# Introduction To Static Timing Analysis

Alexander Gnusin

# Two types of Timing Verification: Dynamic Timing Simulation (VCS or Spice)

Advantages:

- Can check asynchronous interfaces
- No need to specify false and multicycle paths, clock model etc
- Can be very accurate (Spice)

Disadvantages:

- Analysis quality depends on stimulus vectors
- Non-exhaustive – virtually impossible to check every path
- Long run times

# Two types of Timing Verification: Static Timing Simulation (Primetime)

Advantages:

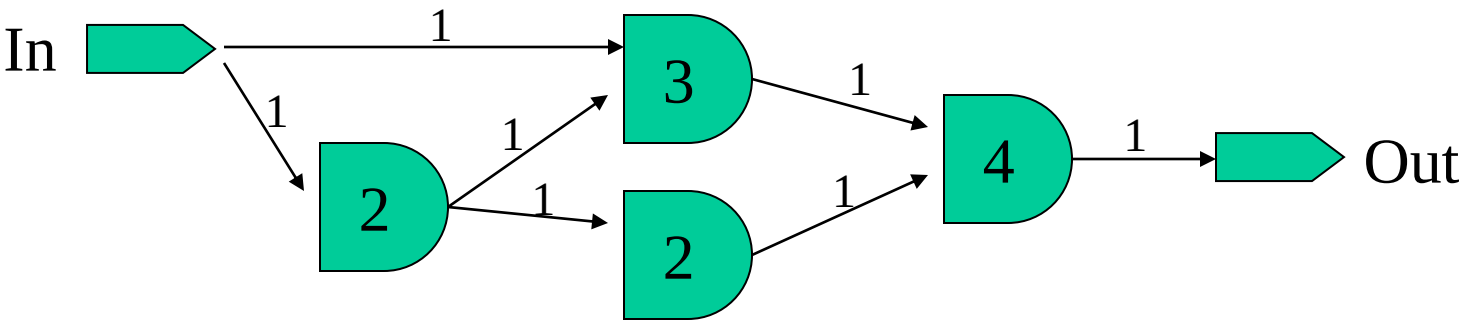
- Fast – design is analyzed in one pass, for one clock cycle
- Exhaustive – checks all topological paths in design
- Doesn’t require verification environment

Disadvantages:

- Less accurate
- Must define timing requirements / exceptions
- Difficulty handling asynchronous designs, false paths

# Principles of Static Timing Verification

- Netlist is represented as DAG – Directed Acyclic Graph
- Delay values associated with nodes (Cells) and links (Nets)
- Total Path delay is the sum of Path delay values



# Block-based vs Path-based STA

***Block–based*** : timing information associated with discrete design elements (ports, pins, gates)

Two types of timing data :

- Arrival times, AT (propagated forward from inputs)
- Required Arrival Times RAT (propagated from outputs)
- Slack is calculated on every design element:

  ***Slack = RAT – AT***

Used in EinsTimer (IBM)

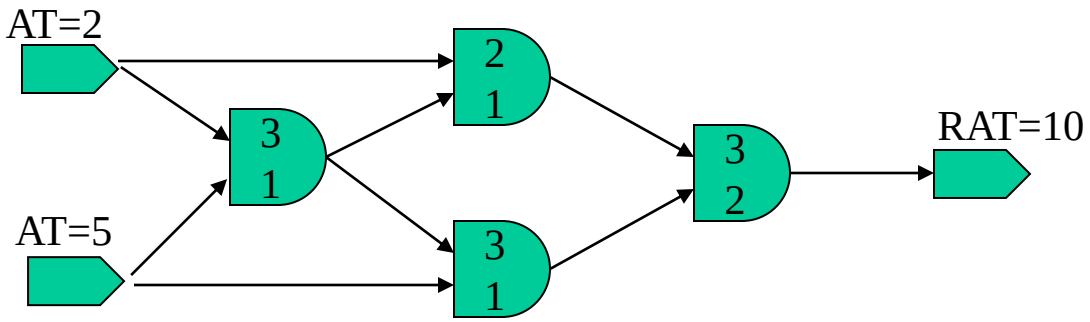
***Path-based*** : timing information associated with topological paths (collections of design elements)

First, extract all possible topological paths

Next, for each path calculate it's delay and compare it with endpoint (required) value

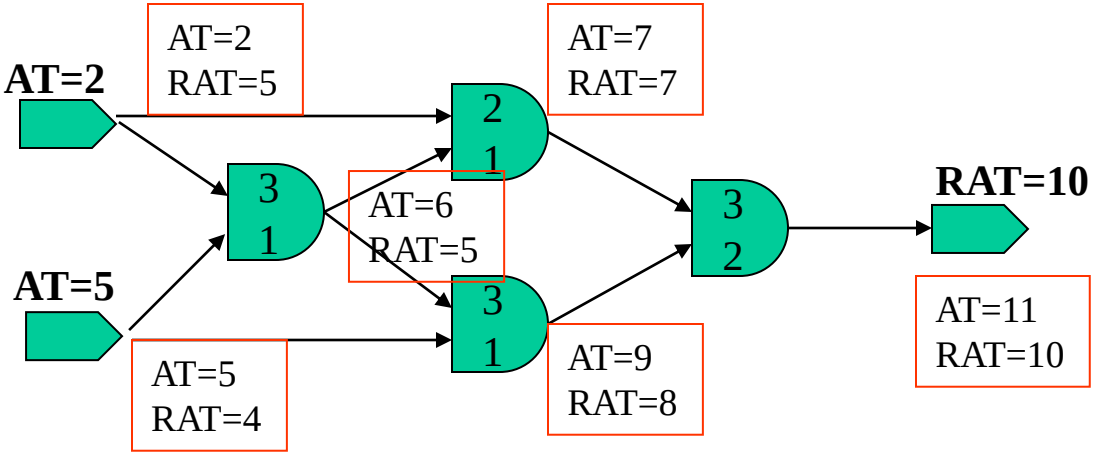
Used in Primetime

# Block-based vs Path-based STA (example)




***Path-based:***

2+2+3 = 7 (OK)
2+3+1+3 = 9 (OK)
2+3+3+2 = 10 (OK)
5+1+1+3 = 10 (OK)
5+1+3+2 = 11 (Problem!)
5+1+2 = 8 (OK)




***Block-based:***

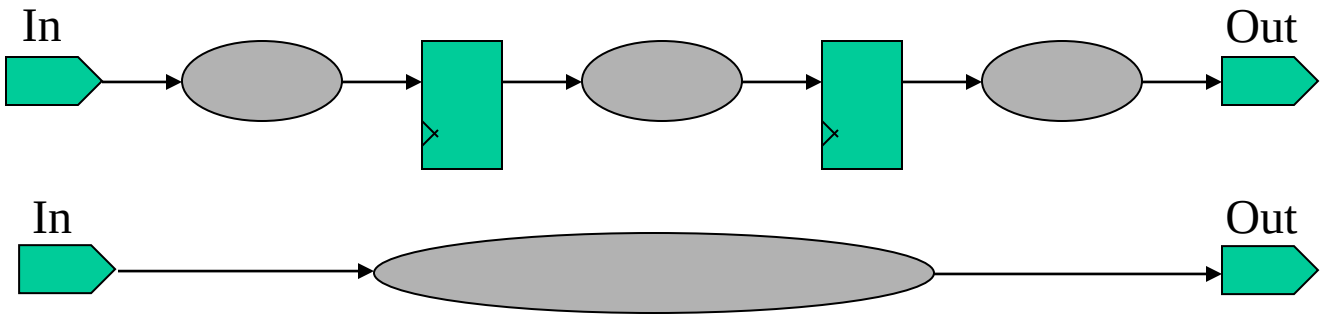
Critical path is determined as collection of gates with the same, negative slack:
**Slack = RAT – AT**
In our case, we see one Critical path with slack = -1

# Four Types of Timing Paths

- Input – to – Register (Synchronous)
- Register – to – Register (Synchronous)
- Register – to – Output (Synchronous)
- Input – to – Output (Asynchronous)

Each path has :

- ✓ Startpoint (Input port or FF output)
- ✓ Endpoint (Output port or FF input)
- ✓ Calculated value for path delay



# Path delay calculation

*The actual path delay is the sum of Net and Cell delays along the timing path*

- Net Delay – total time for charging/discharging all the parasitics of a given net. It is a function of:
  - ✓ Net capacitance (derived from net's length)
  - ✓ Net resistance (derived from net's length)
- Cell Delay – delay arc between corresponding input and output ports of the cell. It is a function of:
  - ✓ Input Transition Time (Slew Rate)
  - ✓ Total Output Load (Net Cap + Sum of attached pin caps)
  - ✓ Process Parameters (Temperature, Power Level)

# Net Delay Calculation (PreLayout)

- Net Length is a function of net fanout and chip area
- For a given area, averages of R and C are calculated for different fanouts
- Net delay is calculated simply as R * C

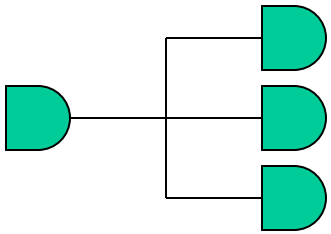
Capacitance as a function of fanout :

| | |
|---|---|
| 1 | 0.015 |
| 2 | 0.030 |
| 3 | 0.046 |

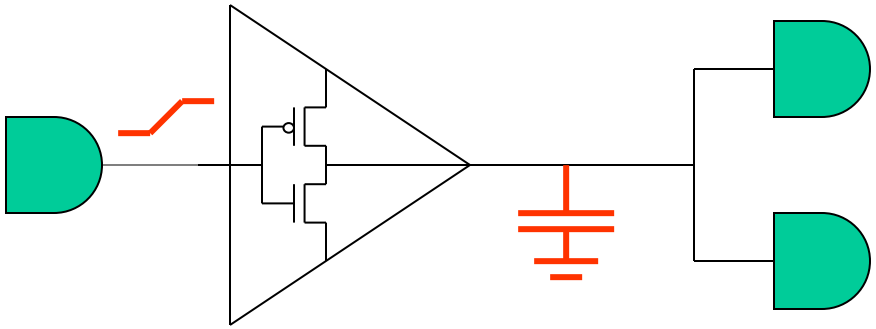
Resistance as a function of fanout :

| | |
|---|---|
| 1 | 0.012 |
| 2 | 0.016 |
| 3 | 0.020 |

For fanout = 3
Net delay = 0.046 * 0.020



# Cell Delay Calculation (PreLayout)



***Temperature, Power Level, Process***

Cell delay is function of :

- Input Transition Time (calculated by previous gate)
- Total Output Capacitance (Net Cap + Sum of attached Pin Caps)
- Operating Conditions

# Cell Delay and Slew Calculation

- Delay is calculated using 2-Dimensional Nonlinear Delay Model. Main calculation part is interpolation between nearest table values:

*Total Cload (fF)*

| *Input Transition (ns)* | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|
| 0 | 3 | 4.5 | 6 | 7 |
| 0.1 | 5 | 8 | 10 | 13 |

- The Similar Model is used for Cell Slew Calculation
- After calculation, delay is scaled by the operating conditions:

$$\text{Final\_delay} = \text{Table delay} * K_{voltage} * K_{temp} * K_{process}$$

# Cell Delay and Slew Calculation (Cont)

Delay Calculation:

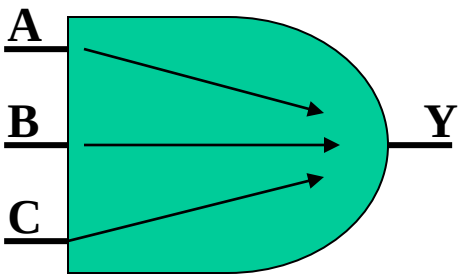
1. Determine input slew rate from the previous cell
2. Determine total output load : net + sum_of_pins
3. Calculate (Interpolate) Cell Delay
4. Scale Cell Delay (based on Operating Conditions)

- Slew Calculation:

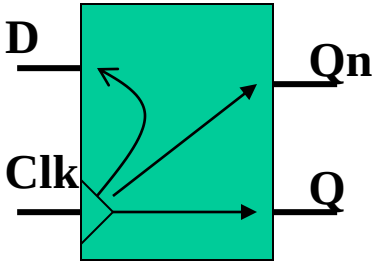
1. Determine input slew rate from the previous cell
2. Determine total output load : net + sum_of_pins
3. Calculate (Interpolate) Output Slew
4. Scale Slew Rate (based on Operating Conditions)

# Types of Cells Timing Arcs

Combinatorial Cells : from all inputs to output:



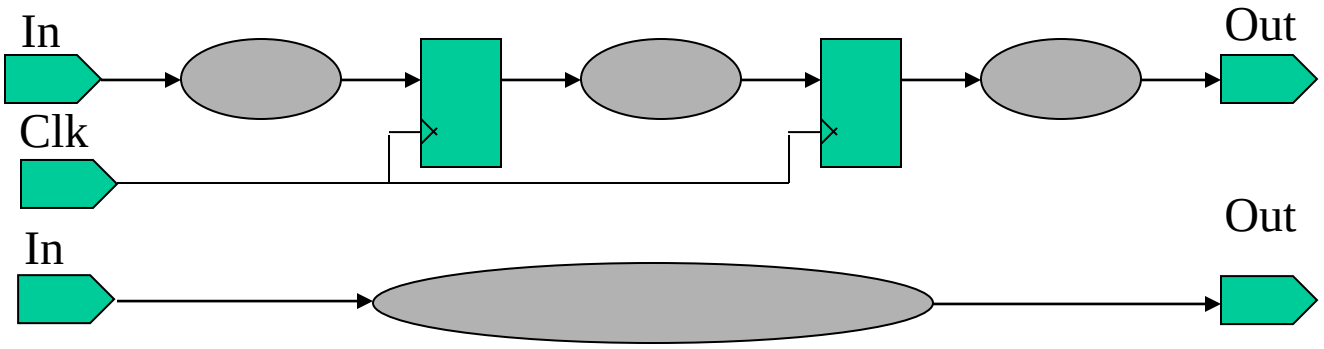

Sequential Cells : from Clock Input to outputs (propagation delay) and from Clock Input to Data Input (setup, hold checks)



# Path Constraints Types

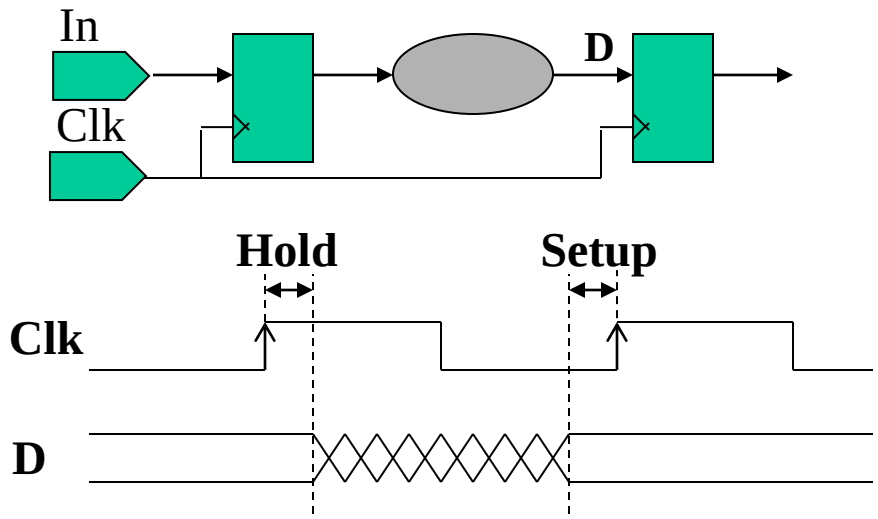
Main path constraints include:

- **Setup time** (Input-to-Reg and Reg-to-Reg paths)
- **Hold time** (Input-to-Reg and Reg-to-Reg paths)
- **Input delay** (Input-to-Reg and Input-to-Output paths)
- **Output delay** (Reg-to-Output and Input-to-Output paths)



# Review: Setup and Hold Times

- **Setup**: amount of time data must be stable at the data pin of FF before the clock capturing edge
- **Hold**: amount of time data must remain stable at the data pin of FF after the clock capturing edge



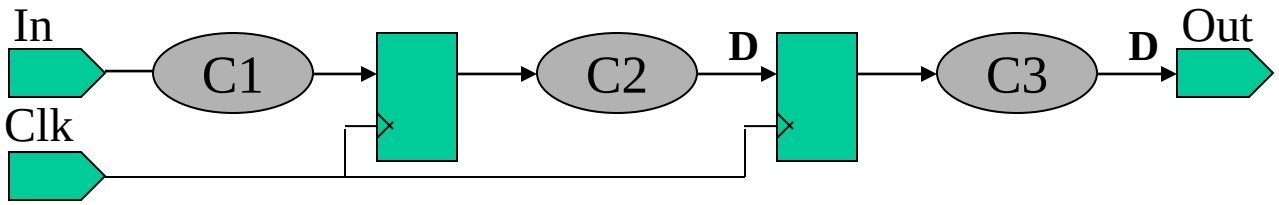

Hold Check refers to the same clock edge:

$$T_D - T_{CLK} > T_{Hold}$$

Setup Check refers to the next Clock edge (add clock period) :

$$(T_{CLK} + T_{Period}) - T_D > T_{Setup}$$

# Setup and Hold Times (Numerical Example)




*Primetime Settings:*
**Create_clock –p 10.0 –waveform {0 5}**
**Set_input_delay 5.0**
**Set_output_delay 6.0**
*Technology Library Data of FF:*
**Tsetup = 0.8 , Thold = 0.6**

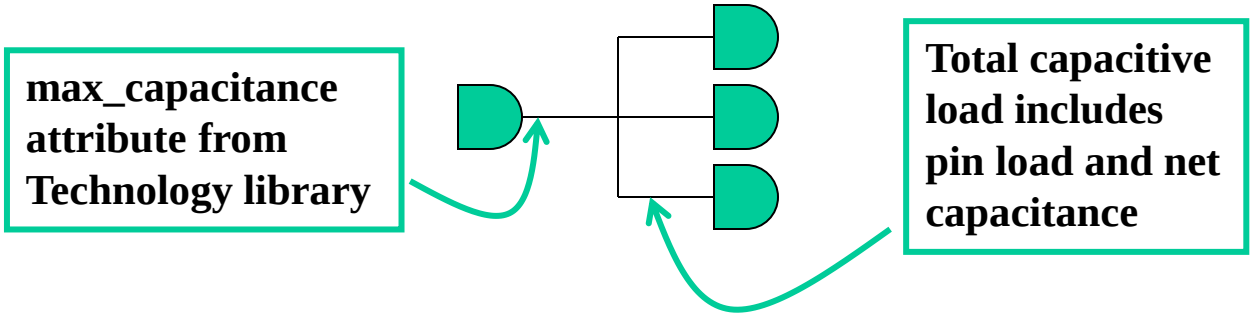
*Please, calculate yourself:*

- C1 constraints : max_delay: min_delay:
- C2 constraints : max_delay: min_delay:
- C3 constraints : max_delay: min_delay:

# Other constraints: Design Rules Check

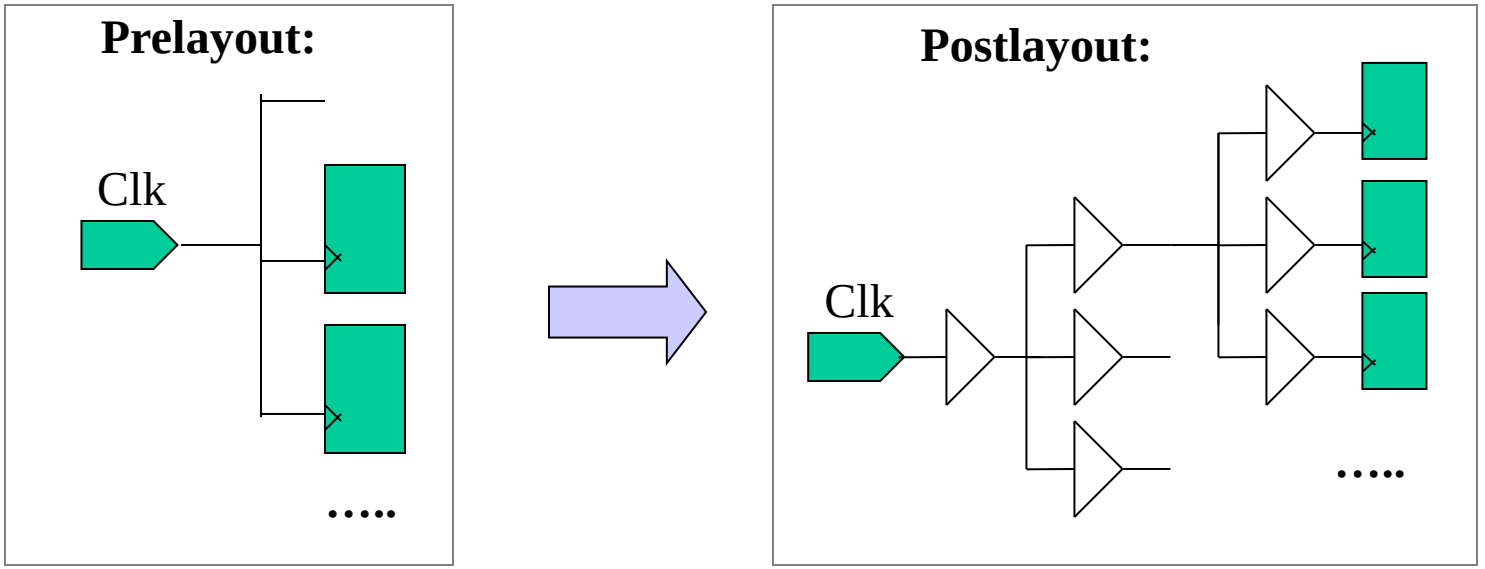
- Design Rules are electrical checks performed on each gate. They are defined in the technology library
- Checking min and max limits for:
  - ✓ Net/Port capacitance
  - ✓ Net Transition times
  - ✓ Net fanout

Max capacitance example:



# Clock Modeling

- Clock reaches FFs in the different times: need to model clock skew
- Need to model the latency of the clock tree
- In prelayout, by default clock tree delay and skew are “0”



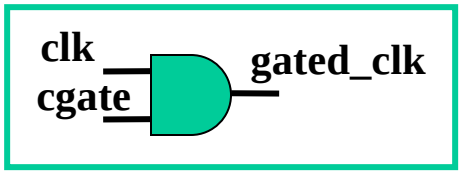

**Set_clock_uncertainty – to model clock skew**
**Set_clock_latency – to model clock tree delay**

**Set_propagated_clock –**
**To calculate real delays**

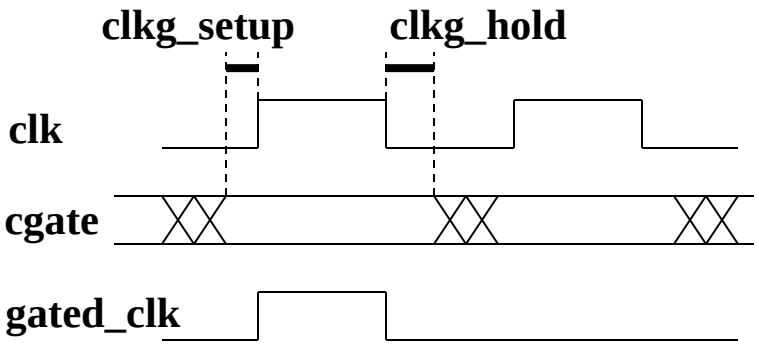
# Clock Gating Checks

Glitches – the main problem of clock gating.

Clock gating checks:

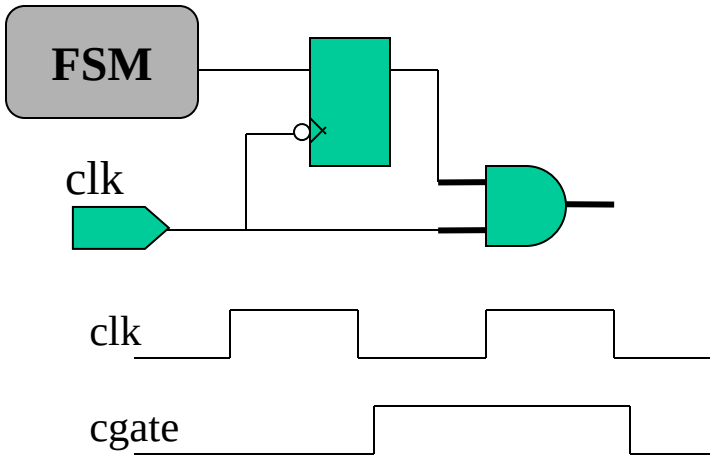



- Clock gating setup : amount of time cgate must be stable before the clock rising edge
- Clock gating hold : amount of time cgate must be stable after the clock falling edge



# Clock Gating Design

- No problems with clock gating setup: cgate changes in the middle of period
- Still some problems with clock gating hold, but usually FF delay + net delay are sufficient

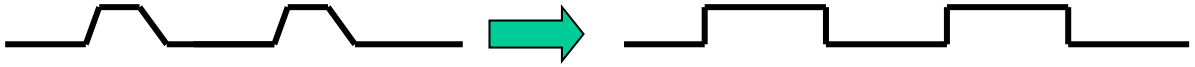



Primetime command:
**set_clock_gating_check –setup 0.5 –hold 0.5 [get_clock CLK]**

# Clock Modeling and PLL

PLL functions:

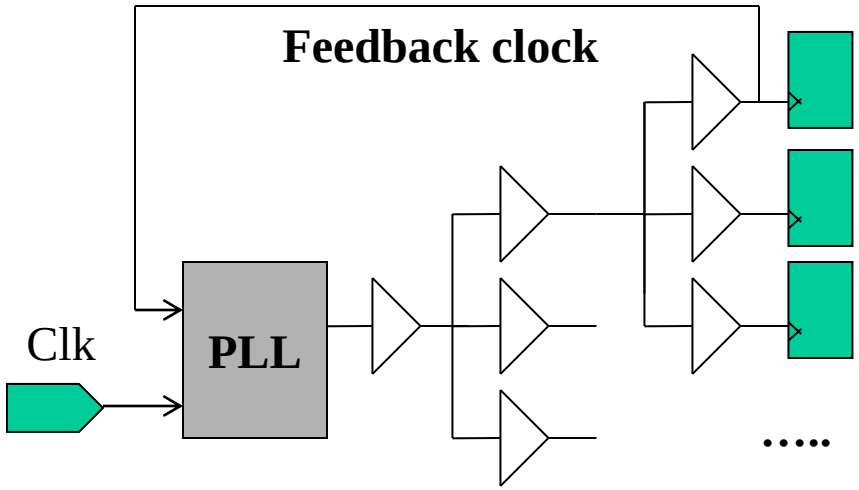
- Correct clock's waveform (duty cycle):



- Multiply reference clock:

- Remove clock tree delay:

**On the output,PLL generates waveform with phase shift. PLL adjusts the value of phase shift in order to receive the reference clock rising edge and the feedback clock rising edge at the same time**



# Clock Modeling and PLL (cont)

So, PLL removes clock tree delay. But PLL itself has:

- **Jitter**: the amount by which PLL output clock period can vary on consecutive cycles: 

*Can be modeled as additional uncertainty for setup checks*

- **Offset**: the amount by which PLL output clock can lead or lag the PLL reference clock

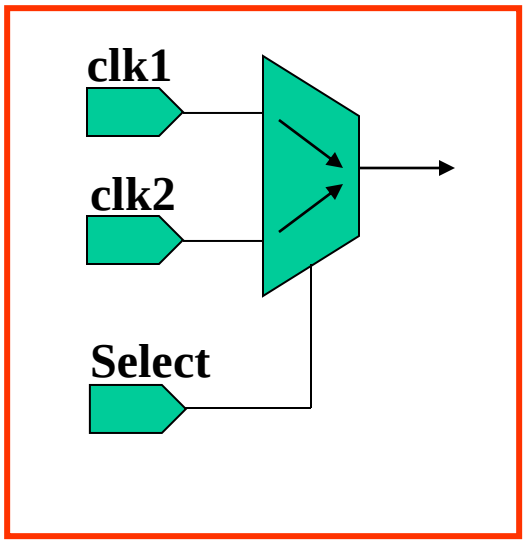
*Can be modeled as additional uncertainty for setup and hold checks*

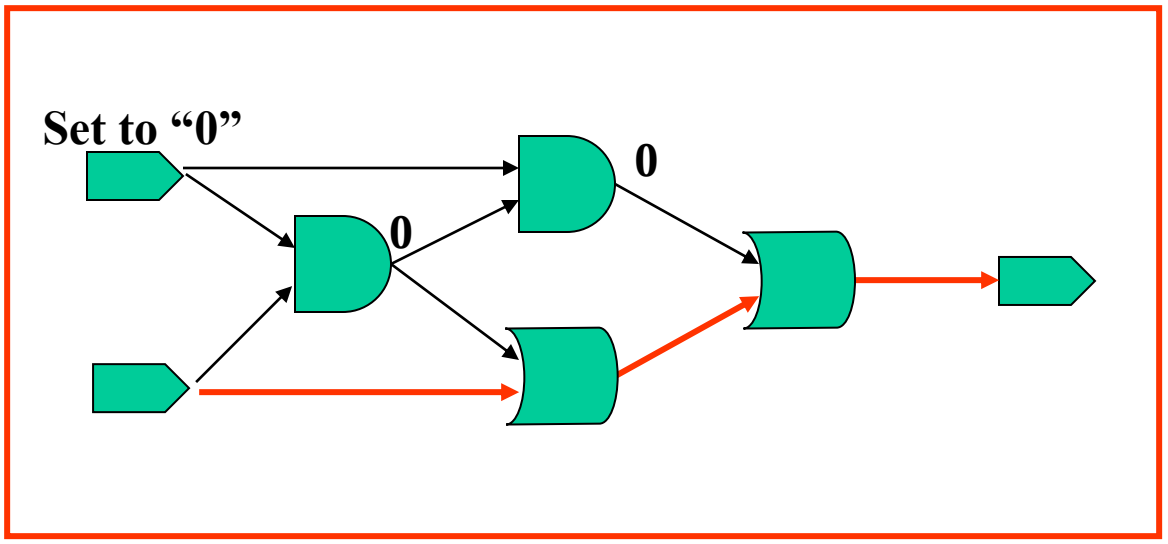
In addition, PLL receives as feedback only one point from the end of clock tree. There is also clock tree skew between this and others FFs.

*Can be modeled creating virtual clock with negative latency (equal to the clock tree delay) on PLL output*

# STA and Case analysis

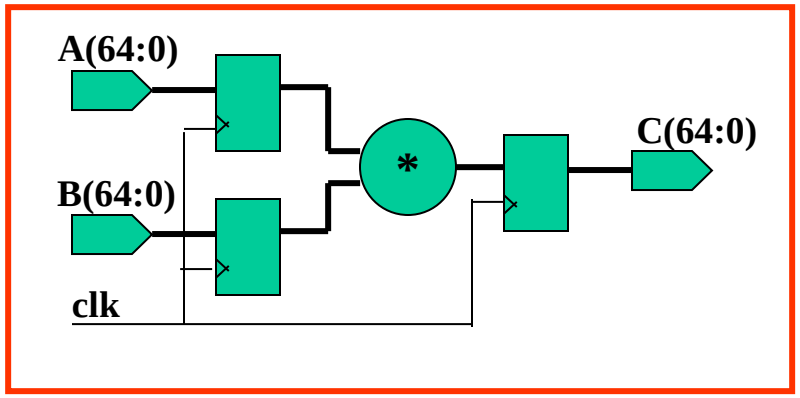
- STA algorithms don’t use functional cell descriptions. However, case analysis brings some elements of functionality to STA
- Setting control input signals to constant logic value reduces the number of topological paths to calculate
- It is very dangerous, however …



# False & Multicycle Paths

- Asynchronous clocks : don't check crossdomain paths!
  set_false_path –from [get_clocks CLKA] –to [get_clocks CLKB]
  set_false_path –from [get_clocks CLKB] –to [get_clocks CLKA]
- Any other path you don't wish to constraint (for example, Reset)
  set_false_path –from [get_ports Reset
- Multicycle Paths constraint (example: multiplier)




**Clk_period = 5ns; 20 < mult delay < 30**

set_multicycle_path -setup 6 ...
set_multicycle_path -hold 4 ...

**•Add 6 periods (instead of 1) for setup**
**•check**
**•Add 4 periods for hold check**
**•Hold fixing forces all data changes**
**•into the last 2 cycles**

# Post-Layout Delay Calculation

Real nets capacitances and resistances are known – more accurate timing.

SDF (Standard Delay Format) is extracted. It can include:

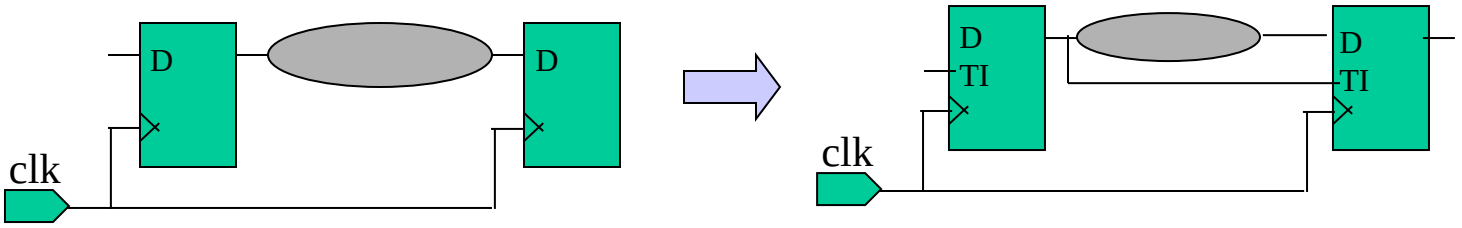
- *All timing information, including nets delays and cell delays*
- *Net delays timing information only.*

Then, netlist and SDF are loaded to STA in order to recheck timings

- *In the first case, STA only accumulates delays provided by STA (without any delay calculations)*
- *In the second case, STA calculates cell delays (from library tables) and uses annotated values of R and C for nets instead of statistical wireload model*

# Scan Insertion and STA

Scan chain insertion has some influence on STA – add 1 more fanout to FF output net:




In order to model scan insertion for STA (when running STA before scan insertion) “pseudo-scan” register are inserted running Synopsys command *compile –scan :*