A Brief History of Timing

David Hathaway
February 28, 2005

Outline

- **Snapshots from past Taus**
- **Delay modeling**
- **Timing analysis**
- **Timing integration**
- **Future challenges**

Tau Workshop

- **Longest title…**
  - ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems
- **… but shortest nickname**
  - τ
- **Held 9 times at irregular intervals since 1990**
- **Workshop focus has shifted over time**

- **My focus will be on:**
  - Timing analysis (not optimization)
  - Synchronous systems
  - Netlist level and below
  - On-chip

Tau 1992

- **General**
  - 2.5 days
  - ~ 50 people?
  - 28 talks, 2 panels
- **Topics**
  - 11 (+1 panel): Asynchronous timing
    - Most heard phrases: “isochronic fork,” “bounded delay”
  - 9: Logical / timing analysis (false paths, etc.)
  - 4: Transparent latch timing / pipelining
  - 1 (+1 panel): Delay modeling

Tau 1997

- **General**
  - 2 days
  - ~ 120 people
  - 22 talks, 18 posters
- **Topics**
  - 9: Impacts of small geometries
    - Most heard phrase: “deep submicron”
    - Included 5 on cross-talk analysis
  - 6: Asynchronous timing
  - 6: Logical / timing analysis
  - 5: Delay modeling
  - 4: Retiming
  - 2: Useful skew

Tau 2005

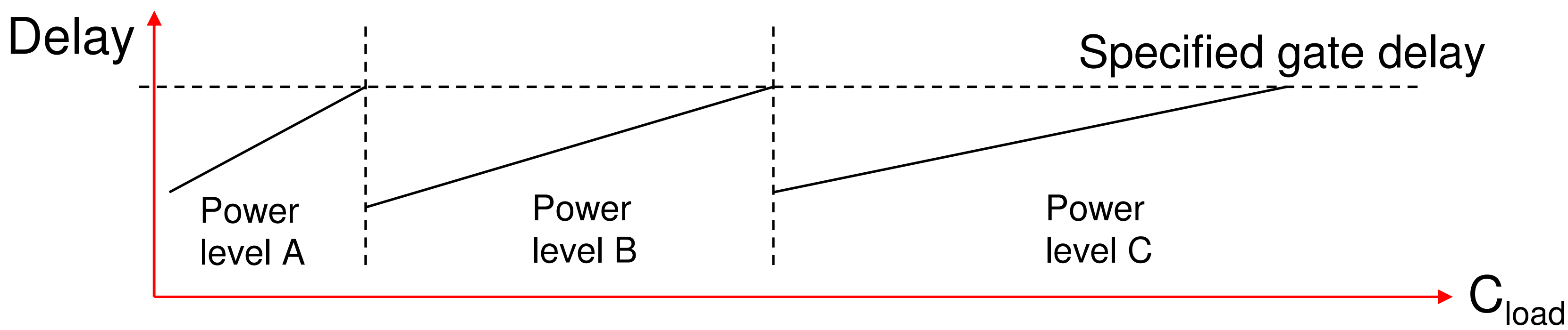
- **General**
  - 1.5 days
  - 16 talks
- **Topics**
  - 8: Statistical timing / optimization
  - 2: Asynchronous / timing of cyclic networks
  - 2: Clocking schemes

Outline

- Snapshots from past Taus
- **Delay modeling**
- Timing analysis
- Timing integration
- Future challenges

Early delay models

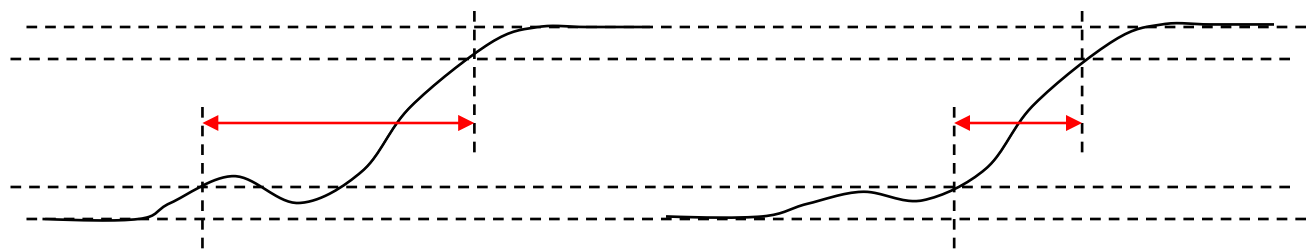
- **Constant delay per gate**
  - Power levels used to keep delay constant



- **CMOS delay load-dependence originally just fanout**
  - Ignored wire load, load difference between gates
- **Bipolar load-dependence more complicated**
  - Included DC currents (like gate leakage?)
  - Delay models included explicit dependence load cell
    - … if block X drives cell Y, use delay d_{XY} …

Slew dependence

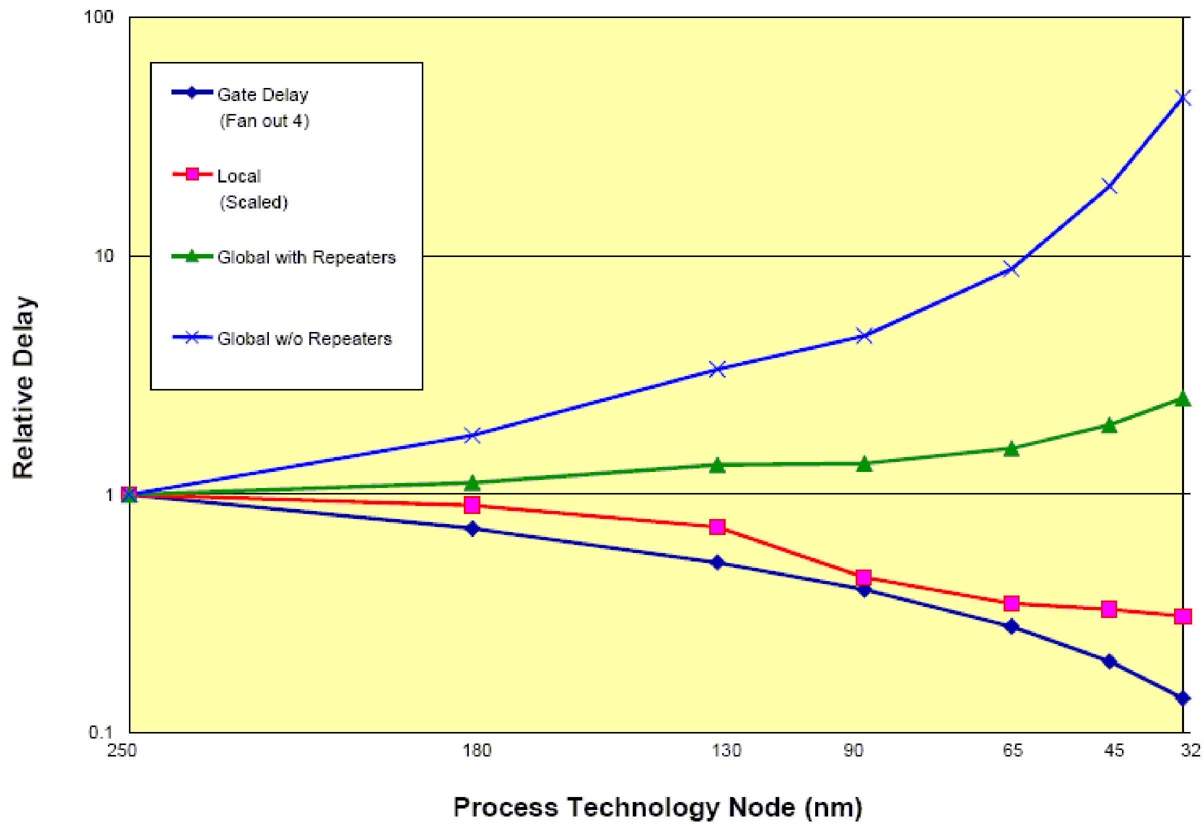
- **Delay models both began to use and produce slews**
  - Typically measured as 10%-90% or 20%-80% time
- **Simple scalar slew model is limited**
  - Shape of waveform may affect delay
  - Discrete crossings can cause discontinuities



- **Recent alternatives**
  - Piecewise-linear waveform
    - Useful for simulation-based delay calculation (e.g., transistor-level)
  - Metrics based on weighted waveform integration

Wire impact on delay

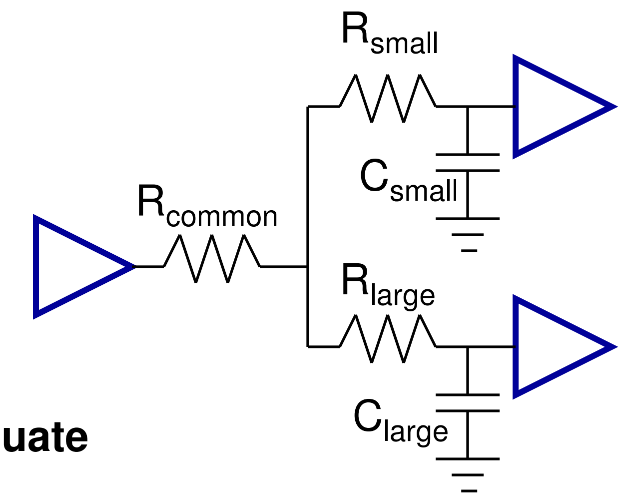
- **Originally considered only wire capacitance**
  - Allowed single timing value (e.g., arrival time) for entire net
  - Used wire load models – no actual placement / wiring data
- **RC wire delay itself became important**



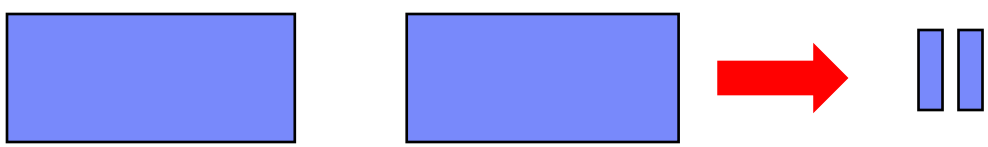
[ITRS 2001 roadmap]

Wire delay models

- **Elmore delay**
  - Analytic form useful in optimization
  - Problems arose due to resistive shielding
- **Model order reduction**
  - AWE, RICE, PRIMA, …
  - Higher computational cost, higher accuracy



- **Pure capacitive gate load model became inadequate**
  - C_{eff}, Pi model
- **Lateral wire capacitance becoming dominant**



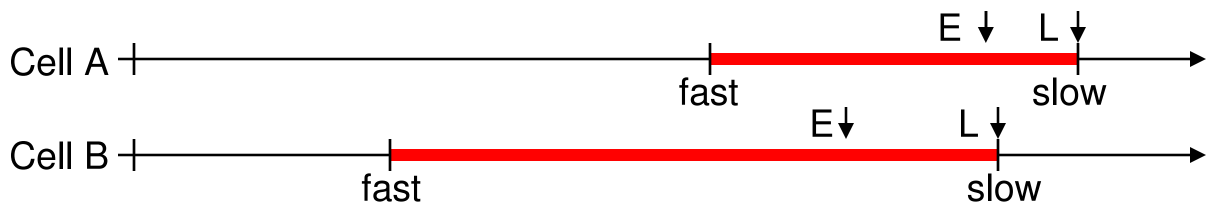
- **Guard banding min/max effective capacitance too pessimistic**
- **Coupling delay models**
  - Equivalent grounded capacitance based on total charge injected into wire
  - Dynamic simulation

Delay variability

- **Initially considered by process corner analysis**
  - All delays “fast” or “slow”
  - Perfect correlation
- **Across chip variation became important**
  - Important for tests comparing early / late times
  - One way: assume x percent min (late) / max (early) delay variation
    - Problem: not all cells have same sensitivity to process variation
  - IBM “LCD” (linear combination of delays) approach

Late @ 1.0 * slow

Early @ 0.8*slow + 0.2 * fast



Delay impact of variations

Parameter	*Delay Impact*
BEOL metal (Metal mistrack, thin/thick wires)	-10% → +25%
Environmental (Voltage islands, IR drop, temperature)	±15 %
Device fatigue (NBTI, hot electron effects)	±10%
V_t and T_{ox} device family tracking (Can have multiple V_t and T_{ox} device families)	± 5%
Model/hardware uncertainty (Per cell type)	± 5%
N/P mistrack (Fast rise/slow fall, fast fall/slow rise)	±10%
PLL (Jitter, duty cycle, phase error)	±10%

- **Requires 2^{20} timing runs or [-65%,+80%] guard band!**

[Courtesy Kerim Kalafala & Chandu Visweswariah]

Delay rules

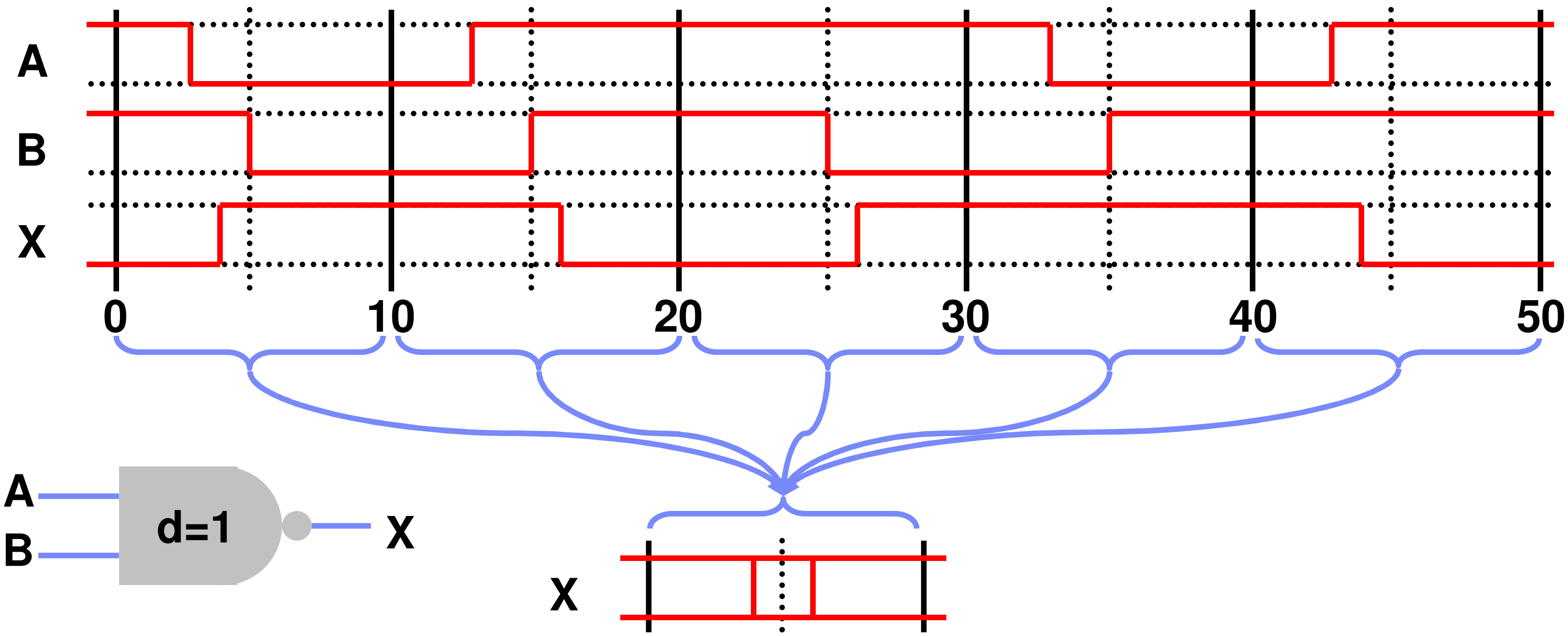
- **Many approaches**
  - Tables
  - Fixed equations
  - Simulation-based methods
    - Fast transistor-level simulator
    - Equivalent current source models
- **Need flexibility**
  - Both dependencies and functional form of delay are changing
  - Need to separate delay calculation algorithm from delay interface
    - Not possible with .lib
    - DCL (Delay Calculation Language)
    - Complicates delay calculation / timing interface
- **Characterization effort increasing**
  - Need to apply dimensionality reduction methods

Outline

- Snapshots from past Taus
- Delay modeling
- **Timing analysis**
- Timing integration
- Future challenges

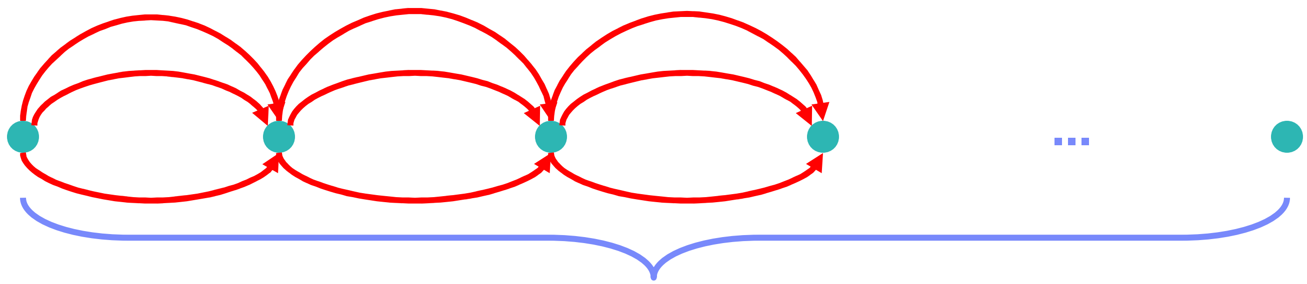
Static Timing

- **Major triumph of static timing analysis**
  - Allows efficient analysis by separating topology from function
  - Avoids exponential blow-up due to sensitization dependencies
  - Requires acyclic timing graph



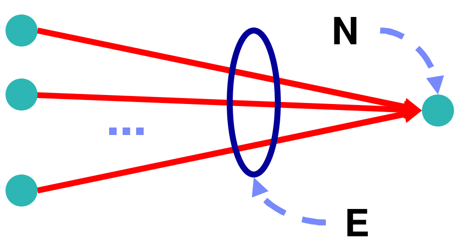
Static Timing – two dominant approaches

- **Path oriented**
  - In pure form can require exponential path tracing



10 stages, 3 reconvergent paths per stage = 30 edges, 59049 paths

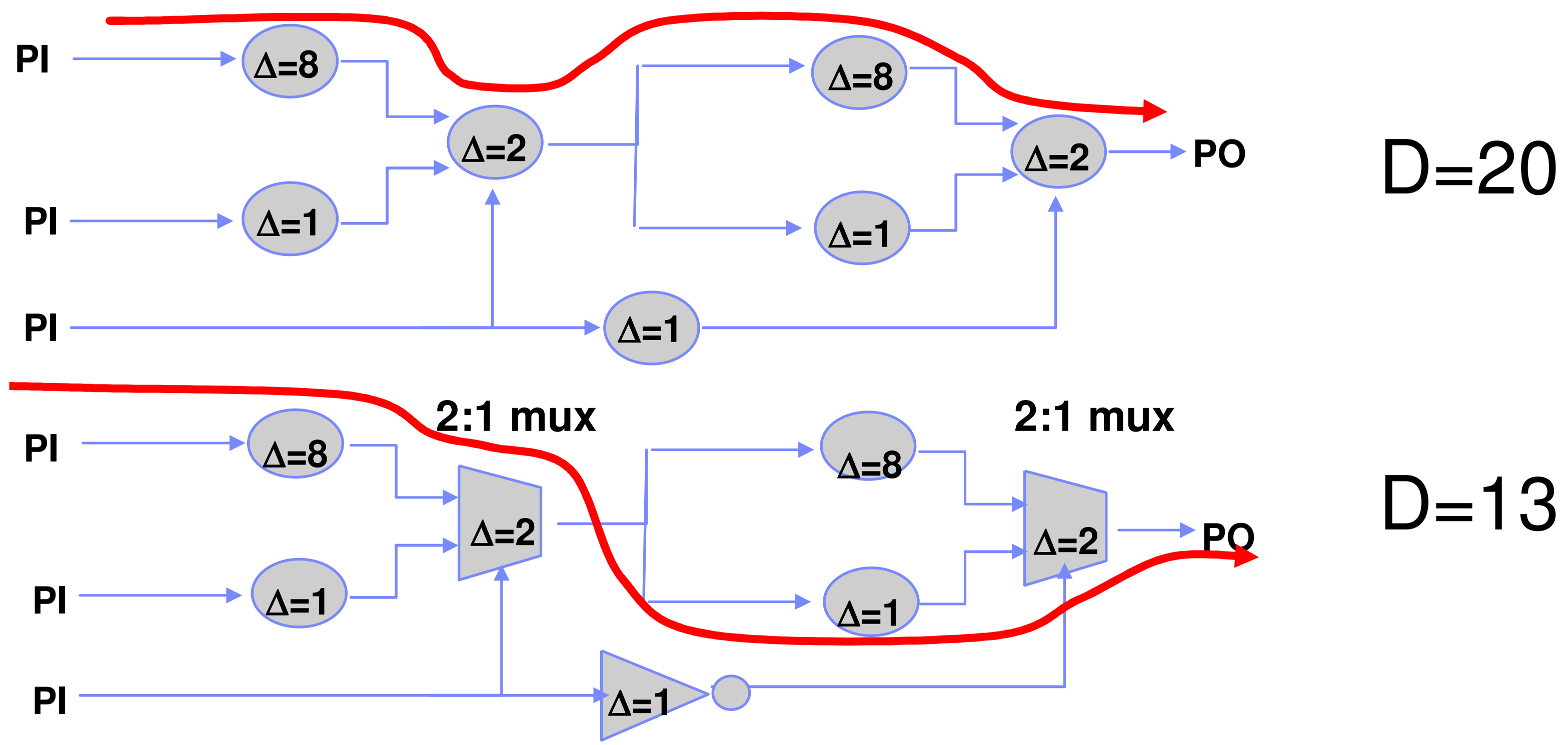
- **Block-oriented**
  - Linear in network size
  - Computes single arrival times (ATs) at each node
  - Usually still can report results in terms of paths



$$AT_{LM}(N) = \max_{E \in \text{in-edges}(N)} \left(AT_{LM}(\text{source}(E)) + \text{delay}_{max}(E) \right)$$

False paths

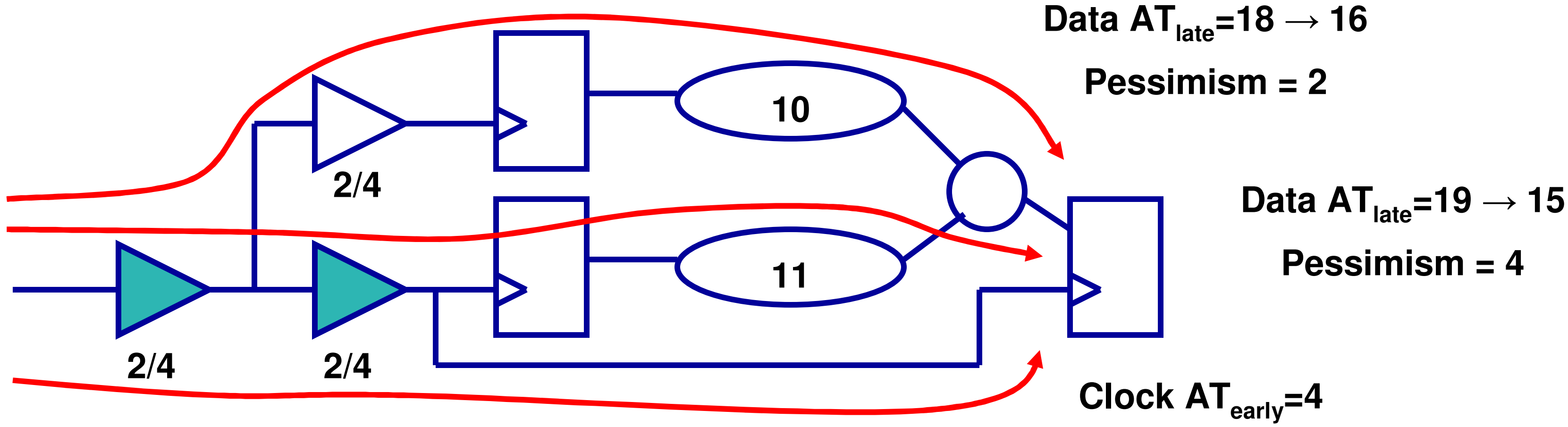
- **Purely topological timing can be pessimistic**



- **Lots of focus on false path identification / removal in 1990s**
  – Found that "hidden" false paths are rare
- **Current approach – false path analysis, not identification**
  – Analyze by creating "copies" of topological analysis (false subgraph)

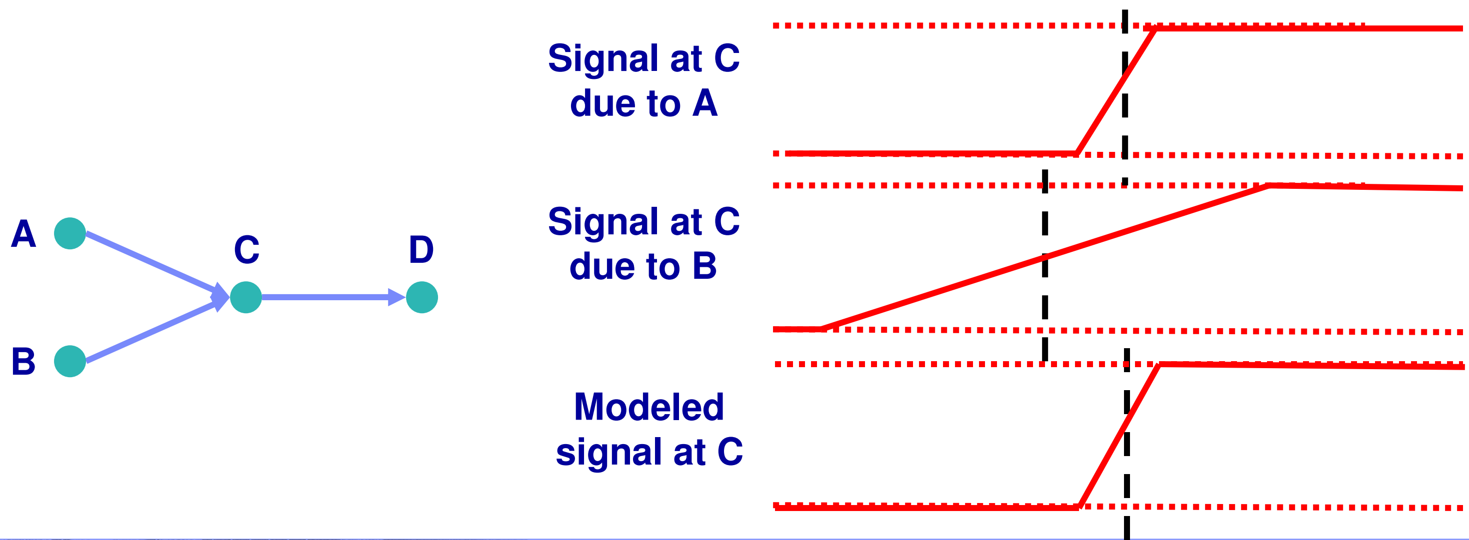
Common path pessimism removal

- **Problem realized once delay variation was considered**
  - ATs in block-oriented analysis “forget” their past
  - Worst early and late paths to a test may pass through common block (generally in clock tree)
- **Solution – selective path tracing**
  - Apply only on “failing” tests
  - May need repeated path tracing



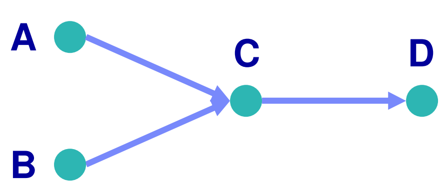
Slew selection

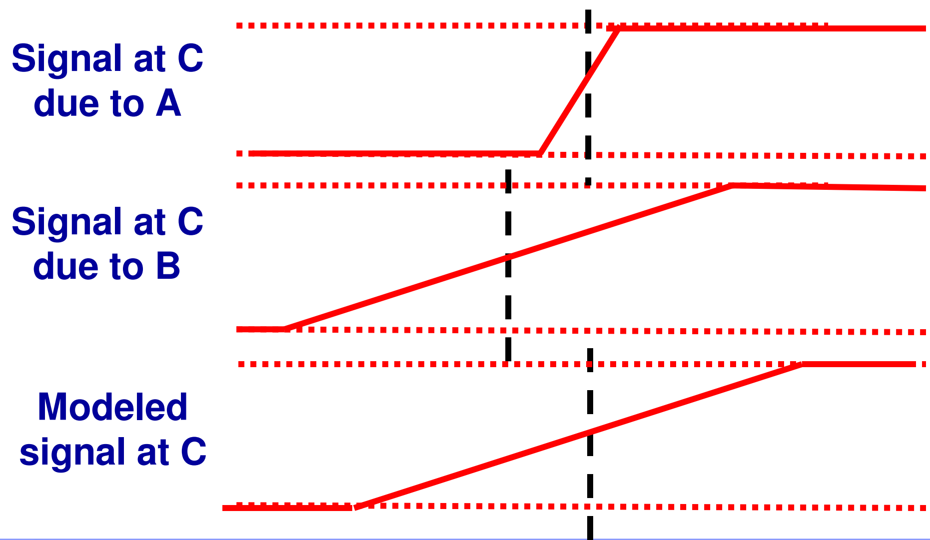
- **Slew depends on path along which signal propagates**
  - Requires integration of delay calculation with timing
- **Various solutions**
  - Choose slew associated with dominant AT – can be optimistic



Slew selection

- **Slew depends on path along which signal propagates**
  - Requires integration of delay calculation with timing
- **Various solutions**
  - Choose worst slew independent of AT – can be pessimistic

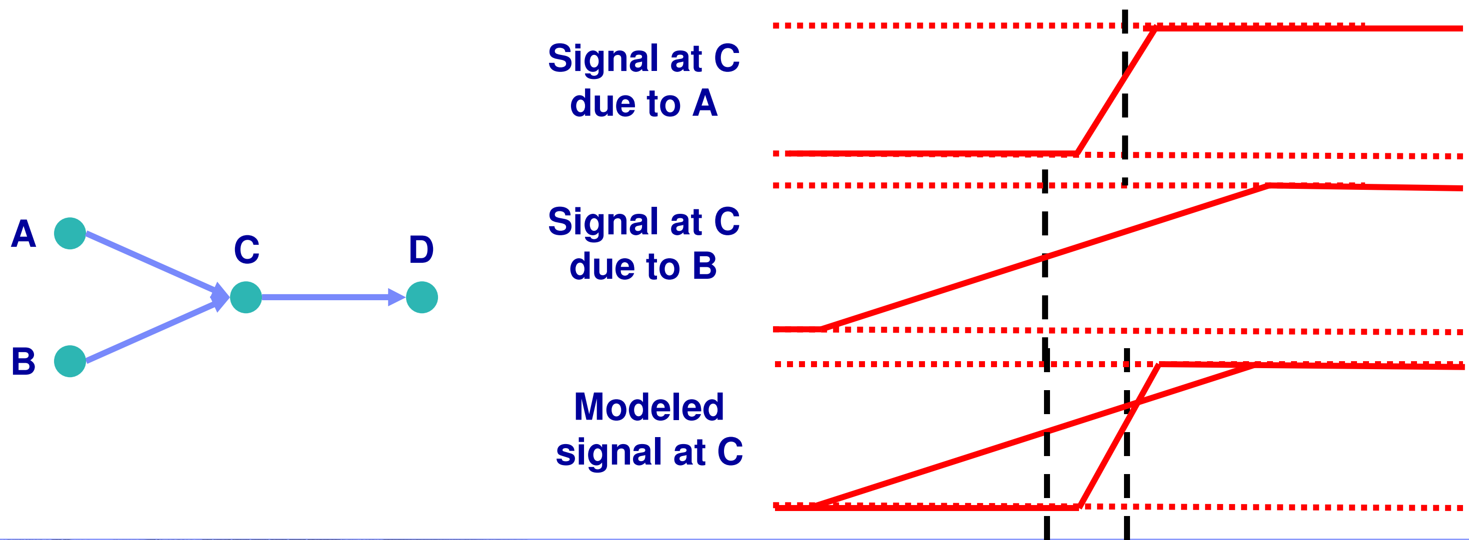




Slew selection

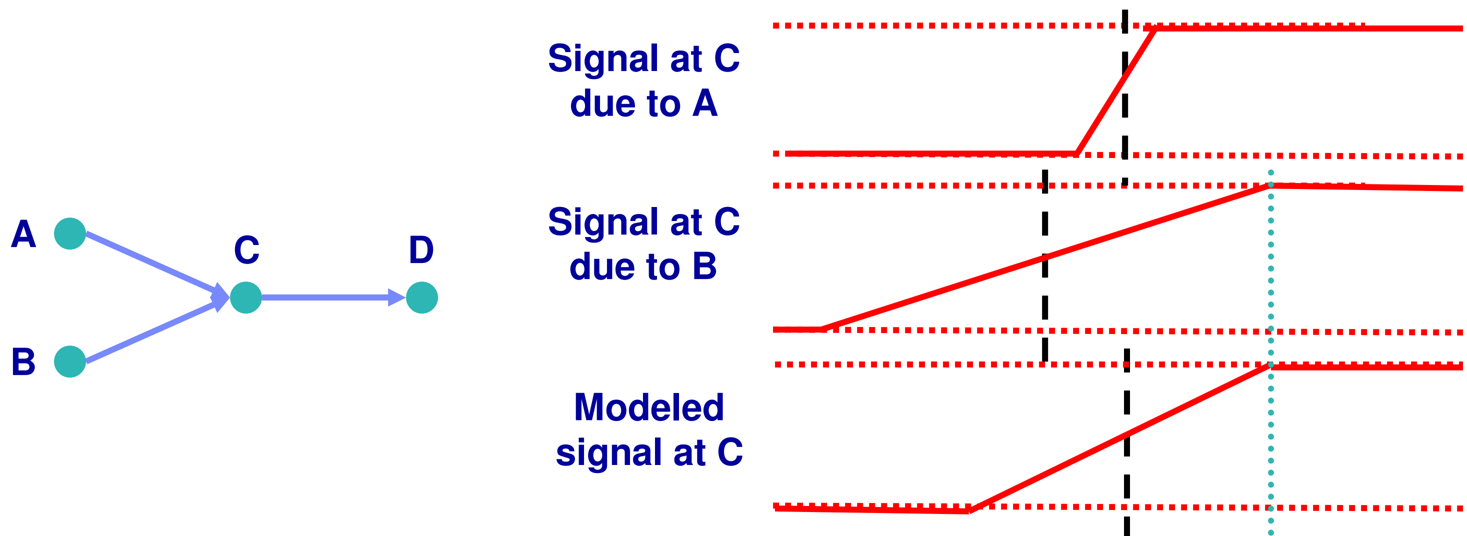
- **Slew depends on path along which signal propagates**
  - Requires integration of delay calculation with timing
- **Various solutions**

  - Carry multiple slews until one dominates – data / computation increase



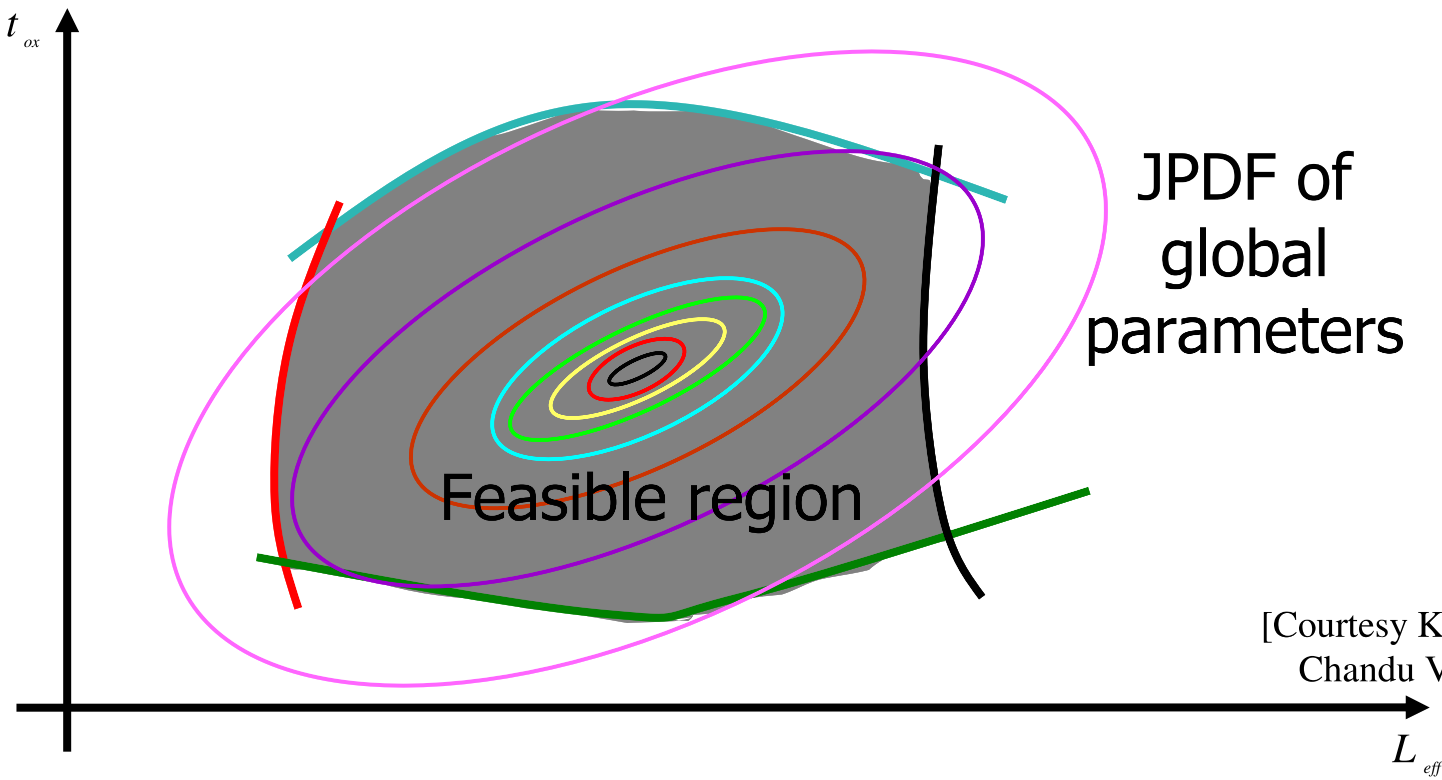
Slew selection

- **Slew depends on path along which signal propagates**
  - Requires integration of delay calculation with timing
- **Various solutions**
  - "Merged" slew – artificial waveform matching worst 50%, 90% points



Statistical timing

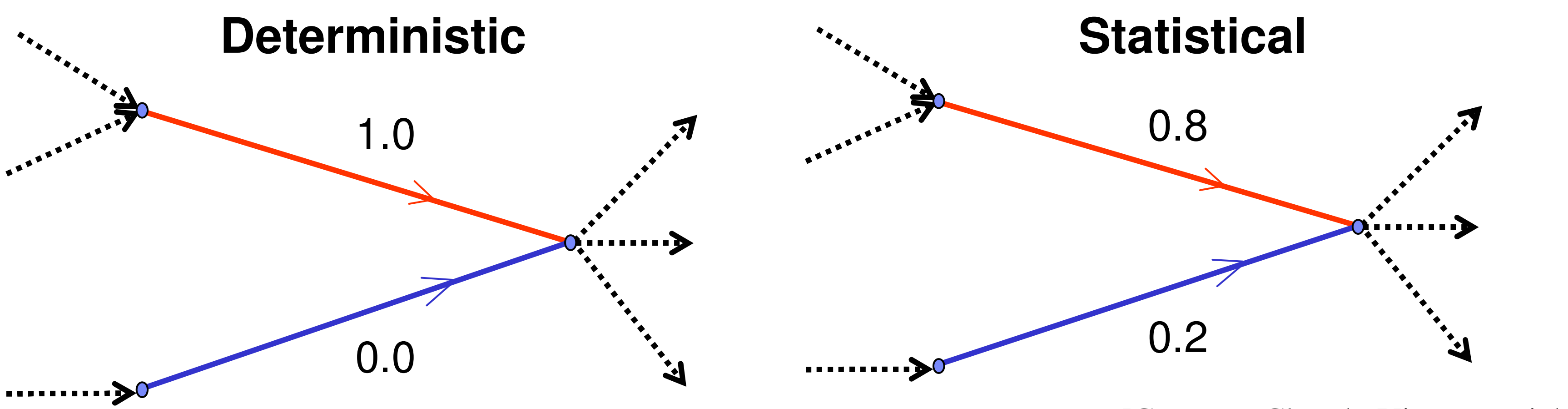
- **New approaches**
  - Parameter space methods
    - Model delays as functions of these statistical parameters



[Courtesy Kerim Kalafala & Chandu Visweswariah]

Statistical timing

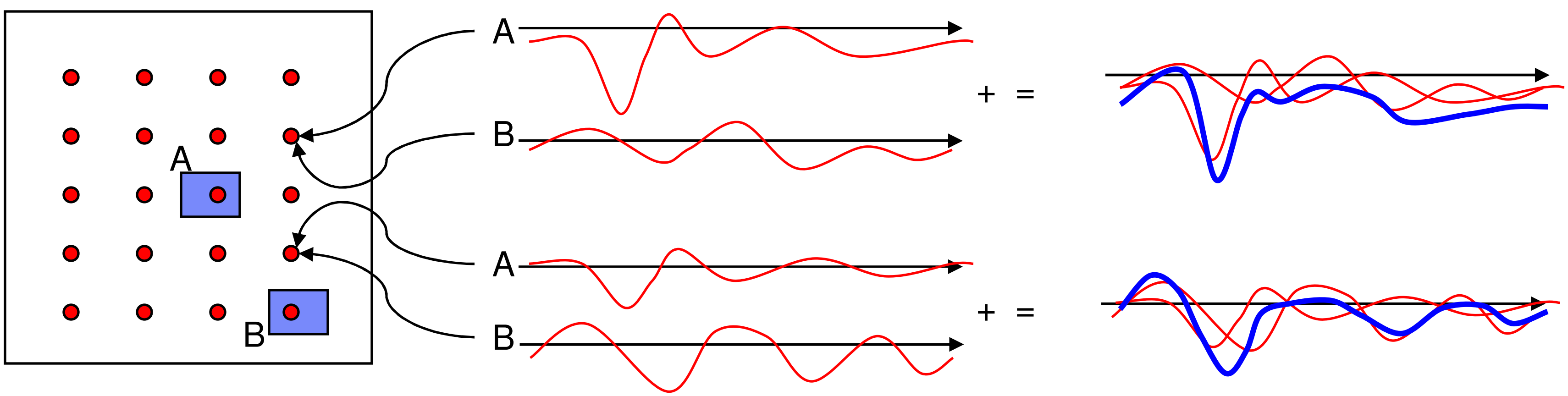
- **Inherent problem with block-based methods**
  - Statistical AT variation depends on path
  - Can use block-based non-statistical method to select paths
  - Block-based statistical methods
    - Approximate actual statistical result by creating “representative” path
    - Estimate dominance *probability* of path (criticality) or edge (tightness)



[Courtesy Chandu Visweswariah]

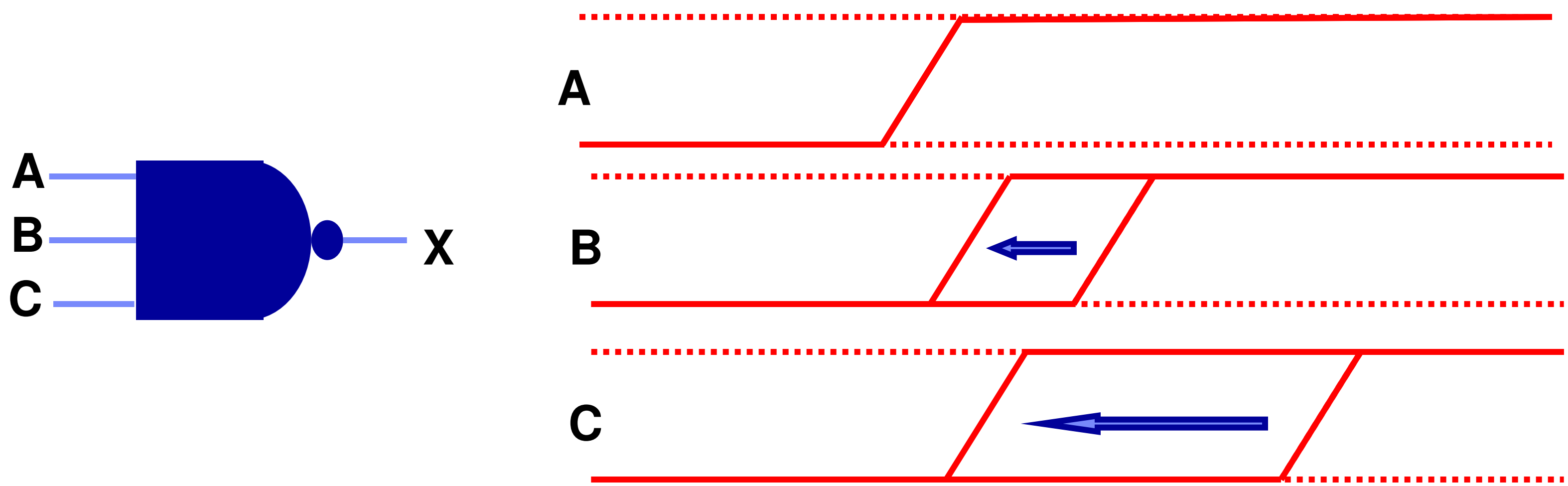
Power supply impacts on timing

- **Hard to determine "worst" condition**
  - Timing tests compare early / late times
    - Worst condition can come from worst noise *difference* on racing paths
  - Transient power supply noise makes this worse
- **Recent methods attempt to cover space**
  - Use superposition to model combined effects of different noise sources at different times
  - Model path delay as function of different noise source activities
  - Use optimization methods to find worst condition



Simultaneous switching

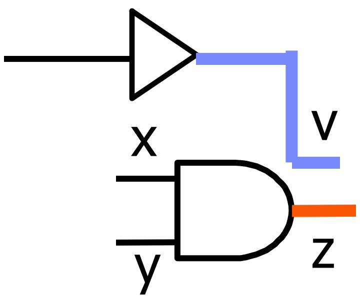
- **Traditionally consider only single input switching**
- **Simultaneous switching becoming more important**
  - Optimization tends to create "slack wall"



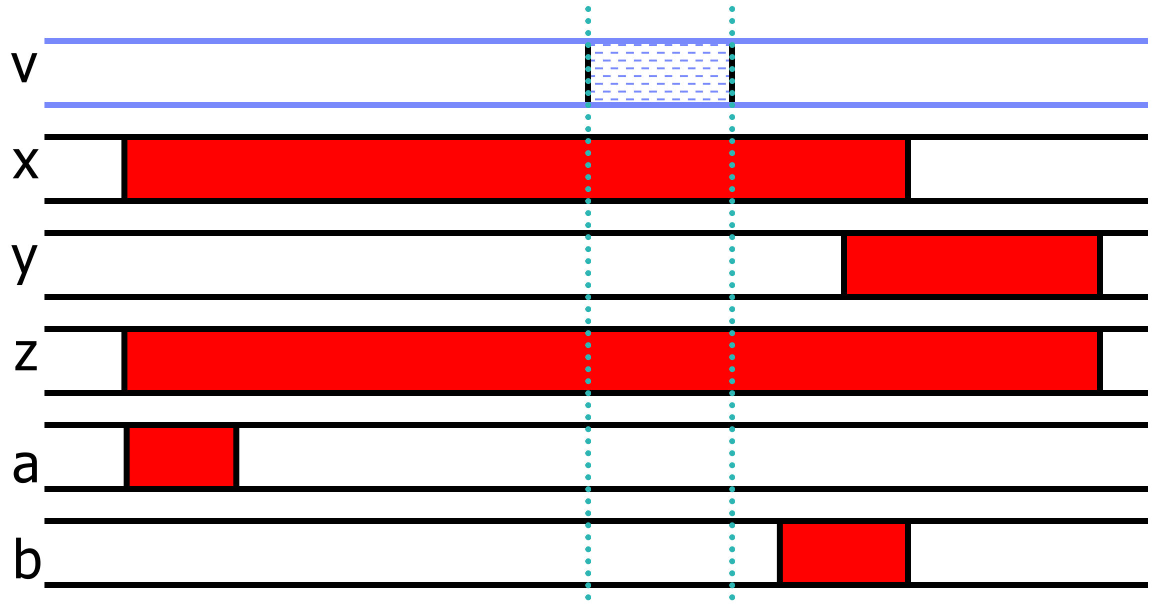
- **Easier in block-oriented than path-oriented analysis !**
- **Increases characterization cost**
  - Grows with number and possible alignments of inputs
  - Easily handled by simulation-based delay calculation

Wire coupling in static timing – aggressor selection

- **Use aggressor timing windows**
  - Complicates timing analysis / delay calculation interaction
  - Can break acyclic timing graph
- **Initially, use single time window per aggressor**

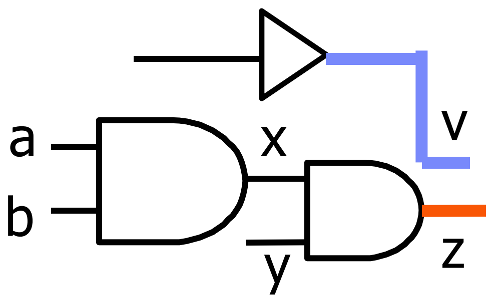


(zero gate-delays)

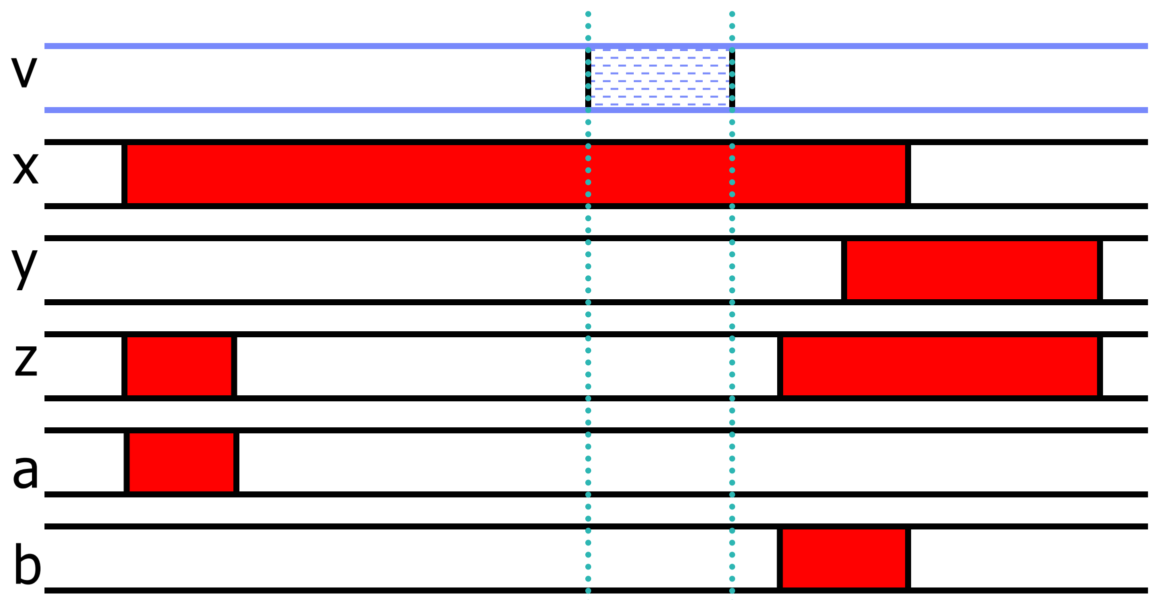


Wire coupling in static timing – aggressor selection

- **Use aggressor timing windows**
  - Complicates timing analysis / delay calculation interaction
  - Can break acyclic timing graph
- **Initially, use single time window per aggressor**
- **Reduced pessimism with multiple windows per aggressor**

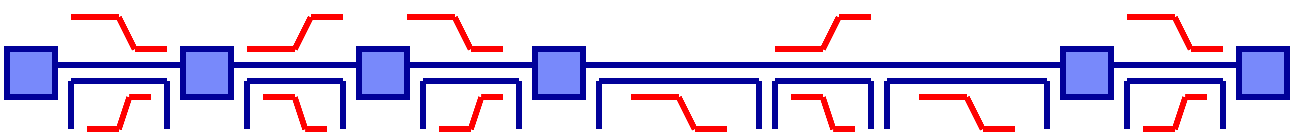


(zero gate-delays)



Should static timing be "safe?"

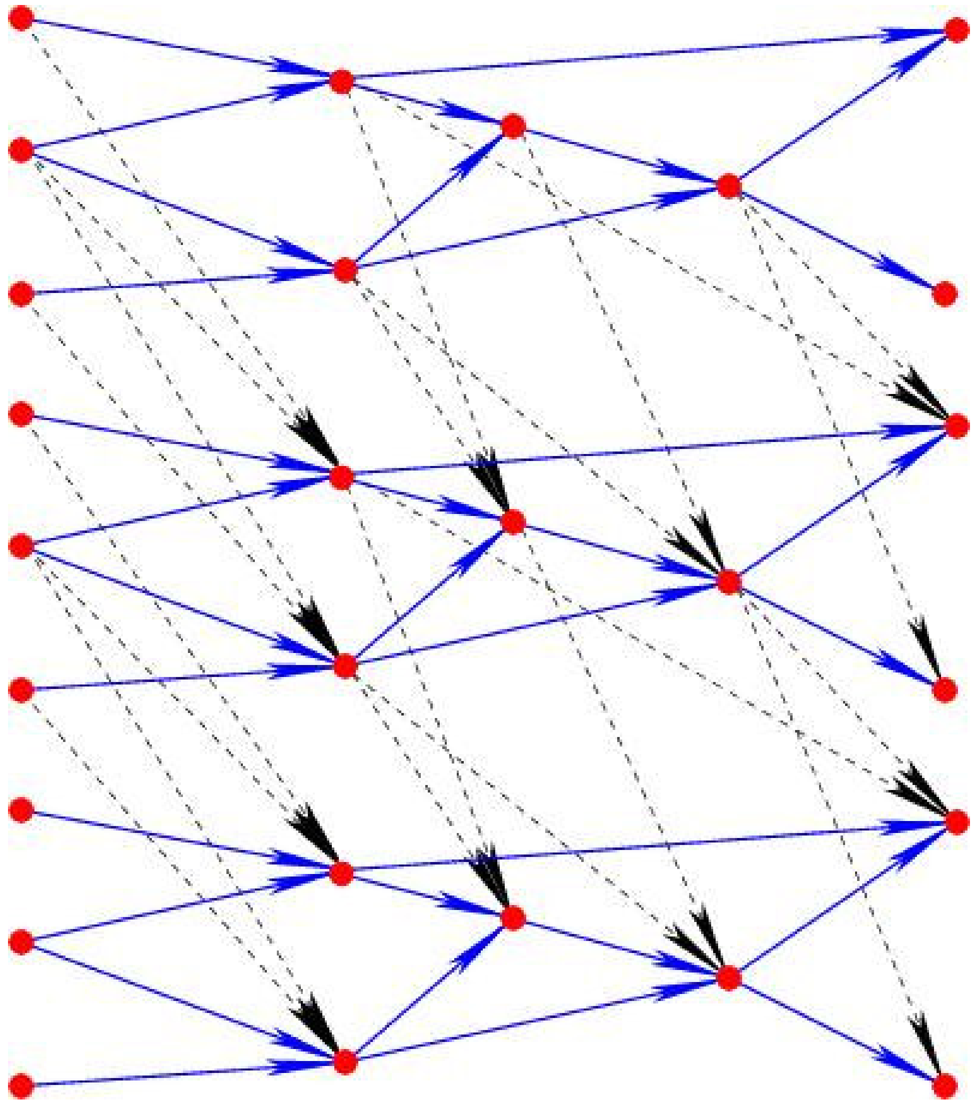
- **For fixed delays, topological analysis guarantees coverage**
  - No pessimism except for false paths
- **But delays depend on things that may not occur often in path**
  - Wire coupling, simultaneous switching
- **Safe approach says assume all bad thing happen together**
  - Every aggressor of every net in path switches in "bad" direction
  - Very conservative



- **Instead – assume some limit on how many bad things happen**
  - Obvious method: path tracing, look at N worst impacts on path
  - Doubly exponential

N-fault timing

- **Turns out we can do this in block-based paradigm**
  - To model N “faults” per path …
  - Create N+1 “copies” of timing graph
  - Add “fault” edges between them
- **Properties**
  - Any path can traverse at most N fault edges
  - Graph contains all paths of N faults



N=2

Outline

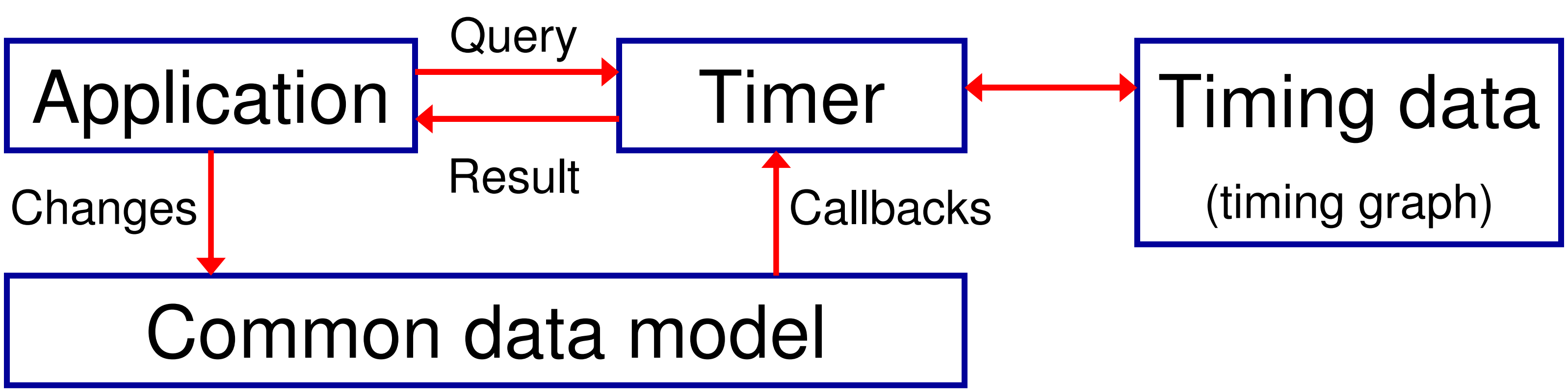
- Snapshots from past Taus
- Delay modeling
- Timing analysis
- **Timing integration**
- Future challenges

Timing integration

- **Why do we need integrated timing analysis?**
  - Timing is complicated
  - Every timing optimization method shouldn't do its own analysis
  - Instead have a timing subsystem
- **Key feature – autonomic control**
  - User of timer shouldn't have to know how it works

Incremental timing

- **What do I mean by incremental?**
  - Keep active timing graph
  - Small design changes → small changes in timing graph values
- **Incremental + autonomic**
  - Requires a common data model w. callbacks
    - Application changes model
    - Timer gets change information of interest from model callbacks



Incremental timing – when to update

- **Change management - obvious approach**
  - Update everything whenever a change is reported
    - Expensive (too much recalculation)
    - Imposes processing order requirements on callbacks
- **Better approach**
  - Only perform invalidation on change report
  - Wait to recompute information until needed

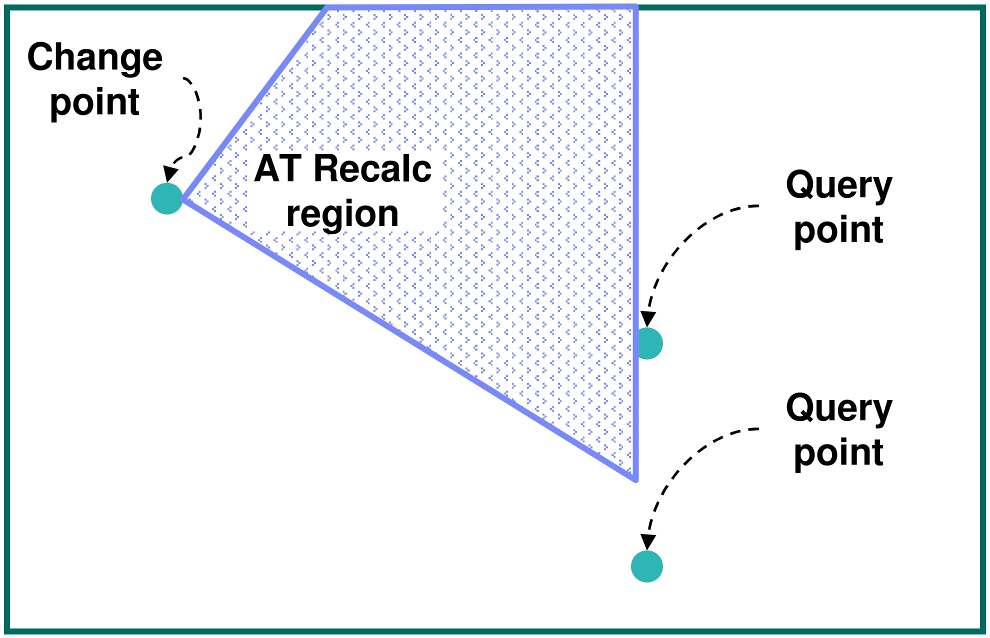
➢ **Lazy evaluation**

Incremental timing – how much to update

- **Simple method**
  - Whenever timing request received, update all affected values
- **Better approach - lazier evaluation**
  - Only update enough to answer the question asked
- **Dominance limiting**
  - Stop propagating when values stop changing
  - Doesn't help much with changes in critical areas
  - Dominance not clear-cut in statistical timing
- **Level-limiting**
  - Propagate changes up to level of query

Level limited incremental timing

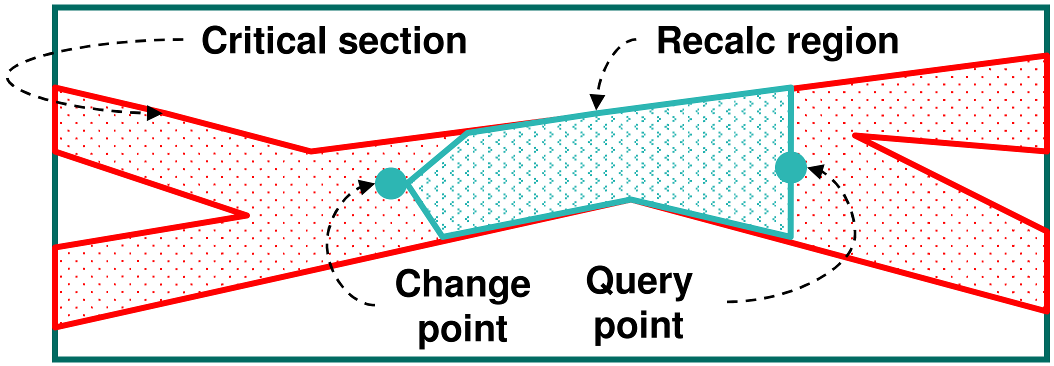
- **Keep levelized list of timing change "frontiers"**
  - On timing request propagate changed values up to request point level



  - Slews, tests, and RATs add complications

Even lazier evaluation

- **Integrated applications generally focus on critical areas**
  – Changes in critical areas tends to propagate everywhere
  – Temporarily limit propagation to "critical section"
    - Not completely safe – critical section can change



  – Keep track of other frontier points for complete update later

Do we really have timing-driven design?

- **No, we have timing-influenced design**
  - Today's timer is still passive.
  - Applications still query timer, but need to know what to ask and where
  - Design change can have unforeseen consequences
    - Change aggressor switching window for coupling
    - Legalization moves stuff
    - Accurate timer understands these interactions better than the optimizer!

Timing-driven design

- **True timing-driven design**
  - Need timer to take control – identify problems
    - Avoid reanalyzing entire design – so don't make optimizers initiate query
  - Report results of *series* of operations
    - Change may be composite
    - Don't accept/reject based on any single step
    - Means that timer must understand and report on "unit of work" between checkpoints

- **Extending to other domains (power, etc.)**
  - Objective-driven design

Outline

- Snapshots from past Taus
- Delay modeling
- Timing analysis
- Timing integration
- **Future challenges**

Asynchronous design

- **Synchronizing clocks across a chip is getting harder**
  - … and more expensive (power, routing)

- **GALS (globally asynchronous / locally synchronous)**
  - Pressure will build to shorten latency across interfaces
  - Will ask new questions of timing

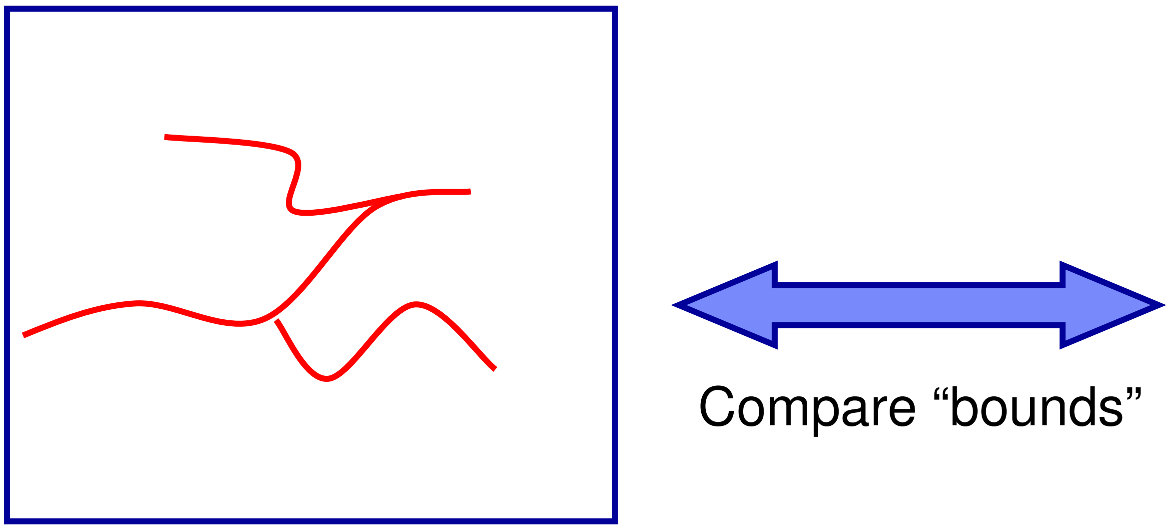
Guide optimization

- **Optimization has *many* options**
  - Have to decide which will be most effective

- **Provide gradients**
  - Don't just say what the slack is
  - Say what it depends on, and how
    - Choice of cells
    - Choice of Vt
    - Choice of metal layers
    - Choice of placement
    - …

Handling variation in everything

- **Continued development of statistical timing**
- **Accurate *relative* statistical timing for adaptive systems**
  Account for process, environment, workload variation



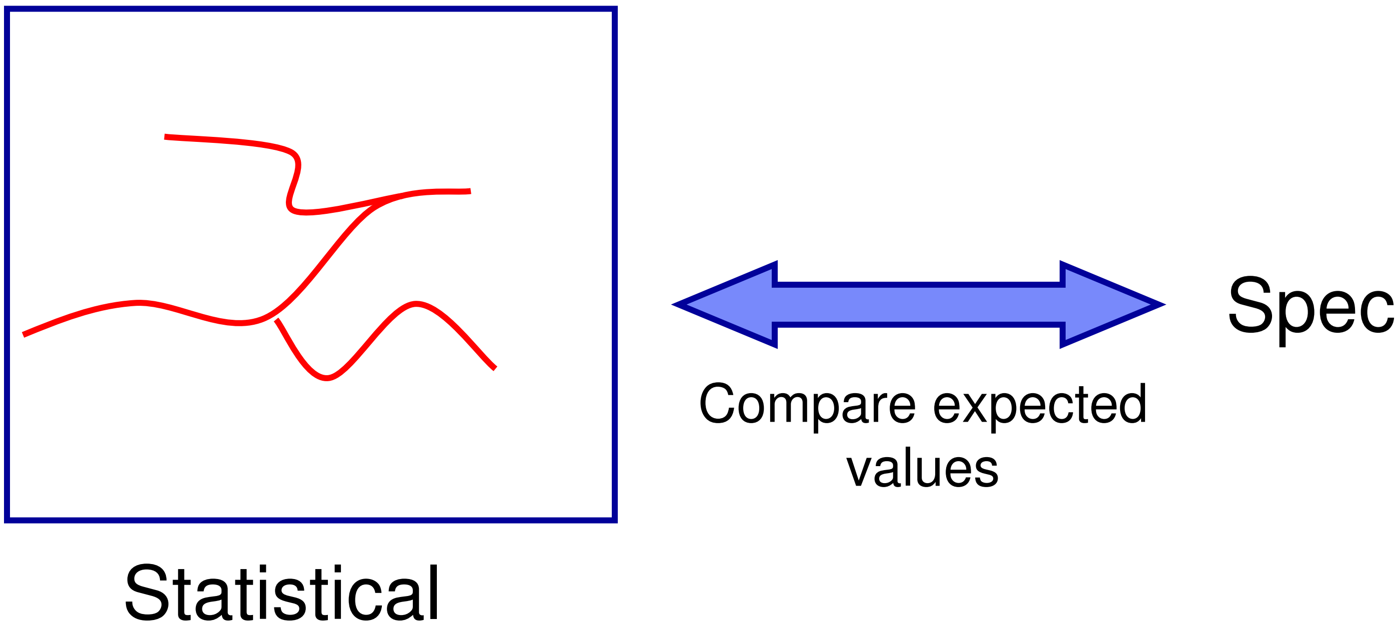
Compare “bounds”

Spec

Deterministic

Handling variation in everything

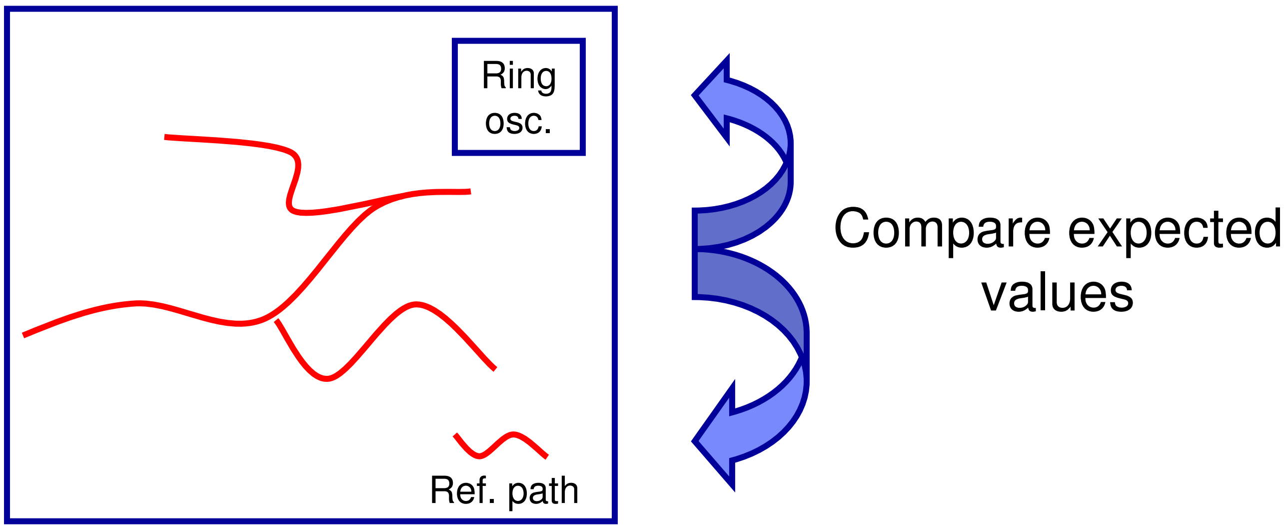
- **Continued development of statistical timing**
- **Accurate *relative* statistical timing for adaptive systems**
  - Account for process, environment, workload variation



Handling variation in everything

- **Continued development of statistical timing**
- **Accurate *relative* statistical timing for adaptive systems**

  Account for process, environment, workload variation



Relative statistical

Find the worst conditions

- **We've been very lucky**
  - Topological timing efficiently bounds performance with little pessimism
    - *…but only for simple delay models & relationships*

- **Bounding in other domains is not so easy**
  - Power supply
  - Activity
  - Process
  - … and these affect timing

- **Use statistical timing**
  - Not all of these are statistical phenomena
  - But use statistical approx. to find important *regions* of condition space

Continue to improve integration

- **Timing isn't the only objective**
  - Other objectives (power, noise) depend on timing
- **Need smooth interaction of integrated incremental subsystems**
  - Provide total picture of design vs. objectives to optimizers
- **Keep incremental analysis close to sign-off analysis**
  - Fails in sign-off timing must be ***very*** rare
  - Productivity needs demand automated design closure

And be careful…

- **New devices, circuits, design styles, & physical effects keep coming**
  - Timing (and other analysis) has to anticipate problems

- **Images of Tacoma Narrows Bridge collapse, 1940**
  - Animation from:
    - http://www.bradford.ac.uk/acad/civeng/marketing/civeng/failtac1.htm
  - Photo from:
    - http://www.scret.org/narrows/index.asp