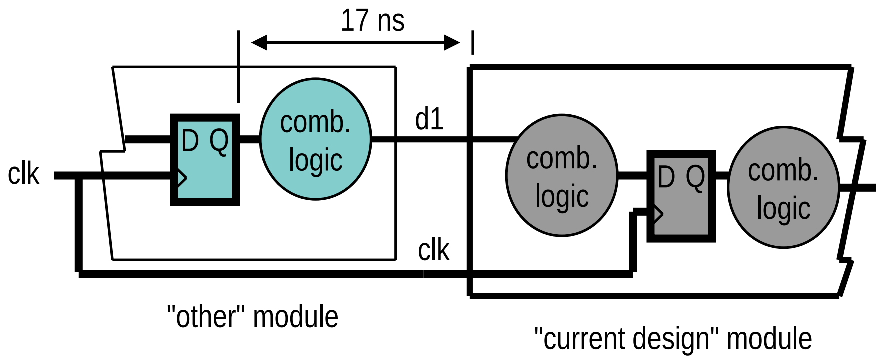


```
set_input_delay 2.7 -clock CK1 -add_delay { IN1 }
set_input_delay 4.2 -clock CK2 -add_delay { IN1 }
```

- Command specifies input delay for IN1 of 2.7ns relative to clock CK1 and 4.2ns relative to CK2

set_input_delay – 2/2





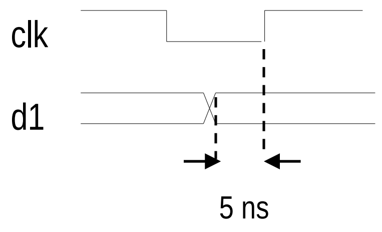
```
set_input_delay 17 -rise -clock clk { d1 }
```

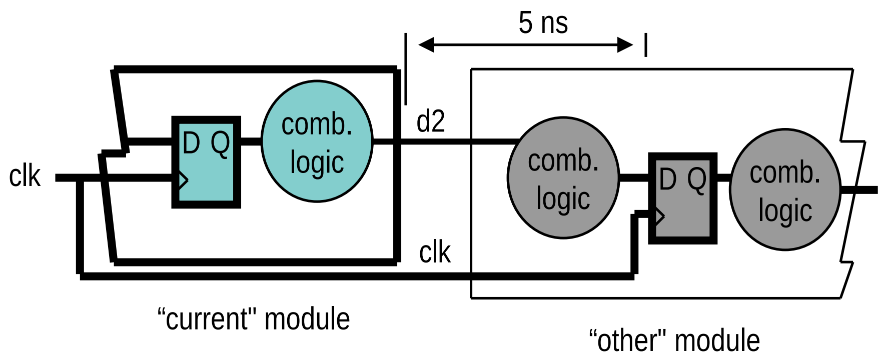
- Command specifies input delay of 17ns relative to clock clk
- “Environmental behavior” or “other module” timing is important for proper constraining for synthesis

set_output_delay

- The command sets output path delay values for the current design.
- The input and output delays characterize the operating environment of the current design
- Output ports have no output delay, unless specified.

```
set_output_delay [-clock clock_name] [-
clock_fall] [-level_sensitive]
[-rise] [-fall] [-max] [-min]
[-add_delay] [-network_latency_included]
[-source_latency_included]
delay_value port_pin_list
```



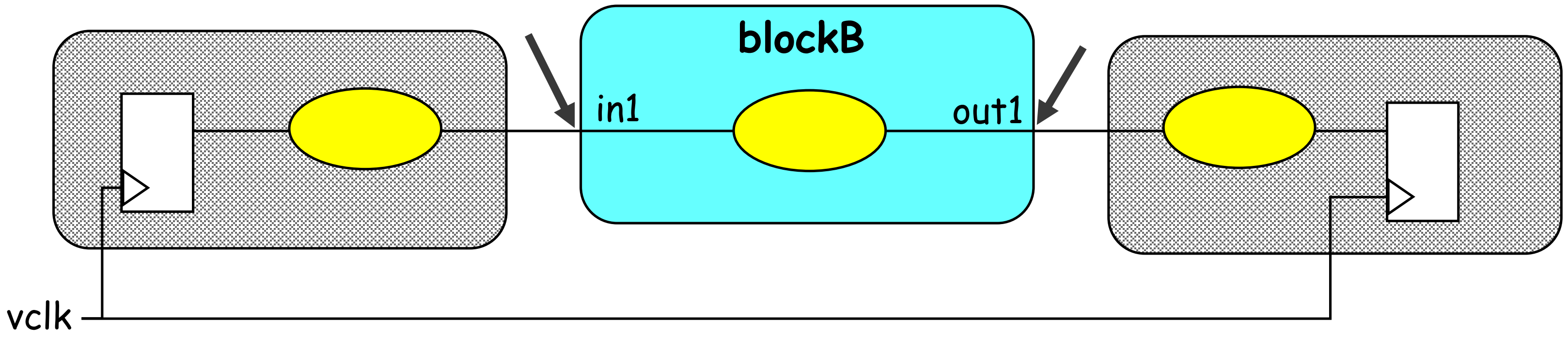


```
set_input_delay 5 -rise -clock clk { d1 }
```

- Command specifies expected output delay of 5 ns relative to clock clk
- "Environmental behavior" or "other module" timing is important for proper output constraints

Understanding Virtual Clock

- Virtual clocks are clocks that exist in memory but are not part of a design
  - Use it as a reference for specifying input and output delays relative to a clock
- This means there is no actual clock source in the design
  - I.e. Assume the block to be synthesized is "blockB"
    - The clock signal, "vclk", would be a virtual clock
    - The input delay and output delay would be relative to the virtual clock



Timing Exceptions

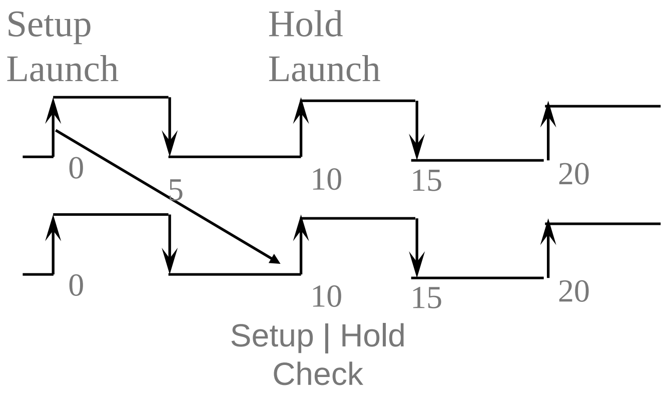
- Set_false_path
  - – Setting a false path removes the timing constraints on the path
  - – Example: clock to clock false path
- Set_min_delay/set_max_delay
  - – Use the set_max_delay and set_min_delay commands to override the default timing constraint values
  - – Example: a combinational path in a design
- Set_multicycle_path
  - – Use the set_multicycle_path command to specify the number of clock cycles required to propagate data from the start to the end of the path.
  - – Example: a path where the data come from a slow source and only get a new word every x clock cycles

Set_clock_groups is not a timing exception

- set_clock_groups
  - exclude paths from consideration that are launched by one clock and captured by another.
  - Achieve the same timing analysis as clock-to-clock false path constraints
  - Not considered a timing exception
    - Not reported by any exception reporting command
    - Not included in part of CCD EXC checks

Understanding Single Cycle Paths

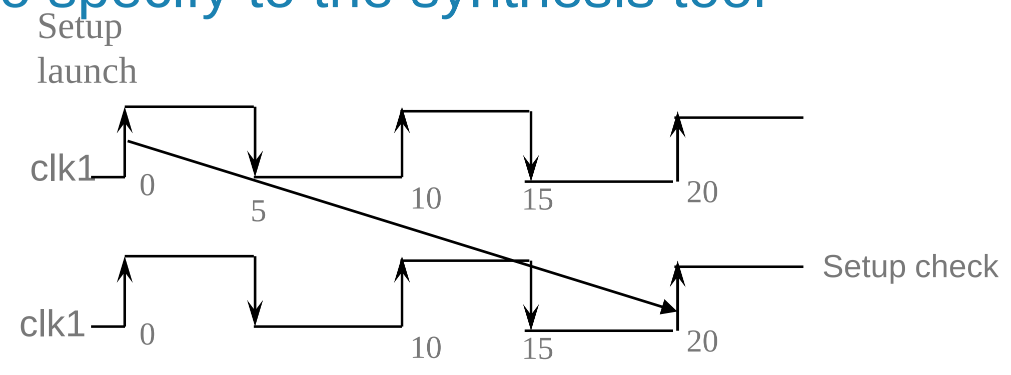
- By default, static timing tools assume all timing paths to be single cycle paths (assuming there is at least a clock defined)



- There could be exceptions defined to the above behavior:
  – Multicycle paths
  – False paths

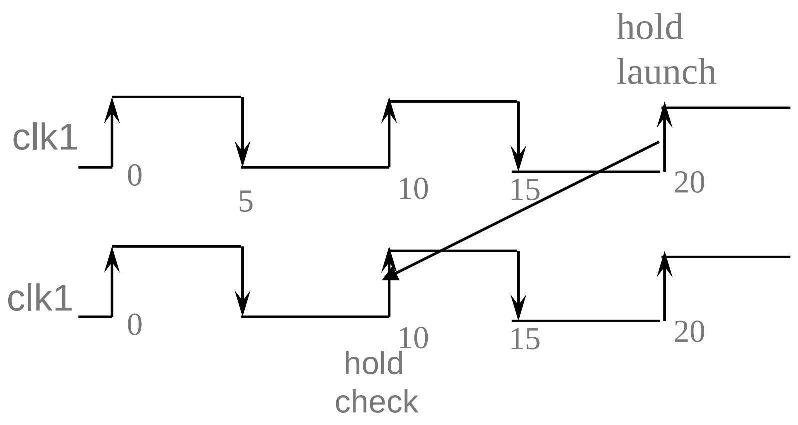
Understanding Multi-Cycle Paths

- ❖Those paths that require more than one clock period for execution
- ❖It's essential that multi-cycle paths in the design be identified both for synthesis and STA
- ❖Synthesis tool allows more than one cycle for the specified paths; more optimistic
- ❖Path through a multiplier takes longer than one clock cycle; designer needs to specify to the synthesis tool
- ❖Late example:



Understanding Multi-Cycle Paths (Cont.)

- Early Example



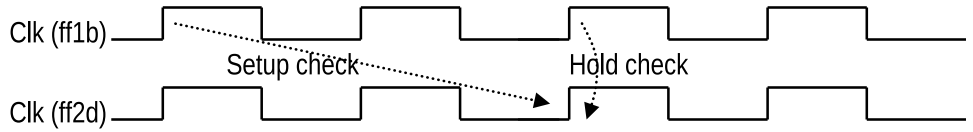
set_multicycle_path

- ❖ The command specifies that designated timing paths in the current design have no default single cycle setup or hold relations, but over multiple clock cycles
- ❖ Default hold check is at 0th cycle of the current clock edge – which is at the present edge

```
set_multicycle_path [-setup]
[-hold] [-rise] [-fall] [-start]
[-end] [-from from_list]
[-to to_list] [-through
through_list]
path_multiplier
```

```
set_multicycle_path –setup 2 -from { ff1b } -to { ff2d}
```

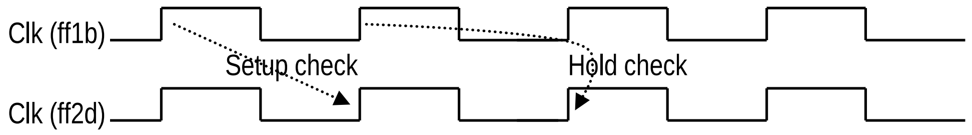
- The exception sets all paths between ff1b and ff2d to 2 cycle paths for setup.
- Hold is measured at the previous edge of the clock at ff2d.



Setup/Hold check wihout MCP

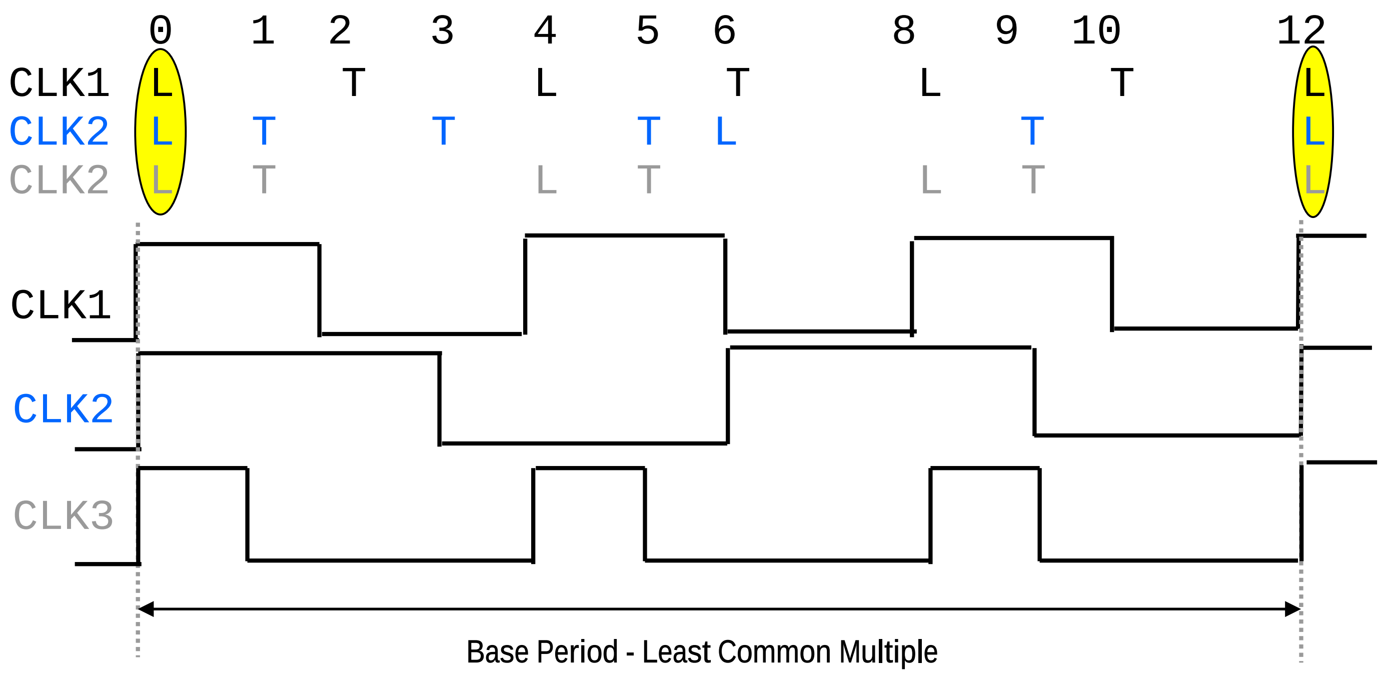
```
set_multicycle_path –hold 1 -from { ff1b } -to { ff2d}
```

- The check moves the hold check to the preceding edge of the start clock.



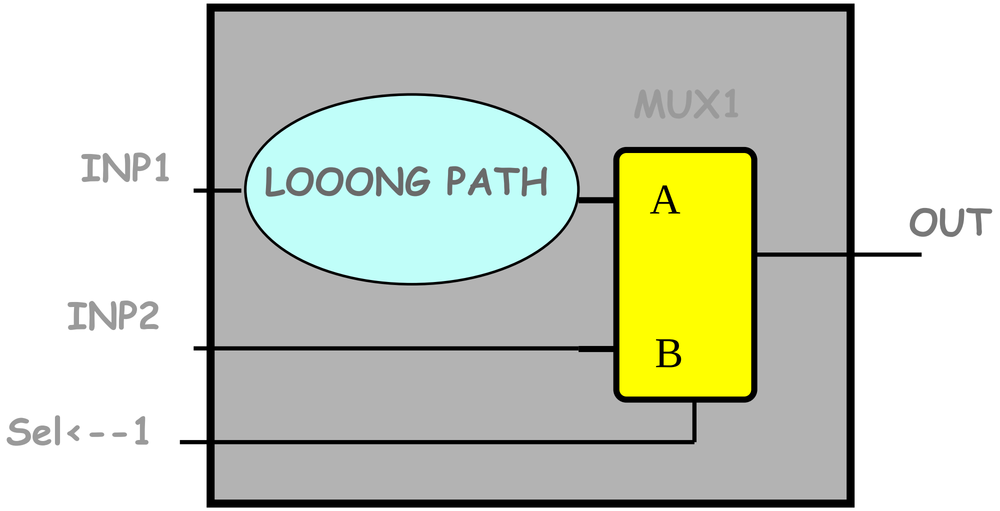
Understanding Multiple Clocks

- ❖ If more than one clock is used to time a design, you can define them to have different waveforms and frequencies
- ❖ If clocks have different frequencies there must be a base period over which all waveforms repeat
  - – Base period is the least common multiple (LCM) of all clock periods



Understanding False Paths

- ❖A path that can never be sensitized in the actual circuit
  - – These paths are those that are logically/functionally impossible



- ❖The designer should specify to the synthesis tool that the LOOONG path(comb or reg to reg) is false
- ❖The goal in static timing analysis is to do timing analysis on all “true” timing paths

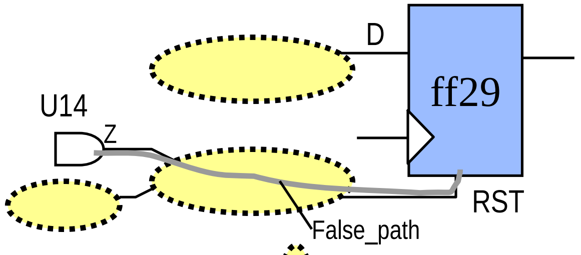
set_false_path

- ❖ Command marks startpoint/endpoint pairs as false timing paths
- ❖ Essentially disables max delay (setup) and min delay (hold) checks

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

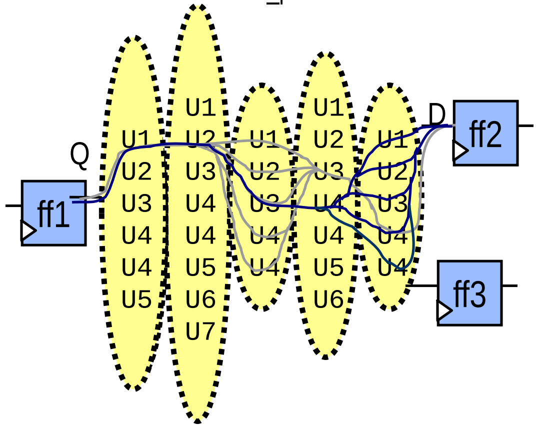
```
set_false_path -from U14/Z -to ff29/RST
```

- Setup/Hold checks from and gate output U14/Z to ff29/RST is disabled. Every other timing paths are honored



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

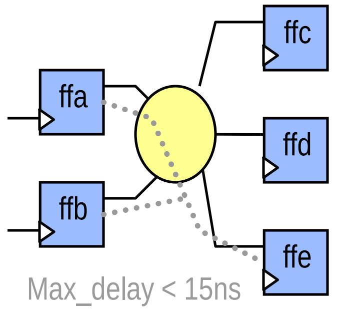
- Command disables all timing paths from ff1/Q to ff2/D which passes through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C}.
- Multiple Paths are Affected! Specify General False Path carefully!



set_max_delay

- ❖ Specifies maximum delay for paths
- ❖ Within a given point-to-point exception command, the more specific command overrides the more general.
- ❖ Precedence list (from specific to general) of what gets overridden and honored
  1. set_max_delay -from pin -to pin
  2. set_max_delay -from clock -to pin
  3. set_max_delay -from pin -to clock
  4. set_max_delay -from pin
  5. set_max_delay -to pin
  6. set_max_delay -from clock -to clock
  7. set_max_delay -from clock
  8. set_max_delay -to clock

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

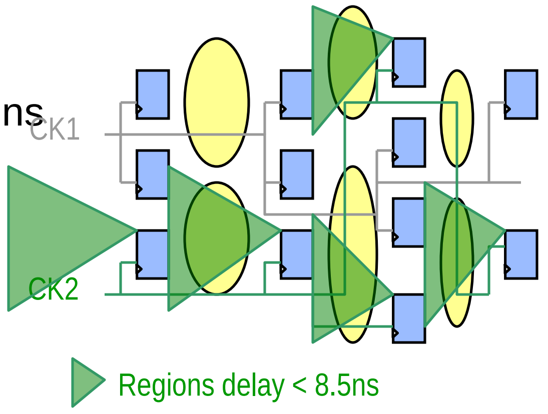


```
set_max_delay 15.0 -from {ffa ffb} -to {ffe}
```

- Specifies all paths between ffa & ffb to ffe should be less than 15ns

```
set_max_delay 8.5 -to [get_clocks CK2]
```

- Specifies all paths to endpoints clocked by CK2 to be less than 8.5ns



Priority of the Commands

- ❖Remember that constraints have priorities in the tools. Sometimes your constraints are overridden by another because of its priority. Make sure you have the appropriate constraints applied to your design.
- ❖For the exceptions through the same path, the following is the priority

`set_false_path >  set_max/min_delay > set_multicycle_path`

✓To check the validity of the exception, as to why it has a problem.

```
report timing [eval [find / paths <EXCEPTION>]]
```

Rule of Specificity

- If you apply the same type of timing exception using commands with different path specifications, the more specific command has priority over the more general one.
- For example,

```
set_max_delay 12 -from [get_clocks CLK1]
set_max_delay 15 -from [get_clocks CLK1] -to [get_clocks CLK2]
```

Rule of Specificity

- Exceptions of any type on more specific objects, such as pins or ports, take precedence over exceptions applied to more general objects, such as clocks.
- For example,

```
Set_false_path –from [get_clocks clk1]
Set_false_path –from [get_pins a_reg/ck]
```

Priority of the Options

- The various -from/-to path specification methods have the following order of priority, from highest to lowest:
  1. -from *pin*, -rise_from *pin*, -fall_from *pin*
  2. -to *pin*, -rise_to *pin*, -fall_to *pin*
  3. -through, -rise_through, -fall_through
  4. -from *clock*, -rise_from *clock*, -fall_from *clock*
  5. -to *clock*, -rise_to *clock*, -fall_to *clock*

❖Static Timing Analysis

Basic Checks for Correct Static Timing Analysis

- The basic requirements for the STA tools to report the path delays appropriately are as following:
  - Complete and correct circuit
  - Complete constraints
  - Realistic constraints
- STA analysis is important for synthesis and all implementation/optimization tools
  - STA run is only as good as your constraints
  - Synthesis QoR is as good as your constraints
- Use CCD to check constraints early and often

Complete Constraints

- All flip-flop to flip-flop paths must be constrained.
- All input to flip-flop paths must be constrained.
- All flip-flop to output paths must be constrained.
- All combinational paths must be constrained.

❖ This means:

- All flip-flops must have a declared clock.
- All inputs must have an input delay in relation to a clock (preferred) or a data arrival time.
- All output must have an external delay in relation to a clock (preferred) or a data required time.
- All combinational paths can have a maximum delay if not constrained by their input output delays.

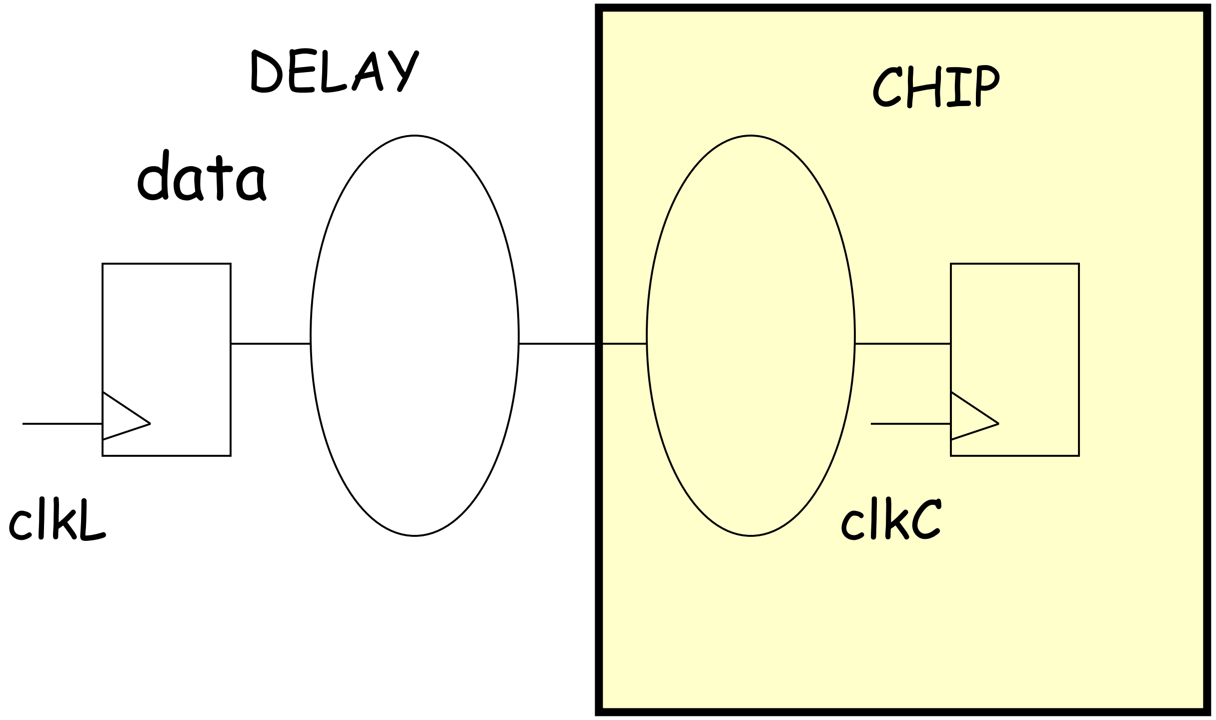
Realistic Constraints

- A constraint is realistic if a positive delay exists such that a circuit with such delay can meet the delay requirement of the path under analysis.
  - Positive required time
- Unrealistic constraints occur at the inputs and outputs of the circuit.
  - Negative required time (CCD_CLK_UNC3 (warning), CCD_MISC_MSC11 (warning) etc)
  - Input delay > 80% of the clock delay then only 20% is left for the rest (CCD_IO_IDL5 (warning), CCD_IO_ODL5 (warning) , sdc_input_delay_ratio, sdc_output_delay_ratio)
  - Set_min_delay is greater than set_max_delay (CCD_DGN_CMB2 (warning))

Complete Set of Constraints

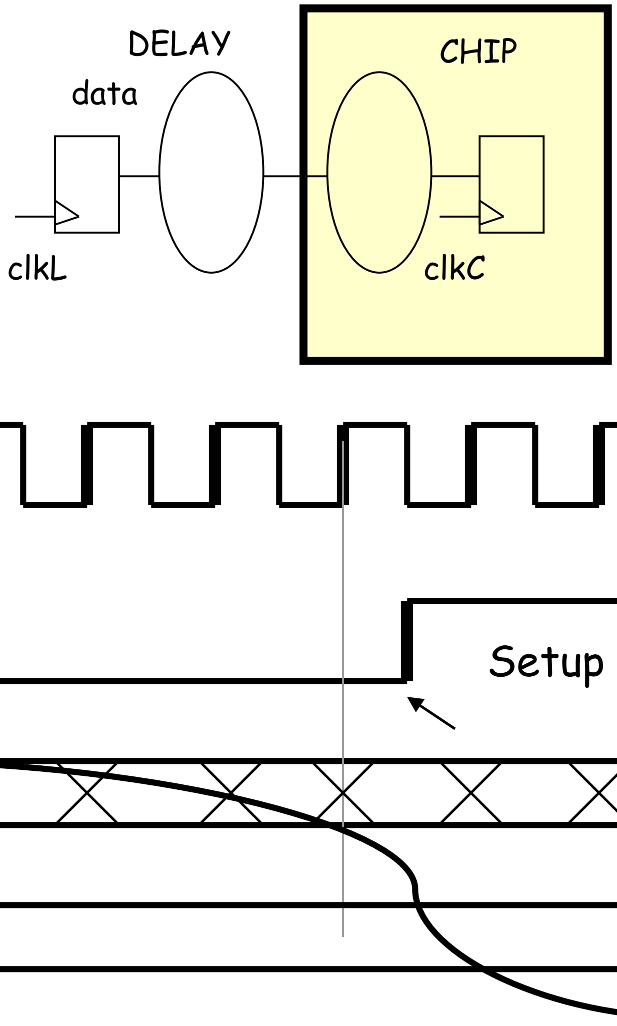
- Having a complete set of constraints might not be as easy as you think
  - STA tool typically makes some assumptions and continue
  - Missing clock uncertainty
    - STA tool may assume it to be zero
  - Missing IO delays
    - STA tool assumes them to be zero and move on
    - Missing output delay may cause the paths to be unconstrained in some implementation tools
  - Conflicting set_case_analysis
    - Constraint propagated downstream in the design
    - STA again makes assumptions and continue
    - Part of the circuits get timed incorrectly

Example 1: Unrealistic Constraint



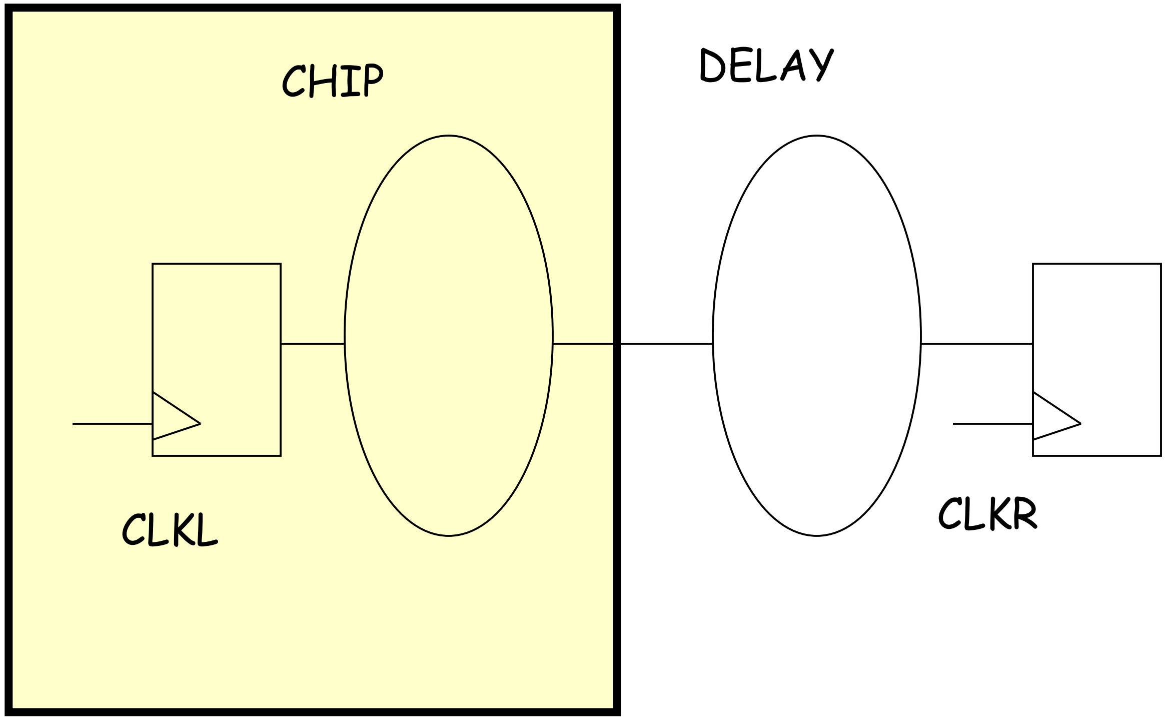
clkL = 100MHz TL = 10ns
clkC = 10MHz TC = 100ns
DELAY = 90nS

Example 1: Unrealistic Constraint (continued)



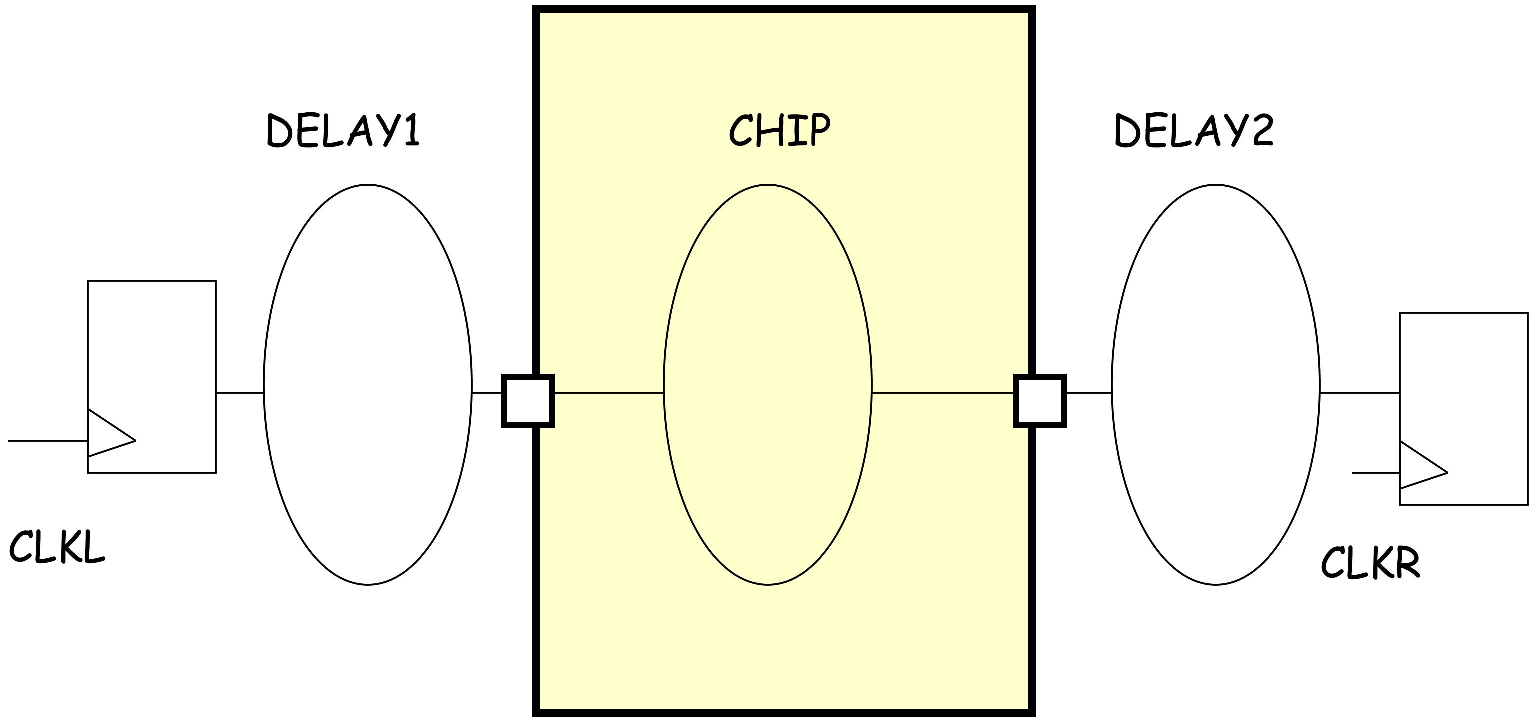
A delay longer than the launch period causes a setup violation.

Activity: Unrealistic Constraint



CLKL = 100MHz TL = 10ns
CLKR = 10MHz TR = 100ns
DELAY = 90ns

Activity: Unrealistic Constraint



CLKL = CLKR = 10 MHz; T = 100ns

DELAY1 = DELAY2 = 90nS

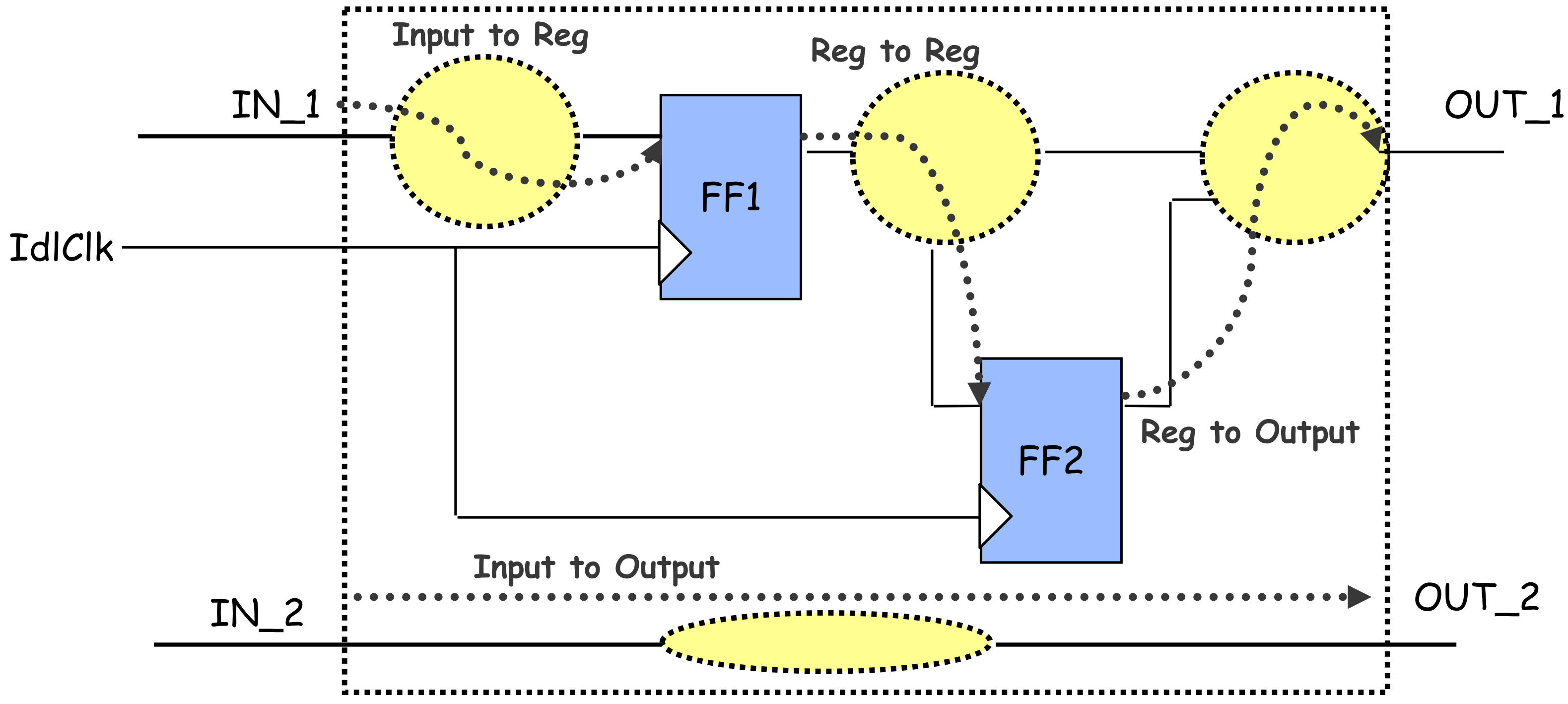
Modes of SDC

- ❖ It is a good practice to use as few SDC files as possible. There are two basic modes of SDC files:
  - – Test mode SDC
    - – Used during the synthesis of DFT test mode.
  - – Functional mode SDC
    - – Used during the synthesis of the normal functionality of the chip.
- ❖ There are two more classifications of SDCs as the following:
  - – Pre-CTS, to use in ideal mode (no clock tree) so that the clock network latency (network insertion) is taken into consideration.
  - – Post-CTS, to use during the clock propagated mode when the clock insertion delays are replaced by the actual clock tree delays.
- ❖ You can have one file for each of the checks for all the above cases:
  - – Setup, and Hold

Multimode SDC

- A mode defines one of possibly many different functional, test behaviors of a design. You can create an SDC file for each of the cases such as setup, hold, functional or test.
- You can combine the SDCs using *set_case_analysis* or by creating virtual clocks in your design, and setting functional false paths between the real and virtual clocks.

Review Timing Path Types



Different Timing Path’s Requirements

- Input port to Register
  - Requires User to define the clocks & set the data arrival time at the input port
- Register to Register
  - Requires User to define the clocks
- Register to Output port
  - Requires User to define the clocks & set the output port’s external delay
- Input port to Output port
  - Requires User to properly budget timing using synchronous input & output delays

Arrival Time

- Time when signal arrives at the checkpoint.
  - Could be a seq. element
  - Could be a checking gate (clock gating checks)
  - Could be a port with external delay statement

```
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
```

Static Timing Analysis

- ❖ All paths are timed between synchronous clocks
  - – Paths originating and ending at the same clock will be given 1 clock cycle
    - – CLKA = 10ns, paths starting from and ending at CLKA will be timed 10ns
  - – When there are cross clock domains (launch on CLKA capture on CLKB) STA analysis will find the worst launch/capture edge pairs
    - – STA needs to unroll the clocks to find the least common multiple of them
  - – Clocks between asynchronous clocks are not timed
- ❖ All paths are single-cycle paths by default in the absence of timing exceptions

cādence™