EE-382M

VLSI-II

Static & Statistical Timing Analysis

Matthew J. Amatangelo, Intel Corp.
Acknowledgements

John M. Keaty
Yaping Zhan
Basics of Static Timing Analysis (STA)

- **Key points of STA**
  - Determines worst case arrival time of signals at all pins of design elements
  - Does not test functionality
    - Does not distinguish between functional and non-functional paths
  - Reduce complexity of analysis to increase volume of coverage
    - Accuracy reduced, not compromised via use of guardbands
  - Underlying assumptions enable STA to produce results
    - Reduced accuracy, ignored connections and effects must be managed
    - Beware of implied synchronicity of cross domain paths
  - Uses ATs and RATs to determine path timing violations
    - Path comprised of launching and capturing components
    - Clocks and sequential elements define RATs
    - Any combinatorial element with clock input is sequential
    - Sorted by capturing clock. Clock phase is important.
  - Uses timing graph of delay arcs and checks to represent the design
    - Particular information is stored at each node
    - Long and short path analysis is performed between source and sink points of graph after ATs and RATs have been propagated through graph
  - Accuracy is only as good as cell timing models
  - Sanity check required – expected results versus actual ones
Basics of Static Timing Analysis (STA)

• Key points of Statistical STA
  – Classical STA must improve by quantifying process variation as an affect on gate and wire delays
  – Classical STA guardbanding becomes too great vs cycle time to meet setup and hold as variations become proportionately large
  – SSTA provides the probability that each path passes over the range of independent variables
  – SSTA points to process parameters that need tweaked:
    • Given the circuit, find sensitivities of process parameter variance versus path delays
  – Components of variance must be carefully considered – PCA
  – Correlation is key in reducing over-pessimism
  – SSTA relies on a great deal of process and circuit analysis
    • Most fabs don’t go beyond standard STA cell libraries per corner
    • EDA industry generally doesn’t have access to detailed process correlation data
Basics of Static Timing Analysis (STA)

• Outline
  – Dynamic versus Static Timing
  – Characteristics of Static Timing
  – Trends: Variable Aware Static Timing Analysis
  – How Static Timing Analyzers Work
  – SSTA
Timing Analysis Techniques

• **Spice for entire networks and/or macro cross-sections.**
• **Dynamic Simulation**
  – **Coverage dependent on quality of the set of input vectors.**
    • circuit delays are state dependent.
  – Examination of logic failures not comprehensive in analyzing problems
    • root cause difficult to determine
  – Determines whether an event will occur.
  – Advantage - Does not time non-functional paths.
  – Disadvantage - How do you know all functional paths were timed?
• **Static Timing Analysis**
  – **Input vector pattern independent**
    • traverses all paths between endpoints
  – Every source of data launch is checked at destinations or sinks
    • min and max delay values saved for each arc
  – Determines the worst possible time an event will occur
  – Advantage - Comprehensive, guarantees all paths are analyzed.
  – Disadvantage - Does not distinguish between functional & non-functional paths.
Static Timing: What Is It?

Inputs:
- Netlist
- Constraints
- Timing Models
- Parasitics
- User commands

Products:
- Readable Slack
- Reports/Histograms
- Design Data: flat netlist, net delay file, timing model, reduced parasitic
- Constraints: arrival times, required arrival times
Timing Analysis Basics & Key Elements

• **What is static timing analysis?**
  
  – **What it is not:**
    
    • Magic -> Test your results against expectations!
    • It does not perform a delay based functional simulation.
    • It does not consider the logical functioning of the circuits (except inversions).
    • It does not analyze individual paths.
    • It doesn’t search or analyze all possible paths (Path Pruning).
  
  – **What it typically does:**
    
    • Calculate latest and earliest possible switching times for each node in the design.
    • Determine the arrival time of signals for the worst case (latest or earliest) of all possible paths leading to a given node in the design.
  
  – Compare calculated signal arrival times with expected (required) arrival times at storage elements, other clock meets data points (such as dynamic circuits) and primary outputs in the design.
Variable Aware Timing Analysis

- Analysis is only as good as the quality of the input data
  - STA Timing models represent one point in Si process space
    - It does not consider variation of delay-critical parameters
    - Same goes for delay-critical parameters of each wiring layer
    - N parameters have $2^N$ corners
      - Exponential STA analysis points to cover process space
      - How to predict yield?
  - Proportion of inter- and intra-die process variation increases with decreasing feature sizes
    - Worst case STA models become too pessimistic
      - Some corners not reachable due to systematic correlations
    - Same goes for temperature and voltage deviations
  - Conclusion: Classical STA must improve by quantifying process variation as an affect on gate and wire delays
A Good Statistical Timer Provides:

- Timing analysis for yield prediction
- Spatial and re-convergence correlations
- Analysis of on-chip and chip-to-chip variations
- Sensitivity analysis for parametric yield optimization
- Built-in variation-aware extractor
- Delay calculation and signal integrity analysis for silicon accuracy
- Monte-Carlo, SPICE, and 3D extraction capabilities
- Process variation modeling capabilities
How Static Timers Work

- Static timing paths
- The Timing Graph – the Timer’s internal map
- Analyses performed
  - Late mode (“max mode”)
  - Early mode (“min mode”)
- Timing graph attributes: ATs, RATs, and Slacks
- Paths: STA versus SSTA
- Clock considerations
- Reporting
- Models
- Special considerations and limitations
- SSTA
How Static Timers Work

- Static timing analysers build a graphical representation of the logical and circuit structure of the chip.
  - The arcs of the graph represent gates and wires in the design and carry delay and slew information.
    - Termed propagation arcs, they typically have this information encoded as equations.
  - The nodes of the graph represent pins on blocks, ports, convergence points of multiple signals, and places where clock meets data.
    - Storage elements and dynamic circuit nodes appear as nodes in this graph.
    - These nodes carry the test required to be performed when clock meets data or whenever event order is important.
    - These nodes also are used to accumulate arrival times (AT’s) and required arrival times (RAT’s).
How Static Timers Work (cont.)

- For timing analysis, an acyclic graph is required.
  - Timing analysers snip all loops of timing arcs in the graph.
    - In transparent latch designs there are frequently many more loops involving multiple latches in the path.
    - Note to self: When loops are broken, there is an implied assumption.
  - All events occur within one cycle of the defined simulation clock
    - adjustments made to arrival times to maintain single cycle context
- The acyclic graph is then traversed from PIs and latch points (and other places loops were snipped) to POs and latch points in a single pass.
  - ATs and RATs are accumulated at every node during this traversal of the graph.
    - Slews are also calculated.
    - These times are calculated for all signals, clock & data.
- The timing analysis occurs when the tests that are resident at each node are applied to the AT information collected from the graph traversal.
  - This is then analysed, sorted, and written out as reports.
A Note about Assumptions

• When loops are broken, there is an implied assumption by the timer
  – Is it valid?

• Asynchronous paths in STA
  – No paths are considered asynchronous unless specifically delineated as such – false synchronicity
    • Clock definitions are considered absolute and a rational number multiplier will be computed and used in slack calculations
    • Undermines coupling analysis by assuming aggressors can only affect victims in synchronously derived windows

• You must understand all timer log messages to right any implied assumptions!
  – Examples:
    • Couldn’t compute delay, so assumed 0
    • Found non-unate logic in clock trace, assuming even logic (r->r)
    • Object is not a valid endpoint, so not tested
    • Cell timing model characterization usually has assumed particular sensitizations don’t occur or always occur
Static Timing Paths

• Paths begin at source pins
  – primary inputs (this may be a clock input)

• Paths end at sink pins
  – primary outputs
  – wherever timing checks are performed
    • storage elements/timing elements
    • timing elements include clock gates
  – where timer breaks loop paths

• Paths represented as a series of segments in a timing graph
  – logical constraints essentially omitted
The Timing Graph

• Arcs
  – represent the min and max input to output propagation delays and appropriate slews for wires and logic blocks
    • provided in timing model per logic cell
    • calculated for wire and to consider input slew, load, noise – normally from 50% input voltage level to 50% of output
  – exist for each appropriate edge
  – uncontrolled or unconditional arcs
    • input signal influences output response directly regardless of any other signal’s state
      – example: clock -> Q of latch
      – example: in -> out of unclocked combinatorial circuit
  – controlled or conditional arcs
    • input signal can only influence the output if the control signal is in a given state
      – example: data -> Q of latch; control signal “enable”
The Timing Graph (cont.)

- Checks (tests)
  - exist whenever a signal arrival time is constrained by another signal's arrival (usually a clock)
  - where they come from:
    - timing model (explicit)
    - timing checks are implied when clock and data signals meet
      - gate timer may provide checks when not defined in model
        - e.g., clock gates
  - circuit functionality determines the checks – these are identified in the graph as timing elements
    - clock gate
    - latch
    - flip flop
    - domino
    - whatever
Example Timing Graph

Arc broken by clock tracing.
Test added by clock tracing.
Late Mode Analysis

• **Long path analysis**
  – verify signals don’t arrive too late
    • Compares latest possible data arrival versus earliest required time
    • Answers: What is the max frequency of the design?
  – “set up time” or “long path” or “late mode” or “max mode analysis”
  – Often conducted at typical process corner under worst PVT conditions for yield considerations

• **Validate registration**
  – checks that data are set up before the required time
    • Usually the required time is determined by a clock
  – launch and capture events are derived from different real time edges of the master clock
    • slowing cycle time remedies these violations
      – This is not true for the rare zero cycle setup check
Late Mode Analysis

---

Diagram:

- clk
- clkb
- sig
- D Q
- EN

Waveforms:
- clk
- sig
- clkb

Setup test
Early Mode Analysis

- **Short paths analysis**
  - Verify signals don’t arrive too early
    - Compares earliest possible data arrival against latest required time
    - Identifies potential race hazards in a design, usually between clock and data signals.
  - “Hold time” or “short path” or “early mode” or “min mode analysis”
  - Often conducted at best process corner under best PVT conditions

- **Validate registration**
  - checks that data held long enough at destination circuit
  - Generally same-edge test
    - Master clock edge causes both capture clock to close and new data to launch
    - can’t slow cycle time to remedy these violations
Early Mode Analysis

![Diagram of early mode analysis with inputs and outputs labeled as clk, sig, clkb, and DQ with EN. A hold test is indicated.]
Static Timing: What’s It All About?

- ATs
- RATs
- Slacks
Static Timing Analysis Example

Arrival time propagation
(earliest, latest)
Assume AT(rise)=AT(fall)
and Rise delay = Fall delay

Start Here

Propagate ATs

0,1
1,2
1,3
0,5
2,3
2.4
3.5
2.3
2,5
1,2
2.5
3,7
1,2
3,6
2,8
3,12
3,7
5,11
3,6
4,12
3,12
3,7
Static Timing Analysis Example

Required time propagation (earliest, latest)
Assume RAT(rise) = RAT(fall)
and Rise delay = Fall delay

Propagate RATs

Start Here
Static Timing Analysis Example

Slacks and critical paths
(earliest, latest)
slack(rise)=slack(fall)
STA vs. SSTA

- **Deterministic**

- **Statistical**

Chandu Visweswarah, IBM Thomas J. Watson Research Center
Statistical STA

• Built on top of STA
• Handles correlations due to re-convergent paths
  – Assuming random process variation (R) is tracked

• Handles spatial correlations
• Statistical sum and max operations

• First-order parameterized sensitivity propagation

\[ A = A_0 + \sum A_i \Delta p_i \]

Where \( p \) includes both systematic and random variation

Mustafa Celik, Extreme-DA
Pre-Clock-Discussion Summary

• Paths
  – Collection of a series of arcs
  – Created from recursive search of AT and RAT propagation of the timing graph
    • Non-extreme ATs and RATs are pruned

• Arcs
  – Delay and slew are attributes
  – Delays and slews are calculated for wires and gates

• Nodes
  – Contain ATs, RATs, and slack per edge (rise and fall)

• SSTA
  – Propagates arrival time probability distributions rather than absolute ATs and should show the probability of path violations (and via sensitivities, how the situation can be improved)
Clock Considerations

- Special signals in STA
- Clock phases and their propagation
- Slack calculations
- Required set up and hold time of sequential devices
Clocks: Those Special Signals

- Clocks are propagated from the primary inputs defined with clock waveforms
  - clock “flag” propagates through leaf cells unless the entering pin is determined to be a clock pin
    - implies that the output is not clock, but data
    - examples: flip flops, latches, domino blocks
- Implicate RATs (example to follow)
  - Every gate where clock meets data, a timing check is performed
  - Combinatorial gates with a clock and data inputs normally become clock gates – their output is also a clock
    - Clocks are causal – combining clocks at a combinatorial gate requires a priori relative arrival time knowledge
    - Clock gate output edges are only allowed to be caused by clock edges
- Much of the clock network is not seen by the static timer (e.g., PLL, major trunks)
  - detailed circuit analysis will provide set of arrival times, slews, uncertainty as “givens” to the logic design
  - Clock team would provide
    - SSTA: clock team would need to perform enough analysis to provide AT pdf for each clock
Footnote on Arrival Time Propagation of Clocks at Combinatorial Elements

[Warning: Combining Clock Signals]

<table>
<thead>
<tr>
<th>Signal</th>
<th>AT(min rise, max rise)</th>
<th>AT(min fall, max fall)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data1</td>
<td>2,6</td>
<td>2,3</td>
</tr>
<tr>
<td>Data2</td>
<td>3,8</td>
<td>2,4</td>
</tr>
<tr>
<td>New_Data</td>
<td>2,8</td>
<td>2,4</td>
</tr>
<tr>
<td>Clock1</td>
<td>0,0</td>
<td>5,5</td>
</tr>
<tr>
<td>Clock2</td>
<td>2,2</td>
<td>7,7</td>
</tr>
<tr>
<td>New_ClockA</td>
<td>0,2</td>
<td>5,7</td>
</tr>
<tr>
<td>But you wanted this</td>
<td>2,2</td>
<td>5,5</td>
</tr>
<tr>
<td>Clock3</td>
<td>0,0</td>
<td>5,5</td>
</tr>
<tr>
<td>Clock4</td>
<td>8,8</td>
<td>3,3</td>
</tr>
<tr>
<td>New_ClockB</td>
<td>0,8</td>
<td>3,5</td>
</tr>
<tr>
<td>But you wanted this</td>
<td>0,0</td>
<td>3,3</td>
</tr>
</tbody>
</table>

But you wanted this

Pulse shaper

One-shot
Clock Phases in STA

- Each timing event is associated with a clock phase
  - Clock phase(s) of input boundary signals are set by the user in arrival constraints
  - A clock phase is propagated with each timing event
  - Clock phases are used to perform "cycle accounting": all timing events are "modulo" the cycle time of the clock, accounting is based on phase time.
  - Data pins may contain arrival times from more than one clock domain.
Clock Phase Propagation

• Combinatorial non-clock arcs
  – Input phases propagated to output
  – Different phases do not get combined

• Sequential arcs
  – Output phase determined by timing event at clock pin
  – How the timing event affects the output phase is built into the model
  – Event time is adjusted as necessary when changing phase, based on cycle accounting rules
Example: Clock Phase Propagation

master clock waveform: mclk

Signal phases:

Boundary constraints:
- clk is clock port for mclk
- a arrival phase mclk \uparrow
- b arrival phase mclk \downarrow
- d \uparrow, d \downarrow: mclk \uparrow
- e \uparrow, e \downarrow: mclk \uparrow
- f \uparrow, f \downarrow: mclk \downarrow
- g \uparrow, g \downarrow: mclk \downarrow
- j \uparrow: mclk \downarrow
- j \downarrow: mclk \downarrow
- c \uparrow: mclk \uparrow, mclk \downarrow
- c \downarrow: mclk \uparrow, mclk \downarrow
- clk \uparrow: mclk \uparrow
- clk \downarrow: mclk \downarrow
Clocks, ATs, RATs, and Slacks

- Trace the clocks
- Propagate the clocks and data arrival times
  - Clock arrivals will establish RATs at data inputs of blocks.
  - For controlled arcs:
    - earliest AT at endpoint of controlled arc always dictated by controlling signal
    - late mode: If AT(launching clock max) > AT(data max), ignore flushing
    - early mode: ignore ATs for flush paths completely (min, max)
- Examine checks
  - Output constraints
  - Setup/Holds (timing element dependent)
    - e.g., clock gate setup checks: gating data must arrive before the asserting edge of the clock (for Nands this is the rising edge of the clock, for Nors it’s the falling one)
    - e.g., latch setup: data must arrive by capture clock edge (end of flush mode)
- Propagate RATs
  - backwards through propagation arcs, as in prior non-clocked example
    - RAT(any node) always comes from a requirement downstream in the design
      - RATs are propagated upstream from primary outputs (explicit constraints) or from places where clock meets data (implicit constraints). See example.
Late Mode Slack Calculation

• Late mode slack = RAT - AT
  – AT is the latest time the data event can occur
  – RAT is the earliest requirement for the event to occur
    • RAT = AT(early reference) - u - su + adjust (if the requirement is clock derived)

• Required element setup time (su)
  – determined from spice simulations (more discussion later)

• Clock uncertainty (u)
  – takes into consideration clock skew and other effects that are not being simulated during static timing analysis

• Cycle time adjustments
  – adjusts are performed by the static timer in order to relate the correct data with the appropriate clock event; static timers analyse in “modulo” master clock cycle mode
    • e.g., latches: when the same master clock event causes both the launch of data and its capture, then the adjust = 1 clock cycle
    • e.g., clock gates: when launch happens after capture, adjust = 1 clock cycle
Late Mode Analysis

Earliest Reference Arrival Time
Latest Data Arrival Time

setup slack
su u
Cycle Adjustments for Tests

master clock waveform: mclk

Boundary constraints:
clk is clock port for mclk
c arrival phase mclk ^

Setup check cycle adjusts:
c: mclk ^ against mclk ^: +1
e: mclk ^ against mclk v: 0
f: mclk v against mclk ^: +1

Hold check cycle adjusts:
c: mclk ^ against mclk ^: 0
e: mclk ^ against mclk v: -1
f: mclk v against mclk ^: 0
Early Mode Slack Calculation

- Early mode slack = AT – RAT
  - AT is the earliest time the event can occur (that means no flushing)
  - RAT is the latest requirement for the event to occur
    - RAT = AT(late reference) + u + h + adjust (if the requirement is clock derived)
    - e.g., reference edge for latch is same as in late mode; clock gate must be checked such that data is held beyond the de-asserting clock edge (falling clock edge to a nand/and)

- Required element hold time (h)
  - determined from spice simulations (more discussion later)

- Clock uncertainty (u)
  - same idea as in late mode except it will probably be gauged against a different design point (faster process and conditions)

- Cycle time adjustments for hold slack
  - again, relate correct data with appropriate clock event
    - e.g., latches and clock gates: when the same master clock event causes both the launch of data and the reference checking edge, then the adjust = 0 clock cycle
  - Must be done carefully – avoid “timing escapes” for odd/wrong designs
Early Mode Analysis

 clk → D EN → Q → sig → D EN → clk

 clk
 sig
 clkb

 Latest Reference Arrival Time
 Earliest Data Arrival Time

 hold slack

 u h
Summary of Clock Considerations

- Presence of a clock means events must occur in a particular order
- All clock nets are identified through clock tracing
- When these clocks arrive at gates with data, timing checks are performed to verify proper data arrival time
- Combine clocks at your own peril
- Each arrival time entry of a node has a clock phase associated with it
- In the slack calculation the RAT is determined in part by the clocks arrival at the test point
How Static Timers Work

- Static timing paths
- The Timing Graph – the Timer's internal map
- Analyses performed
  - Late mode (“max mode”)
  - Early mode (“min mode”)
- Timing graph attributes: ATs, RATs, and Slacks
- Clock considerations
  - Reporting
    - Models
    - Special considerations and limitations
  - SSTA
Reporting

• Variety of report types
  – Instance reports
    • Provides ATs, RATs, corresponding phases, slews, and other node information for every port of instances
    • Enables discovery of unwanted ATs and missing data
  – Path slack report
    • Shows progression of signal through each node of the graph between its launch and capture points including these node attributes: cumulative delay, clock phase, net name and edge direction, slew, capacitance
    • Normally organized in order of largest violations
    • Usually provides downstream path violations that are independent of upstream problems – “Re-Launching”
    • STA adjusts arrival times to be modulo the clock cycle -- “Cycle Accounting”
  – Slack histograms
    • Shows slack distribution
  – STA run errors and warnings
• SSTA provides
  – Probability that each path passes over the pvt range
    • System reliability is as good as the worst path
  – Points to process parameters that need tweaked
    • Find sensitivities of parameter variance vs path delays
    • Large sensitivity coefficients means greater impact on delay
  – Most critical gates/interconnect
Timing Models

• **What they are**
  – **Models represent the timing for a particular block of logic**
    • Reduced data set specifying only what the higher level of analysis needs to experience from this logic block
    • Normally separate model exists per logic block for both max and min modes
    • Programming languages exist to describe complex timing behavior of almost any circuit (e.g., TLF, DCL = Delay Calculator Language, an IEEE Standard).
  – **They define the delay arcs between nodes**
  – **They define the tests at nodes (including the margins) and the reference node and required setup or hold times**
    • Should delineate the operating frequency of the logic (pulse widths on clocks)
  – **Model types**
    • **Black box**
      – Appropriate for flushless designs – test points at model boundary
      – Model only combinatorial paths and paths to and from hard timing boundaries (flip flops and clock gates)
      – Unfortunately internal reg to reg paths are not represented normally
    • **Gray box**
      – Appropriate for latch and domino designs
      – Includes internal test points
Timing Models (cont.)

• Where the numbers come from
  – Early design phase: estimates, scaled delays
  – cell based: models created from simulation
    • delays and slews are modulated by input slews and output loads
    • accuracy degraded because any path is comprised of multiple cells, each
      having an accuracy tolerance
    • the broader the range of electrical tolerances, the greater the error
  – custom or true path analysis: models created by transistor timing tool
    • accuracy greater than cell based approach because each channel
      connected component (CCC) is simulated to the actual in-circuit electrical
      conditions
      – except for the pins exposed to the block’s boundary
      – Often represented with tables
      – Can include simultaneous switching routinely
    • can be extremely time consuming to create models for blocks

• Delay, Slews, and Required Setup/Hold Times (elements where clock meets data) are appropriate for a particular PVT (process, voltage, temperature) corner
  – SSTA would alleviate need for multiple model corners
Timing Models (cont.)

• Models are characterized for a range of circuit conditions
  – Input slews, output loads
  – Never use the model outside of these ranges (STA warning)

• Custom Logic Macros
  – Transistor level timers perform STA and timing rule generation.
    • Black & Gray Box models.

• Synthesized logic
  – Standard Cell Libraries typically come with timing rule libraries in a variety of industry standard formats.

• Embedded Arrays (SRAMs)
  – Spice analysis of hand generated array cross sections is still common.
  – Transistor level timers such as PathMill and Dynacore have abstraction and modeling capabilities that enable them to do timing analysis and timing rule generation on a wide assortment of embedded array array macros for Gray Box modeling.
Timing Models (cont.)

- **SSTA cell models**
  - Add parameter fluctuations to independent variables of CL and input slew rate
    - Params could include $V_t$, $L_{eff}$, etc, or be based upon principal component analysis (derived to cover variance)
      - PCA makes all variables orthogonal to each other

- Outputs both delay and slew with mean and variance
  - PDF options include linear relationship with parameters (assumes pdf always Gaussian) or quadratic (etc) form to achieve better approximations to cover when inputs have vastly differing distributions
Special Considerations

• **Path considerations (See Extras)**
  – Tests, end points, and path enumeration
  – Loops must be broken – what happens

• **Problems with static timing**
  – What can happen with arrival time propagation
  – Worst case delays (min and max) -- simultaneous switching and contention
  – Parallel drivers

• **Wire delay**

• **Special path control**

• **Skew reduction**
What Can Happen with AT Propagation

- STA keeps min and max AT for each edge/phase.
  - However, the output slew usually is that corresponding to the particular AT
    - Could miss the true worst case path, particularly in max mode
    - Breadth-first: prunes away all non-extreme arrival times at the output node
    - It's possible that one of the non-extreme ATs had a much worse slew than the extreme AT as in Net1 -> Net3 below. This gets omitted because from the subsequent delay calculation (for Inst11)
  - Some timers allow propagating worst slew or some composite of worst AT and “nearby” worse slew

<table>
<thead>
<tr>
<th>Node</th>
<th>AT/slew</th>
<th>Net3 result</th>
<th>Net4 (STA)</th>
<th>Net4 (depth-first)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net1</td>
<td>150/300</td>
<td>180/220</td>
<td>-----------</td>
<td>210/150</td>
</tr>
<tr>
<td>Net2</td>
<td>160/130</td>
<td>182/135</td>
<td>195/100</td>
<td>195/100</td>
</tr>
</tbody>
</table>

Net1 — where upstream selection causes a miss in producing the worst arrival time
Simultaneously Switching Inputs

• For multi-input gates, if 2 or more data signals switch together the delay will differ from that calculated when switching is monotonic
  – Series FETs will have slower response
  – Parallel FETs will have faster response

• Problem: Timing models only store delay info to calculate AT from an input despite what other inputs do
  – If the model is built under simultaneously switching (ss) inputs, then it's over-pessimistic for monotonic (usual) cases
  – ..or it is optimistic if built without ss for the unusual case

• Similar problem exists when contention occurs between two signals – the actual delay is slower (max mode) than what STA predicts
Parallel Drivers

• Problem: STA path analysis not suitable for calculating correct timing when blocks are shorted
• Some timers have provision to address this, though not very well
  – Input AT differences ignored
  – Current capacity per driver may be estimated
• Be careful. Simulate such situations in spice.
  – It may be better to add user AT constraint at common outputs
Wire Delay Consideration

• Dilemma: avoid timing escape because of a gracious wire resistance
  – Min mode concern
  – At chip timing level, up the capture path resistance (or reduce launch path resistance) by some proportion
    • Select proportion based on min time path accuracy versus spice
    • Often achieved by 'scaling' the max mode arrivals to be later when checking hold times
  – Use biggest wire delays for clk and data in max mode
• SSTA, done properly, would remove the need for special biasing of wires to cover corner cases
  – Parameter variation (e.g., usually covers the ‘lumped’ fluctuations of metal width/thickness/spacing and dielectric) would be used to determine sensitivities against wiring artifacts (e.g., resistance) or, more likely, against the delay:

\[
\frac{\Delta S}{\Delta R} \quad \frac{\Delta S}{\Delta D}
\]
Parasitic Considerations

• Parasitic data is annotated on top of the netlist
  – Load and driver impedance are analyzed
  – timer’s delay calculator reduces to simple PI model with Cnear, Cfar, Rwire
    • algorithms create entries for driver and wire delays in path reports
  – series inductive effects must be somehow accounted for
    • eventually wave reflections for mismatched impedances may be important

• Noise effect on delay considered via coupling capacitances
  – eventually inductive coupling will be important
Special Path Control

- **False path control**
  - Stop propagation that can't logically happen (but the timer finds a path anyhow)
    - Often used to hide asynchronous paths

- **Multicycle paths**
  - Paths that require multiple clock cycles to complete should be specified

- **Multifrequency paths**
  - Unless defined as asynchronous, all defined clock pairs are assumed to be synchronous
  - GCD determines closest launch/capture edge interval $\omega$ for setup
    - Defines pseudo base frequency
    - If pseudo base cycle phase shift exists between any clock pair:
      $\omega = \min\{\phi \% \text{GCD}(T_1,T_2) , \text{GCD}(T_1,T_2) \% \phi\} \mod \phi$
    - Assumption for hold: capturing (master) edge comes as close to next cycle launching (master) edge without being later

- **Case analysis**
  - Used to analyze mutually exclusive modes of operation

- **Similar provisions in both STA and SSTA**
Skew Reduction

• Reduce skew penalty in min mode if launch and capture path have a common event (one edge) and common node closer to the sequential elements than is the master clock
  – Static timers can’t distinguish that an event on a particular node can only occur once so it normally applies a skew corresponding with the entire clock path
  – Common Path Pessimism Removal (CPPR) reduces the skew by removing that uncertainty normally attributed to the common driver members of both launch and capture paths
• Min skew relief sometimes sought when skew can be controlled for certain circuits
  – distance-based skew is attempted to reduce skews when launch and capture clock grid origination points are significantly closer than the distance used for foundation of the skew
  – OR sometimes fast SPICE analysis of clock generation circuitry
• SSTA removes over-pessimism and alleviates the need to perform such alternative analysis (correlations)
  – Clk uncertainty (u) proportion of cycle time (T) generally increases as the critical dimension decreases
More Call for Statistical Static Timing Analysis

• Clock network complexity yielded STA skew
  – Spec from clock team; avoid STA of clock network
  – The simpler the spec, the greater the guardbanding and more difficult to meet both min and max mode timing
    • Specs are now given per cluster or even per block
    • Further reductions provided by distance-based skew
    • But manufacturing physical deviations on chip and chip-to-chip present greater entities to consider, tending to larger relative skews
  – Race ratio scaling is used to assure races are won (hold margins) which further exacerbates over-pessimism

• SSTA Goal: more directly apply fab data to timing
  – Avoid allowances to cover corner cases when assessing and accounting components of skew (e.g, vdd deviation, temperature variation, etc.)
  – Reduce skew guardbanding and hold margins since spatially local circuits have high PVT correlation
SSTA

- Reduce over-pessimism
- Parametric yield
- Sensitivity Analysis
Corner Simulation Pessimism

- Clock network complexity yielded STA skew (where $x_i$ represents process parameter)
  \[ y = x_1 + x_2 \quad \text{where} \quad x_1, x_2 \sim N(0, \sigma) \]

- Independent intra Die Variation
  \[ y \sim N(0, \sqrt{2} \sigma) \]

Yaping Zhan: Correlation Aware Statistical Timing, Sept. 2004
Slack Distribution and Parametric Yield

- Design slack is the minimum of all slacks

\[ S = \min(S_1, S_2, \ldots, S_n) \]

- Parametric yield is obtained from the design slack distribution for a target clock frequency
Parametric yield curve

Yield

Clock frequency

Classical Yield Curve

“Statistical” Yield Curve

Chandu Visweswariah, IBM Thomas J. Watson Research Center
Canonical variational delay model

- Correlations are the problem
  - in a circuit with 1M nodes and 2M edges and 12 timing values per node/edge, we DO NOT want to store or manipulate a 36M x 36M covariance matrix!
  - instead, parameterize all timing quantities by the sources of variation
  - first-order canonical model:

\[
a_0 + a_1 \Delta X_1 + a_2 \Delta X_2 + \cdots + a_n \Delta X_n + a_{n+1} \Delta R_a
\]

- Constant (nominal value)
- Sensitivities
- Deviation of global sources of variation from their nominal values
- Random uncertainty (deviation from nominal value)
Sensitivity Analysis

• Define the sensitivity of output WRT a delay \( \frac{\Delta A}{\Delta D_i} \)

• Without delay variations, sensitivities w.r.t. delay arcs in the critical path will be one, and others will be zero

• Only these delay elements will be optimized

\[
\begin{align*}
\frac{\partial A}{\partial D_1} &= 0 \\
\frac{\partial A}{\partial D_2} &= 0 \\
\frac{\partial A}{\partial D_3} &= 0 \\
\frac{\partial A}{\partial D_4} &= 1 \\
\frac{\partial A}{\partial D_5} &= 0 \\
\frac{\partial A}{\partial D_6} &= 1
\end{align*}
\]

Optimize D4 and D6

Mustafa Celik, Extreme-DA
Sensitivity Analysis

- With process variations
  - Critical path is not well defined
  - Every path can be “critical” under certain probability
- Now, all the sensitivities have nonzero values and their criticalities are different

\[
\begin{align*}
\frac{\partial A}{\partial D_1} &= 0.56 & \frac{\partial A}{\partial D_2} &= 0.33 \\
\frac{\partial A}{\partial D_3} &= 0.23 & \frac{\partial A}{\partial D_4} &= 0.44 \\
\frac{\partial A}{\partial D_5} &= 0.33 & \frac{\partial A}{\partial D_6} &= 0.67
\end{align*}
\]

Optimize D1 and D6
Sensitivity Analysis

- Yield sensitivity w.r.t. every instance delay: \( \frac{\partial Y}{\partial D_i} \)

- Statistical sensitivities provide much more useful information than critical path analysis

- Calculates the sensitivities of delays, arrivals, slacks, design slack, and parametric-yield w.r.t.
  - design parameters (cell size, wire size, wire spacing)
  - system parameters (vdd, temperature)
  - process parameters

Mustafa Celik, Extreme-DA
SSTA Summary

- Motivation
  - Variability is proportionately increasing
    - As feature sizes decrease, variances increase

- Better predicts performance and yield than STA
  - e.g., P(ATdata < ATclk)
  - Correlations matter
    - 2 sources
      - Design: reconvergent paths
      - Process: systematic
  - Variation sensitivities matter
  - Robustness is an important metric
    - Higher order models (e.g., quadratic)

- As with STA, SSTA lacks accuracy compared to dynamic simulation considering simultaneous switching, parallel drivers
EXTRAS
Slacks with Clock Example

Clock C posedge @ 0 (Leading edge); negedge @ 200 (Trailing edge); Tcycle @ 400

Clock Phases (Domain)
- \( c^\wedge \) same as C; launched from C leading edge
- \( cv \) opposite of C; launched from C trailing edge

Data
- Launched from C leading edge
- Launched from C trailing edge

clock tracing
Slacks with Clock Example

Clock C posedge @ 0 negedge @ 200 Tcycle @ 400

Conditional delay propagation arc

Checks against reference

Question: Why doesn’t QG have propagation arcs to c1c?

Answer: We’re gating the clock. Don’t want to propagate data through nand N1.
Clock C posedge @ 0 negedge @ 200 Tcycle @ 400

1. Clock gate test at nand N1
2. Latch capture test at L1

Late Mode Slack = RAT – AT
RAT = AT(early reference) - su - u + adjust
su(clock gate) = 0, su(latch) = 50, u = 50

<table>
<thead>
<tr>
<th>Node</th>
<th>Late Rise AT</th>
<th>Late Fall AT</th>
<th>AT(early rise ref)</th>
<th>AT(early fall ref)</th>
<th>Late RAT rise</th>
<th>Late RAT fall</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0</td>
<td>200</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c1</td>
<td>250</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c2</td>
<td>100</td>
<td>300</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c1c</td>
<td>350</td>
<td>150</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>QG</td>
<td>350</td>
<td>350</td>
<td>100</td>
<td>100</td>
<td>450</td>
<td>450</td>
</tr>
<tr>
<td>QA</td>
<td>200</td>
<td>200</td>
<td>100</td>
<td>-150</td>
<td>400</td>
<td>150</td>
</tr>
<tr>
<td>D2</td>
<td>500</td>
<td>250</td>
<td>150</td>
<td>150</td>
<td>450</td>
<td>450</td>
</tr>
<tr>
<td>QB</td>
<td>600</td>
<td>450</td>
<td>not specified</td>
<td>not specified</td>
<td>not calculated</td>
<td>not calculated</td>
</tr>
</tbody>
</table>
### Early Mode Graph

1. Clock gate test at nand N1
2. Latch capture test at L1

**Early Mode Slack** = AT – RAT  
RAT = AT(late reference) + h + u + adjust  
h(clock gate) = 0, h(latch) = 0, u = 30

<table>
<thead>
<tr>
<th>Node</th>
<th>Early Rise AT</th>
<th>Early Fall AT</th>
<th>AT(late rise ref)</th>
<th>AT(late fall ref)</th>
<th>Early RAT rise</th>
<th>Early RAT fall</th>
<th>Early Slack rise</th>
<th>Early Slack fall</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0</td>
<td>200</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c1</td>
<td>225</td>
<td>25</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c2</td>
<td>50</td>
<td>250</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c1c</td>
<td>275</td>
<td>75</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>QG</td>
<td>275</td>
<td>275</td>
<td>250</td>
<td>250</td>
<td>280</td>
<td>280</td>
<td>-5</td>
<td>-5</td>
</tr>
<tr>
<td>QA</td>
<td>100</td>
<td>100</td>
<td>50</td>
<td>-175</td>
<td>80</td>
<td>-145</td>
<td>20</td>
<td>245</td>
</tr>
<tr>
<td>D2</td>
<td>350</td>
<td>125</td>
<td>75</td>
<td>75</td>
<td>105</td>
<td>105</td>
<td>245</td>
<td>20</td>
</tr>
<tr>
<td>QB</td>
<td>325</td>
<td>325</td>
<td>not specified</td>
<td>not specified</td>
<td>not calculated</td>
<td>not calculated</td>
<td>not calculated</td>
<td>not calculated</td>
</tr>
</tbody>
</table>

Clock C posedge @ 0 negedge @ 200 Tcycle @ 400
Importance of Establishing Required Element Setup and Hold

- Timing elements (e.g., latches, flip flops, domino, etc.) are reduced representations of real circuits
  - assumptions are made to reduce analysis complexity
    - E.g., avoid clk -> Q delay/slew modulation dependence from relative clk and data arrival times
    - if these assumptions are violated, then static timing results are invalid (e.g., slew limits)
    - required setup and hold times bound the problem space related to circuit behavior
      - relative arrival times between capturing data and clock signal pairs are tested to respect these required times

- Static timing analysis uses these requirements to calculate slacks -- How much of the budget remains from the cycle time considering logic (and wires) timing expense
Path Considerations

• Tests delineate possible end points
• Path enumeration
  – “.. but I can't find the one I'm interested in.”
    • It probably wasn't critical to the timer
    • Perform a search specifically for it
• Loops must be broken – what happens
Path Considerations (cont.)

• **Static logic setup and/or hold tests consist of:**
  – Data to Latch (setup and hold), path ‘a’
  – Clock to Latch (setup and hold), path ‘d-b’
  – Latch to Latch (setup), path ‘a-b’
  – Latch Loop (setup), loop chart, path ‘b-c-d-b’.
  – Data / Clock / Latch to Gated Clock (setup and hold), ‘clkbar-g‘ is a clock to gate path

• **Enumerate all possible paths**
  – Flip flops provide a hard timing boundary -- no transparency
  – Transparency (or flushing) generates many possible paths
    • If data arrives while latch is in flush mode, must check downstream latch timing with this arrival event as well as the current flushing latch (e.g., 2 latches in series represents 3 sources, 3 sinks, and 6 paths)
    • E.g., assuming one primary non-clock input:
      – # of paths for n flops:  n + 1
      – # of paths for n latches:  Sum of n+1 from i=0 to n

\[
\sum_{i=0}^{n} \left( n + 1 \right)
\]
Path Considerations (cont.)

The 6 paths include the following segments:
- a
- ab (flush)
- abc (flush)
- db
- dbc (flush)
- ec
Path Considerations (cont.)

- Enumerate all possible paths (cont.)
  - Loops present another level of difficulty
    - Loop delay effects are visible at downstream latches
    - Normally a re-launch is visible in the slack path report (see example below) or a [pessimistic] timing check is performed where the 'snip' was performed
  - Assuming arrival times warrant that loop elements are transparent, then path ‘abcdbc’ must be represented at latch LA. In other words, data launched from LF can arrive at LB when it is transparent and then again at LF when it is transparent. This path can pose a later acyclic arrival time at LA then the direct LF to LA path of ‘c.’ This loop delay must be taken into account at LA and other downstream latches.

- Data hold ‘loops’ can cause false hold time violations
  - Current and next cycle data is identical
Path Considerations (cont.)

• Relating to early design phase process:
  – Every node of the timing graph must have an AT and a RAT
    • recall for clusters only inputs had RATs and outputs had ATs
    • combinatorial-only cluster arcs complicate process (e.g., c2, c3)

AT process flow

RAT process flow

Late mode example (signals flow from left to right):

AT(c2.e) = max\{ [AT(c1.c)+DW(c1.c->c2.e)], [AT(c1.d)+DW(c1.d->c1.e)] \}
AT(c2.f) = max\{ [AT(c2.e)+D(c2(e->f))], [AT(a)+DW(a->c2.b)+D(c2(b->f))] \}
RAT(c3.v) = min\{ [RAT(c4.x)-DW(c3.v->c4.x)], [RAT(c4.y)-DW(c3.v->c4.y)] \}
RAT(c3.u) = min\{ [RAT(c3.v)-D(c3(u->v))], [RAT(z)-DW(c3.x->z)-D(c3(u->x))] \}
Glossary

arc – a path between pins or nodes of a timing graph that illustrates a signal can pass arrival time and slew from the input pin/node to the output (considering polarity); represents delay/slew of logic blocks or wires between pins of logic blocks

AT – arrival time; the time a signal arrives at a node

clock – signal which defines synchronous behavior; requirement that defines cycle time and required arrival times at timing elements

clock domain – see “phases”

clock gate – non-transparent timing element which imposes a half cycle path; setup tested against asserting clock edge, hold against de-asserting edge

clock tracing – operation of the static timer to identify all block pins that are clocks

counted arc – a timing arc that will not propagate data unless an enable signal is in the proper state
Glossary (cont.)

cycle accounting – adjusting the cumulative arrival time to maintain events within the cycle modulus

cycle adjusts – cycle modulus changes to the required arrival time in order to relate the correct data to the clock event that it must be checked against

domino gate – transparent timing element

data – a signal that is not a clock; a signal that changes once per cycle unless it is a dynamic signal (e.g., domino) which is affected by both edges of the clock

early mode – timing mode to check that shortest paths meet proper registration/don’t arrive too early; use earliest arriving data and compare to latest required arrival time

flip flop – edge triggered (non-transparent) timing element

hold time – timing check to assure new data didn’t change until after reference closed; sometimes refers to the margin associated with particular timing elements to assure the slack calculation takes all necessary circuit issues into consideration

latch – transparent timing element
late mode – mode to check that longest paths meet timing goals; use latest arriving data and compare to earliest required arrival time

near-domino gate – transparent timing element

phases – clock and data signal attribute which indicates the relationship of the signal compare to the master clock (signal from which its timing context is derived); sometimes referred to as signal's clock domain; phases are used in order to provide the static timer the ability to perform cycle adjusts

RAT – required arrival time; defines the acceptable boundaries of a signal's min and max arrival time and considers the reference events, circuit mechanics, and system tolerances

setup time – timing check to assure data arrives before reference closes; sometimes refers to the margin associated with particular timing elements to assure the slack calculation takes all necessary circuit issues into consideration

skew – difference in same clock event arrival time at different pins throughout the design; usually not analyzed by static timer

slack – residual time of the difference between when an event actually occurs versus when it is required to occur; a negative value means that the order of the two events was wrong.

SSTA – Statistical static timing analysis
Glossary (cont.)

static timing – exhaustive method of measuring a design's timing robustness by building a timing graph of the design, providing signal arrival times, propagating these and identifying critical paths

timing elements – logical context defining arcs and checks among three points in a timing graph; examples: latch, flip flop, clock gate

timing graph – a collection of arcs and checks which represents the timing behavior of a logic design

transparent – data propagation through a controlled arc when the controlling signal is enabled

uncertainty – safety margin included in the slack calculation to account for clock skew and other variables affecting clock edge events